

UNIVERSITÄT PASSAU

Fakultät für Mathematik und Informatik

Dissertation

**Visualisierung
biochemischer Reaktionsnetze**

Falk Schreiber

Passau, im Juni 2001

Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften
an der Fakultät für Mathematik und Informatik der Universität Passau.

1. Gutachter: Prof. Dr. Franz J. Brandenburg, Universität Passau
2. Gutachter: Prof. Dr. Ralf Hofestädt, Universität Bielefeld

Danksagungen

BioPath ist entstanden durch die Kooperation von Wissenschaftlern verschiedener Universitäten, insbesondere von Michael Forster, Prof. Johann Gasteiger, Dr. Bernhard Gruber, Dr. Michael Himsolt, Dr. Wolf D. Ihlenfeldt, Carl-Christian Kanne, Prof. Guido Moerkotte, Andreas Pick, Marcus Raitner und Dr. Dietrich Trümbach. Ihnen gilt mein Dank für die hervorragende Zusammenarbeit.

Für die anregenden Diskussionen über biochemische Fragestellungen danke ich besonders Prof. Toni Kazic, Astrid Liedl, Ursula Löffler und Dr. Gerhard Michal. Dr. Konstantin Skodinis fand immer Zeit, Probleme der theoretischen Informatik mit mir zu diskutieren.

Viele Anregungen zur Verbesserung der Arbeit stammen von meinen Kollegen am Lehrstuhl und von Uta Maiwald. Vielen Dank für eure Mühe!

Ein Dank gilt meinen Freunden, die während der Arbeit an dieser Dissertation meinen Mangel an Zeit (und manchmal auch Nerven) verständnisvoll hinnahmen.

Prof. Franz J. Brandenburg danke ich für die Unterstützung und Freiheit, die er mir an seinem Lehrstuhl bot, sowie für die Gelegenheit, mich mit diesem spannenden Thema zu beschäftigen. Es hat Spass gemacht!

Ein großer Dank gilt meiner Familie und besonders meinen Eltern, Almut und Ulrich Schreiber, die mich geführt, inspiriert, ermutigt und unterstützt haben. Ihnen ist diese Arbeit gewidmet.

Inhaltsübersicht

Ausführliches Inhaltsverzeichnis	iii
1 Einführung	1
I Analyse und Modellierung	11
2 Grundlagen	13
3 Darstellungsanforderungen	37
4 Traditionelle Realisierungen	59
5 Modellierung	83
II Das VGL-Zeichenverfahren und seine Anwendung in <i>BioPath</i>	97
6 Ebenenweises Zeichnen gerichteter Graphen	99
7 Erweiterungen des Zeichenverfahrens	179
8 Zeichnungen biochemischer Reaktionsnetze	207
9 Zusammenfassung und Ausblick	237
Verzeichnisse	240

Inhaltsverzeichnis

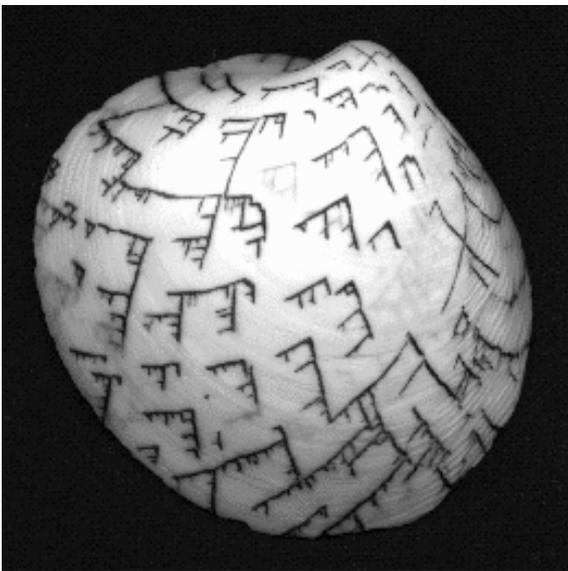
1	Einführung	1
1.1	Informatik und Biologie	2
1.1.1	Konzepte der Biologie in der Informatik	2
1.1.2	Konzepte der Informatik in der Biologie	2
1.2	Biochemische Reaktionsnetze	3
1.2.1	Gen, Enzym, Reaktion	3
1.2.2	Visualisierung biochemischer Reaktionsnetze	6
1.3	Einordnung der Arbeit	7
1.3.1	Anwendungen in der Biologie	7
1.3.2	Anwendungen in der Informatik	7
1.3.3	Zur Entstehung der Arbeit	8
1.4	Struktur der Arbeit	9
1.4.1	Analyse und Modellierung	9
1.4.2	Zeichenverfahren und dessen Anwendung	9
I	Analyse und Modellierung	11
2	Grundlagen	13
2.1	Grundlagen der Biochemie	14
2.1.1	Biochemische Reaktionen	14
2.1.2	Biochemische Reaktionswege und Reaktionsnetze	18
2.1.3	Hierarchien	22
2.2	Grundlagen der Informatik	24
2.2.1	Graphentheorie	24
2.2.2	Visualisierung	26
2.2.3	Zeichnen von Graphen als Visualisierungstechnik	30
2.2.4	Komplexitätstheorie und Experimente	35
3	Darstellungsanforderungen	37
3.1	Einführung	38
3.1.1	Ziele	38
3.1.2	Grundlage der Anforderungen	39
3.2	Anforderung I: Darstellung von Reaktionen	40
3.2.1	Substanzen, Cosubstanzen und Enzyme	40
3.2.2	Reaktionen	43

3.3	Anforderung II: Darstellung von Reaktionswegen und Reaktionsnetzen	46
3.3.1	Allgemeine Reaktionswege und Reaktionsnetze	46
3.3.2	Geschlossene und offene Zyklen	46
3.4	Anforderung III: Kontexterhaltende Navigation und Sichten	50
3.4.1	Navigation durch Reaktionsnetze	50
3.4.2	Sichten auf Reaktionsnetze	52
3.4.3	Kontexterhaltende Darstellungen bei Navigation und Sichten	52
3.4.4	Hierarchisierung	56
3.5	Zusammenfassung der Anforderungen	58
4	Traditionelle Realisierungen	59
4.1	Umsetzung der Anforderungen durch statische Visualisierungen	60
4.1.1	Charakterisierung statischer Visualisierungen	60
4.1.2	Vorkommen statischer Visualisierungen	60
4.1.3	Vorteile statischer Visualisierungen	65
4.1.4	Nachteile statischer Visualisierungen	67
4.2	Umsetzung der Anforderungen durch dynamische Visualisierungen	68
4.2.1	Charakterisierung dynamischer Visualisierungen	68
4.2.2	Bekanntes Verfahren zum Zeichnen von Graphen	68
4.2.3	Vorkommen dynamischer Visualisierungen	71
4.2.4	Vorteile dynamischer Visualisierungen	78
4.2.5	Nachteile dynamischer Visualisierungen	78
4.3	Fazit	79
5	Modellierung	83
5.1	Einführung	84
5.2	Reaktionsgraphen	84
5.3	Hierarchische Reaktionsgraphen	86
5.3.1	Hierarchische Graphen	86
5.3.2	Hierarchische Reaktionsgraphen	89
5.3.3	Ausschnitte hierarchischer Reaktionsgraphen	89
5.3.4	Vergrößerung und Verfeinerung	91
5.4	Hintergründe zur Modellierung	91
5.4.1	Modell für Visualisierungen biochemischer Reaktionsnetze	91
5.4.2	Verwendung bipartiter Graphen	94
5.4.3	Realisierung in BioPath	95
5.4.4	Weitere Modelle für biochemische Reaktionsnetze	96
II	Das VGL-Zeichenverfahren und seine Anwendung in BioPath	97
6	Ebenenweises Zeichnen gerichteter Graphen	99
6.1	Überblick	100
6.1.1	Einführung	100
6.1.2	Beliebige Knotengrößen	101

6.1.3	Lagebeziehungen	108
6.1.4	Zeichenkriterien	110
6.1.5	Grundidee des Zeichenverfahrens	111
6.2	Kantenorientierung	117
6.2.1	Überblick	117
6.2.2	Einführung	118
6.2.3	Theorie	120
6.2.4	Algorithmen	123
6.2.5	Eigenschaften der Heuristik	129
6.2.6	Weitere Verfahren zur Kantenorientierung	134
6.3	y-Koordinaten und Ebenenpartitionierung	136
6.3.1	Überblick	136
6.3.2	Einführung	137
6.3.3	Algorithmen	138
6.3.4	Weitere Verfahren zur Ebenenpartitionierung	154
6.4	Kreuzungsreduzierung	155
6.4.1	Überblick	155
6.4.2	Einführung	156
6.4.3	Theorie	160
6.4.4	Algorithmus	162
6.4.5	Weitere Verfahren zur Kreuzungsreduzierung	166
6.5	x-Koordinaten	176
6.5.1	Überblick	176
6.5.2	Berechnung der x-Koordinaten	176
6.6	Gesamtalgorithmus	177
7	Erweiterungen des Zeichenverfahrens	179
7.1	Geometrische Aspekte	180
7.1.1	Knotenzentrum	180
7.1.2	Individuelle Abstände zwischen Knoten	180
7.1.3	Kantenhöhe und Kantenbreite	181
7.1.4	Kantenrouting	182
7.1.5	Ports	184
7.2	Allgemeine Graphen	188
7.2.1	Reflexive Kanten	188
7.2.2	Parallele und antiparallele Kanten	188
7.2.3	Nicht zusammenhängende Graphen	190
7.3	Weitere Lagebeziehungen	190
7.3.1	Gewichtete Kreuzungszahl	190
7.3.2	hor-Lagebeziehungen	191
7.3.3	ver-Lagebeziehungen	192
7.3.4	Globale l-r-Lagebeziehungen	192
7.3.5	Berücksichtigung weiterer Ebeneneinteilungen	192
7.4	Benennungen von Knoten und Kanten	194
7.4.1	Einführung	194

7.4.2	Benennungen im VGL-Verfahren	196
7.5	Lokale Zeichenverfahren	196
7.5.1	Globale Berücksichtigung lokaler Zeichenverfahren	196
7.5.2	Verbesserung der Zeichnungen durch Ports	198
7.5.3	Weitere Verfahren für lokale Zeichnungen	198
7.6	Folgen von Zeichnungen	201
7.6.1	Einführung	201
7.6.2	Inkrementelle Zeichenverfahren	201
7.6.3	Stabilität in Zeichenverfahren	202
8	Zeichnungen biochemischer Reaktionsnetze	207
8.1	Zeichenverfahren für biochemische Reaktionsnetze	208
8.1.1	Reaktionen	208
8.1.2	Reaktionsnetze	209
8.1.3	Kontexterhaltende Navigation und Sichten	223
8.2	Anwendungen	225
8.2.1	Erkundung des Stoffwechsels	225
8.2.2	Vergleich von Reaktionswegen	226
8.2.3	Verfolgung von Atomen	226
8.2.4	Quantitative Darstellungen	229
8.3	Regulation und deren Darstellung	229
8.3.1	Einführung	229
8.3.2	Darstellung von Regulation	232
9	Zusammenfassung und Ausblick	237
9.1	Zusammenfassung	238
9.1.1	Inhalt der Arbeit	238
9.1.2	Ergebnisse der Arbeit	239
9.2	Ausblick	240
	Literaturverzeichnis	241
	Symbolverzeichnis	263
	Abbildungsverzeichnis	267
	Definitionsverzeichnis	270
	Index	272

1 Einführung



1.1 Informatik und Biologie	2
1.2 Biochemische Reaktionsnetze	3
1.3 Einordnung der Arbeit	7
1.4 Struktur der Arbeit	9

Die Visualisierung ausgewählter biochemischer Information mittels Verfahren aus der Informatik steht im Mittelpunkt dieser Arbeit. Das Zusammenwirken von Biologie und Informatik wird auf ungewöhnliche Weise durch diese Muschel repräsentiert, deren Schalenzeichnung Strukturen aufweist, die einer Visualisierung spezieller Graphen ähneln (aus [BJM⁺99]).

1.1 Informatik und Biologie

Die Bedeutung von Informatik und Biologie für den wissenschaftlichen Fortschritt ist allgemein anerkannt. Der Politikwissenschaftler Fukuyama charakterisiert gar das 21. Jahrhundert als das Jahrhundert der Biologie [Fuk99]. Seit einiger Zeit gibt es eine junge, sich rasch entwickelnde interdisziplinäre Wissenschaft, in der Kenntnisse und Methoden der Biologie¹, Chemie und Informatik zusammengeführt werden: die *Bioinformatik*. Ihr Ziel ist, biochemische Prozesse und Informationen mit Mitteln der Informatik zu erfassen, aufzubereiten und für Anwendungen nutzbar zu machen. Dieses Zusammengehen der Wissenschaften ist erst seit kurzem ins Bewusstsein der Öffentlichkeit gerückt. Biologie und Informatik beeinflussen sich jedoch schon seit langer Zeit gegenseitig.

1.1.1 Konzepte der Biologie in der Informatik

Biologische Konzepte werden schon lange in Form spezieller Rechenverfahren in der Informatik verwendet. Ein Anfang waren künstliche *neuronale Netze* (siehe beispielsweise [Bra97b, Kin94]), deren theoretische Grundlagen in den 40er Jahren durch McCulloch und Pitts [MP43] gelegt wurden. Auch Konzepte wie *zelluläre Automaten* [Cod68], *genetische Algorithmen* [Hei94, SHF94, Thr94] und *künstliches Leben* [Lev93, Thr94] werden schon länger in der Informatik untersucht und zur Lösung von Problemen verwendet. Bei diesen Arbeiten werden biologische Verfahren in Rechnern nachgeahmt, um Aufgaben zu lösen. Die Verfahren haben herkömmliche Lösungsmethoden jedoch nicht in breiter Front verdrängt, sondern ergänzen diese und werden für Spezialanwendungen verwendet, da sowohl die Formulierung der Probleme mittels neuronaler Netze oder genetischer Algorithmen aufwendig als auch die zur Problemlösung benötigte Rechenzeit oft zu hoch ist.

Ein anderer Ansatz ist die Verwendung biochemischer Objekte zur Problemlösung. Adleman [Adl94] zeigte 1994, dass sich biologische Moleküle auch direkt zur Lösung mathematischer Probleme verwenden lassen. Grundlage ist DNA², die mittels spaltenden und verschmelzenden Enzymen manipuliert werden kann. Damit existieren in der Natur Mechanismen, mit denen sich Rechner, die so genannten *DNA-Computer*, konstruieren lassen [Adl96, Lip95, PRS98, RL00, WR96]. Dabei sind die Informationsdichte der biochemischen Moleküle und die parallele Informationsverarbeitung die Stärken dieses Ansatzes, der in den erwähnten Arbeiten auch zu Überlegungen über molekulare Turing-Maschinen geführt hat. Es ist allerdings unwahrscheinlich, dass in naher Zukunft solche universellen DNA-Computer gebaut werden.³

1.1.2 Konzepte der Informatik in der Biologie

Konzepte und Verfahren der Informatik finden in den letzten Jahren immer stärkere Anwendung in der Biologie. Sie helfen dort beispielsweise bei der Erkennung und Aufklärung von Gensequenzen und Vererbungsbeziehungen, der Vorhersage der Struktur von Proteinen oder der Schaffung

¹Biochemie ist eine Teildisziplin der (allgemeinen) Biologie.

²Desoxyribonucleinsäure, ein langes, fadenförmiges Molekül aus Desoxyribonucleotiden. Diese bestehen jeweils aus Base, Zucker und Phosphatgruppe, wobei die Basen die genetische Information tragen, während Zucker und Phosphatgruppe die Struktur bestimmen.

³Hier sind technische Begrenzungen ausschlaggebend, die mit der räumlichen Struktur der DNA zusammenhängen. Die DNA bildet ab gewissen Längen Sekundärstrukturen, was ihre Manipulation durch Enzyme verhindert. Zudem dauert ein Berechnungsschritt zur Zeit noch mehrere Minuten, das gesamte Verfahren mit der Extraktion der die Problemlösung kodierenden Moleküle dauerte bei Adleman [Adl96] etwa eine Woche.

neuer Medikamente [Hof99, Len96, Len97, RCG97, SK96]. Dabei ist es nicht immer möglich, vorhandene Verfahren auf biologische Probleme anzuwenden. Die Zelle ist ein komplexes, schwer zu modellierendes System. Lengauer schreibt in [Len96]:

„Die Evolution verwendet enorme Ressourcen an Zeit und Material, sie erhebt nicht den Anspruch, zu verstehen, was sie tut. Das Ergebnis sind von der Evolution hervorgebrachte Objekte (z. B. Proteine⁴) und Prozesse (z. B. metabolische Pfade⁵), die keinen leicht und präzise klassifizierbaren Kriterien genügen. (...) Dies alles führt dazu, daß wir in der Regel nicht erwarten können, auf schnellem Wege vorgefertigte Modelle und Algorithmen aus der Informatik in der Molekularen Bioinformatik einsetzen zu können.“

Entsprechend müssen Problemstellungen aus der Sicht der Informatik analysiert und abstrahiert und Verfahren zur Problemlösung entwickelt bzw. angepasst werden. Methoden der Informatik werden zur Lösung einer Reihe von Problemen angewandt. Beispiele sind *Sequenzvergleich* [Len96, SK96] für Analyse und Vergleich von DNA- und Aminosäuresequenzen, *Proteinfaltung* [BT99, Len96, Ric91] zur Bestimmung der dreidimensionalen Struktur von Proteinen sowie *Analyse und Modellierung von Reaktionsnetzen* [HM95, HT98, Voi00, Wtd98], aber auch die *Visualisierung biochemischer Information*. Zu diesen Visualisierungen zählt auch die Darstellung biochemischer Reaktionsnetze mittels Verfahren zum Zeichnen von Graphen.

1.2 Biochemische Reaktionsnetze

1.2.1 Gen, Enzym, Reaktion

Das Leben basiert auf biochemischen Reaktionen. Diese liefern Energie für Lebensprozesse, bauen komplexe Moleküle aus einfachen auf und ermöglichen Vererbung und Vermehrung. In jeder Zelle findet eine Vielzahl verschiedener Reaktionen statt. Dabei wird das Produkt einer Reaktion meist als Ausgangsstoff anderer Reaktionen verwendet, wodurch eine Substanz in unterschiedliche Substanzen umgewandelt werden kann. Eine Aneinanderreihung von Reaktionen wird als Reaktions- oder Stoffwechselweg bezeichnet. Biochemische Reaktionsnetze sind beliebige Ausschnitte aus der Gesamtheit aller Reaktionswege.

Biochemische Reaktionen werden indirekt durch Gene beschrieben. Als Gen wird ein DNA-Abschnitt bezeichnet, der für die Synthese eines funktionsfähigen biologischen Produkts erforderlich ist. Oft handelt es sich bei den Produkten um Proteine, das sind überwiegend aus Aminosäuren aufgebaute chemische Verbindungen. Viele Proteine wiederum sind Enzyme, also Moleküle, die biochemische Reaktionen in Zellen erst ermöglichen. Die Beziehung zwischen Genen und Enzymen wird oft durch folgende vereinfachende Feststellung ausgedrückt: *Gene kodieren Enzyme, Enzyme katalysieren Reaktionen*.

Die Entschlüsselung des menschlichen Genoms, der Gesamtheit aller Gene des Menschen, zählt zu den herausragenden wissenschaftlichen Leistungen der letzten Jahrzehnte.⁶ Die Entschlüsselung der Gene ist aber nur der erste Schritt zum Verständnis der biochemischen Prozesse in Zellen. Nach

⁴Proteine sind überwiegend aus Aminosäuren aufgebaute chemische Verbindungen (Anmerkung durch den Autor).

⁵Metabolische Pfade sind Wege zur Umwandlung einer Substanz in eine andere (Anmerkung durch den Autor).

⁶In Zeitschriften wie *Nature* [The01] und *Science* [Ven01] wird zwar die Entschlüsselung des menschlichen Genoms verkündet, dabei handelt es sich jedoch nur um einen groben Überblick über die menschliche DNA-Sequenz und bedeutet nicht die Kenntnis der kompletten DNA-Sequenz eines beliebigen Menschen. So unterscheiden sich die DNA-

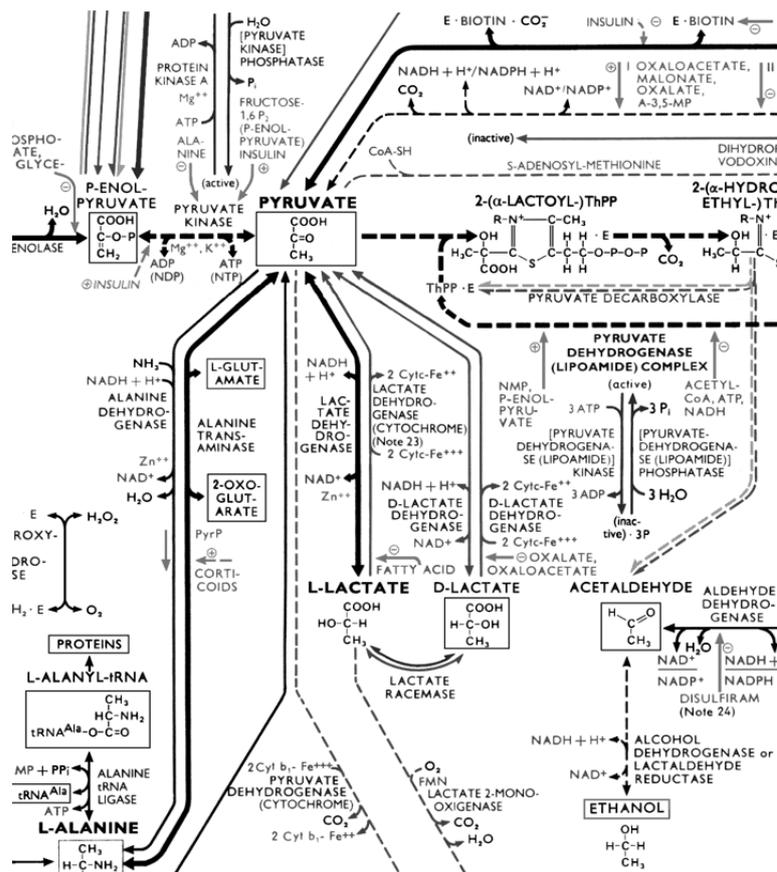


Abbildung 1.1: Biochemisches Reaktionsnetz. Ein kleiner Ausschnitt aus dem Poster *Biochemical Pathways* [Mic93]. Namen bezeichnen Substanzen und Enzyme, die Pfeile zwischen den Substanzen symbolisieren überwiegend deren Umsetzung durch Reaktionen. Auffällig ist die hohe Komplexität der Darstellung.

der Entschlüsselung des menschliche Genoms steht nun die Entschlüsselung des Proteoms, der Gesamtheit der durch die Gene kodierten Proteine sowie der durch diese katalysierten biochemischen Reaktionswege im Zentrum biomolekularer Untersuchungen.

In den letzten Jahren wurden dabei bereits große Fortschritte erzielt. Abbildung 1.1 zeigt einen kleinen Ausschnitt aus dem Poster *Biochemical Pathways* [Mic93], auf dem wesentliche Reaktionen in Zellen abgebildet sind. Die auf dem Poster dargestellten etwa 1500 Reaktionen und ähnlich viele durch die Reaktionen umgesetzte Substanzen stellen einen großen Teil des Wissens über Stoffwechselwege zu Anfang der 90er Jahre dar. Heute sind schätzungsweise zehntausend Reaktionen und Substanzen bekannt.⁷ Doch auch dies ist nur ein kleiner Teil aller in Zellen ablaufenden Re-

Sequenzen zweier Menschen, dies wird z. B. in der forensischen Wissenschaft in Form des so genannten *genetischen Fingerabdrucks* genutzt. Zudem gibt es in den parallel von Venter et al. [Ven01] und dem Human Genom Projekt [The01] bestimmten DNA-Sequenzen große Unterschiede.

⁷Einen aktuellen Überblick bieten Datenbanken mit biochemischen Inhalten, siehe beispielsweise die Datenbanksammlung unter [SRS00].

aktionen. Allein das menschliche Genom wird von Molekularbiologen⁸ auf 30000 bis 120000 Gene geschätzt.⁹ Viele Gene kodieren ein oder mehrere Enzyme und jedes Enzym katalysiert eine oder mehrere Reaktionen.¹⁰ Konservative Schätzungen gehen davon aus, dass typische Zellen höherer Organismen einige Zehntausend verschiedene Proteine synthetisieren [ABL⁺97]. Bahnsen [Bah00] verweist auf Schätzungen, nach denen durch die Information im menschlichen Genom sogar bis zu 20 Millionen verschiedene Proteine produziert werden. Die bisher aufgeklärten Reaktionen stellen also nur einen winzigen Bruchteil aller biochemischen Reaktionen dar. Dieses Wissen wächst jedoch täglich.

Biochemische Reaktionsnetze können sehr groß und komplex sein. Ist ein Anwender an bestimmten Aspekten der Netze interessiert, so helfen ihm dabei geeignete Visualisierungen. Oft sind Visualisierungen der Reaktionsnetze sogar von grundlegender Bedeutung für das Verständnis der komplexen biochemischen Prozesse in Zellen. Die Bedeutung grafischer Darstellungen für das Verständnis biochemischer Reaktionsnetze lässt sich mit einem Vergleich unterstreichen: Man versuche, sich im U-Bahn-Netz einer Großstadt anhand einer textuellen Beschreibung aller Bahnlinien zu orientieren und stelle dem eine grafische Darstellung in Form eines Diagramms bzw. Plans des U-Bahn-Netzes gegenüber.¹¹

Die bisherigen Arten und Verfahren der Darstellung biochemischer Reaktionsnetze sind jedoch in vielen Punkten unzulänglich: Visualisierungen in Form von Postern oder Büchern können nicht die täglich neuen Erkenntnisse berücksichtigen und ermöglichen nur die vom Ersteller vorgegebene Sicht auf Reaktionsnetze oder Ausschnitte der Netze. Visualisierungen auf der Grundlage von Algorithmen zum Zeichnen von Graphen haben sich bisher nicht durchgesetzt, da die damit erzeugten Darstellungen nicht den Anforderungen der Biochemiker genügen. Hier ist ein Qualitätssprung in der grafischen Aufbereitung der Daten über biochemische Reaktionsnetze gefordert. Es sollen Darstellungen erzeugt werden, die den Konventionen der Biochemie genügen und die ein einfaches Erkunden dieser Netze ermöglichen.

Diese Arbeit beschäftigt sich mit einem entsprechenden Verfahren zur automatischen Visualisierung biochemischer Reaktionsnetze. Reaktionsnetze lassen sich als Graphen modellieren. Dabei entsprechen beispielsweise Substanzen den Knoten, die Umsetzung der Substanzen den Kanten eines Graphen. An die Visualisierung dieser Netze werden spezielle Anforderungen gestellt. Diese werden in der Arbeit analysiert, es werden Regeln für verständliche Darstellungen formuliert und ein Modell vorgestellt, das die herausgearbeiteten Anforderungen unterstützt. Grundlage der Darstellungskonventionen sind dabei die Auswertung traditioneller Zeichnungen und empirischer Untersuchungen mit Anwendern. Darauf aufbauend wird ein Verfahren zur interaktiven automatischen Visualisierung biochemischer Reaktionsnetze entwickelt. Das Verfahren ist dabei so flexibel gestaltet, dass es sich auch zur Darstellung anderer netzartiger oder hierarchischer Strukturen mit Objekten verschiedener Größe eignet. Im Ergebnis bietet diese Arbeit Möglichkeiten, die ständig wachsende Da-

⁸Im Text erfolgt die Bezeichnung weiblicher und männlicher Personen aus Gründen der Lesbarkeit jeweils in der maskulinen Form. Mit den verwendeten Personenbezeichnungen sind stets beide Geschlechter gemeint.

⁹Vgl. [Phi00] und die dort zitierten Artikel, in aktuellen Nachrichten wird nur noch von 30000 bis 45000 Genen ausgegangen.

¹⁰Einige Enzyme sind in der Lage, nicht nur einzelne Reaktionen, sondern eine Menge gleichartiger Reaktionen zu katalysieren. Dadurch können ähnliche Substanzen erzeugt werden; Beispiele sind die Synthese von Fettsäuren oder von Polysacchariden. Solche Mengen gleichartiger Reaktionen sind in den Darstellungen oft nur durch eine Reaktion repräsentiert.

¹¹Hier sei auf das Buch von Garland [Gar94] verwiesen, in dem die Entwicklung der Diagramme des Londoner U-Bahn-Netzes beschrieben wird.

tenmenge über biochemische Reaktionen auf intuitive, für Biochemiker leicht verständliche, Weise zugänglich zu machen.

1.2.2 Visualisierung biochemischer Reaktionsnetze

Zeichnungen biochemischer Reaktionsnetze werden bisher überwiegend manuell erstellt. Ein Beispiel ist das Poster *Biochemical Pathways* (Abb. 1.1), das von Michal ohne Hilfe eines Computers gezeichnet wurde [Mic00]. Doch auch Grafikprogramme, beispielsweise CorelDraw [DHS00], können die manuelle Erstellung der Zeichnungen nur unterstützen, diese jedoch nicht entsprechend den Darstellungskonventionen der Biochemie automatisch erzeugen. Eine automatische Visualisierung beliebiger Ausschnitte aus biochemischen Reaktionsnetzen wird jedoch immer notwendiger: Täglich fallen neue Daten über biochemische Reaktionen an. Zum Verständnis der Prozesse in Organismen müssen diese im Kontext des gesamten Stoffwechsels dargestellt und untersucht werden. Da sich Stoffwechselforgänge am besten grafisch darstellen lassen, sind für deren Veranschaulichung in Lehre und Forschung aussagekräftige Visualisierungen unumgänglich. Michal unterstreicht dies, wenn er in [Mic98b] schreibt:

„Gut gestaltete Grafiken sind am anschaulichsten und bieten gleichzeitig einen Überblick über verschiedene Aspekte.“

Bei diesen Visualisierungen müssen gewisse Anforderungen erfüllt werden, um einen Wiedererkennungseffekt für Reaktionen und Stoffwechselwege zu erreichen. Dazu gehört, dass Reaktionen entsprechend ihren Abhängigkeiten angeordnet und bestimmte Komponenten der Reaktionen, z. B. Enzyme und Cosubstanzen, neben den Reaktionspfeil platziert werden.

Biochemische Reaktionsnetze können sehr groß werden und sind dann mit herkömmlichen Ausgabegeräten nicht mehr sinnvoll darstellbar, da die Darstellungsfläche begrenzt ist und bei Zeichnungen mit Tausenden oder gar Millionen Objekten auch Techniken wie Scrollen¹², Zoomen¹³ oder Fisheye-Views [LR96, MS91b, SFM99] unzureichend sind. Um dennoch mit Visualisierungen von Reaktionsnetzen arbeiten zu können, müssen mächtigere Mechanismen zur Navigation angeboten werden. Vorteilhaft ist dabei, dass sich biochemische Reaktionsnetze hierarchisieren lassen: Reaktionen können zu Stoffwechselwegen und Stoffwechselwege zu Stoffwechselklassen zusammengefasst werden. Interaktive Systeme müssen durch entsprechende Visualisierungen ein Navigieren durch diese Hierarchie ermöglichen. Dabei ist oft die Darstellung von Reaktionen im Kontext der sie umgebenden Stoffwechselwege gewünscht. Bei diesen Darstellungen spielen kontexterhaltende Visualisierungen eine zentrale Rolle. Wird eine Zeichnung durch eine Aktion des Benutzers verändert, so sind bei kontexterhaltenden Visualisierungen in der neuen Zeichnung nur jene Teile anders dargestellt, die durch die Aktion verändert wurden. Die relativen Lagebeziehungen der aus der alten Zeichnung übernommenen Teile bleiben jedoch unverändert. Eine solche Nutzeraktion kann beispielsweise die Wahl einer detaillierteren Sicht auf einen Ausschnitt des aktuellen Reaktionsnetzes sein.

Kontexterhaltende Darstellungen zielen auf zwei Anwendungen: Auf biochemische Reaktionsnetze müssen verschiedene *Sichten* angeboten werden, um jeweils verschiedene Aspekte der Netze

¹²Scrollen ist die Veränderung des dargestellten Ausschnitts einer Abbildung durch Verschieben des aktuell sichtbaren Bereichs bei gleichbleibender Auflösung.

¹³Zoomen ist die Veränderung des dargestellten Ausschnitts einer Abbildung durch Veränderung der Auflösung bei gleichbleibender Darstellungsfläche und gleichem Zentrum des Ausschnitts.

unter Beibehaltung deren grundlegender Struktur betrachten zu können. Ein Beispiel sind Darstellungen eines biochemischen Reaktionsnetzes mit bzw. ohne Enzyme. Zudem muss der dargestellte *Ausschnitt* veränderbar sein, um ein Herumwandern in den Netzen zu ermöglichen.

1.3 Einordnung der Arbeit

1.3.1 Anwendungen in der Biologie

In der vorliegenden Arbeit wird dargestellt, wie biochemische Reaktionsnetze modelliert und visualisiert werden können. Visualisierungen biochemischer Reaktionsnetze haben eine Reihe von Anwendungen: Sie unterstützen den Vergleich von Stoffwechselwegen, um Unterschiede zwischen Organismen zu untersuchen, sie werden in der Lehre zur Vermittlung der Prozesse in Zellen verwendet und sie helfen bei der Simulation von Reaktionsnetzen zur Darstellung der Ergebnisse.

So kann beispielsweise der Vergleich von Stoffwechselwegen in verschiedenen Organismen der Entwicklung neuer Medikamente dienen. Man denke dabei an einen Stoffwechselweg, der im Menschen eine untergeordnete Rolle spielt, in Bakterien dagegen von lebensnotwendiger Bedeutung ist. Falls es gelingt, diesen so zu unterbrechen, dass im menschlichen Stoffwechselweg kein Schaden entsteht, so ist mit der unterbrechenden Substanz ein neues Antibiotikum gefunden. Hier helfen geeignete Visualisierungen bei der Suche nach Unterbrechungsstellen der biochemischen Reaktionsnetze.

Darstellungen von Reaktionsnetzen können auch für die Visualisierung der Ergebnisse von Untersuchungen mittels so genannter *DNA-Chips* bzw. *DNA-Mikroarrays* [LW00,Ram98] verwendet werden. Mit DNA-Chips lässt sich die Aktivität von Genen und damit die Menge von Enzymen in der Zelle bestimmen.¹⁴ Da Gene Enzyme kodieren, Enzyme wiederum Reaktionen katalysieren, entspricht das Untersuchungsergebnis indirekt dem Umsatz von Substanzen in Zellen. Visualisierungen biochemischer Reaktionsnetze ermöglichen es, diesen Substanzumsatz auch grafisch darzustellen. Dazu werden Reaktionen mit verschiedenen dicken Pfeilen dargestellt.

1.3.2 Anwendungen in der Informatik

Die Darstellung komplexer Netzwerke ist keine neue Herausforderung in der Informatik, sondern ist aus den unterschiedlichsten Anwendungsgebieten bekannt. Als Beispiele seien hier die Visualisierung von Programmen (siehe beispielsweise [GKNV93, NK94, Rei95, San99]), von PERT¹⁵- und ER¹⁶-Diagrammen [BTT83, DT88], sozialen Netzwerken [BKW99], Zugverbindungen [BW98] oder von UML¹⁷-Diagrammen [Eic00, Mar98, See97] genannt.

Biochemische Reaktionsnetze sind durch mehrere Aspekte charakterisiert: Die hohe Komplexität

¹⁴Mittels DNA-Chips können mehrere tausend Gene gleichzeitig untersucht werden, in gewissem Sinn lässt sich damit ein Abbild des aktuellen Stoffwechsels erstellen. Dazu sind auf einer daumennagelgroßen Fläche Teile mehrerer tausend bekannter Gene in Form von kurzen DNA-Stücken aufgetragen. Diese binden mit komplementären Stücken, die sich in der zu untersuchenden Lösung befinden. Mittels spezieller Techniken wird diese Bindung sichtbar gemacht, wobei auch quantitative Aussagen zur Konzentration dieser Stränge getroffen werden.

¹⁵Program Evaluation and Review Technique, siehe [MP70].

¹⁶Entity-Relationship, siehe [Che76].

¹⁷Unified Modeling Language, siehe [BRJ98, FS98].

und Dichte der Netze¹⁸ erfordert verschiedene Sichten auf die Daten, die Größe mit vielen Tausenden und potenziell Millionen Objekten erfordert eine Hierarchisierung und Verfahren zur Navigation. Das Besondere sind jedoch Konventionen der Biochemie zur Darstellung solcher Reaktionsnetze. Diese erfordern Zeichenverfahren, die zusätzliche Lagebeziehungen berücksichtigen. Trotz der großen Unterschiede in der Darstellung zwischen Reaktionsnetzen und den oben erwähnten Netzen lassen sich hier entwickelte Konzepte, z. B. Lagebeziehungen und kontexterhaltende Navigation, auch auf die Darstellung anderer Netze und hierarchischer Strukturen übertragen. Die in dieser Arbeit entwickelten Verfahren sind dafür so allgemein wie möglich gehalten.

Es werden ebenenweise Zeichenverfahren für gerichtete Graphen mit beliebigen Knotengrößen theoretisch untersucht und Heuristiken zur Lösung der Zeichenprobleme vorgestellt. Diese Verfahren sollen zugleich die relativen Lagebeziehungen der Knoten betreffende Wünsche berücksichtigen und kontexterhaltende Visualisierungen erzeugen. Dies sind neue Anforderungen beim Zeichnen von Graphen. Damit leistet die Arbeit auch einen Beitrag zur Weiterentwicklung dieses Gebiets der Informatik.

1.3.3 Zur Entstehung der Arbeit

Teile dieser Arbeit sind im Rahmen des interdisziplinären Projekts *BioPath* [BGHS98, KST99a, TG00] entstanden, das in einem Teilprojekt am Lehrstuhl für Theoretische Informatik von Prof. Brandenburg in Passau durchgeführt wurde. Ziel des Projekts ist eine elektronische Umsetzung des weltbekannten Posters¹⁹ *Biochemical Pathways* [Mic93], von dem Abbildung 1.1 einen kleinen Ausschnitt zeigt. Im Jahr 1999 erschien der ebenfalls von Michal herausgegebene Atlas *Biochemical Pathways* [Mic99], der ebenfalls Grundlage des Projekts wurde.

Das Projekt wurde im Rahmen des Programms *Multimediales Buch* vom Bundesministerium für Bildung, Forschung und Technologie (BMBF) gefördert und vereinte Forschungsgruppen der Universitäten Mannheim, Erlangen und Passau sowie den Spektrum Akademischer Verlag Heidelberg. In Mannheim wurde ein objektorientiertes Datenbankschema zur Speicherung der komplexen biochemischen Daten entwickelt, in Erlangen wurden die Substanzen, Enzyme und Reaktionen auf atomarer Ebene erfasst und in Passau wurde die Visualisierungskomponente des Projekts entwickelt. Als industrieller Partner begleitete der Spektrum Akademischer Verlag das Projekt, der auch Poster und Atlas *Biochemical Pathways* [Mic93, Mic99] zur Verfügung stellte. Mit Herrn Michal sowie Wissenschaftlern des Max-Planck-Instituts für Biochemie in Martinsried bei München und weiteren Biologen und Biochemikern verschiedener Universitäten standen dem Projekt stets hochkarätige Ansprechpartner für das Gebiet *Biochemische Reaktionsnetze* zur Verfügung. Ergebnisse des *BioPath* Projekts finden sich unter [Bio01] und werden auch von der Firma LION Bioscience [LIO01] verwendet.

Viele der in dieser Arbeit verwendeten Abbildungen wurden mit *BioPath* erzeugt. *BioPath* ist ein aktives Projekt, insbesondere der Datenbestand wird ständig erweitert und aktualisiert. Die Abbildungen spiegeln den aktuellen Datenbestand wider, auf den der Autor keinen Einfluss hat. Das Gewicht der Arbeit liegt auf den Darstellungsmethoden für biochemische Reaktionsnetze. Die Rich-

¹⁸Existierende Darstellungen täuschen lichte Netze vor, oft werden nur kleine Ausschnitte des Netzes visualisiert. In vielen Darstellungen werden zudem Substanzen mehrfach mit jeweils wenigen Kanten dargestellt, um die Dichte des Netzes zu reduzieren, siehe z. B. [Mic93].

¹⁹Die erste Auflage dieses heute aus zwei Teilen bestehenden Posters erschien 1968, seitdem wurde es bis zur letzten Auflage 1993 mehrfach aktualisiert und über eine Million mal gedruckt.

tigkeit der dargestellten biochemischen Information wird angestrebt, im Zweifelsfall sollten jedoch weitere Informationsquellen konsultiert werden.

1.4 Struktur der Arbeit

Diese Arbeit beschäftigt sich mit Informatik und Biochemie, wurde jedoch in erster Linie für Informatiker geschrieben. Deshalb wird intensiv auf biochemische Grundlagen eingegangen, während Grundlagen aus Mathematik und Informatik vorausgesetzt werden. Die Arbeit besteht aus zwei Teilen: Erstens der *Analyse* der Anforderungen an die Darstellung biochemischer Reaktionsnetze und der *Modellierung* solcher Netze unter dem Gesichtspunkt der Visualisierung, und zweitens der Entwicklung und Untersuchung geeigneter *Zeichenverfahren* und deren *Anwendung* auf Daten über biochemische Reaktionen.

1.4.1 Analyse und Modellierung

Im Zentrum des ersten Teils steht die Herausarbeitung der Anforderungen an die Visualisierung biochemischer Reaktionsnetze. Dazu werden in Kapitel 2 Grundlagen aus Biochemie und Informatik betrachtet. Darauf aufbauend beschäftigt sich Kapitel 3 mit den Darstellungsanforderungen. Diese lassen sich in drei Teile gliedern: lokale Anforderungen, globale Anforderungen und Anforderungen an kontexterhaltende Navigation und Sichten. Kapitel 4 untersucht, wie traditionelle Darstellungen diesen Anforderungen gerecht werden. Die entscheidende Schlussfolgerung ist, dass kein bekanntes Visualisierungsverfahren die Anforderungen erfüllt. Ein neues Verfahren ist also notwendig. Der erste Teil schließt mit Betrachtungen zur Modellierung biochemischer Reaktionsnetze.

1.4.2 Zeichenverfahren und dessen Anwendung

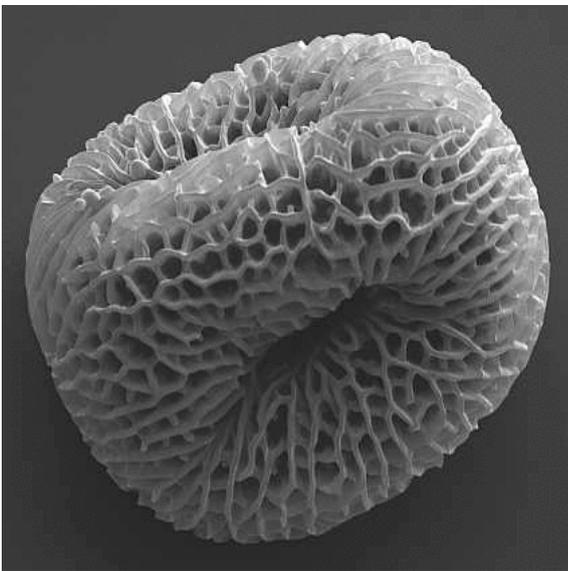
Im zweiten Teil wird ein ebenenweises Zeichenverfahren für Graphen entwickelt, das der Visualisierung biochemischer Reaktionsnetze dient. In Kapitel 6 wird das Zeichenverfahren betrachtet, dabei liegt die Betonung auf der Berücksichtigung beliebiger Knotengrößen und anwendergegebener Lagebeziehungen für Knoten. Für die Phasen des traditionellen ebenenweisen Zeichenverfahrens werden dazu Erweiterungen vorgestellt und untersucht. In Kapitel 7 werden Ergänzungen des Zeichenverfahrens beschrieben, das Ergebnis ist ein universelles ebenenweises Zeichenverfahren für gerichtete Graphen. Die Verwendung des Verfahrens zur Visualisierung biochemischer Reaktionsnetze wird in Kapitel 8 untersucht. Dabei werden auch Anwendungen von Reaktionsnetz-Darstellungen sowie die Darstellung zusätzlicher Information vorgestellt.

Die Arbeit endet mit einer Zusammenfassung und einem Ausblick in Kapitel 9. Zur besseren Orientierung sind ein Verzeichnis der Abbildungen, ein Symbolverzeichnis, ein Verzeichnis der Definitionen sowie ein Index angefügt.

Teil I

Analyse und Modellierung

2 Grundlagen



2.1 Grundlagen der Biochemie	14
2.2 Grundlagen der Informatik	24

Die Zelle ist die kleinste Einheit des Lebendigen. Sie ist die grundlegende Struktur- und Funktionseinheit eines Organismus. Alle Lebewesen bestehen aus Zellen, alle Aktivitäten eines Lebewesens beruhen letztendlich auf zellulären Prozessen. Gerade die Fülle von Formen und Funktionen der Zellen und Zellverbände ermöglicht die biologische Vielfalt. Ein Beispiel dafür zeigt die mit Hilfe eines Rasterelektronenmikroskops entstandene Aufnahme eines Pollenkorns von *Pelargonium sp.* (aus [REM99]).

2.1 Grundlagen der Biochemie

2.1.1 Biochemische Reaktionen

Die Biologie ist die Wissenschaft vom Leben. Bei der Betrachtung des Lebens unterscheidet man zwischen der *mikroskopischen Ebene*, die sich mit Biomolekülen und deren Interaktionen beschäftigt, und der *makroskopischen Ebene*, die sich mit Gestalt und Verhalten von Lebewesen befasst. Hier wird die mikroskopische Ebene untersucht, auf der die makroskopische Ordnung letztendlich beruht.

Biochemische Reaktionen gehören zur mikroskopischen Ebene. Sie finden in Zellen statt, wo sie die Bausteine der Zellen, also chemische Elemente und Verbindungen, umbauen. Die Erzeugung von Energie, die Synthese von Substanzen, Wachstum, Vermehrung und Reaktion auf Umwelteinflüsse sind an biochemische Reaktionen gebunden.

Im Folgenden werden die Grundlagen dieser Prozesse eingeführt. Die Darstellung beschränkt sich dabei auf biochemische Reaktionen und Reaktionsnetze, für weiterführende Information sei auf die Bücher [ABL⁺97, FM99, LNC94, Mic99, Str95] verwiesen.

Die Bedeutung biochemischer Begriffe variiert in der Literatur, zudem unterliegen einige Fachbegriffe einem Bedeutungswandel.¹ Zu diesem Problem schreiben Karp und Mavrovouniotis in [KM94]:

„The definition of concepts such as precursor and cofactor are complex and variable. Biologists should be able to choose among multiple alternative definitions of these terms.“

Zur Vermeidung von Mehrdeutigkeiten werden die wesentlichen biochemischen Begriffe hier in Form von Definitionen eingeführt, auch wenn es sich dabei nicht um Definitionen im mathematischen Sinn handelt.

2.1.1.1 Chemische Verbindungen und deren Umwandlung

DEFINITION 2.1 (CHEMISCHE GRUNDLAGEN)

Als Stoff wird jede Art von Materie bezeichnet, die durch gleich bleibende charakteristische Eigenschaften gekennzeichnet ist.

Ein chemisches Element ist ein mit chemischen Mitteln nicht weiter zerlegbarer Stoff.

Ein Atom ist der kleinste, elektrisch neutrale Bestandteil eines chemischen Elements, bestehend aus dem positiv geladenen Kern und negativ geladenen Elektronen.

Ein Molekül ist ein Stoff, bestehend aus einer endlichen Anzahl von chemischen Elementen, die eine bestimmte räumliche Verteilung aufweisen.

Eine chemische Bindung ist der durch verschiedene Bindungskräfte bewirkte Zusammenhalt von Atomen bzw. Atomgruppen innerhalb von Molekülen.

Ein Ion ist ein elektrisch geladenes Teilchen atomarer oder molekularer Größenordnung.

¹Viele Konzepte, beispielsweise die Unterscheidung von Substanzen und Cosubstanzen oder der Begriff des Cofaktors, werden in der Literatur nicht einheitlich verwendet (vergleiche beispielsweise [Mic99] mit [FM99]). Ein Beispiel für den Bedeutungswandel lässt sich am Begriff *Gen* zeigen: Mendel (1822–1884) prägte dieses Wort für einen Erbfaktor, der ein eindeutiges Merkmal eines Organismus beschreibt. Heute ist der Begriff *Gen* von der äußeren Erscheinungsform der Lebewesen losgelöst und beschreibt einen DNA-Abschnitt, der für die Synthese eines funktionsfähigen biologischen Produkts erforderlich ist.

Oft wird zwischen chemischen Bindungen im engeren und im weiteren Sinn unterschieden. Hier sind chemische Bindungen im engeren Sinn definiert, im weiteren Sinn zählen dazu auch die durch zwischenmolekulare Kräfte erzielten Wechselwirkungen zwischen Molekülen. Moleküle werden auch als *chemische Verbindungen* bezeichnet. In dieser Arbeit werden elektrische Wechselwirkungen zwischen Teilchen keine Rolle spielen, weshalb zwischen ungeladenen und geladenen Teilchen nicht weiter unterschieden wird. Wenn im Folgenden von *chemischen Elementen* oder *Verbindungen* gesprochen wird, so umfasst dies auch die zugehörigen Ionen.

Die chemischen Elemente eines Moleküls weisen eine bestimmte räumliche Anordnung auf, diese wird als *Struktur* des Moleküls bezeichnet. Moleküle werden oft mittels so genannter *Strukturformeln* dargestellt, siehe Abbildung 2.1(a). Die Darstellung ist so gewählt, dass sich die dreidimensionale Struktur und die Bindungen zwischen den Atomen aus der zweidimensionalen Zeichnung ablesen lassen. In Strukturformeln werden die chemischen Elemente durch ihre chemischen Symbole, die Bindungen zwischen ihnen durch Linien repräsentiert. Die Platzierung der chemischen Elemente und verschiedene Linienarten dienen der Kodierung der dreidimensionalen Struktur. Während in einer Strukturformel jedes Atom eines Moleküls explizit dargestellt ist, sind in *vereinfachten Strukturformeln* Atome wie C (Kohlenstoff) und H (Wasserstoff) weggelassen, wenn sie sich aus der Struktur des Moleküls ableiten lassen (vgl. Abb. 2.1(b)). Vereinfachte Strukturformeln werden oft verwendet, um die Darstellung von Molekülen kompakt zu gestalten.

DEFINITION 2.2 (BIOCHEMISCHE REAKTION)

Eine biochemische Reaktion (kurz Reaktion) R ist die Umwandlung von chemischen Elementen oder Verbindungen in andere Stoffe. Sie ist ein Vorgang, bei dem die chemischen Bindungen zwischen Atomen verändert werden. Ein oder mehrere Stoffe, Edukte genannt, werden dabei zu einem oder mehreren anderen Stoffen, den Produkten, umgewandelt.

Edukte und Produkte werden als *Substanzen* einer Reaktion bezeichnet. Edukte werden in der Literatur oft auch als *Ausgangssubstanzen*, *Ausgangsstoffe* oder *Reaktanten*, bei enzymkatalysierten Reaktionen auch als *Substrate* bezeichnet, Produkte als *Endsubstanzen*, *Endstoffe* oder *Reaktionsprodukte*.

2.1.1.2 Enzyme als Katalysatoren

DEFINITION 2.3 (ENZYM)

Ein Enzym ist eine chemische Verbindung, die eine spezifische Reaktion beschleunigt, sie katalysiert diese Reaktion.

Enzyme beschleunigen den Reaktionsablauf, indem sie die Energiebarriere zwischen Edukten und Produkten verringern. Die Beschleunigung erfolgt dabei mit dem Faktor 10^3 bis 10^{14} [FM99, Mic99]. Enzyme ermöglichen dadurch, dass Reaktionen, die sonst nur in spezifischen Umgebungen stattfinden (beispielsweise bei hohen Temperaturen oder speziellen pH-Werten), in der Zelle ablaufen können. Fast alle Reaktionen in Zellen werden durch Enzyme katalysiert.

Enzyme besitzen ein *aktives Zentrum*, welches der Teil der Enzymstruktur ist, der die Bindung des Edukts und die Katalyse bewirkt. Abbildung 2.2 stellt die typischen Komponenten einer enzymkatalysierten Reaktion dar. Enzyme gehen unverändert aus einer Reaktion hervor. Sie werden in Darstellungen deshalb oft nicht an die Enden, sondern neben den Reaktionspfeil geschrieben.

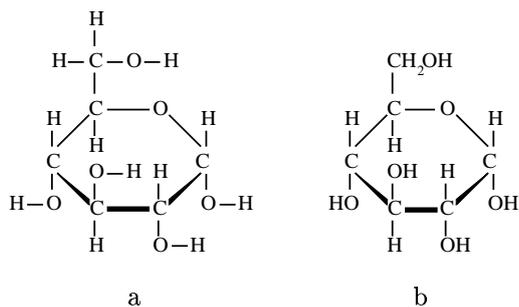


Abbildung 2.1: **Strukturformel.** (a) Strukturformel und (b) vereinfachte Strukturformel des Moleküls *Glucose*.

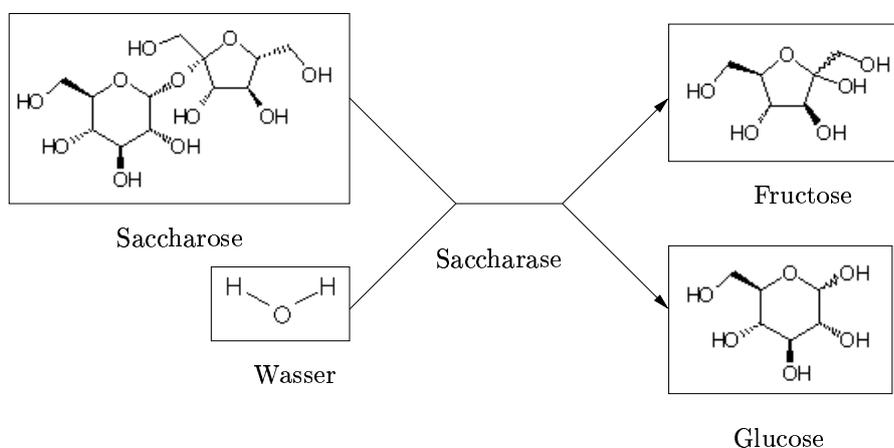


Abbildung 2.2: **Biochemische Reaktion.** Die typischen Komponenten einer enzymkatalysierten Reaktion: Die Edukte *Saccharose* und *Wasser* reagieren zu den Produkten *Fructose* und *Glucose*. Das Enzym *Saccharase* katalysiert diese Reaktion. Für Edukte und Produkte ist zur Veranschaulichung der Veränderung der Bindungen die Molekülstruktur dargestellt. So entspricht der *Glucose*-Ring der linken Teilstruktur des Moleküls *Saccharose*.

Strukturformeln können verschieden dargestellt werden. So unterscheidet sich die hier verwendete vereinfachte Strukturformel des Moleküls *Glucose* von der in Abbildung 2.1.

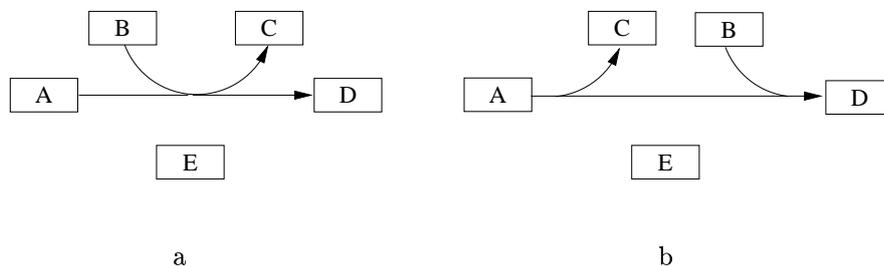


Abbildung 2.3: **Mehrkomponentenreaktionen.** Die durch das Enzym E katalysierte Reaktion $A + B \rightarrow C + D$ kann verschieden ablaufen (zur Erklärung siehe Abschnitt 2.1.1.2).

Viele Enzyme benötigen für ihre Funktion weitere Stoffe, so genannte *Cofaktoren*. Diese können anorganische Stoffe, beispielsweise Metallionen, oder organische Stoffe, so genannte *Coenzyme*, sein. Bezogen auf Reaktionen verhalten sich Cofaktoren unterschiedlich: Sie können unterteilt werden in Cofaktoren, die während der Reaktion verändert werden (z. B. $\text{NAD}^+ \rightarrow \text{NADH} + \text{H}^+$) und solche, die nicht verändert werden (z. B. Mg^{2+}). Erstere werden wegen ihrer Veränderung durch die Reaktion auch zur Gruppe der Substanzen gerechnet, Letztere zur Gruppe der Enzyme. Im Folgenden wird diese Zuordnung verwendet und Cofaktoren werden nicht speziell behandelt.

2.1.1.3 Teilreaktionen

Eine Reaktion zerfällt oft in mehrere Teilschritte. Dabei ist die *Reihenfolge der Teilreaktionen* wichtig und muss in der Visualisierung berücksichtigt werden. Die folgenden Ausführungen sollen dies verdeutlichen.

Bei einer enzymkatalysierten Reaktion $A \rightarrow B$ lagert sich das Edukt A an das Enzym an und bildet mit ihm einen Edukt-Enzym-Komplex (AE). Danach erfolgt die Umwandlung zum Produkt-Enzym-Komplex (BE) und die Freisetzung des Produkts B. Dies lässt sich auch schreiben als $A + E \rightarrow \text{AE} \rightarrow \text{BE} \rightarrow B + E$, diese Art von Reaktion wird als *Einkomponentenreaktion* bezeichnet.

Enzymkatalysierte Reaktionen mit mehreren Edukten verlaufen in gleicher Weise wie Einkomponentenreaktionen über die Bildung von Edukt-Enzym-Komplexen. Dabei gibt es jedoch verschiedene Möglichkeiten für die Reaktionen zwischen den Edukten, die am Beispiel einer Zweikomponentenreaktion mit den Edukten A und B und den Produkten C und D dargestellt werden sollen (siehe auch Abb. 2.3). Die Edukte A und B können mit E einen Übergangskomplex ABE bilden, der zu CDE reagiert und bei Zerfall die Produkte C und D freigibt. Bildung und Zerfall des Übergangskomplexes kann dabei in einer festen Reihenfolge geschehen, beispielsweise $A + E + B \rightarrow \text{AE} + B \rightarrow \text{ABE} \rightarrow \text{CDE} \rightarrow C + D + E$ (Abb. 2.3(a)), oder ungeordnet in einer beliebigen Reihenfolge. Eine weitere Möglichkeit ist die temporäre Umwandlung des Enzyms durch Reaktionen der Art $A + E \rightarrow \text{AE} \rightarrow \text{CF} \rightarrow C + F$ und $B + F \rightarrow \text{BF} \rightarrow \text{DE} \rightarrow D + E$, wobei die beiden Reaktionen als eine biochemische Reaktion betrachtet werden (Abb. 2.3(b)). Reaktionen mit mehreren Edukten bzw. Produkten werden auch als *Mehrkomponentenreaktionen* bezeichnet.

2.1.1.4 Reaktionsgleichgewicht und Richtung der Reaktionen

Biochemische Reaktionen sind *reversibel*, das heißt, sie laufen sowohl in Hin- wie in Rückrichtung ab. Sie sind dabei durch ein *Reaktionsgleichgewicht* charakterisiert. Isoliert würden die Reaktionen ein Gleichgewicht erreichen, bei dem der Umsatz der Hinreaktion gleich dem Umsatz der Rückreaktion ist. Dies wird durch den Doppelpfeil in $A + B \rightleftharpoons C + D$ ausgedrückt. Die Zelle steht jedoch mit der Umgebung in ständigem Stoffaustausch und viele Reaktionen sind *reguliert*, sie werden also durch andere Faktoren gefördert oder gehemmt. Dadurch wird die Gleichgewichtseinstellung verhindert. Man kann von einer *Richtung der Reaktion* sprechen. Dies spiegelt sich auch in der Einteilung der Substanzen in Edukte und Produkte wieder und wird oft durch einen Pfeil in $A + B \rightarrow C + D$ statt dem Doppelpfeil ausgedrückt.

2.1.2 Biochemische Reaktionswege und Reaktionsnetze

In Zellen finden verschiedene Reaktionen statt. Oft kommen Substanzen einer Reaktion auch in anderen Reaktionen als Produkt oder Edukt vor. Die folgenden Definitionen strukturieren dieses dichte Netz, da es bei der Untersuchung der Vorgänge in Zellen üblich ist, nur ausgewählte Verbindungen zwischen den Reaktionen zu betrachten und die Komponenten einer Reaktion in Klassen zu unterteilen.

2.1.2.1 Biochemische Reaktionswege

DEFINITION 2.4 (BIOCHEMISCHER REAKTIONSWEG)

Ein biochemischer Reaktionsweg (kurz: Reaktionsweg) $RW = (R_1, \dots, R_n)$ ist eine Sequenz von Reaktionen R_1, \dots, R_n zur Umsetzung einer Substanz in eine andere. Für alle $1 \leq i < n$ ist mindestens ein Produkt der Reaktion i zugleich Edukt der Reaktion $i + 1$.

Einige Reaktionswege sind besonders ausgezeichnet, da sie für Organismen wichtige Substanzen, sogenannte *Schlüsselsubstanzen*, ineinander umwandeln. Diese ausgezeichneten Reaktionswege haben eigene Namen und werden zur Unterscheidung von beliebigen Reaktionswegen als *Stoffwechselwege* bezeichnet. Bekannte Stoffwechselwege sind der *Citratzyklus* oder die *Glycolyse*. Abbildung 2.4(a) stellt den Stoffwechselweg zur *Synthese von Valin* dar.

Die Auswahl der Schlüsselsubstanzen und die Einteilung der Reaktionen in Stoffwechselwege ist nicht verbindlich geregelt. In dieser Arbeit wird die Einteilung entsprechend [Mic93] und [Mic99] verwendet, siehe auch Abbildung 2.5. Ein Stoffwechselweg, also eine Sequenz von Reaktionen, kann auch als eine Reaktion zwischen Schlüsselsubstanzen gesehen werden. Diese so genannte *Übersichtsreaktion* ist eine abstrakte Reaktion, die Schlüsselsubstanzen ineinander umwandelt (vgl. Abb. 2.4(b)).

DEFINITION 2.5 (SUBSTANZ, COSUBSTANZ)

Wird in einem Reaktionsweg das Produkt einer Reaktion als Edukt der folgenden Reaktion verwendet, so wird dieses als *Substanz des Reaktionswegs* bezeichnet. Am Anfang und Ende des Reaktionswegs wird das chemisch der folgenden bzw. vorhergehenden Substanz ähnlichere Molekül Substanz genannt.

Kommt das Produkt einer Reaktion dagegen nicht als Edukt in der folgenden Reaktion vor bzw. ist das Edukt einer Reaktion nicht Produkt der vorhergehenden Reaktion in dem betrachteten Reaktionsweg, so wird dieses als *Cosubstanz des Reaktionswegs* bezeichnet.

Bei der Betrachtung von Reaktionswegen werden die Ausgangs- und Endstoffe der Reaktionen in zwei Klassen eingeteilt: *Substanzen* (bestehend aus *Edukten* und *Produkten*) und *Cosubstanzen* (bestehend aus *Coedukten* und *Coprodukten*).

Eine Cosubstanz ist eine Substanz, die in der aktuellen Betrachtung eine untergeordnete Rolle spielt. Die Unterteilung in Substanzen und Cosubstanzen einer Reaktion erfolgt relativ zum betrachteten Reaktionsweg. Sie kann durch den Anwender verändert werden, da die Wichtigkeit einer Substanz gegenüber anderen Substanzen von der Anwendung abhängig sein kann. Oft entsprechen Cofaktoren den Cosubstanzen. Im Reaktionsweg in Abbildung 2.4(a) ist *Dihydroxy-Isovalerat* eine Substanz, *Glutamat* ein Coedukt und *2-Oxoglutarat* ein Coprodukt. Substanzen innerhalb des Reaktionswegs werden auch als *Metabolite* bezeichnet.

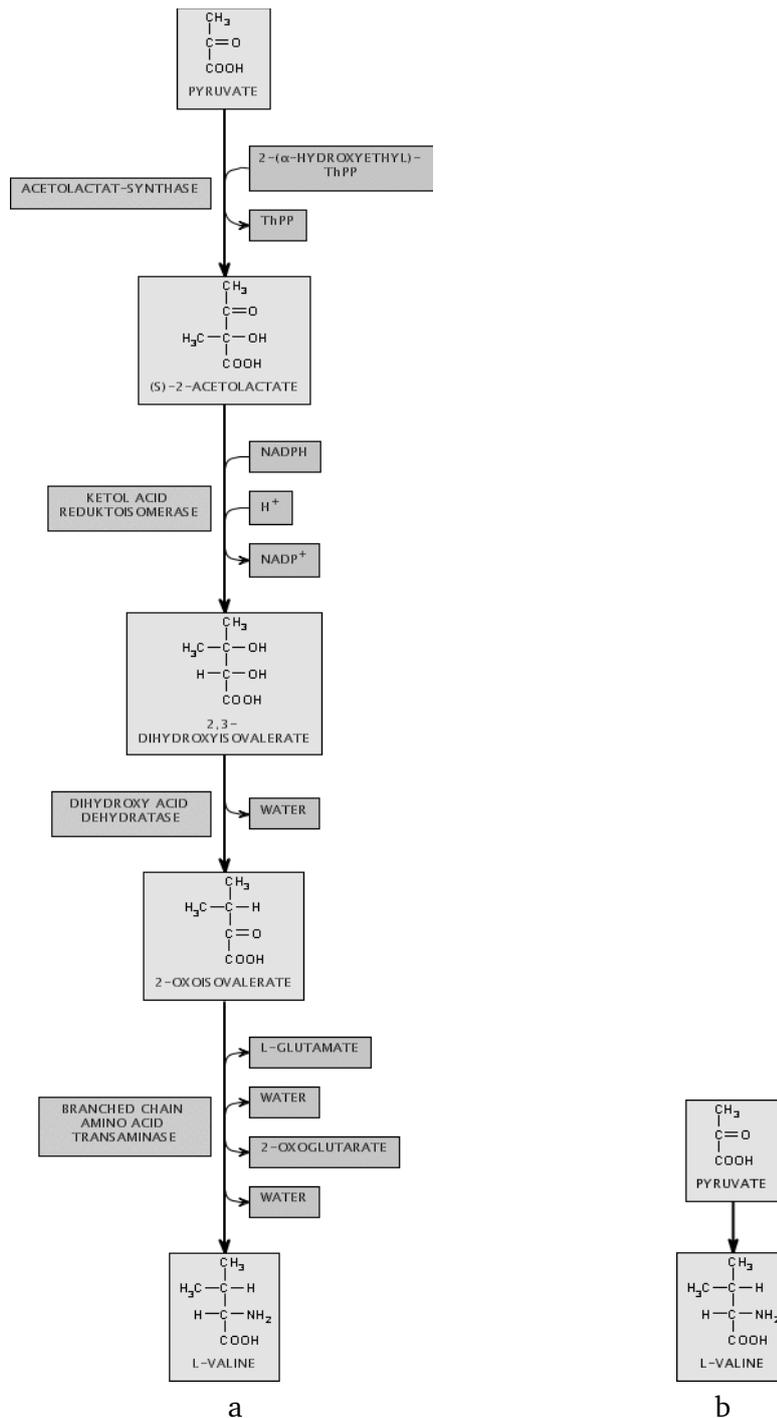


Abbildung 2.4: **Stoffwechselweg.** (a) Der Stoffwechselweg zur Synthese der verzweigt-kettigen Aminosäure *Valin* aus *Pyruvat*.² (b) Darstellung des Stoffwechselwegs als Übersichtsreaktion zwischen den beiden Schlüsselsubstanzen.³

²Im Text werden stets die einfachsten deutschen Namen gewählt, während die automatisch erzeugten Abbildungen meist englische und teilweise erweiterte Namen enthalten. So ist *L-Valine* der englische Name für ein Enantiomer der Aminosäure *Valin*. Enantiomere sind Isomere (Verbindungen mit gleicher Summenformel aber verschiedener Struktur), die sich zueinander wie Bild und Spiegelbild verhalten.

³Darstellungen dieser Art wurden mit *BioPath* [Bio01] erzeugt. *BioPath* ist die Realisierung der in dieser Arbeit beschriebenen Verfahren zur Visualisierung biochemischer Reaktionsnetze.

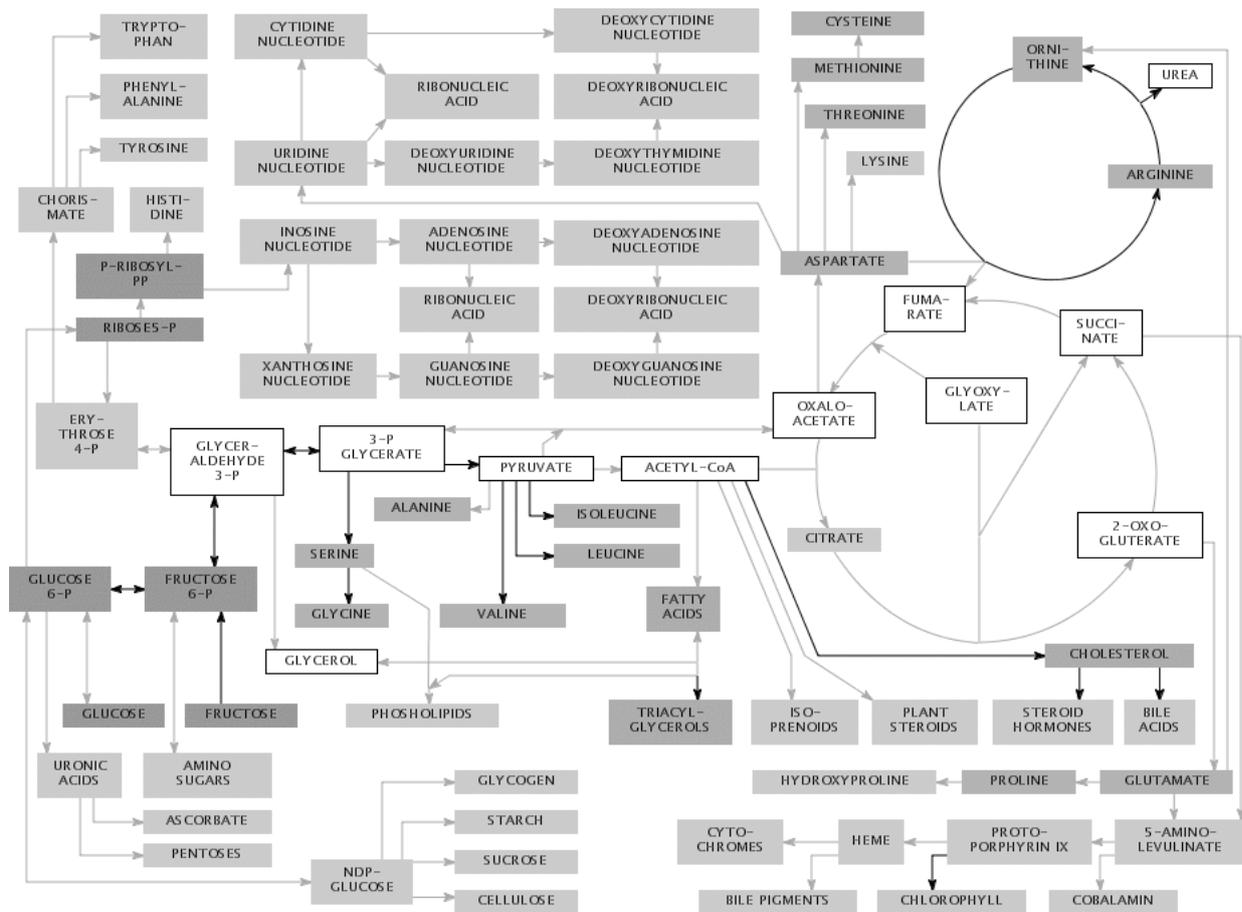


Abbildung 2.5: **Stoffwechselwege - Überblick.** Überblick über die Stoffwechselwege in Zellen (aus BioPath [Bio01], ähnlich zu [Mic99]). Die dargestellten Substanzen sind Schlüsselsubstanzen des Stoffwechsels. Die Farben in der Originaldarstellung charakterisieren verschiedenen Gruppen von Substanzen, z. B. Aminosäuren.

2.1.2.2 Geschlossene und offene Zyklen als spezielle Reaktionswege

DEFINITION 2.6 (GESCHLOSSENE UND OFFENE ZYKLEN)

Ein Reaktionsweg $RW = (R_1, \dots, R_n)$ heißt geschlossener Zyklus, wenn er wenige, zeitlich dicht aufeinander folgende Schritte umfasst, bei denen ein Edukt A in ein der Substanz A gleiches Produkt B umgewandelt wird und sich der Reaktionsweg RW mit B als neuem Edukt wiederholt.

Ein Reaktionsweg $RW = (R_1, \dots, R_n)$ heißt offener Zyklus, wenn er wenige, zeitlich dicht aufeinander folgende Schritte umfasst, bei denen ein Edukt A in eine der Substanz A ähnliches Produkt B umgewandelt wird und sich der Reaktionsweg RW mit B als neuem Edukt wiederholt.

Einige Reaktionswege weisen eine Struktur auf, bei der sich eine Folge der Reaktion nach wenigen Schritten wiederholt. Dabei kann man zwei Mechanismen unterscheiden: geschlossene und offene Zyklen. Beispiele für geschlossene Zyklen sind der Citratzyklus oder der Harnstoffzyklus. Nicht jeder Reaktionsweg, bei dem eine Substanz nach einigen Umwandlungen wieder in ursprünglicher Form

Der Stoffwechsel ist das Gesamtnetzwerk der in einer Zelle ablaufenden Reaktionen.

Hier wird unter dem Stoffwechsel (auch *Metabolismus* genannt) die Gesamtheit aller biochemischen Reaktionswege verstanden. In der Biologie wird der Begriff des Stoffwechsels manchmal weiter gefasst und beinhaltet neben biochemischen Reaktionen auch Transportprozesse und Signalübertragung.

2.1.3 Hierarchien

Das Wissen der Biologie ist in verschiedenen Dimensionen hierarchisch gegliedert. Beispiele für solche Hierarchien sind Abstammungsbäume, die die Entwicklung der Lebewesen widerspiegeln oder der Aufbau von Organismen aus Organen, die selbst aus Zellen bestehen, sowie deren Untergliederung in Zellkompartimente⁴. Im Folgenden werden zwei mit Reaktionen in Beziehung stehende Hierarchien vorgestellt: Die Hierarchie der biochemischen Reaktionen, die auch in dieser Arbeit eine wichtige Rolle spielt, sowie die Hierarchie der Enzyme als Beispiel einer weiteren mit Reaktionen zusammenhängenden Hierarchie.

2.1.3.1 Hierarchie biochemischer Reaktionen

Abbildung 2.7 stellt eine Hierarchisierung biochemischer Reaktionen dar, die ähnlich zur in Buch und Poster *Biochemical Pathways* [Mic93, Mic99] verwendeten ist. Dabei wird der Stoffwechsel in verschiedene Klassen von Stoffwechselwegen unterteilt: *katabole*, Energie gewinnende und Substanzen abbauende Stoffwechselwege, *anabole*, Substanzen synthetisierende Stoffwechselwege, und *amphibole*, die beiden Zielen dienen. Diese Klassen lassen sich weiter in die einzelnen Stoffwechselwege zerlegen, z. B. den Citratzyklus und die Glycolyse. Stoffwechselwege bestehen aus Reaktionen, die sich, wenn der Reaktionsmechanismus bekannt ist, in Teilreaktionen zerlegen lassen. Die Abbildung 2.5 stellt die Ebene der Stoffwechselwege in Zellen ähnlich zu [Mic99] dar, während Abbildung 1.1 auf Seite 4 einen kleinen Ausschnitt der Ebene der Reaktionen zeigt.

Da es verschiedene Ausgestaltungen der Hierarchie gibt,⁵ soll diese hier nicht festgeschrieben werden. Zur Unterstützung der Visualisierung wird von einer Hierarchie biochemischer Reaktionen jedoch gefordert, dass diese

1. *vollständig* ist, d. h. alle Objekte der Hierarchie sind von deren Spitze aus erreichbar.
2. *eindeutig* ist, d. h. ein Objekt ist nur einem in der Hierarchie übergeordneten Objekt zugeordnet.

Diese Eigenschaften unterstützen eine einfache Navigation durch Reaktionsnetze, da alle Reaktionen erreichbar und Auf- und Abwärtsbewegungen in der Hierarchie eindeutig festgelegt sind.

2.1.3.2 Hierarchie der Enzyme

Als eine weitere Hierarchie wird hier die Hierarchie der an den Reaktionen beteiligten Enzyme betrachtet. Enzyme können nach verschiedenen Gesichtspunkten, z.B. ihrem Vorkommen in verschie-

⁴Zellkompartimente sind durch Membranen abgeschlossene Räume in Zellen, z. B. Zellkern oder Mitochondrien.

⁵Auf Hierarchien biochemischer Reaktionen wird im Rahmen der Vorstellung existierender Systeme in Abschnitt 3.4.4 noch ausführlich eingegangen.

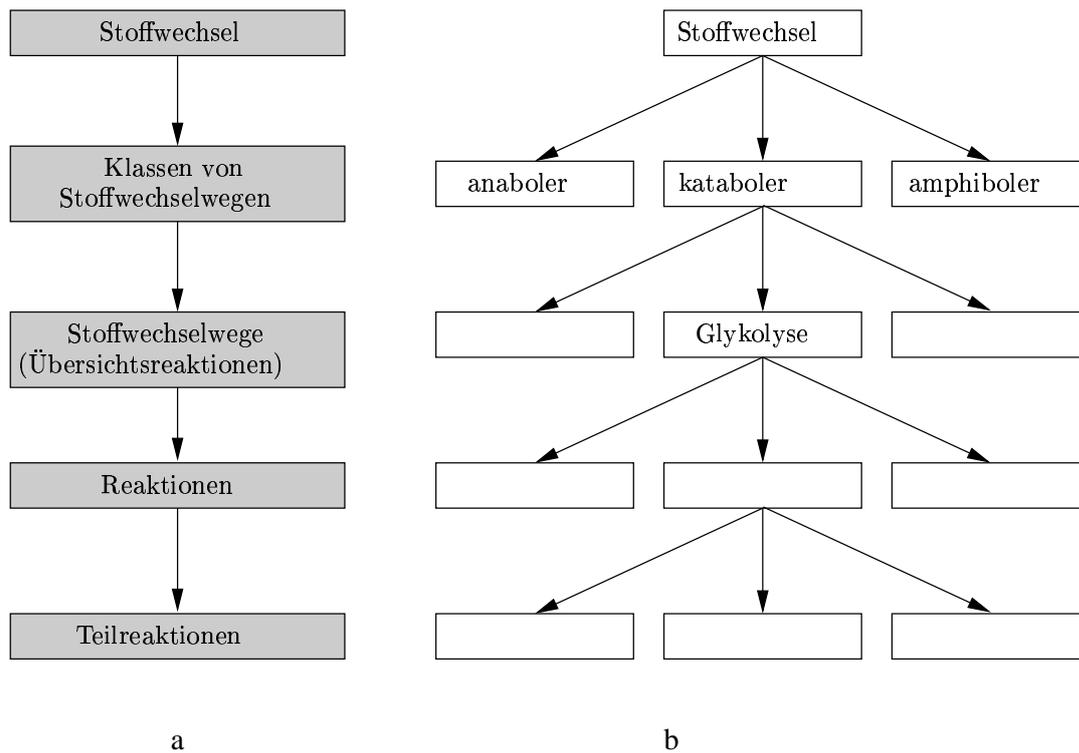


Abbildung 2.7: **Hierarchie biochemischer Reaktionen.** (a) Die Hierarchie biochemischer Reaktionen und (b) ein kleiner Ausschnitt der in der vorliegenden Arbeit verwendeten Hierarchie entsprechend [Mic93, Mic99]. Im *BioPath* Projekt liegen die meisten Informationen auf der Ebene der Reaktionen vor.

denen Organismen oder ihren funktionellen Gruppen⁶ eingeteilt werden. Am gebräuchlichsten ist jedoch die Einteilung nach der auf der Wirkungsspezifität beruhenden *EC-Nomenklatur* [Int92]. Dabei werden Enzyme mit einer vierstelligen *EC-Nummer* bezeichnet, wobei der erste Block die höchste, der letzte Block die niedrigste Hierarchiestufe repräsentiert.⁷

⁶Oft wird das chemische Verhalten eines Moleküls wesentlich durch einige seiner Atome oder Atomgruppen bestimmt. Eine *funktionelle Gruppe* eines Moleküls ist eine Menge von Atomen oder Atomgruppen, die die chemischen Eigenschaften des Moleküls bestimmt. Beispielsweise besitzen alle Alkohole als funktionelle Gruppe die Hydroxylgruppe -OH, alle Carbonsäuren die Carboxylgruppe -COOH.

⁷Der Aufbau dieser Klassifikation ist am Beispiel des Enzyms *Alkohol Dehydrogenase* in der folgenden Tabelle dargestellt. Die dabei auftretenden Begriffe sind nicht wichtig und werden im Folgenden nicht weiter verwendet.

EC-Nummer	Bedeutung
1	Enzyme, die Redoxreaktionen innerhalb eines Substratpaares katalysieren
1.1	Enzyme aus 1, die auf CHOH wirken
1.1.1	Enzyme aus 1.1 mit NAD ⁺ oder NADP ⁺ als Akzeptor
1.1.1.1	Alkohol Dehydrogenase

2.2 Grundlagen der Informatik

2.2.1 Graphentheorie

Zur Modellierung biochemischer Reaktionen als Graphen ist die Einführung einiger mathematischer Strukturen notwendig. Hier werden die wesentlichen Definitionen und Begriffe der Graphentheorie vorgestellt, um die in dieser Arbeit verwendete Terminologie zu klären. Für weiterführende Information sei auf die Bücher [BL95, CH94] verwiesen.

DEFINITION 2.8 (GRAPH)

Ein Graph $G = (V, E)$ besteht aus einer nichtleeren endlichen Menge von Knoten V und einer endlichen Menge von Kanten $E \subseteq V \times V$.⁸

Jede Kante $(u, v) \in E$ besteht aus einem geordneten Paar von Knoten. Dabei wird u als *Anfangsknoten* und v als *Endknoten* bezeichnet. Statt von einer *Knotenmenge* bzw. *Kantenmenge des Graphen* wird kurz von *Knoten* bzw. *Kanten des Graphen* gesprochen.

DEFINITION 2.9 (EIGENSCHAFTEN VON GRAPHEN)

Sei $G = (V, E)$ ein Graph, $(u, v) \in E$ eine Kante und $v \in V$ ein Knoten.

Die Kante (u, v) verbindet u mit v . Die Knoten u und v heißen *adjazent zueinander* und *inzident zu (u, v)* , die Kante (u, v) heißt *inzident zu u und v* . Sie wird als *ausgehende Kante von u* und *eingehende Kante von v* bezeichnet.

Der Knoten u heißt *Vorgänger von v* und der Knoten v *Nachfolger von u* .

$\text{Vor}(v) = \{u \mid (u, v) \in E\}$ bezeichnet die Menge aller Vorgänger von v und $\text{Nach}(v) = \{w \mid (v, w) \in E\}$ die Menge aller seiner Nachfolger.

Die Menge $\text{Ein}(v) = \{(u, v) \mid (u, v) \in E\}$ ist die Menge der eingehenden Kanten, die Menge $\text{Aus}(v) = \{(v, w) \mid (v, w) \in E\}$ die der ausgehenden Kanten von v .

$|\text{Ein}(v)|$, die Anzahl eingehender Kanten, wird als *Eingangsgrad*, $|\text{Aus}(v)|$ als *Ausgangsgrad* von v bezeichnet. $\Delta(G) = \max\{|\text{Ein}(v)| + |\text{Aus}(v)| \mid v \in V\}$ wird als *maximaler Knotengrad* von G bezeichnet.

Eine Kante $(u, v) \in E$ mit $u = v$ heißt *Schlinge*.

DEFINITION 2.10 (HYPERGRAPH)

Ein Hypergraph $G = (V, E)$ besteht aus einer nichtleeren endlichen Menge von Knoten V und einer endlichen Menge von Kanten $E \subseteq \mathcal{P}^+(V) \times \mathcal{P}^+(V)$ mit $\mathcal{P}^+(V) = \mathcal{P}(V) \setminus \{\emptyset\}$.⁹

Jede *Hyperkante* $(\{u_1, \dots, u_n\}, \{v_1, \dots, v_m\}) \in E$ besteht aus einem geordneten Paar von nichtleeren Knotenmengen. Dabei wird $\{u_1, \dots, u_n\}$ als Menge der *Anfangsknoten* und $\{v_1, \dots, v_m\}$ als Menge der *Endknoten* bezeichnet.

DEFINITION 2.11 (BENANNTER, GETYPTER UND GEWICHTETER GRAPH (HYPERGRAPH))

Ein *benannter Graph (Hypergraph)* $G = (V, E, B)$ ist ein Graph (Hypergraph) $G = (V, E)$ mit einer Abbildung $B : V \cup E \rightarrow \Sigma^*$, die jedem Knoten und jeder Kante ein Wort über dem Alphabet Σ zuordnet.

⁸Diese Graphen werden oft auch als *gerichtete Graphen* bezeichnet. Mehrfache Kanten von Knoten u nach Knoten v sind in dieser Definition ausgeschlossen.

⁹ $\mathcal{P}(M)$ bezeichnet die Potenzmenge einer Menge M . Diese Hypergraphen werden oft auch als *gerichtete Hypergraphen* bezeichnet.

Ein getypter Graph (Hypergraph) $G = (V, E, T)$ ist ein Graph (Hypergraph) $G = (V, E)$ mit einer Abbildung $T : V \rightarrow \text{Typen}$, die jedem Knoten ein Element aus einer endlichen Menge von Typen zuordnet.

Ein gewichteter Graph (Hypergraph) $G = (V, E, W)$ ist ein Graph (Hypergraph) $G = (V, E)$ mit einer Abbildung $W : E \rightarrow \mathbb{N}$, die jeder Kante ein Gewicht zuordnet.

Ein Wort wird auch als *Name* oder *Benennung* des Knotens bzw. der Kante bezeichnet. Gewichtete Graphen werden auch *bewertete Graphen* genannt.

Benannte, getypte und gewichtete Graphen lassen sich kombinieren: Seien $G_1 = (V, E, B)$ ein benannter Graph, $G_2 = (V, E, T)$ ein getypter Graph und $G_3 = (V, E, W)$ ein gewichteter Graph mit gleichen Knoten- und Kantenmengen, so ist $G = (V, E, B, T, W)$ ein benannter, getypter, gewichteter Graph.¹⁰ Analoges gilt für Hypergraphen.

DEFINITION 2.12 (PFAD UND ZYKLUS)

Sei $G = (V, E)$ ein Graph.

Ein Pfad $w = (v_1, \dots, v_k) \in V^k$ ist eine Knotenfolge, so dass für $1 \leq i < k$ gilt: $(v_i, v_{i+1}) \in E$. Ein Pfad w heißt Zyklus, falls $v_k = v_1$ mit $k > 1$.

Der Graph G heißt azyklisch, wenn er keinen Zyklus und keine Schlinge enthält.

Für einen Pfad $w = (v_1, \dots, v_k)$ wird auch $v_1 \rightarrow^* v_k$ geschrieben.

DEFINITION 2.13 (ZUSAMMENHANG UND ERREICHBARKEIT)

Sei $G = (V, E)$ ein Graph.

Zwei Knoten $u, v \in V$ heißen zusammenhängend, falls eine Folge von Knoten v_1, \dots, v_k existiert mit $v_1 = u, v_k = v$ und für alle $1 \leq i < k$ gilt: $(v_i, v_{i+1}) \in E$ oder $(v_{i+1}, v_i) \in E$. G heißt zusammenhängend, falls jedes Knotenpaar zusammenhängend ist.

Zwei Knoten $u, v \in V$ heißen stark zusammenhängend, falls zwei Pfade $u \rightarrow^* v$ und $v \rightarrow^* u$ existieren. G heißt stark zusammenhängend, falls jedes Knotenpaar stark zusammenhängend ist.

Ein Knoten $v \in V$ ist vom Knoten $u \in V$ aus erreichbar, falls ein Pfad von u nach v existiert.

Ein Knoten $v \in V$ ist stets von v aus erreichbar. Ein Graph $G = (V, E)$ ist stark zusammenhängend, falls für alle $u, v \in V$ der Knoten u vom Knoten v aus erreichbar ist.

DEFINITION 2.14 (MENGEN ZUSAMMENHÄNGENDER UND ERREICHBARER KNOTEN)

Seien $G = (V, E)$ ein Graph und $v \in V$ ein Knoten.

$ZK(v) = \{w \mid w \text{ und } v \text{ sind zusammenhängend}\}$ wird als Menge der mit v zusammenhängenden Knoten, $SZK(v) = \{w \mid w \text{ und } v \text{ sind stark zusammenhängend}\}$ als Menge der mit v stark zusammenhängenden Knoten und $EK(v) = \{w \mid w \text{ ist von } v \text{ aus erreichbar}\}$ als Menge der von v erreichbaren Knoten bezeichnet.

DEFINITION 2.15 (SPEZIELLE GRAPHEN)

Sei $G = (V, E)$ ein Graph.

¹⁰Solche Graphen lassen sich auch durch allgemeine attributierte Graphen ausdrücken, die eine Abbildung von Knoten und Kanten auf einen Vektor verschiedener Attribute besitzen. Die explizite Definition verschiedener Abbildungen wurde hier gewählt, da im Folgenden oft nur ausgewählte Attribute benötigt werden und die Beschreibung mit den hier verwendeten speziellen Abbildungen dann deutlicher als bei Verwendung eines Attributvektors ist.

G heißt bipartit, falls zwei disjunkte, nichtleere Mengen V_1 und V_2 existieren, so dass

$$\begin{aligned} V &= V_1 \cup V_2 \\ E &= E_1 \cup E_2, E_1 \subseteq V_1 \times V_2, E_2 \subseteq V_2 \times V_1 \end{aligned}$$

G heißt Baum, wenn ein Knoten $w \in V$ existiert, so dass alle Knoten $v \in V$ von w aus erreichbar sind und $|E| = |V| - 1$ gilt.

G heißt DAG¹¹, falls G azyklisch ist.

Ein Graph $G' = (V', E')$ mit $V' \subseteq V$ und $E' \subseteq E \cap (V' \times V')$ heißt Teilgraph von G . Falls $E' = E \cap (V' \times V')$ gilt, dann wird G' als induzierter Teilgraph von G bezeichnet.

Der durch V' induzierte Teilgraph eines Graphen G wird mit $G_{V'}$ bezeichnet. Bipartite Graphen werden als $G = (V_1 \cup V_2, E_1 \cup E_2)$ notiert.

Als Wurzeln eines Baumes oder DAGs werden die Knoten ohne eingehende Kanten und als Blätter die Knoten ohne ausgehende Kanten bezeichnet. Alle Knoten eines Baumes oder DAGs, die keine Blätter sind, heißen innere Knoten.

DEFINITION 2.16 ((STARKE) ZUSAMMENHANGSKOMPONENTE, ERREICHBARKEITSKOMPONENTE)

Sei $G = (V, E)$ ein Graph und $v \in V$ ein Knoten.

Der Graph $G_{ZK(v)}$ wird als die durch v induzierte Zusammenhangskomponente von G bezeichnet.

Der Graph $G_{SZK(v)}$ wird als die durch v induzierte starke Zusammenhangskomponente von G bezeichnet.

Der Graph $G_{EK(v)}$ wird als die durch v induzierte Erreichbarkeitskomponente von G bezeichnet.

DEFINITION 2.17 (TOPOLOGISCHE SORTIERUNG)

Sei $G = (V, E)$ ein DAG und $TS : V \rightarrow \mathbb{N}$ eine Abbildung. TS heißt topologische Sortierung, wenn gilt: $\forall (u, v) \in E \quad TS(u) < TS(v)$.

Topologisches Sortieren kann auch als eine Form des Sortierens gesehen werden, das Elemente einer Menge nicht linear, sondern partiell ordnet. Bei Graphen wird die Menge der Knoten sortiert. Eine wichtige topologische Sortierung ist die Sortierung entsprechend dem längsten Pfad, wobei $TS(v) = \max\{TS(w) \mid w \in \text{Vor}(v)\} + 1$ für alle Kanten $(u, v) \in E$ gilt.¹²

2.2.2 Visualisierung

Unter Visualisierung wird hier sowohl die Erzeugung eines Bildes aus Daten (der Prozess der Visualisierung) als auch das entstehende Bild (die visuelle Repräsentation der Daten) verstanden. Visualisierungen dienen der Darstellung und Exploration von Daten. Sie werden verwendet, weil Menschen eine ausgeprägte visuelle Wahrnehmung besitzen und geeignete, gut gezeichnete Bilder schneller verstehen als entsprechende textuelle oder tabellarische Darstellungen (siehe auch Abb. 2.8). Visualisierungen sollen so gestaltet werden, dass die den Daten zu Grunde liegenden Phänomene visualisiert und dadurch verstehbar werden.

Bei Visualisierungen sind zwei Aspekte zu unterscheiden: Das Modellieren der Daten und das Finden geeigneter grafischer Repräsentationen. Die Modellierung biochemischer Reaktionsnetze erfolgt

¹¹DAG - directed acyclic graph.

¹²Es ist hier nicht gefordert, dass TS bijektiv ist.

mittels Graphen. Das Finden einer geeigneten grafischen Repräsentation entspricht der Wahl einer zur Modellierung passenden Visualisierungstechnik, in dieser Arbeit einem Verfahren zum Zeichnen von Graphen.

2.2.2.1 Visualisierungstechniken

Es gibt verschiedene Arten von Visualisierungen. Diese reichen von Torten- und Balkendiagrammen oder Wetterkarten, die man aus der täglichen Anschauung kennt, bis zu speziellen Techniken, mit denen beispielsweise die Ladungsverteilung zwischen Atomen dargestellt wird (siehe die Beispiele in Abb. 2.9). Die verschiedenen Darstellungsverfahren werden auch als *Visualisierungstechniken* bezeichnet. Auf Grund der Vielzahl verschiedener Visualisierungstechniken ist es hier nicht möglich, einen umfassenden Überblick zu geben. Dazu sei auf Bücher und Artikel wie [ARWS94, BCE⁺92, Che99, EW92] verwiesen.

2.2.2.2 Erzeugung der Visualisierungen

Die Visualisierung von Daten kann manuell durchgeführt werden. Dies ist oft mühsam und erfordert einen Experten, führt andererseits zu guten Darstellungen besonders bei komplexen Zusammenhängen, da das Expertenwissen in die Visualisierung einfließt. Visualisierungen lassen sich auch automatisch mit Hilfe von Computern erzeugen. Der anfänglich meist sehr hohe Aufwand, entsprechende Programme zur Berechnung der Darstellungen zu entwickeln, lohnt insbesondere, wenn sich Daten oft ändern oder beliebige Ausschnitte großer Datenmengen zu visualisieren sind.

Bei einer Reihe von Visualisierungstechniken, beispielsweise Tortendiagrammen oder der Darstellung mathematischer Funktionen, ist die durch automatische Visualisierung erreichte Qualität so hoch, dass diese Verfahren die manuelle Erstellung der Zeichnungen verdrängt haben. In anderen Anwendungen, z. B. beim Erstellen von Straßenkarten, erfolgt die Visualisierung dagegen noch manuell mit Computerunterstützung. Automatische Verfahren liefern hier Lösungsvorschläge, die anschließend verbessert werden; bei Straßenkarten betrifft dies beispielsweise die Platzierung der Städtenamen. Der Vorteil automatischer Verfahren ist neben der beschleunigten Erzeugung der Darstellungen, dass auch Laien durch Anwendung dieser Verfahren Visualisierungen erzeugen können, die gewissen Qualitätsstandards genügen.

2.2.2.3 Sichten

Oft sind Daten Ausdruck komplexer Wechselwirkungen zwischen Objekten. In vielen Anwendungen sollen diese Wechselwirkungen nicht gleichzeitig in einem Bild dargestellt werden, meist ist eine Visualisierung jeder einzelnen oder einiger weniger Wechselwirkungen wesentlich verständlicher. So genannte *Sichten* dienen dazu, die Daten abgestimmt auf die aktuellen Anforderungen zu präsentieren und verschiedene Informationen sichtbar zu machen. *Sichten* sind verschiedene Visualisierungen derselben Daten, die jeweils ausgewählte Aspekte dieser Daten betonen. Ein Beispiel ist in Abbildung 2.10 zu sehen.

„Bei Pflanzen und Bakterien beginnen die Synthesewege für Valin und Isoleucin mit der Kondensation einer 2-Oxosäure mit Hydroxyethyl-ThPP, einem Zwischenprodukt der Acetolactat-Synthese (und der Pyruvat-Dehydrogenase, 3.3.1). Daran schließt sich eine Reduktion-Alkylgruppenwanderung an; ...“

textuelle Darstellung, aus [Mic99].

Edukt	Produkt	Enzym
Pyruvat	(S)-2-Acetolactat	Acetolactat Synthase
(S)-2-Acetolactat	2,3-Dihydroxy-Isovalerat	Ketolsäure Reducto-Isomerase
2,3-Dihydroxy-Isovalerat	2-Oxo-Isovalerat	Dihydroxysäure Dehydratase
2-Oxo-Isovalerat	Valin	Transaminase der verzweigt-kettigen Aminosäuren

tabellarische Darstellung

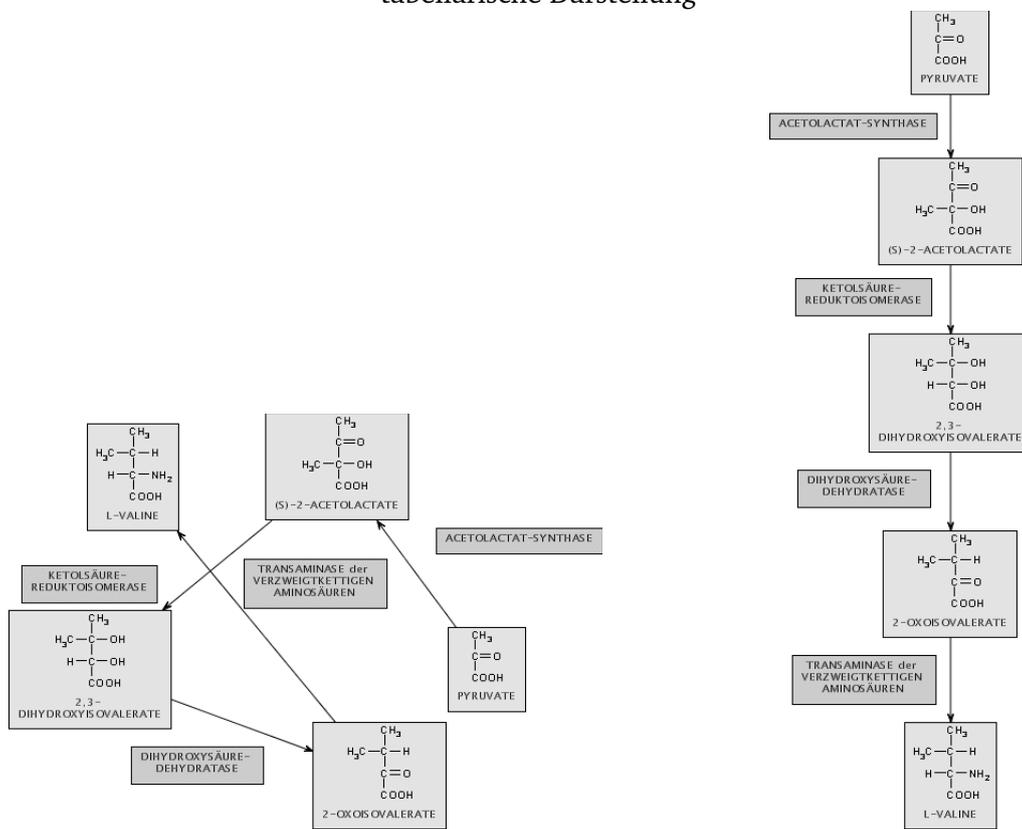


Abbildung 2.8: **Visualisierung.** Geeignete Bilder sind verständlicher als entsprechende textuelle und tabellarische Darstellungen. Beispiel für verschiedene Darstellungen des Stoffwechselwegs *Synthese von Valin* (ohne Cosubstanzen).

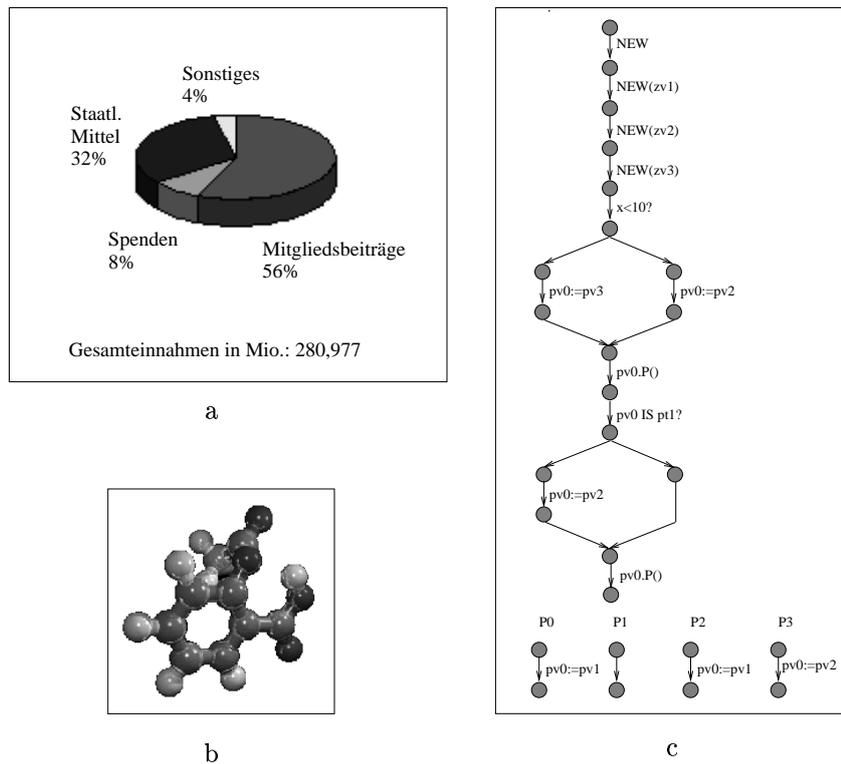


Abbildung 2.9: **Visualisierungstechniken.** Einige Visualisierungstechniken: (a) *Tortendiagramm* des prozentualen Anteils der Einnahmen an den Gesamteinnahmen der SPD 1997 (aus [Par00]), (b) *Kugel-Stab-Modell* des Moleküls Aspirin, (c) *Zeichnung eines Graphen* zur Veranschaulichung des Programmablaufs (aus [KS97]).

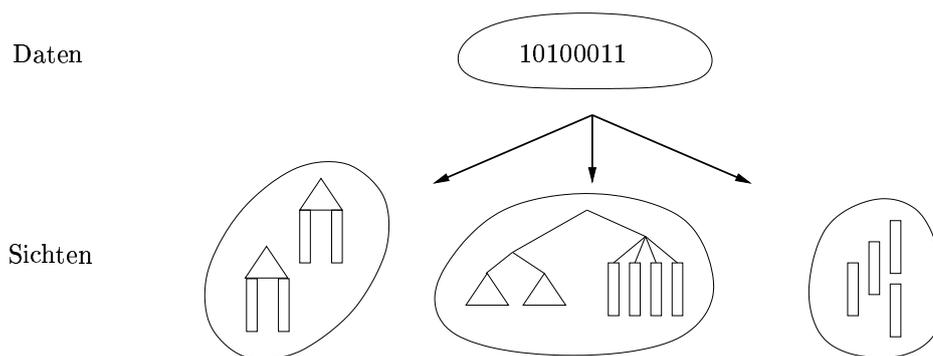


Abbildung 2.10: **Sichten.** Drei verschiedene Sichten auf dieselben Daten. Sichten betonen unterschiedliche Aspekte, können aber wie hier in der rechten Sicht auch Information weglassen.

2.2.3 Zeichnen von Graphen als Visualisierungstechnik

Für die Grundlagen des Zeichnens von Graphen sei auf [DETT99] verwiesen.¹³ Hier werden die im Folgenden benötigten Aspekte wiederholt und an die Erfordernisse der Arbeit angepasst.

Zeichnungen von Graphen erhält man sehr einfach, indem Knoten durch Punkte repräsentiert werden. Kanten werden als Liniensegmente betrachtet, deren Endpunkte mit Knoten übereinstimmen, siehe beispielsweise die Definition von *geometric graphs* im Buch von Pach und Agarwal [PA95].

In vielen Anwendungen haben jedoch die Form und Größe von Knoten oder die Art der Kanten eine Bedeutung. Für Visualisierungen biochemischer Reaktionsnetze sind beispielsweise Rechtecke mit echten Größen für Knoten notwendig.

Erweiterungen von Graphen, die deren realistische geometrische Repräsentation ausdrücken, finden sich in verschiedenen Arbeiten. So führt Brandenburg *decorated graphs* [Bra99] ein, bei denen Knoten durch beliebige Polygone darstellbar sind, Kanten erhalten Information wie Breite der Kante und Berührungspunkt am Polygon. *Realistische Graphen* von Sanders [San96b] erlauben für Knoten Rechtecke, Kanten werden als Polygonzüge dargestellt. Hier werden diese Vorschläge aufgegriffen und Graphen mit zusätzlichen Größeninformationen definiert. Wie bei Sanders [San96b] erfolgt eine Beschränkung auf Rechtecke und Polygonzüge, wobei sich die Definitionen an andere Formen anpassen lassen.

DEFINITION 2.18 (EBENE UND OBJEKTE)

Als Ebene wird der $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$ mit der euklidischen Norm $\| (x, y) \| = \sqrt{x^2 + y^2}$ bezeichnet.

Ein Punkt $(x, y) \in \mathbb{R}^2$ ist ein Element der Ebene, x und y sind seine Koordinaten.

Ein Rechteck mit Höhe h und Breite b am Punkt (x, y) ist eine Menge von Punkten mit

$$\text{Rechteck}(h, b, (x, y)) = \{(x_R, y_R) \mid x - \frac{b}{2} \leq x_R \leq x + \frac{b}{2} \text{ und} \\ y - \frac{h}{2} \leq y_R \leq y + \frac{h}{2}\}$$

Eine Strecke zwischen zwei Punkten (x_1, y_1) und (x_2, y_2) ist eine Menge von Punkten mit

$$\text{Strecke}((x_1, y_1), (x_2, y_2)) = \{(x_s, y_s) \mid (x_s - x_1) * (y_2 - y_1) = (x_2 - x_1) * (y_s - y_1), \\ \min\{x_1, x_2\} \leq x_s \leq \max\{x_1, x_2\}, \min\{y_1, y_2\} \leq y_s \leq \max\{y_1, y_2\}\}$$

Ein Polygonzug entlang von Stützpunkten $((x_1, y_1), \dots, (x_n, y_n))$ ist

$$\text{Polygonzug}(((x_1, y_1), \dots, (x_n, y_n))) = \text{Strecke}((x_1, y_1), (x_2, y_2)) \cup \dots \cup \\ \text{Strecke}((x_{n-1}, y_{n-1}), (x_n, y_n))$$

Ein Polygonzug wird auch *Polylinie* genannt. Punkt, Rechteck und Polygonzug werden als *Objekte* der Ebene bezeichnet.

Die Visualisierung von Graphen wird in zwei Aspekte zerlegt:

1. Die Platzierung der Knoten und Kanten, also die Bestimmung ihrer Koordinaten.

¹³Das Zeichnen von Graphen ist ein aktives Forschungsgebiet der Informatik. Auf Grund der Vielzahl der Arbeiten kann hier nur ein Überblick gegeben werden. Viele weiter gehende Informationen finden sich in Konferenzbänden *Graph Drawing* [Bra96, Di 97, Kra99, Mar01, Nor97, TT95, Whi98] sowie in [BJM97, CT99, DETT94, EM99, Tam99b].

2. Das eigentliche Zeichnen des Graphen, also die Darstellung der Objekte mit ihren Größen und Formen an den vorher bestimmten Koordinaten.

DEFINITION 2.19 (PLATZIERUNG VON GRAPHEN)

Sei $G = (V, E)$ ein Graph. Eine Platzierung $P = (P_V, P_E)$ von G besteht aus zwei Abbildungen $P_V : V \rightarrow \mathbb{R}^2$ und $P_E : E \rightarrow \mathbb{R}^2 \times \dots \times \mathbb{R}^2$, die jedem Knoten $v \in V$ eine x - und y -Koordinate $P_V(v) = (x, y)$ und jeder Kante $e \in E$ die x - und y -Koordinaten ihrer Stützpunkte $P_E(e) = ((x_1, y_1), \dots, (x_n, y_n))$ zuordnen.

Durch eine Platzierung eines Graphen werden die Knoten auf Punkte und die Kanten auf Folgen von Stützpunkten in der Ebene abgebildet. Eine Platzierung entspricht dabei aber nicht der endgültigen Zeichnung, sondern gibt nur die Lage von Knoten und Kanten an. Bei der Visualisierung werden dann Knoten durch Objekte, z.B. Kreise oder Polygone und Kanten durch Polylinien oder Kurven dargestellt. Dadurch wird die Berechnung einer Platzierung getrennt von geometrischen Objekten. So ist die Platzierung eines Rechtecks identisch mit der eines rechteckigen Bildes oder die Platzierung einer durchgezogenen Linie identisch mit der einer gestrichelten Linie.

In dieser Arbeit sind Graphen mit rechteckigen Knoten besonders wichtig. Diese sind wie folgt definiert:

DEFINITION 2.20 (GEOMETRISCHER GRAPH)

Ein geometrischer Graph $G = (V, E, M)$ ist ein Graph $G = (V, E)$ mit einer Abbildung $M : V \rightarrow \mathbb{R}^2$, die jedem Knoten $v \in V$ die Höhe h und Breite b eines Rechtecks zuordnet, $M(v) = (h, b)$.

Für geometrische Graphen $G = (V, E, M)$ werden zwei Abbildungen Höhe, Breite : $V \rightarrow \mathbb{R}$ eingeführt, um leichter auf die Höhe und Breite eines Knotens zugreifen zu können. Für jeden Knoten $v \in V$ mit $M(v) = (h, b)$ gilt $\text{Höhe}(v) = h$ und $\text{Breite}(v) = b$.

DEFINITION 2.21 (ZEICHNUNG GEOMETRISCHER GRAPHEN)

Sei $G = (V, E, M)$ ein geometrischer Graph und $P = (P_V, P_E)$ eine Platzierung des Graphen. Für eine Zeichnung des geometrischen Graphen $Z(G, P)$ gilt:

1. Das zu einem Knoten $v \in V$ mit $M(v) = (h, b)$ und $P_V(v) = (x, y)$ gehörende Rechteck ist die Punktmenge $\text{Rechteck}(h, b, (x, y))$.
2. Der zu einer Kante $e \in E$ mit $P_E(e) = ((x_1, y_1), \dots, (x_n, y_n))$ gehörende Polygonzug ist die Punktmenge $\text{Polygonzug}(((x_1, y_1), \dots, (x_n, y_n)))$.
3. Polygonzüge beginnen und enden an Rechtecken. Für alle Kanten $(u, v) \in E$ mit $\text{Polygonzug}((u, v)) = ((x_1, y_1), \dots, (x_n, y_n))$, $M(u) = (h_u, b_u)$, $M(v) = (h_v, b_v)$, $P_V(u) = (x_u, y_u)$ und $P_V(v) = (x_v, y_v)$ gilt

$$\text{Rechteck}(h_u, b_u, (x_u, y_u)) \cap \text{Polygonzug}(((x_1, y_1), \dots, (x_n, y_n))) = \{(x_1, y_1)\}$$

$$\text{Rechteck}(h_v, b_v, (x_v, y_v)) \cap \text{Polygonzug}(((x_1, y_1), \dots, (x_n, y_n))) = \{(x_n, y_n)\}$$

Für einen Knoten $v \in V$ mit $M(v) = (h, b)$ und $P_V(v) = (x, y)$ liefert $x : V \rightarrow \mathbb{R}$ die x -Koordinate und $y : V \rightarrow \mathbb{R}$ die y -Koordinate des Knotens. Die obere und untere Begrenzung des Rechtecks $y_o, y_u : V \rightarrow \mathbb{R}$ ist gegeben durch $y_o(v) = y - \frac{h}{2}$ und $y_u(v) = y + \frac{h}{2}$, die linke und rechte Begrenzung $x_l, x_r : V \rightarrow \mathbb{R}$ durch $x_l = x - \frac{b}{2}$ und $x_r = x + \frac{b}{2}$.

Für einen geometrischen Graph mit einer Platzierung gibt es genau eine Zeichnung. Wenn im Folgenden von der *Berechnung einer Zeichnung* geredet wird, so ist damit die Berechnung der Platzierung gemeint, aus der sich dann die Zeichnung ergibt. Dieser Prozess wird auch *platzieren* oder *zeichnen* genannt.

Bei Zeichnungen von Graphen hat es sich eingebürgert, für einen Knoten v oder eine Kante e und deren Zeichnung dieselbe Terminologie zu verwenden. Man sagt *eine Kante e ist geradlinig* und meint *die Zeichnung der Kante, die geradlinig ist*. Die *Fläche einer Zeichnung* ist durch die Fläche des kleinsten die Zeichnung umschließenden Rechtecks gegeben.

2.2.3.1 Erzeugung der Zeichnungen

Zeichnen von Graphen ist eine Visualisierungstechnik. Zeichnungen von Graphen können entweder *manuell* oder *automatisch* erzeugt werden. Im ersten Fall bedeutet dies, dass ein Anwender die Objekte platziert, im zweiten Fall, dass eine Platzierung durch einen Algorithmus berechnet wird. Visualisierungen von Graphen sollen *schön* sein und das Verstehen der durch den Graph repräsentierten Information fördern. Dabei hängt die Bedeutung von *schön* vom Typ des Graphen, der aktuellen Anwendung und vom Geschmack des Anwenders ab, sie wird mittels Zeichenkonventionen und Ästhetikkriterien ausgedrückt (siehe den folgenden Abschnitt).

Automatisches Zeichnen von Graphen bietet den Vorteil, dass Anwender von der aufwendigen manuellen Platzierung der Elemente befreit werden. Zudem ist die Erzeugung der Zeichnung schneller als bei manuellen Visualisierungen. Ein weiterer Vorteil automatischer Verfahren ist, dass die Darstellungen einem gewissen Qualitätsstandard genügen, so werden z. B. planare Graphen durch passende Zeichenverfahren stets ohne Kantenkreuzungen dargestellt. Insgesamt ist die Qualität der Zeichnungen aber oft geringer als bei gut gemachten manuellen Zeichnungen. Das liegt daran, dass automatische Zeichenverfahren überwiegend nicht dafür entwickelt wurden, spezielle *Platzierungswünsche* des Anwenders oder einer Anwendung, z. B. die Anordnung der Knoten in einer bestimmten Reihenfolge, zu berücksichtigen.

2.2.3.2 Zeichenkonventionen und Ästhetikkriterien

Um aussagekräftige Zeichnungen von Graphen zu erhalten, müssen *Zeichenkonventionen* eingehalten werden. Zeichenkonventionen legen Mindestanforderungen an die Visualisierung fest. Diese betreffen die lokale Platzierung von Knoten und Kantenstützpunkten (z. B. Gitterzeichnungen), die Kantenführung (z. B. geradlinige Zeichnungen, orthogonale Zeichnungen) und globale Darstellungseigenschaften (z. B. planare Zeichnungen, Abwärtszeichnungen), siehe [DETT99]. Die Konventionen sind oft auf spezielle Anwendungen abgestimmt, beispielsweise werden für die Darstellung elektronischer Schaltungen orthogonale Zeichnungen verlangt.

Eine Darstellung muss die Zeichenkonventionen erfüllen, dies allein ist jedoch nicht ausreichend für schöne Zeichnungen. Dafür ist die Erfüllung weiterer Anforderungen, der so genannten *Ästhetikkriterien* erforderlich. Ästhetikkriterien beschreiben Ziele, deren Erreichen mit schönen Zeichnungen gleichgesetzt werden. Dabei lässt sich der Begriff *schön* nicht formal definieren, er ist subjektiv und hängt vom Anwender und dem Zweck der Visualisierung ab.

Wichtige Ästhetikkriterien sind z. B. die Minimierung der Zahl der Kreuzungen zwischen Kanten, die Minimierung der Fläche der Zeichnung bei vorgegebenem Mindestabstand zwischen den Knoten,

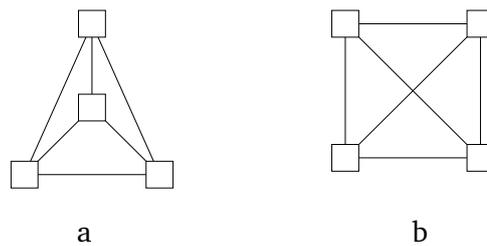


Abbildung 2.11: **Widersprüchliche Ästhetikkriterien.** Zwei Zeichnungen eines Graphen: (a) mit minimaler Zahl von Kreuzungen, (b) mit maximaler Symmetrie.

die Minimierung der Gesamtlänge aller Kanten, die Minimierung der Zahl der Kantenknickpunkte und die Maximierung der Symmetrien [DETT99].

Diesen Kriterien *sollen* die Zeichnungen genügen. Allerdings ist das nicht immer möglich, da sich Ästhetikkriterien widersprechen können. Beispielsweise zeigt Abbildung 2.11(a) eine Zeichnung eines Graphen mit minimaler Zahl von Kreuzungen, während in Abbildung 2.11(b) eine Zeichnung desselben Graphen mit maximaler Symmetrie dargestellt ist.

Während Zeichenkonventionen *Muss*-Kriterien sind, die bei einer Visualisierung des Graphen verpflichtend sind, handelt es sich bei Ästhetikkriterien um *Kann*-Kriterien, deren Erfüllung angestrebt wird, aber nicht unbedingt erforderlich ist. Da die Erfüllung bestimmter Ästhetikkriterien dazu führen kann, dass andere nicht mehr (oder nur noch unzureichend) erfüllt werden können, ist es sinnvoll, eine Reihenfolge der Ästhetikkriterien festzulegen. Purchase untersucht in [PCJ96, Pur97, Pur98] die Auswirkung verschiedener Kriterien auf die Verständlichkeit der Zeichnungen, um deren Wichtigkeit zu werten. So ist nach Einschätzung von Anwendern die Reduzierung von Kantenkreuzungen wichtiger als symmetrische Darstellung isomorpher Teilgraphen.

2.2.3.3 Klassifikation von Zeichenverfahren

Zeichenverfahren für Graphen können nach verschiedenen Gesichtspunkten klassifiziert werden: Sie können eingeteilt werden nach

1. *Klassen von Graphen*, deren Visualisierung damit möglich ist. Wichtige Graphklassen sind z. B. gerichtete azyklische Graphen (DAGs), planare Graphen, zwei- und dreifach zusammenhängende Graphen und Bäume.
2. *Zeichenkonventionen*, die die Zeichnungen erfüllen müssen. Entsprechend den eben erwähnten Zeichenkonventionen lassen sich verschiedene Klassen, beispielsweise orthogonale und ebenenweise Zeichnungen unterscheiden.
3. *Klassen von Verfahren*, die verwendet werden. Dazu gehören z. B. kräftebasierte und ebenenweise Verfahren.

Ausführliche Betrachtungen zu den einzelnen Klassen finden sich in [BJM97, CT99, DETT99, EM99]. Im Folgenden werden die Zeichenverfahren kurz vorgestellt, die prinzipiell für die Visualisierung biochemischer Reaktionsnetze geeignet sind. Die Begründung der Auswahl und eine Untersuchung der Verfahren finden sich in Abschnitt 4.2.2.

2.2.3.3.1 Ebenenweise Zeichenverfahren

Ebenenweise Zeichenverfahren dienen der Erzeugung von Polylinienzeichnungen gerichteter Graphen. Das Verfahren ordnet Knoten auf parallelen Ebenen an. Kanten verlaufen zwischen zwei oder mehreren Ebenen, jedoch nicht innerhalb einer Ebene. Das Verfahren umfasst vier Schritte:

1. Der Graph wird in einen DAG umgewandelt.
2. Die Knoten werden den Ebenen so zugeordnet, dass alle Kanten von oben nach unten verlaufen.

Lange Kanten, die mehr als eine Ebene voneinander entfernte Knoten verbinden, werden nun durch eine Kette kurzer Kanten und temporärer Knoten ersetzt. Im Ergebnis verbinden Kanten nur noch Knoten benachbarter Ebenen.
3. Die Knoten werden innerhalb der Ebenen so permutiert, dass die Zahl der Kantenkreuzungen gering ist.
4. Für die Knoten in den Ebenen werden Positionen berechnet, welche die vorher erzeugten Sortierungen (auf Grund der Kreuzungsreduzierung) erhalten, zugleich aber Knoten ausgewogen platzieren. Lange Kanten werden vertikal gezeichnet.

Diese Verfahren eignen sich besonders gut zur Darstellung von Hierarchien. Sugiyama, Tagawa und Toda [STT81], Gansner et al. [GNV88, GKNV93], Eades et al. [ES90, ELT96] und Sanders [San96b, San99] haben ebenenweise Zeichenverfahren entwickelt und studiert.

2.2.3.3.2 Kräftebasierte Zeichenverfahren

Kräftebasierte Verfahren dienen der Erzeugung geradliniger Zeichnungen von Graphen. Dabei werden physikalische Modelle angewendet: Knoten und Kanten des Graphen werden als Objekte gesehen, auf die Kräfte wirken. Es wird ein Kräftesystem auf dem Graphen simuliert und eine Platzierung der Knoten gesucht, so dass die wirkenden Kräfte minimal werden. Kräftebasierte Verfahren eignen sich besonders gut, um Symmetrien in Graphen hervorzuheben [EL00]. Eades [Ead84], Fruchterman und Reingold [FR91], Kamada und Kawai [KK89] und Sugiyama und Misue [SM95] haben sich mit kräftebasierten Zeichenverfahren beschäftigt.

2.2.3.3.3 Orthogonale Zeichenverfahren

Bei einer orthogonalen (Gitter-)Zeichnung sind die Knoten auf Gitterpunkten platziert. Jede Kante ist auf eine Folge horizontaler und vertikaler Linien abgebildet. Ein Verfahren zur Erzeugung orthogonaler Zeichnungen ist die so genannte *Topology-Shape-Metrics* Methode [BNT86, DETT99, Tam87]. Dieses Zeichenverfahren eignet sich gut zur Darstellung ungerichteter Graphen. Biedl et al. [BK94, BMT97], Fößmeier und Kaufmann [FK97] und Papakostas und Tollis [PT95, PT97] haben ebenfalls orthogonale Zeichenverfahren entwickelt und untersucht.

2.2.4 Komplexitätstheorie und Experimente

2.2.4.1 Komplexitätstheorie

Um Aussagen über die Komplexität von Problemen und über Laufzeiten von Algorithmen zu treffen, werden einige Begriffe aus der Komplexitätstheorie benötigt. Dafür sei auf [CLR90, HU90, Meh84b] verwiesen. Laufzeitabschätzungen werden in der so genannten *O-Notation* vorgenommen.

2.2.4.2 Experimente

In dieser Arbeit werden einige NP-vollständige Probleme betrachtet und verschiedene Heuristiken zu deren Lösung vorgestellt. In einigen Fällen wird es gelingen, Aussagen über die Qualität dieser Verfahren zu beweisen. Dies kann beispielsweise eine obere Schranke der Lösung im Bezug zur optimalen Lösung oder das Erreichen der optimalen Lösung für gewisse Eingabeklassen sein.

Oft lassen sich solche Aussagen jedoch nicht treffen oder die Schranken sind zu grob. Um Heuristiken dennoch bewerten zu können, werden Experimente durchgeführt. Dabei wird die Heuristik auf eine Menge von Eingabewerten angewandt, die gelieferten Ergebnisse werden dann nach verschiedenen Gesichtspunkten ausgewertet.

Um die Wiederholbarkeit der Experimente zu gewährleisten, soll im Folgenden die experimentelle Umgebung beschrieben werden. Dafür werden einige Definitionen benötigt:

DEFINITION 2.22 (KARDINALITÄTSGLEICHE GRAPHEN)

Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen kardinalitätsgleich, wenn $|V_1| = |V_2|$ und $|E_1| = |E_2|$ ist.

DEFINITION 2.23 (ISOMORPHE GRAPHEN)

Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen isomorph, wenn es eine bijektive Abbildung $f: V_1 \rightarrow V_2$ gibt, so dass für alle Knoten $u, v \in V$ gilt $(u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2$.

DEFINITION 2.24 (DICHTE EINES GRAPHEN)

Die Dichte δ eines Graphen $G = (V, E)$ ist

$$\delta = \frac{|E|}{e_{\text{voll}}}, \text{ wobei } e_{\text{voll}} = \begin{cases} |V_1| * |V_2| & \text{falls } G = (V_1 \cup V_2, E) \text{ bipartit} \\ \frac{|V| * (|V| - 1)}{2} & \text{sonst} \end{cases}$$

Dabei ist e_{voll} die Zahl der Kanten eines vollständigen Graphen ohne Schlingen und mit nur einer Kante zwischen jedem Knotenpaar $u, v \in V$. Ein Graph heißt *licht*, wenn er, verglichen mit der Knotenzahl, relativ wenige Kanten enthält. Er heißt *dicht*, falls er verhältnismäßig viele Kanten enthält. Dabei gibt es keinen exakten Schwellwert, ab dem ein Graph als licht oder dicht gesehen wird. Ein Graph mit $|E| < c * |V|$ wird als licht und mit $|E| > \frac{1}{c} * |V|^2$ als dicht betrachtet, wobei c im Allgemeinen ein kleiner Wert, typischerweise 2 oder 3, ist.

Um statistisch relevante Aussagen über die Heuristik zu erhalten, werden Experimente so durchgeführt, dass nicht einzelne Graphen, sondern für jeden zu untersuchenden Graphen eine Menge kardinalitätsgleicher, zufälliger Graphen untersucht wird. Wird statt dessen eine Menge isomorpher Graphen verwendet, so lassen sich Aussagen über die Abhängigkeit des Ergebnisses von der Eingabereihenfolge treffen.¹⁴

¹⁴Siehe [GBS98, SBG99], hier finden sich auch weitere Betrachtungen zu Experimenten mit Graphalgorithmen.

Betrachtet man nicht nur das Ergebnis einer Heuristik, sondern auch die zum Erreichen des Ergebnisses notwendige Rechenzeit, sind weitere Faktoren zu beachten:

1. Die experimentelle Umgebung, also Prozessor (Art, Taktfrequenz, Cache usw.), Speicher (Größe, Zugriffszeit), weitere Hardwareaspekte und die Systembelastung.
2. Das Betriebssystem, die Programmiersprache und der Compiler (Hersteller, Version, Optimierungsstufe) sowie die bei der Implementierung benutzten Bibliotheken.
3. Der Programmierer (Erfahrung).

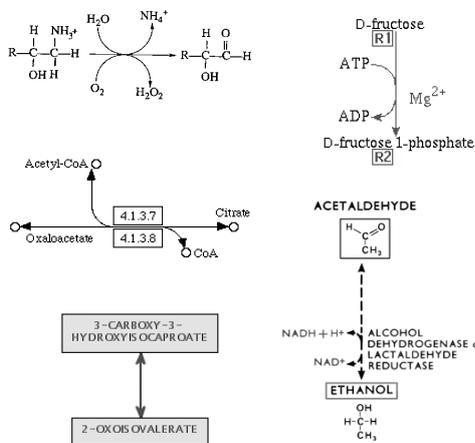
Um vergleichbare Aussagen zu erhalten, müssen Experimente auf identischen Systemen durchgeführt werden und die Algorithmen sollten möglichst von derselben Person innerhalb kurzer Zeit¹⁵ implementiert worden sein.

Die in der Arbeit untersuchten Algorithmen wurden alle unter denselben Bedingungen getestet. Die wesentlichen Daten sind: 450 MHz Pentium III mit 256 MB Hauptspeicher, 100MHz interner Bus, keine zusätzliche Belastung durch weitere Programme außer Betriebssystem Windows2000. Die Algorithmen wurden in C++ mit VisualC++ 6.0 unter Verwendung der Bibliotheken GTL [GTL01], Graphlet [Gra01a, Him97] und STL [MS96, STL00] erstellt. Zudem wurden alle Algorithmen durch den Autor implementiert.

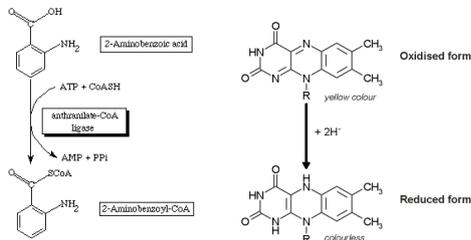
Die verwendeten Graphen wurden durch die Funktionen `RANDOM_GRAPH` bzw. `RANDOM_BIGRAPH` der LEDA-Bibliothek [Näh93, LED00] erzeugt. Für jeden Graphentyp (z.B. ein Graph mit 10 Knoten und 20 Kanten) wurden 100 kardinalitätsgleiche, zufällige Graphen erzeugt. Da `RANDOM_BIGRAPH` in den bipartiten Graphen parallele Kanten erzeugen kann, wurden in diesen Graphen alle parallelen Kanten beseitigt, indem deren Endknoten zufällig auf neue Knoten umgesetzt wurden.

¹⁵Es ist anzunehmen, dass die Person ihr Wissen und ihre Fähigkeiten mit der Zeit verbessert, was bereits Auswirkungen auf die Implementierung verschiedener Verfahren haben kann.

3 Darstellungsanforderungen



- 3.1 Einführung 38
- 3.2 Anforderung I: Darstellung von Reaktionen 40
- 3.3 Anforderung II: Darstellung von Reaktionswegen und Reaktionsnetzen 46
- 3.4 Anforderung III: Kontexterhaltende Navigation und Sichten 50
- 3.5 Zusammenfassung der Anforderungen 58



Darstellungen biochemischer Reaktionen aus unterschiedlichen Quellen.

3.1 Einführung

3.1.1 Ziele

Bisher gibt es keine Untersuchungen zu Anforderungen an die Visualisierung biochemischer Reaktionsnetze. In [Mic98a,Mic98b] und [KP94a] geben die Autoren zwar einige Hinweise zur Gestaltung der Darstellungen, diese sind jedoch durch den persönlichen Geschmack der Autoren bestimmt und zu allgemein gehalten, um daraus Darstellungsanforderungen ableiten zu können. So stellt Michal in [Mic98a] zu Zeichnungen biochemischer Reaktionswege fest:

„Style is a very personal thing. The way biochemical facts are presented in the various textbooks, articles in magazines, etc. differ greatly. This refers to, for example, the use of color, the way compounds are shown and their arrangement, the way reaction arrows are drawn, etc. (...) I personally prefer a relative austere way of presentation, where each element has a strictly defined purpose (e. g. arrow colors: occurrence of the reaction; solid arrows: anabolism, dashed arrows: catabolism, bold arrows: major pathways, points on both ends of the arrow: reversible reaction). However, one can find in various publications the complex network of metabolism shown well in different fashions.“

In diesem Kapitel werden deshalb die Wünsche und Forderungen an die Darstellung biochemischer Reaktionsnetze analysiert. Es wird eine Charakterisierung gegeben, die wesentliche Merkmale der Darstellung festlegt, zugleich aber das nötige Maß an Flexibilität zur Erfüllung anwendergegebener Darstellungswünsche bietet. Die dabei herausgearbeiteten Anforderungen orientieren sich an folgenden Zielen:

1. Berücksichtigung traditioneller Darstellungen

Die traditionellen Arten der Darstellung biochemischer Reaktionsnetze müssen berücksichtigt werden. Dies ist für die Akzeptanz der Visualisierung beim Anwender entscheidend. Oberstes Ziel muss es sein, den Biochemikern Darstellungen zur Verfügung zu stellen, in denen sie sich sofort zurechtfinden. Erfordern Visualisierungen vom Anwender erst ein Umdenken bzw. Umlernen, um sich in der Darstellung zurechtzufinden, so werden diese in der Praxis im Allgemeinen nicht akzeptiert. Anwender sind nicht bereit oder haben in ihrer täglichen Arbeit keine Zeit, sich auf neue, von den herkömmlichen Darstellungskonventionen abweichende Visualisierungen einzustellen.

2. Nutzung der Möglichkeiten automatischer Visualisierungen

Die Anforderungen müssen so gestaltet sein, dass neue Möglichkeiten einer automatischen Visualisierung ausgenutzt, Beschränkungen berücksichtigt und möglichst beseitigt werden. Die Möglichkeiten automatischer Visualisierungsverfahren können weit über die hergebrachten Darstellungsmöglichkeiten hinausgehen, indem z. B. neue Sichten oder Möglichkeiten der Navigation durch biochemische Reaktionsnetze angeboten werden. Andererseits müssen die Anforderungen Beschränkungen durch neue Arten der Darstellung berücksichtigen. Während beispielsweise in Büchern mit ausklappbaren oder losen Karten leicht die Darstellungsfläche vergrößert werden kann, ist diese in elektronischen Systemen durch Größe und Auflösung der Ausgabegeräte (im Allgemeinen des Monitors) beschränkt. Die Techniken des Zoomens oder Scrollens sind hier unzureichend, da der Vorteil einer großen Karte *gleichzeitiger* Überblick und Detaildarstellung ist, während beim Scrollen der Überblick, beim Zoomen die Detaildarstellung verloren geht.

3. Technische Realisierbarkeit

Die Anforderungen sollen so gestaltet sein, dass sie technisch realisierbar sind. Sie sollen ein universelles Verfahren ermöglichen, mit dem alle wesentlichen Nutzerwünsche abgedeckt werden, dass zugleich aber auch einfach zu verstehen, implementieren und warten ist.

Ziel muss also eine *Vereinheitlichung der Struktur der Darstellung* sein. Es ist nicht Ziel, durch die Anforderungen jede bereits in Büchern existierende Darstellung exakt zu beschreiben.

3.1.2 Grundlage der Anforderungen

Basis der hier aufgestellten Darstellungsanforderungen sind Befragungen, Diskussionsrunden sowie Untersuchungen zur Akzeptanz verschiedener Darstellungsformen mit Biochemikern und Biologen am Max-Planck-Institut für Biochemie in Martinsried (München), an den Universitäten Erlangen und Passau sowie im Rahmen der Workshops *Modelling and Simulation of Gene and Cell Regulation and Metabolic Pathways* [CHMM98] und *Molekulare Bioinformatik* [Hof98], die Analyse existierender Zeichnungen sowie die Arbeiten [Mic98a, Mic98b].

Die wichtigsten Resultate der erwähnten Gespräche und Diskussionen sind:

1. Die Visualisierungen müssen leicht als biochemische Reaktionsnetze erkennbar sein und sich an traditionellen Darstellungen orientieren.
2. Die zeitliche Aufeinanderfolge von Reaktionen ist wichtig.
3. Überwiegend werden Reaktionswege und -netze zwischen zwei Substanzen betrachtet, selten ist man an der Darstellung des gesamten Stoffwechsels interessiert und wenn, dann nur auf einer Übersichtsebene.
4. Verschiedene Sichten erleichtern das Verständnis von Reaktionswegen und -netzen und sollen unterstützt werden.
5. Zeichnungen sollen ähnlich zueinander sein: *lokal* ähnlich, indem alle Reaktionen nach demselben Schema dargestellt werden und *global* ähnlich, indem die Zeichnung gleicher Reaktionswege in verschiedenen Darstellungen möglichst gleich ist.

Immer wieder wurde von den Anwendern betont, dass sie Verfahren zur automatischen Visualisierung von biochemischen Reaktionsnetzen als notwendig erachten. Da sich herausstellte, dass von Anwendern die Möglichkeiten und Grenzen automatischer Zeichenverfahren nur ungenügend eingeschätzt werden konnten, bestand ein Teil der Untersuchungen darin, ihnen mit herkömmlichen Zeichenverfahren erzeugte Darstellungen zu präsentieren und deren Akzeptanz zu untersuchen. Die Ergebnisse dazu werden im Abschnitt 4.2.2 dargestellt.

Die Abbildungen 3.2, 3.4 und 3.7 zeigen Ergebnisse der Untersuchung existierender Darstellungen aus Büchern und Systemen für die Biochemie. Dazu wurden insgesamt 175 Abbildungen aus Standardwerken der Biologie und Biochemie [ABL⁺97, Cam98, FM99, LNC94, Mic99, Mic93, Str95] und aus frei über das Internet zugänglichen Systemen [Exp00, KEG00, UMB00, WIT00] ausgewertet. Die Ergebnisse dieser Analyse werden in den folgenden Abschnitten diskutiert.

In [Mic98b] untersucht Michal Gesichtspunkte, die bei biologischen Prozessen eine herausragende Rolle spielen und in der Darstellung berücksichtigt werden sollen. Er stellt dabei grafische Darstellungen in den Mittelpunkt. Michal fasst die Betrachtung der Stoffwechselwege weiter als dies hier geschieht und bezieht beispielsweise auch Regulation, Stöchiometrie und Energetik der Reaktionen

mit ein.¹ Er nennt Informationen, die bei biologischen Prozessen eine herausragende Rolle spielen und geeignet dargestellt werden müssen:

1. Reaktionssequenz mit Namen, Strukturformeln und weiteren Charakterisierungen der Komponenten und der Reaktion
2. Enzyme mit Namen, EC-Nummer und enzym- sowie reaktionsspezifischen Informationen
3. Coenzyme und Cofaktoren
4. Regulation mit Aktivatoren, Inhibitoren und regulationsspezifischen Informationen
5. Lokalisation

Konventionen für die Darstellung dieser Informationen werden dagegen nur sehr allgemein behandelt.

In den folgenden Abschnitten werden Anforderungen an die Darstellung biochemischer Reaktionsnetze formuliert. Die Anforderungen bauen aufeinander auf. Zuerst werden Anforderungen an die Visualisierung einzelner Reaktionskomponenten und Reaktionen dargelegt, anschließend werden die so erhaltenen Reaktionen zu Reaktionsnetzen zusammengefasst und zusätzliche Anforderungen an die Visualisierung der Reaktionsnetze formuliert. Im Abschnitt 3.4 werden Anforderungen an kontexterhaltende Darstellungen und Sichten betrachtet.

3.2 Anforderung I: Darstellung von Reaktionen

Die Darstellung einer biochemischen Reaktion lässt sich unterteilen in die Visualisierung ihrer Komponenten, also der zugehörigen Substanzen, Cosubstanzen und Enzyme und die Visualisierung der Reaktion selbst, also die Anordnung der Komponenten im Bild und die Darstellung des Reaktionspfeils.

3.2.1 Substanzen, Cosubstanzen und Enzyme

ANFORDERUNG 3.1 (DARSTELLUNG VON SUBSTANZEN, COSUBSTANZEN UND ENZYMEN)

Eine Substanz, Cosubstanz oder ein Enzym werden dargestellt durch einen Text und ein optionales Bild, welches über oder unter dem Text angeordnet ist.

Der *Text* ist bei der Darstellung einer Substanz bzw. Cosubstanz ein Name der (Co-)Substanz. (Co-)Substanzen besitzen oft mehrere Namen: generische, Trivial- und abgekürzte Namen². In Fällen, in denen mehrere Namen existieren, wird vom Anwender bzw. der Anwendung einer der Namen ausgewählt. Oft werden für Substanzen generische Namen und für Cosubstanzen Abkürzungen verwendet.

Der *Text* ist bei der Darstellung eines Enzyms ein Name des Enzyms oder dessen EC-Klassifikationsnummer.

¹Michal ist an der Darstellung *aller* Prozesse in Zellen interessiert, so enthalten [Mic93, Mic99] auch Darstellungen des genetischen Codes sowie von Transport- und Signalprozessen.

²Beispielsweise hat die Substanz *Glucose* den Trivialnamen *Traubenzucker* und die Abkürzung *Glc*.

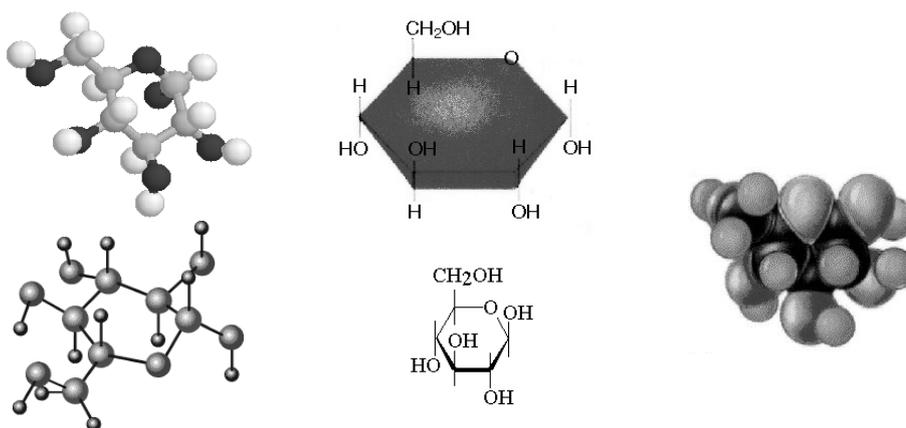


Abbildung 3.1: **Darstellung von Molekülen.** Beispiele für Moleküldarstellungen der Substanz *Glucose*, z.B. (links) Kugel-Stab-Modelle, (mitte) vereinfachte Strukturformeln und (rechts) Kalottenmodell.

Das *Bild* besteht sowohl bei (Co-)Substanzen wie bei Enzymen aus deren (vereinfachter) Strukturformel. Wird es vom Anwender festgelegt, so kann das Bild jede andere Darstellung sein, beispielsweise Kugel-Stab-Modelle³ oder Kalottendarstellungen⁴. Die Abbildung 3.1 zeigt einige Bilder für Moleküle. Das Bild ist optional; für Cosubstanzen und Enzyme wird es oft nicht dargestellt.

3.2.1.1 Begründung der Anforderung

Die Darstellung einer (Co-)Substanz bzw. eines Enzyms in der Visualisierung eines Reaktionsnetzes hängt von verschiedenen Faktoren ab:

1. Sie wird bestimmt durch die aktuelle *Anwendung*, für die eine Visualisierung benötigt wird. So sollen in Übersichtsbildern Substanzen nur mit ihrem Namen dargestellt werden, da nur die Struktur des Reaktionsnetzes interessiert, nicht aber die Feinheiten jeder einzelnen Reaktion. In detaillierten Abbildungen ist dagegen neben dem Namen auch die Darstellung der Strukturformel der Substanz gefordert, da hier oft das Aufbrechen und Neugestalten chemischer Bindungen im Mittelpunkt steht. Diese lassen sich aus einer bildlichen Darstellung der Substanz (mittels Strukturformel) leicht ablesen.
2. Sie hängt ab von der *verfügbaren Information*. Die Struktur vieler Enzyme ist bisher nicht bekannt, entsprechend lassen sich für diese keine Strukturformeln darstellen. Die Visualisierung muss sich dann auf textuelle Information beschränken.
3. Der *Nutzer* möchte oft selbst festlegen können, auf welche Art Substanzen und Enzyme dargestellt werden.

³Kugel-Stab-Modelle geben die dreidimensionale Struktur von Molekülen wieder. Dabei werden Atome als Kugeln, Atombindungen als Stäbe zwischen den Kugeln dargestellt. Das Kugel-Stab-Modell betont die Bindungswinkel des Moleküls.

⁴Kalottenmodelle geben ebenfalls die dreidimensionale Struktur von Molekülen wieder, sie stellen die räumliche Anordnung der Atome dar. Das Kalottenmodell betont die Gestalt eines Moleküls.

		[LNC94]	[Cam98]	[FM99]	[Mic99]	[Mic93,ExP00]	[Str95]	[ABL ⁺ 97]	[KEG00]	[WIT00]	[UMB00]	% der Darstellungen
	Zahl untersuchter Abbildungen	20	5 ⁵	20	20	20 ⁶	20	10 ⁵	20	20	20	100
Substanz	Name	5	1	4	4	0	4	4	20	20	0	35
	Summenformel	0	0	1	0	0	0	0	0	0	0	1
	Strukturformel ⁷	0	0	3	0	0	0	0	0	0	0	2
	Name + Summenformel	0	0	4	0	0	0	0	0	0	0	2
	Name + Strukturformel ⁷	15	4	8	16	20	16	6	0	0	20	60
Cosubstanz	Name	1	0	2	0	2	0	0	0	4	0	5
	Abkürzung/Summenformel ⁸	16	4	12	15	17	14	8	2	16	15	68
	Strukturformel ⁷	0	0	0	1	1	1	0	0	0	0	2
	keine Darstellung	3	1	6	4	0	5	2	18	0	5	25
Enzym	Name	12	3	5	18	20	11	0	0	0	15	48
	EC-Nummer	0	0	0	0	0	0	0	20	20	0	23
	Name + EC-Nummer	0	0	3	0	0	0	0	0	0	0	2
	Abkürzung/Verweis ⁹	2	1	0	0	0	2	3	0	0	1	5
	Name + Strukturformel ⁷	1	0	1	0	0	0	0	0	0	0	1
	keine Darstellung	5	1	11	2	0	7	7	0	0	4	21

Abbildung 3.2: **Analyse herkömmlicher Darstellungen (Teil 1)**. Untersuchung von Darstellungen aus verschiedenen Quellen (siehe oberste Zeile). Um alle betrachteten Bücher und Systeme möglichst gleich zu werten, wurden maximal 20 Abbildungen von Reaktionsnetzen zufällig ausgewählt, wobei diese mindestens zwei Reaktionsschritte umfassen mussten.

Die Einträge einer Spalte seien am Beispiel [LNC94] erklärt: Es wurden 20 Abbildungen von Reaktionsnetzen betrachtet. In 5 Darstellungen wurden Substanzen überwiegend durch ihre Namen, in 15 Abbildungen durch Namen und Strukturformeln dargestellt. In 16 der 20 Abbildungen wurden Cosubstanzen mittels Angabe der Abkürzung (z. B. ATP) oder Summenformel (z. B. CO₂) gezeigt, in einer wurden die Namen der Cosubstanzen ausgeschrieben und in drei Fällen wurden Cosubstanzen nicht visualisiert. Die Einträge für Enzyme ergeben sich analog.

⁵Die Bücher *Biologie* [Cam98] und *Molekularbiologie der Zelle* [ABL⁺97] zählen zu den Standardwerken der Biologie. Aus diesem Grund wurden sie mit in die Untersuchung aufgenommen, obwohl sie nur wenige Darstellungen biochemischer Reaktionsnetze enthalten.

⁶ExPASy [ExP00] ist eine elektronische Version von [Mic93]. Dabei handelt es sich in beiden Fällen um ein sehr großes Bild. Hier wurden 20 der benannten Reaktionswege aus dem Netz herausgelöst und untersucht.

⁷Dies umfasst auch vereinfachte Strukturformeln.

⁸Oft werden für einfache Moleküle wie Wasser und Kohlendioxid Summenformeln, für komplexere Abkürzungen oder Namen verwendet. Falls Abkürzungen oder Summenformeln überwiegen, wurde die Darstellung dieser Zeile zugeordnet, sonst der Zeile *Name*.

⁹So werden beispielsweise Verweise auf zugehörige Textstellen statt der Namen der Enzyme verwendet.

Ein Visualisierungssystem muss also die Möglichkeit bieten, wahlweise unterschiedliche Information für Substanzen und Enzyme darzustellen. Dies entspricht sowohl den Forderungen von Michal wie auch den hergebrachten Darstellungen. Deutlich zeigt sich in Abbildung 3.2 die große Bedeutung der Verwendung von Namen und Strukturformeln für Substanzen, die, zum überwiegenden Teil sogar gleichzeitig, in 95 % der untersuchten Zeichnungen verwendet werden. Für Enzyme und Cosubstanzen werden stärker rein textuelle Darstellungen bevorzugt, aber auch hier ist die Darstellung von Bildern nicht auszuschließen.

3.2.2 Reaktionen

ANFORDERUNG 3.2 (DARSTELLUNG VON REAKTIONEN)

Eine Reaktion wird dargestellt durch die Visualisierung der zur Reaktion gehörenden Substanzen sowie wahlweise der zugehörigen Cosubstanzen und Enzyme entsprechend Anforderung 3.1 und die Darstellung eines verbindenden Reaktionspfeils, der mehrere Enden besitzen kann. Eine Reaktionsdarstellung muss dabei folgenden weiteren Regeln genügen:

- 1. Edukte und Produkte der Reaktion befinden sich an gegenüberliegenden Enden des Reaktionspfeils. Kanten von Edukten vereinigen sich zu einem gemeinsamen Reaktionspfeil, der sich später in Kanten zu Produkten aufspaltet. Die Richtung von den Edukten zu den Produkten wird als Richtung des Reaktionspfeils bezeichnet.*
- 2. Enzyme und Cosubstanzen werden entlang des gemeinsamen Reaktionspfeils neben diesen platziert, Cosubstanzen in der Reihenfolge ihres Eintritts in die bzw. Austritts aus der Reaktion. Dabei werden Cosubstanzen auf einer Seite, Enzyme auf der anderen Seite des gemeinsamen Reaktionspfeils dargestellt.*
- 3. Kantenstücke, die Substanzen mit dem Mittelstück der Reaktionskante verbinden, werden geradlinig dargestellt. Kantenstücke, die Cosubstanzen mit dem Mittelstück der Reaktionskante verbinden, werden als Kreisbögen oder Kurven dargestellt.¹⁰*
- 4. Alle Substanzen, alle Cosubstanzen und alle Enzyme werden jeweils gleichartig dargestellt.¹¹*

Die *Gestalt des Reaktionspfeils* ist charakterisiert durch seine Farbe, die Art der Linie (gestrichelt, durchgezogen) sowie durch die Dicke der Kantenstücke (die zu Substanzen führenden Kantenstücke können dicker dargestellt werden als solche zu Cosubstanzen). Sofern vom Anwender nicht anders festgelegt, dient die Farbe des Reaktionspfeils der Kodierung des Organismus bzw. der Klasse von Organismen, in denen die Reaktion vorkommt. Die Dicke wird zur Repräsentation der Wichtigkeit oder des Stoffumsatzes einer Reaktion verwendet und die Art der Linie zur Unterscheidung der Reaktionsarten, z. B. katabole und anabole Reaktionen.

Viele Reaktionen haben eine *Richtung*, in ihrer Zeichnung soll diese Richtung dargestellt werden. Reaktionen werden von oben nach unten oder von links nach rechts gezeichnet. Die Anforderung, Substanzen, Cosubstanzen und Enzyme jeweils mit einer klassenspezifischen Art zu visualisieren, ist darauf beschränkt, dass der Anwender keine andere Festlegung trifft und die vorhandenen Informationen eine solche Darstellung ermöglichen.

Abbildung 3.3 zeigt einige Darstellungen von Reaktionen.

¹⁰Solche Kanten sind z.B. in Abbildungen 3.3 dargestellt.

¹¹Im Allgemeinen werden die Substanzen der Reaktion mit vereinfachten Strukturformeln und Namen, die Cosubstanzen mit Abkürzungen und die Enzyme mit Namen dargestellt.

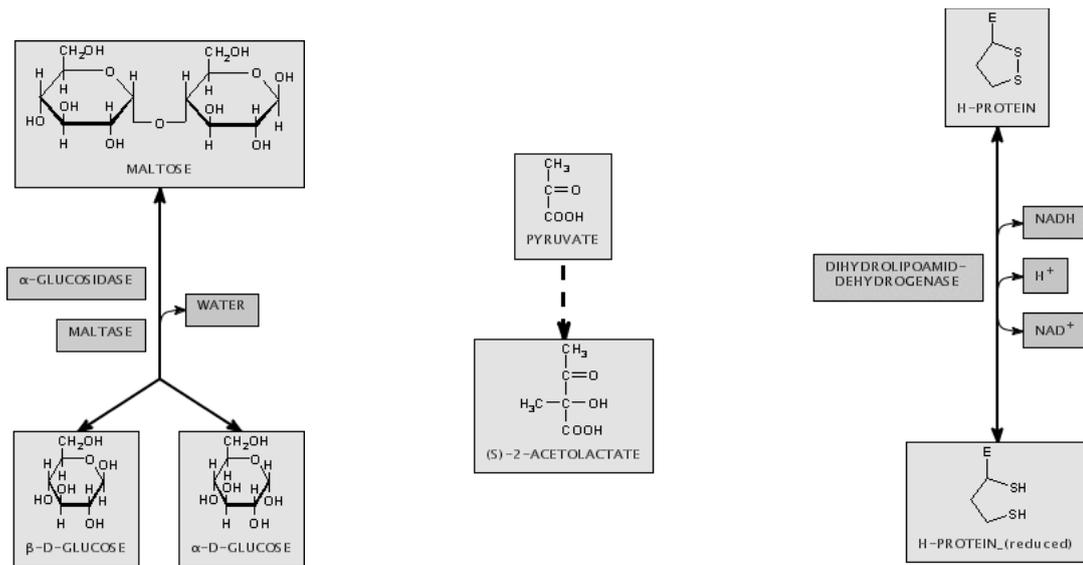


Abbildung 3.3: **Darstellung von Reaktionen.** Beispiele für die Darstellung von Reaktionen.

3.2.2.1 Begründung der Anforderung

Wie die Darstellung der Komponenten einer Reaktion hängt auch die Darstellung der Reaktion selbst wesentlich von drei Aspekten ab:

1. Die aktuelle *Anwendung* bestimmt, welche Komponenten dargestellt und wie der Reaktionspfeil visualisiert werden soll. So sollen in Übersichtsbildern nur die Substanzen dargestellt werden, in detaillierten Darstellungen dagegen auch die Cosubstanzen und Enzyme. In der Visualisierung der Stoffwechselwege eines Organismus werden Reaktionspfeile schwarz dargestellt. Werden dagegen in einer Anwendung Stoffwechselwege in verschiedenen Organismen untersucht, so dienen unterschiedliche Farben der Unterscheidung der Reaktionswege.
2. Auch bei der Visualisierung der Reaktionen bestimmt die *verfügbare Information*, welche Komponenten dargestellt und wie Reaktionspfeile gezeichnet werden. So ist das Vorkommen der Reaktion in speziellen Organismen oft nicht bekannt und es gibt Reaktionen, bei denen man nicht alle Komponenten kennt. Für diese Reaktionen muss sich die Darstellung auf die bekannten Aspekte beschränken.
3. Der *Nutzer* möchte selbst festlegen können, in welcher Art Reaktionen dargestellt werden. So kann eine nutzerspezifische Anforderung sein, ausgewählte Reaktionen mit Enzymen und Cosubstanzen, andere Reaktionen in derselben Abbildung ohne Cosubstanzen darzustellen.

Ein System für die Visualisierung von Reaktionen muss also Möglichkeiten bieten, Reaktionen auf verschiedene Art darzustellen, dabei aber die Grundstruktur solcher Reaktionen erhalten. Dies entspricht den Forderungen Michals nach Darstellung reaktionsspezifischer Information (z. B. verschiedene Arten von Linien für anabole, katabole und amphibole Reaktionen) und Lokalisation (z. B. durch verschiedene Farben) wie auch den hergebrachten Darstellungen.

		[LNC94]	[Cam98]	[FM99]	[Mic99]	[Mic93,Exp00]	[Str95]	[ABL*97]	[KEG00]	[WIT00]	[UMB00]	% der Darstellungen
	Zahl untersuchter Abbildungen	20	5	20	20	20	20	10	20	20	20	100
Pfeilverlauf (Verbindungen zwischen Komponenten und der Reaktion) ¹²	Substanz-Reaktion und Cosubstanz-Reaktion geradlinig ¹³	1	2	4	3	0	5	1	16	0	5	21
	Substanz-Reaktion geradlinig, Cosubstanz-Reaktion Kreisbogen/Kurve	15	2	12	15	17	12	8	2	20	15	67
	Substanz-Reaktion und Cosubstanz-Reaktion Kreisbogen/Kurve ¹⁴	4	1	4	2	3	3	1	2	0	0	11
Anordnung der Komponenten ¹²	Edukte und Produkte an verschiedenen Enden	20	5	20	20	20	20	9	16	20	20	97
	Cosubstanzen neben Reaktionspfeil	17	3	14	16	20	15	8	2	20	15	74 (99 ¹⁵)
	Enzyme neben (auf) Reaktionspfeil ¹⁶	15	4	9	17	20	13	3	20	20	16	78 (99 ¹⁷)
	Reaktionsrichtung ¹⁸	18	3	17	19	9	20	8	18	20	20	87
	überwiegend Doppelpfeile/Pfeilspitzen an beiden Enden	7	1	3	12	14	5	0	15	11	0	39
Pfeilgestaltung	Verwendung von Farbe/Form	6	2	0	16	20	0	1	0	0	0	26

Abbildung 3.4: Analyse herkömmlicher Darstellungen (Teil 2). Siehe Bemerkungen und Fußnoten zu Abbildung 3.2.

¹²Zu Pfeilverlauf und Anordnung der Komponenten siehe Abbildung 3.5.

¹³Geradlinige Darstellungen enthalten fast nie Cosubstanzen, die 21 % auf diese Weise dargestellten Reaktionen gehören überwiegend zu den Reaktionsdarstellungen ohne Cosubstanzen.

¹⁴Der überwiegende Teil der so dargestellten Reaktionen sind offene und geschlossene Zyklen.

¹⁵Entspricht 99 % der Reaktionen mit Cosubstanzen.

¹⁶In KEGG werden einzelne Enzyme einer Reaktion auf dem Reaktionspfeil platziert, mehrere Enzyme rechts und links neben dem Reaktionspfeil.

¹⁷Entspricht 99 % der Reaktionen mit Enzymen.

¹⁸Aus der Darstellung oder einer zusätzlichen Markierung wird die Hauptrichtung der Reaktion leicht ersichtlich.

In der Tabelle in Abbildung 3.4 sind die Untersuchungsergebnisse zum Verlauf des Reaktionspfeils, der Anordnung der Komponenten und der Gestaltung des Pfeils in den untersuchten Zeichnungen zusammengefasst. Wenn Cosubstanzen oder Enzyme dargestellt werden, was in vielen Zeichnungen geschieht, so werden sie in 99 % der Visualisierungen neben dem Reaktionspfeil platziert. Zudem sind Verbindungen zwischen Substanzen und Reaktionen überwiegend geradlinig gezeichnet (Ausnahmen sind die Darstellungen offener und geschlossener Zyklen mittels Kreis oder Spirale). Verbindungen zu Cosubstanzen sind fast immer mittels Kreisbögen dargestellt.

3.3 Anforderung II: Darstellung von Reaktionswegen und Reaktionsnetzen

3.3.1 Allgemeine Reaktionswege und Reaktionsnetze

ANFORDERUNG 3.3 (DARSTELLUNG VON REAKTIONSWEGEN UND REAKTIONSNETZEN)

Reaktionswege und Reaktionsnetze werden dargestellt durch die Visualisierung der zugehörigen Reaktionen entsprechend Anforderung 3.2. Die Darstellung muss dabei folgenden weiteren Regeln genügen:

1. Die Substanzen werden entsprechend der durch die Reaktionen gegebenen zeitlichen Reihenfolge ebenenweise angeordnet. Die parallelen Ebenen verlaufen dabei von oben nach unten oder von links nach rechts, Substanzen, die zeitlich vor anderen Substanzen umgesetzt werden, werden auf Ebenen weiter oben bzw. weiter links platziert. Möglichst viele Reaktionen werden so angeordnet, dass sie von oben nach unten bzw. links nach rechts und nur wenige in der entgegengesetzten Richtung verlaufen.
2. Alle Substanzen, alle Cosubstanzen und alle Enzyme werden jeweils gleichartig dargestellt.
3. Für alle Reaktionen einheitlich werden in Richtung des Reaktionspfeils die Cosubstanzen auf einer, die Enzyme auf der anderen Seite des Reaktionspfeils platziert.
4. Die Reaktionskanten sind kurz, Edukte und Produkte einer Reaktion sind zentriert zum zugehörigen Reaktionspfeil platziert. Reaktionskanten kreuzen sich möglichst wenig; Substanzen, Cosubstanzen und Enzyme überdecken sich nicht, zwischen ihnen sind Mindestabstände eingehalten. Die Fläche der Zeichnung des Reaktionswegs bzw. -netzes ist möglichst klein.

3.3.2 Geschlossene und offene Zyklen

In Abschnitt 2.1.2.2 wurden offene und geschlossene Zyklen als Reaktionswege mit besonderer Struktur eingeführt. Diese Struktur soll auch in der Visualisierung sichtbar sein, sie erfordert eine von der Visualisierung allgemeiner Reaktionsnetze abweichende Darstellung.

ANFORDERUNG 3.4 (DARSTELLUNG VON GESCHLOSSENEN UND OFFENEN ZYKLEN)

Bei geschlossenen Zyklen wird der erste Teil des Reaktionswegs ebenenweise von oben nach unten (bzw. links nach rechts) angeordnet, der zweite Teil des Reaktionswegs in umgekehrter Richtung von unten nach oben (rechts nach links). Beide Teile überkreuzen sich dabei nicht.

Bei offenen Zyklen werden die sich wiederholenden Teile des Reaktionswegs von links nach rechts nebeneinander dargestellt, gleichartige Reaktionen, gleichartige Edukte und gleichartige Produkte

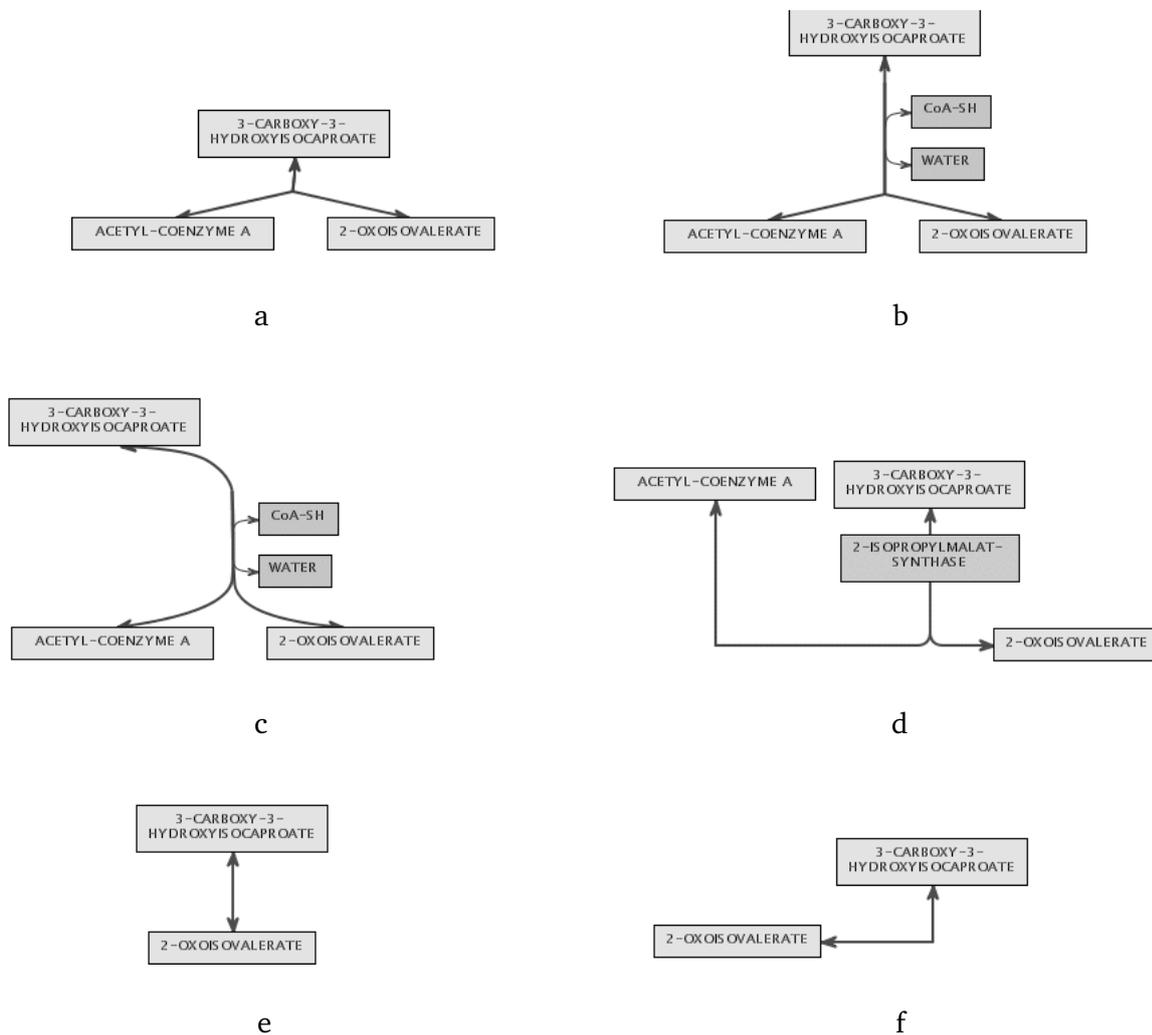


Abbildung 3.5: **Darstellung des Reaktionspfeils und Komponentenordnung.** Pfeilverlauf: (a) geradlinige Zeichnung, wie sie oft bei Darstellungen ohne Cosubstanzen auftritt (geradlinig bezieht sich auf die Verbindung Substanz-Reaktion), (b) Substanz-Reaktion geradlinig, Cosubstanz-Reaktion mittels Kreisbogen oder Kurven, (c) Substanz-Reaktion und Cosubstanz-Reaktion mittels Kreisbogen oder Kurven.

Anordnung der Komponenten: Edukte und Produkte (b) deutlich getrennt an verschiedenen Enden des Reaktionspfeils (die Trennung ist trotz Darstellung von Hin- und Rückrichtung der Reaktion leicht erkennbar) und (d) kaum getrennt (eine Darstellung, wie sie z. B. in KEGG auftritt, die Zuordnung zu Edukten und Produkten ist schwierig). (e) Darstellung einer deutlichen Haupt-Reaktionsrichtung (diese ist oft von oben nach unten oder links nach rechts definiert) und (f) keine Richtung der Reaktion erkennbar.

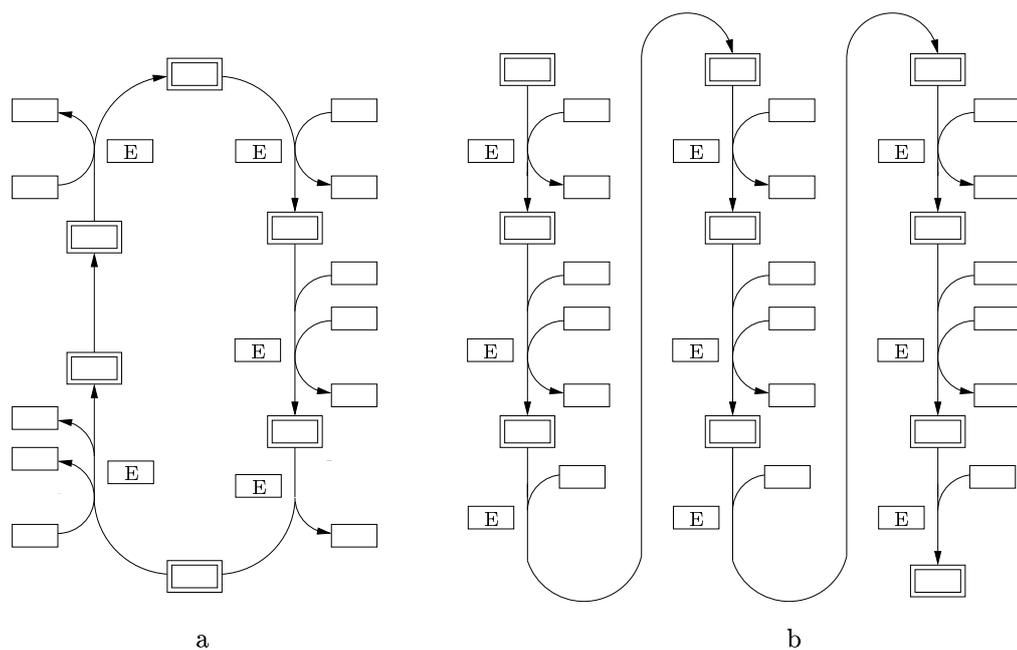


Abbildung 3.6: **Spezielle Reaktionswege.** Schematische Darstellung von (a) geschlossenen Zyklen und (b) offenen Zyklen. Einfache Rechtecke symbolisieren Cosubstanzen, doppelte Rechtecke symbolisieren Substanzen und Rechtecke mit dem Buchstaben E stehen für Enzyme.

sind jeweils auf derselben Ebene platziert. Auch hier überkreuzen sich die Teile des Reaktionswegs nicht.

Bei geschlossenen Zyklen kann der Anwender festlegen, auf welcher Seite der erste Teil des Wegs und auf welcher der zweite Teil liegt, ob der Zyklus also rechts oder links herum verläuft. An den Umkehrpunkten der Richtung des Reaktionswegs werden Kanten zwischen den Substanzen nicht geradlinig, sondern als Kreisbogen oder Kurve dargestellt. Abbildung 3.6 zeigt die schematische Darstellung der Visualisierung von geschlossenen und offenen Zyklen.

3.3.2.1 Begründung der Anforderungen

Die aufgeführten Anforderungen spiegeln besonders die Wünsche der beteiligten Biochemiker nach Darstellung der zeitlichen Abfolge der Reaktionen und der klassenspezifischen Darstellung der Komponenten der Reaktionen (z. B. alle Enzyme mittels EC-Nummer) wider. Sie berücksichtigen zudem die herkömmlichen Zeichnungen, bei denen Ebenenorientierung und Kreuzungsvermeidung wesentliche Aspekte sind, siehe Abbildungen 3.7. Daneben spielt die Vermeidung von Kantenknicken und eine kompakte Darstellung mit geringer Fläche eine große Rolle, über 80 bzw. 90 Prozent der untersuchten Zeichnungen erfüllen diese Kriterien. Eine balancierte bzw. symmetrische Gesamtdarstellung ist hingegen nicht wichtig. Insgesamt sind nur 31 % der Zeichnungen entsprechend dargestellt. Bei den meisten dieser Darstellungen handelt es sich zudem um Reaktionswege, die

	[LNC94]	[Cam98]	[FM99]	[Mic99]	[Mic93,Exp00]	[Str95]	[ABL97]	[KEG00]	[WIT00]	[UMB00]	% der Darstellungen
Zahl untersuchter Abbildungen	20	5	20	20	20	20	10	20	20	20	100
Minimierung von Kreuzungen	20	5	18	19	15	20	10	20	20	20	95
Orientierung an Ebenen ¹⁹	17	3	15	16	9	17	7	7	20	20	75
Eine Richtung für alle Reaktionen ²⁰	14	3	11	13	4	14	9	2	12	20	58
Balanciertheit/Symmetrie ²¹	13	3	5	2	2	9	4	0	5	15	31
Geradlinigkeit (Darstellung ohne Kantenknicke)	16	4	13	7	5	11	9	15	15	18	65
Minimierung der Zahl der Kantenknicke ²²	18	5	16	10	13	20	9	18	16	18	82
Anordnung mit kurzen Kanten	19	5	18	13	12	20	10	15	16	20	85
Knotengrößen differieren mehr als 50%	17	4	14	16	20	17	6	0	0	7	58
Verwendung spezieller Darstellungen (z. B. für Zyklen)	2	1	4	2	2	2	1	2	0	0	9
Minimierung der Fläche/kompakte Darstellung	18	5	18	19	20	20	10	17	16	19	93

Abbildung 3.7: **Analyse herkömmlicher Darstellungen (Teil 3)**. Hier wird das Erscheinungsbild der gesamten Zeichnung analysiert. Für weitere Informationen siehe auch die Abbildungen 3.2 und 3.4.

bei linearer Anordnung der Reaktionen und Substanzen automatisch balanciert gezeichnet werden. Interessant ist hier auch, dass bei mehr als der Hälfte der untersuchten Zeichnungen sehr unterschiedlich große Komponenten auftraten. Dies liegt an den sehr verschiedenen Größen der Strukturformeln.

Die hier aufgestellten Anforderungen an die Darstellung von geschlossenen und offenen Zyklen weichen von den bisher in der Biochemie verwendeten Darstellungen ab. Geschlossene Zyklen werden meist als Kreis, offene Zyklen als Spirale dargestellt (siehe Abb. 2.6 auf Seite 21). Die hier geforderte Art der Darstellung hat zwei wesentliche Vorteile:

1. Das Erscheinungsbild der Visualisierungen ist einheitlicher und entspricht der Forderung, dass

¹⁹Die überwiegende Zahl der Komponenten ist so angeordnet, dass eine ebene Einteilung deutlich wird.

²⁰Alle Reaktionen verlaufen in eine Richtung, z. B. von oben nach unten. Doppelpfeile werden dabei ignoriert.

²¹Dies trifft fast nur auf Darstellungen von Reaktionswegen (also lineare Abfolgen von Reaktionen) zu.

²²Geradlinige Zeichnungen gehören ebenfalls dazu.

Reaktionen ebenenweise angeordnet sein und möglichst viele Reaktionen in der Richtung von oben nach unten verlaufen sollen. Besonders bei Spiralen ist diese Forderung nicht erfüllt.

Die in Büchern und auf dem Poster *Biochemical Pathways* als Spirale dargestellten offenen Zyklen umfassen nur wenige Windungen und stellen damit nur einen kleinen Ausschnitt des Reaktionswegs dar. So benötigt die Synthese einiger Fettsäuren mehrere Dutzend Durchläufe durch die sich wiederholende Reaktionsfolge, die Synthese von Polysacchariden teilweise hunderte Durchläufe. Die Darstellung eines kompletten Reaktionswegs würde zu sehr großen Spiralen führen. Dabei würden die äußeren Windungen sehr lange Kanten für die Reaktionen enthalten, da gleiche Reaktionen benachbart dargestellt werden sollen (vergleiche Abb. 3.8, die den Effekt der langen Kanten bei mehr Spiralwindungen verdeutlicht). Da für einen Anwender die Verfolgung langer Reaktionskanten schwieriger als die kurzer Kanten ist, sollen Zeichnungen mit langen Kanten vermieden werden. Die hier geforderte Art der Darstellung vermeidet lange Kanten.

2. Offene und geschlossene Zyklen lassen sich mit der hier geforderten Darstellung besser auf einer begrenzten Fläche unterbringen, die Fläche wächst nur linear zur Zahl der Reaktionen. Die Fläche bei Darstellungen mittels Spirale wächst dagegen quadratisch, da jeder weitere Durchlauf der Reaktionssequenz nicht nur die Breite, sondern auch die Höhe der Abbildung vergrößert, siehe Abbildung 3.8. Das Problem des erhöhten Flächenbedarfs für die Visualisierung tritt auch bei der Darstellung von geschlossenen Zyklen mittels Kreisen auf.

Bei elektronischen Systemen kommt hinzu, dass bei großen Kreisen und Spiralen ein Scrollen bzw. Verschieben des aktuellen Ausschnitts in zwei Richtungen nötig ist, um den Reaktionsweg verfolgen zu können. Dies ist für einen Nutzer sehr unbequem. In der hier geforderten Darstellung ist dagegen oft kein Verschieben nötig oder ein Verschieben in eine Richtung ausreichend (siehe Abb. 3.8). Zudem schneiden bei Spiralen Kanten von außerhalb ins Zentrum der Spirale andere Kanten unnötig oft (siehe Abb. 2.6).

3.4 Anforderung III: Kontexterhaltende Navigation und Sichten

3.4.1 Navigation durch Reaktionsnetze

Visualisierungen von Reaktionsnetzen zeigen üblicherweise nur Ausschnitte des Gesamtstoffwechsels auf einer bestimmten Hierarchiestufe. Das Wechseln zwischen verschiedenen Teilen des Stoffwechsels oder verschiedenen Stufen der Hierarchie wird als *Navigieren* bezeichnet. Die *Navigation* durch biochemische Reaktionsnetze ist also das Verändern der aktuellen Visualisierung, um den Stoffwechsel zu erkunden.

Dabei werden zwei Mechanismen unterschieden, vergleiche dazu auch Abbildung 3.9:

1. Die Navigation *innerhalb derselben Hierarchieebene*, die *Erweiterung* und die *Reduktion* eines Reaktionsnetzes bzw. einer zugehörigen Visualisierung.
Erweiterung ist das Hinzufügen zusätzlicher Reaktionen in ein Reaktionsnetz. *Reduktion* ist das Entfernen von Reaktionen eines Reaktionsnetzes.
2. Die Navigation *zwischen verschiedenen Hierarchieebenen*, die *Verfeinerung* und die *Vergrößerung* eines Reaktionsnetzes bzw. einer zugehörigen Visualisierung.

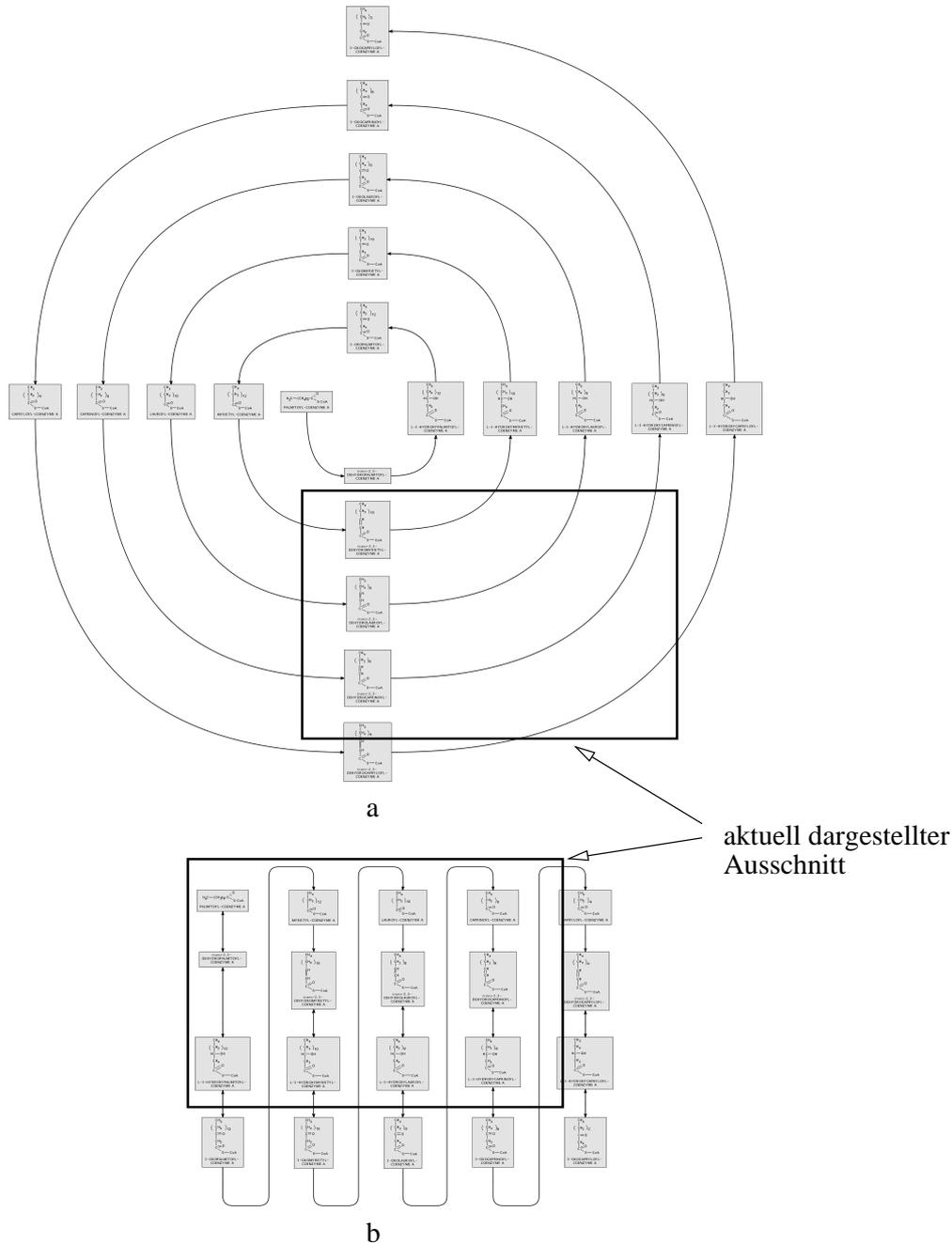


Abbildung 3.8: **Darstellungsformen für offene Zyklen.** Zwei Arten der Darstellung offener Zyklen: (a) als Spirale, (b) entsprechend den hier formulierten Anforderungen. Da gleichartige Reaktionsschritte benachbart sein sollen (um die Wiederholung der Reaktionsfolge hervorzuheben), besitzen äußere Durchläufe der Spirale sehr lange Kanten, ein Phänomen, dass in der unteren Darstellung nicht auftritt. Beim Zufügen weiterer Durchläufe der Reaktionssequenz wächst die Spirale in Breite und Höhe, die Darstellung in (b) dagegen nur in der Breite. Zugleich ist in beiden Visualisierungen ein möglicher Ausschnitt dargestellt, der auf der begrenzten Ausgabefläche (Monitor) aktuell dargestellt wird. Dabei wird die schlechte Ausnutzung der Fläche bei Spiraldarstellungen deutlich.

Verfeinerung ist das Ersetzen einer Reaktion durch ein Reaktionsnetz, das mit denselben Edukten wie die Reaktion beginnt und mit denselben Produkten endet. *Vergrößerung* ist das Ersetzen eines Reaktionsnetzes durch eine Reaktion, die dieselben Edukte und Produkte wie das entfernte Reaktionsnetz hat.

Verfeinerung entspricht auch einem Abstieg in der Hierarchie, dem Ersetzen einer übergeordneten Reaktion durch die Menge zugehöriger Einzelreaktionen. Vergrößerung entspricht dem Aufsteigen in der Hierarchie, dem Ersetzen der Menge zusammengehöriger Reaktionen durch die übergeordnete Reaktion.

3.4.2 Sichten auf Reaktionsnetze

Sichten sind Visualisierungen *eines* Reaktionsnetzes, in denen unterschiedliche Informationen dargestellt werden. Sichten entstehen beispielsweise durch Ein- oder Ausblenden bestimmter Komponenten (Cosubstanzen oder Enzyme) oder durch die Verwendung der Strukturformeln zusätzlich zum Namen. In Abbildung 3.11 sind verschiedene Sichten eines Reaktionsnetzes dargestellt.

Der Unterschied zur Navigation liegt darin, dass die Navigation stets die Struktur des Reaktionsnetzes durch Einfügen oder Entfernen von Reaktionen verändert, während bei Sichten diese erhalten bleibt.

3.4.3 Kontexterhaltende Darstellungen bei Navigation und Sichten

Kontexterhaltende Navigation und *kontexterhaltende Sichten* sind das Erzeugen von Visualisierungen eines Reaktionsnetzes auf der Grundlage einer bereits bestehenden Visualisierung mittels Erweiterung, Reduktion, Vergrößerung, Verfeinerung oder der Anwendung von Sichten. Dabei sollen die Bereiche des Reaktionsnetzes, die bereits in der bestehenden Visualisierung enthalten sind, so dargestellt werden, dass sie schnell wiedererkennbar sind. Die neu hinzukommenden Teile sollen sich also in die vorhandene Darstellung einpassen. Die bereits bestehende Visualisierung wird als *Ursprungszeichnung* bezeichnet.

ANFORDERUNG 3.5 (KONTEXTERHALTENDE DARSTELLUNG)

Eine kontexterhaltende Darstellung eines Reaktionsnetzes auf der Grundlage einer Ursprungszeichnung eines Teils dieses Netzes erfolgt entsprechend den Anforderungen 3.1-3.4. Die Darstellung muss dabei folgenden weiteren Regeln genügen:

- 1. Reaktionen und deren Komponenten, die bereits in der Ursprungszeichnung enthalten sind, werden in der neuen Darstellung mit derselben Reaktionsrichtung und der gleichen Anordnung der Komponenten, also der gleichen Lage der Komponenten bezogen auf den gemeinsamen Reaktionspfeil, dargestellt.*
- 2. Eine Reaktion, die in der Ursprungszeichnung auf einer höheren Ebene als eine zweite Reaktion platziert ist, wird in der neuen Darstellung ebenfalls auf einer höheren Ebene als die zweite Reaktion dargestellt.*

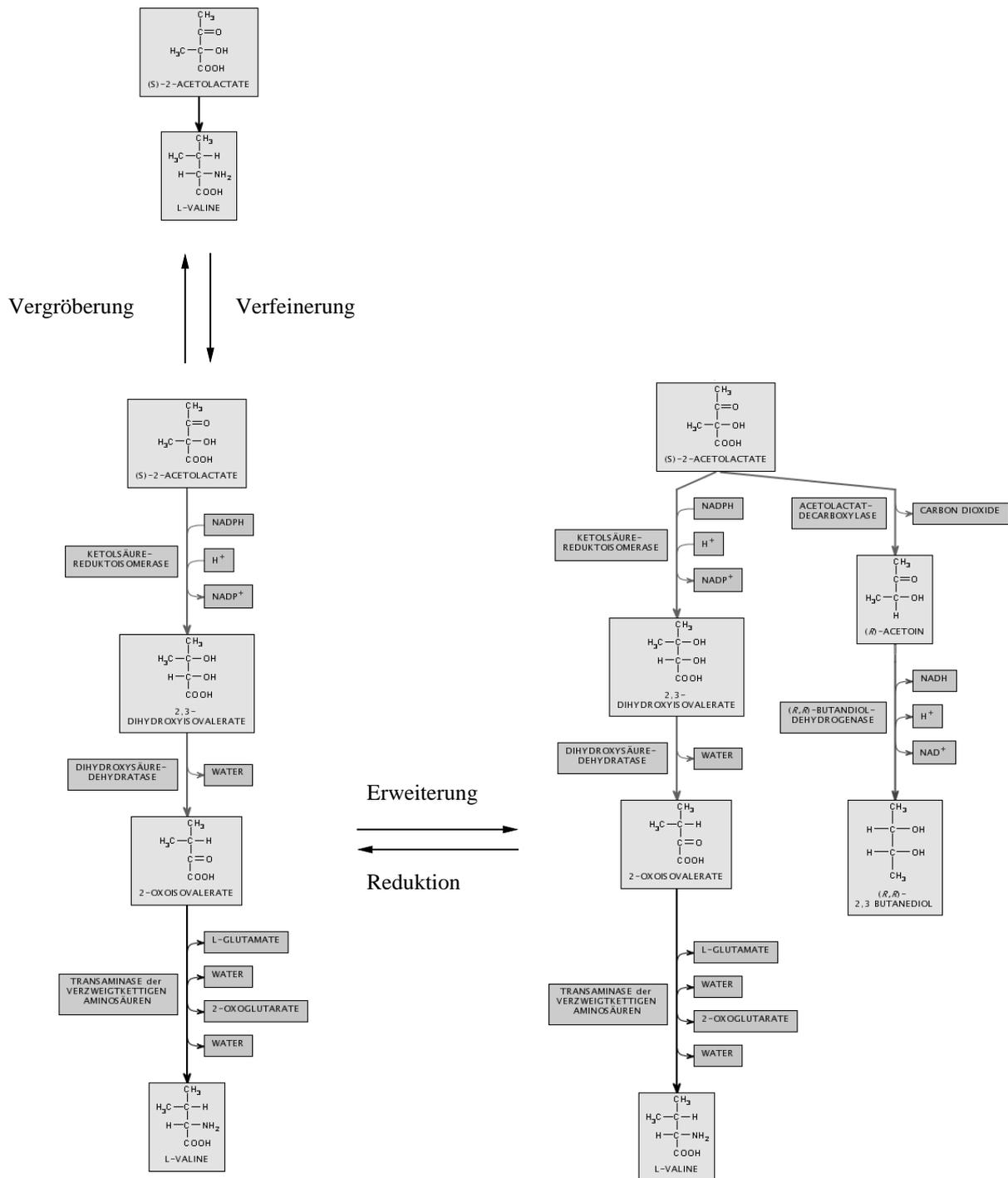


Abbildung 3.9: **Navigation.** Navigation innerhalb einer Hierarchieebene (Erweiterung und Reduktion) sowie zwischen Hierarchieebenen (Vergrößerung und Verfeinerung), zur Erklärung siehe Abbildung 3.10.

3 Darstellungsanforderungen

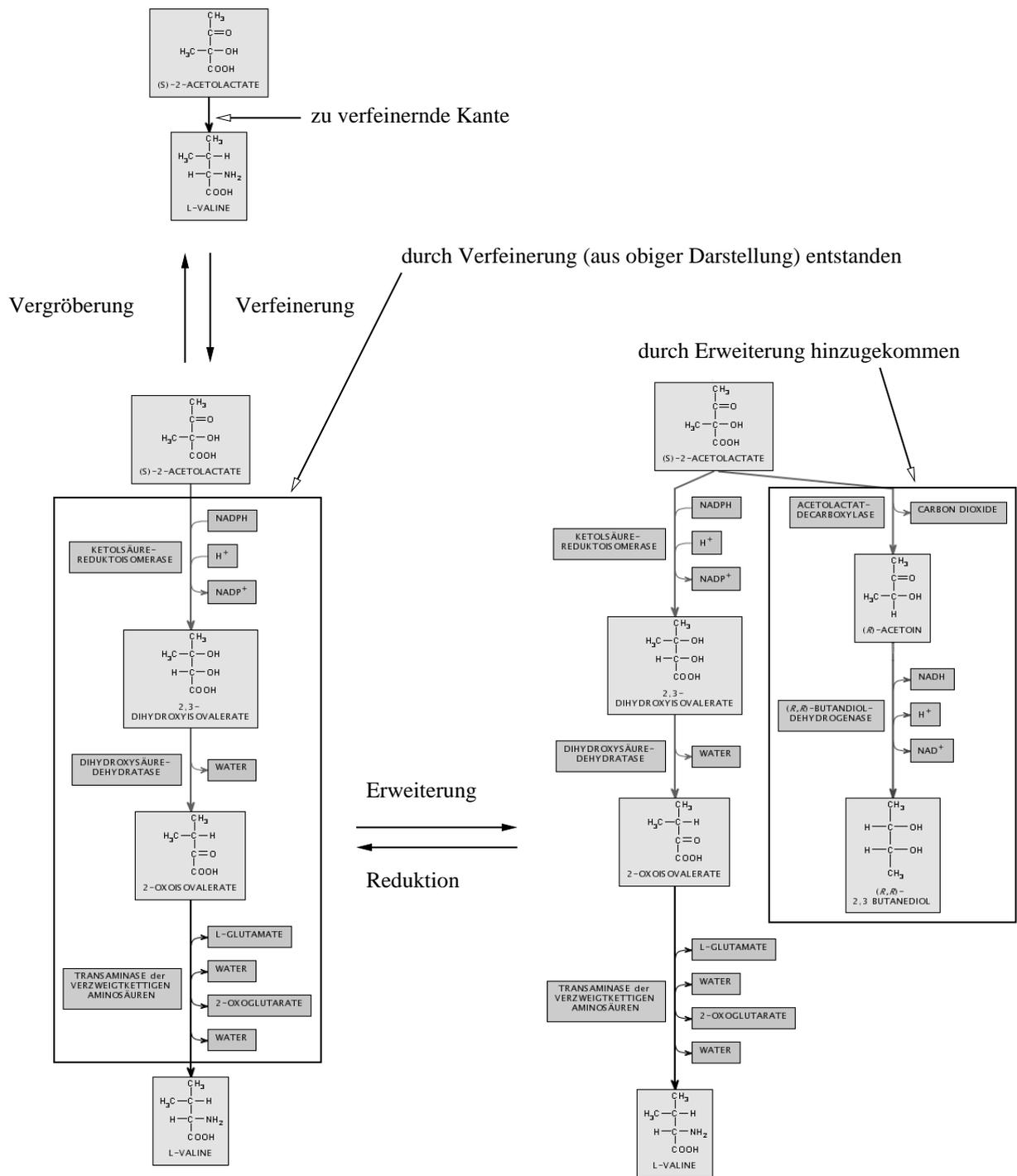


Abbildung 3.10: Navigation - Verdeutlichung der Schritte. (links) Eine Kante und der für ihre Verfeinerung eingefügte Bereich. (rechts) Eine Erweiterung des linken Reaktionsnetzes.

3.4 Anforderung III: Kontexterhaltende Navigation und Sichten

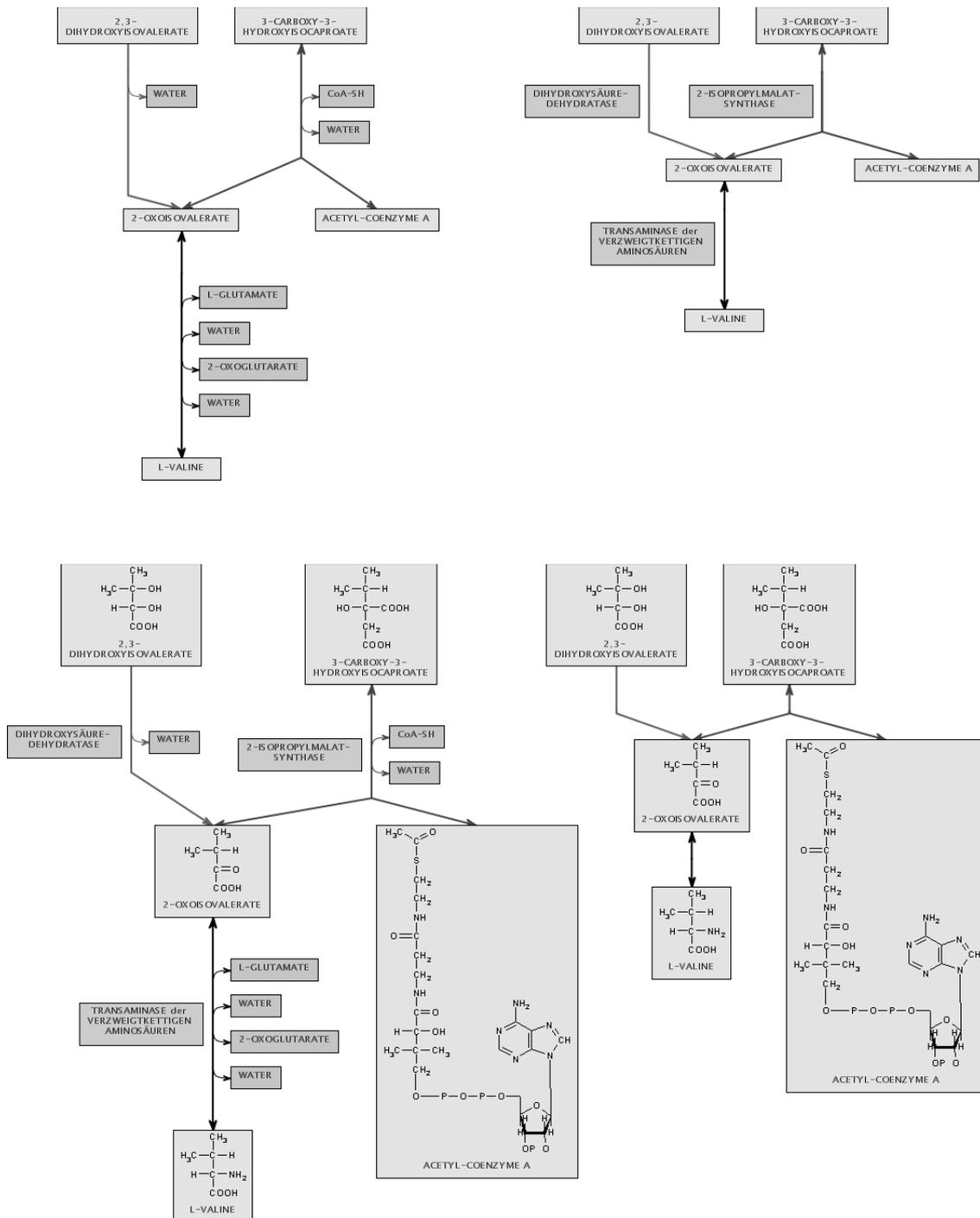


Abbildung 3.11: **Sichten.** Darstellung verschiedener Sichten desselben Reaktionsnetzes, z. B. (oben links) Verwendung von Substanzen und Cosubstanzen mit ihren Namen bzw. Abkürzungen, (unten rechts) Darstellung der Substanzen mit Name und Strukturformel, keine Enzyme oder Cosubstanzen.

3.4.3.1 Begründung der Anforderung

Die beiden Punkte von Anforderung 3.5 sollen im Folgenden erörtert werden. Ziel kontexterhaltender Darstellungen ist, das Verstehen der neuen Zeichnung zu erleichtern, indem die Wiedererkennung der in der ursprünglichen Zeichnung enthaltenen Teile gefördert wird. Dies lässt sich bei biochemischen Reaktionsnetzen in zwei Aspekte zerlegen: Einerseits einen lokalen Aspekt, die Gruppierung der Komponenten einer Reaktion und andererseits einen globalen Aspekt, die relative Lage der einzelnen Reaktionen zueinander.

Im ersten Punkt werden einzelne Reaktionen betrachtet. Es wird gewährleistet, dass die Darstellung einer Reaktion in der neuen Zeichnung ähnlich der Visualisierung in der Ursprungszeichnung ist. Dabei wird davon ausgegangen, dass Anwender zwei Darstellungen von Reaktionen dann als ähnlich betrachten, wenn sie dieselben Komponenten besitzen, in derselben Richtung verlaufen, die relative Lage der Komponenten bezogen auf den Reaktionspfeil gleich ist und die Komponenten ähnlich weit wie in der Ursprungszeichnung voneinander entfernt sind. Diese Annahmen decken sich mit Nutzerbefragungen bei den bereits erwähnten Treffen und mit Aussagen von Michal [Mic00].

Wenn Reaktionen mehrere Edukte oder Produkte haben, so ist nach Aussagen von Anwendern auch die Erhaltung der links-rechts-Anordnung der Edukte bzw. Produkte in der neuen Zeichnung wesentlich für ein schnelles Orientieren bzw. Verstehen der Zeichnung. Die Edukte bzw. Produkte einer Reaktion sind dabei oft auf einer Ebene platziert.

Im Punkt 2 wird die Lage der Reaktionen zueinander betrachtet. Die Darstellung der zeitlichen Reihenfolge ist ein wichtiger Aspekt bei der Visualisierung von Reaktionsnetzen. Es ist nur konsequent, wenn die zeitliche Reihenfolge von Reaktionen und ihren Komponenten aus der Ursprungszeichnung in der neuen Zeichnung erhalten bleibt. Die Reihenfolge ist bei Zeichnungen von oben nach unten durch die Lage einer Reaktion über oder unter einer anderen gegeben. Diese ebenenweise Abhängigkeit soll möglichst erhalten bleiben.

Hier wird darauf verzichtet, den Erhalt aller oben-unten und rechts-links Beziehungen zwischen Reaktionen bzw. Komponenten zu fordern. Ein solches Vorgehen würde zwar die relative Lage der Objekte der Ursprungszeichnung erhalten, führt aber zu Visualisierungen, die der Anforderung 3.3 widersprechen können. Beispielsweise sind nun Edukte und Produkte im Bezug zum Reaktionspfeil u. U. nicht mehr zentriert (siehe Abb. 3.12). Bei kontexterhaltenden Darstellungen biochemischer Reaktionsnetze geht es nicht um den exakten Erhalt der Lagebeziehungen zwischen Reaktionen, sondern darum, dass die Umgebung einer Reaktion möglichst erhalten bleibt, so dass der Anwender eine Reaktion schnell in den Kontext der sie umgebenden Reaktionen einordnen kann.

3.4.4 Hierarchisierung

Voraussetzung für Vergrößerung und Verfeinerung ist eine Hierarchie biochemischer Reaktionen. Die Auszeichnung von Substanzen als *Schlüsselsubstanzen*, die daraus resultierenden Stoffwechselwege und damit ein wesentlicher Teil der Hierarchie ist nicht einheitlich festgelegt. So verwenden das Poster und der Atlas *Biochemical Pathways* [Mic93, Mic99] sowie die *Kyoto Encyclopedia of Genes and Genomes*, *KEGG* [KEG00, KG00] Zuordnungen, die sich historisch entwickelt haben. Dennoch unterscheiden sich Atlas und KEGG. So sind im Atlas die Synthesen von Hämoglobin und Chlorophyll zwei verschiedene Stoffwechselwege, in KEGG werden sie unter dem Stoffwechselweg *Porphyrin and chlorophyll metabolism* zusammengefasst. Der Stoffwechselweg *Ribulose Monophosphate Pathway* aus [Mic99] kommt in *KEGG* nicht als eigener Stoffwechselweg vor. Ein Nachteil

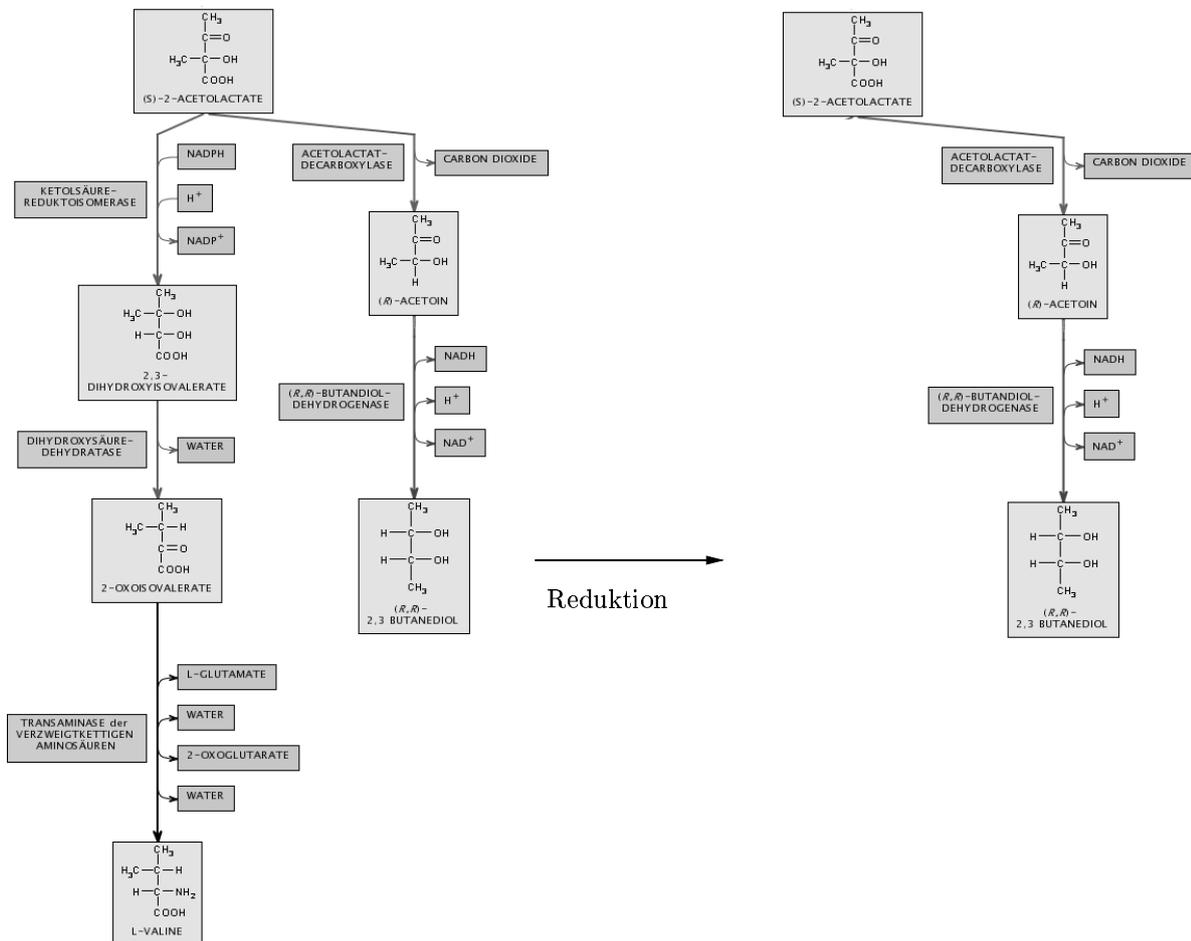


Abbildung 3.12: **Kontexterhaltende Darstellung.** Ein Erhalt aller links-rechts Beziehungen zwischen Reaktionen und Komponenten kann zu unerwünschten Visualisierungen führen. Beispielsweise wurde hier das Reaktionsnetz nach der Reduktion so visualisiert, dass alle links-rechts Beziehungen erhalten bleiben. So liegt die Substanz (*R*)-Acetoin rechts der Substanz (*S*)-2-Acetolactate. Es werden aber nun nicht mehr alle Edukte und Produkte gegenüber ihrer Reaktion zentriert.

dieser historisch hergebrachten Hierarchisierungen ist, dass die Hierarchie nicht eindeutig ist, so sind einige Reaktionen mehreren Stoffwechselwegen zugeordnet. Ein weiterer Nachteil ist, dass die Vollständigkeit der Hierarchie nur dadurch erreicht wird, dass alle Reaktionen, die keinem Stoffwechselweg zugeordnet sind, direkt der Spitze der Hierarchie zugeordnet werden.

Neuere Hierarchisierungen sind dagegen eindeutig. So ist die Zuordnung in der *Metabolic Pathways Database MPW* [SGMS98, WIT00] so gestaltet, dass Stoffwechselwege auf zusammengehörige Reaktionen beschränkt sind, bei denen die Substanzen unter spezifischen physiologischen Bedingungen in keinem anderen Reaktionsweg vorkommen.

Nicht nur die Zusammenfassung von Reaktionen zu Stoffwechselwegen ist unterschiedlich, auch deren Zusammenfassung zu Stoffwechselklassen und sogar die Tiefe der Hierarchie sind nicht einheitlich festgelegt. So enthält *KEGG* im Vergleich zu [Mic99] andere Stoffwechselklassen und die Ebene der Teilreaktionen ist leer.

Hier werden deshalb keine Anforderungen an die Gestaltung der Hierarchie biochemischer Reaktionsnetze gestellt, die über die in Abschnitt 2.1.3.1 beschriebene Vollständigkeit und Eindeutigkeit der Hierarchie hinausgehen. Im Kapitel 5 wird auch die Erweiterung auf nicht eindeutige Hierarchien betrachtet, wobei Eigenschaften einer einfachen Navigation jedoch teilweise verloren gehen.

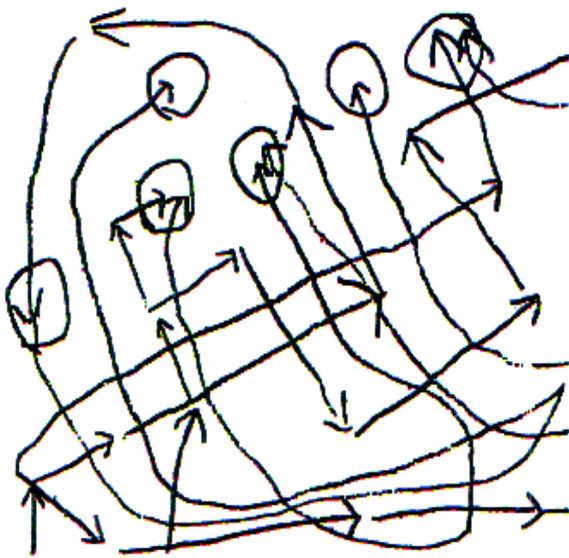
3.5 Zusammenfassung der Anforderungen

Zusammenfassend lassen sich die in den Abschnitten 3.2 bis 3.4 dargestellten Anforderungen klassifizieren zu:

1. *lokalen* Anforderungen. Diese betreffen die Darstellung der Komponenten einer Reaktion (siehe Abschnitt 3.2).
2. *globalen* Anforderungen. Diese betreffen die Darstellung des Reaktionsnetzes (siehe Abschnitt 3.3).
3. Anforderungen an *kontexterhaltende Navigation und Sichten*. Diese betreffen die Darstellung von Reaktionen und Reaktionsnetzen in aufeinander folgenden Visualisierungen (siehe Abschnitt 3.4).

Nachdem Anforderungen an die Visualisierung biochemischer Reaktionsnetze formuliert wurden, wird im folgenden Kapitel untersucht, wie vorhandene Visualisierungen und insbesondere Darstellungen in Informationssystemen diese Anforderungen erfüllen.

4 Traditionelle Realisierungen



4.1	Umsetzung der Anforderungen durch statische Visualisierungen	60
4.2	Umsetzung der Anforderungen durch dynamische Visualisierungen	68
4.3	Fazit	79

Die Arten der Darstellung biochemischer Reaktionsnetze sind vielfältig. Hier eine ungewöhnliche Art aus [Mic98a]. Dazu schreibt Michal:

„This complexity impressed my 4-year old grandson, when watching me work, and caused him to design his own version of metabolism.“

4.1 Umsetzung der Anforderungen durch statische Visualisierungen

4.1.1 Charakterisierung statischer Visualisierungen

Die heute häufigste Form der Visualisierung biochemischer Reaktionsnetze sind manuell erstellte Zeichnungen. Bei diesen Visualisierungen handelt es sich um vorgefertigte Darstellungen, die *vor* dem Zeitpunkt erzeugt werden, zu dem ein Anwender sie benötigt. Diese Visualisierungen werden als *statische Visualisierungen* bezeichnet und sind wie folgt charakterisiert:

1. Sie werden zeitlich lange *vor* ihrer Verwendung als Abbild der zum Zeitpunkt der Erstellung *vorhandenen* Daten (bzw. eines Teils der Daten) erzeugt.
2. Sie liefern eine vom *Erzeuger* festgelegte Sicht auf die Daten.
3. Sie werden *oft* verwendet und vielfältigt.
4. Sie sind im Allgemeinen *nicht* bzw. nur mit viel Aufwand *änderbar*, wobei Änderungen eine Wiederholung des Publikationsprozesses erfordern. Auch Änderungen während des Erstellungsprozesses können aufwendig sein.

Das Erzeugen statischer Visualisierungen von biochemischen Reaktionsnetzen ist eine manuelle Tätigkeit, die Experten erfordert. Auch wenn statische Visualisierungen oft mit Hilfe von Computerprogrammen wie CorelDraw [DHS00] erzeugt werden, so bleibt doch der Ersteller der Zeichnung für die Platzierung der Objekte verantwortlich.

4.1.2 Vorkommen statischer Visualisierungen

4.1.2.1 Publikationen

Es gibt eine Vielzahl von Büchern und Veröffentlichungen in der Biochemie, die statische Visualisierungen von Reaktionsnetzen beinhalten. Da nahezu jedes Buch und jede Veröffentlichung in der Darstellung variiert, können in dieser Arbeit nur exemplarisch typische Fälle betrachtet werden. In den Tabellen der Abbildungen 3.2, 3.4 und 3.7 wurden bereits statische Visualisierungen weit verbreiteter Bücher analysiert.

Die dort verwendeten Darstellungen erfüllen überwiegend die Anforderungen an lokale Eigenschaften: So werden Substanzen mit Namen und Strukturformel dargestellt und die Substanzen sind an den Enden, die Cosubstanzen und Enzyme neben dem Reaktionspfeil platziert. Globale Eigenschaften werden dagegen oft nur für einzelne Stoffwechselwege erfüllt. Reaktionsnetze werden nicht mit einer Richtung dargestellt, sondern Teile der Netze verlaufen in verschiedene Richtungen (siehe z. B. [Mic93], wo einzelne Stoffwechselwege horizontal, vertikal, kreisförmig oder als Spirale verlaufen, das gesamte Netz jedoch keine Richtung besitzt). Grund für solche Darstellungen ist die begrenzte Fläche, die optimal ausgenutzt werden soll.

Die Möglichkeit der Navigation durch den Stoffwechsel ist bei diesen Darstellungen nur selten gegeben, ein Beispiel findet sich in [Mic99]. Dort stellt die Umschlagseite ein Übersichtsbild über den Stoffwechsel dar, das Bild selbst enthält Verweise auf die Kapitel, in denen sich zu den einzelnen Stoffwechselwegen nähere Informationen finden. Diese Navigation ist nicht kontexterhaltend, da nur der Ausschnitt des Stoffwechsels in einer detaillierten Sicht dargestellt wird.

4.1 Umsetzung der Anforderungen durch statische Visualisierungen

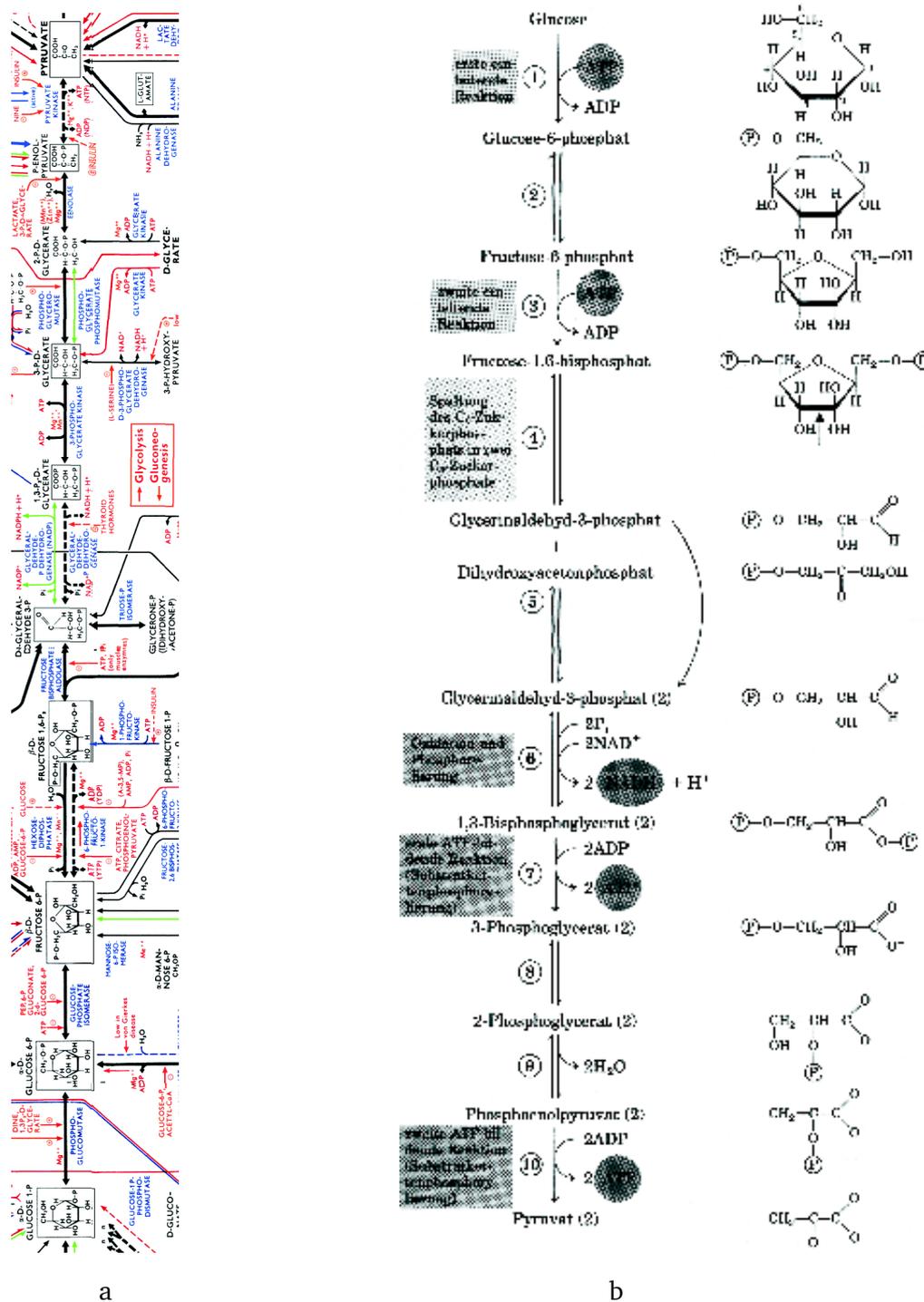


Abbildung 4.1: **Statische Visualisierung biochemischer Reaktionsnetze (Teil 1)**. Die Visualisierung des Stoffwechselweges *Glycolyse* aus (a) dem Poster *Biochemical Pathways* [Mic93] und (b) dem Buch *Prinzipien der Biochemie* [LNC94]. Der Ausschnitt aus dem Poster wirkt sehr unübersichtlich, da von den Substanzen weitere Reaktionswege abgehen. Dagegen zeigen Darstellungen aus Büchern oft nur den Stoffwechselweg, aber keine Verweise auf weitere Reaktionswege, wodurch sie übersichtlicher sind.

4.1.2.2 Informationssysteme

Auch die meisten Informationssysteme verwenden statische Visualisierungen. Dabei bedienen sich die Biochemiker häufig des Internets, um aktuelle Forschungsergebnisse darzustellen. Es existieren zahlreiche Anbieter biochemischer Daten, vorwiegend Universitäten und Forschungsinstitute.¹ Nur wenige dieser Informationssysteme stellen auch biochemische Reaktionswege und -netze (so genannte *Metabolic Pathways* oder *Biochemical Pathways*) bereit, diese werden im Folgenden vorgestellt.

Bei allen betrachteten Informationssystemen und besonders den in Abschnitt 4.2 dargestellten Systemen mit automatischen Visualisierungsverfahren handelt es sich um aktive Projekte, die ständig weiterentwickelt werden. Da zu den meisten Systemen keine Veröffentlichungen über die verwendeten Visualisierungsmethoden vorliegen und die Ziele für die nächsten Entwicklungsschritte der Systeme oft nicht oder nur vage formuliert sind, kann die hier vorgenommene Beschreibung nur eine Momentaufnahme dieser Systeme darstellen.

4.1.2.2.1 ExPASy

Das *Expert Protein Analysis System ExPASy* [ABH94, ExP00] des *Swiss Institute of Bioinformatics* verwendet eine in der grafischen Darstellung unveränderte Version des Posters *Biochemical Pathways* [Mic93] (siehe Abb. 4.1(a)), wobei jeweils ein oder zwei Planquadrate des Posters dargestellt werden.

Positiv an diesem System ist die Verknüpfung der grafischen Darstellung mit zusätzlichen Funktionen wie Suche nach Stoffwechselwegen, Substanzen und Enzymen und Darstellung der entsprechenden Ausschnitte des Posters. Diese Funktion kann als elektronischer Index des Posters aufgefasst werden und stellt eine wesentlich schnellere Navigation als mittels des gedruckten Index dar. Vorteilhaft ist weiter die Verlinkung in entgegengesetzter Richtung: Jede Darstellung eines Enzyms verweist auf den entsprechenden Eintrag in der Enzymdatenbank *ENZYME* [ENZ00].

Negativ ist die Beschränkung der Darstellung auf maximal zwei benachbarte Planquadrate. Zwar lässt sich in der Darstellung das nächste Quadrat einer beliebigen Seite anfügen, da aber eine Übersichtsdarstellung fehlt, ist die Orientierung schwieriger als im gedruckten Original. Durch die Begrenzung auf maximal zwei Quadrate ist es beispielsweise nicht möglich, den Stoffwechselweg *Glycolyse* wie in Abbildung 4.1(a) darzustellen, da dieser über vier Quadrate geht.

4.1.2.2.2 KEGG

Die unter Leitung von Kanehisa entwickelte *Kyoto Encyclopedia of Genes and Genomes KEGG* [Kan96, KEG00, KG00, OGS⁺99] ist eine oft zitierte und nach Aussagen von Biochemikern die am meisten verwendete Datenbank für biochemische Reaktionswege. Abbildung 4.2(a) gibt die Darstellung des Stoffwechselwegs *Glycolyse* aus diesem Informationssystem wieder. *KEGG* umfasst neben Stoffwechselwegen, die als vorgefertigte klickbare Bilder (Stoffwechselkarten) existieren, auch Darstellungen von regulatorischen Mechanismen sowie genetische Informationen.

Die statischen Darstellungen sind so aufbereitet, dass bei Anfragen Teile der Abbildung eingefärbt werden können. Wird die Darstellung auf spezielle Organismen eingeschränkt, so werden jene Teile

¹Einen Überblick über Informationssysteme für die Biochemie bietet [SRS00].

des Stoffwechselwegs eingefärbt, die in dem gesuchten Organismus vorkommen. Eine Suche nach einer Substanz oder einem Enzym führt zur farblichen Markierung des gesuchten Objekts in der statischen Darstellung. Im Gegensatz zu *ExpASy*, wo sich das nächste Quadrat an eine aktuelle Darstellung anfügen lässt, ist in *KEGG* kein Anfügen möglich. Dafür existiert ein klickbarer Verweis auf den anschließenden Stoffwechselweg. Dies vereinfacht die Navigation zwischen Stoffwechselwegen.

Negativ ist die stark von den Anforderungen abweichende Darstellung der Reaktionen, die im Vergleich zu anderen Abbildungen eine Orientierung erschwert. So werden Cosubstanzen nicht, Enzyme nur mit ihren EC-Nummern und Substanzen nur mit ihrem Namen dargestellt. Immerhin erhält man beim Klick auf eine Substanz weitere Informationen, u. a. auch ihre Strukturformel. Neben einer textuellen Übersicht über die Stoffwechselwege gibt es auch ein Übersichtsbild, eine kontexterhaltende Navigation ist aber nicht möglich. Von Nachteil ist weiterhin, dass es mit *KEGG* zwar möglich ist, die Reaktionswege zwischen zwei Substanzen zu berechnen, eine Anzeige jedoch nur in Form einer Liste der Schritte für jeden einzelnen Reaktionsweg erfolgt, nicht aber in Form der grafischen Darstellung des Reaktionsnetzes.

KEGG stellt, obwohl es statische Visualisierungen verwendet, geschlossene und offene Zyklen nicht als Kreise und Spiralen, sondern platzsparender im Stil der anderen Reaktionswege dar. So werden offene Zyklen entsprechend Anforderung 3.4 mit langen vertikalen Bereichen und geschlossene Zyklen, z. B. die Komponenten des *Citratzyklus*, auf dem Rand eines gedachten Rechtecks platziert, siehe Abbildung 4.2(b).

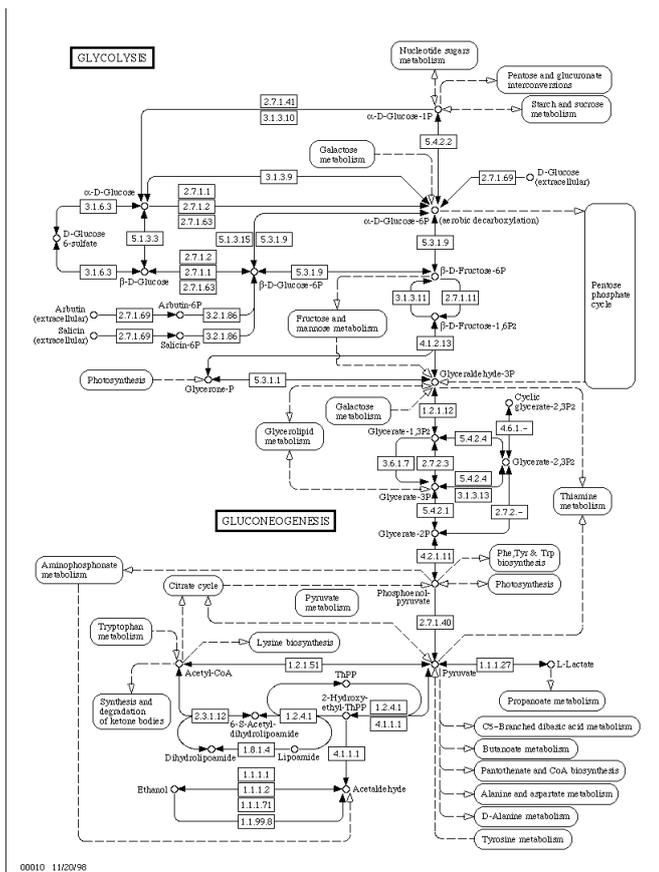
4.1.2.2.3 EMP, MPW, WIT, WIT2

Diese Begriffe stehen für mehrere von Selkov et al. entwickelte Systeme: Die *Metabolic Pathways Database MPW*, die *Enzymes and Metabolic Pathways Database EMP* und die auf diesen Datenbanken beruhenden Systeme *WIT* und *WIT2: What Is There – Interactive Metabolic Reconstruction on the WEB* [EMP00, SBG+96, SGI+97, SGMS98, WIT00]. Die Datenbank *EMP* enthält etwa 3000 Diagramme von Stoffwechselwegen, 50 Wissenschaftler sind unter anderem damit beschäftigt, pro Jahr etwa 1000 neue Darstellungen von Stoffwechselwegen zu zeichnen.² Neben vorgefertigten, klickbaren Abbildungen zur Darstellung von Stoffwechselwegen enthält *WIT2* auch Informationen über Gene und regulatorische Prozesse. Die Abbildung 4.3(a) zeigt eine Darstellung des Stoffwechselwegs *Glycolyse* aus *EMP/WIT2*.

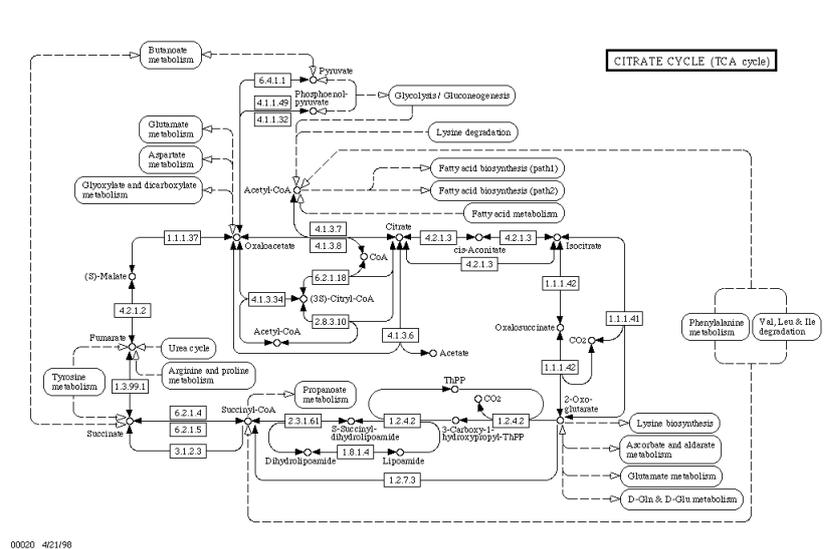
Positiv bei den Darstellungen der Stoffwechselwege ist die Erfüllung der lokalen und globalen Anforderungen für die Darstellung von Reaktionen, insbesondere die Platzierung von Cosubstanzen und Enzymen neben dem Reaktionspfeil und die Ausrichtung der Stoffwechselwege. Diese Darstellungen werden durch das große Expertenteam ermöglicht, das an der Weiterentwicklung der Datenbank und der Erstellung der Zeichnungen arbeitet. Vorteilhaft ist zudem die umfangreiche Verlinkung. So führt die Suche mittels Substanzen oder Enzymen zu den zugehörigen Stoffwechselwegen. Jedes Objekt der Darstellung, also Substanzen, Cosubstanzen, Enzyme und die Reaktionen, ist mit Datenbankeinträgen verlinkt.

Negativ sind auch hier fehlende Übersichtsdarstellungen, die Verwendung der Substanznamen, jedoch nicht der Strukturformeln in den Abbildungen und die fehlende kontexterhaltende Navigation

²Siehe Projektdaten in [EMP00].



a



b

Abbildung 4.2: Statische Visualisierung biochemischer Reaktionsnetze (Teil 2). Die Darstellung der Stoffwechselwege (a) Glycolyse und (b) Citratzyklus aus KEGG.

durch den Stoffwechsel. Zudem ist es nicht möglich, beliebige Reaktionswege und -netze zwischen zwei Substanzen zu berechnen und darzustellen.

4.1.2.2.4 UM-BBD

Schwerpunkt der *University of Minnesota Biocatalysis/Biodegradation Database UM-BBD* [EHW99, EHW00, UMB00, WE96] ist die Darstellung ausgewählter Stoffwechselwege, die beispielsweise Abbauprozesse von für den Menschen toxischen Substanzen beinhalten. Der Stil der Darstellungen ist nicht einheitlich, entspricht im wesentlichen aber dem Beispiel in Abbildung 4.3(b).

Positiv bei diesen Visualisierungen ist die Erfüllung der lokalen und globalen Anforderungen. So werden Cosubstanzen und Enzyme neben dem Reaktionspfeil dargestellt und Reaktionswege haben eine Richtung. Auch in *UM-BBD* erhält man beim Klick auf eine Substanz bzw. ein Enzym weitere Informationen. Vorteilhaft ist die Existenz einer klickbaren Übersichtsdarstellung für die in der Datenbank vorhandenen Stoffwechselwege.

Negativ bei diesen Visualisierungen ist, dass nur ein kleiner Ausschnitt des Stoffwechsels dargestellt wird und dass zwischen den einzelnen Abbildungen keine Verlinkung existiert. Die Navigation im Stoffwechsel beschränkt sich damit auf den Weg vom Übersichtsbild zu den einzelnen Stoffwechselwegen, kontexterhaltende Navigation ist nicht vorhanden. Außerdem ist es nicht möglich, beliebige Reaktionswege zwischen zwei Substanzen zu berechnen und darzustellen.

4.1.3 Vorteile statischer Visualisierungen

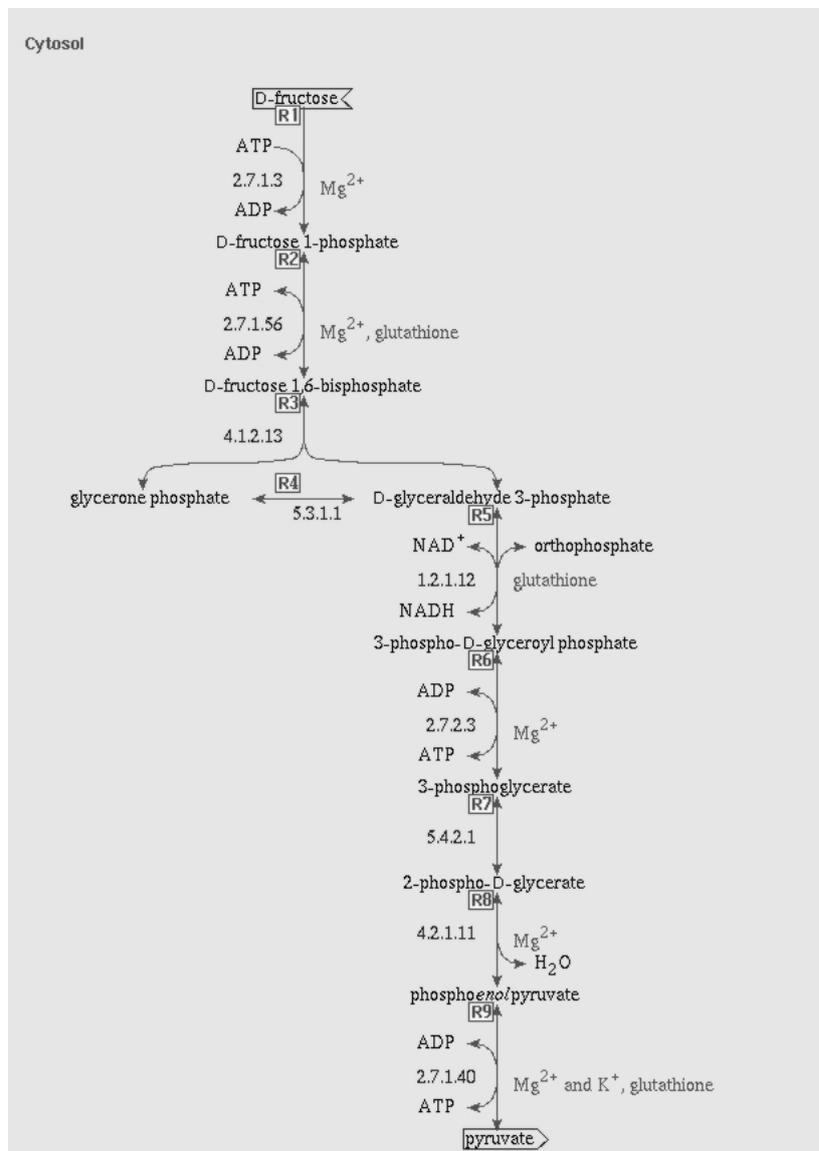
4.1.3.1 Erkennen biochemischer Zusammenhänge

Viele Prozesse der Biochemie haben sich erst durch statische Visualisierung veranschaulichen lassen. Zum Verständnis der Stoffwechselwege wie auch für Überblicke über den Zellstoffwechsel lassen sich solche Visualisierungen nicht hoch genug einschätzen. Die Bedeutung statischer Visualisierungen wird unter anderem daran deutlich, dass das Poster *Biochemical Pathways* seit 1968 publiziert wird, dass es über eine Million Mal gedruckt wurde und dass Michal mehr als ein Jahr für die Erstellung der letzten Edition 1993 benötigte [Mic00]. Auch ein ähnliches, nicht so umfangreiches Poster [Nic97] ist bereits in der 20. überarbeiteten Fassung erschienen.

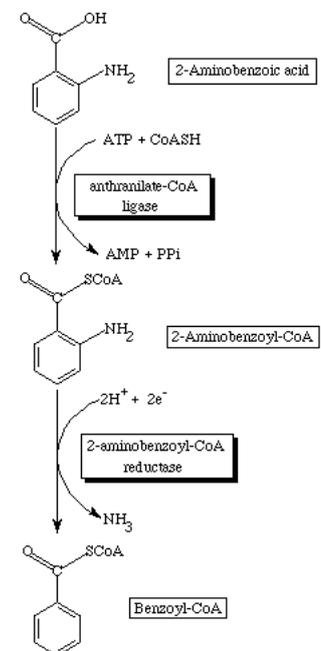
Die ersten Darstellungen von Stoffwechselwegen entstanden zudem zu einer Zeit, in der Bilder nur manuell erstellt werden konnten. So wurden einige bedeutende Stoffwechselwege bereits in der ersten Hälfte des zwanzigsten Jahrhunderts entdeckt, beispielsweise der Citratzyklus in den 30er Jahren durch Krebs und Johnson [KJ37].

4.1.3.2 Angepasste Darstellungen

Statische Darstellungen ermöglichen es, an spezielle Anforderungen angepasste Visualisierungen zu erstellen. Dabei steht neben der Übersichtlichkeit der Darstellung und der Herausarbeitung ausgewählter Aspekte eines Reaktionsnetzes die Beschränkung auf eine vorgegebene Fläche im Vordergrund. Der zur Erstellung solcher statischen Darstellungen nötige Aufwand ist jedoch so groß, dass er sich nur bei ausgewählten Abbildungen und deren wiederholter Verwendung lohnt, beispielsweise bei Postern oder in Lehrbüchern.



a



b

Abbildung 4.3: **Statische Visualisierung biochemischer Reaktionsnetze (Teil 3).** (a) Die Visualisierung des Stoffwechselwegs *Glycolyse* aus dem Informationssystem *WIT* und (b) ein Stoffwechselweg aus *UM-BBD*.

4.1.4 Nachteile statischer Visualisierungen

4.1.4.1 Festlegung auf eine Sicht

Neben den eben dargestellten Vorteilen haben statische Visualisierungen eine Reihe von Nachteilen. So bieten die Visualisierungen nur die vom Ersteller gewählte Sicht auf die Information. Oft sind jedoch unterschiedliche Sichten auf dieselben Daten gewünscht. Dies erfordert vom Ersteller eine wohl überlegte Auswahl der dargestellten Information. So führt der Versuch, möglichst viel Information in einer Darstellung unterzubringen, schnell zu sehr komplexen und teilweise unübersichtlichen Darstellungen (siehe Abb. 1.1 auf Seite 4).

4.1.4.2 Aufwendige und teure Erstellung, schwierige Aktualisierung

Darüber hinaus ist die Erstellung statischer Visualisierungen oft aufwendig und kostenintensiv. Gute Darstellungen erfordern einen Experten mit Kenntnissen in Biochemie wie im Erstellen ansprechender Zeichnungen. Dies gilt auch für Aktualisierungen existierender Darstellungen, da lokale Änderungen meist eine Neuplatzierung bereits vorhandener Objekte erfordern. Die aufwendige und teure Erstellung ist besonders problematisch, da heute täglich neue Daten über biochemische Reaktionen entdeckt werden, die unmittelbar im Kontext der zugehörigen Reaktionen visualisiert werden sollen.

4.1.4.3 Schlechte Navigation

Statische Visualisierungen nutzen die Hierarchie der biochemischen Reaktionen nur ungenügend aus. So zeigt Abbildung 2.5 auf Seite 20 einen Überblick über alle biochemischen Reaktionen im Zentralstoffwechsel. Dabei sind nur die Schlüsselsubstanzen oder Substanzklassen und zwischen ihnen existierende Stoffwechselwege dargestellt. Das Poster *Biochemical Pathways* dagegen stellt biochemische Reaktionen auf der darunter liegenden Hierarchieebene dar (vergleiche auch Abb. 1.1 auf Seite 4). Oft werden jedoch Darstellungen gewünscht, bei denen Reaktionen im Kontext der sie umgebenden Stoffwechselwege gezeigt werden oder bei denen der Anwender eine Sicht vorgeben kann. Statische Visualisierungen leisten dies nicht und sind daher für eine Navigation durch Stoffwechselwege und besonders für kontexterhaltende Navigation ungeeignet.

4.1.4.4 Bedingte Eignung für Informationssysteme

Bei der Verwendung in Informationssystemen haben statische Visualisierungen weitere Nachteile: Es ist unmöglich, für alle Datenkombinationen, die sich aus Datenbankanfragen ergeben können, Abbildungen schon im Vorfeld bereitzustellen. Auch eine tägliche Aktualisierung der Darstellungen auf Grund der Datenaktualisierung ist nicht möglich.

Zusammenfassend lässt sich feststellen, dass statische Visualisierungen für die Darstellung von biochemischen Reaktionsnetzen in Büchern sehr gut, für Darstellungen in Informationssystemen dagegen kaum geeignet sind.

4.2 Umsetzung der Anforderungen durch dynamische Visualisierungen

4.2.1 Charakterisierung dynamischer Visualisierungen

In Abschnitt 4.1 wurde festgestellt, dass statische Visualisierungen biochemischer Reaktionsnetze unzureichend sind. Insbesondere für die interaktive Exploration der Daten in Informationssystemen ist eine andere Art der Erzeugung der Visualisierungen erforderlich, bei denen Zeichnungen *zum Zeitpunkt* erzeugt werden, zu dem ein Anwender sie benötigt. Dabei sollten spezielle Wünsche des Anwenders berücksichtigt werden, beispielsweise die Darstellungen vorgegebener Ausschnitte, spezielle Sichten oder die Richtung der Zeichnung. Diese Visualisierungen werden als *dynamische Visualisierungen* bezeichnet und sind wie folgt charakterisiert:

1. Sie werden *zum Zeitpunkt* der Verwendung als Abbild der *aktuellen* Daten (bzw. eines Teils der Daten) erzeugt.
2. Sie liefern eine (zumindest teilweise) vom *Anwender* festgelegte Sicht auf die Daten.
3. Sie werden im Allgemeinen *einmal* verwendet.
4. Sie sind *leicht veränderbar*.

Diese Art der Visualisierung lässt sich in Systemen verwenden, in denen der Publikationsprozess einfach und mit geringen Kosten erfolgen kann, also insbesondere in Informationssystemen.

4.2.2 Bekannten Verfahren zum Zeichnen von Graphen

In diesem Abschnitt werden bekannte Verfahren zum Zeichnen von Graphen auf ihre Eignung zur Visualisierung biochemischer Reaktionsnetze untersucht. Dazu wurden verschiedene Reaktionsnetze mit Hilfe der bereits in Abschnitt 2.2.3.3 vorgestellten Graph-Zeichenverfahren visualisiert. Die Darstellungen wurden anschließend von den beteiligten Biochemikern auf ihre Verständlichkeit bewertet. Die Abbildungen 4.4 und 4.5 enthalten Zeichnungen desselben Reaktionswegs, die mit einem ebenenweisen, einem kräftebasierten und einem orthogonalen Zeichenverfahren erzeugt wurden. Zum Vergleich ist eine Darstellung entsprechend den hier aufgestellten Anforderungen zugefügt.

4.2.2.1 Modellierung

Die Reaktionsnetze wurden als benannte Graphen modelliert. Substanzen, Cosubstanzen, Enzyme und Reaktionen entsprechen den Knoten, diese sind jeweils mit dem Namen der (Co-)Substanz bzw. des Enzyms benannt. Kanten verbinden Substanzen, Cosubstanzen und Enzyme mit der zugehörigen Reaktion.

Diese Modellierung ist eine vereinfachte Repräsentation biochemischer Reaktionsnetze, beispielsweise geht dabei die Unterscheidung zwischen Substanzen und Cosubstanzen verloren. Dies wurde aus zwei Gründen erlaubt: Erstens war die Frage zu klären, ob die Unterscheidung zwischen Substanzen und Cosubstanzen wirklich notwendig ist. Und zweitens können herkömmliche Zeichenverfahren von Graphen im Allgemeinen nicht zwischen verschiedenen Knoten- oder Kantentypen unterscheiden. Ausgenommen sind Verfahren für spezielle Anwendungen, z. B. das Zeichnen von UML³-Diagrammen [See97, Mar98], die beispielsweise verschiedene Typen für Kanten berücksich-

³Unified Modeling Language

tigen können. Durch diese Verfahren werden jedoch keine brauchbaren Darstellungen geliefert, da sich die Anforderungen an die Darstellung von UML-Diagrammen stark von den hier aufgestellten Anforderungen unterscheiden. Eine andere Anwendung sind Petri-Netze⁴, die eine ähnliche Struktur wie biochemische Reaktionsnetze haben. So lassen sich die Substanzen und Cosubstanzen als Plätze, die Reaktionen als Transitionen modellieren. Tatsächlich werden Petri-Netze auch zur Modellierung biochemischer Reaktionen verwendet, worauf in Abschnitt 5.4.4 noch eingegangen wird. Für Petri-Netze stellt Stübinger [Stü97] ein Zeichenverfahren vor, das jedoch auf planare Graphen eingeschränkt ist. Zudem unterscheiden sich die Anforderungen bei der Visualisierung von Petri-Netzen von den Anforderungen bei der Darstellung biochemischer Reaktionsnetze, so dass auch dieses Verfahren keine brauchbaren Zeichnungen liefert.

Aus diesen Gründen und um den Biochemikern verschiedene Arten der Darstellung zu präsentieren, wurde bewusst diese einfache Modellierung gewählt. Dadurch konnten viele der bekannten Zeichenverfahren für Graphen angewandt werden. In Abbildung 4.4(a) wurde ein kräftebasiertes Verfahren (Universal Spring Embedder [For99]) angewandt, in 4.4(b) ein ebenenorientiertes Verfahren (nach [STT81]). Die Abbildung 4.4(a) zeigt vom Ergebnis eine orthogonale Zeichnung, die durch das *Iterative Constraint Spring Embedder* Verfahren erzeugt wurde. Alle Darstellungen wurden mit dem Graphlet-System [Gra01a, Him00] erzeugt, was den Vorteil hat, dass für Substanzen auch die Strukturformeln als Bild dargestellt werden konnten. Der Prozess der Erzeugung war unwichtig, den Anwendern wurden nur die Ergebnisse in Form von Ausdrucken präsentiert. Dabei wurden vor der Präsentation in allen Zeichnungen Knotenüberdeckungen manuell beseitigt. Es wurden keine planaren Zeichenverfahren untersucht, da biochemische Reaktionsnetze nicht planar⁵ sein können.

4.2.2.2 Bewertung der Zeichenverfahren

Die im Rahmen der bereits beschriebenen Untersuchungen befragten Biochemiker waren überrascht von den Möglichkeiten automatischer Visualisierungen. Sie bezeichneten jedes der vorgestellten Verfahren als interessant, um überhaupt eine Visualisierung der Daten zu erhalten. Die Verfahren wurden im Einzelnen wie folgt bewertet:

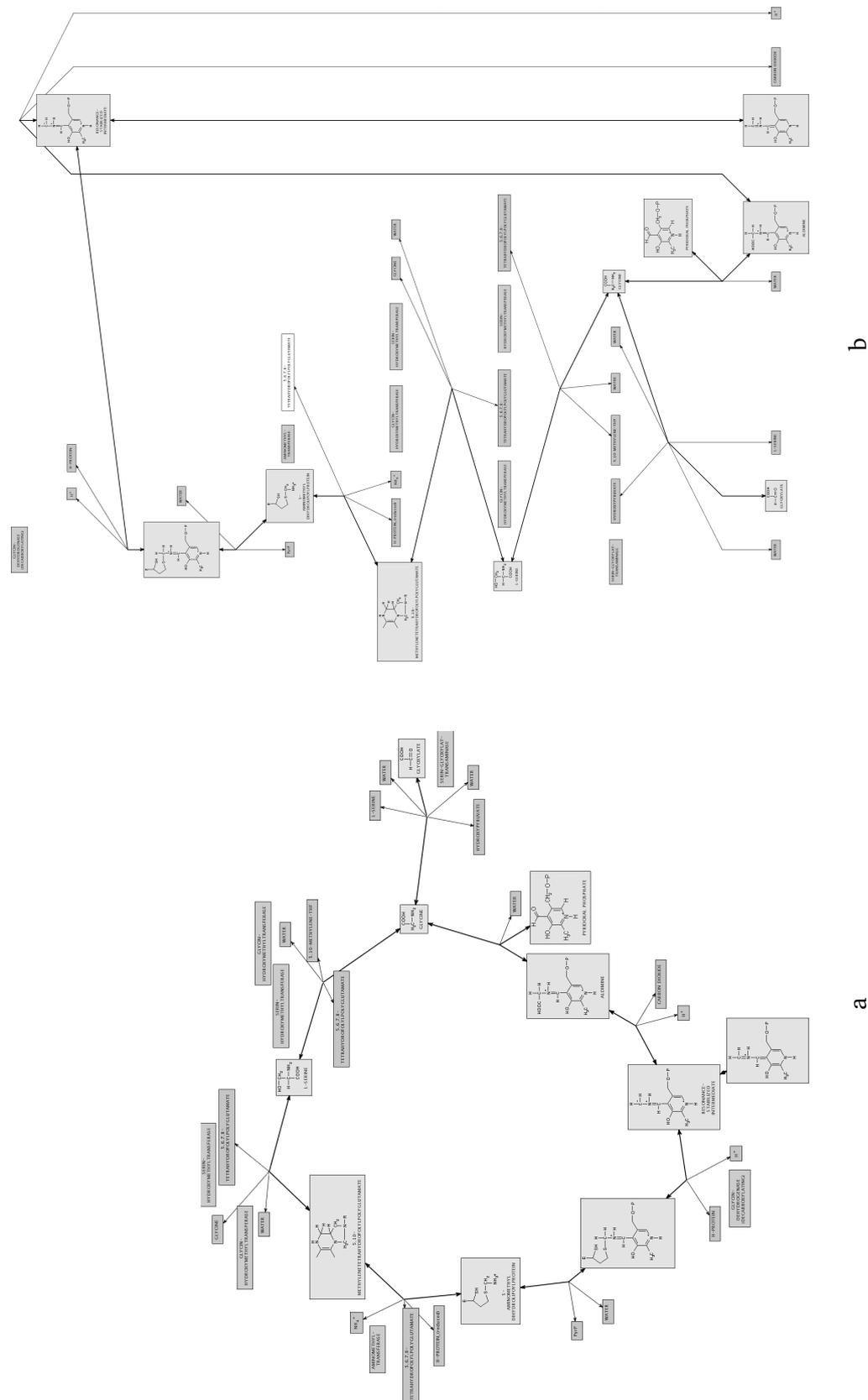
4.2.2.2.1 Kräftebasierte Zeichenverfahren

Positiv wurden die relativ geradlinige Darstellung von nicht verzweigenden Reaktionsnetzen (Wegen) und das Hervorheben von Zyklen bewertet. Zugleich wurde jedoch als stark nachteilig empfunden, dass auch zufällig verbundene Reaktionswege wie Zyklen kreisförmig gezeichnet werden (siehe auch Abbildung 4.4(a)). Als negativ wurde weiterhin die ungleichmäßige Verteilung der Objekte betrachtet.⁶ Ebenfalls bemängelt wurde, dass die Reihenfolge der Cosubstanzen nicht der Reihenfolge entspricht, in der sie in die Reaktion eintreten bzw. aus ihr hervorgehen und dass Cosubstanzen wahllos auf verschiedenen Seiten des Reaktionspfeils platziert werden.

⁴Ein Petri-Netz $PN = (P, T, A, W, M)$ ist ein bipartiter Graph mit: P ist die Menge der Plätze (Bedingungen, Stellen, Zustände); T ist die Menge der Transitionen (Ereignisse, Instanzen); $A \subseteq (P \times T) \cup (T \times P)$ ist die Menge der Kanten; $W : A \rightarrow \mathbb{N}$ ist eine Gewichtsfunktion und $M : P \rightarrow \mathbb{N}$ ist die initiale Markierung (siehe auch [Bau90, Rei86]).

⁵Ein Graph bzw. ein Reaktionsnetz heißt planar, wenn er in der Ebene ohne Kantenkreuzungen gezeichnet werden kann.

⁶Der Springembedder-Algorithmus liefert Zeichnungen, bei denen Knoten, die sich in sehr dichten Teilgraphen befinden, näher zueinander platziert werden als Knoten, die in lichten Teilgraphen enthalten sind.



b

a

Abbildung 4.4: Zeichenverfahren für Graphen (Teil 1). Darstellung eines Reaktionsnetzes mittels (a) eines kräftebasierten Zeichenverfahrens und (b) eines ebenenweisen Zeichenverfahrens.

4.2.2.2.2 Ebenenweise Zeichenverfahren

Positiv wurde die Anordnung der Reaktionen entsprechend ihrer zeitlichen Reihenfolge in der überwiegenden Richtung von oben nach unten bewertet. Kritikpunkte waren die Reihenfolge der Cosubstanzen, die nicht immer ihrer Reihenfolge in den Reaktionen entsprachen, die Platzierung der Enzyme sowie die Darstellung zyklischer Strukturen, die ebenfalls fast vollständig von oben nach unten mit einer Rückwärtskante gezeichnet werden.

4.2.2.2.3 Orthogonale Zeichenverfahren

Wie in den anderen Darstellungen wurde auch bei diesen Zeichnungen die Möglichkeit der automatischen Darstellung der Daten gelobt. Sonst überwog eine negative Bewertung: Die Reihenfolge der Reaktionen ist nicht sofort ersichtlich, die Reihenfolge der Cosubstanzen entspricht nicht ihrer Reihenfolge in den Reaktionen, Reaktionswege sind nicht immer geradlinig, sondern wechseln ihre Richtung sehr oft und zyklische Strukturen wie der Citratzyklus sind nur schwer als solche erkennbar.

4.2.2.2.4 Zusammenfassung

Insgesamt wurde keines der vorgestellten Verfahren als ausreichend für die Erfordernisse der Biochemiker bewertet. Bei den Visualisierungen wurden ebenenweise Zeichnungen bevorzugt, danach folgten mittels kräftebasierter Verfahren erzeugte Darstellungen. Orthogonale Zeichenverfahren schnitten bei der Bewertung am schlechtesten ab.

Es sei betont, dass die Biochemiker von den Möglichkeiten automatischer Zeichenverfahren im Hinblick auf die großen zu bearbeitenden Datenmengen sehr beeindruckt waren und entsprechende Verfahren begrüßten. Zugleich wurden jedoch alle vorgestellten Verfahren als unzureichend für die Zwecke der Biochemie bewertet. Insbesondere wurde Wert auf die Unterscheidung von Substanzen und Cosubstanzen durch Anordnung der Cosubstanzen neben dem Reaktionspfeil in der Reihenfolge ihres Reagierens sowie auf die zeitliche Reihenfolge der Reaktionen gelegt.

4.2.3 Vorkommen dynamischer Visualisierungen

4.2.3.1 Informationssysteme

4.2.3.1.1 EcoCyc und MetaCyc

Karp und Paley stellten mit *EcoCyc* und *MetaCyc* [Eco00, KOP96, KP94b, KR97, KRS⁺00] die ersten Systeme vor, welche lineare, verzweigte und zyklische Stoffwechselwege und Reaktionsnetze dynamisch darstellen. Die Reaktionsnetze werden dabei als getypte Graphen (mit den Typen *Substanz*, *Cosubstanz* und *Enzym*) modelliert, wobei Substanzen, Cosubstanzen und Enzyme den Knoten entsprechen. Die Kanten verbinden einerseits Substanzen untereinander, diese bilden die Struktur des Reaktionsnetzes, andererseits Gruppen von Cosubstanzen miteinander [KP94a].

Die Platzierung der Komponenten linearer Sequenzen erfolgt entlang von Zick-Zack-Linien, Verzweigungen werden als Bäume und zyklische Strukturen als Kreise dargestellt. Komplexe Reaktionsnetze werden in die eben genannten Teilgraphen zerlegt. Diese Teilgraphen werden mit Zeichenalgorithmen des Programms *Grasper-CL* [LC79, KLSW94] gezeichnet und anschließend wieder zu einer Zeichnung zusammengefügt. Dabei wird nur der Graph verwendet, der aus den Substanzen und diese verbindenden Kanten besteht. Die Knotenabstände werden so gewählt, dass für da-

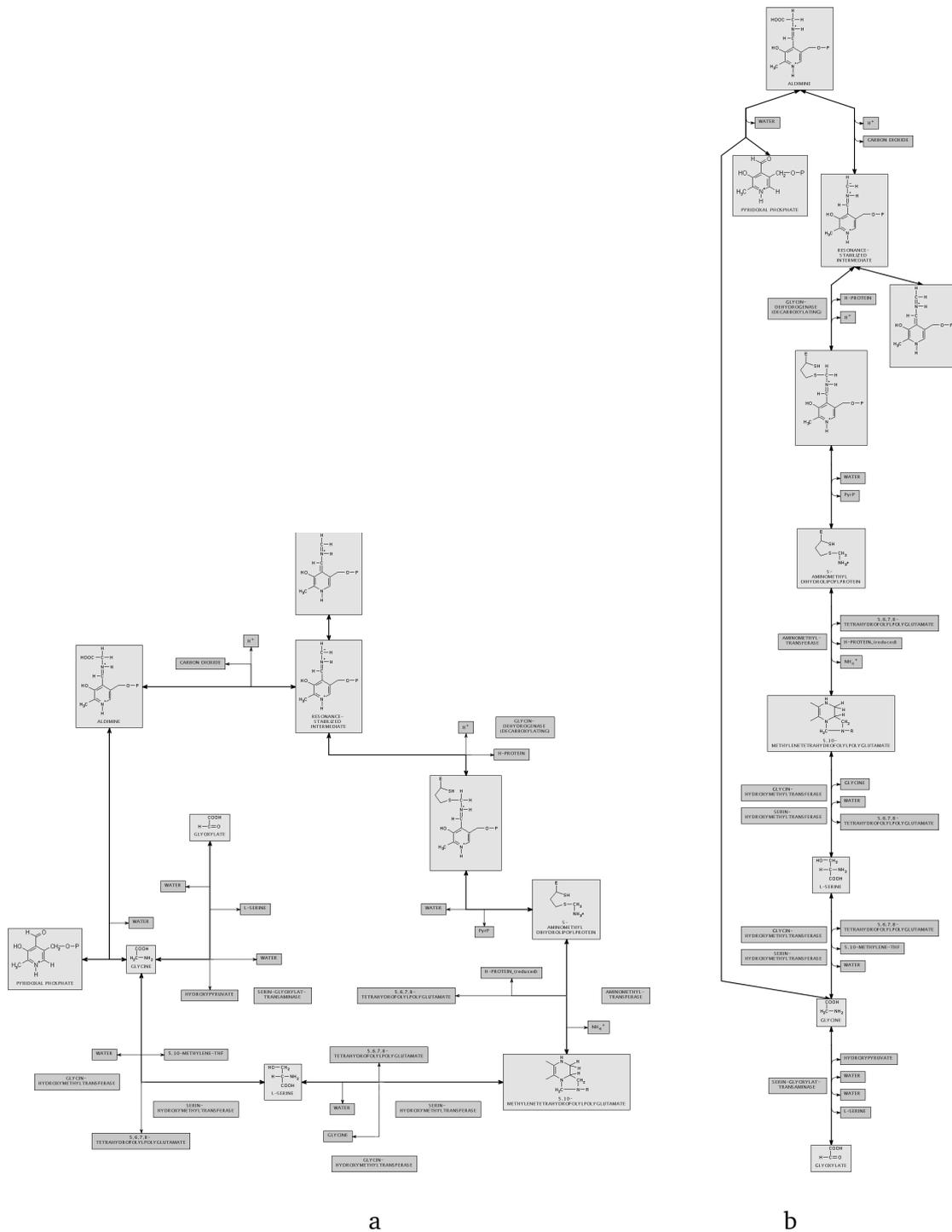


Abbildung 4.5: Zeichenverfahren für Graphen (Teil 2). Darstellung des Reaktionsnetzes aus Abbildung 4.4 als (a) orthogonale Zeichnung und (b) entsprechend den Anforderungen für Darstellungen biochemischer Reaktionsnetze.

zwischenliegende Cosubstanzen und Enzyme Platz bleibt. Deren Platzierung wird anschließend bestimmt, sie können als Benennung der Reaktionskante betrachtet werden.

Das System wird kommerziell vertrieben. Da es trotz mehrerer Versuche nicht möglich war, eine Testversion des Programms zu erhalten, ist keine abschließende Bewertung möglich. Die Qualität der erzeugten Zeichnungen lässt sich nur durch [KP94a] und auf Grund der öffentlich zugänglichen Beispiele der Darstellung des Stoffwechsels *Synthese von L-Methionin* (vgl. Abb. 4.6, in der zwei Sichten dargestellt sind) bewerten.

Das Verfahren erfüllt eine Reihe der in den Abschnitten 3.2 - 3.4 aufgestellten Anforderungen. So können Substanzen sowohl mit Namen wie mit ihren Strukturformeln dargestellt werden. Cosubstanzen und Enzyme werden entsprechend den Anforderungen neben den Reaktionspfeil platziert. Die Darstellung verschiedener Sichten, beispielsweise mit und ohne Cosubstanzen, ist möglich.

Das Visualisierungsverfahren hat aber auch eine Reihe Schwächen. In der Abbildung 4.6 fallen bereits Probleme beim Zusammenfügen der Komponenten auf, die sich durch Überschneidungen (siehe Kante zu *pyruvate NH₃*) äußern. Dies liegt daran, dass bei der Anordnung der Substanzen die Cosubstanzen nicht berücksichtigt werden. Ein weiteres Problem ist, dass die Reihenfolge der Cosubstanzen nicht eingehalten wird, da alle Coedukte zu einem Vorkommen, alle Coprodukte zu einem zweiten Vorkommen zusammengefasst werden. Darstellungen wie in Abbildung 2.3(b) auf Seite 16 sind damit nicht möglich. Insbesondere können durch *EcoCyc* erzeugte Darstellungen chemische Sachverhalte falsch wiedergeben. Reaktionswege haben keine feste Richtung, sondern können innerhalb einer Zeichnung von oben nach unten, links nach rechts oder als Zick-Zack-Linien verlaufen. Auch die Darstellung größerer Reaktionsnetze lässt sich nicht beurteilen. Es ist aber zu vermuten, dass die Qualität der Zeichnungen rasch schlechter wird, Karp und Paley schreiben in [KP94a] dazu:

„Results on the few nontree and complex superpathways are not particularly good.“

Dies liegt sicher auch daran, dass bei ihren ebenenweisen Zeichnungen komplexer Reaktionsnetze keine Verfahren zur Kreuzungsminimierung zwischen Kanten angewendet werden [KP94a].

Abschließend sollen die Navigationsmöglichkeiten betrachtet werden: Es existiert eine Übersichts-darstellung, von der aus Verweise auf die Stoffwechselwege bestehen. Diese ermöglicht in begrenztem Umfang die Navigation zwischen Reaktionswegen und der Übersichts-darstellung. Es gibt aber keine kontexterhaltende Navigation durch Reaktionsnetze. So ist das Hinzufügen von Reaktionswegen zu existierenden Visualisierungen oder die Verfeinerung von Teilen des dargestellten Reaktionsnetzes nicht möglich.

Zusammenfassend lässt sich feststellen, dass mit *EcoCyc* ein sehr früher und interessanter Versuch der automatischen Visualisierung biochemischer Reaktionsnetze unternommen wurde, der sich aber auf Grund der beschriebenen Unzulänglichkeiten nicht durchgesetzt hat.

4.2.3.1.2 PFBP

Existierende Algorithmen zum Zeichnen von Graphen sollen im *Protein Function and Biochemical Pathways Project (PFBP)* [PFB01, vNM⁺00] zur Darstellung von Reaktionswegen verwendet werden. Zu den verwendeten Zeichenverfahren waren keine Informationen erhältlich, der WWW-Zugang ist noch in Entwicklung. Die folgenden Betrachtungen der Darstellungsqualität beruhen daher auf Aussagen von [PFB01] und Tests der zugänglichen Teile des Systems. Neben statischen Visualisierungen

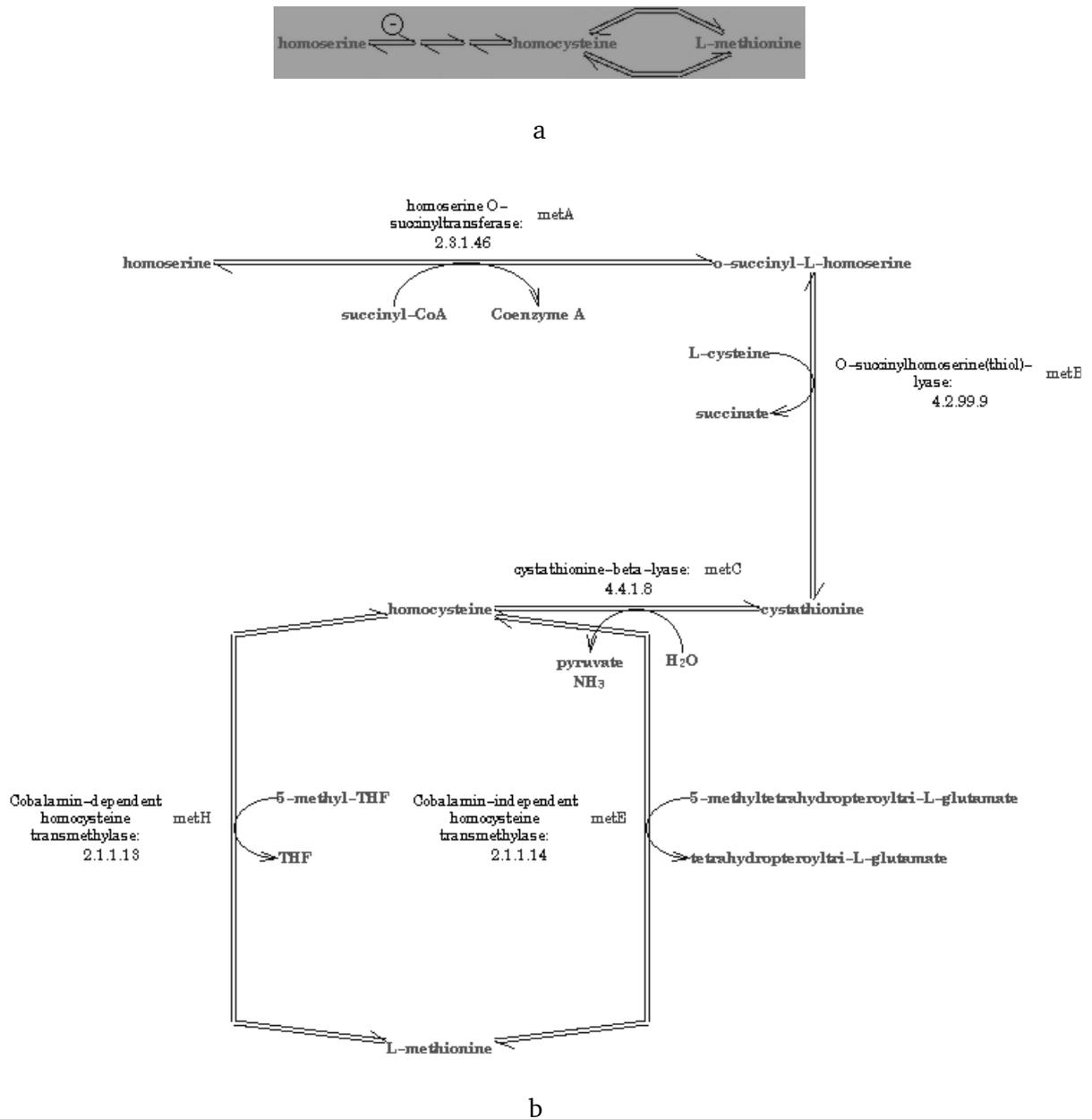


Abbildung 4.6: **Dynamische Visualisierung biochemischer Reaktionsnetze (Teil 1)**. EcoCyc: Zwei Sichten auf den Stoffwechselweg *Synthese von L-Methionin*. (a) Übersichtsreaktion, (b) detaillierte Sicht.

gen von Stoffwechselwegen sollen einige Stoffwechselwege mittels automatischer Zeichenverfahren darstellbar sein. Dazu heißt es in [PFB01]

„Using this library⁷ graphical diagrams of the 30 or so PFBP pathways, generated using an automatic pathway layout routine, can be displayed upon request and interactively manipulated.“

Diese Darstellungen konnten nicht analysiert werden, da dieser Teil von *PFBP* nicht zugänglich war, weshalb hier nur die Darstellung von Reaktionswegen zwischen zwei Substanzen untersucht wird. In *PFBP* können Wege zwischen zwei Substanzen berechnet und dargestellt werden. Einfache Darstellungen wie die in Abbildung 4.7 werden dabei dynamisch erzeugt, was ein großer Vorteil gegenüber textuellen Darstellungen solcher berechneten Wege ist.

Der Nachteil dieser Darstellungen ist, dass in den Zeichnungen Reaktionswege, auch zyklische wie der *Citratzyklus*, linear angeordnet werden, Reaktionsnetze können gar nicht dargestellt werden. Es gibt keine Möglichkeit der Navigation oder der Erzeugung verschiedener Sichten. Zudem werden Substanzen nur mit ihrem Namen, aber nicht mit ihrer Strukturformel dargestellt. Substanzen und Cosubstanzen werden nicht unterschieden und die Reihenfolge der Cosubstanzen entspricht nicht immer der Reihenfolge in der Reaktion, was zu chemisch falschen Darstellungen führt.

Eine abschließende Bewertung ist hier nicht möglich. Das momentane Verfahren ist unzureichend und von der Qualität deutlich schlechter als beispielsweise *EcoCyc*. Es ist aber zu erwarten, dass in der Zukunft leistungsfähigere Zeichenverfahren entwickelt werden, auf der Web-Seite [PFB01] heißt es dazu:

„Note however that the provided layout algorithm is still in its very early stages, and the results are far from perfect . . . the display of branched and cyclic pathways is still not solved. These are displayed linearly with some connections missing.“

4.2.3.1.3 BioJake

BioJake [Bio00, SMKS99] bietet einen Editor, mit dem Darstellungen von Reaktionen nachträglich verändert werden können. Mittels der in *Graphviz* realisierten Zeichenverfahren [GKNV93, GN00, Gra01b] können automatische Visualisierungen erzeugt werden, siehe Abbildung 4.8. Dabei werden jedoch nur Substanzen und Enzyme unterschieden, Cosubstanzen werden wie Substanzen dargestellt und die Reihenfolge der Cosubstanzen wird nicht berücksichtigt. Die Komponenten der Reaktionen werden durch Icons repräsentiert, die Verwendung von Bildern, z.B. für Strukturformeln, aber auch von Namen für Substanzen ist nicht möglich. Die erhaltenen Visualisierungen sind deshalb weit von den Anforderungen entfernt.

Das vorhandene Zeichenverfahren soll nicht weiterentwickelt werden [Won00], statt dessen überlegen die Autoren, ein dreidimensionales Zeichenverfahren zu realisieren [SMKS99]. Die Probleme bei der Verwendung herkömmlicher Graph-Zeichenverfahren wurden bereits in Abschnitt 4.2.2 untersucht, sie treffen auch hier zu. Auf Grund der beschriebenen Einschränkungen erfüllt *BioJake* nicht die Anforderungen an Visualisierungen biochemischer Reaktionsnetze.

⁷Die Tom Sawyer Java Graphic library, Anmerkung des Autors.

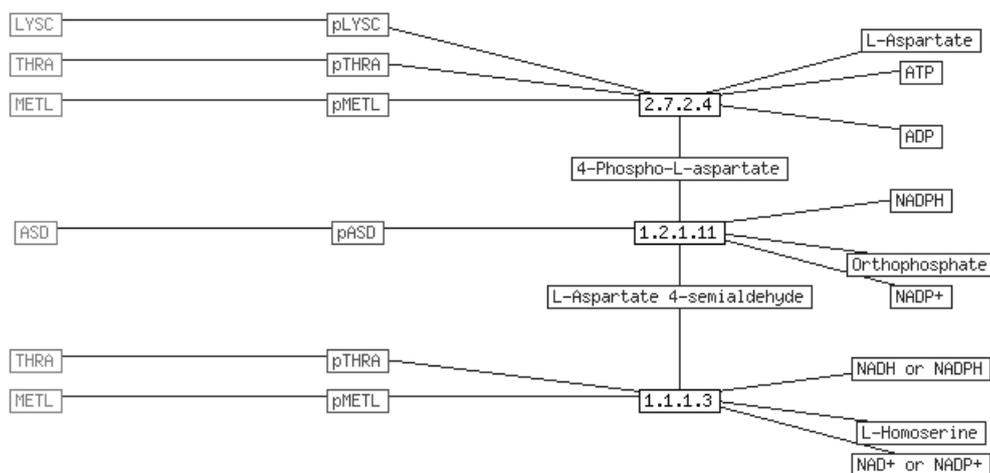


Abbildung 4.7: **Dynamische Visualisierung biochemischer Reaktionsnetze (Teil 2)**. PFBP: Darstellung des Reaktionswegs von *L-Aspartat* nach *L-Homoserin*. Nur die rechte Seite stellt biochemische Reaktionen dar.

4.2.3.1.4 PathDB

Das unter Leitung von Mendes entwickelte Informationssystem *PathDB* [Men00, Pat00] verwendet ebenfalls die Bibliothek von Tom Sawyer [Tom00] zum Zeichnen von Stoffwechselwegen. Die Stoffwechselwege werden durch Graphen modelliert, Cosubstanzen und Enzyme werden durch Kantenbenennungen repräsentiert. Einerseits sind Stoffwechselwege mit Koordinaten für die Objekte annotiert, dies entspricht einer statischen Visualisierung. Andererseits lassen sich diese Stoffwechselwege durch den Anwender oder die beiden Zeichenverfahren *zirkuläres Zeichnen* [ST99] und *ebenenweises Zeichnen* [STT81, GKNV93] neu platzieren (siehe Abb. 4.9). Verschiedene Sichten auf Reaktionswege sind möglich, die Visualisierung wird dabei aber nicht neu berechnet, sondern es werden beispielsweise Cosubstanzen aus der aktuellen Zeichnung ausgeblendet.

Bereits in Abschnitt 4.2.2 wurde festgestellt, dass die Anwendung herkömmlicher Graph-Zeichnverfahren unzureichend ist. Auch die hier verwendete Erweiterung der Darstellung von Cosubstanzen und Enzymen mittels Kantenbenennungen ist eine unzulängliche Lösung, wie an der Reaktion von *Isocitrat* nach *2-Oxoglutarat* ersichtlich ist (Abb. 4.9(b)). Nachteile sind zudem, dass Knoten alle die gleiche Größe haben und für die Substanzen nur Namen, aber keine Strukturformeln dargestellt werden. Eine Erweiterung der Stoffwechselwege, interaktive Navigation oder die Suche und Darstellung des Reaktionsnetzes zwischen zwei beliebigen Substanzen sind in *PathDB* nicht möglich. Auch dieser Ansatz erfüllt also die aufgestellten Anforderungen nicht.

4.2.3.1.5 yFiles und UM-BBD

Als Anwendung der *yFiles* [yFi00] stellt Eiglsperger dynamische Visualisierungen der Stoffwechselwege der *University of Minnesota Biocatalysis/Biodegradation Database UM-BBD* (siehe Abschnitt

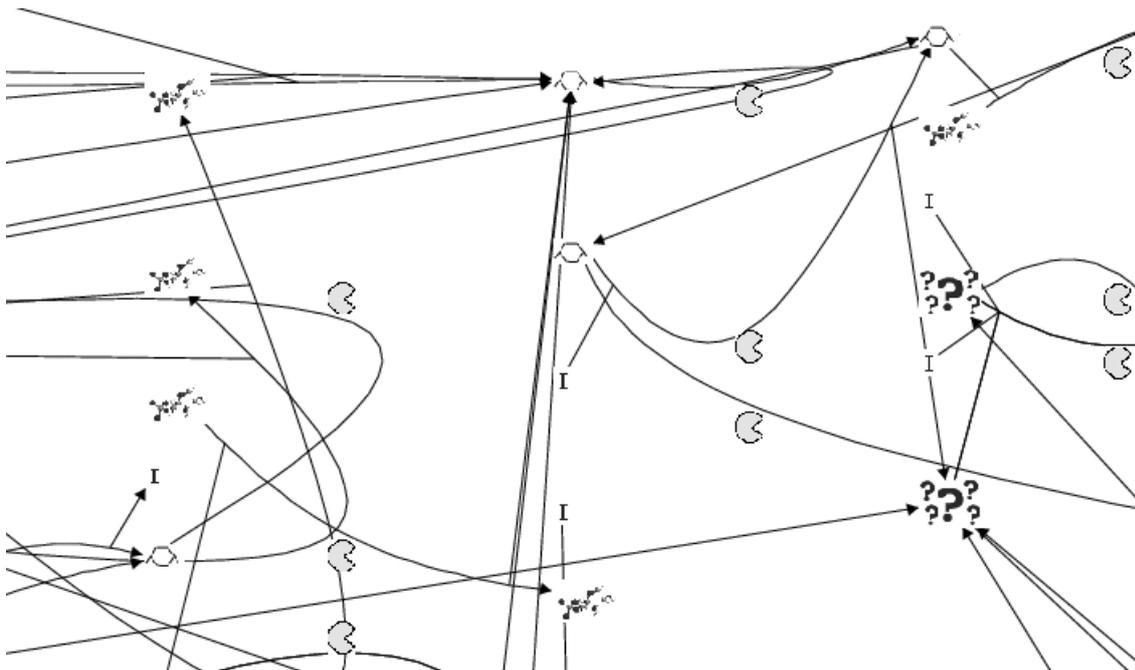


Abbildung 4.8: **Dynamische Visualisierung biochemischer Reaktionsnetze (Teil 3)**. BioJake: Ausschnitt aus der Visualisierung des Reaktionswegs *Glycolyse und Gluconeogenese*. Da Namen fehlen, ist diese Darstellung sehr unübersichtlich (Namen sind nur durch Klick auf die Icons erhältlich).

4.1.2.2.4) vor. Die Modellierung der Reaktionsnetze erfolgt wie beim eben betrachteten *PathDB*-System. Es können orthogonale und ebenenweise Zeichenverfahren angewandt werden. Dabei treten dieselben Probleme wie bei *PathDB* auf: Die Reihenfolge der Cosubstanzen wird u. U. nicht korrekt wiedergegeben, für die Substanzen werden nur Namen, aber keine Strukturformeln dargestellt und Navigation wird nicht unterstützt. Zudem werden offene und geschlossene Zyklen nicht entsprechend Anforderung 3.4 dargestellt.

4.2.3.2 Graph Drawing Contest

Die Anwendung existierender Algorithmen zum Zeichnen von Graphen auf biochemische Reaktionsnetze bzw. die Entwicklung angepasster Verfahren war eine der Herausforderungen des *Graph Drawing Contest '99* [BJM⁺99]. Aufgabe war die möglichst automatische Visualisierung des Übersichtsbildes 2.5 auf Seite 20. Die beiden Siegerlösungen sind in Abbildung 4.10 dargestellt.

Das für den Wettbewerb gewählte Reaktionsnetz ist insofern einfach, als keine Cosubstanzen und Enzymnamen auftreten. Zudem musste nur dieses spezielle Reaktionsnetz gezeichnet werden, es konnte also ein dafür geeignetes Verfahren gewählt werden. Da keine Cosubstanzen und Enzyme enthalten sind, spielen lokale Anforderungen (die Platzierung der Komponenten) keine Rolle. Es wurden keine Anforderungen an die Zeichnung gestellt, so dass die Teilnehmer die Begrenzung der Fläche und das Herausarbeiten bekannter Strukturen wie den Citratzyklus in den Mittelpunkt stellten. Für die dabei verwendeten kräftebasierten bzw. orthogonalen Zeichenverfahren gelten die

Bemerkungen aus Abschnitt 4.2.2.

4.2.3.3 Zu biochemischen Reaktionsnetzen ähnliche Strukturen

Neben biochemischen Reaktionsnetzen gibt es weitere Netze in der Biologie, die sich als Graphen darstellen lassen und für die Visualisierungen untersucht und automatische Zeichenverfahren entwickelt werden. So beschäftigen sich Basalaj und Eilbeck [BE99] mit der Darstellung von Proteinnetzen⁸ und deren Zeichnung durch kräftebasierte Verfahren.

Im Gegensatz zu biochemischen Reaktionsnetzen sind Proteinnetze von sehr einfacher Struktur. So kommen beispielsweise keine neben die Kante zu platzierenden Cosubstanzen vor und die Typen aller Objekte sind gleich, jeder Knoten repräsentiert im Proteinnetz ein gleichberechtigtes Protein. Zudem gibt es an die Darstellung dieser Netze keine speziellen Anforderungen wie an die Darstellung biochemischer Reaktionsnetze. Deshalb werden zu ihrer Visualisierung bekannte Zeichenverfahren genutzt, deren Unzulänglichkeit für die Darstellung biochemischer Reaktionsnetze bereits dargestellt wurde.

In [INK95, TINK98] werden Signalübertragungsnetze⁹ untersucht. Diese beschreiben innere Zellreaktionen auf äußere Reize. Dabei können neben biochemischen Reaktionen auch Bildung von Proteinkomplexen und Transportprozesse durch Membranen auftreten. Signalübertragungsnetze lassen sich ebenfalls als Graphen darstellen. Wong [Won01] verwendet für die Visualisierung von Signalübertragungsnetzen das ebenenweise Zeichenverfahren aus *Graphviz* [GKNV93, GN00]. Die damit erzielten Visualisierungen sind jedoch unzureichend. Sie entsprechen Darstellungen ähnlich zu Abbildung 4.8, die bereits als nicht ausreichend zur Visualisierung biochemischer Reaktionsnetze charakterisiert wurden.

4.2.4 Vorteile dynamischer Visualisierungen

Der wesentliche Vorteil dynamischer gegenüber statischen Visualisierungen ist die Möglichkeit, beliebige Ausschnitte der aktuellen Daten und beliebige Sichten auf diese darzustellen. Dies gibt dem Anwender viele Freiheiten und erleichtert das Verständnis der komplexen Zusammenhänge in biochemischen Reaktionsnetzen. Da man nun nicht mehr gezwungen ist, möglichst viel Information in einer Abbildung darzustellen, sind weniger komplexe und dadurch übersichtlichere Darstellungen möglich. Sind Verfahren zur dynamischen Visualisierung einmal vorhanden, so ist die Erstellung der Abbildung zudem wesentlich schneller und preiswerter. Dynamische Visualisierungen ermöglichen die interaktive Exploration biochemischer Reaktionsnetze. Sie sind ideal für Informationssysteme geeignet.

4.2.5 Nachteile dynamischer Visualisierungen

Allen beschriebenen Verfahren, bei denen existierende Zeichenalgorithmen verwendet werden, ist gemein, dass sie keine an die Erfordernisse der Biochemie angepassten Darstellungen liefern. Sie haben sich deshalb auch nicht zur Visualisierung biochemischer Reaktionsnetze durchgesetzt. Ein Grund dafür ist, dass die Modellierung der Reaktionsnetze nicht adäquat gewählt wurde; gesucht

⁸In Proteinnetzen (*Protein-Protein interaction pathways*) sind Proteine als Knoten, Beziehungen zwischen Proteinen, z. B. Aktivierung oder Hemmung der Aktivität der Proteine, als Kanten dargestellt.

⁹Signalübertragungsnetze werden in Abschnitt 9.2 vorgestellt.

sind Modellierungen, die die speziellen Anforderungen wie Unterscheidung von Komponenten und Hierarchisierung unterstützen. Entscheidender Nachteil ist aber der große Aufwand, den die Entwicklung angepasster Zeichenverfahren erfordert.

4.3 Fazit

Weder existierende statische noch dynamische Visualisierungen erfüllen die Anforderungen 3.1–3.5. Zur Visualisierung biochemischer Reaktionsnetze entsprechend diesen Anforderungen ist deshalb die Entwicklung eines neuen Verfahrens notwendig.

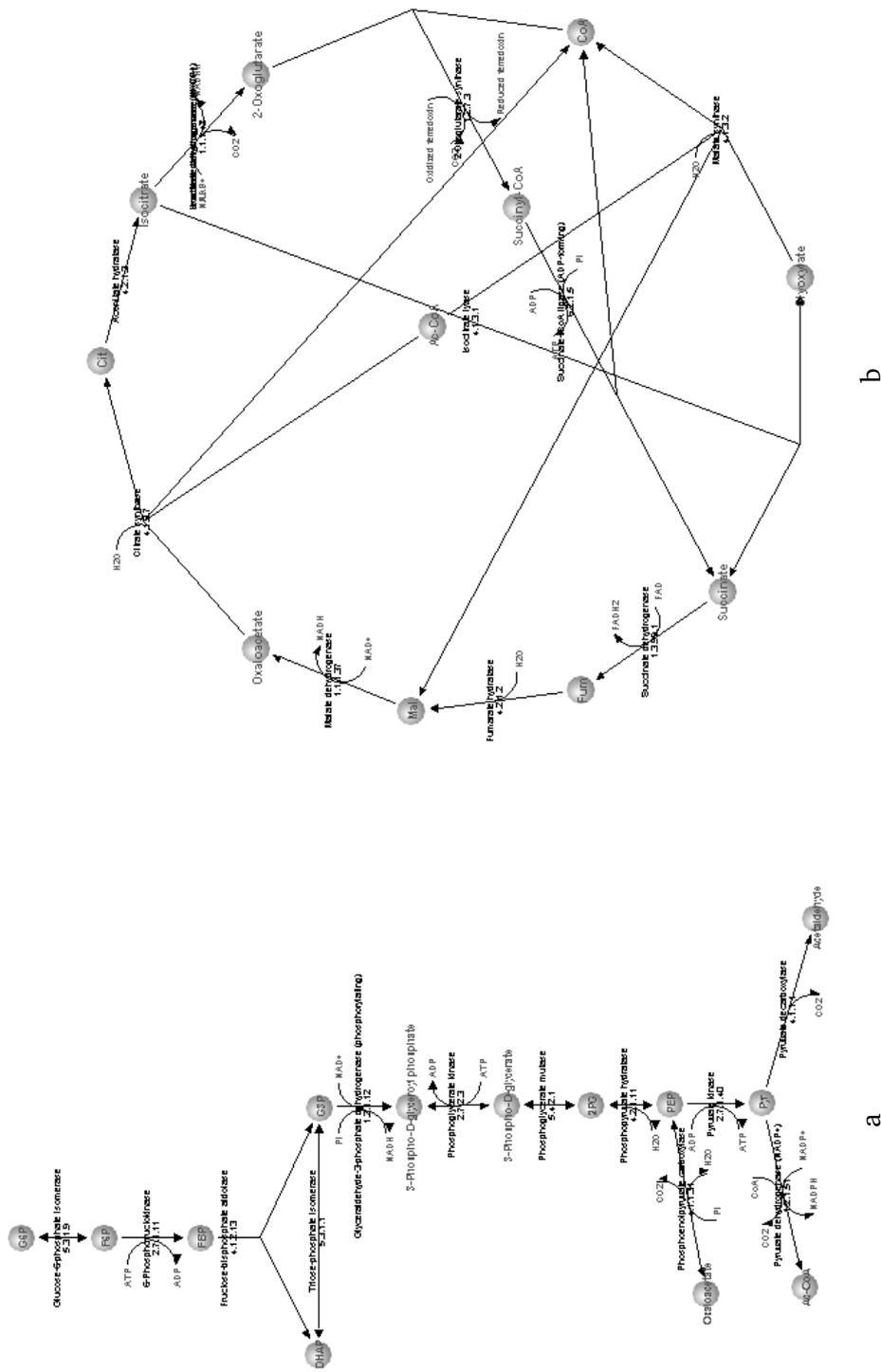
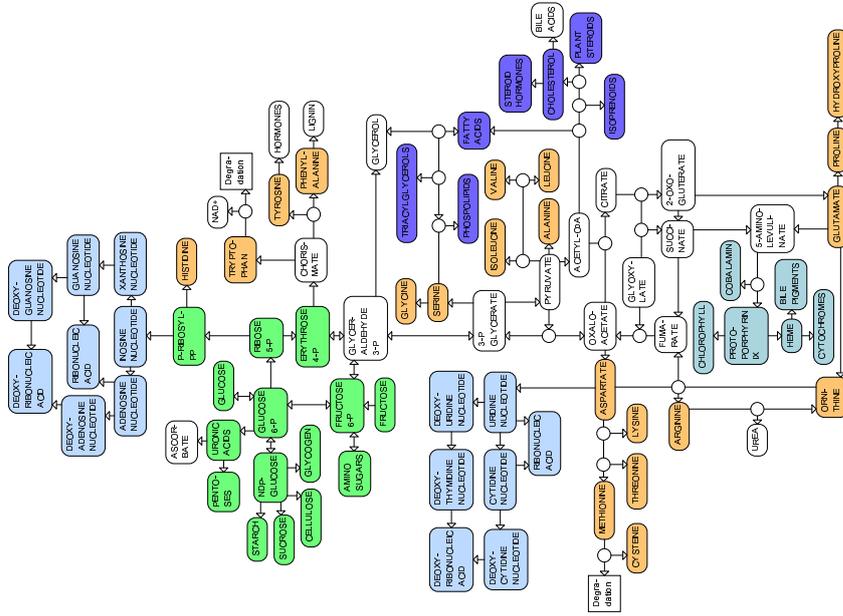
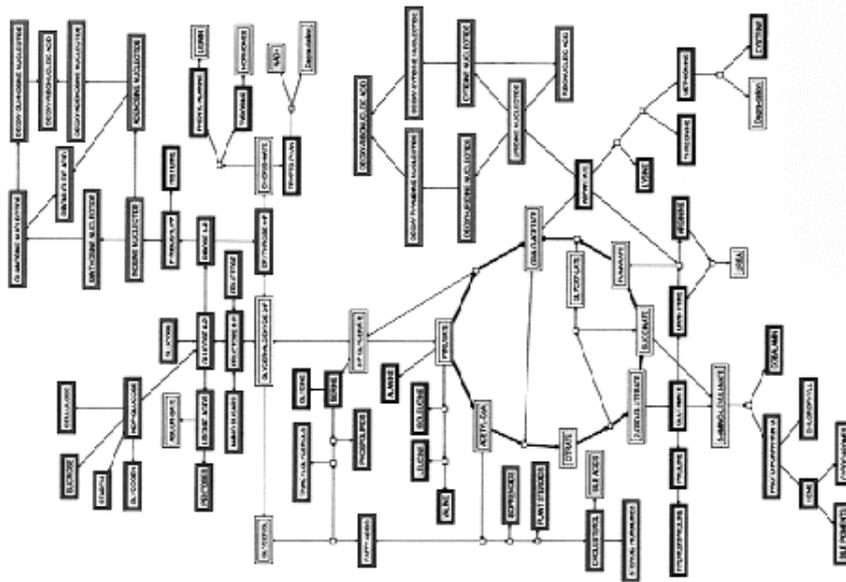


Abbildung 4.9: Dynamische Visualisierung biochemischer Reaktionsnetze (Teil 4). PathDB: (a) Eine manuell gefertigte Zeichnung des Stoffwechselswegs Glycolyse und (b) Darstellung des Citratzyklus mittels Circular-Zeichenverfahren.



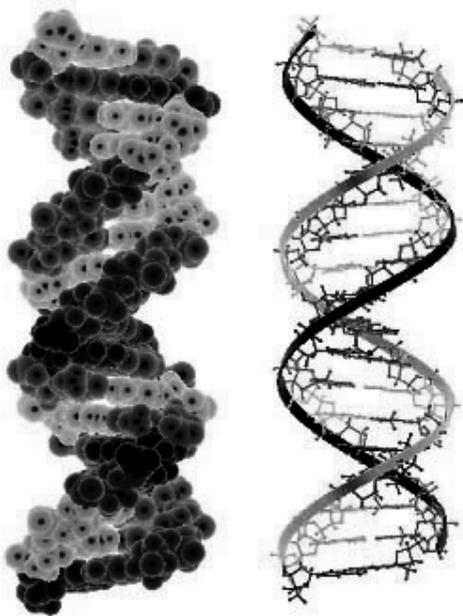
b



a

Abbildung 4.10: Graph Drawing Contest '99. (a) Mankelows Visualisierung und (b) die Lösung von Freivalds und Kikust sind die Siegerlösungen des Wettbewerbs in der Kategorie *biochemische Reaktionsnetze* (aus [BJM⁺99]). Der Ausgangsgraph ist in Abbildung 2.5 auf Seite 20 dargestellt.

5 Modellierung



5.1 Einführung	84
5.2 Reaktionsgraphen	84
5.3 Hierarchische Reaktionsgraphen	86
5.4 Hintergründe zur Modellierung	91

Modellierungen spielen in der Biologie eine wichtige Rolle. 1953 überraschten Watson und Crick die wissenschaftliche Welt mit ihrem Modell des DNA-Moleküls, das sie in einem nur einseitigen Artikel der Zeitschrift *Nature* vorstellten [WC53]. Die Doppelhelix (in der Abbildung zwei unterschiedliche Darstellungsweisen) ist heute das Symbol der Molekularbiologie.

5.1 Einführung

Dieses Kapitel behandelt die Modellierung biochemischer Reaktionsnetze.

Aus Sicht eines Graphentheoretikers lassen sich solche Netze sehr einfach mittels benannter, getyp-ter Hypergraphen $G = (V, E, B, T)$ repräsentieren. Dabei entsprechen Substanzen und Cosubstanzen den Knoten, deren Umwandlung (die eigentlichen Reaktionen) den Kanten des Hypergraphen. Die Abbildung B liefert für die Knoten die Namen der Substanzen bzw. Cosubstanzen, für die Kanten die Namen der katalysierenden Enzyme. Die Abbildung T ordnet den Knoten Typen zu. Damit werden Knoten zu Substanzen oder Cosubstanzen zugeordnet. Als Beispiel sei die Reaktion $H + H + O \rightarrow H_2O$ betrachtet, deren zugehöriger Hypergraph vier Knoten a, b, c, d und eine Hyperkante $(\{a, b, c\}, \{d\})$ enthält.

Obwohl Hypergraphen zur Modellierung biochemischer Reaktion auf den ersten Blick geeignet scheinen, werden sie hier nicht verwendet. Auf die Gründe dafür wird in Abschnitt 5.4.2 eingegangen, dabei werden Hypergraphen auch mit der im Folgenden vorgestellten Modellierung verglichen.¹

Die in diesem Kapitel betrachtete Modellierung biochemischer Reaktionsnetze erfüllt zwei Ziele:

1. Sie spiegelt die Struktur der Netze wider und unterstützt die Unterteilung der Reaktionskomponenten in Substanzen, Cosubstanzen, Enzyme und die eigentlichen Reaktionen.
2. Sie ermöglicht, aus einem gegebenen Reaktionsnetz mit einer zugehörigen Hierarchisierung beliebige Ausschnitte zu erzeugen. Ausschnitte sind Reaktionsnetze, die verschiedene Hierarchieebenen umfassen können, siehe Abbildung 5.1.

Für den ersten Punkt werden *Reaktionsgraphen* eingeführt, die später auch Eingabe für ein Visualisierungsverfahren sind. Für den zweiten Punkt werden Reaktionsgraphen zu *hierarchischen Reaktionsgraphen* erweitert. Durch entsprechende Operationen auf hierarchischen Reaktionsgraphen wird eine Navigation durch biochemische Reaktionsnetze ermöglicht.

5.2 Reaktionsgraphen

Biochemische Reaktionsnetze werden mittels Reaktionsgraphen modelliert, die wie folgt definiert sind:

DEFINITION 5.1 (REAKTIONSGRAPH)

Sei $RG = (K \cup R, A \cup Z, B, T)$ ein benannter getyp-ter bipartiter Graph und RN ein biochemische Reaktionsnetz. RG heißt Reaktionsgraph, falls gilt:

Die Knotenmenge K entspricht den Komponenten der Reaktionen aus RN , also den Substanzen, Cosubstanzen und Enzymen. Die Knotenmenge R entspricht den Reaktionen aus RN .

¹Hier werden statt dessen bipartite Graphen verwendet. Bei bipartiten Graphen und Hypergraphen handelt es sich um ähnliche Strukturen, so dass die beiden Modellierungen nicht grundsätzlich verschieden sind.

Jeder Hypergraph $G_H = (V_H, E_H)$ lässt sich durch einen bipartiten Graph $G = (V_1 \cup V_2, E_1 \cup E_2)$ repräsentieren. Dazu werden die Knoten des Hypergraphen zu Knoten der Menge V_1 . Jede Hyperkante $e \in E_H$ wird durch einen Knoten in V_2 und mehrere Kanten in $E_1 \cup E_2$ repräsentiert. Sei dazu $v_e \in V_2$ der für die Hyperkante $e = (\{u_1, \dots, u_n\}, \{v_1, \dots, v_m\})$ in G eingefügte Knoten, so werden für alle $u_i \in \{u_1, \dots, u_n\}$ Kanten (u_i, v_e) und für alle $v_i \in \{v_1, \dots, v_m\}$ Kanten (v_e, v_i) in G eingefügt.

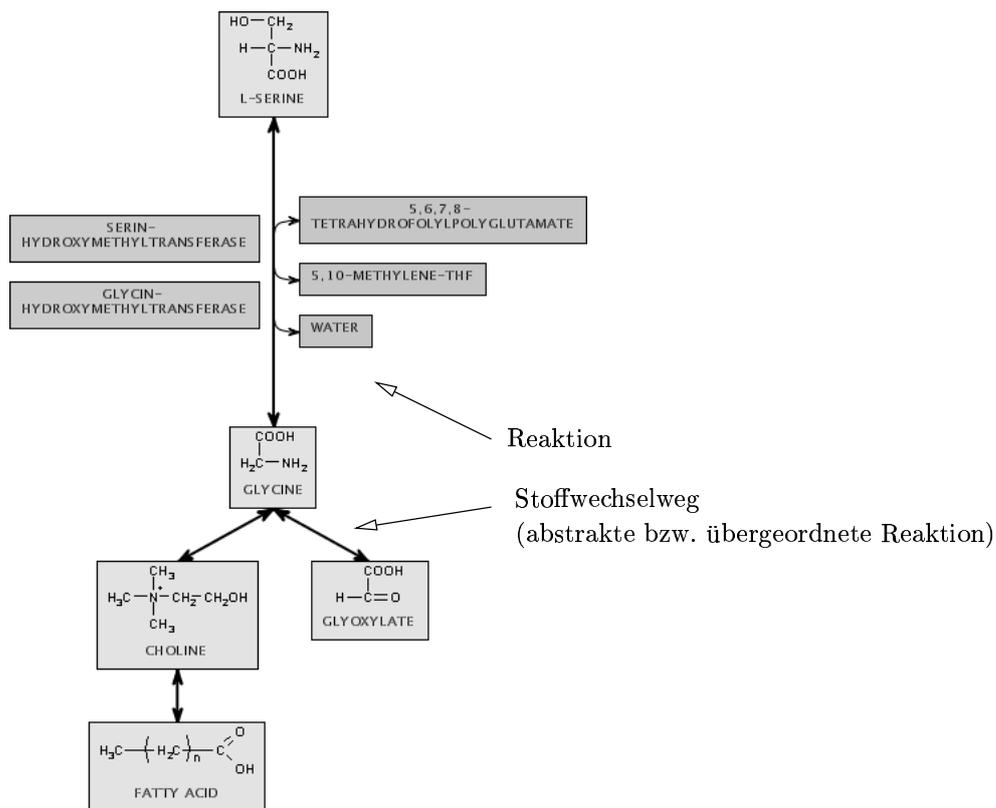


Abbildung 5.1: **Hierarchieebenen.** Verschiedene Hierarchieebenen von biochemischen Reaktionen in einer Darstellung: Eine Reaktion von *Serin* nach *Glycin* (Hierarchieebene der Reaktionen) im Kontext einiger benachbarter Stoffwechselwege (Hierarchieebene der Stoffwechselwege).

Die Kantenmenge $A \subseteq K \times R$ entspricht den Teilen der Reaktionen, die von Edukten, Coedukten und Enzymen zu Reaktionen führen, die Kantenmenge $Z \subseteq R \times K$ den Teilen, die von den Reaktionen zu Produkten, Coprodukten und Enzymen führen.

Die Abbildung $B : K \rightarrow \text{Namen}$ liefert für jeden Knoten $v \in K$ den Namen der zugehörigen chemischen Verbindung bzw. des chemischen Elements. Die Abbildung $T : K \rightarrow \{\text{Substanz, Cosubstanz, Enzym}\}$ liefert für jeden Knoten $v \in K$ einen biochemischen Typ, der angibt, ob der Knoten als Substanz, Cosubstanz oder Enzym betrachtet wird.

In Abbildung 5.2 ist der Reaktionsgraph RG der durch das Enzym E katalysierten Reaktion $A + B \rightarrow C + D$ dargestellt. Es gilt: $RG = (K \cup R, A \cup Z, B, T)$ mit $K = \{A, B, C, D, E\}$, $R = \{\rightarrow\}$, $A = \{(A, \rightarrow), (B, \rightarrow), (E, \rightarrow)\}$ und $Z = \{(\rightarrow, C), (\rightarrow, D), (\rightarrow, E)\}$.

Reaktionsgraphen spiegeln die Verbindungsstruktur der Netze wider und unterstützen die Unterscheidung der Komponenten der Reaktionen. Ob die durch einen Reaktionsgraphen repräsentierten Reaktionen tatsächlich existieren, lässt sich natürlich nur anhand biochemischer Daten überprüfen. Voraussetzung ist also ein korrektes Reaktionsnetz RN .

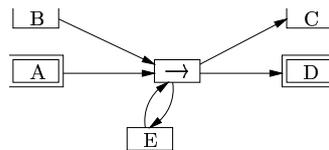


Abbildung 5.2: **Reaktionsgraph**. Der Reaktionsgraph einer durch das Enzym E katalysierten Reaktion $A + B \rightarrow C + D$. A und D sind Substanzen, B und C Cosubstanzen der Reaktion. In dieser und den folgenden Abbildungen gilt: Rechtecke mit Pfeil symbolisieren Reaktionen, Rechtecke mit dem Buchstaben E stehen für Enzyme, Rechtecke mit anderen Buchstaben symbolisieren Cosubstanzen, doppelte Rechtecke symbolisieren Substanzen.

Reaktionsgraphen werden im Folgenden erweitert, um auch die Hierarchie biochemischer Reaktionen (siehe Abschnitt 2.1.3.1) zu repräsentieren und damit die Navigation durch biochemische Reaktionsnetze mittels entsprechender Operationen zu unterstützen.

5.3 Hierarchische Reaktionsgraphen

5.3.1 Hierarchische Graphen

Ein *hierarchischer Graph* entsteht dadurch, dass der Ausgangsgraph (rekursiv) in kleinere Teile zerlegt wird. Die dabei entstehende Struktur ist eine baum- oder DAG-artige Dekomposition des Graphen. Hierarchische Graphen werden nicht nur für die Repräsentation der Hierarchie biochemischer Reaktionen benötigt, sondern schon länger in vielfältiger Weise verwendet. Beispiele sind Design und Visualisierung von Hypertext [BSR92] oder Statecharts zur hierarchischen Repräsentation reaktiver Systeme [Har95, Har96].

Zur Repräsentation der Hierarchie wird hier ein Modell von Buchsbaum und Westbrook [BW00] verwendet.² Dieses erwies sich als guter Ausgangspunkt für eine Modellierung der Hierarchie biochemischer Reaktionen, da es die Erzeugung von Ausschnitten bzw. Sichten auf den Graphen unterstützt. Es wird hier angepasst, um spezielle Ausschnitte hierarchischer Reaktionsgraphen zu beschreiben. Diese Ausschnitte sind Reaktionsgraphen, die Informationen aus verschiedenen Ebenen der Hierarchie enthalten können (siehe Abb. 5.1).

Zuerst werden einige Begriffe eingeführt (siehe auch Abb. 5.3):

²Im Gegensatz zur klassischen Theorie von (nichthierarchischen) Graphen gibt es bisher keine umfassende Theorie für hierarchische Graphen. Busatto et al. schreiben dazu in [BEMW00]:

„Although hierarchical graphs are often used, and in spite the fact that there is an overall agreement on the basic concepts like the use of hierarchies and of encapsulation, there does not exist a general approach to hierarchical graphs nor any general-purpose hierarchical graph data model.“

Ansätze für hierarchische Graphen stammen u. a. von Eades und Feng [EF96, EFL97], die *clustered graphs* vorstellen, Sanders [San96b] mit *verschachtelten Graphen* und Sugiyama und Misue [MS91b, SM91] mit *compound digraphs*. Ein ähnliches Graphmodell wird von North [Nor96] verwendet. Harel [Har95] stellt *higraphs* vor und Brandenburg betrachtet *two leveled clustered graphs* [Bra97a] für zweistufige Hierarchien.

In vielen Modellierungen findet eine Hierarchisierung durch Zusammenfassen von Teilgraphen zu neuen Knoten statt. Himolt [Him93] verallgemeinert das Konzept, so dass Teilgraphen sowohl zu Knoten als auch zu Kanten eines Graphen zusammengefasst werden können.

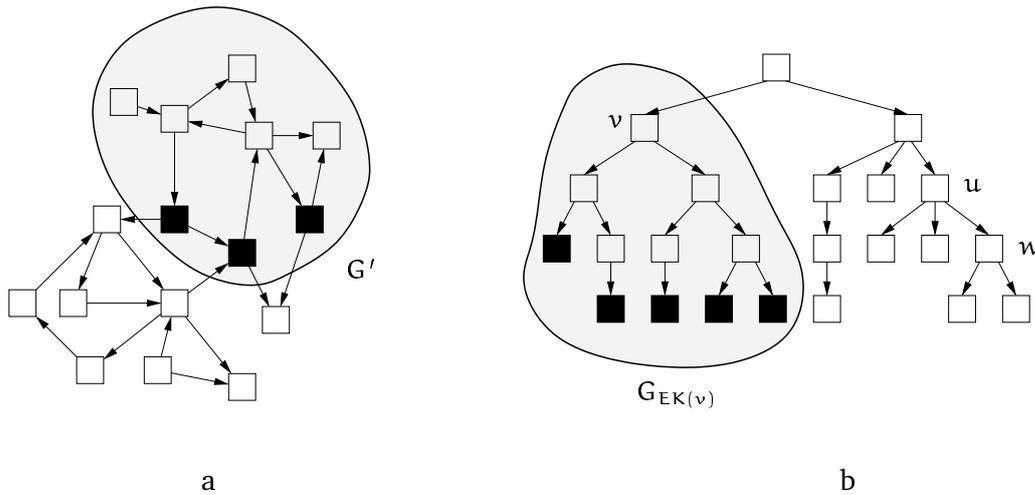


Abbildung 5.3: **Rand, Blätter und Ebeneneindeutigkeit.** (a) Graph $G = (V, E)$, die Knoten des Rands des Teilgraphen $G' \subseteq G$ sind schwarz dargestellt. (b) Baum $B = (V, E)$ mit Teilbaum $G_{EK(v)}$ (grau) und Blättern $Bl(v)$ (schwarz) eines Knotens v . Die Menge $U = \{v, w\}$ ist ebeneneindeutig, die Menge $U' = \{u, w\}$ ist nicht ebeneneindeutig.

DEFINITION 5.2 (RAND, BLÄTTER EINES KNOTENS BZW. EINER MENGE, EBENENEINDEUTIGKEIT)

Sei $G = (V, E)$ ein Graph und $G' \subseteq G$. Die Menge $Rand(G') = \{v \mid v \in G', \exists u \in G \setminus G' \text{ mit } u \text{ adjacent zu } v\}$ wird als Rand von G' bezeichnet.

Sei $B = (V, E)$ ein Baum, $v \in V$ und $U \subseteq V$. Die Menge $Bl(v) = \{u \mid u \in EK(v), Aus(u) = 0\}$ wird als Blätter des Knotens v bezeichnet. Die Menge $Bl(U) = \cup_{v \in U} Bl(v)$ heißt Blätter der Menge U . Die Menge U heißt ebeneneindeutig, falls gilt: $\forall u, v \in U \quad u \in EK(v) \Rightarrow u = v$.

Eine ebeneneindeutige Menge lässt sich auch dadurch charakterisieren, dass kein Knoten der Menge von einem anderen Knoten dieser Menge aus erreichbar ist. Anschaulich sind die durch Knoten einer ebeneneindeutigen Menge U induzierten Teilbäume $B_{EK(u_i)}, u_i \in U$ alle disjunkt.

Ebeneneindeutige Mengen haben eine Eigenschaft, die später für die Navigation durch Reaktionsnetze benötigt wird:

LEMMA 5.3

Sei $B = (V, E)$ ein Baum und $U \subseteq V$ ebeneneindeutig.

1. Für einen Knoten $v \in U$ ist die Menge $U' = (U \setminus \{v\}) \cup Nach(v)$ ebeneneindeutig.
2. Für einen Knoten $v \in V$ mit $Nach(v) \subseteq U$ ist die Menge $U' = (U \setminus Nach(v)) \cup \{v\}$ ebeneneindeutig.

Zum Beweis dieses Lemmas werden zwei Hilfs-Lemmata benötigt:

LEMMA 5.4

Sei $B = (V, E)$ ein Baum und $v, v_1, v_2 \in V$. Es gilt: $v_1 \notin EK(v) \wedge v_2 \in EK(v) \Rightarrow v_1 \notin EK(v_2)$.

Beweis : In einem Baum mit $v_2 \in EK(v)$ kann $v_1 \in EK(v_2)$ nur gelten, falls auch $v_1 \in EK(v)$ gilt. \square

LEMMA 5.5

Sei $B = (V, E)$ ein Baum und $v_1, v_2 \in V$. Es gilt: $(v_1 \notin EK(v_2) \wedge v_2 \notin \text{Nach}(v_1)) \Rightarrow (\forall w \in \text{Nach}(v_1) \ w \notin EK(v_2))$.

Beweis : Da B Baum, gilt für einen beliebigen Knoten $v \in V$: Die Nachfolger $w \in \text{Nach}(v)$ haben nur den Knoten v als Vorgänger. Jeder Weg zu w führt über v .

Sei v_1 ein Knoten, der nicht vom Knoten v_2 aus erreichbar ist ($v_1 \notin EK(v_2)$) und gelte $v_2 \notin \text{Nach}(v_1)$. Nun sind auch die Nachfolger von v_1 nicht von v_2 aus erreichbar, da die Wege von v_2 zu diesen Knoten über v_1 gehen müssten. \square

Beweis zu Lemma 5.3: Zu 1.: Sei $B = (V, E)$ ein Baum, $U \subseteq V$ ebeneneindeutig, $v \in U$ und $U' = (U \setminus \{v\}) \cup \text{Nach}(v)$. Es ist zu zeigen, dass gilt

$$\forall v_1, v_2 \in U' \ v_1 \in EK(v_2) \Rightarrow v_1 = v_2 \quad (5.1)$$

Es werden vier Fälle unterschieden:

1. Seien $v_1, v_2 \in U' \setminus \text{Nach}(v)$. Dann gilt $v_1, v_2 \in U$. Da U ebeneneindeutig ist, gilt $v_1 \in EK(v_2) \Rightarrow v_1 = v_2$.
2. Seien $v_1, v_2 \in \text{Nach}(v)$. Da B ein Baum ist, gilt für v : Zwischen den Teilbäumen $G_{EK(w_i)}, w_i \in \text{Nach}(v)$ existieren keine Kanten. Daraus folgt: $v_1 \in EK(v_2) \Rightarrow v_1 = v_2$.

In den Fällen 3 und 4 wird gezeigt, dass $v_1 \notin EK(v_2)$, wodurch $v_1 \in EK(v_2) \Rightarrow v_1 = v_2$ ebenfalls erfüllt ist.

3. Sei $v_1 \in U' \setminus \text{Nach}(v)$ und $v_2 \in \text{Nach}(v)$. Es gilt: $v_1 \in U \setminus \{v\}$ (und damit $v_1 \neq v$) sowie $v_2 \in EK(v)$. Wegen der Ebeneneindeutigkeit von U und $v_1 \in U \setminus \{v\}$ gilt: $v_1 \notin EK(v)$.
Aus $v_1 \notin EK(v)$, $v_2 \in EK(v)$ und Lemma 5.4 folgt $v_1 \notin EK(v_2)$.
4. Sei $v_1 \in \text{Nach}(v)$ und $v_2 \in U' \setminus \text{Nach}(v)$. Es gilt $v_1 \in EK(v)$ sowie $v_2 \in U \setminus \{v\}$ (und damit $v_2 \neq v$). Wegen der Ebeneneindeutigkeit von U und $v_2 \in U \setminus \{v\}$ gilt: $v \notin EK(v_2)$.
Aus $v_1 \in \text{Nach}(v)$, $v \notin EK(v_2)$, $v_2 \notin \text{Nach}(v)$ und Lemma 5.5 folgt $v_1 \notin EK(v_2)$.

Es gilt also in allen Fällen die Aussage 5.1.

Zu 2.: Sei $B = (V, E)$ ein Baum, $U \subseteq V$ ebeneneindeutig, $v \in V$ mit $\text{Nach}(v) \subseteq U$ und $U' = (U \setminus \text{Nach}(v)) \cup \{v\}$. Es ist wieder zu zeigen, dass gilt

$$\forall v_1, v_2 \in U' \ v_1 \in EK(v_2) \Rightarrow v_1 = v_2 \quad (5.2)$$

Es werden drei Fälle unterschieden:

1. Seien $v_1, v_2 \in U' \setminus \{v\}$. Dann sind $v_1, v_2 \in U$ und da U ebeneneindeutig ist, gilt $v_1 \in EK(v_2) \Rightarrow v_1 = v_2$.
2. Sei $v_1 \in U' \setminus \{v\}$ und $v_2 = v$. Es gilt $v_1 \neq v_2$ und $\forall w \in \text{Nach}(v_2) \ v_1 \notin EK(w)$ (da $w, v_1 \in U$, $w \neq v_1$ und U ebeneneindeutig). Da für alle $w \in \text{Nach}(v_2)$ der Knoten v_2 der einzige Vorgänger von w ist, gilt auch $v_1 \notin EK(v_2)$.
3. Sei $v_1 = v$ und $v_2 \in U' \setminus \{v\}$. Es gilt $v_1 \neq v_2$ und $\forall w \in \text{Nach}(v_1) \ w \notin EK(v_2)$ (da $w, v_2 \in U$, $w \neq v_2$ und U ebeneneindeutig). Da für alle $w \in \text{Nach}(v_1)$ der Knoten v_1 der einzige Vorgänger von w ist, gilt auch $v_1 \notin EK(v_2)$.

Es gilt also in allen Fällen die Aussage 5.2. \square

5.3.2 Hierarchische Reaktionsgraphen

DEFINITION 5.6 (HIERARCHISCHER REAKTIONSGRAPH)

Sei $RG = (K \cup R, A \cup Z, B, T)$ ein Reaktionsgraph und $B = (V, E)$ ein Baum. $HRG = (RG, B)$ heißt hierarchischer Reaktionsgraph, falls gilt:

1. $Bl(V) = K \cup R$

(Die Blätter des Baumes entsprechen den Knoten des Reaktionsgraphen.)

2. $\forall v \in V$ (v ist innerer Knoten) $\Rightarrow \text{Rand}(RG_{Bl(v)}) \subseteq R$

(Für jeden inneren Knoten $v \in V$ induziert $Bl(v)$ einen Teilgraphen $RG' \subseteq RG$, dessen Rand $\text{Rand}(RG')$ nur aus Knoten der Menge R besteht.)

Abbildung 5.4 zeigt einen hierarchischen Reaktionsgraphen.

Die Einschränkung $\text{Rand}(RG_{Bl(v)}) \subseteq R$ für jeden inneren Knoten v bewirkt, dass sich innere Knoten des Baumes wie eine *abstrakte Reaktion* verhalten. Würde man den durch $Bl(v)$ induzierten Teilgraphen $RG' \subseteq RG$ durch den Knoten v ersetzen und v der Menge R des Reaktionsgraphen RG zuordnen, so bliebe RG ein Reaktionsgraph. Dieses Zusammenspiel von konkreten und abstrakten Reaktionen wird im Folgenden näher betrachtet.

Ziel ist eine Notation, mit der Ausschnitte von hierarchischen Reaktionsgraphen beschrieben werden können. Ausschnitte sind dabei neue Reaktionsgraphen, die Teile des Reaktionsnetzes in unterschiedlichen Abstraktionsebenen enthalten. Als Beispiel sei auf Abbildung 5.1 verwiesen, wo eine Reaktion (Hierarchieebene der Reaktionen) im Kontext der sie umgebenden Stoffwechselwege (Hierarchieebene der Stoffwechselwege) gezeigt wird.

5.3.3 Ausschnitte hierarchischer Reaktionsgraphen

In hierarchischen Reaktionsgraphen $HRG = (RG, B)$ entspricht der Reaktionsgraph RG der Ebene der Reaktionen, innere Knoten des Baumes B entsprechen den Stoffwechselwegen bzw. abstrakten Reaktionen. Für einen speziellen Ausschnitt aus dem hierarchischen Reaktionsgraphen müssen Kanten zwischen den verschiedenen Hierarchieebenen eingefügt werden. Diese Kanten sind nicht direkt in HRG enthalten, ergeben sich aber entsprechend der folgenden Definition:

DEFINITION 5.7 (AUSSCHNITT EINES HIERARCHISCHEN REAKTIONSGRAPHEN)

Sei $HRG = (RG, B)$ ein hierarchischer Reaktionsgraph mit $RG = (K \cup R, A \cup Z, B, T)$ und $B = (V, E)$ und sei $U \subseteq V$ in B eine ebeneneindeutige Teilmenge.

Ein durch U gegebener Ausschnitt $\text{Ausschnitt}(U)$ von HRG ist definiert als Reaktionsgraph $RG' = (K' \cup R', A' \cup Z', B, T)$ mit

$$K' = \{k \mid k \in K, k \in U\}$$

$$R' = \{r \mid r \in R, r \in U\} \cup (U \setminus Bl(U))$$

$$A' = \{(u, v) \mid u, v \in U, \exists (u, y) \in A \text{ mit } y \in Bl(v)\}$$

$$Z' = \{(v, u) \mid u, v \in U, \exists (y, u) \in Z \text{ mit } y \in Bl(v)\}$$

SATZ 5.8

Jeder Ausschnitt ist ein Reaktionsgraph.

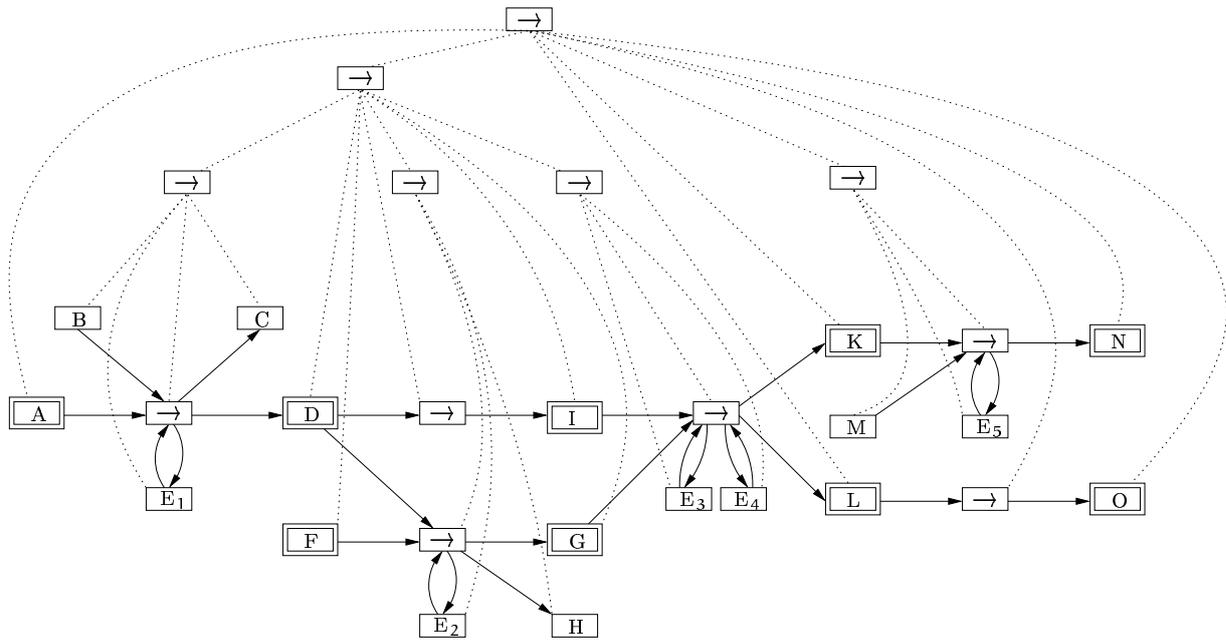


Abbildung 5.4: **Hierarchischer Reaktionsgraph.** Ein hierarchischer Reaktionsgraph $HRG = (RG, B)$. Mit durchgezogenen Linien sind die Kanten des Reaktionsgraphen $RG = (K \cup R, A \cup Z, B, T)$, mit gestrichelten Linien die Kanten des darüber liegenden Baumes $B = (V, E)$ gekennzeichnet. Zur besseren Lesbarkeit sind Baumkanten ohne Pfeilspitzen dargestellt; Baumkanten verlaufen von oben nach unten.

Beweis: Sei $HRG = (RG, B)$ ein hierarchischer Reaktionsgraph mit $RG = (K \cup R, A \cup Z, B, T)$ und $B = (V, E)$, $U \subseteq V$ ebeneneindeutig in B und $RG' = \text{Ausschnitt}(U)$ mit $RG' = (K' \cup R', A' \cup Z', B, T)$. Es ist zu zeigen, dass RG' ein Reaktionsgraph ist, dass also gilt: $K' \cap R' = \emptyset$, $A' \subseteq K' \times R'$ und $Z' \subseteq R' \times K'$. Zudem muss $K' \subseteq K$ und $R' \subseteq R \cup \{v \mid v \text{ innerer Knoten von } V\}$ gelten, um die Unterteilung in Substanzen, Cosubstanzen und Enzyme einerseits und die Reaktionen andererseits zu erhalten.

Für die Knoten gilt: Knoten $u \in U$, $u \notin R$, $u \notin K$ werden entsprechend Definition 5.7 der Menge R' zugefügt. Nach Definition 5.7 gilt also $K' \subseteq K$ und $R' \subseteq R \cup \{v \mid v \text{ innerer Knoten von } V\}$. Weiter gilt für RG' : Da $K \cap R = \emptyset$ und die Knoten $U \setminus \text{Bl}(U)$ nur in R' eingefügt werden, so gilt auch $K' \cap R' = \emptyset$. Für die Kanten gilt: Zuerst werden Kanten in A' betrachtet, es ist $A' \subseteq K' \times R'$ zu zeigen. Dabei müssen zwei Fälle untersucht werden. Sei $(u, v) \in A'$:

1. Falls $(u, v) \in A$, so gilt $u \in K$ und $v \in R$ und damit auch $u \in K'$ und $v \in R'$.
2. Falls $(u, v) \notin A$, so existiert $(u, y) \in A$ mit $y \in \text{Bl}(v)$. Damit gilt $u \in K$, es folgt $u \in K'$. Da v innerer Knoten von V , gilt $v \in R'$.

Für beide Fälle gilt: $A' \subseteq K' \times R'$.

Der Nachweis $Z' \subseteq R' \times K'$ läuft analog. Benennung und Typ der Knoten aus K ändern sich nicht. Der Graph RG' ist also ein Reaktionsgraph. □

Ein Ausschnitt eines hierarchischen Reaktionsgraphen $\text{HRG} = (\text{RG}, \text{B})$ ist also ein Reaktionsgraph, siehe auch Abbildung 5.5. Der Reaktionsgraph RG' besteht dabei aus den Substanzen, Cosubstanzen und Enzymen, die in der Menge U gegeben werden, sowie aus den in U gegebenen Reaktionen und abstrakten Reaktionen. Die Kanten des Reaktionsgraphen RG' setzen sich zusammen aus originalen Kanten des Reaktionsgraphen RG und neuen Kanten, die durch innere Knoten des Baumes B induziert werden.

Nicht alle durch Ausschnitte entstehenden Reaktionsgraphen sind sinnvoll. Die Auswahl passender Ausschnitte kann jedoch nur ein Anwender durchführen. Teilweise sollen in Darstellungen Substanzen weggelassen oder nur abstrakte Reaktionen (Stoffwechselwege) dargestellt werden. Aus diesem Grund reichen auch biochemische Kriterien zur Festlegung sinnvoller Ausschnitte nicht aus.

5.3.4 Vergrößerung und Verfeinerung

Ein vom Anwender gegebener Ausschnitt kann als Grundlage für weitere Ausschnitte dienen, bei denen übergeordnete Reaktionen verfeinert oder Reaktionen zu einer übergeordneten Reaktion zusammengefasst werden. Dies wird *Verfeinerung* bzw. *Vergrößerung* eines Ausschnitts genannt.

DEFINITION 5.9 (VERFEINERUNG EINES AUSSCHNITTS, VERGRÖßERUNG EINES AUSSCHNITTS)

Sei $\text{HRG} = (\text{RG}, \text{B})$ ein hierarchischer Reaktionsgraph mit $\text{RG} = (\text{K} \cup \text{R}, \text{A} \cup \text{Z}, \text{B}, \text{T})$ und $\text{B} = (\text{V}, \text{E})$ und $\text{U} \subseteq \text{V}$ in B ebeneneindeutig. Durch U wird der Ausschnitt $\text{Ausschnitt}(\text{U})$ gegeben.

Sei $v \in \text{U}$ und $\text{U}' = (\text{U} \setminus \{v\}) \cup \text{Nach}(v)$. Der Ausschnitt $\text{Ausschnitt}(\text{U}')$ wird Verfeinerung von $\text{Ausschnitt}(\text{U})$ genannt.

Sei $v \in \text{U}$, $\text{Nach}(v) \subseteq \text{U}$ und $\text{U}' = (\text{U} \setminus \text{Nach}(v)) \cup \{v\}$. Der Ausschnitt $\text{Ausschnitt}(\text{U}')$ wird Vergrößerung von $\text{Ausschnitt}(\text{U})$ genannt.

SATZ 5.10

Jede Verfeinerung und Vergrößerung eines Ausschnitts ist ein Reaktionsgraph.

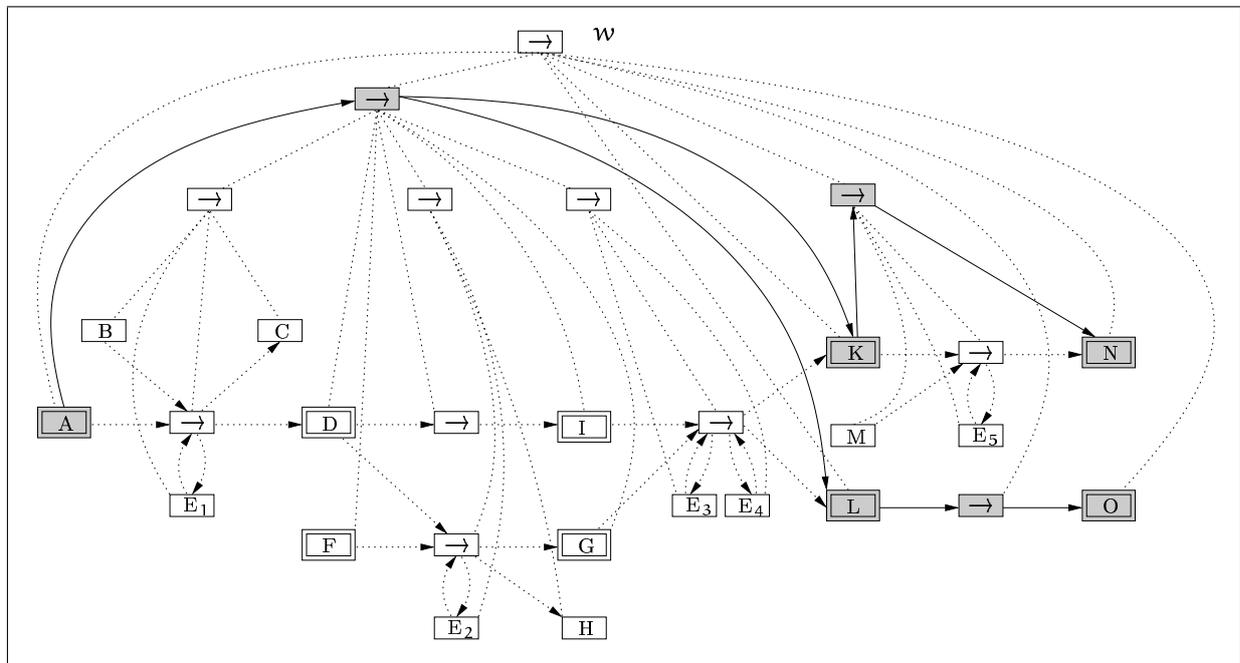
Beweis: Da U' bei Verfeinerung und Vergrößerung nach Lemma 5.3 wieder eine ebeneneindeutige Teilmenge von V ist, ist jeder durch Verfeinerung oder Vergrößerung erzeugte Ausschnitt nach Satz 5.8 ein Reaktionsgraph. \square

Abbildung 5.4 zeigt einen hierarchischen Reaktionsgraph, die Abbildungen 5.5 und 5.6 stellen zwei Ausschnitte dar. Dabei ist der in Abbildung 5.6 gezeigte Ausschnitt eine Verfeinerung des Ausschnitts aus Abbildung 5.5 (und umgekehrt der Ausschnitt aus 5.5 eine Vergrößerung des Ausschnitts aus 5.6). In Abbildung 5.7 wird gezeigt, wie sich die Stufe der Teilreaktionen in das Modell einfügt.

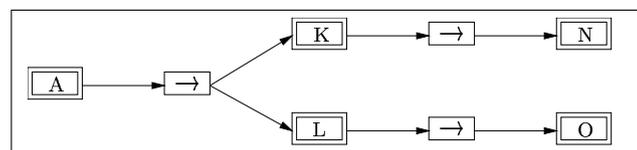
5.4 Hintergründe zur Modellierung

5.4.1 Modell für Visualisierungen biochemischer Reaktionsnetze

Reaktionsgraphen unterstützen die Unterteilung der Reaktionen in Komponenten. Dies ist Grundlage für angepasste Visualisierungen, bei denen Substanzen, Cosubstanzen und Enzyme einer Reaktion unterschiedlich dargestellt werden. Hierarchische Reaktionsgraphen sowie Ausschnitte und deren Vergrößerung und Verfeinerung dienen der Erzeugung verschiedener Sichten und zur Navigation durch Reaktionsnetze. Die Anwendung dieser Verfahren wird in Kapitel 8 betrachtet.

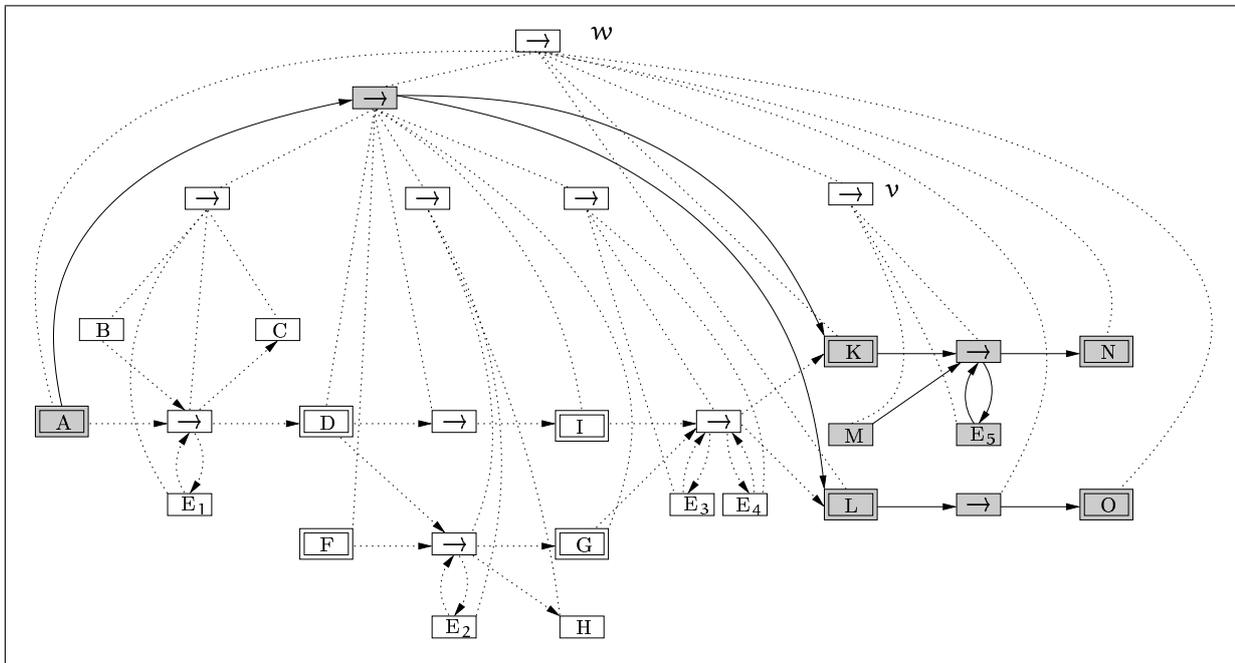


a

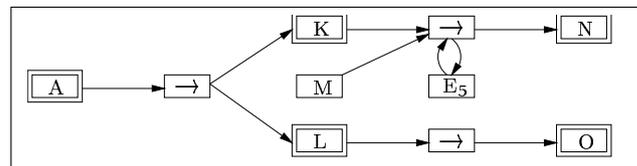


b

Abbildung 5.5: **Ausschnitte hierarchischer Reaktionsgraphen (Teil 1).** (a) Hierarchischer Reaktionsgraph (aus Abbildung 5.4) mit einem Ausschnitt U . Der Ausschnitt wird durch die Menge $U = \text{Nach}(w)$ erzeugt (die Knoten der Menge U sind grau dargestellt, die induzierten Kanten sind durchgezogen). (b) Der dem Ausschnitt entsprechende Reaktionsgraph.



a



b

Abbildung 5.6: **Ausschnitte hierarchischer Reaktionsgraphen (Teil 2)**. (a) Neuer Ausschnitt des hierarchischen Reaktionsgraphen aus Abbildung 5.5, der durch Verfeinerung des Knotens v entsteht (die Knoten von $U' = (U \setminus \{v\}) \cup \text{Nach}(v)$ sind grau dargestellt), (b) der dem neuen Ausschnitt entsprechende Reaktionsgraph.

unterschiedlichen Farben dargestellt werden.

Bisherige Systeme zur Darstellung von Graphen unterstützen überwiegend Graphen mit einfachen Kanten, deren Darstellung nicht in mehrere Teile zerfällt. Dies gilt auch für das im Rahmen dieses Projekts verwendete System Graphlet [Him97, Gra01a]. Hinzu kommt, dass sich nur wenige Arbeiten mit Zeichenverfahren für Hypergraphen beschäftigen, die überwiegende Zahl der Entwicklungen und Veröffentlichungen für Zeichenalgorithmen beziehen sich auf (einfache) Graphen.⁴ Um möglichst viele vorhandene Entwicklungen zu nutzen, werden bipartite Graphen in der Modellierung verwendet.

3. Die Visualisierung ist nur eine der Aufgaben bei der Untersuchung von Reaktionsnetzen. Eine weitere Aufgabe ist deren Simulation. In Abschnitt 5.4.4 werden weitere Modellierungen biochemischer Reaktionsnetze vorgestellt. Einige der dort betrachteten Arbeiten verwenden Petri-Netze⁵ zur Simulation biochemischer Reaktionsnetze. Da es sich bei Petri-Netzen ebenfalls um bipartite Graphen handelt, lassen sich dafür entwickelte Verfahren relativ einfach auf die hier definierten Reaktionsgraphen übertragen. Zudem bietet das Konzept der Ausschnitte auch für Simulationssysteme neue Ansätze zur Behandlung großer Reaktionsnetze.
4. Die Verwendung bipartiter Graphen schränkt nicht die Ausdrucksstärke der Modellierung ein. Wie bereits dargestellt, lassen sich Hypergraphen mittels bipartiter Graphen modellieren und umgekehrt.

5.4.3 Realisierung in BioPath

Bei den hier vorgestellten hierarchischen Reaktionsgraphen handelt es sich um ein Modell, das Visualisierungsanforderungen, z. B. eine einfache Navigation durch die Hierarchie, unterstützt. Dieses Modell unterscheidet sich von dem wesentlich komplexeren Modell, welches im *BioPath*-Projekt verwendet wird. Das *BioPath*-Modell bzw. Datenbankschema ist in [Dat00] beschrieben. Es dient der Repräsentation vielfältiger biochemischer Informationen, die weit über Reaktionsnetze hinausgehen. Das Modell erlaubt auch beliebige Hierarchien über biochemische Reaktionen, wobei Ausschnitte solcher Hierarchien nicht notwendigerweise Reaktionsnetze darstellen müssen.

Der wichtigste Unterschied zwischen Hierarchien in *BioPath*, deren Ausschnitte Reaktionsnetze darstellen, und dem hier vorgestellten Modell liegt darin, dass in *BioPath* die Struktur der Hierarchie keinem Baum, sondern einem DAG entspricht. Die dort für Reaktionen realisierte Hierarchie orientiert sich an der historisch entstandenen Hierarchie aus [Mic93]. Während neu entwickelte Hierarchisierungen die Forderung nach Eindeutigkeit und Vollständigkeit der Hierarchie erfüllen (vergleiche auch Abschnitt 3.4.4), gilt dies für die historisch entstandene Hierarchie nicht. In ihr werden einige Reaktionen mehreren Stoffwechselwegen zugeordnet, ein Beispiel ist die Reaktion *Dimethylcitrat* ↔ *3-Isopropylmalat*, die sowohl der *Synthese von Valin* wie der *Synthese von Leucin* zugeordnet ist.

Ein Ziel des *BioPath*-Projekts ist die elektronische Realisierung des Posters *Biochemical Pathways* [Mic93], deshalb ist dort die Verwendung der DAG-artigen Hierarchie nötig. DAG-artige Strukturen

⁴Siehe [CM93, Gro96, KLMS95, Mäk90b, Min98], die sich mit Zeichenverfahren für Hypergraphen beschäftigen. Im Gegensatz dazu beschäftigen sich mehrere hundert Arbeiten mit dem Zeichnen von Graphen, siehe z. B. die Verweise in [DETT94, DETT99].

⁵Siehe Fußnote auf Seite 69.

haben einen wesentlichen Nachteil: Bei der Bestimmung von übergeordneten (abstrakten) Reaktionen muss u. U. vom Anwender spezifiziert werden, welche übergeordnete Reaktion verwendet werden soll. Dies liegt daran, dass ein Knoten mehrere Vorgänger haben kann. Bei diesen Strukturen ist also eine zusätzliche Interaktion mit dem Anwender notwendig.

Das hier vorgestellte baumartige Modell benötigt diese Interaktion nicht. Es kann mit dem Datenbankschema realisiert werden, wenn die Hierarchie bzw. der unterliegende Reaktionsgraph entsprechend angepasst werden [Kan01].

5.4.4 Weitere Modelle für biochemische Reaktionsnetze

Für Verfahren zur automatischen Visualisierung wurden biochemische Reaktionsnetze bisher nur in Form einfacher oder benannter Graphen wie in *EcoCyc*, *PathDB* und ähnlichen Systemen modelliert (vergleiche Abschnitt 4.2). Für den Zweck der Simulation gibt es dagegen bereits eine Reihe Modellierungen, die hier kurz zusammengefasst werden sollen. Diese Modelle lassen sich grob in *analytische* und *diskrete* Modelle unterteilen.

Analytische Modelle sind von der hier gewählten Modellierung weit entfernt und beruhen auf der Darstellung von biochemischen Reaktionen oder Reaktionsnetzen mittels differentialer Gleichungssysteme. So modellieren z. B. Waser et al. [WGKG83] und Franco und Canelas [FC84] ausgewählte Stoffwechselforgänge mittels Differentialgleichungen.

Diskrete Modelle sind der hier gewählten Modellierung wesentlich näher. Sie beruhen auf Zustandsübergangsdiagrammen und Petri-Netzen und dienen überwiegend der qualitativen Simulation von Reaktionsnetzen. In diskreten Modellen wird der Stoffwechsel als eine Folge sequenziell oder parallel ablaufender diskreter Schritte betrachtet.

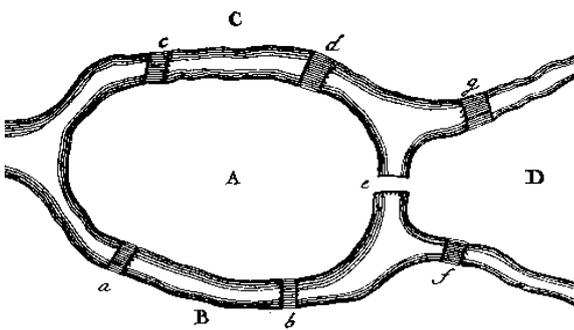
Reddy et al. [RML93] führen Petri-Netze zur Modellierung biochemischer Reaktionsnetze ein. Dabei entsprechen die Plätze den Substanzen und Enzymen, die Transitionen den Reaktionen. Hofestädt und Thelen [HT98] verwenden Petri-Netze für quantitative Modellierung biochemischer Reaktionsnetze. Kohn modelliert in MetaNet [Koh92] Reaktionsnetze ebenfalls mittels Graphen, wobei die Knoten in zwei Klassen unterteilt werden, *Chemnodes* für Substanzen, Enzyme und DNA, *Relnodes* für Beziehungen wie Substratbindung. Der Graph ist jedoch nicht bipartit. Wichert [Wie94] beschreibt die Verwendung bipartiter Graphen zur Darstellung von Reaktions- und Atomnetzen. Knoten entsprechen Atomen, Kanten können einerseits die Zugehörigkeit verschiedener Atome zum gleichen Molekül, andererseits die Umsetzung der Atome während einer Reaktion ausdrücken. Im KEGG-System [GBO⁺97] repräsentieren binäre Relationen Beziehungen zwischen interagierenden Molekülen oder Genen, beispielsweise zwischen Enzym und Substrat.

Diese Modellierungen sind als Grundlage für Visualisierungen biochemischer Reaktionsnetze entsprechend den Darstellungsanforderungen in Kapitel 3 ungeeignet. Sie unterstützen die Unterteilung der Komponenten der Reaktionen nur ungenügend und die Hierarchisierung von Reaktionen gar nicht.

Teil II

Das VGL-Zeichenverfahren und seine Anwendung in *BioPath*

6 Ebenenweises Zeichnen gerichteter Graphen



6.1 Überblick	100
6.2 Kantenorientierung	117
6.3 y-Koordinaten und Ebenenpartitionierung	136
6.4 Kreuzungsreduzierung	155
6.5 x-Koordinaten	176
6.6 Gesamtalgorithmus	177

Eulers *Königsberger Brückenproblem* gilt als erste Arbeit zur Graphentheorie. In der Abbildung (aus [Eul36], zitiert aus [Wil86]) entsprechen die Landflächen den Knoten, die Brücken den Kanten eines Graphen.

Die allgemeinverständliche Darstellung seiner Ergebnisse war Euler ein wichtiges Anliegen. Dazu schreibt Wußmann in [Wuß89]:

„Alle Lehrbücher Eulers sind mit wunderbarer Klarheit geschrieben und mit größtem methodischem Geschick aufgebaut. Die „Vollständige Anleitung zur Algebra“ soll Euler sogar seinem Diener, einem Schneidergesellen, diktieren haben und nicht eher den Text endgültig verabschiedet haben, bis dieser Laie den Inhalt ganz habe begreifen können.“

6.1 Überblick

6.1.1 Einführung

Gerichtete Graphen werden in vielen Anwendungen genutzt, um Abhängigkeiten zwischen Objekten darzustellen. Beispiele sind PERT¹-, ER²- und Workflow-Diagramme, Programm- und Datenflussgraphen und Abhängigkeiten in Dateisystemen, siehe beispielsweise die Arbeiten [BNT86, BTT83, DPTT89, Lin94, Rei95, San99]. Diesen Anwendungen ist gemein, dass die Kanten des Graphen Abhängigkeiten repräsentieren.

Diese Abhängigkeiten sollen in einer Visualisierung auf einen Blick erfassbar sein. Ebenenweise Zeichnungen, bei denen Abhängigkeiten in einer Richtung, z. B. von oben nach unten, verlaufen, sind dafür besonders geeignet. In solchen Visualisierungen muss ein Betrachter die Objekte nicht erst selbst entsprechend ihren Abhängigkeiten aufreihen, diese Arbeit wird ihm durch das Zeichenverfahren abgenommen (siehe beispielsweise Abb. 6.1).

Biochemische Reaktionsnetze enthalten Abhängigkeiten in Form der Reihenfolge der Reaktionen. Durch eine entsprechende Anordnung der Reaktionen und ihrer Komponenten von oben nach unten lässt sich diese Reihenfolge darstellen, wodurch das Verstehen der Reaktionsfolge unterstützt wird. Ebenenweise Zeichnungen sind also prinzipiell zur Darstellung biochemischer Reaktionsnetze geeignet. In Kapitel 4 wurde jedoch schon deutlich, dass die bisherigen Verfahren für ebenenweise Zeichnungen zur Visualisierung von Reaktionsnetzen nicht ausreichend sind.

Ebenenweise Zeichenverfahren werden seit über 20 Jahren untersucht und haben sich in der Praxis bewährt. Als grundlegende Arbeit über ebenenweises Zeichnen gerichteter Graphen gilt [STT81] von Sugiyama, Toda und Tagawa, auch wenn bereits vorher Warfield [War77] und Carpano [Car80] Methoden zur Reduzierung der Kantenkreuzungen für ebenenweise Zeichenverfahren betrachtet haben. Das Verfahren zum ebenenweisen Zeichnen von Graphen nach Sugiyama et al. wird oft als *Sugiyama-Algorithmus* bezeichnet.

Das in dieser Arbeit entwickelte Verfahren erweitert die bekannten Algorithmen. Die Erweiterungen betreffen besonders die Berücksichtigung der Knotengröße bei der Platzierung und die effiziente Behandlung von anwendergegebenen Lagebeziehungen zwischen Knoten. Knotengröße und Lagebeziehungen sowie allgemeine Anforderungen an ebenenweise Zeichnungen geometrischer Graphen werden im Folgenden betrachtet. Die wesentlichen Schritte des Verfahrens werden in den Abschnitten 6.2 bis 6.5 erörtert, Kapitel 7 zeigt Erweiterungen auf. In Kapitel 8 wird dargestellt, wie sich dieses Zeichenverfahren zur Visualisierung biochemischer Reaktionsnetze nutzen lässt.

Bei den folgenden Betrachtungen wird davon ausgegangen, dass die Ebenen horizontal verlaufen und von oben nach unten angeordnet sind, der Graph also von oben nach unten gezeichnet wird. Die x -Achse verläuft von links nach rechts, die y -Achse von oben nach unten. Zudem sind alle Koordinaten der Objekte einer Zeichnung positiv, der Nullpunkt befindet sich links oben. Prinzipiell ist jede Richtung für die Darstellung möglich, es bedarf lediglich einer Anpassung der richtungsabhängigen Aussagen. Die Anforderungen an das Zeichenverfahren wurden hier bewusst allgemein gehalten und von den Darstellungsanforderungen für die Visualisierung biochemischer Reaktionsnetze getrennt, um das Verfahren für viele verschiedene Anwendungen nutzen zu können.

¹Program Evaluation and Review Technique

²Entity-Relationship

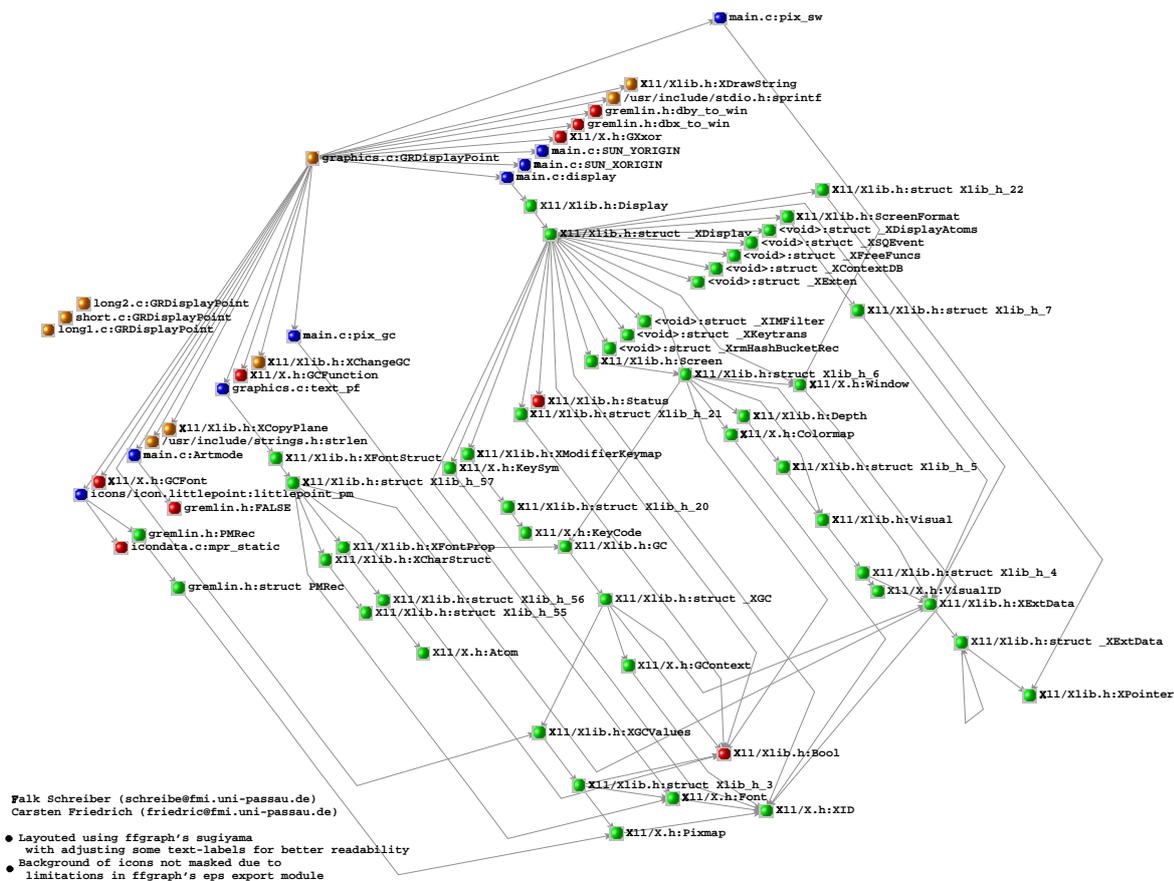


Abbildung 6.1: **Ebenenweise Zeichnungen.** Darstellung der Abhängigkeiten einiger Module des XWindows-Systems. Dabei sind die Ebenen nicht horizontal, sondern schräg angeordnet, um die Lesbarkeit der Knotenbenennungen zu verbessern (aus [EM96], diese Darstellung wurde auch in [DETT99] übernommen).

6.1.2 Beliebige Knotengrößen

In Abschnitt 2.2.3.3 wurden bereits verschiedene Klassen von Zeichenverfahren vorgestellt. Arbeiten zu diesen Verfahren berücksichtigen Knotengrößen oft nicht, die Knoten werden als Punkte ohne Ausdehnung betrachtet, siehe beispielsweise [Ead84, FR91, SM95] für kräftebasierte Zeichenverfahren, [STT81] für ebenenweise Zeichenverfahren und [Tam87, PT95] für orthogonale Zeichenverfahren.

Diese Ansätze waren für die Entwicklung und Untersuchung der Zeichenverfahren durchaus berechtigt, allerdings erfordert die Verwendung von Zeichenalgorithmen für konkrete Anwendungen fast immer die Berücksichtigung der Knotengröße. Man denke beispielsweise an die Darstellung von UML³-Klassendiagrammen, in denen die Klassen auf Grund unterschiedlich vieler Attribute

³Unified Modeling Language, siehe auch [BRJ98, FS98].

und Methoden verschieden groß dargestellt werden, an Darstellungen von Programmabläufen mit verschieden großen Blöcken oder an die Visualisierung biochemischer Reaktionsnetze mit unterschiedlich großen Knoten entsprechend den zugehörigen Strukturformeln bzw. der dargestellten Information.

Um schöne Zeichnungen zu erhalten, müssen die Graphen u. a. kompakt dargestellt werden, unnötige Abstände zwischen Knoten sind zu vermeiden. Dazu muss eine Kette kleiner Knoten neben einem großen Knoten platziert werden, wenn die Struktur des Graphen dies ermöglicht, siehe Abbildung 6.4. Um dies zu erreichen, müssen Knotengrößen berücksichtigt werden.

Im folgenden Abschnitt wird eine Zusammenfassung der Möglichkeiten zur Berücksichtigung der Knotengrößen in allgemeinen Zeichenverfahren gegeben. Abschnitt 6.1.2.2 beschäftigt sich mit der Behandlung der Knotengröße in ebenenweisen Zeichenverfahren.

6.1.2.1 Allgemeine Berücksichtigung beliebiger Knotengrößen

Beim Zeichnen von Graphen lassen sich drei Ansätze der Berücksichtigung von Knotengrößen unterscheiden:

1. Die Vergrößerung aller Knoten bis zur Größe des maximalen Knotens. Anschließend wird der so geänderte Graph durch Algorithmen zum Zeichnen von Graphen mit uniformen Knotengrößen platziert. Dieser Ansatz findet sich bereits in frühen Arbeiten zu Graphzeichenalgorithmen. So stellen Batini et al. ein Verfahren vor, bei dem alle Knoten als gleich große Rechtecke betrachtet und auf einem Gitter angeordnet werden [BFN85, BNT86]. Gängige Verfahren für ebenenweise Zeichnungen z. B. von Gansner et al. [GKNV93] und Sanders [San96b] fallen ebenfalls in diese Kategorie.

Nachteil bei der Anwendung dieser Lösung für ebenenweise Zeichenverfahren ist, dass damit keine Darstellungen wie in Abbildung 6.4 möglich sind. Ebenenweise Zeichenverfahren werden im nächsten Abschnitt genauer betrachtet.

2. Die Nachbearbeitung einer Zeichnung. Dabei werden die Knoten zunächst als punktförmig betrachtet und durch Algorithmen zum Zeichnen von Graphen mit punktförmigen Knoten platziert. Dann werden die Knoten wieder auf ihre Ausgangsgröße gebracht. Anschließend wird die Zeichnung durch Verschieben von Knoten so verändert, dass Knotenüberdeckungen möglichst beseitigt werden. Im Allgemeinen hat die relative Anordnung der Knoten eine Bedeutung und soll erhalten bleiben. In [HIMF98, MELS95, MST01, Tam99a] werden entsprechende Verfahren zum Verschieben der Knoten unter Beibehaltung relativer Lagebeziehungen vorgestellt.

Nachteil bei der Anwendung dieser Lösung ist das Entstehen von Visualisierungen, die den Ästhetikkriterien nicht genügen, siehe Abbildung 6.2. Gleiches gilt auch für einen ähnlichen Ansatz, bei dem nach dem Vergrößern aller Knoten auf maximale Größe die Zeichnung erstellt wird. Anschließend werden Knoten auf ihre ursprüngliche Größe verkleinert und die Zeichnung wird durch Verschieben von Knoten unter Erhalt der relativen Lagebeziehungen kompakter gestaltet.

3. Die Berücksichtigung der tatsächlichen Knotengröße durch Erweiterung bestehender Verfahren bzw. Entwicklung neuer Algorithmen. Für orthogonale Zeichenverfahren wird dieser Ansatz von Biedl et al. [BMT97], Di Battista et al. [DDPP99] und Fölsmeier und Kauf-

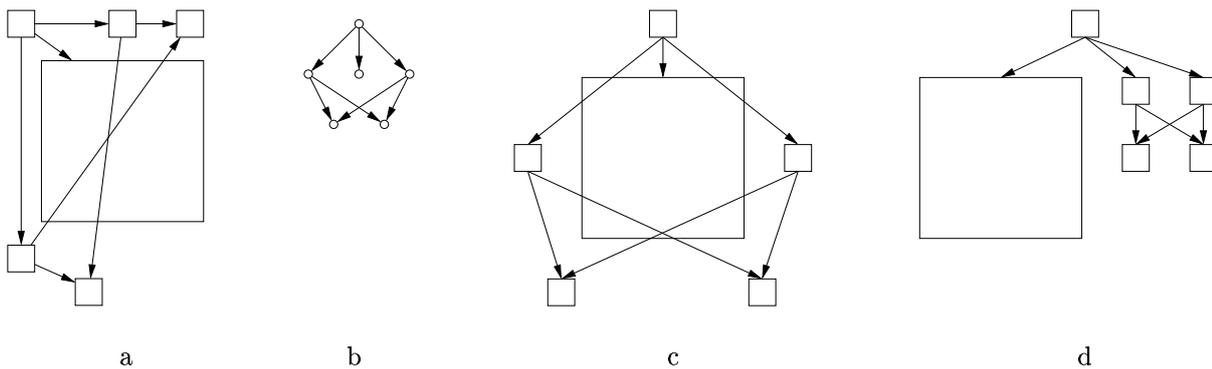


Abbildung 6.2: **Nachbearbeitung einer Zeichnung.** Der im Punkt 2 von Abschnitt 6.1.2.1 beschriebene Ansatz kann bei ebenenweisen Zeichnungen zu Darstellungen führen, die den Ästhetikkriterien nicht genügen. (a) Graph, (b) Platzierung von punktförmigen Knoten durch ebenenweises Zeichenverfahren, (c) Vergrößerung der Knoten auf Originalgröße und Verschieben unter Beibehaltung relativer Lagebeziehungen zwischen Knoten. Diese Zeichnung enthält unnötige Kreuzungen zwischen Knoten und Kanten und ist zudem höher als notwendig, was allgemeinen Ästhetikkriterien widerspricht. (d) Eine Zeichnung des Graphen, die den Ästhetikkriterien besser entspricht.

mann [FK97] untersucht. Mit der Berücksichtigung der Knotengröße in anderen Klassen von Zeichenverfahren beschäftigen sich beispielsweise Bloesch [Blo93] und Bachl [Bac97] für das Zeichnen von Bäumen und Wang und Miyamoto [WM96], Forster [For99] sowie Gansner und North [GN98] für kräftebasierte Verfahren.

Für ebenenweise Zeichenverfahren gibt es dagegen bisher keine entsprechenden Algorithmen. Da auch die anderen oben genannten Ansätze für ebenenweise Zeichenverfahren nicht ausreichend sind, muss eine neue Lösung entwickelt werden. Dies ist Ziel dieses Kapitels.

6.1.2.2 Knotengrößen in ebenenweisen Zeichenverfahren

6.1.2.2.1 Geometrische topologische Platzierungen

Wie lassen sich bei ebenenweisen Zeichnungen beliebige Knotengrößen berücksichtigen? Um diese Frage zu beantworten, sollen zunächst die verschiedenen Möglichkeiten der Zuordnung der Knoten zu y -Koordinaten, und damit zu Ebenen, vorgestellt werden (vergleiche auch Abb. 6.3).

DEFINITION 6.1 (GEOMETRISCHE TOPOLOGISCHE PLATZIERUNG)

Sei $G = (V, E, M)$ ein geometrischer azyklischer Graph, α_{\min} der Mindestabstand zwischen Knoten und $P = (P_V, P_E)$ eine Platzierung von G . Die Platzierung P heißt

1. *größenfrei*, wenn gilt

$$\forall (u, v) \in E \quad y(u) + \alpha_{\min} \leq y(v) \quad (\text{siehe Abb. 6.3(a)})$$

2. größenmaximal, wenn gilt

$$\forall (u, v) \in E \quad y_o(u) + h_{\max} + a_{\min} \leq y_o(v)$$

mit $h_{\max} = \max\{\text{Höhe}(w) \mid w \in V\}$ (siehe Abb. 6.3(b))

3. größenebenenmaximal, wenn gilt

$$\forall (u, v) \in E \quad y_o(u) + h_e + a_{\min} \leq y_o(v)$$

mit $h_e = \max\{\text{Höhe}(w) \mid w \in V, y_o(u) = y_o(w)\}$ ⁴ (siehe Abb. 6.3(c))

4. größenecht, wenn gilt

$$\forall (u, v) \in E \quad y_o(u) + \text{Höhe}(u) + a_{\min} \leq y_o(v)$$

(siehe Abb. 6.3(d))

DEFINITION 6.2 (MINIMALE GEOMETRISCHE TOPOLOGISCHE PLATZIERUNG)

Eine geometrische topologische Platzierung P eines Graphen $G = (V, E, M)$ heißt minimal, wenn gilt: $\max\{y(v) \mid v \in V\}$ ist minimal und $y(v) \geq 0$ für alle Knoten $v \in V$.

Im Folgenden wird einfach von *größenfreien Platzierungen* statt *minimalen größenfreien geometrischen topologischen Platzierungen* gesprochen, Analoges gilt für die übrigen Begriffe.

6.1.2.2.2 Realisierungen der Platzierungen

Bei herkömmlichen Verfahren für ebenenweise Zeichnungen wird entweder die Höhe von Knoten nicht berücksichtigt oder alle Knoten des Graphen bzw. alle Knoten einer Ebene werden auf die maximale Höhe der betrachteten Knoten gedehnt.⁵ So werden in [STT81, War77] die Knotengrößen nicht beachtet, es entstehen größenfreie Platzierungen wie in der Abbildung 6.3(a). In [Car80, KN93, RDM⁺87, San96b] werden die Knoten entsprechend einer topologischen Sortierung auf Ebenen mit uniformem Ebenenabstand angeordnet. Dabei gibt der höchste Knoten des Graphen den Ebenenabstand vor, dies entspricht einer größenmaximalen Platzierung (siehe Abb. 6.3(b)). Sanders [San96b] schlägt vor, den Knoten bei solchen Platzierungen nicht dieselbe y_o -Koordinate zuzuweisen. Stattdessen sollten die Knoten einer Ebene mit gleichen y -Koordinaten platziert werden, was einer Zentrierung in y -Richtung innerhalb der Ebene entspricht.

In [GKNV93, MRH91, San99] werden Knoten ebenfalls entsprechend einer topologischen Sortierung auf Ebenen mit uniformem Ebenenabstand angeordnet. Nun gibt jedoch für jede Ebene der höchste Knoten der Ebene den Abstand zur nächsten Ebene vor, was einer größenmaximalen Platzierung entspricht (Abb. 6.3(c)).

Diese drei Ansätze lassen sich ohne großen Aufwand ineinander transformieren, wenn die Größe der Knoten bekannt ist. Die Zuordnung der Knoten zu Ebenen erfolgt bei diesen Ansätzen ohne Berücksichtigung der Knotengröße, sie wird auch *globale Ebeneneinteilung* genannt.

⁴ h_e entspricht also der Höhe des größten Knotens in einer Ebene.

⁵Die Knotenbreite wird dagegen von den meisten Verfahren korrekt betrachtet. Die Vernachlässigung der Knotenhöhe liegt darin begründet, dass die übliche Ebeneneinteilung nur die Topologie, nicht aber die Geometrie des Graphen berücksichtigt. Innerhalb der Ebenen ist es hingegen einfach, bei der Bestimmung der x -Koordinaten die Knotenbreite zu berücksichtigen.

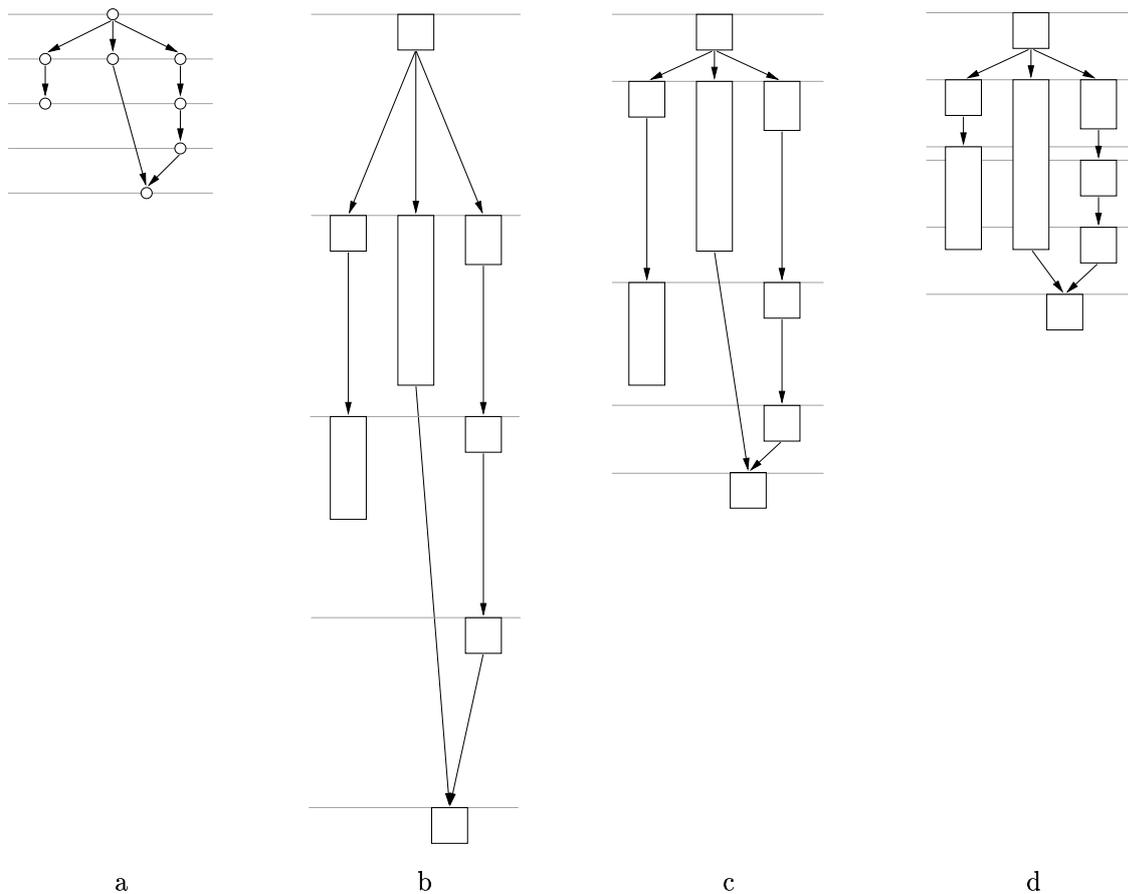


Abbildung 6.3: **Geometrische topologische Platzierungen.** Verschiedene ebenenweise Zeichnungen eines Graphen: Ebeneneinteilung

(a) ohne Berücksichtigung der Knotengröße (*größenfrei*).

(b) mit Berücksichtigung der Knotengröße, uniformem Ebenenabstand und globaler Ebeneneinteilung. Die Einteilung der Ebenen erfolgt zuerst ohne Beachtung der Knotengrößen wie in (a). Die Ebenen werden anschließend so verschoben, dass der höchste Knoten des Graphen den Abstand der Ebenen bestimmt (*größenmaximal*).

(c) mit Berücksichtigung der Knotengröße, variablem Ebenenabstand und globaler Ebeneneinteilung. Die Einteilung der Ebenen erfolgt zuerst ohne Berücksichtigung der Knotengrößen wie in (a). Die Ebenen werden anschließend so verschoben, dass der höchste Knoten pro Ebene den Abstand zur nächsten Ebene bestimmt (*größenebenenmaximal*).

(d) mit Berücksichtigung der Knotengröße und lokaler Ebeneneinteilung. Die Ebene jedes Knotens wird durch den längsten Pfad von einer Wurzel zum Knoten unter Berücksichtigung der Knotenhöhen bestimmt (*größenecht*).

Während in den Fällen (a)-(c) die Zahl der Ebenen gleich ist, treten bei größenechten Platzierungen oft mehr Ebenen auf (vergleiche (c) mit (d)).

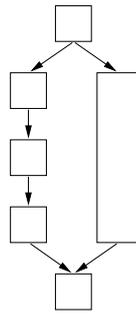


Abbildung 6.4: **Kompaktheit von Zeichnungen.** Kompakte Platzierung kleiner und großer Knoten am Beispiel der Anordnung einer Kette kleiner Knoten neben einem großen Knoten. Eine solche Darstellung wird durch herkömmliche Verfahren für ebenenweise Zeichnungen nicht erreicht.

Wenn Graphen Knoten unterschiedlicher Größe enthalten, so ist der Nachteil globaler Ebeneneinteilungen eine auseinander gezogene und dadurch unübersichtliche Zeichnung. Größenechte Platzierungen haben diesen Nachteil nicht. Sie liefern kompakte und übersichtliche Zeichnungen, siehe Abbildung 6.3(d). Die Zuordnung der Knoten zu Ebenen erfolgt nun nicht mehr auf Grund einer rein topologischen Sortierung, sondern es werden die Knotenhöhen mit berücksichtigt. Dabei werden Knoten, die bei traditionellen Verfahren auf derselben Ebene platziert würden, oft in verschiedene Ebenen verteilt. Eine solche Zuordnung wird auch *lokale Ebeneneinteilung* genannt. Größenechte Platzierungen führen zu neuen Problemen:

SATZ 6.3

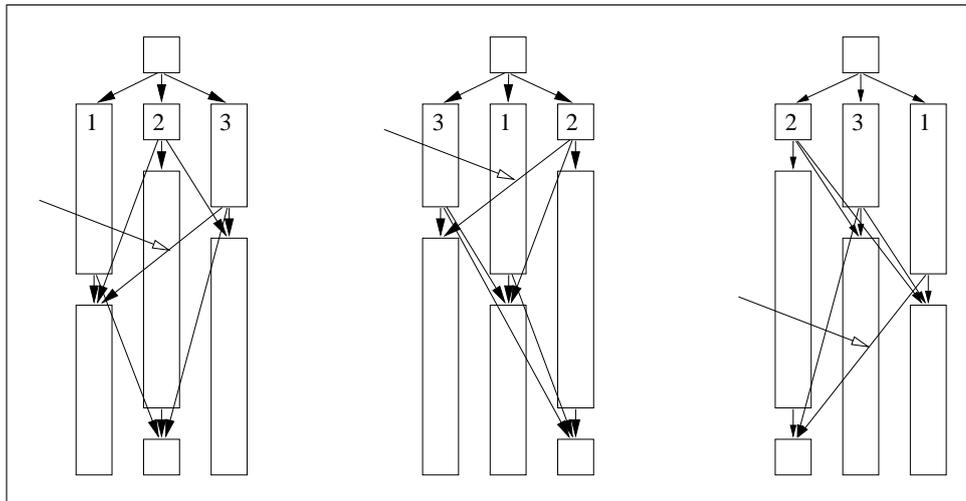
Bei größenechten Platzierungen geometrischer Graphen können Überdeckungen von Knoten mit Kanten unvermeidbar sein.

Beweis: Abbildung 6.5(a) zeigt einen geometrischen Graphen, dessen größenechte Platzierung stets zu Knoten-Kanten-Überdeckungen führt. Unabhängig von der Reihenfolge der Knoten innerhalb der Ebenen findet sich keine größenechte Platzierung, die frei von solchen Überdeckungen ist. In der Abbildung sind dazu drei Permutationen der Knoten 1,2,3 dargestellt, die anderen drei Permutationen dieser Knoten ergeben sich, indem jede Darstellung von rechts nach links betrachtet wird.

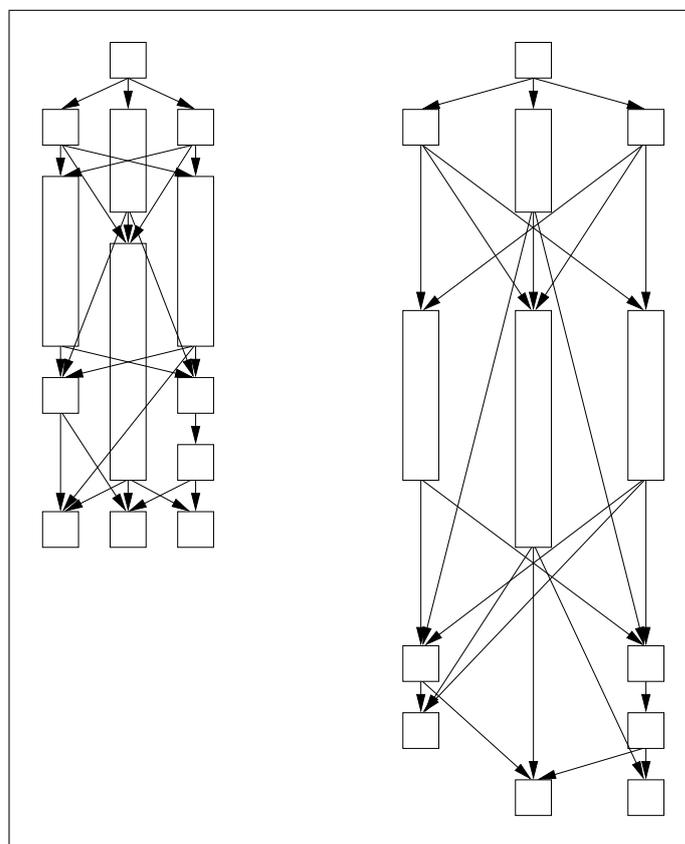
Die Breite der Visualisierung kann die Zahl der Überdeckungen verändern, die jeweils durch einen Pfeil gekennzeichnete Kante überdeckt jedoch auch in einer beliebige breiten Darstellung einen Knoten. Für diesen Graphen sind also Knoten-Kanten-Überdeckungen unvermeidbar. \square

Dagegen lassen sich Überdeckungen vermeiden, wenn die Höhe der Zeichnung vergrößert wird, siehe Abbildung 6.5(b). Die traditionellen größtenmaximalen und größenebenenmaximalen Platzierungen kommen stets ohne Überdeckungen von Knoten und Kanten aus, da die Höhe der Zeichnung genügend groß gewählt wird.

Eine Visualisierung mit Überdeckungen von Knoten mit Kanten ist nicht automatisch schlechter als eine ohne. So können Zeichnungen mit Überdeckungen schöner oder leichter verständlich als solche



a



b

Abbildung 6.5: **Knoten-Kanten-Überdeckungen.** (a) Bei grössenechten Platzierungen lassen sich Knoten-Kanten-Überdeckungen nicht immer vermeiden. Hier als Beispiel drei verschiedene Zeichnungen desselben Graphen, bei dem bei beliebiger grössenechter Platzierung Überdeckungen von Knoten und Kanten unvermeidbar sind. In jeder Darstellung ist eine nicht vermeidbare Überdeckung mit einem Pfeil gekennzeichnet.

(b) Das Vermeiden von Knoten-Kanten-Überdeckungen führt nicht immer zu besseren Zeichnungen, oft ist eine kompakte Darstellung mit kurzen Kanten übersichtlicher, da sich kurze Kanten leichter verfolgen lassen als lange.

ohne Überdeckungen sein, da sich beispielsweise die dabei entstehenden kurzen Kanten leichter verfolgen lassen. Überdeckungen von Knoten und Kanten sind der Preis für kompakte Zeichnungen.

6.1.3 Lagebeziehungen

6.1.3.1 Graph mit Lagebeziehungen

Bei Zeichnungen sollen oft vom Anwender oder einer Anwendung gegebene Wünsche an die Platzierung von Knoten berücksichtigt werden. Dadurch können neben der Struktur des Graphen auch semantische Aspekte, z. B. die Wichtigkeit von Knoten und Verbindungen, berücksichtigt werden.

Lagebeziehungen haben noch eine weitere Anwendung: Optimale ebenenweise Zeichnungen großer Graphen sind oft nicht effizient berechenbar, da viele NP-vollständige Probleme zu lösen sind. Selbst für kleine Graphen bewegen sich die Rechenzeiten oft im Minutenbereich. Im Allgemeinen hat ein Anwender eine genaue Vorstellung von einer *schönen* Zeichnung. Er sieht mögliche Verbesserungen, die ein automatisches Verfahren nur bei einer optimalen Lösung berechnen würde, meist sehr schnell. Durch Platzierungswünsche bzw. Lagebeziehungen hat der Anwender die Möglichkeit, dem Algorithmus Verbesserungen der Zeichnung vorzuschlagen.

Die Lagebeziehungen sollen so gestaltet sein, dass deren Wirkung für den Anwender möglichst offensichtlich ist, dass eine Lagebeziehung möglichst keine Seiteneffekte hat und dass Lagebeziehungen die Rechenzeit nicht negativ beeinflussen. Hier werden Lagebeziehungen in Form linearer Abhängigkeiten zwischen Knoten behandelt, die wie folgt definiert sind:

DEFINITION 6.4 (GRAPH MIT LAGEBEZIEHUNGEN)

Ein Graph mit Lagebeziehungen $G = (V, E, M, LB)$ besteht aus einem geometrischen Graph $G = (V, E, M)$ und einer Menge von Lagebeziehungen $LB \subseteq V \times V \times LBT \times \mathbb{N}$. Es werden folgende Lagebeziehungstypen $t \in LBT$ definiert, sei dazu $(u, v, t, g) \in LB$:

t	Beschreibung	Realisierung in einer Visualisierung
o-u	oben-unten Knoten u soll oberhalb von Knoten v liegen	$y_u(u) < y_o(v)$
hor	horizontal Knoten u und v sollen mit derselben y -Koordinate beginnen	$y_o(u) = y_o(v)$
l-r	links-rechts Knoten u soll links von Knoten v liegen, wenn sich beide Knoten innerhalb eines gemeinsamen horizontalen Bereichs befinden	$x_r(u) < x_l(v)$ falls $\exists y ((y_o(u) \leq y \leq y_u(u)) \wedge (y_o(v) \leq y \leq y_u(v)))$
ver	vertikal Knoten u und v sollen dieselbe x -Koordinate besitzen	$x(u) = x(v)$

Sei $G = (V, E, M, LB)$ ein Graph mit Lagebeziehungen. Die Abbildungen $A : LB \rightarrow V$, $Z : LB \rightarrow V$, $T : LB \rightarrow LBT$ und $W : LB \rightarrow \mathbb{N}$ ordnen jeder Lagebeziehung $(u, v, t, g) \in LB$ den zugehörigen Anfangsknoten $A((u, v, t, g)) = u$, Zielknoten $Z((u, v, t, g)) = v$, Lagebeziehungstyp $T((u, v, t, g)) = t$ und das Gewicht $W((u, v, t, g)) = g$ zu.

Auf die Behandlung von Lagebeziehungen in anderen Arbeiten wird bei der Betrachtung der einzelnen Lagebeziehungstypen eingegangen. Hier sei nur allgemein auf die Arbeiten [BP90, Bra95,

DETT99,DFM93,HG95,HM97,RMS97,San96b,Tam98] verwiesen, die sich mit der Behandlung von Lagebeziehungen in Zeichenverfahren beschäftigen und die Übersichten zu allgemeinen Lösungen und zu speziellen Ansätzen für ebenenweise Zeichenverfahren bieten.

Da bei geometrischen Graphen Knoten durch Rechtecke mit echten Größen repräsentiert werden, orientieren sich die Kriterien für o-u-, hor- und l-r-Lagebeziehungen an den Knotenrändern statt am Mittelpunkt der Knoten. Dadurch ist sichergestellt, dass z. B. ein Knoten u vollständig oberhalb eines Knotens v liegt ($y_u(u) < y_o(v)$). Lagebeziehungen werden auch *Constraints* genannt. Sie können gewichtet sein, um bei widersprüchlichen Lagebeziehungen⁶ die höhere Wichtigkeit einzelner Lagebeziehungen gegenüber anderen hervorzuheben, dabei seien im Folgenden die Gewichte für Lagebeziehungen stets größer eins.⁷

Die Einschränkung auf l-r-Lagebeziehungen innerhalb einer Ebene hat einen praktischen Grund, auf den bereits Waddle [Wad01] verweist: Oft werden solche Lagebeziehungen verwendet, um die Ordnung von Söhnen eines Knotens festzulegen. Dies macht jedoch nur dann Sinn, wenn diese auf derselben Ebene liegen bzw. gemeinsame y-Koordinaten besitzen. Wird dagegen auf Grund der Ebeneneinteilung ein Sohn über dem Vaterknoten platziert, so spielt dessen Anordnung in Bezug zu den anderen, unter dem Vaterknoten platzierten Söhnen keine Rolle. Zudem orientieren sich die Lagebeziehungen an den Phasen des Algorithmus für ebenenweise Zeichnungen. Werden l-r-Lagebeziehungen nur innerhalb der Ebenen betrachtet, so lassen sie sich leichter algorithmisch behandeln, da sie nur auf eine Phase des Verfahrens wirken. In Abschnitt 7.3.4 wird zusätzlich die Realisierung globaler l-r-Lagebeziehungen, die unabhängig von Ebenen sind, betrachtet.

Die hier definierten Lagebeziehungen ermöglichen an spezielle Anwenderwünsche angepasste Zeichnungen. Für die Biochemie sind solche Anforderungen beispielsweise, dass auf verschiedenen Reaktionswegen entstehende, chemisch ähnliche Substanzen auf der gleichen Ebene platziert (hor-Lagebeziehung) oder die Komponenten eines Hauptreaktionswegs in einer vertikalen Spalte dargestellt werden (ver-Lagebeziehung). Für die Darstellung spezieller Reaktionswege wie offene und geschlossene Zyklen werden o-u- und l-r-Lagebeziehungen benötigt.

Für einen Graph mit Lagebeziehungen $G = (V, E, M, LB)$ wird kurz $G = (V, E, LB)$ geschrieben, wenn die Geometrie nicht benötigt wird.

6.1.3.2 Einschränkung der Lagebeziehungen

Im weiteren Verlauf der Arbeit sollen nur Graphen betrachtet werden, deren Lagebeziehungen folgendem Kriterium genügen: Sei $G = (V, E, LB)$ ein Graph mit Lagebeziehungen. G sei frei von entgegengesetzten Lagebeziehungen gleichen Typs zwischen zwei Knoten $u, v \in V$, d. h. es gibt keine zwei Lagebeziehungen $(u, v, t, g_1) \in LB$ und $(v, u, t, g_2) \in LB$, die den gleichen Typ t und zwei beliebige Gewichte g_1, g_2 besitzen.

Dies schränkt nicht die Beeinflussbarkeit der Zeichnung durch Lagebeziehungen ein, da bei $t = o-u$ und $t = l-r$ prinzipiell nur einer der beiden entgegengesetzten Lagebeziehungen erfüllbar ist. Für $t = hor$ und $t = ver$ drücken beide Lagebeziehungen dasselbe aus, hier reicht eine der beiden Lagebeziehungen.

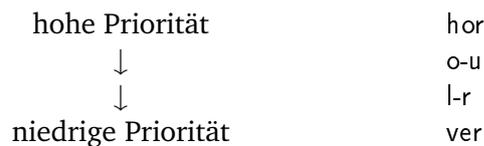
⁶Widersprüchliche Lagebeziehungen sind z. B. $(u, v, o-u, 2)$ und $(u, v, hor, 2)$ sowie zyklische Abhängigkeiten wie $(u, v, o-u, 2), (v, w, o-u, 2)$ und $(w, u, o-u, 2)$.

⁷Der Wert eins für Gewichte von Lagebeziehungen wird später teilweise verwendet, um mit diesen Lagebeziehungen normale Kanten zu repräsentieren.

6.1.3.3 Priorisierung von Lagebeziehungen

Lagebeziehungen können widersprüchlich sein. Solche Lagebeziehungen sind erlaubt, die Auflösung der Widersprüche, also die Auswahl nicht erfüllter Lagebeziehungen, erfolgt später durch den Platzierungsalgorithmus. Diese Auflösung der Widersprüche wird durch Prioritäten der Lagebeziehungstypen und das Gewicht der Lagebeziehungen unterstützt. Es gilt:

1. Widersprüche zwischen Lagebeziehungen mit verschiedenen Lagebeziehungstypen werden entsprechend folgender Priorisierung gelöst:



Diese Reihenfolge der Prioritäten ist durch die Phasen des Algorithmus gegeben, der zuerst die horizontale Platzierung der Knoten und danach deren Anordnung innerhalb der Ebenen bestimmt.

Wirksam wird die Priorisierung z. B. bei den widersprüchlichen Lagebeziehungen $(u, v, \text{hor}, 2)$ und $(u, v, \text{o-u}, 2)$, bei denen nur die Lagebeziehung $(u, v, \text{hor}, 2)$ erfüllt wird.

2. Widersprüche zwischen Lagebeziehungen mit gleichem Lagebeziehungstyp werden entsprechend der Gewichte dieser Lagebeziehungen gelöst, wobei hohe Gewichte einer hohen Priorität entsprechen.

Ein Beispiel sind die widersprüchlichen Lagebeziehungen $(u, v, \text{o-u}, 5)$, $(v, w, \text{o-u}, 6)$ und $(w, u, \text{o-u}, 2)$, bei denen nur die beiden Lagebeziehungen $(u, v, \text{o-u}, 5)$ und $(v, w, \text{o-u}, 6)$ erfüllt werden.

6.1.4 Zeichenkriterien

6.1.4.1 Einleitung

Für einen Graph⁸ $G = (V, N, M, LB)$ soll eine Platzierung $P = (P_V, P_E)$ berechnet werden. Die Anforderungen an die Platzierung bzw. an die daraus entstehende Zeichnung werden dabei in *Zeichenkonventionen* und *Ästhetikkriterien* unterteilt.⁹

6.1.4.2 Zeichenkonventionen

Eine ebenenweise Zeichnung *muss* folgenden *Zeichenkonventionen* genügen:

ZK1 Die Knoten sind auf horizontalen Ebenen angeordnet. Die Ebenen liegen parallel zueinander und sind von oben nach unten geordnet.

⁸Graphen $G = (V, E)$, benannte Graphen $G = (V, E, B)$, geometrische Graphen $G = (V, E, M)$ und geometrische Graphen mit Lagebeziehungen $G = (V, N, M, LB)$ werden im weiteren Verlauf einfach als Graphen bezeichnet, wenn aus dem Kontext hervorgeht, welcher Graphtyp gemeint ist.

⁹Eine Einführung zu Zeichenkonventionen und Ästhetikkriterien enthält Abschnitt 2.2.3.2.

ZK2 Kanten verlaufen von oben nach unten oder von unten nach oben.¹⁰

ZK3 Der vorgegebene Mindestabstand zwischen den Knoten ist eingehalten.¹¹

ZK4 Kanten dürfen sich kreuzen und Knoten überdecken, Kanten dürfen Knicke besitzen, zwischen denen sie geradlinig gezeichnet sind.

Diese Konventionen entsprechen im Wesentlichen den bekannten Konventionen für ebenenweise Zeichnungen in [DETT99, ES90, GNV88, STT81]. Nur das sonst übliche Verbot von Knoten-Kanten-Überdeckungen ist hier aufgehoben, weil solche Überdeckungen bei der Berechnung kompakter Platzierungen unvermeidbar sein können (siehe Satz 6.3).

6.1.4.3 Ästhetikkriterien

Während Zeichenkonventionen den Typ der Zeichnung bestimmen (z. B. ebenenweise Zeichnung oder planare Zeichnung), werden durch Ästhetikkriterien *schöne* Zeichnungen charakterisiert. Folgenden *Ästhetikkriterien* sollen die hier betrachteten ebenenweisen Zeichnungen genügen:

ÄK1 Lagebeziehungen zwischen Knoten sollen erfüllt werden.

ÄK2 Möglichst viele Kanten sollen von oben nach unten verlaufen.

ÄK3 Kanten sollen mit wenigen Knicken dargestellt werden.

ÄK4 Kanten sollen möglichst kurz und die Entfernung von Wurzeln zu Knoten möglichst gering sein.

ÄK5 Die Zahl der Kreuzungen zwischen Kanten und die Zahl der Überschneidungen zwischen Knoten und Kanten sollen klein sein.

ÄK6 Knoten sollen so auf den Ebenen platziert werden, dass sie horizontal zwischen ihren Vorgängern und Nachfolgern balanciert sind.

ÄK7 Die Zeichnung soll eine geringe Fläche einnehmen.

Diese Ästhetikkriterien werden allgemein mit schönen Darstellungen verbunden.¹² Da sich Ästhetikkriterien widersprechen können, sind diese entsprechend der obigen Anordnung, beginnend mit ÄK1 als wichtigstem Kriterium, geordnet.¹³

6.1.5 Grundidee des Zeichenverfahrens

Das Verfahren zum ebenenweisen Zeichnen von Graphen verwendet mehrere Phasen, statt alle Zeichenkonventionen und Ästhetikkriterien in einem Schritt zu berücksichtigen. Der Grund dafür

¹⁰Eine Kante mit Stützpunkten $(x_1, y_1), \dots, (x_n, y_n)$ verläuft von oben nach unten oder von unten nach oben, falls entweder für alle $1 \leq i < n$ gilt: $y_i < y_{i+1}$ oder falls für alle $1 \leq i < n$ gilt: $y_i > y_{i+1}$. Eine solche Kante wird auch als *streng y-monoton steigend* bzw. *fallend* bezeichnet.

¹¹Dadurch werden Knoten auch überdeckungsfrei dargestellt.

¹²Siehe auch die in Abschnitt 6.1.4.2 erwähnten Arbeiten sowie [BFN85, DM90].

¹³Hier sei wieder auf Abschnitt 2.2.3.2 und die dort erwähnten Arbeiten verwiesen. So stellen Purchase et al. in [PCJ96] fest, dass eine geringe Zahl von Kantenknicken und wenige Kreuzungen die Verständlichkeit von Zeichnungen erhöhen, während die Hervorhebung lokaler Symmetrien nur geringe Auswirkung auf die Verständlichkeit hat. Dies spiegelt sich auch in der Reihenfolge der Ästhetikkriterien wider.

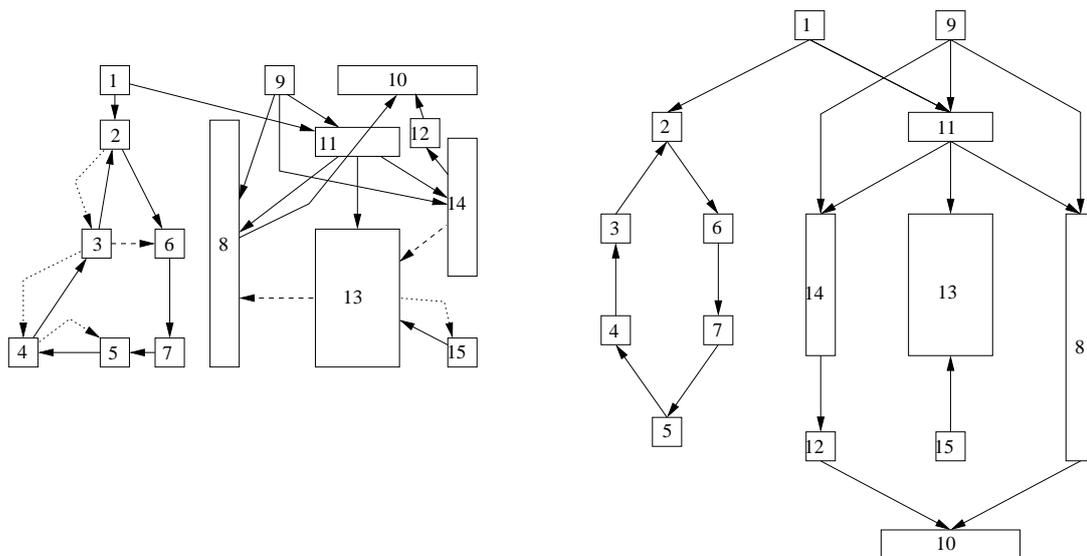


Abbildung 6.6: **Ebenenweises Zeichnen von Graphen (Gesamtverfahren)**. (links) Ausgangsgraph, (rechts) Ergebnis des VGL-Verfahrens. Die einzelnen Schritte des Verfahrens und die Bedeutung der Kantenarten sind in den Abbildungen 6.7 und 6.8 erläutert.

ist, dass viele Kriterien NP-hart zu optimieren sind und zusätzlich eine gute Lösung für ein Kriterium in Konflikt zu anderen Kriterien stehen kann. Es ist nicht zu erwarten, dass ein globales Optimum in erträglicher Zeit berechnet werden kann, weshalb die Idee der Phasenaufteilung bereits in [STT81] vorgeschlagen wurde. Dabei erfolgt die Aufteilung in einer sinnvollen Optimierungsreihenfolge, so dass die Lösungen der vorherigen Phasen möglichst erhalten bleiben. Dieses Phasenmodell hat sich vielfach bewährt und für das Zeichnen gerichteter Graphen durchgesetzt.

Um den Anforderungen aus den Abschnitten 6.1.4.2 und 6.1.4.3 gerecht zu werden, werden traditionelle Verfahren für ebenenweise Zeichnungen erweitert. Das neue Verfahren wird *Verfahren für größeunte ebenenweise Zeichnungen mit Lagebeziehungen* (kurz VGL-Verfahren) genannt und besteht aus vier Schritten (siehe auch Abb. 6.6 bis 6.8):

1. *Kantenorientierung*: Für den Graph G wird eine Kantenmenge E_R bestimmt, deren Umdrehen G azyklisch macht. Die Menge E_R wird dabei so gewählt, dass möglichst viele und hoch gewichtete o - u -Lagebeziehungen erfüllt und ansonsten möglichst wenige Kanten umgedreht werden. Die Kanten der Menge E_R bleiben bis zum Ende des VGL-Verfahrens umgedreht und werden erst nach dem letzten Schritt wieder wie ursprünglich orientiert.

Im Gegensatz zu herkömmlichen Verfahren, die bei der Beseitigung der Zyklen nur auf die Zahl der umgedrehten Kanten achten, sind hier o - u -Lagebeziehungen bei der Wahl dieser Kanten zu berücksichtigen. Dies wird ausführlich in Abschnitt 6.2 betrachtet.

2. *y -Koordinaten und Ebenenpartitionierung*: Den Knoten des im ersten Schritt berechneten Graphen werden y -Koordinaten zugeordnet. Für die Knoten wird eine minimale größeunte Platzierung berechnet. Darauf aufbauend werden Ebenen L_1, \dots, L_l für die Knoten bestimmt.

Knoten und Kanten können sich über mehrere Ebenen erstrecken. Solche großen Knoten werden in eine Kette *temporärer Knoten*¹⁴ aufgeteilt. Abschließend werden lange, über mehrere Ebenen gehende Kanten ebenfalls durch eine Kette temporärer Knoten ersetzt.

Im Gegensatz zu herkömmlichen Verfahren sind hier Knotengrößen im Sinn einer größenrechten Platzierung zu berücksichtigen, dies wird in Abschnitt 6.3 untersucht. Abschnitt 7.3.5 beschäftigt sich mit der Berücksichtigung traditioneller Strategien zur Ebenenpartitionierung im VGL-Verfahren.

3. *Kreuzungsreduzierung*: Die Anordnung der Knoten auf jeder Ebene L_1, \dots, L_l wird bestimmt. Diese ist so gewählt, dass möglichst viele und hoch gewichtete l-r-Lagebeziehungen zwischen den Knoten einer Ebene erfüllt werden und ansonsten die Zahl der Kreuzungen zwischen Kanten möglichst klein ist. Zudem werden große Knoten berücksichtigt, deren Anordnung über mehrere Ebenen erhalten bleiben muss.

Im Gegensatz zu herkömmlichen Verfahren, die bei der Kreuzungsreduzierung nur auf die Zahl der Kreuzungen achten, sind hier zusätzlich l-r-Lagebeziehungen und große Knoten zu berücksichtigen. Abschnitt 6.4 beschäftigt sich mit dieser Phase des VGL-Verfahrens.

4. *x-Koordinaten*: Die x-Koordinaten der Knoten werden so berechnet, dass lange Kanten möglichst geradlinig verlaufen und Teile großer Knoten dieselbe x-Koordinate erhalten, dies wird in Abschnitt 6.5 betrachtet.

Am Ende werden die temporären Knoten und Kanten entfernt und stattdessen an den zugehörigen Kanten Stützpunkte eingefügt bzw. die zugehörigen großen Knoten wieder zusammengefügt. Außerdem werden alle Kanten wieder in Originalrichtung orientiert.

In den Abbildungen 6.7 bis 6.9 ist auch dargestellt, auf welche Schritte die Zeichenkonventionen und Ästhetikkriterien jeweils wesentlichen Einfluss haben.

In den folgenden Abschnitten werden die eben skizzierten Schritte untersucht. Eingabe für den VGL-Algorithmus sind Graphen ohne Schlingen, die Erweiterung auf allgemeine Graphen mit reflexiven Kanten und Mehrfachkanten, die Berücksichtigung weiterer Lagebeziehungen sowie Besonderheiten werden in Kapitel 7 betrachtet. Der überwiegende Teil des VGL-Verfahrens und seiner Erweiterungen wurde im *BioPath*-Projekt realisiert. Wo dies nicht geschah, ist es entsprechend vermerkt.

¹⁴*Temporäre Knoten* sind Knoten, die während des Algorithmus in den Graphen eingefügt und vor Ende des Algorithmus wieder aus ihm entfernt werden. Sie werden auch als *virtuelle Knoten* bezeichnet. Entsprechend sind *temporäre Kanten* solche, die während des Algorithmus in den Graphen eingefügt und vor dessen Ende wieder aus dem Graphen entfernt werden.

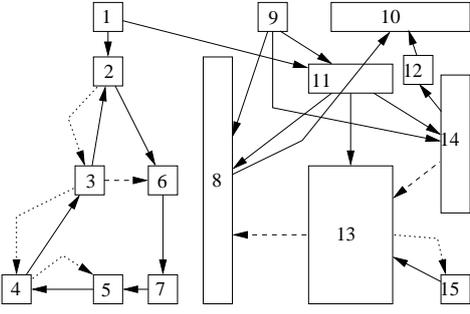
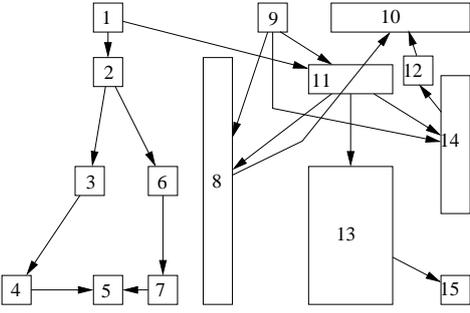
Schritt	Ergebnis	Beispiel	Konventionen und Ästhetik-kriterien
	Ausgangsgraph (gepunktet: o-u-Lagebeziehungen, gestrichelt: l-r-Lagebeziehungen)		
1. Kantenorientierung	azyklischer Graph, Kanten entsprechend der o-u-Lagebeziehungen gerichtet		ZK2 ÄK1 ÄK2

Abbildung 6.7: **Ebenenweises Zeichnen von Graphen (Teil 1)**. Schritte des Verfahrens, Ergebnisse und die Zeichenkonventionen und Ästhetikkriterien, welche die Schritte jeweils beeinflussen.

Schritt	Ergebnis	Beispiel	Konventionen und Ästhetik-kriterien
2. y-Koordinaten und Ebenenpartitionierung	partitionierter Graph mit y-Koordinaten, große Knoten und lange Kanten über mehrere Ebenen durch Ketten temporärer Knoten ersetzt (gestrichelt: l-r-Lagebeziehungen, runde Knoten: temporäre Knoten für lange Kanten, bei dicht benachbarten Knoten wurden die Pfeilspitzen weggelassen)		ZK1 ZK2 ZK3 ÄK1 ÄK4 ÄK7
3. Kreuzungsreduzierung	Anordnung der Knoten auf den Ebenen entsprechend der l-r-Lagebeziehungen und mit wenigen Kantenkreuzungen		ZK4 ÄK1 ÄK3 ÄK5

Abbildung 6.8: **Ebenenweises Zeichnen von Graphen (Teil 2)**. Schritte des Verfahrens, Ergebnisse und die Zeichenkonventionen und Ästhetikkriterien, welche die Schritte jeweils beeinflussen.

Schritt	Ergebnis	Beispiel	Konventionen und Ästhetik-kriterien
4. x-Koordinaten	Graph mit x-Koordinaten für Knoten		ZK3 ZK4 ÄK1 ÄK3 ÄK6 ÄK7
5.	Ergebnis des VGL-Verfahrens		

Abbildung 6.9: **Ebenenweises Zeichnen von Graphen (Teil 3)**. Schritte des Verfahrens, Ergebnisse und die Zeichenkonventionen und Ästhetikkriterien, welche die Schritte jeweils beeinflussen.

6.2 Kantenorientierung

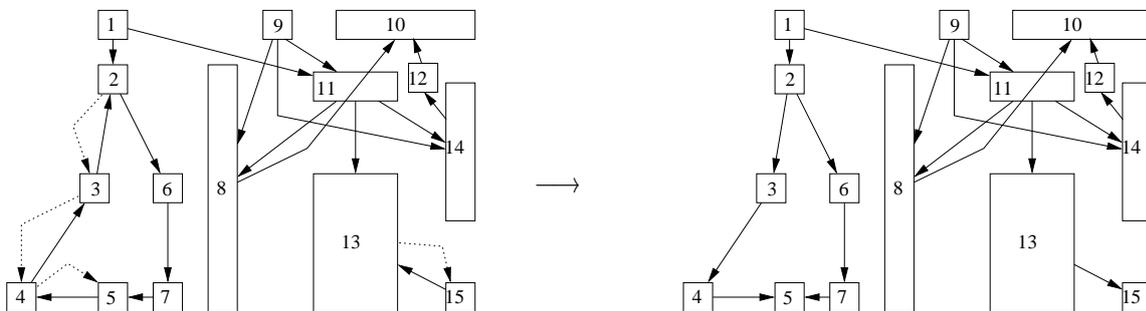
6.2.1 Überblick

Abschnitt 6.2 behandelt die erste Phase des VGL-Verfahrens: Die *Kantenorientierung*. Im Folgenden wird eine Kurzfassung des Abschnitts gegeben.

Ziel der Kantenorientierung:

Ausgangsgraph mit o-u-Lagebeziehungen \longrightarrow kantenorientierter Graph¹⁵, d. h.

- DAG
- Kanten entsprechend o-u-Lagebeziehungen gerichtet



Aufbau von Abschnitt 6.2:

- 6.2.2: Die Problemstellung wird eingeführt. Im Mittelpunkt stehen dabei zwei Möglichkeiten, wie widersprüchliche o-u-Lagebeziehungen berücksichtigt werden können. Dies führt zu zwei Ansätzen: Der *globalen* und der *lokalen* Kantenorientierung.
- 6.2.3: Die Ansätze zur globalen und lokalen Kantenorientierung werden untersucht. Für beide Probleme wird ihre NP-Vollständigkeit nachgewiesen.
- 6.2.4: Eine Heuristik zur Lösung der lokalen Kantenorientierung wird vorgestellt. Die Heuristik wird durch zwei Algorithmen realisiert.¹⁷ In Abschnitt 6.2.4.2 wird der Algorithmus KANTENORIENTIERUNG betrachtet, der eine leicht verständliche Realisierung der Heuristik darstellt. In Abschnitt 6.2.4.3 wird eine weitere Realisierung der Heuristik betrachtet, die EBENENGESTÜTZTE KANTENORIENTIERUNG. Dieser Algorithmus liefert dieselben Ergebnisse wie der Algorithmus KANTENORIENTIERUNG bei wesentlich kürzeren Laufzeiten.

¹⁵Der Begriff *kantenorientierter Graph* ist dadurch motiviert, dass Kanten entsprechend der gegebenen Bedingungen orientiert sind.

¹⁶Hier beschreibt die Heuristik die allgemeine Lösungs idee. Diese wird von der exakten algorithmischen Realisierung getrennt.

- 6.2.5: Die Algorithmen werden miteinander verglichen und es werden Eigenschaften der Heuristik betrachtet.
- 6.2.6: Es wird ein Überblick über traditionelle Verfahren zur Zyklensbeseitigung und zur Berücksichtigung von Lagebeziehungen gegeben.

6.2.2 Einführung

6.2.2.1 Grundlagen

Der erste Schritt des VGL-Algorithmus ist die Beseitigung der Zyklen im Graphen unter Beachtung der o-u-Lagebeziehungen. In einer Zeichnung sollen Kanten, die zu einem Zyklus führen, berücksichtigt werden. Diese Kanten werden deshalb bei der Zyklensbeseitigung nicht aus dem Graph entfernt, sondern bis zum Ende des VGL-Verfahrens in ihrer Richtung umgedreht.

Das Gewicht liegt hier auf der Entwicklung von Algorithmen mit geringer Laufzeit. Dies ist besonders wichtig, da die Kantenorientierung später auch als Teilalgorithmus der Kreuzungsreduzierung dienen soll und damit die Rechenzeit des Algorithmus für die Kreuzungsreduzierung beeinflusst.

Für die Kantenorientierung werden schwach zusammenhängende, einfache Graphen vorausgesetzt. Die Erweiterung auf beliebige Graphen wird in Kapitel 7 betrachtet.

DEFINITION 6.5 (EINFACHER GRAPH)

Sei $G = (V, E)$ ein Graph. G heißt einfach, wenn G keine Schlingen enthält und für je zwei Knoten $u, v \in V$ nur eine Kante $(u, v) \in E$ oder $(v, u) \in E$ existiert.

Einfache Graphen lassen sich auch dadurch charakterisieren, dass der unterliegende ungerichtete Graph schlingenfrei und frei von Mehrfachkanten ist.¹⁷ Es genügt, wenn im Folgenden nur einfache Graphen betrachtet werden. Berger und Shor stellen in [BS90] eine Technik zum Entfernen und Wiedereinfügen von Kanten aus Zyklen der Länge 2 vor, so dass sich Verfahren für einfache Graphen auch auf schlingenfreie Graphen mit Zyklen der Länge 2 anwenden lassen.

DEFINITION 6.6 (UMDREHEN VON KANTEN)

Sei $G = (V, E)$ ein einfacher Graph und $E_R \subseteq E$ eine Kantenmenge. $G' = (V', E')$ sei der Graph, der aus G durch Umdrehen der Richtung der Kanten $e \in E_R$ entsteht, dies wird als $G' = RK(G, E_R)$ notiert.

Der Graph G' ist dabei offensichtlich wieder einfach.

Für die Bewertung von Lösungen der Kantenorientierung ist es nötig, Zahlenfolgen zu vergleichen und eine minimale Folge zu bestimmen. Der Vergleich von Folgen natürlicher Zahlen beruht auf der lexikografischen Ordnung.

6.2.2.2 Erfüllte o-u-Lagebeziehungen

Nun sollen erfüllte von nicht erfüllten o-u-Lagebeziehungen unterschieden werden. Ob o-u-Lagebeziehungen erfüllt sind, lässt sich in einer Zeichnung anhand der y-Koordinaten der Knoten bestimmen. Für eine erfüllte o-u-Lagebeziehung $(u, v, o-u, g)$ gilt $y_u(u) < y_o(v)$, da es sich bei den hier

¹⁷Einfache Graphen werden auch als *frei von Schlingen und Zyklen der Länge zwei* [ELS93] oder als *orientierte Graphen* [BJG01] bezeichnet.

erzeugten Zeichnungen um so genannte *downwards*-Zeichnungen handelt, bei denen Kanten von oben nach unten verlaufen.

Im Moment haben Knoten jedoch noch keine y -Koordinaten. Bei der späteren Zuweisung der y -Koordinaten und der Ebenenpartitionierung des Graphen¹⁸ gilt für alle Kanten $(u, v) \in E : y_u(u) < y_o(v)$. Dieser Zusammenhang kann hier bereits ausgenutzt werden: Erfüllte o - u -Lagebeziehungen können durch entsprechend gerichtete Kanten repräsentiert werden. Im VGL-Verfahren werden dazu alle o - u -Lagebeziehungen durch Kanten dargestellt. Dadurch können diese Lagebeziehungen im nachfolgenden Schritt der Ebenenpartitionierung wie normale Kanten behandelt werden.

DEFINITION 6.7 (ERFÜLLTE o - u -LAGEBEZIEHUNG)

Sei $G = (V, E, LB)$ ein DAG und $(u, v, o-u, g) \in LB$ eine o - u -Lagebeziehung. Die o - u -Lagebeziehung heißt erfüllt, wenn $(u, v) \in E$ ist.

Nach diesen einführenden Definitionen wird nun untersucht, wie einfache Graphen unter Berücksichtigung von o - u -Lagebeziehungen in azyklische Graphen umgewandelt werden können. Das Problem ist eine mögliche Existenz widersprüchlicher o - u -Lagebeziehungen. Ein einfaches Beispiel ist ein Graph mit den Knoten a, b, c und den drei Lagebeziehungen $(a, b, o-u, 2)$, $(b, c, o-u, 2)$ und $(c, a, o-u, 2)$. Es gibt keinen azyklischen Graphen, der alle drei Lagebeziehungen erfüllt. Es können also Graphen mit nicht erfüllbaren o - u -Lagebeziehungen auftreten. Dabei besteht jedoch die Wahl, welche Lagebeziehungen nicht erfüllt werden. Im eben betrachteten Beispiel könnte jede der drei Lagebeziehungen als nicht erfüllbar ausgewählt werden, danach findet sich stets ein Graph, der azyklisch ist und die verbleibenden zwei Lagebeziehungen erfüllt.

Die Auswahl nicht erfüllter o - u -Lagebeziehungen wird durch deren Gewichte unterstützt. Das Gewicht einer Lagebeziehung gibt deren Wichtigkeit an, je höher eine Lagebeziehung gewichtet ist, umso wichtiger ist ihre Erfüllung. Dies führt zu zwei unterschiedlichen Ansätzen bei der Bestimmung eines azyklischen Graphen unter Auflösung widersprüchlicher Lagebeziehungen (siehe auch Abb. 6.10):

1. Es werden die Gewichte aller nicht erfüllten o - u -Lagebeziehungen summiert. Der Graph wird so in einen azyklischen Graph umgewandelt, dass diese Summe minimiert wird. Die Idee dieses Ansatzes ist eine globale Sicht auf die Lagebeziehungen, er soll darum im Folgenden als *globale Kantenorientierung* bezeichnet werden. Formal wird dies im nächsten Abschnitt in Definition 6.9 formuliert.
2. Es werden höher gewichtete Lagebeziehungen so lange wie möglich erfüllt. Dazu bilden die Gewichte der nicht erfüllten Lagebeziehungen eine absteigend sortierte Folge A . Der Graph wird so in einen azyklischen Graph umgewandelt, dass die Folge A bezüglich der lexikografischen Sortierung minimal wird. Die Idee dieses Ansatzes ist die besondere Beachtung hoch gewichteter Lagebeziehungen. Er hat eine mehr lokale Sicht auf hoch gewichtete Lagebeziehungen und soll im Folgenden als *lokale Kantenorientierung* bezeichnet werden. Formal wird dieser Ansatz im nächsten Abschnitt in Definition 6.10 formuliert.

In beiden Ansätzen soll zudem die Menge der umgedrehten Kanten unter den obigen Bedingungen minimal sein.

¹⁸siehe Abschnitt 6.3

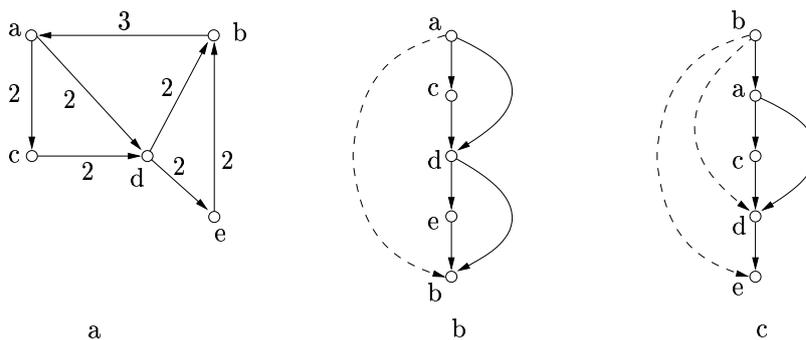


Abbildung 6.10: **Ansätze bei widersprüchlichen o-u-Lagebeziehungen.** (a) Graph G (jede Kante mit Gewicht), (b) globale Betrachtung, bei der W_{nicht} , die Summe der Gewichte nicht erfüllter Lagebeziehungen, minimiert wird (unerfüllte o-u-Lagebeziehungen sind gestrichelt dargestellt, es gilt $W_{\text{nicht}} = 3$), (c) lokale Betrachtung, bei der höher gewichtete Lagebeziehungen so lange wie möglich erfüllt werden (die Lagebeziehung von b nach a ist die Lagebeziehung mit dem höchsten Gewicht und muss auf jeden Fall erhalten bleiben). Es gilt nun $W_{\text{nicht}} = 4$.

6.2.3 Theorie

6.2.3.1 Kantenorientierungs-Probleme

Zunächst werden die beiden oben skizzierten Ansätze formal definiert. Normale Kanten und o-u-Lagebeziehungen sollen gemeinsam betrachtet werden. Dazu werden die Kanten des Graphen G gewichtet und o-u-Lagebeziehungen als gewichtete Kanten in G eingefügt, es wird ein *Lagebeziehungsgraph* aufgebaut.

DEFINITION 6.8 (LAGEBEZIEHUNGSGRAPH)

Sei $G = (V, E, LB)$ ein einfacher Graph mit Lagebeziehungen. Ein zu G gehörender Lagebeziehungsgraph $G' = (V', E', W)$ ist ein gewichteter Graph mit

$$\begin{aligned}
 V' &= V \\
 E' &= (E \setminus \{(v, u) \mid (u, v, o-u, g) \in LB\}) \cup \{(u, v) \mid (u, v, o-u, g) \in LB\} \\
 W((u, v)) &= \begin{cases} 1 & \text{falls } (u, v, o-u, g) \notin LB \\ g & \text{falls } (u, v, o-u, g) \in LB \end{cases}
 \end{aligned}$$

Offensichtlich ist ein zu einem einfachen Graph mit Lagebeziehungen $G = (V, E, LB)$ gehörender Lagebeziehungsgraph $G' = (V', E', W)$ wieder einfach.¹⁹

Die beiden Ansätze zur Erzeugung azyklischer Graphen bei widersprüchlichen Lagebeziehungen können nun definiert werden als:

¹⁹Das Gewicht $W(e) = 1$ für normale Kanten $e \in E$ ist nötig, damit bei der Summe der Gewichte der umgedrehten Kanten auch normale Kanten berücksichtigt werden. Deshalb wurde auch für Gewichte von Lagebeziehungen ein Wert größer eins gefordert.

DEFINITION 6.9 (GLOBALES KANTENORIENTIERUNGS-PROBLEM (GKP))

Sei $G = (V, E, W)$ ein Lagebeziehungsgraph. Gesucht ist eine Kantenmenge $E_R \subseteq E$, so dass $RK(G, E_R)$ azyklisch und $\sum_{e \in E_R} W(e)$ minimal ist.

DEFINITION 6.10 (LOKALES KANTENORIENTIERUNGS-PROBLEM (LKP))

Sei $G = (V, E, W)$ ein Lagebeziehungsgraph. Gesucht ist eine Kantenmenge $E_R \subseteq E$, so dass $RK(G, E_R)$ azyklisch und die absteigend sortierte Folge A der Gewichte nicht erfüllter o - u -Lagebeziehungen bezüglich der lexikografischen Sortierung minimal ist.

6.2.3.2 Komplexität der Probleme

Für die Komplexität der Probleme GKP und LKP werden zunächst das FEEDBACK SET PROBLEM und das zugehörige Entscheidungsproblem betrachtet.

DEFINITION 6.11 (FEEDBACK SET PROBLEM²⁰(FSP) [DETT99])

Sei $G = (V, E)$ ein Graph. Gesucht ist eine Kantenmenge $E_R \subseteq E$, so dass $RK(G, E_R)$ azyklisch und $|E_R|$ minimal ist.

DEFINITION 6.12 (FEEDBACK SET ENTSCHEIDUNGSPROBLEM (FSEP))

Instanz: Gerichteter Graph $G = (V, E)$, natürliche Zahl $k \leq |E|$

Frage: Gibt es eine Teilmenge $E_R \subseteq E$ mit $|E_R| \leq k$, so dass $RK(G, E_R)$ azyklisch ist?

SATZ 6.13 ([DETT99])

FSEP ist NP-vollständig.

Bemerkung: Das FEEDBACK SET PROBLEM ist gleich dem FEEDBACK ARC SET PROBLEM²¹, bei dem eine *minimale* Kantenmenge gesucht wird, deren *Entfernen* den Graph azyklisch macht.²²

Das FEEDBACK SET PROBLEM ist der Ansatz zur Untersuchung der Probleme GKP und LKP. Es ist offensichtlich, dass das GLOBALE KANTENORIENTIERUNGS-PROBLEM (GKP) eine Generalisierung

²⁰In der Literatur wird der Begriff FEEDBACK SET PROBLEM auch für die Gesamtheit von FEEDBACK VERTEX SET und FEEDBACK ARC SET Problemen verwendet, dies ist hier jedoch nicht gemeint.

²¹Das zugehörige Entscheidungsproblem ist wie folgt definiert:

FEEDBACK ARC SET ENTSCHEIDUNGSPROBLEM (FASEP) [GJ79, Kar72]

Instanz: Gerichteter Graph $G = (V, E)$, natürliche Zahl $k \leq |E|$

Frage: Gibt es eine Teilmenge $E_R \subseteq E$ mit $|E_R| \leq k$, so dass E_R mindestens eine Kante jedes Zyklus in G enthält?

Das FEEDBACK ARC SET ENTSCHEIDUNGSPROBLEM ist NP-vollständig [Kar72]. Es bleibt NP-vollständig für verschiedene Graphklassen, z. B. Graphen mit maximalem Knotengrad $\Delta(G) \leq 3$ [GJ79] sowie für Turniere [BJT92] (Ein Turnier ist ein einfacher Graph, bei dem jedes Knotenpaar durch eine Kante verbunden ist.) und damit auch für einfache Graphen.

Zum FEEDBACK ARC SET PROBLEM gibt es eine Version, bei der die Kantengewichte berücksichtigt werden. Dieses wird als WEIGHTED FEEDBACK ARC SET PROBLEM oder LINEAR ORDERING PROBLEM bezeichnet, das zugehörige Entscheidungsproblem ist ebenfalls NP-vollständig.

²²Hier ist die *Minimalität* der Mengen wichtig. Die Probleme, zur Zyklensbeseitigung eine Kantenmenge zu entfernen (*Feedback Arc Set*) oder umzudrehen (*Feedback Set*) sind *sonst nicht gleich*. Betrachtet man eine Menge E_{alle} , die alle Kanten der Zyklen eines Graphen enthält, so ist E_{alle} ein (nicht minimaler) *Feedback Arc Set*, da E_{alle} mindestens eine Kante jedes Zyklus enthält. Ein *Entfernen* der Kanten macht G azyklisch.

Werden die Kanten statt dessen *umgedreht*, so entstehen neue Zyklen, da die Richtung der ursprünglichen Zyklen nur umgedreht wird. E_{alle} ist ein *Feedback Arc Set*, aber kein *Feedback Set*.

des FEEDBACK SET PROBLEMS (FSP) ist. Dass zu GKP zugehörige Entscheidungsproblem ist NP-vollständig.²³

Nun wird noch das LOKALE KANTENORIENTIERUNGS-PROBLEM (LKP) betrachtet. Auch dabei handelt es sich um eine Generalisierung des FSP, die jedoch nicht so offensichtlich ist. Das zugehörige Entscheidungsproblem ist wie folgt definiert:

DEFINITION 6.14 (LOKALES KANTENORIENTIERUNGS-ENTSCHEIDUNGSPROBLEM (LKEP))

Instanz: Lagebeziehungsgraph $G = (V, E, W)$, absteigend sortierte Folge A natürlicher Zahlen mit $A \leq B$, wobei B die absteigend sortierte Folge der Gewichte $W(e)$ aller $e \in E$ ist.

Frage: Gibt es eine Teilmenge $E_R \subseteq E$, so dass $RK(G, E_R)$ azyklisch ist und dass für die absteigend sortierte Folge C der Gewichte $W(e)$ aller $e \in E_R$ gilt: $C \leq A$?

SATZ 6.15

LKEP ist NP-vollständig.

Beweis: LKEP ist in NP, da für eine Menge E_R in Polynomzeit getestet werden kann, ob $RK(G, E_R)$ azyklisch ist und $C \leq A$ gilt.

LKEP ist NP-vollständig, es ist eine Generalisierung des FEEDBACK SET ENTSCHEIDUNGSPROBLEMS (FSEP). Dazu betrachte man einen Graphen, bei dem die Menge der ursprünglichen o-u-Lagebeziehungen leer ist, also jede Kante $e \in E$ das Gewicht $W(e) = 1$ hat. Nun entspricht LKEP dem FSEP mit $k = |A|$, $|E| = |B|$ und $|C| = |E_R|$. \square

Zur Unterscheidung der Probleme GKP und LKP sei nochmals auf Abbildung 6.10 verwiesen, 6.10(b) entspricht einem minimalen Weighted Feedback Arc Set und damit einer Lösung des GLOBALEN KANTENORIENTIERUNGS-PROBLEMS, während 6.10(c) eine Lösung des LOKALEN KANTENORIENTIERUNGS-PROBLEMS ist.

6.2.3.3 Einschränkung auf lokale Kantensorientierung

Im Folgenden soll nur das LOKALE KANTENORIENTIERUNG-PROBLEM (LKP) weiter untersucht werden. Dies hat mehrere Gründe:

1. LKP entspricht der Intuition des Anwenders. Wichtiges wird von diesem mit hohen Gewichten versehen. Gewichte von Lagebeziehungen sollen deshalb so betrachtet werden, dass höher gewichtete Lagebeziehungen möglichst immer erfüllt werden, auch wenn dadurch eventuell eine größere Zahl geringer gewichteter Lagebeziehungen nicht erfüllt werden kann.
2. Lösungen des LKPs werden für die Heuristik der Kreuzungsreduzierung in Abschnitt 6.4 benötigt. Für die Entwicklung einer schnellen Kreuzungsreduzierung ist hier die Entwicklung einer effizienten Heuristik für LKP nötig.
3. Zur Lösung des GKP können Verfahren verwendet werden, die das WEIGHTED FEEDBACK ARC SET PROBLEM lösen, siehe beispielsweise [ENSS95, Flo90]. Es ist nicht nötig, hier weitere Algorithmen zu entwickeln.

²³Das GLOBALE KANTENORIENTIERUNGS-PROBLEM ist auch gleich dem WEIGHTED FEEDBACK ARC SET PROBLEM.

6.2.4 Algorithmen

Zur Lösung des Problems der lokalen Kantensorientierung wird eine Heuristik vorgestellt. Dabei werden zwei Algorithmen betrachtet, die die Idee der Heuristik unterschiedlich umsetzen: Zuerst der Algorithmus KANTENORIENTIERUNG, mit dem das Prinzip der Heuristik verdeutlicht werden soll und danach der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG, der dieselben Ergebnisse liefert, in der Praxis aber eine deutlich bessere Laufzeit hat.

6.2.4.1 Idee der Heuristik

Die Idee der Heuristik (und damit der beiden Algorithmen) ist, einen azyklischen Graphen durch sukzessives Hinzunehmen der Kanten entsprechend ihrer Gewichte zu konstruieren. Dabei werden höher gewichtete Kanten zuerst berücksichtigt. Kanten werden umgedreht, falls sonst ein Zyklus in dem neu aufgebauten Graph entstehen würde.

In den algorithmischen Realisierungen wird aus Gründen der Effizienz kein neuer Graph aufgebaut, sondern die Richtung ausgewählter Kanten im gegebenen Graph verändert. Es werden Kantenmarkierungen verwendet, um bereits betrachtete von noch nicht betrachteten Kanten zu unterscheiden. Für die Kanten des Graphen speichert das Feld $g[]$ ihr Gewicht, alle Kanten $e \in E$ haben zu Beginn das Gewicht $g[e] = 1$.

6.2.4.2 Algorithmus KANTENORIENTIERUNG

Der Algorithmus KANTENORIENTIERUNG besteht aus zwei Teilen:

1. (Teil 1, Zeilen 1-7) σ - u -Lagebeziehungen werden als Kanten in den Graph eingefügt. Wenn für eine Lagebeziehung (u, v, σ, g) bereits eine Kante zwischen den Knoten u und v existiert, so wird Richtung und Gewicht dieser Kante angepasst, ansonsten wird eine neue Kante eingefügt. Insgesamt erhält man einen Lagebeziehungsgraph.
2. (Teil 2, Zeilen 9-17) Die Kanten werden absteigend nach ihren Gewichten sortiert. Entsprechend der dabei erhaltenen Reihenfolge wird jede Kante getestet, ob sie in originaler Richtung erhalten bleiben kann oder umgedreht werden muss. Die Menge B enthält dazu die bereits besuchten Kanten. Kanten werden in originaler Richtung eingefügt, solange dadurch kein Zyklus im Graph (V, B) entsteht, ansonsten wird die Kante in umgekehrter Richtung eingefügt.

Nach Ausführung des Algorithmus enthält E_R die Menge aller in umgedrehter Richtung eingefügten Kanten.

Algorithmus: KANTENORIENTIERUNG.

Gegeben: Einfacher, schwach zusammenhängender Graph mit Lagebeziehungen $G = (V, E, LB)$.

Gesucht: Lokal kantensorientierter Graph.

```

1 /* Teil 1, Konstruktion des Lagebeziehungsgraphen */
2 forall (u, v, o-u, g) ∈ LB
3   if (v, u) ∈ E
4     Drehe Richtung von (v, u) um;  $E_R := E_R \cup \{(v, u)\}$ ;
5   if ((u, v) ∉ E) and ((v, u) ∉ E)
6     Füge neue Kante (u, v) in Graph (V, E) ein;
7    $g[(u, v)] := g$ ;
8
9 /* Teil 2, Konstruktion eines azyklischen Graphen */
10 B := ∅;
11 L := Liste aller e ∈ E;
12 Sortiere L entsprechend den Gewichten  $g[e]$  in absteigender Reihenfolge;
13 while Liste L nicht komplett durchlaufen
14   e := next24(L);
15   if e führt im Graph (V, B) zu einem Zyklus
16     Drehe Richtung von e um;  $E_R := E_R \cup \{e\}$ ;
17   B := B ∪ {e};

```

Einen einfachen Graph G mit $o-u$ -Lagebeziehungen und den durch den Algorithmus KANTENORIENTIERUNG berechneten Graph G' zeigt Abbildung 6.11.

SATZ 6.16

Sei $G = (V, E, LB)$ ein schwach zusammenhängender, einfacher Graph mit Lagebeziehungen. Der Algorithmus KANTENORIENTIERUNG beseitigt alle Zyklen in G .

Der Algorithmus KANTENORIENTIERUNG benötigt $O(e^2)$ mit $e = |E| + |OU|$ und OU ist die Menge der $o-u$ -Lagebeziehungen.

Die Beweise der Aussagen von Satz 6.16 lassen sich leicht erhalten. Sie sollen hier nicht betrachtet werden, da dieser Algorithmus im Folgenden nicht weiter verwendet wird. Er dient lediglich der Veranschaulichung der Idee der Heuristik.²⁵

²⁴Die Funktion $next(L)$ liefert das nächste Element der Liste L bzw. das erste Element, wenn noch kein Listenelement betrachtet wurde.

²⁵Die Zeitabschätzung ist mit $O(e^2)$ konservativ und geht davon aus, dass in jedem Schleifendurchlauf $O(e)$ für den Test auf einen Zyklus benötigt wird. Dazu kann z. B. mittels Tiefensuche getestet werden, ob für $e = (u, v)$ der Knoten v vom Knoten u aus im Graph $G = (V, B)$ erreichbar ist. (Genauer benötigt dieser Test in Zeile 15 nicht $O(e)$, sondern nur $O(|B|)$, da es ausreicht, die bisher in B eingefügten Kanten und zu diesen Kanten inzidenten Knoten zu berücksichtigen. Mit $|B| = i$ und i geht von 0 bis $(|E| + |OU| - 1)$ für die Anzahl der Schleifendurchläufe folgt $\sum_{i=0}^{|E|+|OU|-1} i$ und damit $O(e^2)$ für den gesamten Algorithmus.)

Ein Ansatzpunkt zur Verbesserung der Komplexität ist ein schnellerer Test auf einen Zyklus in Zeile 15. Der Trick, an jedem Knoten die erreichbaren Nachfolger zu speichern und beim Test nur noch nachzuschauen, ob der gesuchte Knoten in der Menge der Nachfolger enthalten ist, führt jedoch nicht zu einer Verbesserung der Komplexität. Das Problem ist, dass die Knotenmengen an jedem Knoten verfügbar sein müssen und dass beim Einfügen einer Kante in B u. U. bei $O(|V|)$ Knoten in die Mengen erreichbarer Nachfolger $O(|V|)$ Elemente eingefügt werden müssen. Auch eine Verwaltung der Mengen ähnlich zu Union-Find-Problemen [IR99] führte hier nicht weiter.

Die Gewichte der o-u-Lagebeziehungen spiegeln sich in den Gewichten der Kanten wider. Die Kanten sind absteigend sortiert, die Liste L besteht zuerst aus Kanten für o-u-Lagebeziehungen, erst danach aus originalen Kanten von G . o-u-Lagebeziehungen werden deshalb jeweils so lange erfüllt, wie sie keinen Zyklus bilden.

Nachteil des Algorithmus ist die hohe Rechenzeit, besonders bei dichten Graphen. Das Problem bei diesem Algorithmus ist, dass der Test auf Entstehen eines Zyklus in Zeile 15 unnötig oft durchgeführt wird. Abbildung 6.12 zeigt ein Experiment mit jeweils 100 kardinalitätsgleichen Graphen mit 50 Knoten und steigender Dichte. Dabei wurden in jedem Graph 25 Kanten als o-u-Lagebeziehungen betrachtet und mit zufälligen Gewichten versehen. Die hohen Rechenzeiten auch für kleine Graphen motivieren die Entwicklung einer schnellen Heuristik für die lokale Kantenorientierung.

6.2.4.3 Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG

6.2.4.3.1 Einführende Definitionen

DEFINITION 6.17 (L-EBENENPARTITIONIERUNG [DETT99, San96b])

Eine l -Ebenenpartitionierung eines Graphen $G = (V, E)$ ist eine Partitionierung V_1, \dots, V_l der Knoten und E_1, \dots, E_{l-1} der Kanten, so dass gilt:

- $V = V_1 \cup \dots \cup V_l, \quad V_i \cap V_j = \emptyset$ für $i \neq j$ und $1 \leq i, j \leq l$
- $E = E_1 \cup \dots \cup E_{l-1}, \quad E_i \cap E_j = \emptyset$ für $i \neq j$ und $1 \leq i, j < l$
- Für alle $(u, v) \in E$ mit $u \in V_i$ und $v \in V_j$ gilt $i < j$.

Für einen Graphen $G = (V, E)$ mit einer l -Ebenenpartitionierung V_1, \dots, V_l und E_1, \dots, E_{l-1} wird auch kurz $G = (V_1 \cup \dots \cup V_l, E_1 \cup \dots \cup E_{l-1})$ geschrieben. Eine l -Ebenenpartitionierung G lässt sich veranschaulichen, indem alle Knoten aus V_i auf der horizontalen Linie L_i nebeneinander angeordnet und die Linien L_1 bis L_l untereinander platziert werden. Jede Kante $e \in E$ geht von einer niedrigeren zu einer höheren Ebene.

6.2.4.3.2 Algorithmus

Um die Berechnung einer lokalen Kantenorientierung zu beschleunigen, werden Knoten Ebenen zugeordnet und Zyklen dadurch vermieden, dass alle Kanten stets von einer niedrigeren Ebene zu einer höheren Ebene, also von oben nach unten, gehen.

Im Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG wird zusätzlich zur Menge B , welche die bereits besuchten Kanten enthält, eine Menge M für bereits besuchte Knoten verwendet. Die Knoten werden erst dann in M eingefügt, wenn eine inzidente Kante in die Menge B eingefügt wird. Das Feld `ebene[]` ordnet jedem Knoten seine Ebene zu. Das Feld `g[]` wird für alle $e \in E$ mit 1 initialisiert. Wie im vorherigen Algorithmus wird zuerst im Teil 1 ein Lagebeziehungsgraph konstruiert. Anschließend werden im Teil 2 alle Kanten absteigend nach ihrem Gewicht sortiert. Entsprechend der dabei erhaltenen Reihenfolge wird der Graph schrittweise aufgebaut, indem jede Kante getestet wird, ob sie in originaler Richtung erhalten bleiben kann oder umgedreht werden muss. Dieser Test geschieht folgendermaßen:

Eine Kante (u, v) wird direkt in B eingefügt, wenn

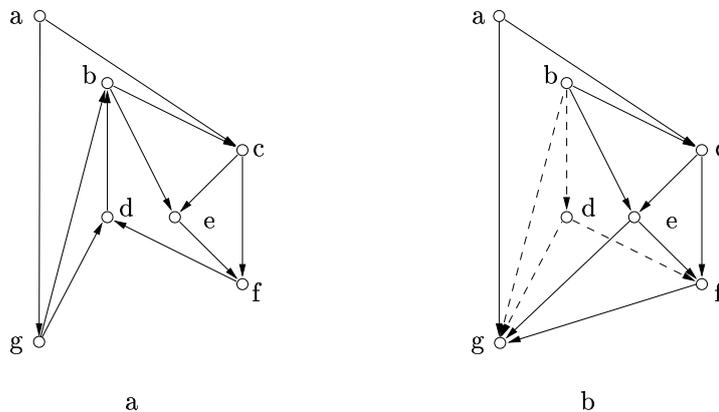


Abbildung 6.11: **Orientierung der Kanten.** (a) Graph $G = (V, E)$ mit Lagebeziehungen $\{(b, d, o-u, 2), (b, g, o-u, 2), (d, f, o-u, 2), (e, f, o-u, 2), (e, g, o-u, 2), (d, g, o-u, 2), (f, g, o-u, 2)\}$ und (b) kantenorientierter Graph G' mit zusätzlichen Kanten (e, g) und (f, g) . Die umgedrehten Kanten (also die Menge E_R) sind gestrichelt dargestellt.

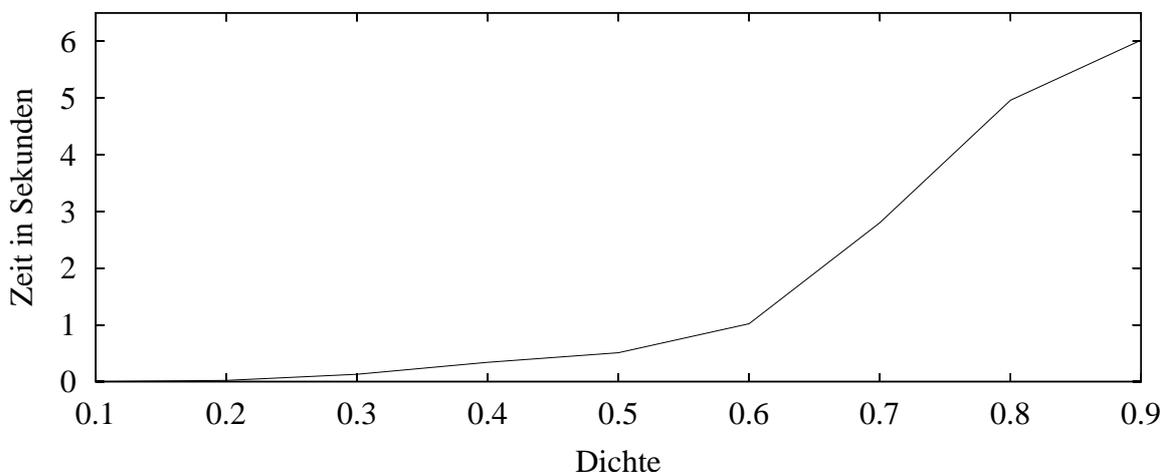


Abbildung 6.12: **Rechenzeiten des Algorithmus KANTENORIENTIERUNG.** Die durchschnittlichen Rechenzeiten für Graphen mit 50 Knoten und Dichten 0,1 bis 0,9 (in Schritten von 0,1; pro Schritt wurden 100 kardinalitätsgleiche Graphen untersucht). Bereits bei diesen relativ kleinen Graphen treten Rechenzeiten von mehreren Sekunden auf, was die Entwicklung eines schnelleren Verfahrens motiviert (siehe Abschnitt 6.2.4.3).

- $\text{ebene}[u] < \text{ebene}[v]$ gilt oder
- mindestens ein Knoten der Kante noch nicht besucht wurde. In diesem Fall sind die Ebenen der bzw. des neuen Knoten(s) anzupassen (siehe Zeilen 32-37).

Ist keiner dieser Fälle erfüllt, also insbesondere $\text{ebene}[u] \geq \text{ebene}[v]$, so wird getestet, ob durch Einfügen der Kante in den bisher aufgebauten Graph (M, B) ein Zyklus entsteht. Falls ein Zyklus entstehen würde, wird die Kante umgedreht. Falls (u, v) zu keinem Zyklus in (M, B) führt, so wird die Kante normal eingefügt und alle von v erreichbaren Knoten werden in höhere Ebenen verschoben (Zeilen 40-41). Dadurch wird sichergestellt, dass für alle bereits besuchten Kanten $(v_1, v_2) \in B$ stets gilt: $\text{ebene}[v_1] < \text{ebene}[v_2]$.

Algorithmus: EBENENGESTÜTZTE KANTENORIENTIERUNG.

Gegeben: Einfacher, schwach zusammenhängender Graph mit Lagebeziehungen $G = (V, E, LB)$.

Gesucht: Lokal kantenorientierter Graph.

```

18 /* Teil 1, Konstruktion des Lagebeziehungsgraphen */
19 forall (u, v, o-u, g) ∈ LB
20     if (v, u) ∈ E
21         Drehe Richtung von (v, u) um;  $E_R := E_R \cup \{(v, u)\}$ ;
22     if ((u, v) ∉ E) and ((v, u) ∉ E)
23         Füge neue Kante (u, v) in Graph (V, E) ein;
24     g[(u, v)] := g;
25
26 /* Teil 2, Konstruktion eines azyklischen Graphen */
27 B := ∅; M := ∅;
28 L := Liste aller e ∈ E;
29 Sortiere L entsprechend den Gewichten g[e] in absteigender Reihenfolge;
30 while Liste L nicht komplett durchlaufen
31     e := next(L) mit e = (u, v);
32     if (u ∉ M) and (v ∉ M)
33         ebene[u] := |V| - 1; ebene[v] := |V|;
34     if (u ∈ M) and (v ∉ M)
35         ebene[v] := ebene[u] + 1;
36     if (u ∉ M) and (v ∈ M)
37         ebene[u] := ebene[v] - 1;
38     if (u ∈ M) and (v ∈ M) and (ebene[u] ≥ ebene[v])
39         if ¬∃ Pfad v → u im Graph (M, B)
40             forall w ∈ EK26(v, (M, B))
41                 ebene[w] := ebene[w] + (ebene[u] - ebene[v] + 1);
42     else
43         Drehe Richtung von e um;  $E_R := E_R \cup \{e\}$ ;
44     B := B ∪ {e}; M := M ∪ {u, v};

```

²⁶EK(v, (V, E)) liefert die Menge der im Graph $G = (V, E)$ von v erreichbaren Knoten.

LEMMA 6.18

Für die **while**-Schleife des Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG gilt die Invariante:

$$\forall (v_1, v_2) \in B \quad \text{ebene}[v_1] < \text{ebene}[v_2] \quad (6.1)$$

Beweis: Invariante 6.1 gilt vor dem ersten Schleifendurchlauf, da $B = \emptyset$.

Angenommen, die Invariante gilt zu Beginn eines beliebigen Schleifendurchlaufs. Sei (u, v) die aktuell zu betrachtende Kante. In den Fällen

1. $\text{ebene}[u] < \text{ebene}[v]$
wird die Kante (u, v) direkt in die Menge B eingefügt.
2. $u \notin M$ oder $v \notin M$
wird die Kante (u, v) in B eingefügt, nachdem u, v oder beide Knoten einer Ebene mit $\text{ebene}[u] < \text{ebene}[v]$ zugewiesen wurden (Zeilen 32-37).
3. $\text{ebene}[u] > \text{ebene}[v]$ und \exists Pfad $v \rightarrow u$ in (M, B)
wird die Kante (u, v) umgedreht und in B eingefügt.

In diesen drei Fällen gilt die Invariante 6.1 offensichtlich auch nach Ende des Schleifendurchlaufs, da die Ebenen der zu anderen Kanten gehörenden Knoten nicht verändert werden. Nur zwei Fälle verdienen nähere Betrachtung:

4. Im Fall $\text{ebene}[u] \geq \text{ebene}[v]$ und $\neg \exists$ Pfad $v \rightarrow u$ in (M, B)
gilt: Da aus einer Erreichbarkeitskomponente keine Kanten herausgehen, gilt nach Verschieben aller von v erreichbaren Knoten in den Zeilen 40-41: $\forall (v_1, v_2) \in B \quad \text{ebene}[v_1] < \text{ebene}[v_2]$. Da auch der Knoten v verschoben wird, gilt auch $\text{ebene}[u] < \text{ebene}[v]$ am Ende des Schleifendurchlaufs.
5. Der Fall $\text{ebene}[u] = \text{ebene}[v]$ und \exists Pfad $v \rightarrow u$ in (M, B)
kann nie auftreten, da im Graph (M, B) wegen Invariante 6.1 zwischen zwei Knoten v_a und v_e nur dann ein Pfad existieren kann, falls gilt: $\text{ebene}[v_a] < \text{ebene}[v_e]$. Dies ist ein Widerspruch zu $\text{ebene}[u] = \text{ebene}[v]$.

Damit gilt in allen Fällen die Invariante auch nach Ende eines beliebigen Schleifendurchlaufs. \square

SATZ 6.19

Sei $G = (V, E, LB)$ ein schwach zusammenhängender, einfacher Graph mit Lagebeziehungen. Der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG beseitigt alle Zyklen in G .

Beweis: Nach jedem Durchlauf der **while**-Schleife, und damit auch nach ihrem letzten Durchlauf, gilt nach Lemma 6.18: $\forall (v_1, v_2) \in B \quad \text{ebene}[v_1] < \text{ebene}[v_2]$, der aus besuchten Knoten und Kanten bestehende Graph ist also azyklisch. Gleichzeitig wird der Graph vollständig betrachtet, da alle Kanten (originale ebenso wie die zusätzlichen für α - u -Lagebeziehungen) in die Liste L eingefügt, und damit in der Schleife betrachtet werden. \square

SATZ 6.20

Der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG benötigt $O(e^2)$ mit $e = |E| + |OU|$ und OU ist die Menge der o-u-Lagebeziehungen.

Beweis: Sei OU die Menge der o-u-Lagebeziehungen und $e = |E| + |OU|$. Das Einfügen von o-u-Lagebeziehungen in G benötigt $O(|OU|)$, da bei der Verwendung einer Adjazenzmatrix²⁷ in konstanter Zeit getestet werden kann, ob eine Kante bereits existiert. Das Sortieren der Liste L benötigt $O(e \log e)$.²⁸ Jeder Schleifendurchlauf läuft im günstigsten Fall in konstanter Zeit (Zeilen 31-37), im ungünstigsten Fall wird zur Berechnung des Pfads oder der Erreichbarkeitskomponente $O(e)$ benötigt. Die Komplexität des Algorithmus ist damit $O(e^2)$.²⁹ \square

6.2.5 Eigenschaften der Heuristik

Wird in den beiden Algorithmen KANTENORIENTIERUNG und EBENENGESTÜTZTE KANTENORIENTIERUNG die Liste L gleich sortiert, so sind auch die Mengen der umgedrehten Kanten gleich. In beiden Algorithmen wird eine Kante umgedreht, wenn sie im bisher aufgebauten Graph zu einem Zyklus führt, lediglich das Erkennen von Zyklen wurde im Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG anders gestaltet.

6.2.5.1 Laufzeiten

Zuerst sollen die Laufzeiten der beiden Algorithmen miteinander verglichen werden. Ziel ist es, zu zeigen, dass sich der Aufwand für den Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG lohnt. Die Abbildungen 6.13 bis 6.16 zeigen Ergebnisse von Laufzeittests. Dabei wurden Graphen von 10,20,30,...,200 Knoten mit Dichten zwischen 0,1 und 0,9 (in Schritten von 0,1) betrachtet. Für jeden dadurch entstehenden Graph wurden jeweils 100 kardinalitätsgleiche Graphen im Experiment untersucht.

Bereits für kleine, dichte Graphen mit 40 Knoten ist der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG deutlich schneller als die naive Kantenorientierung. Für Graphen mit mehr als 80 Knoten ist er auch für lichte Graphen schneller als der Algorithmus KANTENORIENTIERUNG. Besonders deutlich werden die Rechenzeitunterschiede in Abbildung 6.16, in der die Summe von Knoten und Kanten als Eingabegröße betrachtet wird.

Der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG ist auch zur Berechnung einer lokalen Kantenorientierung für größere Graphen gut geeignet. Selbst bei Graphen, die 10000 Knoten und Kanten enthalten, liegt die nötige Rechenzeit nur bei knapp einer Sekunde.

Die bei den Experimenten erzielten deutlich kürzeren Rechenzeiten für den Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG liegen darin begründet, dass nicht bei jedem Schleifendurchlauf auf das Entstehen eines Zyklus getestet werden muss.³⁰

²⁷Eine Adjazenzmatrix ist eine Datenstruktur zur Speicherung von Graphen, siehe z. B. [CH94].

²⁸Sortierverfahren und deren Komplexität sind in [Meh84a, Sed92] beschrieben.

²⁹Hier gelten wieder die Bemerkungen in Fußnote 25.

³⁰In der Realisierung lässt sich zudem die Suche nach einem Pfad zwischen zwei Knoten $v_1, v_2 \in V$ auf Ebenen zwischen v_1 und v_2 einschränken, diese Verbesserung wurde im VGL-Algorithmus realisiert.

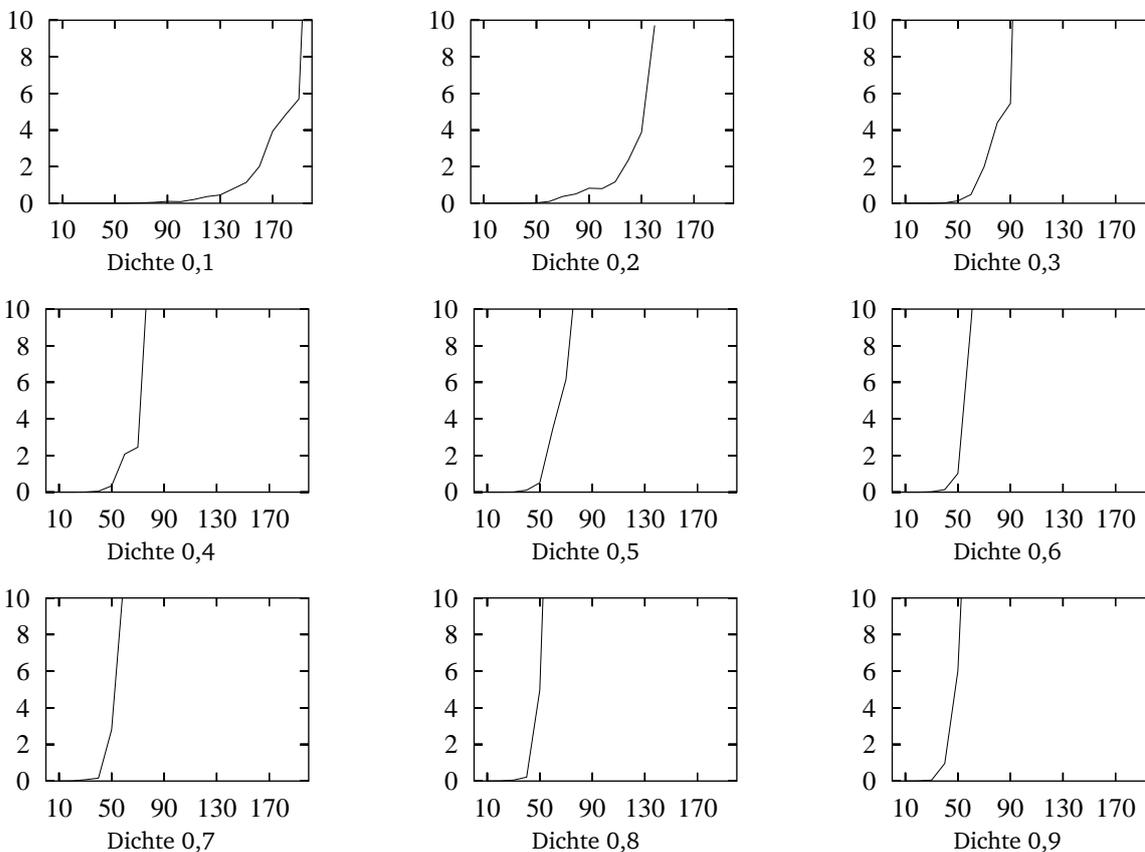
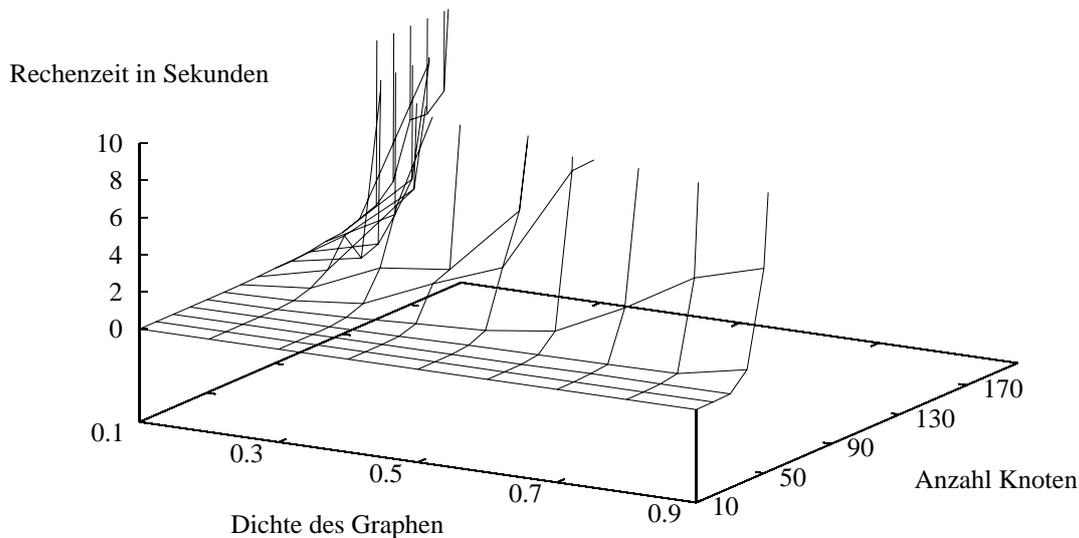


Abbildung 6.13: **Algorithmen zur Kantensorientierung - Rechenzeiten (Teil 1)**. Rechenzeiten des Algorithmus KANTENSORIENTIERUNG. Messergebnisse für Graphen mit 10,20,...,200 Knoten mit Dichten von 0,1 bis 0,9. Jeder Graph hat $\lfloor \frac{M}{2} \rfloor$ zufällig gewichtete Lagebeziehungen. Für jeden dadurch entstehenden Graph wurden 100 kardinalitätsgleiche Graphen untersucht und der Durchschnitt der Messergebnisse gebildet. Rechenzeiten von über 10 Sekunden sind im Diagramm nicht berücksichtigt (in Abb. 6.16 sind größere Werte dargestellt).

Rechenzeit in Sekunden

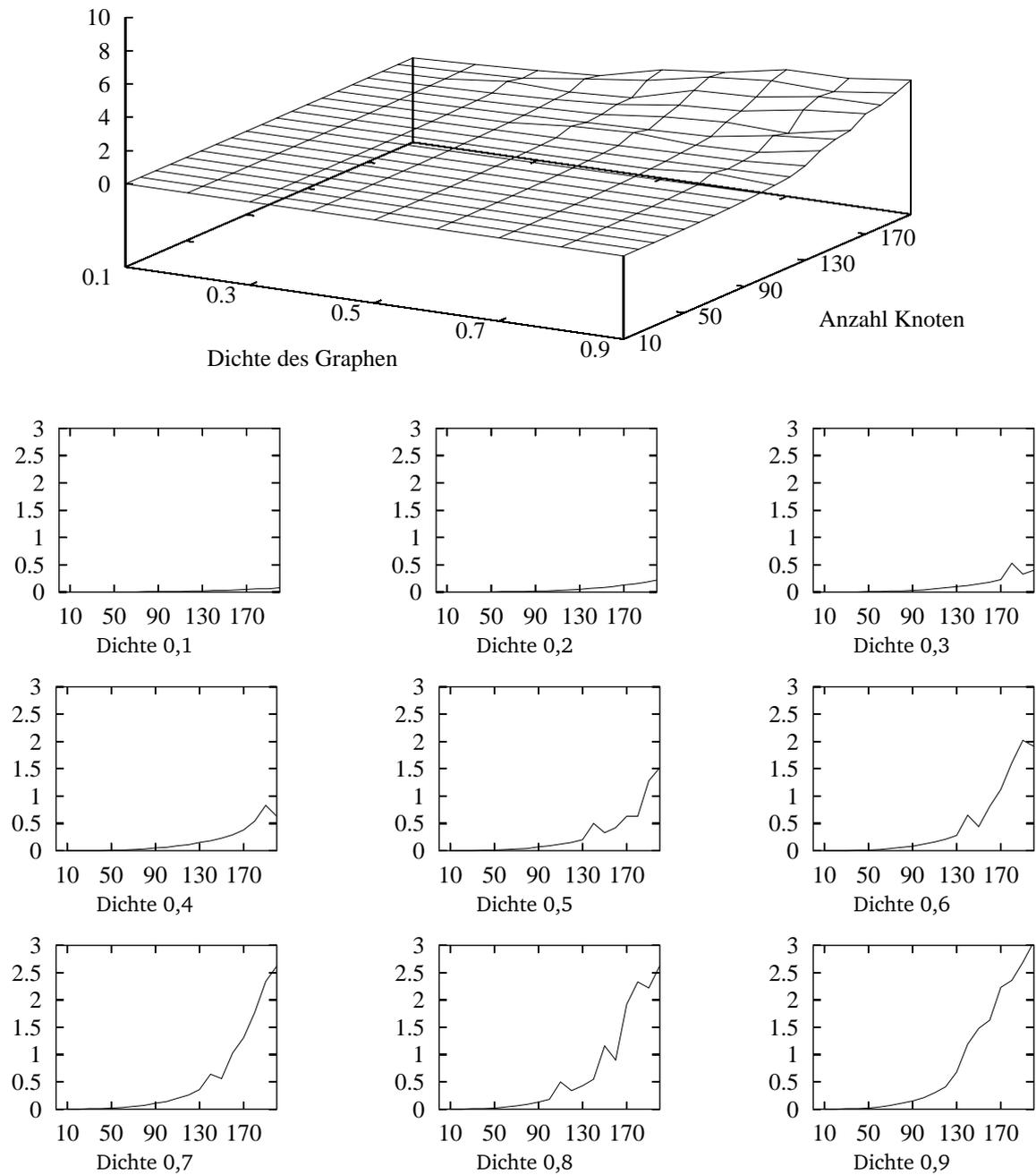


Abbildung 6.14: **Algorithmen zur Kantenorientierung - Rechenzeiten (Teil 2)**. Rechenzeiten des Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG. Es wurden dieselben Graphen wie in Abbildung 6.13 verwendet.

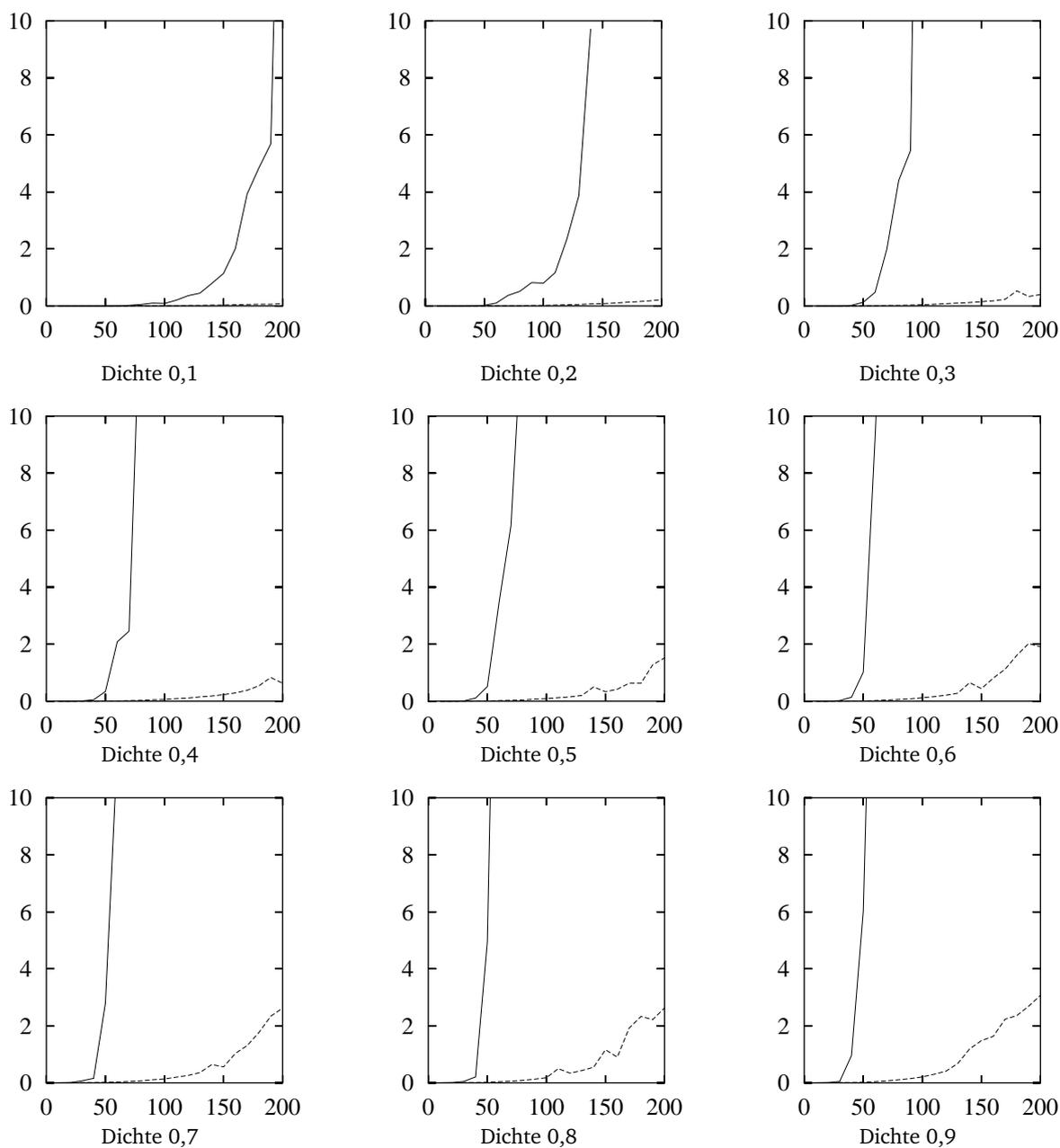


Abbildung 6.15: **Algorithmen zur Kantensorientierung - Rechenzeiten (Teil 3)**. Vergleich der Ergebnisse der beiden Algorithmen KANTENORIENTIERUNG und EBENENGESTÜTZTE KANTENORIENTIERUNG aus den Abbildungen 6.13 und 6.14.

Durchgezogene Linie: Algorithmus KANTENORIENTIERUNG

Gestrichelte Linie: Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG

x-Achse: Anzahl der Knoten, y-Achse: Rechenzeit in Sekunden

Selbst für dichte Graphen mit 100 Knoten und 4455 Kanten (Dichte 0,9) beträgt im Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG die Rechenzeit durchschnittlich nur 0,21 Sekunden.

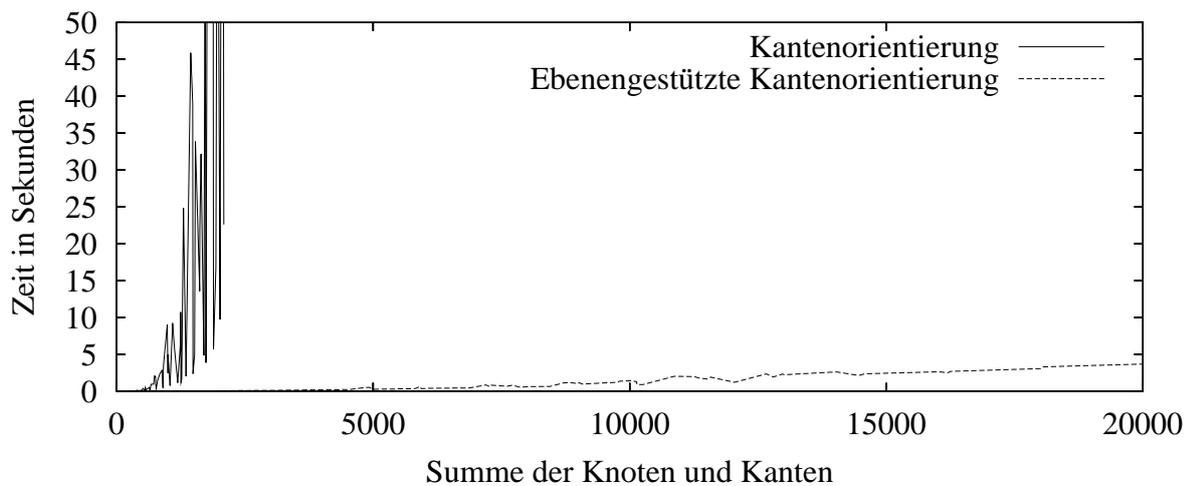


Abbildung 6.16: **Algorithmen zur Kantenorientierung - Rechenzeiten (Teil 4)**. Vergleich der Rechenzeiten der Algorithmen, bezogen auf die Summe von Knoten und Kanten. Es werden dieselben Graphen wie in den Abbildungen 6.13 und 6.14 betrachtet.

Die starken Schwankungen der Ergebnisse des Algorithmus KANTENORIENTIERUNG liegen am unterschiedlichen Einfluss von Knoten und Kanten. So benötigen lichte Graphen mit 150 Knoten und 1117 Kanten durchschnittlich 1,14 Sekunden, dichte Graphen mit 50 Knoten und 1102 Kanten dagegen durchschnittlich 6,05 Sekunden. In beiden Fällen ist die Summe von Knoten und Kanten fast gleich.

6.2.5.2 Qualitative Aussagen

Neben der Laufzeit interessiert das Ergebnis der Heuristik, also die Menge E_R . Da o-u-Lagebeziehungen den vorhandenen Kanten im Graphen entgegengesetzt sein können, lässt sich über die Zahl der umgedrehten Kanten nur folgende Aussage treffen:

SATZ 6.21

Die Algorithmen KANTENORIENTIERUNG und EBENENGESTÜTZTE KANTENORIENTIERUNG berechnen jeweils eine Kantenmenge E_R mit $|E_R| \leq (|E| + |OU|) - (|V| - 1)$. Dabei ist OU wieder die Menge der o-u-Lagebeziehungen.

Beweis: Es können mindestens $|V| - 1$ Kanten in Richtung von o-u-Lagebeziehungen oder originalen Kanten orientiert werden. Andererseits tritt der Fall $|E_R| = (|E| + |OU|) - (|V| - 1)$ beispielsweise bei einer Kette von n Knoten und $n - 1$ Kanten auf, wenn zu jeder Kante eine Lagebeziehung in zur Kantenrichtung entgegengesetzten Richtung vorhanden ist. \square

Sind keine o-u-Lagebeziehungen vorhanden, so wird durch die Algorithmen die Qualität einfacher traditioneller Verfahren zur Zyklensbeseitigung (z. B. mittels Tiefensuche, siehe Abschnitt 6.2.6.1) erreicht.

Wichtiger ist folgende Aussage:

SATZ 6.22

1. Falls für den einfachen Graph $G = (V, E, LB)$ eine Kantensorientierung existiert, die alle o - u -Lagebeziehungen erfüllt, so wird diese durch die Algorithmen KANTENORIENTIERUNG bzw. EBENENGESTÜTZTE KANTENORIENTIERUNG berechnet.
2. Falls für den einfachen Graph $G = (V, E, LB)$ eine Kantensorientierung existiert, die alle o - u -Lagebeziehungen erfüllt und bei der $E_R = \emptyset$ gilt, so wird diese durch die Algorithmen KANTENORIENTIERUNG bzw. EBENENGESTÜTZTE KANTENORIENTIERUNG berechnet.

Beweis: Kanten für o - u -Lagebeziehungen werden durch die Algorithmen zuerst berücksichtigt. Ist die Menge der o - u -Lagebeziehungen frei von Zyklen, so ist auch der Lagebeziehungsgraph frei von Zyklen und entsprechend den o - u -Lagebeziehungen gerichtet. Dadurch entsteht in beiden Algorithmen durch das Hinzunehmen von Kanten in der **while**-Schleife nie ein Zyklus, es wird also keine Kante umgedreht und die Menge E_R ist leer.

Dies gilt auch, falls zusätzlich bereits der aus Kanten für o - u -Lagebeziehungen und originalen Kanten bestehende Graph azyklisch ist. \square

6.2.6 Weitere Verfahren zur Kantensorientierung

6.2.6.1 Ohne Lagebeziehungen

Abschließend werden bekannte Verfahren zur Beseitigung von Zyklen durch Umdrehen von Kanten zusammengefasst. Dabei wird von schwach zusammenhängenden, einfachen Graphen ausgegangen.

So lässt sich mittels Tiefensuche und Umdrehen der Rückkanten in $O(|E|)^{31}$ eine Kantenmenge E_R mit $|E_R| \leq |E| - (|V| - 1)$ bestimmen [AHU74, Tar72]. Ein weiteres einfaches Verfahren ist die beliebige Aufreihung der Knoten von links nach rechts. Alle Kanten werden so orientiert, dass sie von links nach rechts verlaufen. Falls dazu mehr als die Hälfte der Kanten umgedreht wird, wird stattdessen die Richtung von rechts nach links für die Orientierung der Kanten gewählt. Auch dieses Verfahren läuft in $O(|E|)$ mit $|E_R| \leq \frac{|E|}{2}$. Ebenfalls $O(|E|)$ benötigt eine Heuristik, die alle Kanten des Graphen betrachtet und eine Kante in die Menge S einfügt, wenn Graph (V, S) azyklisch bleibt, sonst wird die Kante in die Menge T eingefügt. Die kleinere der beiden Mengen entspricht am Ende des Algorithmus der Menge E_R , auch hier gilt $|E_R| \leq \frac{|E|}{2}$. Die eben genannten Verfahren finden sich z. B. in [DETT99].

Ein besseres Ergebnis erhält man mit einer von Berger und Shor in [BS90] vorgestellten Heuristik. Das Verfahren berechnet $|E_R| \leq (\frac{1}{2} - \frac{c_1}{\sqrt{\Delta(G)}})|E|$, benötigt dafür aber $O(|E| * |V|)$.³² Hassin und Rubinstein [HR94] stellen für diesen Algorithmus eine Verbesserung der Zeitkomplexität mit $O(|E| + \Delta(G))$ vor, die vorteilhaft für Graphen mit kleinem Knotengrad ist. Eades et al. [ELS93] betrachten einen *Greedy-Cycle-Removal* genannten Algorithmus. Dieser löst die Zyklenbeseitigung in $O(|E|)$ mit $|E_R| \leq \frac{|E|}{2} - \frac{|V|}{6}$.

Eine exakte Lösung des Feedback Arc Set Problems wird beispielsweise in Form eines Branch-And-Bound-Verfahrens³³ von Kaas [Kaa81] vorgestellt.

³¹Da G schwach zusammenhängend ist, gilt $|V| \leq |E| + 1$.

³² c_1 ist dabei eine positive Konstante. Für einfache Graphen mit $\Delta(G) = 2$ oder $\Delta(G) = 3$ gilt beispielsweise $|E_R| \leq \frac{1}{3}|E|$.

³³Ziel von Branch-And-Bound-Verfahren ist das Finden der besten Lösung eines gegebenen ganzzahligen Optimierungs-

6.2.6.2 Mit Lagebeziehungen

o - u -Lagebeziehungen werden in traditionellen ebenenweisen Zeichenverfahren oft erst bei der Ebenenpartitionierung betrachtet. Da sie jedoch auch auf die Richtung von Kanten wirken, sollen sie hier zusammengefasst werden.

Gansner et al. [GKNV93] bieten in ihrem System DOT die Möglichkeit, Knoten auf die oberste oder unterste Ebene und Knotenmengen auf dieselbe Ebene zu platzieren. Dazu werden alle Kanten der obersten Knoten zu ausgehenden Kanten, alle Kanten der untersten Knoten zu eingehenden Kanten. Die Platzierung auf derselben Ebene wird durch temporäres Zusammenfassen von Knoten erreicht. Sanders [San96b] betrachtet Zyklenbeseitigung und Ebeneneinteilung als gemeinsamen Schritt und realisiert verschiedene Lagebeziehungen im VCG-System wie folgt: Die Lagebeziehungen werden zuerst in einen so genannten *Constraintgraph* übersetzt, um deren Erfüllbarkeit zu testen. Vor der Ebeneneinteilung werden Zyklen im Constraintgraph mittels einer Heuristik für das WEIGHTED FEEDBACK ARC SET PROBLEM beseitigt. Wird anschließend bei der Ebeneneinteilung ein Knoten v einer Ebene zugeordnet und kommt v auch im Constraintgraph vor, so wird die Ebene auf Grund der Information des Constraintgraphen bestimmt und dieser danach angepasst.

Der Vorteil dieses Verfahrens ist, dass auch Lagebeziehungen der Form *Knoten u soll genau 5 Ebenen höher als Knoten v liegen* berücksichtigt werden. Diese Lagebeziehungen sind bei größenmaximalen oder größenebenenmaximalen Platzierungen sinnvoll, da dabei die Ebenen für den Anwender offensichtlich sind. Bei den hier verwendeten größenrechten Platzierungen sind dagegen die Ebenen für einen Anwender meist nicht offensichtlich. Sie können sich zudem schon dadurch ändern, dass die Höhe eines Knotens verändert wird. Deshalb werden solche Lagebeziehungen hier nicht berücksichtigt. Nachteile von Sanders Lösung sind deren höhere Komplexität und der für die Realisierung nötige höhere Programmieraufwand. Zudem betrachtet Sanders o - u -Lagebeziehungen im Sinne einer globalen Kantenorientierung.

Auch Paulisch et al. [BP90, PT90] betrachten Lagebeziehungen in ebenenweisen Zeichenverfahren. Ihr Ansatz ist jedoch grundsätzlich anders: Die Behandlung der Lagebeziehungen wird nicht in andere Algorithmen integriert, sondern für alle Platzierungsentscheidungen werden Lagebeziehungen generiert. Der gesamte Prozess der Platzierung des Graphen ist anschließend ein Auflösen der verschiedenen Lagebeziehungen. Ein Beispiel ist die Berücksichtigung der oben-unten-Beziehung zwischen Knoten. Neben o - u -Lagebeziehungen wird für jede Kante (u, v) eine weitere Lagebeziehung der Form $\text{Rang}(u) + 1 \leq \text{Rang}(v)$ erzeugt. Nachteilig bei diesem Verfahren ist die große Zahl von Lagebeziehungen, die zur Berechnung einer Platzierung aufgelöst werden müssen.

problems. Das Problem wird dazu schrittweise in einen Baum von Teilproblemen zerlegt. Inneren Knoten des Baums wird ein Wert zugewiesen, der als Schranke für den Wert der Zielfunktion des Problems dient. Branch-And-Bound-Verfahren beruhen auf Backtracking. Das Verfahren bearbeitet zuerst Zweige, die einen größtmöglichen Wert für die Zielfunktion erwarten lassen. Ist ein Zweig fertig bearbeitet, werden all jene Zweige desselben Teilbaums entfernt, deren Schranken unterhalb des besten bisher erreichten Wertes der Zielfunktion liegen. Im schlechtesten Fall muss dabei jeder Zweig betrachtet werden.

6.3 y-Koordinaten und Ebenenpartitionierung

6.3.1 Überblick

Abschnitt 6.3 behandelt die zweite Phase des VGL-Verfahrens: Die *Bestimmung der y-Koordinaten* und die *Ebenenpartitionierung*. Im Folgenden wird eine Kurzfassung des Abschnitts gegeben.

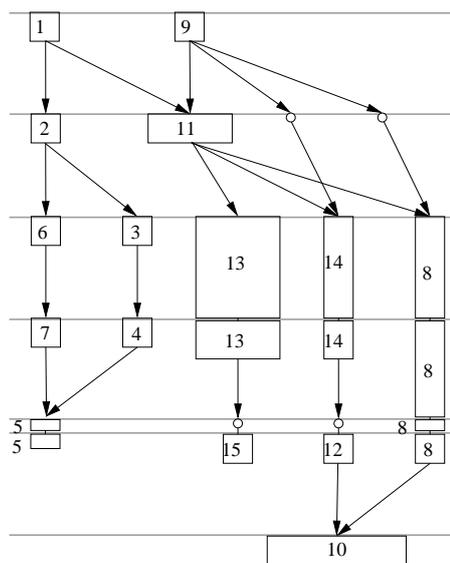
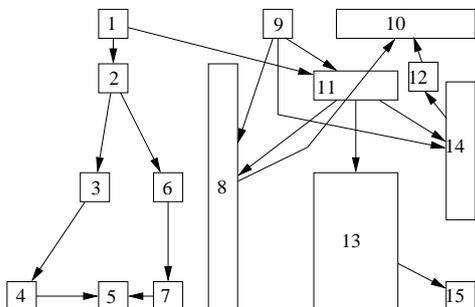
Ziel der Bestimmung der y-Koordinaten und der Ebenenpartitionierung:

kantenorientierter Graph (bzw. DAG)



y-Koordinaten für Knoten und einfache l-Ebenenpartitionierung, d. h.

- Zuordnung der Knoten zu Ebenen
- Kanten nur zwischen Knoten benachbarter Ebenen
- Aufteilung langer Kanten und großer Knoten in Ketten temporärer Knoten



Aufbau von Abschnitt 6.3:

6.3.2: Die Problemstellung wird eingeführt.

6.3.3: Zwei Verfahren zur Berechnung der y-Koordinaten der Knoten und der einfachen

l-Ebenenpartitionierung werden vorgestellt. In Abschnitt 6.3.3.1 wird der Algorithmus GRÖSSENECHTE EBENENEINTEILUNG betrachtet, der zuerst eine grössenechten Platzierung und darauf aufbauend eine *l*-Ebenenpartitionierung berechnet. Da bei dieser Lösung sehr viele temporäre Knoten entstehen können, wird in Abschnitt 6.3.3.2 ein weiterer Algorithmus, die RASTERGESTÜTZTE EBENENEINTEILUNG, betrachtet. Anschließend wird untersucht, wie verschiedene Werte des Parameters *Rasterhöhe* die Zeichnung beeinflussen.

6.3.4: Es wird ein Überblick über traditionelle Verfahren zur Bestimmung von Ebenen und *y*-Koordinaten der Knoten gegeben.

6.3.2 Einführung

Im zweiten Teil des VGL-Verfahrens werden *y*-Koordinaten für Knoten bestimmt und die Knoten in Ebenen partitioniert. Ziel ist die Berechnung einer grössenechten Platzierung und die Konstruktion einer einfachen *l*-Ebenenpartitionierung für einen geometrischen Graph $G = (V, E, M)$. Dabei sei G ein schwach zusammenhängender DAG, im Allgemeinen das Ergebnis der Kantenorientierung aus dem vorherigen Abschnitt. Bei der Ebenenpartitionierung wird der Graph verändert. Große Knoten über mehrere Ebenen werden in eine Kette kleiner, temporärer Knoten aufgespalten und lange Kanten über mehrere Ebenen (mit einer Spanne > 1) werden ebenfalls durch eine Kette temporärer Knoten ersetzt. Es gelten folgende Definitionen:

DEFINITION 6.23 (EIGENSCHAFTEN VON *l*-EBENENPARTITIONIERUNGEN [DETT99, San96b])

Sei $G = (V_1 \cup \dots \cup V_l, E_1 \cup \dots \cup E_{l-1})$ eine *l*-Ebenenpartitionierung.

Für $V_i = \{v_1, \dots, v_k\}$ heißt die Anordnung $L(V_i) = (v_1, \dots, v_k)$ die *i*-te Ebene von G .

Der Rang eines Knotens ist definiert als $\text{Rang}(v) = i \Leftrightarrow v \in V_i$.

Die Spanne einer Kante $(u, v) \in E$ ist gegeben durch $\text{Spanne}(u, v) = \text{Rang}(v) - \text{Rang}(u)$.

Eine *l*-Ebenenpartitionierung heißt einfach, wenn gilt: $\forall e \in E \text{ Spanne}(e) = 1$.

In einer einfachen *l*-Ebenenpartitionierung gibt es nur Kanten zwischen benachbarten Ebenen, es gilt $E_i \subseteq V_i \times V_{i+1}$. Für $L(V_i)$ wird auch kurz L_i geschrieben und für eine *l*-Ebenenpartitionierung mit fester Anordnung der Knoten auf den Ebenen kurz $G = (L_1 \cup \dots \cup L_l, E_1 \cup \dots \cup E_{l-1})$. Für einen Knoten $v \in L_i$ liefert $L_i(v)$ dann die *relative Position des Knotens* auf der Ebene, für $u, v \in L_i$ gilt $L_i(u) < L_i(v) \Leftrightarrow u$ liegt links von v auf der Ebene L_i .

Zur Berechnung der *y*-Koordinaten der Knoten und der einfachen *l*-Ebenenpartitionierung werden zwei Algorithmen vorgestellt: Der Algorithmus GRÖSSENECHTE EBENENEINTEILUNG und der Algorithmus RASTERGESTÜTZTE EBENENEINTEILUNG.

Der Algorithmus GRÖSSENECHTE EBENENEINTEILUNG besteht aus zwei Teilen: Zuerst wird eine grössenechte Platzierung berechnet. Darauf aufbauend wird der Graph in eine einfache *l*-Ebenenpartitionierung umgewandelt. Dieser Algorithmus eignet sich gut zur Darstellung des Konzepts und erleichtert das Verstehen des Verfahrens. Es wird sich jedoch zeigen, dass in die dabei berechnete *l*-Ebenenpartitionierung viele temporäre Knoten eingefügt werden. Dies ist negativ für die weiteren Schritte des VGL-Verfahrens, da deren Rechenzeit von der Zahl der Knoten abhängt. Deshalb wird von der strikten grössenechten Platzierung abgewichen und stattdessen der Algorithmus RASTERGESTÜTZTE EBENENEINTEILUNG vorgestellt, der nahe benachbarte Ebenen zusammenfasst. Dadurch lässt sich die Zahl der Ebenen und der temporären Knoten entscheidend verringern.

Bei größenechten Platzierungen können Knoten einer Ebene Auswirkungen auf die Platzierung von Knoten der nächste Ebene haben, siehe auch Abbildung 6.17. Dies ist ein Phänomen, das bei den traditionellen größenmaximalen und größenebenenmaximalen Platzierungen nicht auftrat. Damit alle Knoten der Ebene L_i bei der Platzierung der Knoten der Ebene L_{i+1} korrekt berücksichtigt werden, ist es nötig, den Mindestabstand zwischen Knoten ebenfalls zu betrachten. Der Mindestabstand a_{\min} wird deshalb zur Knotenhöhe zugefügt. Für jeden Knoten $v \in V$ mit $M(v) = (h, b)$ gilt für den oberen und unteren Knotenrand $y_o[v]$ und $y_u[v]$ ³⁴:

$$y_o[v] = y(v) - \frac{h}{2}$$
$$y_u[v] = y(v) + \frac{h}{2} + a_{\min}$$

6.3.3 Algorithmen

6.3.3.1 Algorithmus GRÖSSENECHTE EBENENEINTEILUNG

6.3.3.1.1 Einführung

Der Algorithmus EBENENEINTEILUNG besteht aus zwei Teilen und drei Teilalgorithmen (siehe auch Abb. 6.18):

1. (Teil 1, Zeilen 45-59) Der Berechnung einer größenechten Platzierung der Knoten entsprechend dem längsten Pfad zu jedem Knoten unter Berücksichtigung der Knotenhöhe. Damit verbunden ist die Berechnung der y_o -Koordinaten der Knoten (Algorithmus GRÖSSENECHTE PLATZIERUNG).
2. (Teil 2, Zeilen 60-79) Der Berechnung einer einfachen l -Ebenenpartitionierung mit Aufspalten großer Knoten (Algorithmus EBENENAUFTEILUNG) und langer Kanten (Algorithmus EBENENEINFACHHEIT). Dabei gehen große Knoten bzw. lange Kanten vor dem Aufspalten über mehrere Ebenen.

6.3.3.1.2 Teil 1

Zuerst wird der Algorithmus GRÖSSENECHTE PLATZIERUNG vorgestellt. Dabei speichert das Feld $ein[]$ für jeden Knoten die Zahl der noch nicht betrachteten eingehenden Kanten bzw. Vorgänger des Knotens. Ein Knoten $v \in V$ wird platziert, wenn alle seine Vorgänger platziert wurden, also $ein[v] = 0$ ist. Die Felder $y_o[]$ und $y_u[]$ enthalten die y -Koordinaten der oberen und unteren Knotenränder unter Beachtung des Mindestabstands.

³⁴In den Algorithmen werden die Werte für den oberen und unteren Knotenrand in Feldern gespeichert, deshalb wird hier $y_o[]$ und $y_u[]$ verwendet.

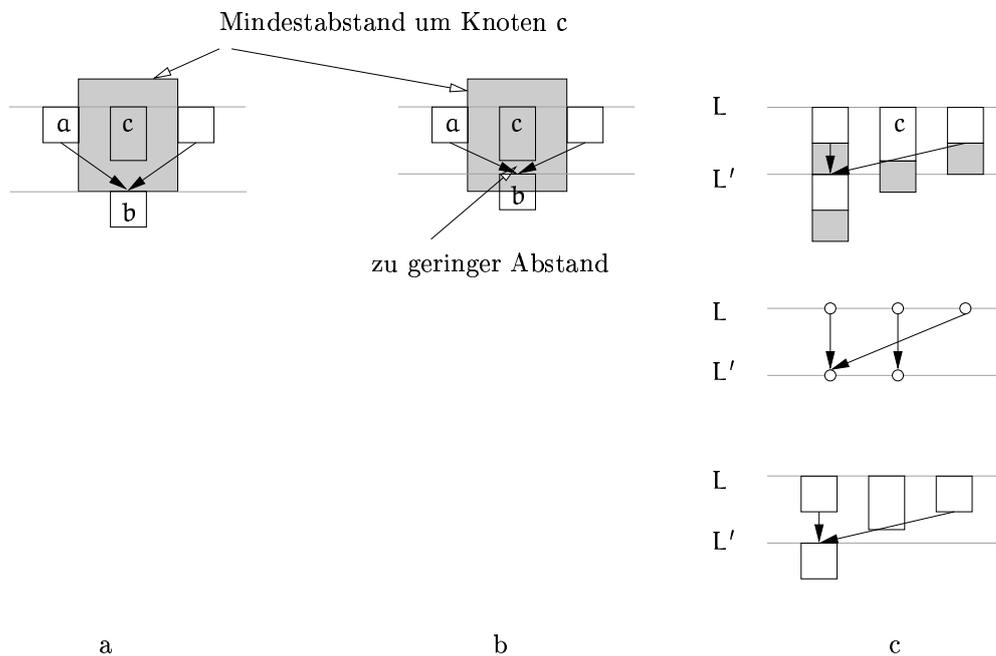
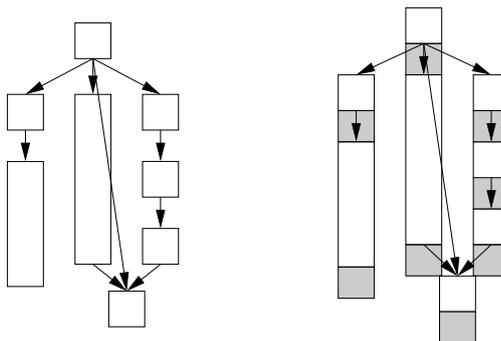


Abbildung 6.17: **Ebenenübergreifende Auswirkung einer Knotenplatzierung.** (a) In traditionellen Verfahren mit globaler Ebeneneinteilung beeinflusst der größte Knoten (oder der größte Knoten einer Ebene) die Lage der nächsten Ebene, Knoten einer Ebene ragen dabei nicht in die nächste Ebene hinein. Dadurch wird zwischen allen Knoten zweier Ebenen der (vertikale) Mindestabstand garantiert.

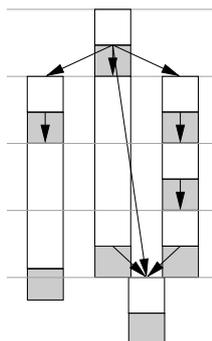
(b) Im Gegensatz dazu können nun bei größenrechten Platzierungen Knoten in die nächste Ebene hineinragen. Sie können sogar Platzierungen von Knoten in der nächsten Ebene behindern, ohne selbst direkt in diese Ebene hineinzuragen. Deshalb ist es nötig, den Abstand zwischen allen Knoten benachbarter Ebenen zu berücksichtigen. Anderenfalls kann die in (b) angedeutete Situation auftreten, dass zwischen nicht adjazenten Knoten der Mindestabstand nicht eingehalten wird. Während der Abstand zwischen den adjazenten Knoten a und b ausreicht, gilt dies nicht für den Abstand zwischen c und b. Knoten b befindet sich teilweise innerhalb des Mindestabstands um c.

(c) Eine Lösung für dieses Problem ist die Vergrößerung der Knotenhöhe um den (vertikalen) Mindestabstand, dies wird durch die grauen Bereiche angedeutet. Dadurch ragt der Knoten c in die nächste Ebene L' hinein, es wird ein temporärer Knoten auf diese Ebene eingefügt (siehe (c, mitte)). Damit wird Knoten c bei der Platzierung der Knoten der Ebene L' berücksichtigt. In (c, unten) ist eine solche korrekte Platzierung dargestellt. Die Überdeckung zwischen Knoten und Kante kann später durch einen zusätzlichen Kantenstützpunkt bzw. durch eine entsprechende Sortierung der Knoten vermieden werden.

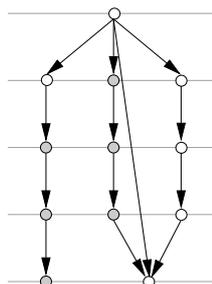
Nach Ausführung des Algorithmus GRÖSSENECHTE PLATZIERUNG. (Der vertikale Mindestabstand zwischen Knoten, der als zusätzliche Knotenhöhe eingeht, ist zur Verdeutlichung rechts grau dargestellt.)



Nach dem ersten Teil des Algorithmus EBENENAUFTEILUNG (Zuordnung der Ebenen). Jeder Wert $y_o[v], v \in V$ führt zu einer neuen Ebene.



Nach dem zweiten Teil des Algorithmus EBENENAUFTEILUNG. (Zuordnung der Knoten zu Ebenen, Aufspaltung großer Knoten, z. B. des mittleren Knoten in drei neue Knoten.)



Nach Ende des Algorithmus EBENENEINFACHHEIT.

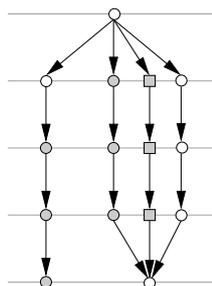


Abbildung 6.18: **Algorithmen GRÖSSENECHTE PLATZIERUNG, EBENENAUFTEILUNG und EBENENEINFACHHEIT.** Die Darstellung illustriert die Ergebnisse der Algorithmen. In den beiden unteren Bildern gilt: Erhalten gebliebene originale Knoten werden durch weiße Kreise, neu eingefügte Knoten für große Knoten durch graue Kreise und neu eingefügte Knoten für lange Kanten durch graue Rechtecke gekennzeichnet. Nach der Ebenenaufteilung wird die Knotenhöhe nicht mehr benötigt, deshalb werden Knoten gleicher Größe verwendet.

Algorithmus: GRÖSSENECHTE PLATZIERUNG.

Gegeben: Schwach zusammenhängender DAG $G = (V, E, M)$, Mindestabstand $\alpha_{\min} > 0$.

Gesucht: Minimale größenechte Platzierung von G .³⁵

Bemerkung: Bei dem Algorithmus handelt es sich um ein erweitertes topologisches Sortieren.

```

45 forall  $v \in V$ 
46      $\text{ein}[v] := \text{Ein}(v)$ ;
47      $h[v] := \text{Höhe}(v) + \alpha_{\min}$ ;
48 forall  $v \in V$ 
49     if  $\text{ein}[v] = 0$ 
50         Berechne  $y_o(v)$ ;
51
52 procedure Berechne  $y_o(v : \text{Knoten})$ 
53      $y_o[v] := \max(\{y_u[u] \mid u \in \text{Vor}(v)\} \cup \{0\})$ ;
54      $y_u[v] := y_o[v] + h[v]$ ;
55     forall  $w \in \text{Nach}(v)$ 
56          $\text{ein}[w] := \text{ein}[w] - 1$ ;
57         if  $\text{ein}[w] = 0$ 
58             Berechne  $y_o(w)$ ;
59 end procedure;
```

SATZ 6.24

Sei $G = (V, E, M)$ ein schwach zusammenhängender DAG. Der Algorithmus GRÖSSENECHTE PLATZIERUNG berechnet eine minimale größenechte Platzierung von G .

Beweis: Jeder Knoten $v \in V$ wird genau dann platziert, wenn $\text{ein}[v] = 0$ gilt. Dies tritt ein, wenn entweder der Knoten ein Wurzelknoten ist oder wenn alle seine Vorgänger bereits ihre y -Koordinate erhalten haben. Der zweite Fall gilt, da nach Platzierung eines Knotens für jeden seiner Nachfolger w der Wert von $\text{ein}[w]$ um 1 verringert wird.

Für Wurzelknoten $v \in V$ gilt $y_o[v] = 0$ (Zeile 53), für alle anderen Knoten $v \in V$ gilt $y_o[v] \geq y_u[u]$ für alle Vorgänger $u \in V$ (Zeile 53). Für jeden Knoten $v \in V$ ist im Wert von $y_u[v]$ der Mindestabstand berücksichtigt, so dass der Algorithmus eine größenechte Platzierung berechnet. Diese ist minimal, da ein Knoten $v \in V$ genau dann seine y_o -Koordinate erhält, wenn alle seine Vorgänger $u \in V$ ihre y_u -Koordinate erhalten haben. Dabei gilt $y_o[v] = y_u[u_m]$ des Vorgängers u_m mit der größte y_u -Koordinate. \square

SATZ 6.25

Der Algorithmus GRÖSSENECHTE PLATZIERUNG benötigt $O(|V| + |E|)$.

Beweis: Der Algorithmus GRÖSSENECHTE PLATZIERUNG erweitert die topologische Sortierung, die $O(|V| + |E|)$ benötigt [AHU83]. Die zusätzlichen Berechnungen von $y_o[v]$ und $y_u[v]$ werden für jeden Knoten $v \in V$ nur einmal ausgeführt und benötigen konstante Zeit, so dass die Gesamtkomplexität $O(|V| + |E|)$ ist. \square

³⁵Für einen Knoten $v \in V$ bestimmt der Wert $y_o[v]$ oder $y_u[v]$ stets auch die y -Koordinate von v , da die Höhe des Knotens gegeben ist. Es ist nicht nötig, die y -Koordinate des Knotens zusätzlich im Algorithmus zu berechnen.

6.3.3.1.3 Teil 2

Der zweite Teil besteht aus den Algorithmen EBENENAUFTEILUNG und EBENENEINFACHHEIT (siehe Abb. 6.18). Ziel des Algorithmus EBENENAUFTEILUNG ist die Berechnung der Ebenen in Abhängigkeit der y_o -Koordinaten, die Zuweisung der Knoten zu den Ebenen und die Aufspaltung großer Knoten.

Die y_o -Koordinaten der Knoten definieren die Ebenen. Im Algorithmus EBENENAUFTEILUNG ist L eine Liste von Tupeln der Form (*y-Koordinate der Ebene, Ebenennummer*). Falls mehrere Knoten mit gleicher y_o -Koordinate existieren, so kommt nur ein Tupel (y_o, i) in L vor. Das Feld `ebene[]` ordnet jedem Knoten seine Ebene zu.

Algorithmus: EBENENAUFTEILUNG.

Gegeben: Graph aus Algorithmus GRÖSSENECHTE PLATZIERUNG mit den Feldern `y_o[]`, `y_u[]`.

Gesucht: l -Ebenenpartitionierung unter Einfügen temporärer Knoten für große, über mehrere Ebenen gehende Knoten.

```

60 forall  $v \in V$ 
61   Ebene := Ebene  $\cup$   $(y_o[v], 0)$ ;
62  $L :=$  Liste aller  $eb \in$  Ebene;
63 Sortiere  $L$  entsprechend der 1. Komponente der Tupel (y-Koordinate) aufsteigend;
64 Nummeriere in  $L$  die 2. Komponente der Tupel (Ebenennummer) fortlaufend von 1 bis  $l$ ;
65
66 /* Zuordnung von Ebenen zu Knoten, Aufspalten großer Knoten */
67 forall  $v \in V$ 
68   Bestimme  $i$ , so dass  $(y_o[v], i) \in L$ ;
69   Bestimme  $j$ , so dass  $(y, j) \in L$  mit  $y = \max\{y_k \mid (y_k, k) \in L, y_k < y_u[v]\}$ ;36
70   if  $j > i$ 
71     Ersetze Knoten  $v$  durch eine Kette  $(v_i, \dots, v_j)$  mit  $j - i + 1$  Knoten;
72      $ebene[v_i] := i$ ;  $ebene[v_{i+1}] := i + 1$ ;  $\dots$ ;  $ebene[v_j] := j$ ;
73   else
74      $ebene[v] := i$ ;

```

Der folgende Algorithmus EBENENEINFACHHEIT berechnet eine einfache l -Ebenenpartitionierung, indem lange Kanten durch Ketten von Knoten ersetzt werden.

Algorithmus: EBENENEINFACHHEIT.

Gegeben: l -Ebenenpartitionierung aus Algorithmus EBENENAUFTEILUNG mit dem Feld `ebene[]`.

Gesucht: Einfache l -Ebenenpartitionierung unter Einfügen temporärer Knoten für lange Kanten.

```

75 forall  $(u, v) \in E$ 
76    $i := ebene[u]$ ;  $j := ebene[v]$ ;
77   if  $j > i + 1$ 
78     Ersetze Kante  $(u, v)$  durch eine Kette  $(v_{i+1}, \dots, v_{j-1})$  mit  $j - i - 1$  Knoten;
79      $ebene[v_{i+1}] := i + 1$ ;  $\dots$ ;  $ebene[v_{j-1}] := j - 1$ ;

```

³⁶Die Ebene j entspricht also der untersten Ebene, bis zu der der Knoten v reicht.

SATZ 6.26

Sei $G = (V, E, M)$ ein schwach zusammenhängender DAG und sei eine größenechte Platzierung durch $y_o[], y_u[]$ gegeben. Die Abfolge der Algorithmen EBENENAUFTEILUNG und EBENENEINFACHHEIT berechnen eine einfache l-Ebenenpartitionierung $G' = (V', E')$ mit $V' = V'_1 \cup \dots \cup V'_l$ und $E' = E'_1 \cup \dots \cup E'_{l-1}$.

Beweis: Es ist zu zeigen, dass gilt: $\forall (u, v) \in E' \text{ ebene}[v] = \text{ebene}[u] + 1$.

Zuerst wird gezeigt, dass $\forall (u, v) \in E' \text{ ebene}[v] > \text{ebene}[u]$ gilt. Da für G eine größenechte Platzierung gegeben ist, gilt vor Beginn der Algorithmen:

$$\forall (u, v) \in E \quad y_o[v] \geq y_u[u] \quad (6.2)$$

Da weiterhin die Knotenhöhe größer Null ist, gilt auch:

$$\forall v \in V \quad y_u[v] > y_o[v] \quad (6.3)$$

In Zeile 63 werden die Ebenen entsprechend den y_o -Koordinaten aufsteigend sortiert, deshalb gilt wegen 6.2 und 6.3:

$$\forall (u, v) \in E \quad \text{ebene}[v] > \text{ebene}[u] \quad (6.4)$$

Große Knoten $v_g \in V$ werden in eine Kette kleiner Knoten v_1, \dots, v_k aufgeteilt. Dabei gilt für v_1 : $\text{ebene}[v_1] = \text{ebene}[v_g]$. Es gilt nach 6.4: $\forall (u, v_g) \in E \quad \text{ebene}[v_g] > \text{ebene}[u]$, also gilt auch: $\forall (u, v_1) \in E' \quad \text{ebene}[v_1] > \text{ebene}[u]$. Der Knoten v_k wird einer Ebene j zugeordnet, für deren y -Koordinate y_j gilt: $y_u[v_g] > y_j = y_o[v_k]$ (Zeile 69). Nach 6.4 gilt: $\forall (v_g, w) \in E \quad \text{ebene}[w] > j$ und damit gilt auch: $\forall (v_k, w) \in E' \quad \text{ebene}[w] > j = \text{ebene}[v_k]$. Für v_g wird eine Kette temporärer Knoten eingefügt. Für die eingefügten Knoten v_1, \dots, v_k gilt: $\text{ebene}[v_k] > \dots > \text{ebene}[v_2] > \text{ebene}[v_1]$ (Zeile 72).

Damit gilt für die l-Ebenenpartitionierung nach dem Algorithmus EBENENAUFTEILUNG $\forall (u, v) \in E' \quad \text{ebene}[v] > \text{ebene}[u]$, die Ebeneneinteilung ist eine l-Ebenenpartitionierung.

Nun ist noch die Einfachheit der l-Ebenenpartitionierung zu zeigen. Falls es Kanten $(u, v) \in E'$ mit $\text{ebene}[v] > \text{ebene}[u] + 1$ gibt, so werden diese Kanten durch Ketten von Knoten ersetzt, so dass anschließend $\forall (u, v) \in E' \quad \text{ebene}[v] = \text{ebene}[u] + 1$ gilt (Zeilen 75-79).

Also gilt: $\forall (u, v) \in E' \quad \text{ebene}[v] = \text{ebene}[u] + 1$. □

SATZ 6.27

Die Algorithmen EBENENAUFTEILUNG und EBENENEINFACHHEIT benötigen $O(|V| * |E|)$.

Beweis: Die Liste L hat maximal $|V|$ Einträge, deren Sortieren benötigt $O(|V| \log |V|)$. Da bis zu $|V|$ Ebenen entstehen können und für jeden große Knoten und jede lange Kante dadurch bis zu $|V|$ temporäre Knoten eingefügt werden können, ist die Komplexität der **forall**-Schleife in den Zeilen 67-74 $O(|V| * |V|)$ und die Komplexität der **forall**-Schleife in den Zeilen 75-79 $O(|V| * |E|)$. G ist schwach zusammenhängend, damit gilt: $|E| \geq |V| - 1$. Die Gesamtkomplexität ist $O(|V| * |E|)$. □

6.3.3.1.4 Bewertung

Der Nachteil dieses Verfahrens ist die große Zahl temporärer Knoten, die beim Aufspalten großer Knoten und langer Kanten entsteht.

Selbst wenn sich Knoten in ihren y_o -Koordinaten nur minimal unterscheiden, wird für jeden dieser Knoten eine neue Ebene angelegt. Viele Ebenen führen jedoch zu vielen temporären Knoten, wobei die Zahl temporärer Knoten quadratisch zur Zahl der Ebenen abgeschätzt werden kann. Abbildung 6.19 gibt einen Eindruck von der enormen Zahl neuer Knoten, die bereits bei kleinen Graphen entstehen. Abbildung 6.20 verdeutlicht, wieso viele temporäre Knoten entstehen können. Da viele Knoten und viele Ebenen die Berechnungszeit der nächsten Schritte des VGL-Algorithmus negativ beeinflussen, soll die Anzahl der Ebenen möglichst gering gehalten werden.

Es reicht hier jedoch nicht, Knoten mit ähnlichen y_o -Koordinaten nachträglich zu einer gemeinsamen y_o -Koordinate (also einer gemeinsamen Ebene) zusammenzufassen, da dann die Mindestabstände zu darüber oder darunter liegenden Knoten nicht mehr erfüllt sind. Man könnte das Problem lösen, indem ein Algorithmus nach Zusammenfassen von Ebenen die Platzierung der von zusammengefassten Knoten abhängigen Knoten neu berechnet, um die Einhaltung des Mindestabstands zu sichern. Bei dieser Lösung wird jedoch u. U. die Rechenzeit stark erhöht, wie folgendes Gedankenexperiment zeigt: Man betrachte einen Graphen, bei dem nach Platzierung der Knoten die ersten beiden Ebenen zusammengefasst werden. Dadurch müssen alle darunter liegenden Ebenen neu platziert werden. Nun werden die zweite und dritte Ebene zu einer zusammengefasst, was wieder eine Neuplatzierung der darunter liegenden Knoten bewirkt, dann die dritte und vierte Ebene usw.

Dies motiviert die Entwicklung eines neuen Verfahrens.

6.3.3.2 Algorithmus RASTERGESTÜTZTE EBENENEINTEILUNG

Die Idee ist, während der Berechnung der y_o -Koordinaten jene Ebenen zusammenzufassen, die innerhalb eines vorgegebenen horizontalen Streifens beginnen. Jede Knotenplatzierung wird dabei nur einmal berechnet.

Wie im Algorithmus EBENENEINTEILUNG wird die Knotenhöhe um den Mindestabstand zwischen Knoten vergrößert. Für die folgende Erklärung sei auch auf Abbildung 6.21 verwiesen. Der Algorithmus RASTERGESTÜTZTE EBENENEINTEILUNG fasst Ebenen zusammen, die eine maximal um den *Rasterhöhe* r_h genannten Betrag größere y -Koordinate als die aktuelle Ebene haben. Dazu werden die Ebenen der Reihe nach von oben nach unten berechnet. Knoten, die auf Ebene i beginnen, sind entsprechend ihrer Höhe in einem Heap³⁷ H angeordnet, die Wurzel des Heaps enthält den niedrigsten Knoten. Die Wurzel v des Heaps liefert eine erste y -Koordinate für die Ebene $i + 1$, diese wird mit r_s bezeichnet ($r_s = y_o[v] + h[v]$). Die maximal mögliche y -Koordinate der Ebene $i + 1$ ist $r_e = r_s + r_h$.

Nun werden alle Knoten v , für die $y_o[v] + h[v] \leq r_e$ gilt, auf Ebene i platziert und aus dem Heap H entfernt. Als tatsächliche y -Koordinate der Ebene $i + 1$ wird schließlich $y = y_o[v] + h[v]$ des letzten aus dem Heap entfernten Knotens verwendet. Dieser Knoten ist zugleich der höchste noch komplett auf Ebene i gesetzte Knoten.

³⁷Ein *Heap* ist ein vollständiger binärer Baum, bei dem jeder Knoten einen kleineren Wert als jeder seiner Nachfolger (Söhne) hat, siehe z. B. [Sed92].

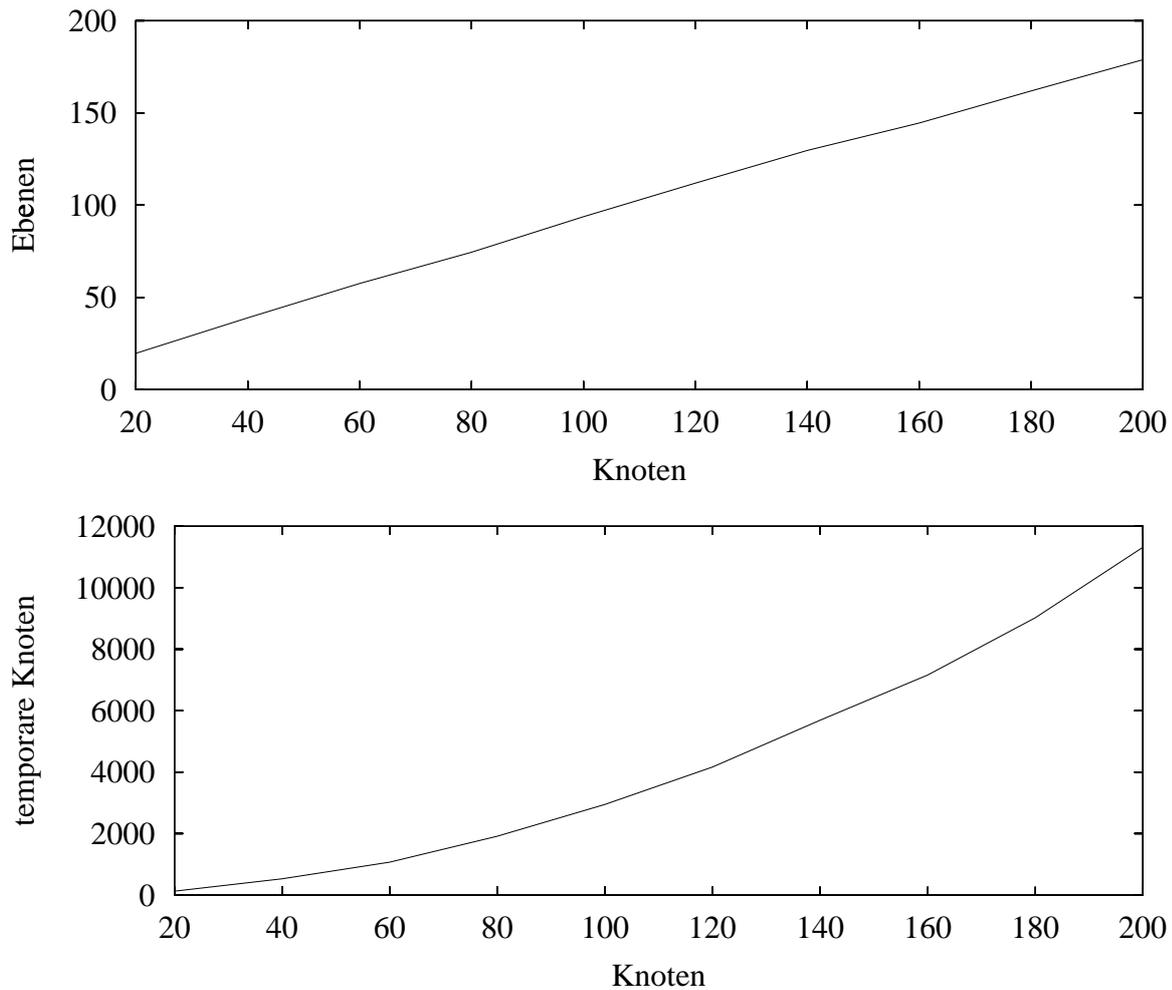


Abbildung 6.19: **Ebenen und temporäre Knoten.** (oben) Durchschnittliche Zahl der Ebenen und (unten) durchschnittliche Zahl der temporären Knoten für Graphen mit 20,40,...,200 Knoten und Dichte 0,15. Es wurden jeweils 100 kardinalitätsgleiche Graphen mit zufälligen Knotengrößen zwischen 1 und 100 betrachtet. Auffallend ist der sehr starke Anstieg der temporären Knoten.

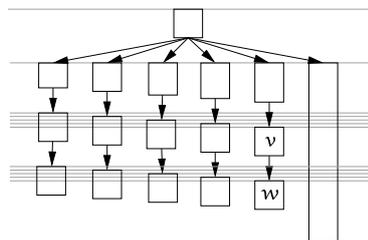


Abbildung 6.20: **Entstehung vieler Ebenen.** Ein Beispiel für die Entstehung vieler Ebenen, und damit vieler temporärer Knoten. Der große Knoten rechts wird in 11 temporäre Knoten zerlegt, die Kante zwischen den Knoten *v* und *w* immerhin in eine Kette von vier temporären Knoten.

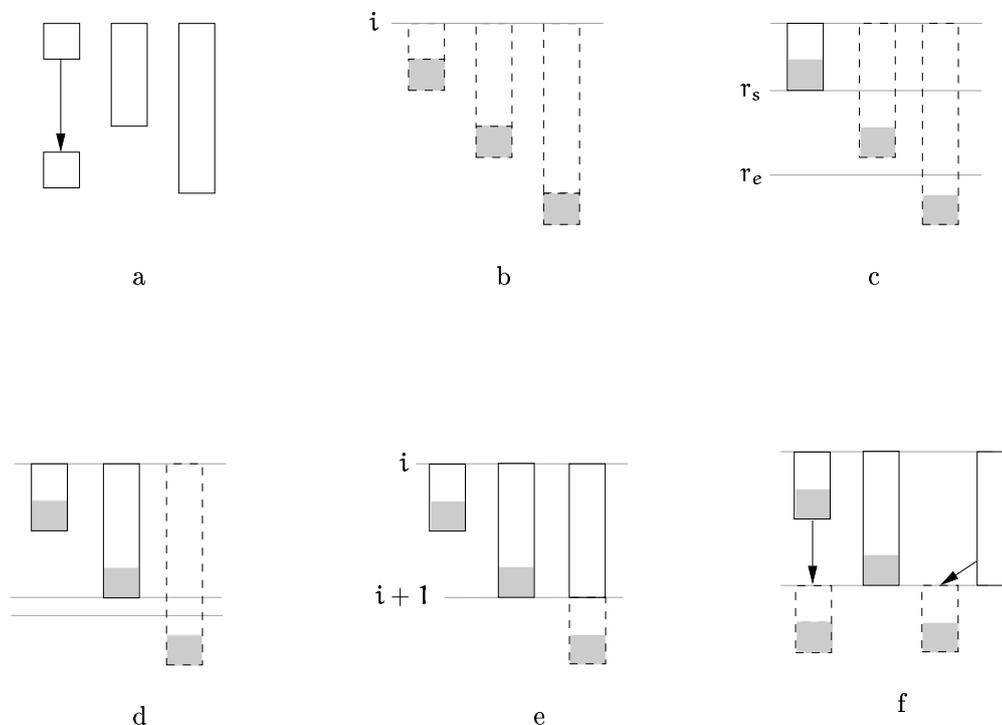


Abbildung 6.21: **Rastergestützte Ebeneneinteilung.** (a) Teilgraph, die drei oberen Knoten werden auf eine Ebene platziert (beispielsweise handelt es sich bei ihnen um Wurzelknoten).

(b) Ausgangssituation, kein Knoten der Ebene i ist platziert (noch nicht betrachtete Knoten sind gestrichelt dargestellt, der in der Höhe des Knotens berücksichtigte Mindestabstand ist grau eingezeichnet).

(c) Die Höhe des niedrigsten Knotens der Ebene i (die Wurzel des Heaps H) gibt die minimale y -Koordinate für die Ebene $i+1$ vor, dies entspricht r_s . Die maximale y -Koordinate für die Ebene $i+1$ wird bestimmt durch $r_e = r_s + r_h$ (r_h ist die Rasterhöhe). (d) Solange Knoten innerhalb des Bereichs zwischen r_s und r_e enden, werden sie komplett auf Ebene i gesetzt, der größte noch vollständig auf Ebene i gesetzte Knoten bestimmt die y -Koordinate für Ebene $i+1$. (e) Größere Knoten werden aufgespalten und ihr oberer Teil wird auf Ebene i platziert. Nun sind alle Knoten der Ebene i platziert, jetzt können die Knoten der Ebene $i+1$ betrachtet werden.

(f) Ausgangssituation für die Ebene $i+1$, kein Knoten der Ebene $i+1$ ist platziert. Nun wiederholen sich die Schritte wie bei Ebene i .

Alle anderen Knoten müssen auf diese und mindestens die nächste Ebene aufgespalten werden. Jeder solche Knoten v wird durch zwei Knoten v_1, v_2 und eine Kante (v_1, v_2) ersetzt, die ursprünglich zu v führenden Kanten führen nun zu v_1 , die von v ausgehenden Kanten gehen nun von v_2 aus. Der Knoten v_1 wird auf Ebene i platziert. Der Knoten v_2 wird später auf Ebene $i + 1$ gesetzt. Wichtig ist die Höhe des Knotens v_2 . Diese entspricht $h[v]$, verringert um den Abstand zwischen Ebene i und Ebene $i + 1$. Im Heap wird der Knoten v durch v_2 ersetzt. Da alle im Heap H verbliebenen Knoten in ihrer Höhe gleich verringert werden, ändert sich die Anordnung der Knoten im Heap nicht. Nun werden die weiteren auf Ebene $i + 1$ zu platzierenden Knoten in den Heap eingefügt und dann die Ebene $i + 1$ berechnet. Der komplette Algorithmus sieht wie folgt aus:

Algorithmus: RASTERGESTÜTZTE EBENENPARTITIONIERUNG.

Gegeben: Schwach zusammenhängender DAG $G = (V, E, M)$, Mindestabstand $a_{\min} > 0$, Rasterhöhe r_h .

Gesucht: Einfache l -Ebenenpartitionierung unter Einfügen temporärer Knoten für große Knoten und lange Kanten, Zusammenfassen nahe beieinander liegender Ebenen (wie oben beschrieben), y -Koordinaten für Knoten.

```

80 forall  $v \in V$ 
81    $h[v] := \text{Höhe}(v) + a_{\min}$ ;
82 Füge alle Knoten  $v \in V$  mit  $\text{Ein}(v) = 0$  in den Heap  $H$  ein;38
83  $y_{i+1} := 0$ ;  $\text{ebene} := 0$ ;  $V_b = \emptyset$ ;39
84
85 while  $H \neq \emptyset$ 
86   /* Finden und Platzieren aller Knoten, die vollständig auf Ebene  $i$  liegen */
87    $\text{ebene} := \text{ebene} + 1$ ;  $y_i := y_{i+1}$ ;
88    $v := \text{delmin}^{40}(H)$ ;  $\text{ebene}[v] := \text{ebene}$ ;  $y_o[v] := y_i$ ;  $V_b := V_b \cup \{v\}$ ;
89    $r_s := y_i + h[v]$ ;  $r_e := r_s + r_h$ ;41
90   while  $y_i + h[\text{min}^{42}(H)] \leq r_e$ 
91      $v := \text{delmin}(H)$ ;  $\text{ebene}[v] := \text{ebene}$ ;  $y_o[v] := y_i$ ;  $V_b := V_b \cup \{v\}$ ;
92
93    $y_{i+1} := y_i + h[v]$ ;43
94   /* Aufspalten großer Knoten, die auf Ebene  $i$  und Ebene  $i + 1$  liegen */
95   while Heap  $H$  nicht komplett durchlaufen
96      $v := \text{next}(H)$ ;
97     Ersetze Knoten  $v$  durch die Knoten  $v_1, v_2$  und die Kante  $(v_1, v_2)$ ;44
98      $V_b := (V_b \setminus \{v\}) \cup \{v_1\}$ ;  $V := (V \setminus \{v\}) \cup \{v_1, v_2\}$ ;
99      $\text{ebene}[v_1] := \text{ebene}$ ;  $y_o[v_1] := y_i$ ;  $h[v_2] := h[v] - (y_{i+1} - y_i)$ ;
100    Ersetze im Heap  $H$  den Knoten  $v$  durch den Knoten  $v_2$ ;

```

³⁸ H enthält alle Wurzelknoten von G , der Knoten mit kleinster Höhe bildet die Wurzel des Heaps.

³⁹ V_b wird verwendet, um bereits betrachtete Knoten (in der Menge V_b) von noch nicht betrachteten zu unterscheiden.

⁴⁰Die Funktion $\text{delmin}(H)$ liefert das Wurzelement des Heaps, entfernt dieses aus dem Heap und aktualisiert ihn.

⁴¹Das Raster für die nächsten Ebene startet mit der y -Koordinate r_s , es endet mit der y -Koordinate r_e .

⁴²Die Funktion $\text{min}(H)$ liefert das Wurzelement des Heaps.

⁴³Innerhalb des Rasters beginnt die nächste Ebene bei y_{i+1} , v ist der letzte aus dem Heap H entfernte Knoten.

⁴⁴Nach dieser Ersetzung sind alle eingehenden Kanten von v nun eingehende Kanten von v_1 und alle ausgehenden Kanten von v nun ausgehende Kanten von v_2 .

```

101  /* Bestimmung der weiteren Knoten der nächsten Ebene */
102  Füge alle Knoten  $v \in V, v \notin H, v \notin V_b$  mit  $\text{Vor}(v) \subseteq V_b$  in den Heap H ein;45
103
104  /* Aufspalten langer Kanten */
105  Wende Algorithmus EBENENEINFACHHEIT (siehe Seite 142) an;

```

LEMMA 6.28

Für die **while**-Schleife in den Zeile 85-102 gilt die Invariante:

$$\forall (u, v) \in E, u \in V_b, v \in V_b \quad \text{ebene}[v] > \text{ebene}[u] \quad (6.5)$$

Beweis: Die Invariante 6.5 gilt vor dem ersten Schleifendurchlauf, da $V_b = \emptyset$. Angenommen, die Invariante gilt zu Beginn eines beliebigen Schleifendurchlaufs. Sei V_b die Menge der bereits betrachteten Knoten vor dem Schleifendurchlauf und V'_b die Menge aller nach dem Schleifendurchlauf betrachteten Knoten. Es sind drei Arten von Kanten zu unterscheiden:

1. Kanten zwischen Knoten in V_b
2. Kanten zwischen Knoten, von denen ein Knoten in V_b , der zweite in $V'_b \setminus V_b$ liegt
3. Kanten zwischen Knoten in $V'_b \setminus V_b$

Zu 1.:

Knoten aus V_b werden in der Schleife keinen neuen Ebenen zugeordnet, für Kanten zwischen diesen Knoten gilt nach Ende des Schleifendurchlaufs weiterhin $\forall (u, v) \in E, u \in V_b, v \in V_b \quad \text{ebene}[v] > \text{ebene}[u]$.

Zu 2.:

Sei $V' = V'_b \setminus V_b$ und E' die Menge von Kanten zwischen Knoten aus V_b und Knoten aus V' . Da Knoten nur dann betrachtet werden, wenn bereits alle Vorgänger der Knoten in V_b sind, gibt es nur Kanten zwischen Knoten aus V_b zu Knoten aus V' , jedoch keine Kanten von Knoten aus V' zu Knoten aus V_b .

Alle Knoten $v \in V'$ werden auf Ebene i platziert. Da alle Vorgänger dieser Knoten bereits auf eine Ebene $< i$ platziert wurden bzw. da die Knoten im Falle von Wurzelknoten keine Vorgänger haben, gilt nach Ende des Schleifendurchlaufs $\forall (u, v) \in E', u \in V_b, v \in V' \quad \text{ebene}[v] > \text{ebene}[u]$.

Zu 3.:

Der Heap H enthält pro Schleifendurchlauf nur nicht adjazente Knoten, entweder die Wurzelknoten (Zeile 82) oder die Knoten der nächsten Ebene entsprechend Zeile 102. Deshalb können Kanten zwischen Knoten in $V'_b \setminus V_b$ nicht auftreten.

Damit gilt $\forall (u, v) \in E, u \in V'_b, v \in V'_b \quad \text{ebene}[v] > \text{ebene}[u]$ auch nach Ende eines Schleifendurchlaufs. \square

SATZ 6.29

Sei $G = (V, E, M)$ ein schwach zusammenhängender DAG. Der Algorithmus RASTERGESTÜTZTE EBENENPARTITIONIERUNG berechnet eine einfache l -Ebenenpartitionierung $G' = (V', E')$ mit $V' = V'_1 \cup \dots \cup V'_l$ und $E' = E'_1 \cup \dots \cup E'_{l-1}$ entsprechend einer größenrechten geometrischen topologischen Platzierung.

⁴⁵Der Heap H enthält danach alle Knoten, deren Vorgänger vollständig platziert und die selbst noch nicht betrachtet wurden.

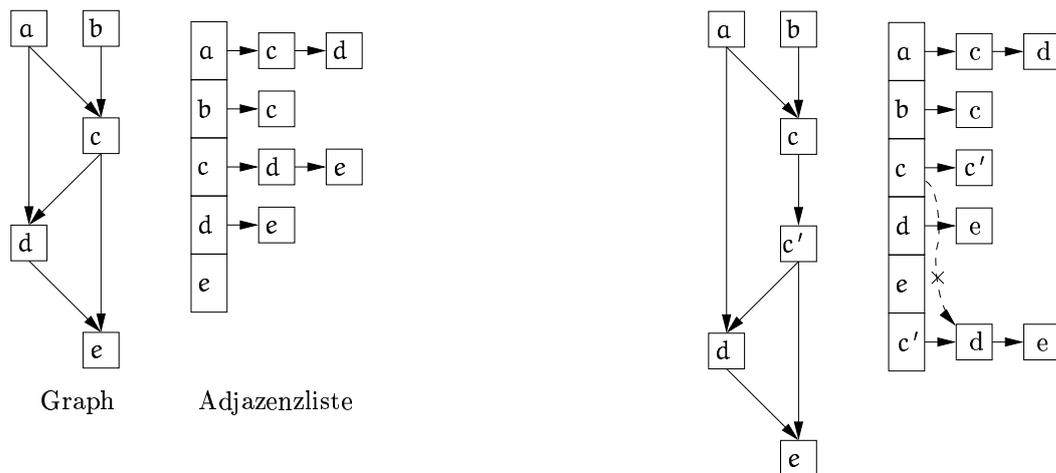


Abbildung 6.22: **Spalten von Knoten.** Adjazenzlisten sind eine Datenstruktur für Graphen, die das Spalten von Knoten unterstützt. (links) Ein Graph mit zugehöriger Adjazenzliste, wobei auf die Elemente der linken Spalte ein Zugriff in konstanter Zeit möglich ist [AHU83]. (rechts) Spalten des Knotens c in c und c' . Der Knoten c' wird der Datenstruktur⁴⁷ zugefügt, der Verweis auf die ursprünglich von c wegführenden Kanten zum Knoten c' umgehoben und die Kante (c, c') eingefügt.

Beweis: Es ist zu zeigen, dass nach Ende des Algorithmus gilt:

1. $\forall (u, v) \in E' \text{ ebene}[v] = \text{ebene}[u] + 1$
2. $\forall (u, v) \in E' \ y_o[v] \geq y_o[u] + \text{Höhe}(u) + \alpha_{\min}$

Zu 1.:

Wegen Lemma 6.28 und da jeder Knoten in V_b eingefügt wird,⁴⁶ gilt nach Ende des Algorithmus $\forall (u, v) \in E' \text{ ebene}[v] > \text{ebene}[u]$. Auf Grund der Bemerkungen zum Algorithmus EBENENEINFACHHEIT im Beweis zu Satz 6.26 gilt $\forall (u, v) \in E' \text{ ebene}[v] = \text{ebene}[u] + 1$ (wegen Zeile 105).

Zu 2.:

Es gilt $\forall u, v \in V' \text{ ebene}[v] > \text{ebene}[u] \Rightarrow y_o[v] > y_o[u]$, da die Knotenhöhen größer 0 sind, und damit bei jedem Schleifendurchlauf $y_{i+1} > y_i$ gilt (Zeile 93). Für einen beliebigen Knoten w auf Ebene i gilt $y_{i+1} - y_i \geq h[w]$. Da wegen Lemma 6.28 gilt: $\forall (u, v) \in E' \text{ ebene}[v] > \text{ebene}[u]$, so gilt auch: $\forall (u, v) \in E' \ y_o[v] \geq y_o[u] + h[u]$. Wegen $y_o[u] + h[u] = y_o[u] + \text{Höhe}(u) + \alpha_{\min}$ ergibt sich: $\forall (u, v) \in E' \ y_o[v] \geq y_o[u] + \text{Höhe}(u) + \alpha_{\min}$. \square

SATZ 6.30

Der Algorithmus RASTERGESTÜTZTE EBENENPARTITIONIERUNG benötigt $O(|V| * |E|)$.

⁴⁶Falls ein Knoten aus V_b entfernt wird, so werden die ihn ersetzenden Knoten in V_b eingefügt.

⁴⁷Dabei handelt es sich beispielsweise um ein Feld, das zu Beginn des Algorithmus genügend viele leere Elemente enthält. Die Größe des Feldes ist begrenzt, da jeder Knoten während der Ebenenpartitionierung höchstens in $|V|$ neue Knoten aufgespalten wird.

Beweis: Die **forall**-Schleife in den Zeilen 80-81 benötigt $O(|V|)$, der Aufbau des Heaps in Zeile 82 $O(|V| \log |V|)$. Für die äußere **while**-Schleife in den Zeilen 85-102 gilt: Es können $|V|$ Ebenen entstehen, die Schleife kann also $|V|$ mal durchlaufen werden.

Pro Ebene können bis zu $|V|$ Knoten vorhanden sein bzw. durch das Aufspalten großer Knoten entstehen. Die innere **while**-Schleife in den Zeilen 90-91 benötigt $O(|V|)$. Die innere **while**-Schleife in den Zeilen 95-100 benötigt ebenfalls nur $O(|V|)$, das Aufspalten eines Knotens v in v_1 und v_2 kann dabei in konstanter Zeit erfolgen, wie Abbildung 6.22 demonstriert. Das Bestimmen der zusätzlich in den Heap einzufügenden Knoten in Zeile 102 kann in $O(|E| + |V|)$ mit einer abgewandelten topologischen Sortierung erfolgen.⁴⁸

Mit $|E| > |V| - 1$ wegen des schwachen Zusammenhangs des Graphen benötigt damit die äußere **while**-Schleife in den Zeilen 85-102 insgesamt $O(|V| * |E|)$. Dabei wurde jedoch das Einfügen in den Heap bisher nicht betrachtet. Das Einfügen eines Elements in den Heap benötigt $O(\log |V|)$. Es werden stets nur originale Knoten in den Heap eingefügt, da die aufgespaltenen Knoten bereits in der richtigen Anordnung im Heap enthalten sind. Im gesamten Algorithmus müssen $|V|$ Einfügeoperationen für die originalen Knoten ausgeführt werden, die sich beliebig auf die Schleifendurchläufe verteilen. Es gilt für die insgesamt im Algorithmus für Einfügen in den Heap nötige Zeit: $O(|V| \log |V|) \leq O(|V| * |E|)$

Nach Satz 6.27 benötigt der Algorithmus EBENENEINFACHHEIT $O(|V| * |E|)$. Damit gilt für die Komplexität des Algorithmus RASTERGESTÜTZTE EBENENPARTITIONIERUNG $O(|V| * |E|)$. \square

6.3.3.3 Auswirkungen der Rasterhöhe

Die Wahl der Rasterhöhe r_h bietet eine einfache Möglichkeit, verschiedene Typen von Ebenenpartitionierungen zu erhalten:

1. Wird die Rasterhöhe $r_h = 0$ gesetzt, so liefert der Algorithmus RASTERGESTÜTZTE EBENENPARTITIONIERUNG eine *größenechte Platzierung* (siehe Abb. 6.23(a)).
2. Bei $r_h = \max\{\text{Höhe}(v) \mid v \in V\} + a_{\min}$ erfolgt eine *größenebenenmaximale Platzierung* (siehe Abb. 6.23(c)).
3. Eine *größenmaximale Platzierung* kann man erreichen, wenn $r_h = \max\{\text{Höhe}(v) \mid v \in V\} + a_{\min}$ gilt und Zeile 93 geändert wird in $y_{i+1} := y_i + r_h$.

Der Parameter Rasterhöhe beeinflusst damit auch die Rechenzeit der folgenden Schritte des VGL-Verfahrens. Ein kleiner Wert für r_h bedeutet im Allgemeinen viele Ebenen, viele temporäre Knoten und eine zeitaufwendigere Berechnung in den nächsten Schritten; ein großer Wert wenige Ebenen, wenige temporäre Knoten und eine schnelle Berechnung. In den Abbildungen 6.24-6.26 sind die Beziehungen zwischen Rasterhöhe, Höhe der Zeichnung, Zahl der Ebenen und Zahl der temporären Knoten dargestellt. Dabei wurden lichte Graphen betrachtet, da diese bei ebenenweisen Zeichenverfahren überwiegen.

⁴⁸Es genügt hier sogar, sich die Knoten der Ebene i zu merken und nur deren Nachfolger zu betrachten.

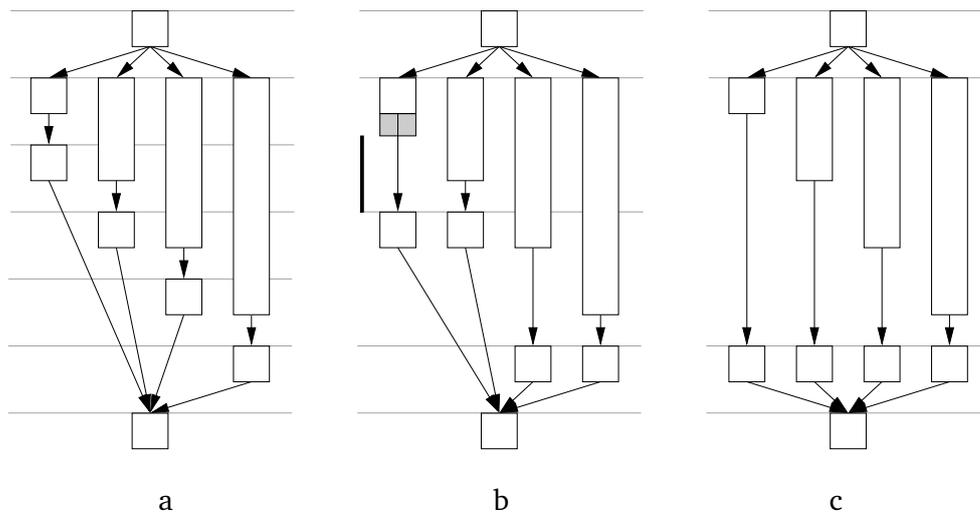


Abbildung 6.23: **Auswirkungen der Rasterhöhe (Teil 1)**. Platzierung der Knoten eines Graphen mit verschiedenen Rasterhöhen r_h . (a) *Größenechte Platzierung* ($r_h = 0$), (b) beliebiger Wert für r_h , z. B. $r_h =$ Höhe des schwarzen Knotens neben dem Graphen (zusätzlich ist auch der Mindestabstand für einen Knoten grau dargestellt), (c) *größenebenenmaximale Platzierung* ($r_h =$ Höhe des größten Knotens). Hier wird auch die Verringerung der Zahl der Ebenen deutlich, die mit Vergrößerung der Rasterhöhe oft einhergeht. Für eine größenebenenmaximale Platzierung muss der Algorithmus, wie in Abschnitt 6.3.3.3 beschrieben, geändert werden. Die Zeichnung wäre ähnlich zu (c), alle Ebenenabstände wären uniform.

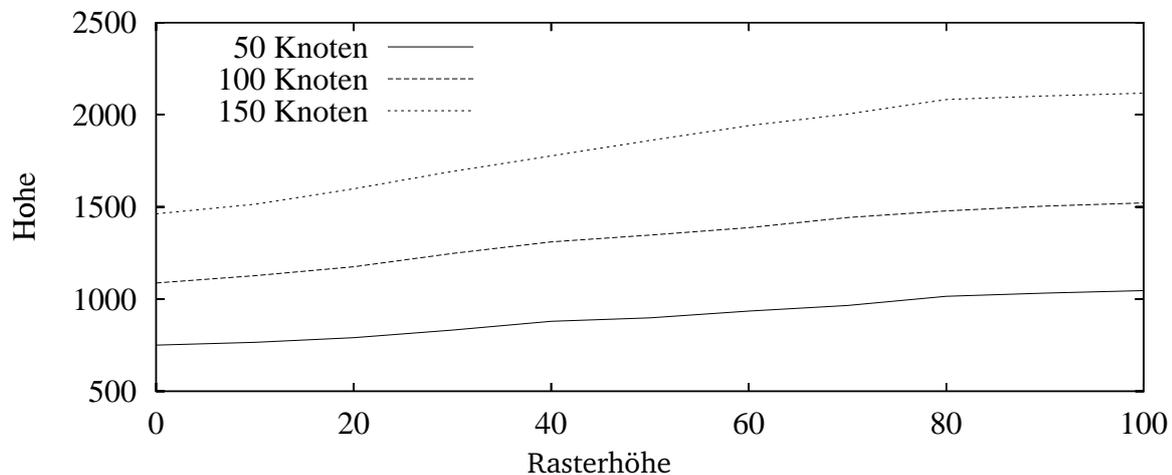


Abbildung 6.24: **Auswirkungen der Rasterhöhe (Teil 2)**. Durchschnittliche Höhe der Zeichnung in Abhängigkeit der Rasterhöhe, Werte für 100 kardinalitätsgleiche Graphen mit 50,100,150 Knoten, Dichte 0,15. Wie bei allen Experimenten für die Ebenenpartitionierung hat jeder Knoten eine zufällige Knotenhöhe zwischen 1 und 100. Mit steigender Rasterhöhe steigt auch die Höhe der Zeichnung.

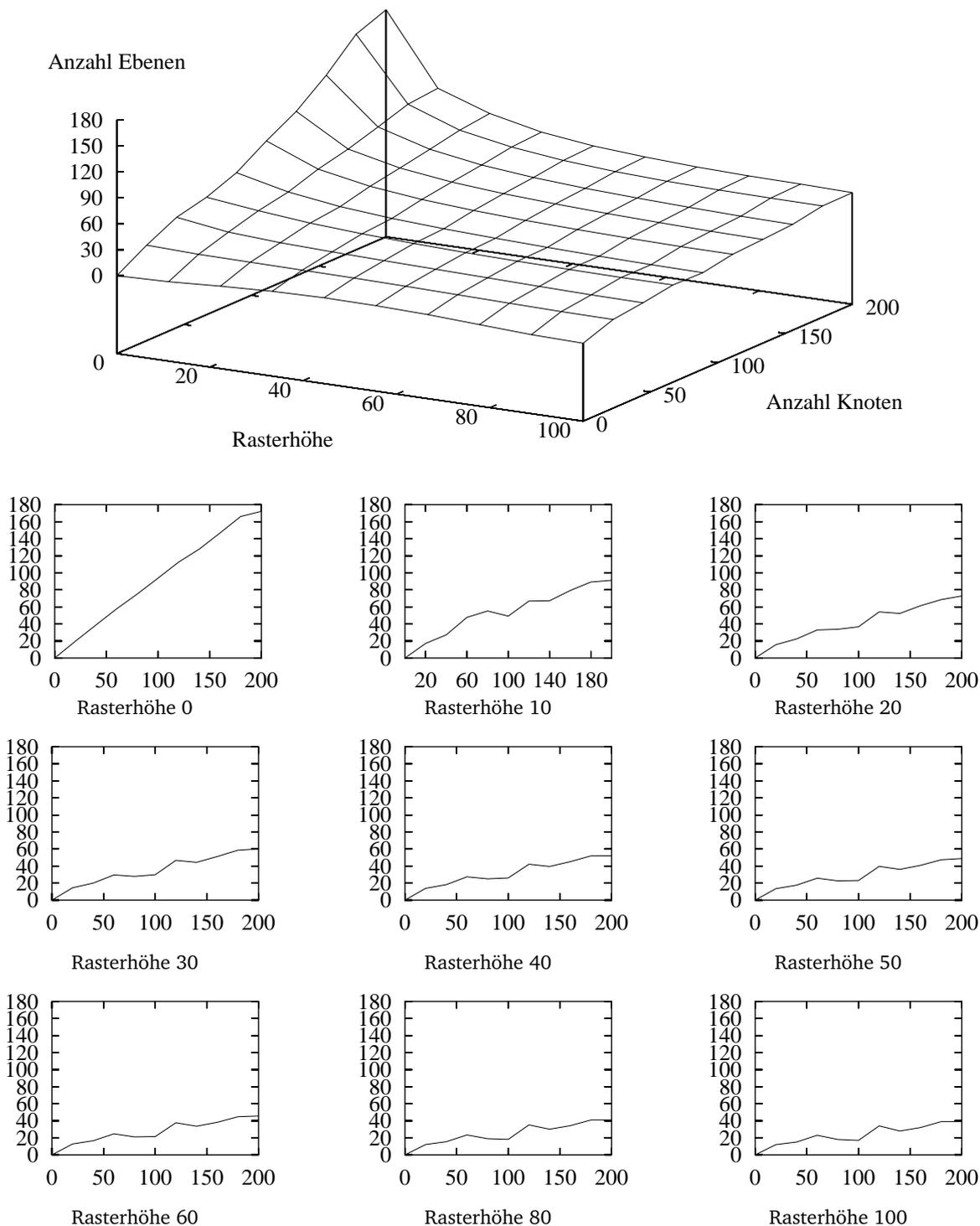


Abbildung 6.25: **Auswirkungen der Rasterhöhe (Teil 3)**. Graphen mit 20,40,...,200 Knoten, Dichte 0,15. Im Experiment wurden jeweils 100 isomorphe Graphen mit zufälligen Knotenhöhen zwischen 1 und 100 bei steigendem Raster (0,10,...,100) untersucht. Auffällig ist der starke Anstieg der Zahl der Ebenen bei sehr kleinen Werte der Rasterhöhe gegenüber einem moderaten Anstieg bei größeren Werten der Rasterhöhe. Die Rasterhöhe $r_h = 0$ entspricht einer grössenechten, die Rasterhöhe $r_h = 100$ einer (traditionellen) grössenebenenmaximalen Platzierung.

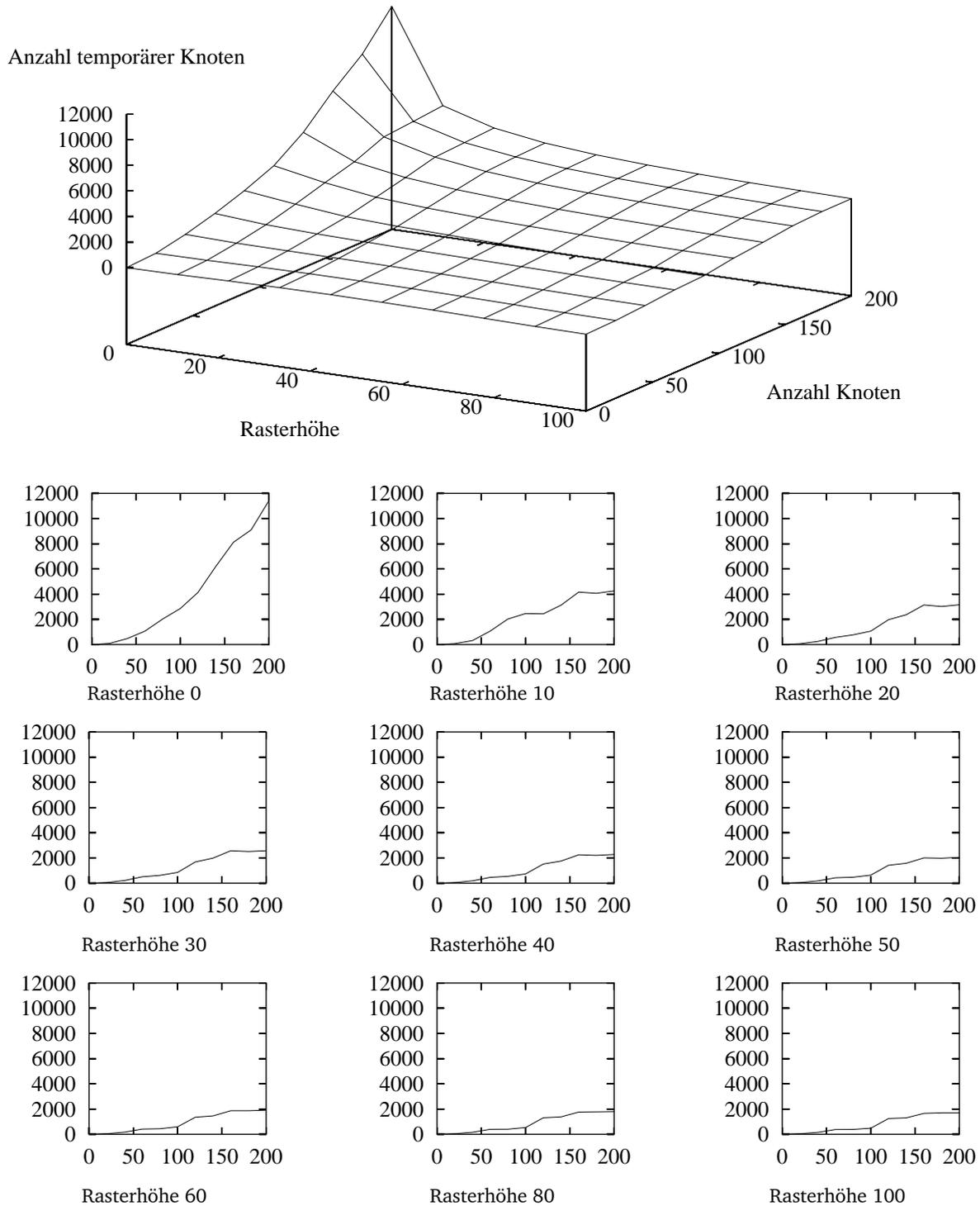


Abbildung 6.26: **Auswirkungen der Rasterhöhe (Teil 4).** Untersuchung der Zahl der temporären Knoten. Es wurden die gleichen Graphen wie in Abbildung 6.25 untersucht. Hier fällt der starke Anstieg der Zahl temporärer Knoten bei sehr kleinen Werte der Rasterhöhe auf.

6.3.4 Weitere Verfahren zur Ebenenpartitionierung

Die Partitionierung der Knoten in Ebenen ist eine Idee, die bereits durch Sugiyama et al. [STT81] formuliert wurde. Sie erfolgt, um einerseits die Kreuzungsreduzierung von einem globalen Problem auf ein einfacheres (jedoch dabei immer noch NP-vollständiges) Problem der Reduzierung von Kreuzungen zwischen benachbarten Ebenen zu vereinfachen. Andererseits bietet die Partitionierung in Ebenen eine Möglichkeit, die Breite der Zeichnung zu beeinflussen, da diese wesentlich durch die Zahl der Knoten pro Ebene und deren Breite beeinflusst wird.

Bei den traditionellen Verfahren der Ebeneneinteilung spielt die Knotenhöhe keine Rolle. Diese Verfahren sind so gestaltet, dass bestimmte Kriterien berücksichtigt werden. So können Darstellungen mit minimaler Höhe, also wenigen Ebenen, gewünscht sein. Ein anderes Ziel sind Zeichnungen, bei denen auf keiner Ebene mehr als k Knoten platziert werden. Auch eine geringe Zahl der für lange Kanten einzufügenden temporären Knoten ist ein mögliches Kriterium für die Ebeneneinteilung. Im Folgenden werden solche Verfahren zusammengefasst. Auch hier werden schwach zusammenhängende DAGs vorausgesetzt. Verfahren zur Ebeneneinteilung in herkömmlichen ebenenweisen Zeichenverfahren sind:

1. Ebeneneinteilung auf Grund einer topologischen Sortierung [Kah62] entsprechend dem längsten Pfad zu jedem Knoten [ES90]. Dies führt zu Zeichnungen mit minimaler Zahl von Ebenen. Diese Sortierungen können in $O(|E|)$ berechnet werden.
2. Ebeneneinteilung mittels *Coffman-Graham-Verfahren* [CG72, ES90]. Dies führt zu Zeichnungen mit maximal k echten Knoten auf jeder Ebene, temporäre Knoten für lange Kanten werden nicht berücksichtigt. Die Zeichnung hat höchstens $(2 - \frac{2}{k}) * h_{\min}$ Ebenen, wobei h_{\min} die minimale Zahl der Ebenen ist [LS77].
3. Ebeneneinteilung mittels *Network Simplex Algorithmus* [GKNV93]. Dies führt zu Zeichnungen mit minimaler Zahl temporärer Knoten für lange Kanten. Die Komplexität dieses Verfahrens ist unbekannt.⁴⁹

Andere Ebeneneinteilungen sind schwieriger zu berechnen. So ist die Bestimmung von Ebeneneinteilungen mit minimaler Zahl temporärer Knoten für lange Kanten und minimaler Zahl von Ebenen NP-vollständig [DETT99].

Die traditionellen Ebeneneinteilungen bieten Möglichkeiten zur Gestaltung einer ebenenweisen Zeichnung. Diese Ebeneneinteilungen, und damit die Gestaltungsmöglichkeiten, lassen sich jedoch nicht direkt im hier vorgestellten Verfahren der Ebenenpartitionierung verwenden.

Abschnitt 7.3.5 behandelt eine Erweiterung, mit der traditionelle Verfahren der Ebeneneinteilung auch im VGL-Verfahren verwendet werden können. Dabei werden mittels o - u -Lagebeziehungen vorgegebene Ebeneneinteilung bei der Ebenenpartitionierung berücksichtigt.

⁴⁹Die Autoren schreiben dazu: „Although its time complexity has not been proven polynomial, in practice it takes few iterations and runs quickly.“ [GKNV93]

6.4 Kreuzungsreduzierung

6.4.1 Überblick

Abschnitt 6.4 behandelt die dritte Phase des VGL-Verfahrens: Die *Kreuzungsreduzierung*. Im Folgenden wird eine Kurzfassung des Abschnitts gegeben.

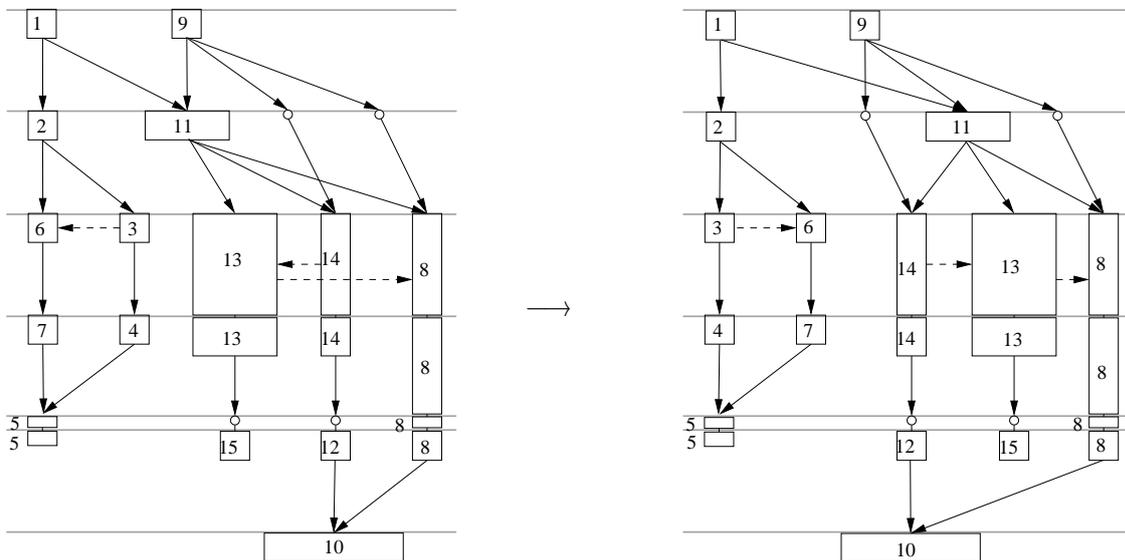
Ziel der Kreuzungsreduzierung:

einfache l-Ebenenpartitionierung

→

kreuzungsreduzierte einfache l-Ebenenpartitionierung, d. h.

- wenige Kantenkreuzungen
- Knotenreihenfolge entsprechend der l-r-Lagebeziehungen



Aufbau von Abschnitt 6.4:

- 6.4.2: Die Problemstellung wird eingeführt. Dabei wird besonders die Behandlung großer, über mehrere Ebenen gehender Knoten betrachtet.
- 6.4.3: Das Kreuzungsreduzierungsproblem wird theoretisch untersucht.
- 6.4.4: Eine Heuristik zur Verringerung der Zahl der Kantenkreuzungen unter Berücksichtigung von l-r-Lagebeziehungen wird vorgestellt.
- 6.4.5: Es wird ein Überblick über traditionelle Verfahren zur Kreuzungsreduzierung gegeben und das hier entwickelte Verfahren mit anderen Lösungen verglichen.

6.4.2 Einführung

Dieser Teil des VGL-Verfahrens beschäftigt sich mit der Anordnung der Knoten einer einfachen l -Ebenenpartitionierung innerhalb der Ebenen. Ziel ist eine Minimierung der Zahl der Kantenkreuzungen und die Erfüllung möglichst vieler und hoch gewichteter l - r -Lagebeziehungen.

6.4.2.1 Gründe für ein weiteres Verfahren zur Kreuzungsreduzierung

Der Schritt der Kreuzungsreduzierung ist der am intensivsten untersuchte Bereich des *Sugiyama*-Algorithmus. Dafür wurden verschiedene Verfahren entwickelt, die im Abschnitt 6.4.5 zusammengefasst sind. Es stellt sich die Frage, ob ein weiterer Algorithmus zur Kreuzungsreduzierung notwendig ist. Dies kann aus drei Gründen bejaht werden:

1. Keines der bekannten Verfahren berücksichtigt l - r -Lagebeziehungen zwischen Knoten. Für einige Algorithmen existieren zwar Erweiterungen, diese benötigen jedoch zusätzliche Rechenzeit und liefern schlechte Ergebnisse.
2. Es ist sicherzustellen, dass sich Bereiche zweier großer Knoten nie kreuzen, d. h. dass sie auf allen Ebenen in der gleichen Reihenfolge angeordnet sind (siehe auch Abb. 6.27). Auch dies wird durch bekannte Verfahren nicht geleistet.
3. Es ist sinnvoll, Kreuzungen zwischen Kanten unterschiedlich zu gewichten. Dadurch können beispielsweise Kreuzungen zwischen einer normalen Kante und einer Kante, die Teile eines großen Knotens verbindet, stärker vermieden werden als Kantenkreuzungen zwischen zwei normalen Kanten. Im Ergebnis werden damit Überschneidungen zwischen Kanten und großen Knoten vermieden. Auch dies wird durch traditionelle Verfahren nicht unterstützt.

6.4.2.2 Globale Kreuzungsreduzierung

Das Problem der Kreuzungsreduzierung ist sowohl für allgemeine Zeichnungen von Graphen wie auch für ebenenweise Zeichnungen NP-vollständig [GJ83, EW94]. Es werden oft Heuristiken zur Berechnung kreuzungsreduzierter Knotenanordnungen verwendet.

In der Praxis wird die Kreuzungsreduzierung einer einfachen l -Ebenenpartitionierung $G = (L_1 \cup \dots \cup L_l, E_1 \cup \dots \cup E_{l-1})$ auf eine Folge von Kreuzungsreduzierungen von 2-Ebenenpartitionierungen aufgeteilt. Der dabei verwendete Algorithmus sieht wie folgt aus:

Algorithmus: Globale Kreuzungsreduzierung [ES90, San96b].

Gegeben: Einfache l -Ebenenpartitionierung $G = (L_1 \cup \dots \cup L_l, E_1 \cup \dots \cup E_{l-1})$, Abbruchkriterien: I -Zahl der Schleifendurchläufe, K -Zahl der Kreuzungen.

Gesucht: Kreuzungsreduzierte einfache l -Ebenenpartitionierung.

```

106  $k :=$  aktuelle Zahl der Kreuzungen;  $i := 0$ ;
107 while ( $k > K$ ) and ( $i < I$ )
108   for ( $j := 1$ ;  $j < l$ ;  $j := j + 1$ )
109     Sortiere Knoten von  $L_{j+1}$  auf Grund der Anordnung der Knoten von  $L_j$ ;
110   for ( $j := l$ ;  $j > 1$ ;  $j := j - 1$ )
111     Sortiere Knoten von  $L_{j-1}$  auf Grund der Anordnung der Knoten von  $L_j$ ;
112    $k :=$  aktuelle Zahl der Kreuzungen;  $i := i + 1$ ;

```

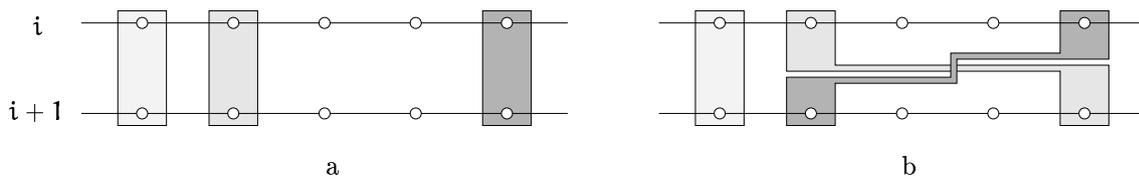


Abbildung 6.27: **Kreuzungsreduzierung und große Knoten.** (a) Zulässige 2-Ebenenpartitionierung und (b) nicht zulässige 2-Ebenenpartitionierung, bei der die Teile zweier großer Knoten in unterschiedlichen Reihenfolgen vorliegen. Die Teile der drei großen Knoten sind in der Darstellung verschieden grau eingezeichnet.

Dieses Verfahren wird als Grundlage für die Kreuzungsreduzierung verwendet. Im Folgenden werden nur noch die darin verwendeten Kreuzungsreduzierungen für 2-Ebenenpartitionierungen $G = (L_i \cup L_{i+1}, E_i)$ untersucht. Dabei wird sich auf die Berechnung der Sortierung der Knoten von Ebene L_{i+1} bei fester Anordnung der Knoten in Ebene L_i beschränkt (siehe Zeile 109), die entgegengesetzte Richtung (Zeile 111) verläuft analog.

6.4.2.3 Bemerkungen zu l-Ebenenpartitionierungen

Im Folgenden sind alle l-Ebenenpartitionierungen einfach. Als *große Knoten* werden solche Knoten bezeichnet, die bei der Ebenenpartitionierung aufgespalten werden. Um die Menge der für einen großen Knoten eingefügten Knoten zu beschreiben, wird die Abbildung R eingeführt.

DEFINITION 6.31 (REPRÄSENTANTEN GROSSER KNOTEN)

Sei $G = (V, E)$ ein Graph und $G' = (L_1 \cup \dots \cup L_l, E_1 \cup \dots \cup E_{l-1})$ eine aus G entstandene l-Ebenenpartitionierung. Die Abbildung $R : V \rightarrow \mathcal{P}(L_1 \cup \dots \cup L_{l-1})$ liefert für jeden großen Knoten $v \in V$ die Menge seiner Repräsentanten $\{v_1, \dots, v_n\}$.

Der Ausgangsgraph G , aus dem die großen Knoten stammen, wird auch weggelassen, falls er aus dem Kontext hervorgeht.

l-r-Lagebeziehungen, an denen große Knoten beteiligt sind, werden auf deren Repräsentanten erweitert. Sei z. B. $(u, v, l-r, g)$ eine l-r-Lagebeziehung und v ein großer Knoten, so werden für alle Repräsentanten $v_i \in R(v)$ Lagebeziehungen $(u, v_i, l-r, g)$ eingefügt.

Nun sollen Kreuzungsreduzierungen für 2-Ebenenpartitionierungen untersucht werden. Zuerst werden erfüllte von nicht erfüllten l-r-Lagebeziehungen unterschieden. Ob l-r-Lagebeziehungen erfüllt werden, lässt sich in einer Zeichnung anhand der x- und y-Koordinaten der Knoten bestimmen. Eine l-r-Lagebeziehung $(u, v, l-r, g)$ ist in der Zeichnung erfüllt, wenn u links von v liegt, falls beide Knoten eine gemeinsame y-Koordinate besitzen, also wenn gilt:

$$(\exists y (y_o(u) \leq y \leq y_u(u)) \wedge (y_o(v) \leq y \leq y_u(v))) \Rightarrow (x_r(u) < x_l(v)).$$

Im Moment haben die Knoten auf Grund der Ebeneneinteilung schon y-Koordinaten, aber noch keine x-Koordinaten. Es ist also eine andere Charakterisierung von erfüllten l-r-Lagebeziehungen gesucht, die unabhängig von x-Koordinaten ist.

Für die Knoten einer Ebene gilt später: $\forall u, v \in L_i \quad L_i(u) < L_i(v) \Leftrightarrow x_r(u) < x_l(v)$. Dieser Zusammenhang kann hier bereits ausgenutzt werden. l - r -Lagebeziehungen können durch die Anordnung der Knoten in den Ebenen repräsentiert werden. Da nur 2-Ebenenpartitionierungen $G = (L_1 \cup L_2, E)$ betrachtet werden, bei denen L_1 fest ist, lässt sich die Behandlung von l - r -Lagebeziehungen weiter vereinfachen. Es sind dabei nur l - r -Lagebeziehungen zwischen Knoten von L_2 bedeutsam.

DEFINITION 6.32 (ERFÜLLTE l - r -LAGEBEZIEHUNG IN 2-EBENENPARTITIONIERUNG)

Sei $G = (L_1 \cup L_2, E)$ eine 2-Ebenenpartitionierung und (u, v, l, r, g) eine l - r -Lagebeziehung mit $u, v \in L_2$. Die l - r -Lagebeziehung heißt erfüllt, falls gilt: $L_2(u) < L_2(v)$.

Wie bereits in Abbildung 6.27 dargestellt, sind bei Berücksichtigung großer Knoten nicht mehr alle Permutationen der Knoten in den Ebenen zulässig, da Teile zweier großer Knoten sich gegenseitig nicht kreuzen dürfen. Zulässige 2-Ebenenpartitionierungen sind wie folgt definiert:

DEFINITION 6.33 (ZULÄSSIGE 2-EBENENPARTITIONIERUNG)

Sei $G = (L_1 \cup L_2, E)$ eine 2-Ebenenpartitionierung. G zulässig, falls für $u_1 \in L_1, u_2 \in L_2$ mit $u_1, u_2 \in R(u)$ ⁵⁰ und $v_1 \in L_1, v_2 \in L_2$ mit $v_1, v_2 \in R(v)$ gilt:

$$L(u_1) < L(v_1) \Leftrightarrow L(u_2) < L(v_2).$$

Die Kreuzungsreduzierung lässt sich nun als folgende Aufgabe beschreiben: Eine 2-Ebenenpartitionierung G soll in eine zulässige 2-Ebenenpartitionierung G' umgewandelt werden, so dass in G' die l - r -Lagebeziehungen erfüllt werden und zugleich G' möglichst wenige Kantenkreuzungen enthält.

Analog zu o - u -Lagebeziehungen (siehe Abschnitt 6.2) existiert auch hier das Problem, dass widersprüchliche l - r -Lagebeziehungen vorhanden sein können, so dass nicht alle l - r -Lagebeziehungen erfüllbar sind. Die Auswahl der nicht erfüllten l - r -Lagebeziehungen wird durch deren Gewicht unterstützt. Wie bei o - u -Lagebeziehungen könnte man wieder zwei verschiedene Ansätze betrachten: Einen globalen Ansatz, der die Summe der Gewichte nicht erfüllter l - r -Lagebeziehungen minimiert, und einen lokalen, der höher gewichtete Lagebeziehungen so lange wie möglich erfüllt.

6.4.2.4 Lokaler Ansatz zur Kreuzungsreduzierung

Hier wird nur der lokale Ansatz weiter verfolgt, und zwar aus folgendem Grund: Wenn die Anordnung der Knoten in einer der beiden Ebenen fest ist, kann die Zulässigkeit einer 2-Ebenenpartitionierung $G = (L_1 \cup L_2, E)$ durch l - r -Lagebeziehungen repräsentiert werden. Werden diese Lagebeziehungen höher als alle anderen l - r -Lagebeziehungen gewichtet, so werden sie durch den lokalen Ansatz immer erfüllt, es wird also stets eine zulässige 2-Ebenenpartitionierung berechnet. Dies soll im Folgenden genauer untersucht werden.

Zuerst werden 2-Ebenenpartitionierungen ohne anwendergegebene l - r -Lagebeziehungen betrachtet. Sei $G = (L_1 \cup L_2, E)$ eine 2-Ebenenpartitionierung und wie vorher die Anordnung von L_1 fest. Für jedes Paar u, v großer Knoten mit Repräsentanten $u_1, u_2 \in R(u), u_1 \in L_1, u_2 \in L_2$ und $v_1, v_2 \in R(v), v_1 \in L_1, v_2 \in L_2$ wird eine l - r -Lagebeziehung (u_2, v_2, l, r, g) eingefügt, falls gilt: $L_1(u_1) < L_1(v_1)$. Die Lagebeziehung drückt aus, dass auch auf Ebene L_2 der Repräsentant von u links des Repräsentanten von v liegen soll.

⁵⁰Die beiden Knoten u_1 und u_2 wurden für den großen Knoten u eingefügt, sie sind Repräsentanten dieses Knotens.

Eine solche 2-Ebenenpartitionierung wird *2-Ebenenpartitionierung mit l-r-Lagebeziehungen für große Knoten* oder kurz *2-LRG-Ebenenpartitionierung* genannt.

DEFINITION 6.34 (2-LRG-EBENENPARTITIONIERUNG)

Sei $G = (L_1 \cup L_2, E)$ eine 2-Ebenenpartitionierung. Die 2-LRG-Ebenenpartitionierung $G' = (L_1 \cup L_2, E, LB)$ entsteht aus G , indem für jedes Paar u, v großer Knoten mit $u_1, u_2 \in R(u)$, $u_1 \in L_1$, $u_2 \in L_2$ und $v_1, v_2 \in R(v)$, $v_1 \in L_1$, $v_2 \in L_2$ sowie $L_1(u_1) < L_1(v_1)$ eine l-r-Lagebeziehung $(u_2, v_2, l-r, g)$ in LB eingefügt wird.

Eine 2-LRG-Ebenenpartitionierung enthält nur l-r-Lagebeziehungen für große Knoten. Für Paare u, v großer Knoten mit Repräsentanten wie oben gilt: $L_1(u_1) < L_1(v_1) \Leftrightarrow (u_2, v_2, l-r, g) \in LB$. Für den Wert von g muss dabei wie bei allen Lagebeziehungen gelten: $g > 1$.

SATZ 6.35

Eine 2-Ebenenpartitionierung $G = (L_1 \cup L_2, E)$ ist genau dann zulässig, wenn die zugehörige 2-LRG-Ebenenpartitionierung $G' = (L_1 \cup L_2, E, LB)$ alle l-r-Lagebeziehungen erfüllt.

Beweis: Es sind zwei Richtungen zu zeigen:

1. Wenn die 2-Ebenenpartitionierung $G = (L_1, L_2, E)$ zulässig ist, so werden in der 2-LRG-Ebenenpartitionierung $G' = (L_1 \cup L_2, E, LB)$ alle l-r-Lagebeziehungen erfüllt.
2. Wenn in der 2-LRG-Ebenenpartitionierung $G' = (L_1 \cup L_2, E, LB)$ alle l-r-Lagebeziehungen erfüllt werden, so ist die 2-Ebenenpartitionierung $G = (L_1 \cup L_2, E)$ zulässig.

Zu 1:

Sei $G = (L_1 \cup L_2, E)$ zulässig und seien u und v zwei beliebige große Knoten mit $u_1, u_2 \in R(u)$, $u_1 \in L_1$, $u_2 \in L_2$ und $v_1, v_2 \in R(v)$, $v_1 \in L_1$, $v_2 \in L_2$. Dann gilt $L_1(u_1) < L_1(v_1) \Leftrightarrow L_2(u_2) < L_2(v_2)$ nach Definition 6.33 und $L_1(u_1) < L_1(v_1) \Leftrightarrow (u_2, v_2, l-r, g) \in LB$ nach Definition 6.34. Für eine l-r-Lagebeziehung folgt damit $(u_2, v_2, l-r, g) \in LB \Leftrightarrow L_2(u_2) < L_2(v_2)$, also die l-r-Lagebeziehung wird erfüllt.

Da dies für beliebige Paare großer Knoten gilt, werden alle l-r-Lagebeziehungen erfüllt.

Zu 2:

Erfülle $G' = (L_1 \cup L_2, E, LB)$ alle l-r-Lagebeziehungen und seien u und v zwei beliebige große Knoten mit $u_1, u_2 \in R(u)$, $u_1 \in L_1$, $u_2 \in L_2$ und $v_1, v_2 \in R(v)$, $v_1 \in L_1$, $v_2 \in L_2$. Dann gilt $(u_2, v_2, l-r, g) \in LB \Rightarrow L_2(u_2) < L_2(v_2)$, da alle l-r-Lagebeziehungen erfüllt werden. Die umgekehrte Richtung $L_2(u_2) < L_2(v_2) \Rightarrow (u_2, v_2, l-r, g) \in LB$ gilt, da für die Repräsentanten u_2, v_2 eine der l-r-Lagebeziehungen $(u_2, v_2, l-r, g)$ oder $(v_2, u_2, l-r, g)$ in LB eingefügt wurde und alle l-r-Lagebeziehungen erfüllt werden. Insgesamt gilt damit: $L_2(u_2) < L_2(v_2) \Leftrightarrow (u_2, v_2, l-r, g) \in LB$.

Nach Definition 6.34 gilt $(u_2, v_2, l-r, g) \in LB \Leftrightarrow L_1(u_1) < L_1(v_1)$. Wie gewünscht folgt nun $L_2(u_2) < L_2(v_2) \Leftrightarrow L_1(u_1) < L_1(v_1)$.

Da dies für alle l-r-Lagebeziehungen gilt, ist die 2-Ebenenpartitionierung zulässig. \square

Die Zulässigkeit von 2-Ebenenpartitionierungen lässt sich also mittels erfüllter l-r-Lagebeziehungen ausdrücken. Die durch große Knoten induzierten l-r-Lagebeziehungen sind zyklensfrei,⁵¹ mit dem lokalen Lösungsansatz wird garantiert, dass diese Lagebeziehungen immer berücksichtigt werden.

⁵¹Eine l-r-Lagebeziehung von u_2 nach v_2 wird nur dann eingefügt, wenn in Ebene L_1 der Knoten u_1 links von v_1 liegt. Der durch l-r-Lagebeziehungen für große Knoten entstehende Graph ist ein DAG und damit frei von Zyklen.

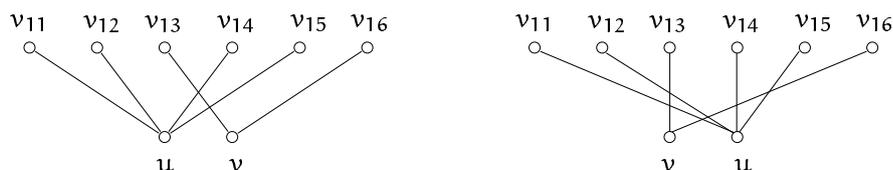


Abbildung 6.28: **Kreuzungen von Kanten.** (links) Die Zahl der sich kreuzenden Kanten inzident zu u oder v bei $L_2(u) < L_2(v)$ ist $k_{u,v} = 2$. In diesem Beispiel ist die Kreuzungszahl $K(L_1, L_2) = 2$, da nur das Knotenpaar u, v zu betrachten ist. (rechts) Die Sortierung $L_2(v) < L_2(u)$ ergibt $k_{v,u} = 6$.

Damit wird in 2-Ebenenpartitionierungen die korrekte Anordnung der Repräsentanten großer Knoten auf Ebene L_2 sichergestellt.

Kommen nun anwendergegebene l - r -Lagebeziehungen hinzu, so werden die Lagebeziehungen zwischen Repräsentanten großer Knoten durch höchstgewichtete l - r -Lagebeziehungen ausgedrückt.

Bei der Berechnung einer kreuzungsreduzierten Anordnung bilden die Gewichte nicht erfüllter l - r -Lagebeziehungen in der Ebene L_2 eine absteigend sortierte Folge A . Die neue Anordnung der Knoten in Ebene L_2 soll so erzeugt werden, dass A minimal bezüglich der lexikografischen Sortierung ist. Außerdem soll die Anordnung unter dieser Bedingung kreuzungsminimal sein. Da die für große Knoten eingefügten l - r -Lagebeziehungen das höchste Gewicht haben und zyklensfrei sind, werden sie immer erfüllt.

6.4.3 Theorie

Der eben skizzierte lokale Ansatz der Kreuzungsreduzierung wird formal definiert und untersucht. Dazu sind einige weitere Begriffe nötig (siehe auch Abb. 6.28):

DEFINITION 6.36 (KREUZUNGSZAHL [DETT99, San96b])

In einer Zeichnung einer 2-Ebenenpartitionierung $G = (L_1 \cup L_2, E)$, bei der Knoten aus L_1 und L_2 auf zwei parallelen Ebenen angeordnet und durch geradlinige Kanten verbunden sind, ist die Zahl der Kreuzungen zwischen Kanten inzident zu $u_1, u_2 \in L_2$ gegeben durch

$$k_{u_1, u_2} = |\{((o_1, u_1), (o_2, u_2)) \mid o_1, o_2 \in L_1, (o_1, u_1), (o_2, u_2) \in E, \\ L_1(o_1) > L_1(o_2), L_2(u_1) < L_2(u_2)\}|$$

k_{u_1, u_2} wird paarweise Kreuzungszahl genannt.

Für die Anordnung L_1 und L_2 gibt die Kreuzungszahl $K(L_1, L_2) = \sum_{u, v \in L_2, L_2(u) < L_2(v)} k_{u, v}$ die Gesamtzahl der Kreuzungen in der Zeichnung der 2-Ebenenpartitionierung an.

Nun wird eine 2-Ebenenpartitionierung definiert, die sowohl benutzergegebene l - r -Lagebeziehungen wie auch l - r -Lagebeziehungen für große Knoten enthält. Dabei muss garantiert werden, dass die Lagebeziehungen für große Knoten höher gewichtet sind als die anwendergegebenen Lagebeziehungen. Eine solche Ebenenpartitionierung wird als 2-Ebenenpartitionierung mit l - r -Lagebeziehungen (kurz 2-LR-Ebenenpartitionierung) bezeichnet.

DEFINITION 6.37 (2-LR-EBENENPARTITIONIERUNG)

Sei $G = (L_1 \cup L_2, E)$ eine 2-Ebenenpartitionierung und $G' = (L_1 \cup L_2, E, LB')$ die zugehörige 2-LRG-Ebenenpartitionierung. Sei weiter LB eine Menge von l - r -Lagebeziehungen, für die gilt: $\forall (u, v, l-r, g) \in LB \quad u, v \in L_2$. Sei $g_{\max} = \max\{g \mid (u, v, l-r, g) \in LB\}$ das maximale Gewicht der anwendergegebenen Lagebeziehungen.

Die zu G und G' gehörige 2-LR-Ebenenpartitionierung $G'' = (L_1'' \cup L_2'', E'', LB'')$ ist eine 2-Ebenenpartitionierung mit

$$\begin{aligned} L_1'' &= L_1 \\ L_2'' &= L_2 \\ E'' &= E \\ LB'' &= \{(u, v, l-r, g_{\max} + 1) \mid (u, v, l-r, g) \in LB'\} \cup \\ &\quad \{(u, v, l-r, g) \mid (u, v, l-r, g) \in LB, \\ &\quad (u, v, l-r, g_1) \notin LB', (v, u, l-r, g_2) \notin LB'\} \end{aligned}$$

Bei einer 2-LR-Ebenenpartitionierung handelt es sich also um eine 2-Ebenenpartitionierung, die l - r -Lagebeziehungen für die Anordnung großer Knoten und für anwendergegebene Lagebeziehungen enthält.

Nun lässt sich das Problem der Kreuzungsreduzierung für 2-LR-Ebenenpartitionierungen beschreiben.

DEFINITION 6.38 (KREUZUNGSREDUZIERUNGS-PROBLEM (KRP))

Sei $G = (L_1 \cup L_2, E, LB)$ eine 2-LR-Ebenenpartitionierung mit einer festen Anordnung L_1 . Gesucht ist eine Sortierung L_2' der Knoten von L_2 , so dass für $G' = (L_1 \cup L_2', E, LB)$ gilt:

1. G' ist eine zulässige 2-Ebenenpartitionierung
2. Die absteigend sortierte Folge A der Gewichte nicht erfüllter l - r -Lagebeziehungen in G' ist bezüglich der lexikografischen Sortierung minimal.
3. Unter den Bedingungen 1. und 2. ist die Kreuzungszahl $\kappa(L_1, L_2')$ minimal.

Um die Komplexität des KREUZUNGSREDUZIERUNGS-PROBLEMS (KRP) zu betrachten, wird zunächst das CROSSING PROBLEM wiederholt.

DEFINITION 6.39 (CROSSING PROBLEM (CP) [EW94])

Sei $G = (L_1 \cup L_2, E)$ eine 2-Ebenenpartitionierung mit einer festen Anordnung L_1 . Gesucht ist Sortierung L_2' der Knoten von L_2 , so dass die Kreuzungszahl $\kappa(L_1, L_2')$ minimal ist.

Das zum CROSSING PROBLEM zugehörige Entscheidungsproblem ist NP-vollständig [EW94]. Es ist offensichtlich, dass das KREUZUNGSREDUZIERUNGS-PROBLEM (KRP) eine Generalisierung des CROSSING PROBLEMS (CP) ist. Dazu betrachte man Graphen ohne l - r -Lagebeziehungen, bei diesen ist das KRP gleich dem CP.

Im Folgenden wird eine Heuristik zur Lösung des KREUZUNGSREDUZIERUNGS-PROBLEMS vorgestellt. Dabei wird die enge Verwandtschaft von CROSSING PROBLEM und FEEDBACK ARC SET PROBLEM ausgenutzt. Die zwischen den beiden Problemen existierende Beziehung wurde schon früh erkannt, so verwenden Sugiyama et al. [STT81] einen minimalen Feedback Arc Set zur Bestimmung einer optimalen Kreuzungsreduzierung, Eades und Wormald [EW94] reduzieren für ihren NP-Vollständigkeitsbeweis das FEEDBACK ARC SET PROBLEM auf das CROSSING PROBLEM.

6.4.4 Algorithmus

6.4.4.1 Idee der Heuristik

Die der Heuristik zu Grunde liegende Idee ist, das Problem der Kreuzungsreduzierung auf die Kantenorientierung zurückzuführen. Dazu wird aus der gegebenen 2-LR-Ebenenpartitionierung ein Graph aufgebaut, dessen Kanten sowohl l-r-Lagebeziehungen wie auch die Anordnung der Knoten auf Grund der paarweisen Kreuzungszahlen repräsentieren. Für diesen Graph wird eine Kantenorientierung mit der in Abschnitt 6.2 entwickelten Heuristik berechnet. Die dadurch erhaltene ebenenweise Anordnung der Knoten entspricht dann der links-rechts-Anordnung der Knoten in Ebene L_2 .

6.4.4.2 Algorithmus KREUZUNGSREDUZIERUNG

6.4.4.2.1 Teil 1

Im ersten Teil des Algorithmus KREUZUNGSREDUZIERUNG (siehe Seite 165) wird aus der gegebenen 2-LR-Ebenenpartitionierung $G = (L_1 \cup L_2, E, LB)$ und den Informationen über Repräsentanten großer Knoten der Graph $G' = (L_2, E', W)$ konstruiert (Zeilen 116-128 und Abb. 6.29). G' ist ein gewichteter Graph, der aus den Knoten von L_2 und neuen Kanten besteht. Diese Kanten drücken eine gewünschte Sortierung der Knoten in der Ebene L_2 aus. Sie bestehen

1. aus Kanten für l-r-Lagebeziehungen und
2. aus Kanten zwischen Knotenpaaren $u, v \in L_2$, die die lokal bessere Anordnung dieser Knoten repräsentieren. Für zwei Knoten $u, v \in L_2$ ist die Anordnung $(\dots, v, \dots, u, \dots)$ lokal besser als die Anordnung $(\dots, u, \dots, v, \dots)$, falls $k_{v,u} < k_{u,v}$ ist. Die lokal bessere Anordnung besitzt also weniger Kreuzungen der zu u und v inzidenten Kanten als die andere Anordnung. Im Falle $k_{v,u} = k_{u,v}$ sind beide Anordnungen gleich gut.

In G' werden zuerst Kanten für l-r-Lagebeziehungen eingefügt, diese erhalten $|E|^2 + g$ als Gewicht und sind dadurch höher als die Kanten für die lokal bessere Anordnung auf Grund der Kreuzungszahl gewichtet. Für jedes noch nicht durch eine Kante verbundene Knotenpaar u, v aus L_2 werden dann die paarweisen Kreuzungszahlen $k_{u,v}$ und $k_{v,u}$ verglichen. Es wird eine neue Kante in Richtung der kleineren paarweisen Kreuzungszahl eingefügt, also für $k_{u,v} < k_{v,u}$ die Kante (u, v) . Diese Kante erhält $ABS(k_{u,v} - k_{v,u})$ als Gewicht. Die Idee dabei ist, dass das Gewicht einer Kante umso höher ist, je mehr Kreuzungen lokal eingespart werden.

Die Gewichte der Kanten des Graphen G' drücken also aus, wie wichtig der Erhalt der durch sie gegebenen Teilordnung ist. Je höher das Gewicht, umso wichtiger ist die l-r-Lagebeziehung oder umso mehr Kreuzungen lassen sich vermeiden. Dabei ist darauf zu achten, dass Gewichte für l-r-Lagebeziehungen stets größer als Gewichte auf Grund $ABS(k_{u,v} - k_{v,u})$ sind. Kanten für l-r-Lagebeziehungen erhalten das Gewicht $|E|^2 + g$, da gilt: $\forall u, v \in L_2 \quad ABS(k_{u,v} - k_{v,u}) \leq |E|^2$.

6.4.4.2.2 Teil 2

Im zweiten Teil des Algorithmus KREUZUNGSREDUZIERUNG wird der eben konstruierte Graph G' von Zyklen befreit und die Kanten unter Berücksichtigung ihrer Gewichte orientiert (Zeile 130 und

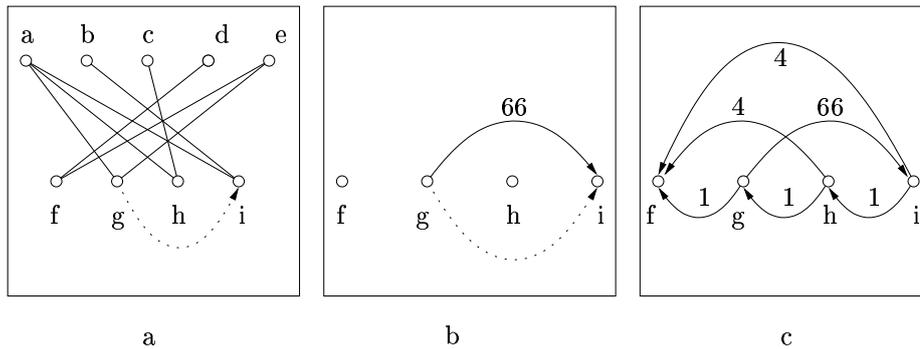


Abbildung 6.29: **Algorithmus KREUZUNGSREDUZIERUNG (Teil 1)**. (a) Aus der 2-LR-Ebenenpartitionierung $G = (L_1 \cup L_2, E, LB)$ mit der anwendergegebenen l-r-Lagebeziehung $(g, i, l-r, 2) \in LB$ (diese ist gepunktet dargestellt) wird G' berechnet.

(b) G' besteht aus den Knoten in L_2 , für l-r-Lagebeziehungen werden Kanten mit Gewicht $|E|^2 + g$ eingefügt, (c) für alle anderen Knotenpaare werden Kanten entsprechend der Beschreibung in Abschnitt 6.4.4.2.1 eingefügt. Es gilt beispielsweise $k_{f,g} = 2$ und $k_{g,f} = 1$.

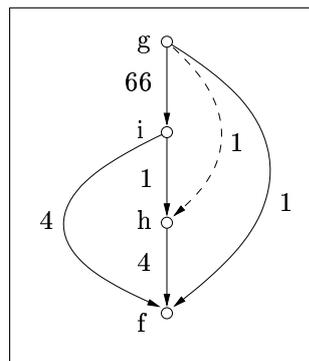


Abbildung 6.30: **Algorithmus KREUZUNGSREDUZIERUNG (Teil 2)**. Ergebnis der Kantenorientierung und der topologischen Sortierung. Die durch die Kantenorientierung umgedrehte Kante ist gestrichelt dargestellt.

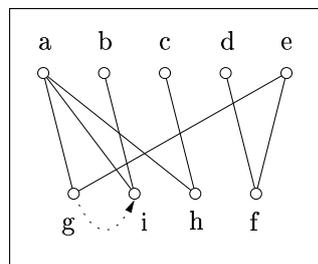


Abbildung 6.31: **Algorithmus KREUZUNGSREDUZIERUNG (Teil 3)**. Übertragung der berechneten Anordnung auf die Knoten der Ebene L_2 .

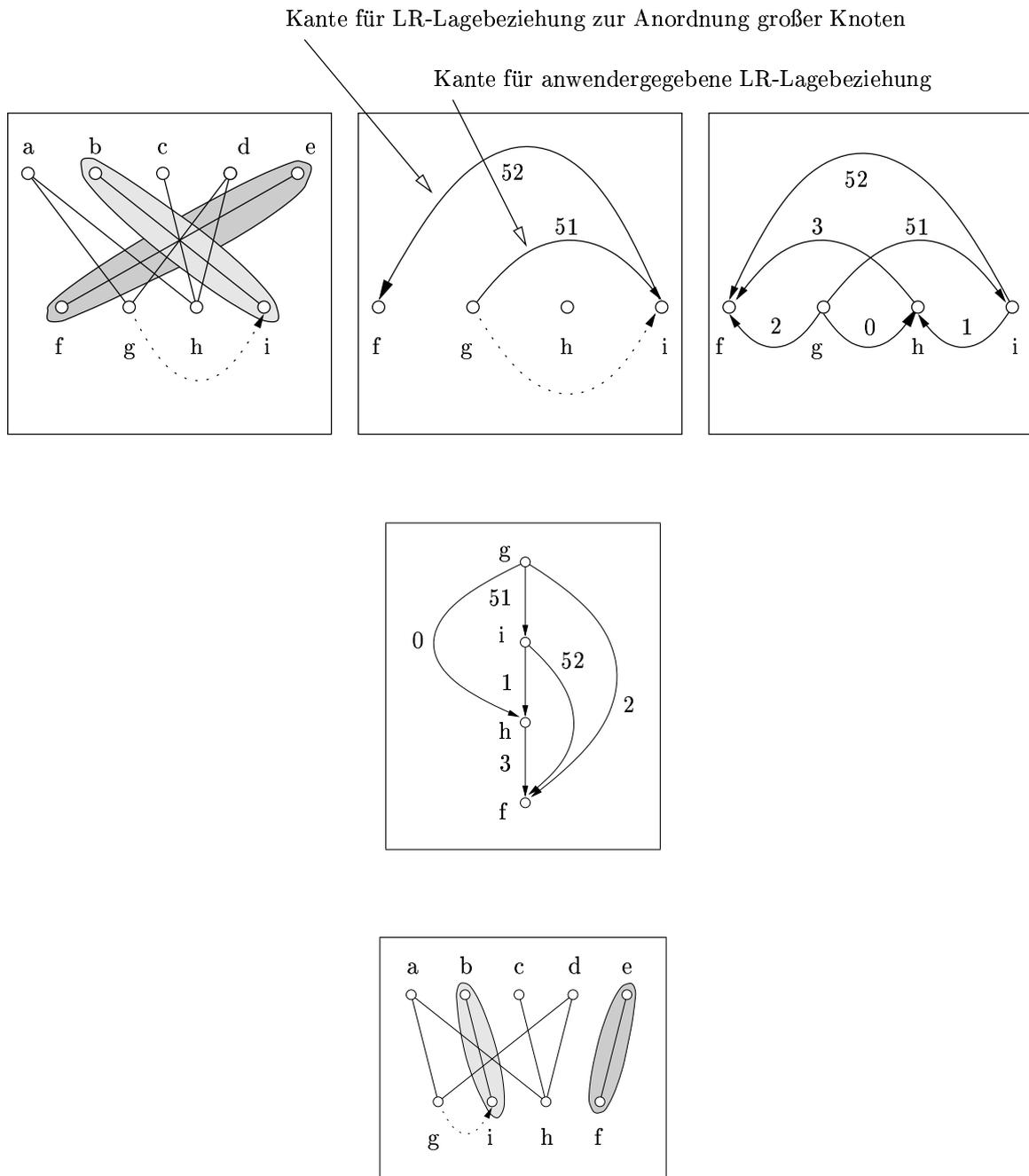


Abbildung 6.32: **Kreuzungsreduzierung mit großen Knoten.** Zwei große Knoten u, v mit $R(u) = \{b, i\}$, $R(v) = \{e, f\}$. Die Schritte der Kreuzungsreduzierung entsprechen denen in den Abbildungen 6.29-6.31, auch hier ist eine anwendergegebene l-r-Lagebeziehung $(g, i, l-r, 2)$ gepunktet dargestellt.⁵²

⁵²Auf Grund der großen Knoten hat der Graph in dieser Darstellung andere Kanten als der Graph in Abbildung 6.29.

Abb. 6.30). Dazu wird auf G' der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG (siehe Seite 127) angewandt. Die vorher eingefügten Kanten werden dabei als gewichtete o-u-Lagebeziehungen betrachtet. Gedanklich wird also der Graph um 90 Grad gedreht und statt einer links-rechts Ordnung wird nun eine oben-unten Ordnung berechnet.

Als Vereinfachung können statt o-u-Lagebeziehungen die Gewichte der Kanten direkt verwendet werden, indem der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG (Seite 127) ab Zeile 26 ausgeführt wird. Durch den Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG werden die Kanten so orientiert, dass G' zyklensfrei und unter Berücksichtigung der Gewichte gerichtet ist.

G' wird anschließend topologisch sortiert, die dabei entstehende Anordnung der Knoten der Ebene L_2 wird als L_2' bezeichnet (Zeile 131 und Abb. 6.30). Da G' azyklisch ist und durch die Konstruktion zwischen jedem Knotenpaar eine Kante existiert, liefert jede topologische Sortierung der Knoten des Graphen G' die gleiche eindeutige Sortierung.

6.4.4.2.3 Teil 3

Die neu berechnete Sortierung L_2' kann direkt als Anordnung der Knoten aus L_2 übernommen werden (Zeile 132 und 6.31).

Die Abbildung 6.32 zeigt die drei Schritte bei einer 2-LR-Ebenenpartitionierung, die zusätzlich Repräsentanten großer Knoten enthält.

6.4.4.2.4 Algorithmus

Algorithmus: KREUZUNGSREDUZIERUNG.

Gegeben: 2-LR-Ebenenpartitionierung $G = (L_1 \cup L_2, E, LB)$, Information über Repräsentanten großer Knoten.

Gesucht: Kreuzungsreduzierte zulässige 2-Ebenenpartitionierung $G = (L_1 \cup L_2', E)$.

```

113 Berechne für alle  $u, v \in L_2$  den Wert  $k_{u,v}$ ; 53
114
115 /* Konstruiere Graph  $G' = (L_2, E')$  mit Gewichten  $g[]$  */
116 forall  $(u, v, l-r, g) \in LB$ 
117      $E' := E' \cup \{(u, v)\}$ ;
118      $g[(u, v)] := |E|^2 + g$ ;
119
120 forall  $u \in L_2$ 
121     forall  $v \in L_2$ 
122         if  $((u, v) \notin E')$  and  $((v, u) \notin E')$ 
123             if  $k_{u,v} \leq k_{v,u}$ 
124                  $E' := E' \cup \{(u, v)\}$ ;
125                  $g[(u, v)] := (k_{v,u} - k_{u,v})$ ;
126             else
127                  $E' := E' \cup \{(v, u)\}$ ;
128                  $g[(v, u)] := (k_{u,v} - k_{v,u})$ ;

```

⁵³Die Werte $k_{u,v}$ für alle $u, v \in L_2$ können als Matrix gespeichert werden.

```

129 /* Zyklenbeseitigung und Anordnung auf  $L_2$  */
130 Wende auf  $G' = (L_2, E')$  den Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG an;
131 Sortiere  $G' = (L_2, E')$  topologisch;54
132 Ordne Knoten in  $L_2$  entsprechend der Sortierung an;
133 /* Die neue Anordnung wird als  $L_2'$  bezeichnet. */

```

SATZ 6.40

Sei $G = (L_1 \cup L_2, E, LB)$ eine 2-LR-Ebenenpartitionierung und seien Informationen über große Knoten gegeben. Der Algorithmus KREUZUNGSREDUZIERUNG berechnet eine zulässige 2-l-r-Ebenenpartitionierung.

Beweis: Sei $G = (L_1 \cup L_2, E, LB)$ eine 2-LR-Ebenenpartitionierung. Zunächst wird die zugehörige 2-LRG-Ebenenpartitionierung $G'' = (L_1 \cup L_2, E, LB)$ betrachtet, die nur l-r-Lagebeziehungen für große Knoten entsprechend Definition 6.34 enthält. Für diese Lagebeziehungen werden in den Zeilen 116-118 Kanten in den Graphen $G' = (L_2, E')$ eingefügt. Da die l-r-Lagebeziehungen für große Knoten frei von Zyklen sind, ist auch G' ein azyklischer Graph. Nach Satz 6.22 wird in diesem Fall durch den Algorithmus KANTENORIENTIERUNG keine Kante umgedreht. Eine topologische Sortierung der Knoten liefert eine Anordnung der Knoten für die 2-Ebenenpartitionierung, in der jede l-r-Lagebeziehung erfüllt ist. Nach Satz 6.35 ist damit die am Ende des Algorithmus entstehende kreuzungsreduzierte 2-LRG-Ebenenpartitionierung zulässig.

Für die 2-LR-Ebenenpartitionierung G haben alle anwendergegebenen l-r-Lagebeziehungen ein geringeres Gewicht als die l-r-Lagebeziehungen für große Knoten, deshalb werden nach Satz 6.22 auch in diesem Fall alle Lagebeziehungen für große Knoten erfüllt. Damit ist wieder die durch den Algorithmus KREUZUNGSREDUZIERUNG berechnete 2-LR-Ebenenpartitionierung zulässig. \square

SATZ 6.41

Der Algorithmus KREUZUNGSREDUZIERUNG benötigt $O(\max\{|E|^2, |L_2|^4\})$.

Beweis: Im ersten Teil des Algorithmus (Zeilen 116-128) wird der Graph G' erzeugt. Dabei benötigt die erste **forall**-Schleife $O(|LB|)$ mit $|LB| < |L_2|^2$. Die Berechnung der Werte von $k_{u,v}$ für alle $u, v \in L_2$ kann in $O(|E|^2)$ durchgeführt werden [VML96,YS99]. Die Werte werden in einer Matrix gespeichert. Da der Graph G' $\frac{|L_2| \cdot (|L_2| - 1)}{2}$ Kanten enthält, benötigt Zeile 130, die Kantensorientierung, $O(|L_2|^4)$ und die Zeile 131, die topologische Sortierung, $O(|L_2|^2)$. Das Übertragen der Anordnung ist schließlich in $O(|L_2|)$ möglich.

Die gesamte Heuristik benötigt damit $O(\max\{|E|^2, |L_2|^4\})$. \square

6.4.5 Weitere Verfahren zur Kreuzungsreduzierung

6.4.5.1 Verfahren

Abschließend werden weitere Verfahren zur Reduzierung von Kantenkreuzungen wiederholt. Dabei wird das hier entwickelte Verfahren auch mit anderen Verfahren, die zur Berücksichtigung von l-r-Lagebeziehungen erweitert wurden, verglichen.

⁵⁴Ein Algorithmus zur topologischen Sortierung findet sich z. B. [AHU83].

Bereits länger bekannt sind *Barycenter*- [Car80, STT81, GNV88] und *Median*-Methoden [EW86, EW94, GKNV93], bei denen die Knoten auf Ebene L_2 nur auf Grund der relativen Lage ihrer Vorgänger in Ebene L_1 angeordnet werden, die Zahl der Kantenkreuzungen wird dabei nicht berücksichtigt. Die Position eines Knotens bestimmt sich bei *Barycenter* aus dem arithmetischen Mittel der Positionen seiner Vorgänger, bei *Median* wird die Position des mittleren Vorgängers verwendet. Rowe et al. [RDM⁺87] erweitern *Barycenter*, so dass zur Bestimmung der Position eines Knotens vorherige und nachfolgende Ebenen gemeinsam betrachtet werden.

Eades und Kelly [EK86] schlagen die Heuristiken *Splitt*, *Greedy Switch* und *Greedy Insert* vor, die die paarweisen Kreuzungszahlen berücksichtigen. Bei *Greedy Switch* werden benachbarte Knoten testweise vertauscht, bei *Greedy Insert* wird die Anordnung der Knoten ebenenweise aufgebaut, wobei für den aktuellen Knoten jeweils die beste Position ermittelt wird. *Splitt* wählt ein Pivotknoten v und platziert alle anderen Knoten links oder rechts von v , je nachdem, wo weniger Kreuzungen entstehen. Dieses Verfahren zur Knotenanordnung wird rekursiv auf die linke und rechte Seite angewandt. Mäkinen [Mäk90a] verbessert *Greedy Switch* mittels der Verwendung von *Barycenter* als Präprozess.

Sanders [San96b] initialisiert die Knotenreihenfolge, indem mittels Tiefensuche ein Wald aufspannender Bäume berechnet und die Knoten entsprechend angeordnet werden. Danach wird *Barycenter*, *Median* bzw. eine Kombination der beiden Verfahren angewandt. Zudem können Knoten testweise vertauscht werden. Laguna et al. [LMV97] lösen das Problem mittels *Tabu Search*⁵⁵. Molitor et al. [MSM99, GSBM01] verwenden *Sifting* [Rud93], Mäkinen and Sieranta [MS94] stellen eine Heuristik vor, die auf genetischen Algorithmen basiert. Dresbach [Dre95] schlägt eine Einfügemethode mit stochastischem Ansatz vor. Zur Ermittlung der Einfügereihenfolge werden Bewertungszahlen für noch nicht berücksichtigte Knoten und freie Positionen verwendet. Der Knoten mit der kleinsten Bewertungszahl wird an der Position eingefügt, an der er diese Bewertungszahl annimmt. Catarci [Cat95] reduziert das Problem der Kreuzungsreduzierung auf ein Zuweisungs-Problem.

Melancon und Hermann [MH99] schlagen vor, statt der üblichen Verfahren Spannbäume zu berechnen und deren Knoten mittels Algorithmen zum Zeichnen von Bäumen zu platzieren. Tamassia et al. [TDB88] und Mutzel [Mut97] berechnen einen entsprechend der Ebenenpartitionierung maximalen planaren Teilgraphen. Dessen Platzierung bestimmt die Anordnung der Knoten, die dabei temporär entfernten Kanten werden am Ende des Verfahrens wieder eingefügt.

Einen Überblick und Vergleiche zu Heuristiken bieten Jünger und Mutzel [JM96a, JM97] sowie Martí und Laguna [ML97]. Dabei zeigt sich, dass die Unterschiede der meisten Heuristiken nur für lichte Graphen signifikant sind. Da *Barycenter* oft gute Ergebnisse liefert, sehr kurze Rechenzeiten benötigt und zudem einfach zu implementieren ist, hat sich dieses Verfahren als Heuristik zur Kreuzungsreduzierung durchgesetzt. Und dies, obwohl für *Barycenter* keine qualitativen Aussagen nachgewiesen werden konnten. Im Gegensatz dazu handelt es sich bei *Median* [EW86] und bei einem Verfahren von Yamaguchi und Sugimoto [YS99] um Approximationsalgorithmen. Vergleiche in den eben erwähnten Arbeiten zeigen jedoch, dass *Median* in praktischen Anwendungen deutlich schlechter als *Barycenter* ist.

Exakte Lösungen stammen von Valls et al. [VML96] und Jünger et al. [JLMO97, JM97]. In diesen Arbeiten wird das Problem als ganzzahliges Programm formuliert und mit *Branch-And-Bound*- und *Branch-And-Cut*-Verfahren⁵⁶ gelöst. Für Graphen mit bis zu 60 Knoten in der Ebene empfehlen Jün-

⁵⁵Tabu Search ist eine Meta-Heuristik zur Lösung kombinatorischer Optimierungsprobleme, siehe [Glo89].

⁵⁶Siehe Fußnote auf Seite 134.

ger und Mutzel die Bestimmung der exakten Lösung, für Ebenen mit mehr Knoten die Verwendung der Barycenter-Heuristik [JM97]. In [SBG99] beschäftigen sich die Autoren mit der Abschätzung der besten Heuristik für eine gegebene Rechenzeit.

Die Arbeiten [JM97, SBG99, SSSV97] betrachten auch exakte Lösungen des zweiseitigen Kreuzungsreduzierungsproblems. Bei diesem ist die Anordnung der Knoten der Ebene L_1 nicht fixiert, sondern beide Ebenen können gleichzeitig permutiert werden. Die Erweiterung der Heuristiken für zweiseitige Kreuzungsreduzierung wurde schon in Form des globalen Verfahrens (siehe Seite 156) vorgestellt.

6.4.5.2 Erweiterung existierender Heuristiken um Lagebeziehungen

Prinzipiell lassen sich bekannte Heuristiken zur Kreuzungsreduzierung so erweitern, dass sie l - r -Lagebeziehungen berücksichtigen. Dazu müssen bei der Bestimmung der Anordnung der Knoten in Ebene L_2 nur jene Permutationen ausgeschlossen werden, welche die Lagebeziehungen nicht erfüllen.

Entsprechende Erweiterungen der traditionellen Verfahren wurden bisher jedoch kaum beachtet. Lediglich Sanders [San96b] und Waddle [Wad01] schlagen vor, herkömmliche Verfahren wie *Barycenter* oder *Median* so zu erweitern, dass vor einer Neuordnung eines Knotens getestet wird, ob dadurch die l - r -Lagebeziehungen erfüllt werden. Ist dies nicht der Fall, so wird der Knoten nicht auf die neue Position gesetzt, sondern behält seine alte Position bei.

Im Folgenden soll das hier vorgestellte Verfahren mit anderen Lösungen verglichen werden. Dazu werden die Ideen von Sanders und Waddle aufgegriffen und die Algorithmen BARYCENTER und MEDIAN zur Berücksichtigung von l - r -Lagebeziehungen, wie eben beschrieben, erweitert. Zyklische Abhängigkeiten zwischen l - r -Lagebeziehungen werden in einem Präprozess beseitigt.

6.4.5.3 Vergleich der Verfahren

In den Abbildungen 6.33–6.35 wird der Algorithmus KREUZUNGSREDUZIERUNG mit den um die Berücksichtigung von l - r -Lagebeziehungen erweiterten Verfahren BARYCENTER und MEDIAN verglichen. Zusätzlich ist auch die Zahl der Kreuzungen vor der Kreuzungsreduzierung angegeben.

Aus den Diagrammen lassen sich mehrere Aussagen ableiten:

1. Durch die Kreuzungsreduzierung wird die Zahl der Kantenkreuzungen deutlich verringert. Diese Beobachtung wurde auch in den in Abschnitt 6.4.5.1 erwähnten Arbeiten gemacht.
2. Bei steigender Zahl vorgegebener l - r -Lagebeziehungen sinkt die Zahl der eingesparten Kreuzungen.

Dies liegt daran, dass durch mehr l - r -Lagebeziehungen die Zahl möglicher Permutationen in der Ebene L_2 eingeschränkt wird. Besonders stark tritt dieser Effekt bei BARYCENTER und MEDIAN auf, während der Algorithmus KREUZUNGSREDUZIERUNG auch bei vielen l - r -Lagebeziehungen die Zahl der Kantenkreuzungen noch stark reduziert.

3. Der Algorithmus KREUZUNGSREDUZIERUNG liefert in den meisten Fällen bessere Ergebnisse als die angepassten Algorithmen BARYCENTER und MEDIAN.

Insbesondere bei einer größeren Zahl von l - r -Lagebeziehungen ist das Ergebnis des Algorithmus KREUZUNGSREDUZIERUNG stets deutlich besser als die Ergebnisse der anderen Verfahren.

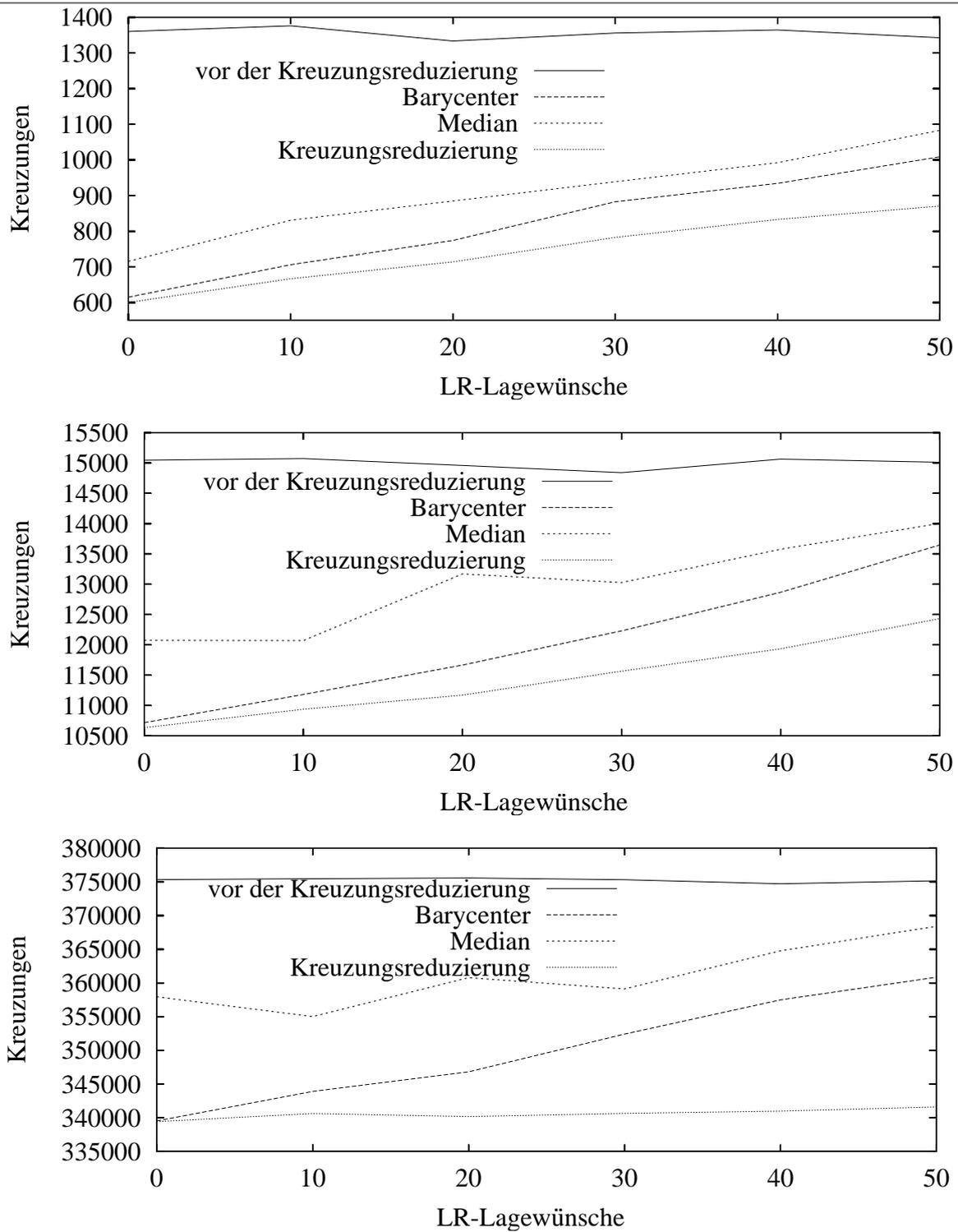


Abbildung 6.33: **Vergleiche zur Kreuzungsreduzierung (Teil 1)**. Tests mit kardinalitätsgleichen 2-LR-Ebenenpartitionierungen $V = (L_1 \cup L_2, E, LB)$ mit $|L_1| = |L_2| = 50$ und $0, 10, \dots, 50$ zufälligen l-r-Lagebeziehungen (jeweils 100 kardinalitätsgleiche Graphen). Es sind die Zahl der Kreuzungen vor und nach dem Algorithmus KREUZUNGSREDUZIERUNG sowie nach den angepassten Algorithmen BARYCENTER und MEDIAN dargestellt: (oben) $|E| = 75$ (Dichte 0,03), (mitte) $|E| = 250$ (Dichte 0,1), (unten) $|E| = 1250$ (Dichte 0,5). Da ebenenweisen Zeichenverfahren überwiegend auf lichte Graphen angewandt werden, wurden hier besonders solche Graphen berücksichtigt.

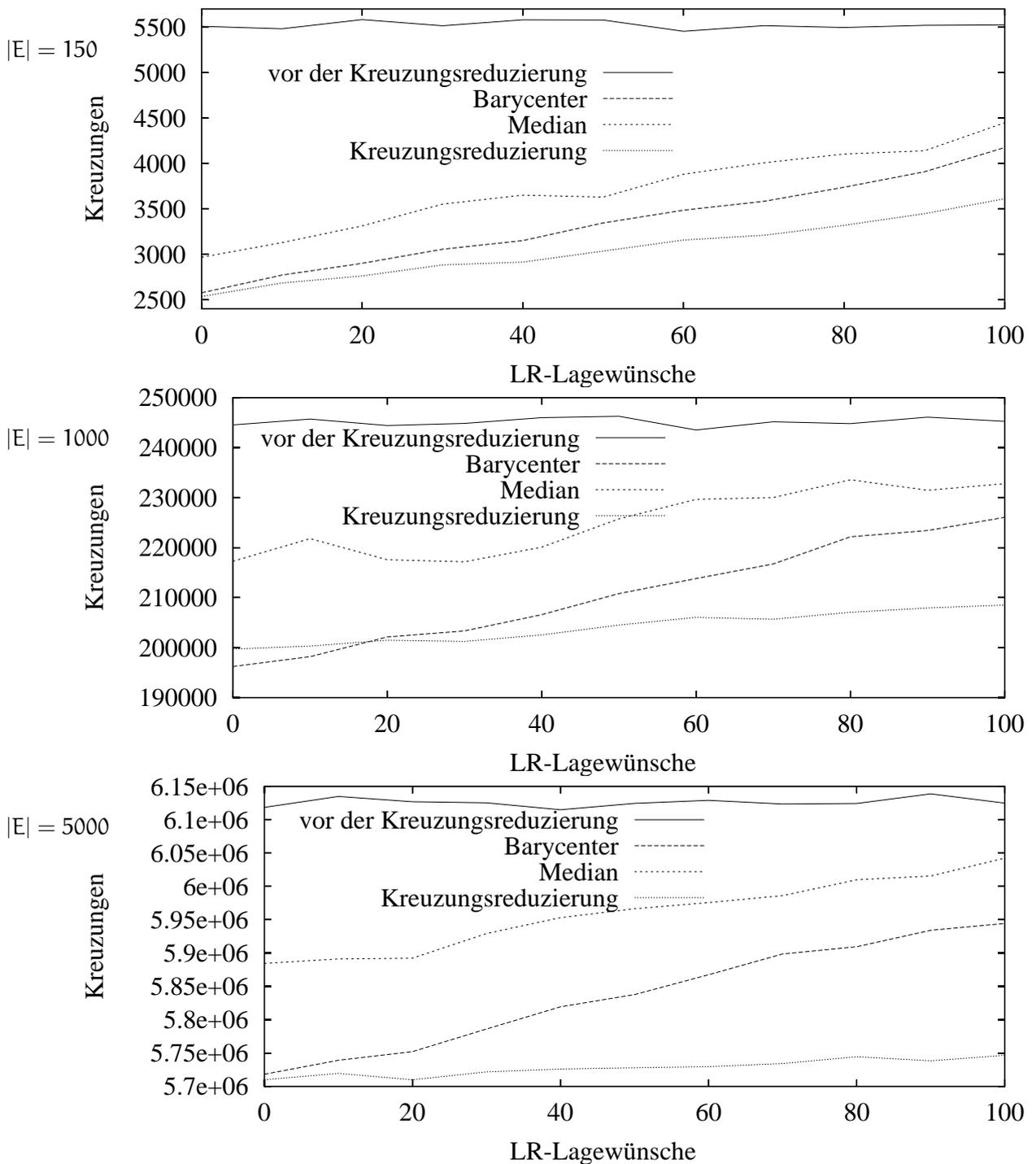


Abbildung 6.34: **Vergleiche zur Kreuzungsreduzierung (Teil 2)**. Tests mit kardinalitätsgleichen 2-LR-Ebenenpartitionierungen $V = (L_1 \cup L_2, E, LB)$ mit $|L_1| = |L_2| = 100$ und $0, 10, \dots, 100$ zufälligen l - r -Lagebeziehungen (jeweils 100 kardinalitätsgleiche Graphen). Es sind die Zahl der Kreuzungen vor und nach dem Algorithmus KREUZUNGSREDUZIERUNG sowie nach den angepassten Algorithmen BARYCENTER und MEDIAN dargestellt: (oben) $|E| = 150$ (Dichte 0,015), (mitte) $|E| = 1000$ (Dichte 0,1), (unten) $|E| = 5000$ (Dichte 0,5).

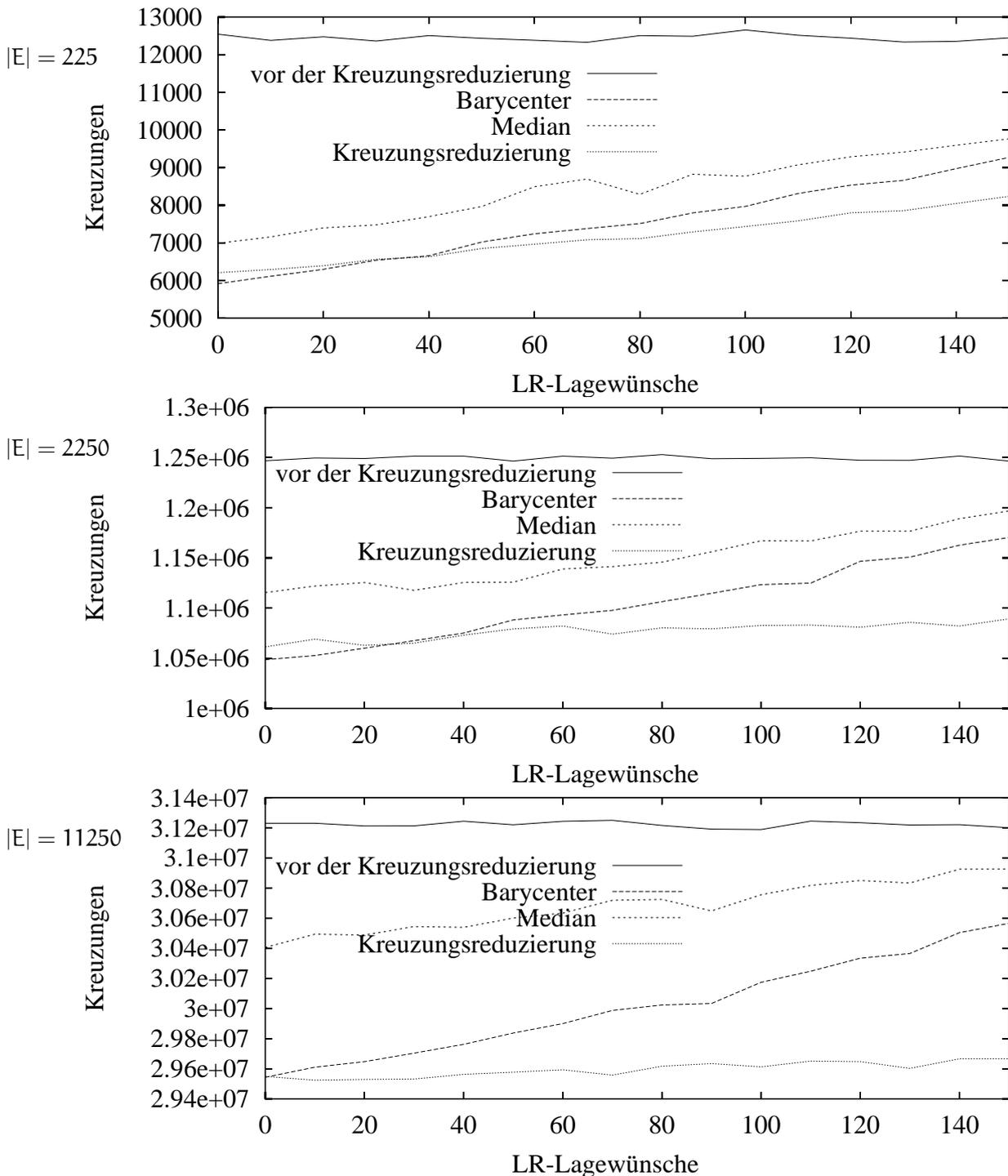


Abbildung 6.35: **Vergleiche zur Kreuzungsreduzierung (Teil 3)**. Tests mit kardinalitätsgleichen 2-LR-Ebenenpartitionierungen $V = (L_1 \cup L_2, E, LB)$ mit $|L_1| = |L_2| = 150$ und $0, 10, \dots, 150$ zufälligen l-r-Lagebeziehungen (jeweils 100 kardinalitätsgleiche Graphen). Es sind die Zahl der Kreuzungen vor und nach dem Algorithmus KREUZUNGSREDUZIERUNG sowie nach den angepassten Algorithmen BARYCENTER und MEDIAN dargestellt: (oben) $|E| = 225$ (Dichte 0,01), (mitte) $|E| = 2250$ (Dichte 0,1), (unten) $|E| = 11250$ (Dichte 0,5).

Lediglich für Graphen ohne bzw. mit einigen wenigen l-r-Lagebeziehungen liefert BARYCENTER bessere Ergebnisse.

4. BARYCENTER liefert im Vergleich zu MEDIAN nicht nur bei Graphen ohne l-r-Lagebeziehungen bessere Ergebnisse [JM96b], sondern ist auch bei Graphen mit l-r-Lagebeziehungen besser.

Die Abbildungen 6.36 bis 6.37 zeigen die benötigten Rechenzeiten für 2-LR-Ebenenpartitionierungen mit 100 bzw. 150 Knoten pro Ebenen. Die schnellsten der hier untersuchten Verfahren sind die erweiterten Verfahren BARYCENTER und MEDIAN, während der Algorithmus KREUZUNGSREDUZIERUNG mehr Rechenzeit benötigt. Für 2-LR-Ebenenpartitionierungen mit 50 Knoten pro Ebene lagen die Rechenzeiten für die erweiterten Verfahren unterhalb der Messschwelle von 0,01 Sekunden, für den Algorithmus KREUZUNGSREDUZIERUNG zwischen 0,01 und 0,03 Sekunden.

In den Diagrammen ist für den Algorithmus KREUZUNGSREDUZIERUNG die Zeit mit und ohne Bestimmung der paarweisen Kreuzungszahlen $k_{u,v}$ angegeben. Diese werden nicht nur hier, sondern auch in vielen traditionellen Verfahren in einem Präprozess berechnet. Heuristiken mit Berücksichtigung lokaler Kreuzungszahlen, aber auch exakte Berechnungsverfahren benötigen mindestens die zur Bestimmung der lokalen Kreuzungszahlen nötige Rechenzeit. Diese wird in der hier untersuchten Implementierung wesentlich durch die Verwendung von GTL [GTL01] als Graphbibliothek bestimmt. Besonders bei großen Graphen überwiegt die zur Berechnung der lokalen Kreuzungszahlen nötige Zeit deutlich gegenüber der Zeit zur eigentlichen Bestimmung der Knotenanordnung.

Die Rechenzeit für den Algorithmus KREUZUNGSREDUZIERUNG liegt selbst für Graphen mit 150 Knoten pro Ebene für die Bestimmung der Anordnung (ohne Berechnung der paarweisen Kreuzungszahlen) noch unter einer Sekunde. Da bei diesen Graphen (insbesondere bei Dichten $\geq 0,5$) die Gesamtzeit des Algorithmus bereits pro Messung bei mehreren Minuten lag, wurden keine größeren Graphen untersucht. Die Gesamtlaufzeit ergibt sich u. a. aus der für das Laden des Graphen und damit verbunden dem Aufbau der Datenstruktur, dem Aufbau der Ebenenpartitionierung und weiterer Verwaltungsarbeiten notwendigen Zeit.

6.4.5.4 Exakte Lösung

Für die Bestimmung der minimalen Zahl von Kreuzungen in 2-Ebenenpartitionierungen $G = (L_1 \cup L_2, E)$ ohne l-r-Lagebeziehungen geben Jünger und Mutzel [JM97] sowie Sanders [San96b] ein Verfahren mittels ganzzahliger linearer Programmierung an. Dazu wird für jedes Knotenpaar u, v der Ebene L_2 eine Variable $l_{u,v}$ eingeführt, um die aktuelle Anordnung zu beschreiben. Es gilt

$$l_{u,v} = \begin{cases} 1 & \text{falls } L_2(u) < L_2(v) \\ 0 & \text{sonst} \end{cases}$$

Die Zahl der Kreuzungen einer Anordnung der Knoten ist nun gegeben durch

$$K(L_1, L_2) = \sum_{u,v \in L_2} k_{u,v} * l_{u,v} \quad (6.6)$$

Nebenbedingung ist dabei, dass

1. nur einer der beiden Fälle u links von v oder v links von u auftreten darf. Dies wird ausgedrückt durch

$$l_{u,v} + l_{v,u} = 1 \text{ für alle } u, v \in L_2 \quad (6.7)$$

2. keine zyklischen Abhängigkeiten, z. B. u links von v , v links von w , w links von u vorhanden sein dürfen. Dies wird ausgedrückt durch

$$1 \leq l_{u,v} + l_{v,w} + l_{w,u} \leq 2 \text{ für alle } u, v, w \in L_2 \quad (6.8)$$

Gesucht ist eine ganzzahlige Lösung, welche die Summe in 6.6 unter den Nebenbedingungen 6.7 und 6.8 minimiert.

Dieser Ansatz lässt sich leicht auf l-r-Lagebeziehungen erweitern. Dafür werden in einem Präprozess zyklische Abhängigkeiten zwischen den l-r-Lagebeziehungen beseitigt. Danach können die verbliebenen Lagebeziehungen in das Gleichungssystem integriert werden, indem für jede Lagebeziehung $(u, v, l-r, g)$ statt 6.7 die Nebenbedingung

$$l_{u,v} = 1 \text{ und } l_{v,u} = 0$$

eingefügt wird.

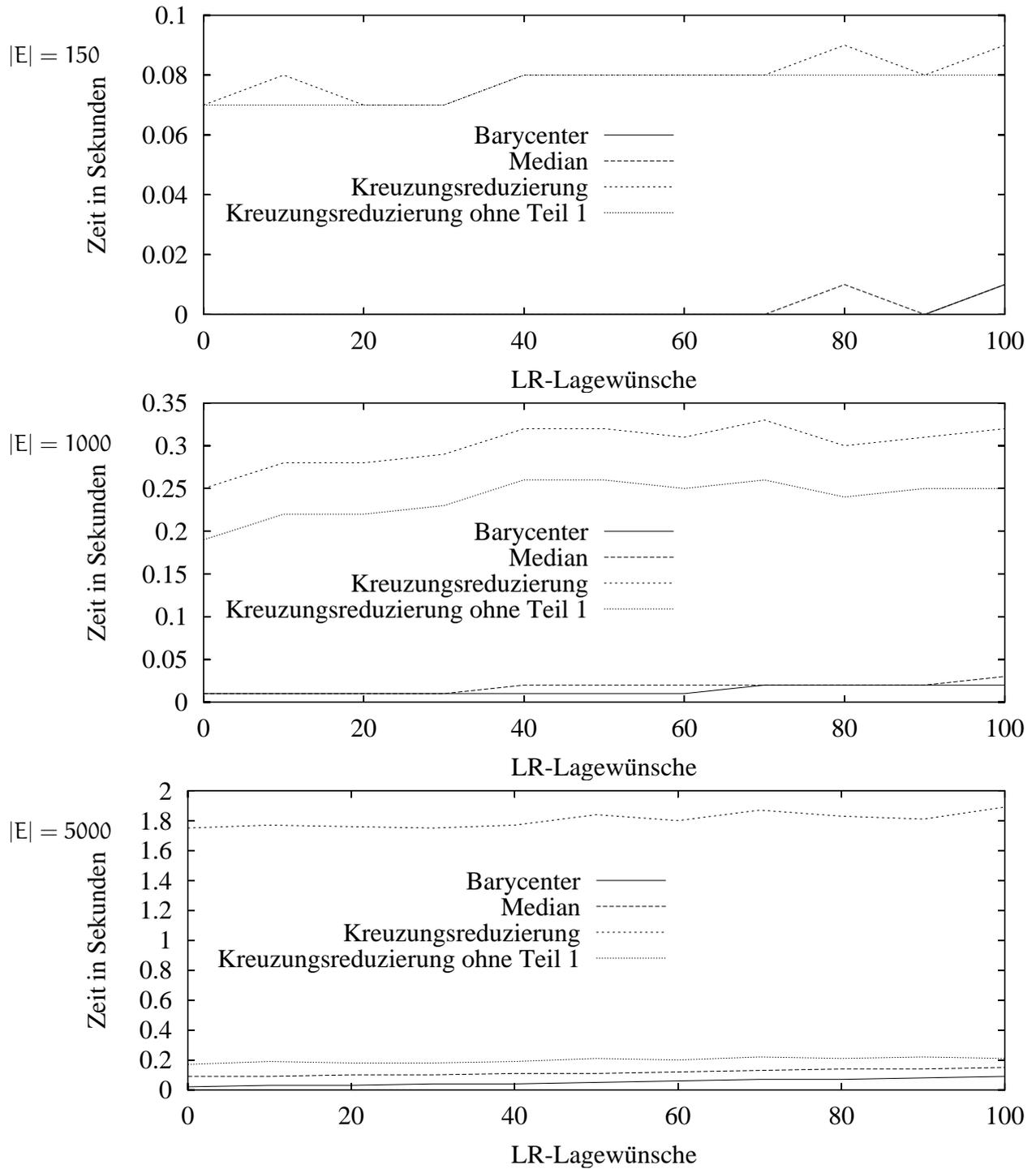


Abbildung 6.36: **Vergleiche zur Kreuzungsreduzierung - Rechenzeit (Teil 1)**. Rechenzeiten der angepassten Algorithmen BARYCENTER und MEDIAN im Vergleich zum Algorithmus KREUZUNGSREDUZIERUNG, die Diagramme korrespondieren mit denen in Abbildung 6.34 ($|L_1| = |L_2| = 100$, (oben) $|E| = 150$ (Dichte 0,015), (mitte) $|E| = 1000$ (Dichte 0,1), (unten) $|E| = 5000$ (Dichte 0,5)). KREUZUNGSREDUZIERUNG *ohne Teil 1* gibt die Rechenzeit für die KREUZUNGSREDUZIERUNG ohne die Zeit zur Bestimmung der paarweisen Kreuzungszahlen an.

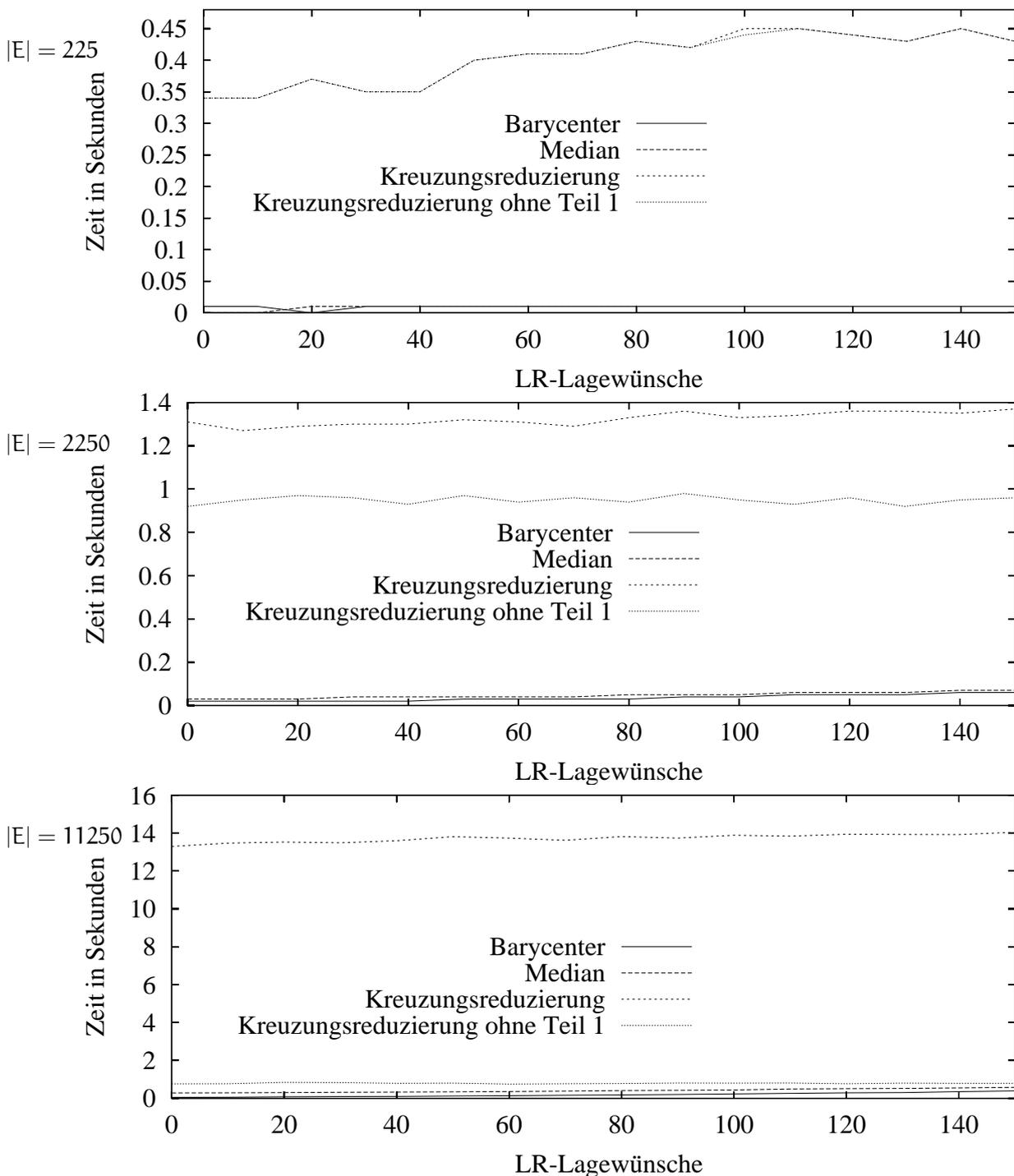


Abbildung 6.37: **Vergleiche zur Kreuzungsreduzierung - Rechenzeit (Teil 2)**. Rechenzeiten der angepassten Algorithmen BARYCENTER und MEDIAN im Vergleich zum Algorithmus KREUZUNGSREDUZIERUNG, die Diagramme korrespondieren mit denen in Abbildung 6.35 ($|L_1| = |L_2| = 150$, (oben) $|E| = 225$ (Dichte 0,01), (mitte) $|E| = 2250$ (Dichte 0,1), (unten) $|E| = 11250$ (Dichte 0,5)). KREUZUNGSREDUZIERUNG *ohne Teil 1* gibt die Rechenzeit für die KREUZUNGSREDUZIERUNG ohne die Zeit zur Bestimmung der paarweisen Kreuzungszahlen an.

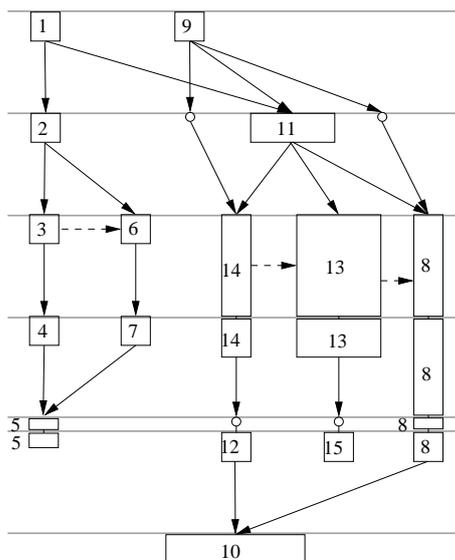
6.5 x-Koordinaten

6.5.1 Überblick

Abschnitt 6.5 behandelt die vierte Phase des VGL-Verfahrens: Die *Bestimmung der x-Koordinaten*.

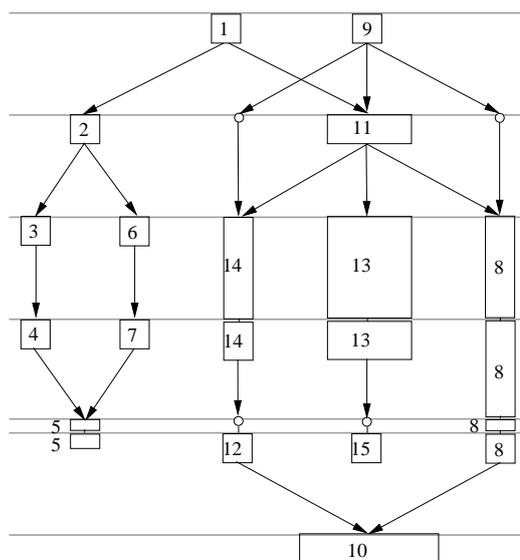
Ziel der Bestimmung der x-Koordinaten:

kreuzungsreduzierte l-Ebenenpartitionierung



→

l-Ebenenpartitionierung mit x-Koordinaten für Knoten



→

6.5.2 Berechnung der x-Koordinaten

Durch die Berechnung der x-Koordinaten wird die endgültige Visualisierung des Graphen bestimmt. Die vorherigen Schritte des VGL-Verfahrens legten bisher nur die y-Koordinaten und die Reihenfolge der Knoten auf den Ebenen fest, nun werden Knoten auf der Ebene fixiert. Die Berechnung der x-Koordinaten wird durch das Ästhetikkriterium ÄK_6 beeinflusst: Knoten sollen so auf den Ebenen platziert werden, dass sie zwischen ihren Vorgängern und Nachfolgern balanciert sind. Außerdem sollen alle Teilkanten langer Kanten (außer dem ersten und dem letzten Kantensegment) vertikal gezeichnet werden, da dadurch das Erscheinungsbild der Zeichnung verbessert wird und der Anwender langen Kanten leichter folgen kann. Bei der Berechnung der x-Koordinaten ist nun wichtig, dass alle Repräsentanten eines großen Knotens dieselbe x-Koordinaten erhalten.

Die x-Koordinaten der Knoten lassen sich trotz der zusätzlichen Anforderung mit herkömmlichen Verfahren berechnen. So kann Sanders Gummibandmethode mit linearen Segmenten [San96a, San96b] verwendet werden. Bei der Gummibandmethode richtet sich die Platzierung eines Knotens an den x-Koordinaten seiner Vorgänger und Nachfolger aus, die Kanten werden als Gummibänder betrachtet, die möglichst gleich stark gespannt sind. Die optimale x-Koordinate eines Knotens in

der Ebene L_i ergibt sich aus dem arithmetischen Mittel⁵⁷ der x -Koordinaten seiner Vorgänger und Nachfolger. *Lineare Segmente* sind Folgen von adjazenten Knoten aufeinander folgender Ebenen, die gemeinsam verschoben werden und in denen alle Knoten dieselbe x -Koordinate besitzen. Dabei hat jeder Knoten der Folge nur einen Vorgänger und einen Nachfolger, nur der erste Knoten kann mehrere Vorgänger und der letzte Knoten mehrere Nachfolger besitzen. Zur Berücksichtigung linearer Segmente führt Sanders einen *Segmentordnungsgraph* ein, der die linearen Segmente von links nach rechts anordnet. Dabei muss durch den vorherigen Schritt der Kreuzungsreduzierung sichergestellt sein, dass sich Segmente nicht kreuzen. Große Knoten werden berücksichtigt, indem die Repräsentanten eines großen Knotens als lineares Segment betrachtet werden.

Für die Berechnung der x -Koordinaten kann auch das Verfahren von Buchheim et al. [BJL01] verwendet werden. Auch dabei werden große Knoten durch lineare Segmente repräsentiert. Buchheim et al. berechnen wie Sanders zuerst einen Segmentordnungsgraph. Die originalen Knoten werden dann entsprechend der gegebenen Anordnung in der Ebene zwischen die vertikalen Segmente platziert. Vorteile des Verfahrens in [BJL01] sind die Komplexität von $O(|E| \log |E|)$ für zusammenhängende Graphen und die optimale Platzierung der Knoten zwischen vertikalen Segmenten.

Die Ansätze von Sanders [San96b] und Buchheim et al. [BJL01] garantieren zudem, dass Kanten mit maximal zwei Knicken und einem vertikalem Mittelstück zwischen den Knicken dargestellt werden.

Neben diesen beiden Verfahren gibt es einige weitere Ansätze. So unterscheiden Sugiyama et al. [STT81] bei der Berechnung der x -Koordinaten zwischen temporären Knoten, die zuerst platziert werden, und normalen Knoten. Für Knotenfolgen, die linearen Segmenten entsprechen, wird dadurch jedoch keine vertikale Anordnung garantiert, so dass das Verfahren hier nicht geeignet ist. Auch ein Vorschlag von Eades und Sugiyama [ES90], die die Berechnung der x -Koordinaten als quadratisches Optimierungsproblem betrachten, garantiert für Teile großer Knoten keine gemeinsame x -Koordinate. Das Ziel des Platzierungsverfahren in [ES90] ist die Vermeidung von Knickpunkten, wodurch lineare Segmente im Allgemeinen schräg statt vertikal dargestellt werden. Gansner et al. [GKNV93] stellen zwei weitere Lösungen vor. Dabei erhalten Kanten nichtnegative Gewichte. Beide Lösungen reduzieren die gewichtete Summe aller Kantenlängen zwischen benachbarten Ebenen. Lange Kanten über mehrere Ebenen erhalten hohe Gewichte, um ein Platzieren dieser Kanten mit vielen Knicken zu vermeiden. Platzierungen im Sinne von vertikalen Segmenten, also eine gemeinsame x -Koordinate für eine Folge von Knoten, werden jedoch auch hier nicht garantiert.

6.6 Gesamtalgorithmus

In den Abschnitten 6.2 bis 6.5 wurden Algorithmen für die vier wesentlichen Schritte des VGL-Verfahrens untersucht, nun soll das Gesamtverfahren betrachtet werden. Dieses hat folgende Form:

Algorithmus: VERFAHREN FÜR GRÖSSENECHTE EBENENWEISE ZEICHNUNGEN MIT LAGEBEZIEHUNGEN (kurz VGL).

Gegeben: Geometrischer Graph mit Lagebeziehungen $G = (V, E, M, LB)$.

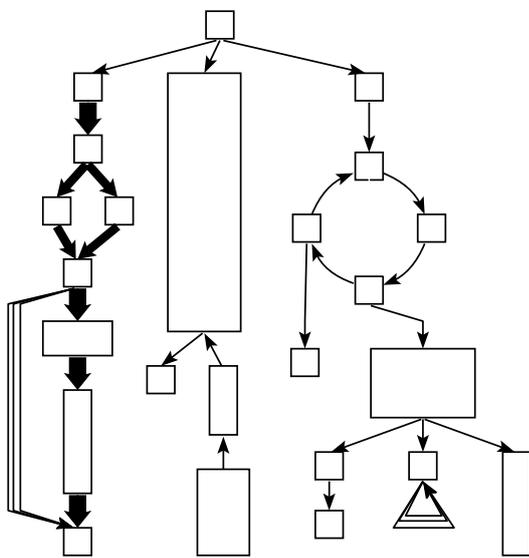
Gesucht: Platzierung $P = (P_V, P_E)$ und damit Zeichnung $Z(G, P)$ von G entsprechend den Konventionen und Ästhetikriterien in Abschnitt 6.1.4.

⁵⁷Sei M eine nichtleere Menge von Zahlen. Das *arithmetische Mittel* D der Menge M ist definiert als $D(M) = \frac{\sum_{m \in M} m}{|M|}$.

- 134 Vorverarbeitung;
- 135 Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG (Seite 127) ;
- 136 Algorithmus RASTERGESTÜTZTE EBENENEINTEILUNG (Seite 147) ;
- 137 Entfernung der für o - u -Lagebeziehungen zusätzlich eingefügten Kanten;
- 138 Algorithmus KREUZUNGSREDUZIERUNG (Seite 156) ;
- 139 Berechnung der x -Koordinaten (siehe Abschnitt 6.5);
- 140 Orientierung der Kanten in Originalrichtung; Beseitigung temporärer Knoten und Kanten;
- 141 Nachbearbeitung;

Die Vor- und Nachbearbeitung sind zwei zusätzliche Schritte, in denen der Graph auf vielfältige Weise verändert wird. Dabei werden beispielsweise parallele Kanten zu einer Kante vereinigt, komplexe Lagebeziehungen in einfache zerlegt, nicht zusammenhängende Graphen bearbeitet oder Gruppen von Knoten zusammengefasst. Diese und weitere Aspekte, z. B. die Behandlung von hor- und ver-Lagebeziehungen, sollen im folgenden Kapitel vorgestellt werden. Das Ergebnis der Vorverarbeitung ist ein einfacher Graph mit Lagebeziehungen, wie er in der Voraussetzung des Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG gefordert wird.

7 Erweiterungen des Zeichenverfahrens



7.1 Geometrische Aspekte	180
7.2 Allgemeine Graphen	188
7.3 Weitere Lagebeziehungen	190
7.4 Benennungen von Knoten und Kanten	194
7.5 Lokale Zeichenverfahren	196
7.6 Folgen von Zeichnungen	201

Darstellung eines Graphen, bei dessen Visualisierung einige der in diesem Kapitel betrachteten Erweiterungen verwendet wurden.

7.1 Geometrische Aspekte

Während die im vorherigen Kapitel beschriebenen Phasen die Grundlage des VGL-Verfahrens bilden, ergibt sich dessen praktische Handhabbarkeit erst durch vielfältige Anpassungen. Zuerst werden geometrische Aspekte von Knoten und Kanten betrachtet.

7.1.1 Knotenzentrum

In bisherigen Realisierungen des Sugiyama-Algorithmus wird das Knotenzentrum in der Knotenmitte angenommen. Durch eine außermittige Platzierung ergeben sich neue Freiheiten bei der Knotenplatzierung, die später z. B. für zusammengesetzte Knoten und für Kantenbenennungen verwendet werden. Statt einer Gesamtbreite werden rechte und linke Knotenbreiten b_r und b_l eingeführt, siehe Abbildung 7.1(a). Abbildung 7.1(b) zeigt eine Platzierung bezogen auf das Knotenzentrum. Damit die rechten und linken Knotenbreiten berücksichtigt werden, wird der Algorithmus zur Bestimmung der x -Koordinaten angepasst. Statt $\frac{b}{2}$ wird jetzt jeweils b_r bzw. b_l bei der Berechnung der Knotenabstände verwendet. In der Nachbearbeitung sind die Endpunkte der Kanten so zu setzen, dass eine gedachte Verlängerung jeder Kante das Knotenzentrum schneidet. Alternativ können Ports verwendet werden (siehe Abschnitt 7.1.5).

7.1.2 Individuelle Abstände zwischen Knoten

Bisher werden zwischen allen Knoten die gleichen Mindestabstände verwendet. Dies ist bei traditionellen ebenenweisen Zeichenverfahren schon durch die Verwendung größenmaximaler oder größenebenenmaximaler Platzierungen begründet.

Das hier vorgestellte Zeichenverfahren mit Berücksichtigung der Knotengröße lässt sich dagegen auf lokale, knotenbezogene Mindestabstände erweitern. So kann der Mindestabstand um einen Knoten z. B. von seiner Breite bzw. Höhe abhängen. Dadurch können um große Knoten große Abstände, um kleine Knoten kleine Abstände berücksichtigt werden (siehe Abbildung 7.2), was große Knoten in der Visualisierung stärker betont.

Zur Berücksichtigung lokaler Abstände sind zwei Änderungen nötig:

1. y -Koordinaten und Ebenenpartitionierung

Es reicht nun nicht mehr, die Mindestabstände wie in Abschnitt 6.3 beschrieben zur Knotenhöhe zuzufügen. Statt dessen erhält jeder Knoten einen oberen und einen unteren Rand.

Sei $a_{\min,v}$ der individuelle Mindestabstand um den Knoten v . Der untere Rand hat die Höhe $a_{\min,v}$. Der obere Rand kann um die Höhe des minimalen unteren Rands seiner Vorgänger reduziert werden. Der Grund dafür ist, dass die Mindestabstände zwischen zwei Knoten nicht addiert werden müssen, es genügt, das Maximum der beiden Mindestabstände zu betrachten.

Der danach für einen Knoten verbleibende obere Rand ist gegeben durch $a_o = \max(\{a_{\min,v} - a_{\min,u} \mid (u, v) \in E\} \cup \{0\})$. Im Algorithmus EBENENPARTITIONIERUNG bzw. RASTERGESTÜTZTE EBENENPARTITIONIERUNG berechnet sich $h[v]$ nun zu $h[v] = \text{Höhe}(v) + a_{\min,v} + a_o$. Für den oberen und unteren Rand jedes Knotens gilt:

$$y_o[v] = y(v) - \frac{h}{2} - a_o$$

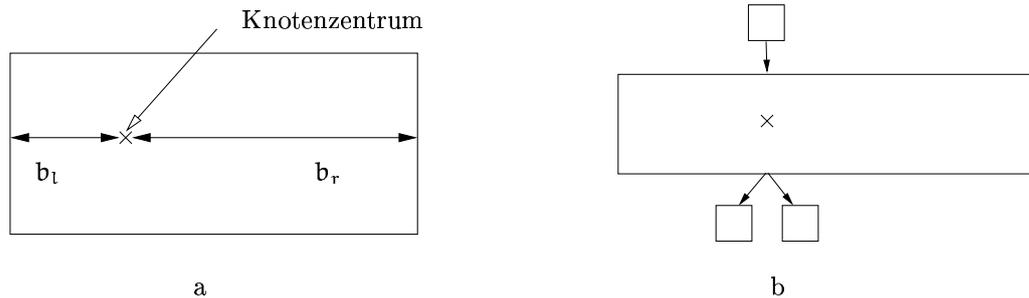


Abbildung 7.1: **Knotenzentrum.** (a) Rechte und linke Knotenbreiten bei Berücksichtigung eines Knotenzentrums, (b) Beispiel einer am Knotenzentrum orientierten Platzierung.

$$y_u[v] = y(v) + \frac{h}{2} + a_{\min,v}$$

Der Knoten wird also nicht nur um den unteren Mindestabstand, sondern auch um einen resultierenden oberen Mindestabstand vergrößert. Der Platzierungsalgorithmus ändert sich sonst nicht.

2. Berechnung der x-Koordinaten

Statt dem einheitlichen Mindestabstand zwischen Knoten sind hier individuelle Abstände zu berücksichtigen, wobei der horizontale Mindestabstand zwischen zwei Knoten das Maximum ihrer beiden individuellen Abstände ist. Die gängigen Algorithmen zur Bestimmung von x-Koordinaten lassen sich leicht entsprechend erweitern.

Dieses Konzept der individuellen Abstände zwischen Knoten lässt sich problemlos auf individuelle Abstände nach oben, unten, rechts und links für jeden Knoten erweitern.

7.1.3 Kantenhöhe und Kantenbreite

7.1.3.1 Kantenhöhe

Die Höhe $\text{Höhe}(e)$ einer Kante e lässt sich durch eine einfache Erweiterung des Algorithmus GRÖSSENECHTE PLATZIERUNG (Seite 141) berücksichtigen. Dazu wird die Zeile 53 ersetzt durch

$$y_o[v] := \max(\{y_u[u] + \text{Höhe}((u, v)) \mid u \in \text{Vor}(v)\} \cup \{0\});$$

Da bei der Berechnung von $y_u[u]$ der Mindestabstand a_{\min} bereits berücksichtigt wird, kann $\text{Höhe}(e)$ um a_{\min} verringert werden, solange danach $\text{Höhe}(e) > 0$ gilt.

Die Struktur des Graphen kann dazu führen, dass Kanten eine größere als ihre vorgegebene Höhe erhalten. Die hier betrachteten Kantenhöhen entsprechen also Mindesthöhen.

Kantenhöhen können beispielsweise bei der Darstellung von PERT-Diagrammen [DPTT89] verwendet werden. Ein PERT-Diagramm ist ein DAG, dessen Kanten mit Aufgaben eines Projekts und dessen Knoten mit Ereignissen in der Projektentwicklung assoziiert sind, beispielweise dem Start und der Beendigung von Teilaufgaben. Jede Kante hat ein Gewicht, das die erwartete Zeitdauer zur Lösung

der assoziierten Aufgabe angibt. Dieses Gewicht lässt sich durch eine entsprechende Höhe der Kante ausdrücken, siehe Abbildung 7.3(b).

7.1.3.2 Kantenbreite

Die Breite von Kanten wird in herkömmlichen ebenenweisen Zeichenverfahren nicht berücksichtigt. Die Berücksichtigung der Kantenbreite wird in der Praxis jedoch für verschiedene Visualisierungen benötigt. Ein Beispiel sind biochemische Reaktionsnetze, bei denen verschiedene Kantenbreiten die unterschiedliche Bedeutung von Reaktionen bzw. Reaktionswegen hervorheben.

Kantenbreiten werden wie folgt berücksichtigt: Bei langen, über mehrere Ebenen gehenden Kanten, erhalten nach der Ebenenpartitionierung die temporären Knoten als Knotenbreite die Breite der zugehörigen Kante. Dadurch wird die Kantenbreite bei der Platzierung berücksichtigt. Dieser Ansatz kann auch für die Berücksichtigung der Breite kurzer Kanten zwischen zwei Ebenen erweitert werden. Für diese Kanten werden temporäre Knoten mit der Kantenbreite als Knotenbreite eingefügt. Allerdings erfordert dies eine Wiederholung der Ebenenpartitionierung, da die neuen Knoten zu neuen Ebenen führen.

Da kurze Kanten in Visualisierungen biochemischer Reaktionsnetze bisher nicht zu Konflikten führten, wird für kurze Kanten in *BioPath* die Kantenbreite nicht berücksichtigt. Dadurch wird eine erneute Berechnung der Ebenenpartitionierung vermieden.

Abbildung 7.4 zeigt eine Visualisierung eines Graphen unter Berücksichtigung der Kantenbreite.

7.1.4 Kantenrouting

7.1.4.1 Splines

Die Kanten des Graphen lassen sich statt als Polylinien auch als Splines darstellen. Gansner et al. [GNV88, GKNV93] und Sanders [San96b] schlagen entsprechende Nachbearbeitungen der Kanten vor, bei denen die Kantenknicke als Stützpunkte für Splines betrachtet werden.¹ Diese Verfahren lassen sich auch hier anwenden.

7.1.4.2 Orthogonales Kantenrouting

Das von Sander beschriebene orthogonale Kantenrouting [San96a] ist dagegen nicht anwendbar. Voraussetzung für dieses Verfahren ist eine globale Ebeneneinteilung, um zwischen zwei Ebenen nachträglich Platz für horizontale Kantenstücke zu schaffen. Dazu werden ab Ebene L_i alle Knoten dieser und der nachfolgenden Ebenen gleichförmig nach unten verschoben.

Bei der hier verwendeten lokalen Ebeneneinteilung ist ein nachträgliches Verschieben von Ebenen jedoch nicht möglich. Große Knoten über mehrere Ebenen wären nach einem Verschieben von Ebenen nicht mehr korrekt platziert. Auch eine Neuberechnung der Ebenenpartitionierung nach dem Verschieben hilft nicht weiter. Durch eine Neuberechnung ändert sich die Ebeneneinteilung der Knoten, was im Allgemeinen wiederum zu einem neuen Kantenrouting führt. Es liegt also eine zyklische Abhängigkeit zwischen diesen Schritten vor.

¹Weitere Informationen zu Splines und das Gestalten des Berührungspunktes von Knoten und Kante finden sich im Buch von Foley et al. [FvFH90].

Dobkin et al. [DGKN97] stellen auch ein Verfahren vor, um Kanten nachträglich als Splines in die Zeichnung einzufügen, ohne die bisherigen Stützpunkte zu verwenden.

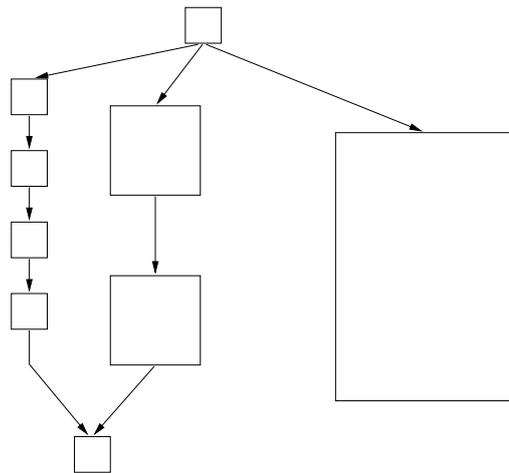


Abbildung 7.2: **Individuelle Abstände zwischen Knoten.** Platzierung der Knoten unter Verwendung individueller Abstände. Der Abstand um einen Knoten ist proportional zur Knotengröße, dadurch werden kleine Knoten dichter zueinander platziert, während um große Knoten mehr Abstand ist.

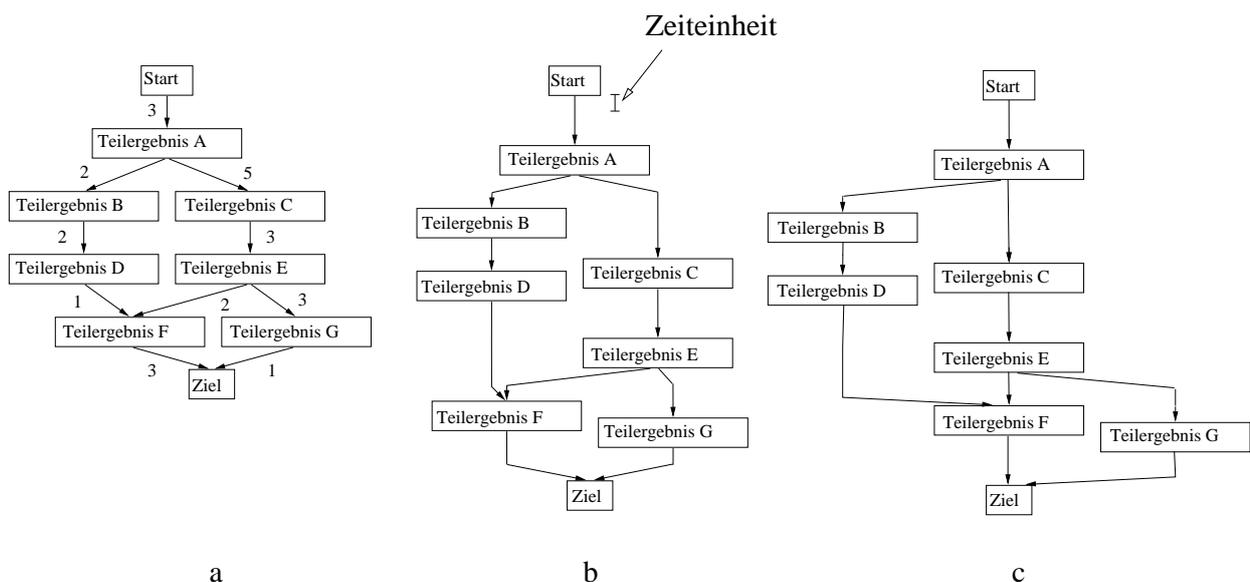


Abbildung 7.3: **Kantenhöhe.** (a) Darstellung eines PERT-Diagramms, die Kantenbenennung entspricht der nötigen Zeit zur Erledigung einer Teilaufgabe. (b) Visualisierung unter Berücksichtigung von Höhen für Kanten, der senkrechte Strich entspricht einer Zeiteinheit. Nun lässt sich direkt ablesen, welche Teilaufgaben zu einem bestimmten Zeitpunkt abgeschlossen sein müssen. (c) Berücksichtigung von Kantenhöhen und von ver-Lagebeziehungen zur Darstellung des kritischen Weges (siehe Abschnitt 7.3.3).

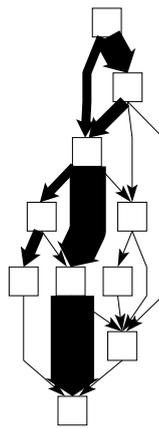


Abbildung 7.4: **Kantenbreite.** Darstellung eines Graphen unter Berücksichtigung von Kantenbreiten.

7.1.5 Ports

Oft wird vom Anwender gewünscht, den Berührungspunkt zwischen Kante und Knoten selbst festzulegen. Dieser Punkt wird als *Port* des Knotens bezeichnet. Im Gegensatz dazu wird ein Berührungspunkt, der nach der Platzierungsphase durch einen Algorithmus festgelegt wird, *Anker* genannt. Anker haben keinen Einfluss auf die Platzierung von Knoten, Ports dagegen beeinflussen deren Platzierung, siehe Abbildung 7.5.

Für das *BioPath*-Projekt war die Implementierung von Ports nicht nötig. Sie werden hier betrachtet, da sie eine interessante Erweiterung für verschiedene Anwendungen darstellen, z. B. bei der Visualisierung von Datenstrukturen [San96b, Wad01].

Ports können wie folgt berücksichtigt werden (siehe auch Abb. 7.6 und 7.7):

7.1.5.1 Obere Ports

Sei v_p ein Knoten mit Ports. In der Vorverarbeitung wird für jeden oberen Port in die zugehörige Kante ein temporärer Knoten eingefügt. Diese Knoten werden in der Reihenfolge der Ports durch l-r- und hor-Lagebeziehungen verbunden. hor-Lagebeziehungen sorgen dafür, dass die Knoten auf einer Ebene liegen, wodurch die l-r-Lagebeziehungen zwischen ihnen berücksichtigt werden. Die l-r-Lagebeziehungen übertragen die Reihenfolge der Ports auf die Reihenfolge der temporären Knoten. Alle Kanten werden so orientiert, dass sie von temporären Knoten zu oberen Ports verlaufen (siehe Abb. 7.6(b)). In der Nachbearbeitung werden die Kanten so platziert, dass sie den Knoten am Port berühren.

Für einen Knoten v_p mit Ports werden die Endknoten von ausgehenden Kanten, die an einem oberen Port ansetzen, neben den Knoten v platziert (in Abb. 7.6(c) z. B. der Knoten e). Die Platzierung dieser Knoten kann auch anders gestaltet werden, siehe Abbildung 7.6(d). Dabei werden alle zu oberen Ports inzidenten Kanten so orientiert, dass sie zum Port hinführen. Eine weitere Möglichkeit wäre die Verwendung von o-u-Lagebeziehungen, wodurch in Abbildung 7.6(c) beispielsweise der Knoten e mittels der Lagebeziehung $(i, e, o-u, g)$ unterhalb des Knotens i platziert werden kann.

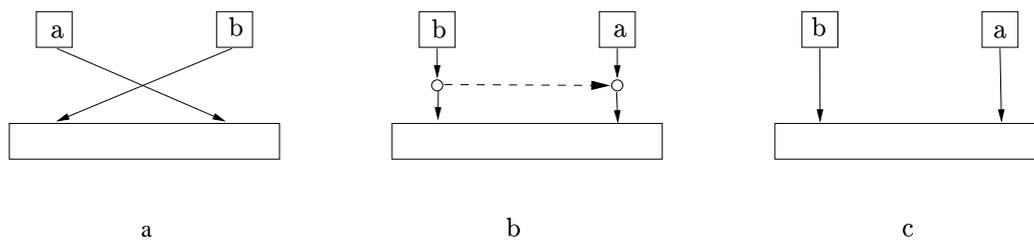


Abbildung 7.5: **Ports.** Ports haben Einfluss auf die Platzierung von Knoten. (a) Ein Graph, dessen unterer Knoten zwei Ports für Kanten besitzt. Die Darstellung zeigt eine mögliche, durch das VGL-Verfahren berechnete Platzierung. Diese Visualisierung hat eine Kreuzung. Der Algorithmus KREUZUNGSREDUZIERUNG kann hier keine Kantenkreuzung feststellen, für ihn beginnen beide Kanten in der Knotenmitte. (b) Verwendung temporärer Knoten mit l-r-Lagebeziehung entsprechend der Reihenfolge der Ports und hor-Lagebeziehung (beide gestrichelt als gemeinsame Kante), dadurch kreuzungsreduzierende Anordnung der Knoten a und b. (c) Visualisierung nach Ende des Gesamtalgorithmus.

7.1.5.2 Untere Ports

Untere Ports werden analog zu oberen Ports behandelt.

7.1.5.3 Seitliche Ports

Sei v_P wieder ein Knoten mit Ports. Seitliche Ports werden in obere und untere Ports umgewandelt und dann wie diese behandelt. Im Folgenden werden Ports auf der linken Seite eines Knotens betrachtet, die Behandlung rechtsseitiger Ports erfolgt analog.

Nach der Kantenorientierung wird der Knoten v_P vergrößert, um die seitlichen Teilstücke, die von oben zu seitlichen Ports führen oder von seitlichen Ports nach unten verlaufen, in den Knoten zu integrieren (siehe Abbildung 7.7(b)). Die innerhalb des vergrößerten Knotens liegenden Kantenstücke werden als *innere Kanten* bezeichnet.

Für die seitlichen Ports werden obere und untere Ports eingefügt. Dabei verlaufen eingehende Kanten des Knotens v_P zu oberen Ports, ausgehende Kanten von v_P gehen von unteren Ports ab. Die Reihenfolge der oberen und unteren Ports, und damit die Zahl der Kreuzungen zwischen inneren Kanten, lässt sich mit Hilfe des Algorithmus KREUZUNGSREDUZIERUNG wie folgt bestimmen (siehe dazu Abb. 7.8):

Die Ports der linken Seite eines Knotens v werden auf der Ebene L_1 einer 2-LR-Ebenenpartitionierung $G = (L_1 \cup L_2, E)$ platziert. Die zu den Ports gehörenden adjazenten Knoten von v_P werden auf der Ebene L_2 platziert. Sei E_{ein} die Menge der linksseitig eingehenden Kanten von v_P und V_{ein} die Menge der Anfangsknoten der Kanten aus E_{ein} . Sei E_{aus} die Menge der linksseitig ausgehenden Kanten von v_P mit der Menge zugehöriger Endknoten V_{aus} . Es werden l-r-Lagebeziehungen von allen Knoten in V_{ein} zu allen Knoten in V_{aus} eingefügt. Der Algorithmus KREUZUNGSREDUZIERUNG bestimmt nun eine Anordnung der Knoten in Ebene L_2 , die zu möglichst wenigen Kreuzungen innerer Kanten führt. Dabei wird durch die l-r-Lagebeziehungen die Unterteilung in obere und untere

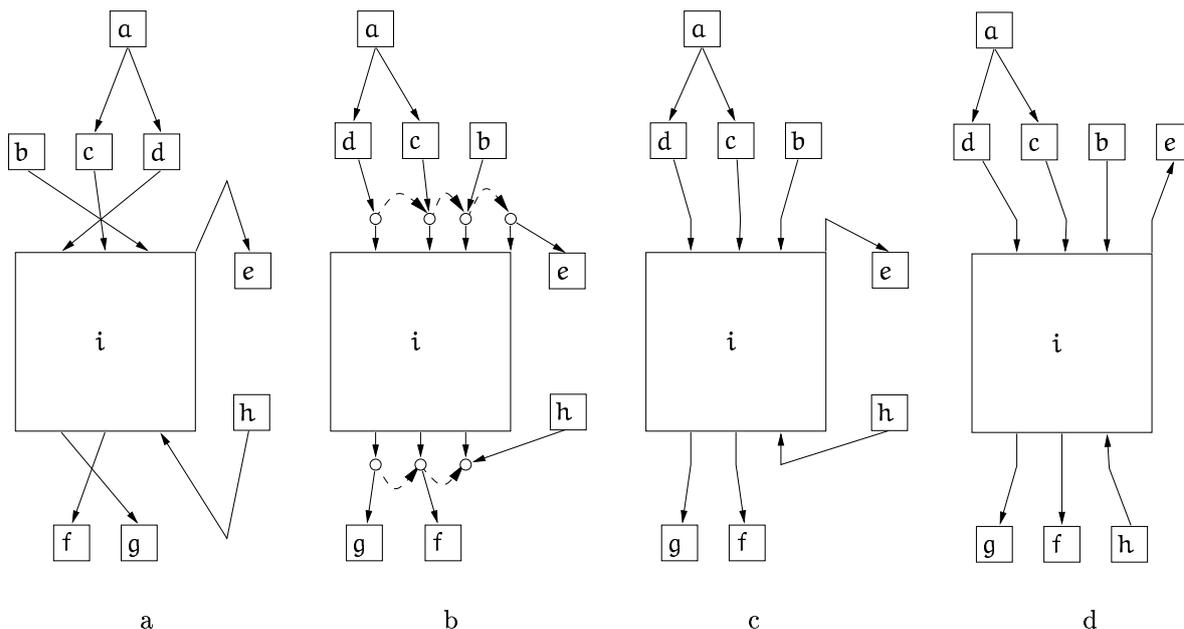


Abbildung 7.6: **Obere und untere Ports.** (a) Graph, der Knoten i hat obere und untere Ports. (b) Realisierung der Ports durch temporäre Knoten, die gestrichelten Pfeile geben die gemeinsamen l-r- und hor-Lagebeziehungen an. (c) Endgültige Visualisierung und (d) Visualisierung, falls die Kanten (i, e) und (h, i) temporär umgedreht werden.

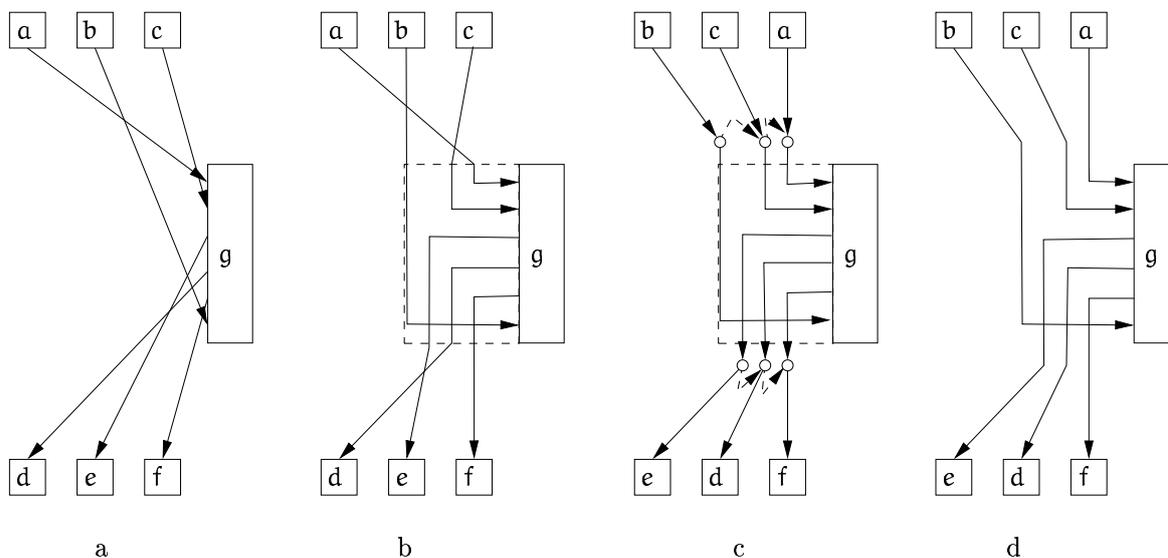


Abbildung 7.7: **Seitliche Ports.** (a) Graph, der Knoten g hat seitliche Ports. (b) Realisierung der Ports durch Vergrößerung des Knotens und Umwandlung der seitlichen in obere und untere Ports. Die Bestimmung der Reihenfolge der oberen und unteren Ports wird in Abbildung 7.8 betrachtet. (c) Nun wird das Verfahren für obere und untere Ports angewandt (Einfügen temporärer Knoten, gestrichelt wieder die gemeinsamen l-r- und hor-Lagebeziehungen, siehe auch Abb. 7.6). (d) Die endgültige Visualisierung des Graphen.

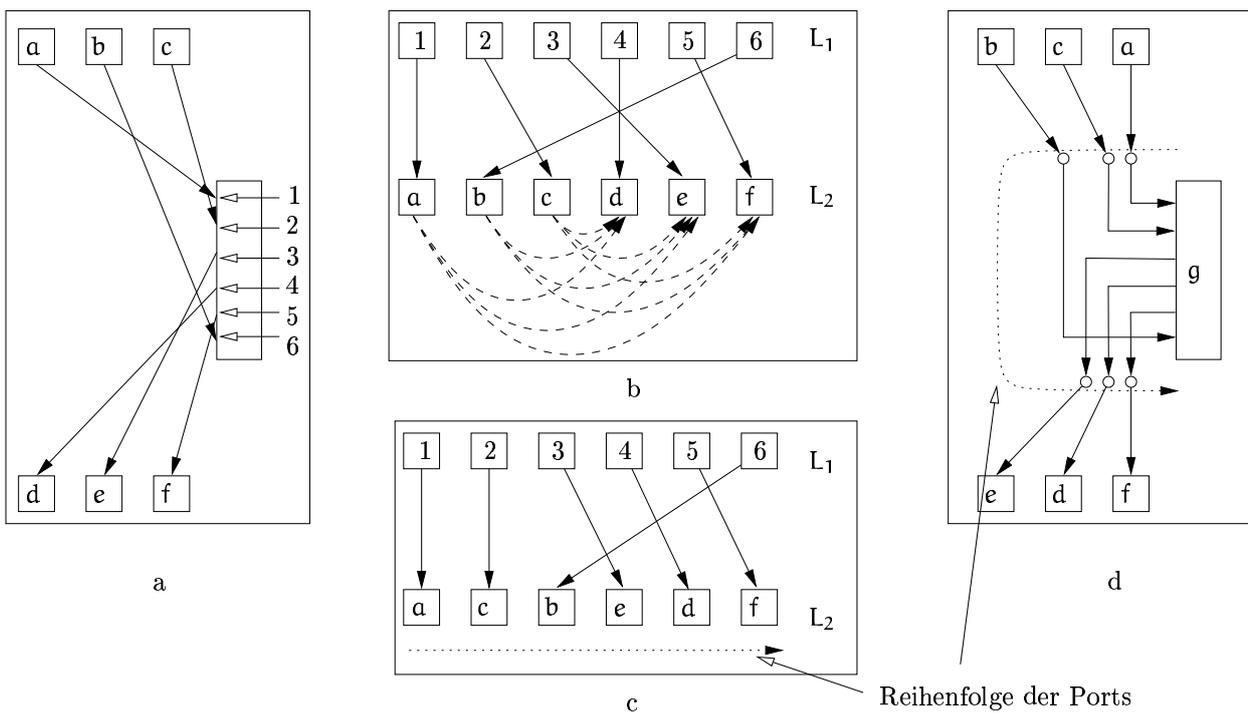


Abbildung 7.8: **Reihenfolge der Ports.** (a) Graph aus Abbildung 7.7. (b) Bestimmung der Reihenfolge der oberen und unteren Ports: Füge in L_1 für jeden Port einen Knoten, in L_2 die zugehörigen Anfangs- bzw. Endknoten ein. Alle Kanten verlaufen von L_1 nach L_2 . Wie in Abschnitt 7.1.5.3 beschrieben, werden l-r-Lagebeziehungen eingefügt, diese sind gestrichelt dargestellt. (c) Der Algorithmus KREUZUNGSREDUZIERUNG bestimmt eine Anordnung der Knoten in L_2 , die zu möglichst wenigen Kreuzungen führt. (d) Übertrage diese Reihenfolge als Anordnung der oberen und unteren Ports. Der gepunktete Pfeil verdeutlicht die Reihenfolge in (c) und (d), die lineare Folge aus (c) wird also zwischen den oberen und unteren Ports aufgeteilt.

Ports berücksichtigt.

Die so berechneten oberen und unteren Ports werden entsprechend den Verfahren in den Abschnitten 7.1.5.1 und 7.1.5.2 behandelt. In der Nachbearbeitung werden für temporäre Knoten Stützpunkte in die zugehörigen Kanten eingefügt, siehe Abbildung 7.7(d).

7.1.5.4 Weitere Verfahren zur Berücksichtigung von Ports

Auch andere Autoren betrachten Ports, jedoch berücksichtigen sie die Reihenfolge der Ports bei der Platzierung der adjazenten Knoten nicht oder nur für eine eingeschränkte Menge von Kanten. So verwendet Waddle [Wad01] eine Erweiterung des Barycenter-Algorithmus zur Berücksichtigung von Ports. Für untere Ports werden dazu statt der relativen Anordnung der Knoten die x -Koordinaten der Ports verwendet. Bei seitlichen Ports haben alle Knoten dieselbe x -Koordinate. Hier wird deshalb ein sekundärer Sortierschlüssel verwendet, der so initialisiert wurde, dass Knoten mit wenigen

Kreuzungen angeordnet werden. Dabei betrachtet Waddle nur seitliche Ports, deren Kanten nach unten verlaufen, jedoch keine von oben kommenden Kanten.

Gansner et al. [GKNV93] und Sanders [San95] betrachten ebenfalls Ports in ebenenweisen Zeichenverfahren. Die durch die Ports gegebene Reihenfolge der Kanten wird jedoch bei der Kreuzungsreduzierung nicht berücksichtigt. Dadurch kann es zu unnötigen Kantenkreuzungen kommen.

Insgesamt stellen traditionelle ebenenweise Zeichenverfahren Ansätze zur Behandlung von Ports zur Verfügung, die jedoch schlechtere Ergebnisse als das hier beschriebene Verfahren liefern.

7.2 Allgemeine Graphen

7.2.1 Reflexive Kanten

Reflexive Kanten führen dazu, dass ein Graph nicht azyklisch ist und mit dem üblichen Mechanismus des Umdrehens ausgewählter Kanten auch nicht azyklisch gemacht werden kann. Reflexive Kanten werden deshalb in der Vorverarbeitung behandelt, wobei sich zwei Ansätze unterscheiden lassen (siehe Abb. 7.9):

1. Jede reflexive Kante $(u, u) \in E$ wird durch drei Knoten v_1, v_2, v_3 und Kanten $(u, v_1), (u, v_2), (v_1, v_3), (v_2, v_3)$ ersetzt, siehe Abbildung 7.9(b). Da keine anderen Kanten zu den Knoten v_1, v_2, v_3 führen, platziert die Ebenenpartitionierung diese Knoten auf die nächsten Ebenen. In der Nachbearbeitung werden in der reflexiven Kante Stützpunkte statt der Knoten v_1, v_2, v_3 eingefügt.
2. Jede reflexive Kante $(u, u) \in E$ wird seitlich oder unterhalb des Knotens platziert, der Knoten wird entsprechend vergrößert, siehe Abbildung 7.9(c). In der Vorverarbeitung wird dazu die Kante aus dem Graph entfernt und der entsprechende Knoten vergrößert, in der Nachbearbeitung wird die Kante wieder eingefügt und der Knoten auf Originalgröße reduziert.

Der zweite Ansatz hat mehrere Vorteile:

1. Es werden keine weiteren Knoten in den Graph eingefügt. Dies ist für die Rechenzeit des Gesamtverfahrens positiv.
2. Reflexive Kanten können verschieden dargestellt werden, z. B. als Kreisbogen oder auf einer beliebigen Seite des Knotens.
3. Die Darstellung mehrerer reflexiver Kanten kann kompakter gestaltet sein (siehe Abb. 7.10).

Da der VGL-Algorithmus beliebige Knotengrößen berücksichtigt, wurde der zweite Ansatz realisiert.

7.2.2 Parallele und antiparallele Kanten

Graphen können so definiert werden, dass mehrere Kanten von Knoten u zu Knoten v möglich sind. Diese Kanten werden als *parallele Kanten* oder *Mehrfachkanten* bezeichnet. In einem Graph $G = (V, E)$ können *entgegengesetzte* Kanten $(u, v) \in E$ und $(v, u) \in E$ auftreten. Diese werden auch als *antiparallele* Kanten bezeichnet.

Ähnlich zur Behandlung reflexiver Kanten können parallele und antiparallele Kanten ebenfalls auf zwei Arten behandelt werden (siehe Abbildung 7.11):

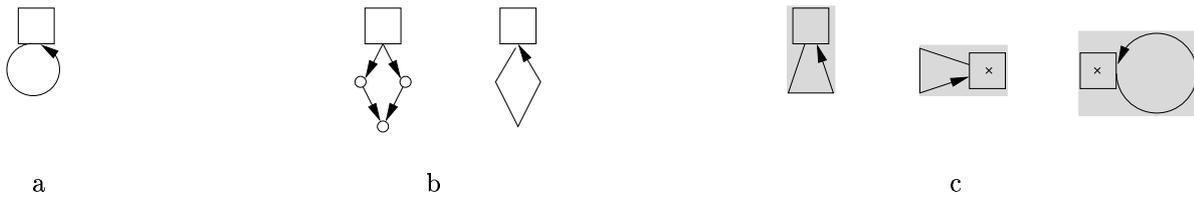


Abbildung 7.9: **Behandlung reflexiver Kanten.** (a) Reflexive Kante, (b) und (c) zwei Ansätze zur Behandlung reflexiver Kanten: (b) Ersetzen der Kante durch drei Knoten und zugehörige Kanten, dabei ist links die Ersetzung der Kante durch Knoten, rechts das Resultat der Visualisierung dargestellt. (c) Die Vergrößerung des inzidenten Knotens. Dabei gibt es verschiedene Möglichkeiten für die Darstellung der reflexiven Kante. Der vergrößerte Knoten ist grau dargestellt. Das neue Knotenzentrum ist mit einem Kreuz markiert. Durch das Knotenzentrum werden weitere inzidente Kanten korrekt zum Knoten ausgerichtet.

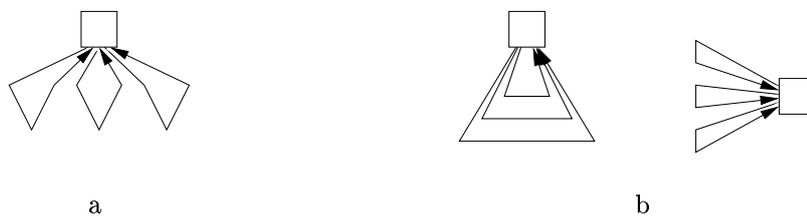


Abbildung 7.10: **Mehrfache reflexive Kanten.** Darstellung mehrfacher reflexiver Kanten durch (a) den ersten Ansatz und (b) den zweiten Ansatz (mit zwei Beispielen).

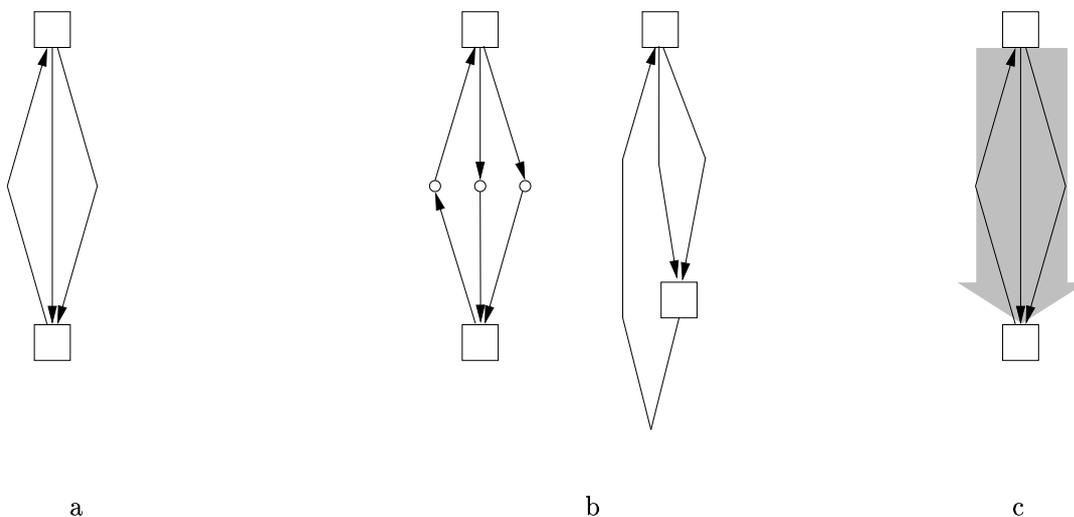


Abbildung 7.11: **Behandlung paralleler und antiparalleler Kanten.** (a) Parallele und antiparallele Kanten, (b) und (c) zwei Ansätze zur Behandlung dieser Kanten: (b) Ersetzen jeder Kante durch einen Knoten und zugehörige Kanten, dabei ist links der entstehende Graph, rechts das Resultat der Visualisierung dargestellt. Durch die Kantenorientierung mit Beseitigung der Zyklen wurde die Kante von unten nach oben wie abgebildet platziert. (c) Zusammenfassen paralleler und antiparalleler Kanten zu einer Kante, grau die zusammengefasste Kante. Die Darstellung zeigt zugleich das Ergebnis der Visualisierung.

1. In der Vorverarbeitung wird für jedes Knotenpaar u, v mit parallelen oder antiparallelen Kanten jede Kante (u, v) bzw. (v, u) durch einen Knoten v_1 und Kanten $(u, v_1), (v_1, v)$ bzw. $(v, v_1), (v_1, u)$ ersetzt, siehe Abbildung 7.11(b). In der Nachbearbeitung werden diese Knoten zu zusätzlichen Stützpunkten der ursprünglichen Kanten.
2. In der Vorverarbeitung wird für jedes Knotenpaar u, v mit parallelen oder antiparallelen Kanten die Menge aller Kanten (u, v) und (v, u) zu einer Kante e zusammengefasst. Diese Kante erhält als Breite die Ursprungsbreite der Kanten und die Mindestabstände zwischen den Kanten. Dabei entscheidet, falls vorhanden, eine o-u-Lagebeziehung über die Richtung der Kante e . Falls keine o-u-Lagebeziehung vorhanden ist, wird für e die Richtung gewählt, in der mehr Kanten verlaufen. In der Nachbearbeitung wird die Kante wieder durch die originalen Kanten ersetzt, siehe Abbildung 7.11(c).

Der zweite Ansatz hat gegenüber dem ersten zwei Vorteile:

1. Es werden keine weiteren Knoten in den Graph eingefügt.
2. Antiparallele Kanten werden wie die zugehörigen parallelen Kanten dargestellt, während die Kantenführung für antiparallele Kanten im ersten Ansatz von der Kantenführung der parallelen Kanten verschieden ist, vergleiche Abbildung 7.11(b) mit 7.11(c).

Aus diesen Gründen und da das Zeichenverfahren Kantenbreiten berücksichtigt, wurde der zweite Ansatz realisiert.

7.2.3 Nicht zusammenhängende Graphen

Nicht zusammenhängende Graphen werden oft dadurch behandelt, dass die einzelnen Zusammenhangskomponenten separat bearbeitet und anschließend nebeneinander platziert werden. Die Teilalgorithmen des VGL-Verfahrens sind so gestaltet, dass sie auch für nicht zusammenhängende Graphen korrekte Ergebnisse liefern.

Für die Kreuzungsreduzierung hat es sich dabei als vorteilhaft erwiesen, wenn die schwachen Zusammenhangskomponenten K_1, \dots, K_z in der Vorverarbeitung von links nach rechts sortiert werden und dann von jedem Knoten der Komponente K_i zu jedem Knoten der Komponente K_{i+1} , $1 \leq i < z$ eine l-r-Lagebeziehung eingefügt wird.

7.3 Weitere Lagebeziehungen

7.3.1 Gewichtete Kreuzungszahl

In Abschnitt 6.4 wurde bereits erwähnt, dass es sinnvoll sein kann, Kreuzungen unterschiedlich zu gewichten. So kann die Kreuzung einer Kante zwischen Repräsentanten eines großen Knotens mit einer normalen Kante höher als eine Kreuzung zwischen zwei normalen Kanten gewichtet werden. Dadurch können Kreuzungen zwischen Kanten und großen Knoten vermieden werden. Außerdem kann die Gewichtung einer Kreuzung von der Breite der sich kreuzenden Kanten oder davon abhängig sein, ob parallele Kanten daran beteiligt sind.

Als allgemeiner Ansatz wird für jede Kante ein Kreuzungsgewicht $KG : E \rightarrow \mathbb{N}$ eingeführt. Je höher das Kreuzungsgewicht einer Kante ist, umso stärker sollen Kreuzungen mit dieser Kante vermieden



Abbildung 7.12: **Gewichtete Kreuzungszahl.** Eine breite Kante e_b soll sich möglichst wenig mit anderen Kanten kreuzen. (links) Bei fester Anordnung der Knoten in L_1 ist hier die Gesamtzahl der Kreuzungen minimal, jedoch kreuzt e_b vier andere Kanten. Erhalten die Kanten Kreuzungsgewichte mit $KG(e_b) = 3$ und $KG(e) = 1$ für alle anderen Kanten e , so beträgt die gewichtete Kreuzungszahl 18. (rechts) Nun ist eine Anordnung mit kleinerer gewichteter Kreuzungszahl möglich, auch wenn die Gesamtzahl der Kreuzungen größer ist.

werden. Im *BioPath*-Projekt wird als Kreuzungsgewicht einer Kante ihre Breite bzw. die Breite des großen Knotens, falls die Kante zwei Repräsentanten eines großen Knotens verbindet, gewählt. Als Gewichtung der Kreuzung zweier Kanten wird die Summe der Kreuzungsgewichte der beiden Kanten berücksichtigt. Eine Kreuzung zwischen e_1 und e_2 zählt dann nicht mehr als eine Kreuzung, sondern als $KG(e_1) + KG(e_2)$ Kreuzungen. Hier sind auch andere Arten der Gewichtung denkbar, z. B. $KG(e_1) * KG(e_2)$. Die Abbildung 7.12 zeigt die Anwendung der gewichteten Kreuzungszahl.

Um *l-r*-Lagebeziehungen zwischen Knoten weiterhin zu berücksichtigen, muss deren Gewicht erhöht werden. Dazu wird in einem Präprozess der Kreuzungsreduzierung die maximale gewichtete Kreuzungszahl k_{\max} zwischen Kanten bestimmt. Lagebeziehungen der Form $(u, v, l-r, g)$ werden dann umgewandelt in $(u, v, l-r, g + k_{\max})$.

7.3.2 hor-Lagebeziehungen

o-u- und *l-r*-Lagebeziehungen wurden bereits in Kapitel 6 betrachtet, im Folgenden wird die Realisierung von *hor*- und *ver*-Lagebeziehungen untersucht.

Knoten $u, v \in V$, für die eine Lagebeziehung der Form $(u, v, hor, g) \in LB$ existiert, werden vor der Kantenorientierung (Zeile 135 auf Seite 178) zu einem Knoten zusammengefasst. Dadurch wird sichergestellt, dass sie später auf dieselbe Ebene platziert werden. Nach der Ebenenpartitionierung (Zeile 136) werden die Knoten wieder getrennt.

Falls zwischen zwei Knoten $u, v \in V$ mit $(u, v, hor, g) \in LB$ eine Kante existiert, so wird auf dieser Kante ein temporärer Knoten eingefügt. Dieser Knoten wird durch das VGL-Verfahren auf die nächste Ebene platziert. Dadurch wird garantiert, dass Kanten nur zwischen Ebenen, aber nicht innerhalb einer Ebene, verlaufen. Alternativ können auch *hor*-Lagebeziehungen zwischen adjazenten Knoten verboten werden.

Die Idee, Knoten durch temporäres Zusammenfassen auf derselben Ebene zu platzieren, findet sich auch in [GKNV93, San96b, Wad01].

7.3.3 ver-Lagebeziehungen

Lagebeziehungen der Form (u, v, ver, g) werden durch vertikale Segmente realisiert. Dazu müssen die Knoten u und v durch eine Kante verbunden sein. Falls nötig wird in der Vorverarbeitung eine temporäre Kante mit Kreuzungsgewicht 0 eingefügt. Diese Kante wird dadurch bei der Berechnung von Kantenkreuzungen ignoriert.

Jeder Knoten kann nur eine *ver*-Lagebeziehung erfüllen, sind mehrere *ver*-Lagebeziehungen für einen Knoten vorhanden, so wird die mit dem höchsten Gewicht verwendet. Die durch eine *ver*-Lagebeziehung verbundenen Knoten und die zugehörige Kante werden während der Kreuzungsreduzierung wie große Knoten und während der Bestimmung von x -Koordinaten wie lineare Segmente behandelt.

Eine Anwendung für *ver*-Lagebeziehungen sind Visualisierungen der bereits erwähnten PERT-Diagramme. Als *kritischer Pfad* eines solchen Diagramms wird der längste Pfad vom Anfangs- zum Endknoten des Diagramms bezeichnet. Werden Aufgaben auf diesem Pfad nicht fristgerecht gelöst, so bewirkt dies eine Verzögerung des Gesamtprojekts. Di Battista et al. empfehlen, die Knoten des kritischen Pfads auf einer vertikalen Linie zu platzieren [DPTT89], dies lässt sich hier einfach durch *ver*-Lagebeziehungen realisieren. Abbildung 7.3(c) stellt ein PERT-Diagramm unter Verwendung von Mindesthöhen für die Zeitdauer einer Teilaufgabe und *ver*-Lagebeziehungen zur vertikalen Anordnung der Teile des kritischen Pfads dar.

7.3.4 Globale l-r-Lagebeziehungen

Bisher wurden nur so genannte *lokale* *l-r*-Lagebeziehungen betrachtet. Dabei handelt es sich um *l-r*-Lagebeziehungen zwischen Knoten, die eine gemeinsame y -Koordinate besitzen. Die Gründe dafür wurden bereits im Abschnitt 6.1.3 erörtert.

Manchmal werden jedoch *globale* *l-r*-Lagebeziehungen vom Anwender gefordert, also links-rechts Beziehungen zwischen Knoten unabhängig von deren y -Koordinate. Auch diese lassen sich im VGL-Verfahren realisieren.

Sei $(u, v, \text{l-r}, g)$ eine globale *l-r*-Lagebeziehung zwischen den Knoten u und v , wobei die Knoten keine gemeinsame y -Koordinate besitzen (siehe Abb. 7.13(a)). Nach der Ebeneneinteilung wird für die *l-r*-Lagebeziehung ein großer Knoten v_g eingefügt. Dieser Knoten geht über alle Ebenen zwischen u und v , auf jeder Ebene existiert also ein Repräsentant $v_i \in R(v_g)$ des großen Knotens. Statt $(u, v, \text{l-r}, g)$ werden Lagebeziehungen $(u, v_g, \text{l-r}, g)$ und $(v_g, v, \text{l-r}, g)$ eingefügt, siehe Abbildung 7.13(b). Die Breite von v_g bzw. seinen Repräsentanten und das Kreuzungsgewicht der Kanten zwischen den Repräsentanten ist 0. Dadurch beeinflusst v_g die Kreuzungsreduzierung nicht. Bei der Bestimmung der x -Koordinaten werden die Repräsentanten von v_g als vertikales Segment betrachtet, dadurch wird Knoten u links von v_g und Knoten v rechts von v_g platziert. Abbildung 7.13(d) zeigt die endgültige Platzierung des Graphen aus Abbildung 7.13(a).

7.3.5 Berücksichtigung weiterer Ebeneneinteilungen

Im Abschnitt 6.3.4 wurden verschiedene traditionelle Verfahren zur Ebeneneinteilung beschrieben. Diese können auch im VGL-Algorithmus berücksichtigt werden. Ein Beispiel für verschiedene Zeichnungen eines Graphen auf Grund unterschiedlicher Ebeneneinteilungen zeigt Abbildung 7.14.

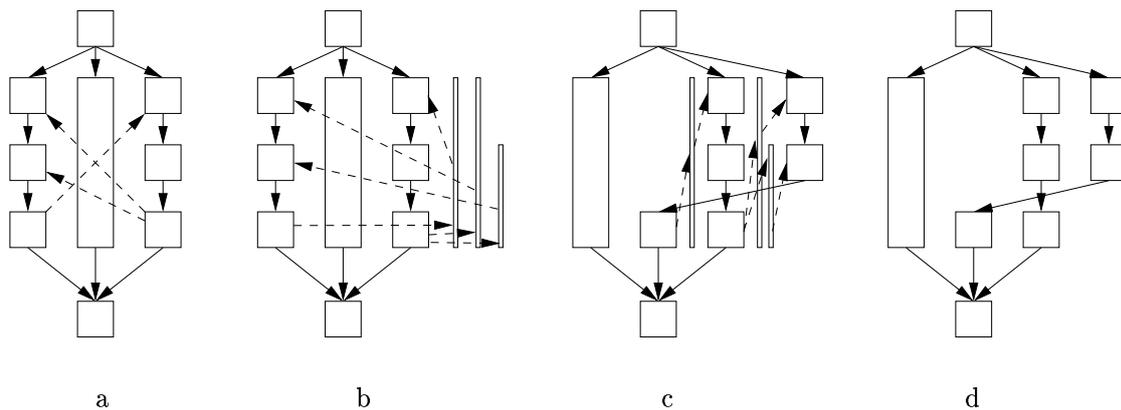


Abbildung 7.13: **Globale l-r-Lagebeziehungen.** (a) Drei globale l-r-Lagebeziehungen sind gestrichelt dargestellt. (b) Für jede globale l-r-Lagebeziehung $(u, v, l-r, g)$ werden ein großer Knoten v_g und Lagebeziehungen $(u, v_g, l-r, g)$, $(v_g, v, l-r, g)$ eingefügt. (c) Platzierung mit den großen Knoten und (d) endgültige Visualisierung.

Für den VGL-Algorithmus ist dazu eine Erweiterung nötig: Zwischen der Kantenorientierung (Zeile 135 auf Seite 178) und der Ebenenpartitionierung (Zeile 136) werden folgende Schritte eingefügt:

1. Berechne eine Ebeneneinteilung L_1, \dots, L_l ohne Berücksichtigung der Knotengröße
2. Für $1 \leq i < l$ füge für jeden Knoten der Ebene L_i zu jedem Knoten der Ebene L_{i+1} eine o-u-Lagebeziehung ein.² Dies garantiert den Erhalt der Ebeneneinteilung im VGL-Verfahren.

Das Ergebnis dieses Verfahrens unterscheidet sich nicht mehr vom Ergebnis der Ebeneneinteilung im herkömmlichen *Sugiyama*-Verfahren, es liefert eine grössenebenenmaximale Platzierung.

Es ist jedoch nicht nötig, die komplette Ebeneneinteilung zu fixieren. Man kann auch nur für Teilgraphen eine Sortierung festschreiben. Dadurch wird der restliche Graph wie im bisherigen VGL-Verfahren behandelt. Ein Beispiel für diesen Ansatz ist die Platzierung aller Wurzelknoten nicht auf der obersten Ebene (siehe Abb. 7.15(a)), sondern so, dass Kanten von Wurzelknoten zu deren Nachfolgern möglichst kurz werden (siehe Abb. 7.15(b)). Dazu wird eine Ebeneneinteilung des längsten Pfads³ von den Blättern aus berechnet. Anschließend werden o-u-Lagebeziehungen nur dann vom Knoten u zum Knoten v eingefügt, wenn v eine Wurzel ist (siehe Abb. 7.15(c)).

Da die traditionelle Ebeneneinteilung die Höhe der Knoten nicht berücksichtigt, vergrößert sich bei diesem Vorgehen im Allgemeinen die Gesamthöhe der Zeichnung (vergleiche auch Abb. 7.15).

²Geht man streng nach dem VGL-Algorithmus vor, so werden nur o-u-Lagebeziehungen berücksichtigt, wenn diese vor der Kantenorientierung in den Graphen eingefügt wurden und nicht wie hier erst danach. Hier wird deshalb statt einer o-u-Lagebeziehung in den Graph eine entsprechende Kante eingefügt. Diese wird als o-u-Lagebeziehung markiert, so dass sie in Zeile 137 (Seite 178) des VGL-Algorithmus wieder korrekt aus dem Graph entfernt werden kann.

³Siehe [DETT99], hier wird zur Berechnung dieser Ebeneneinteilung die Richtung aller Kanten temporär umgedreht.

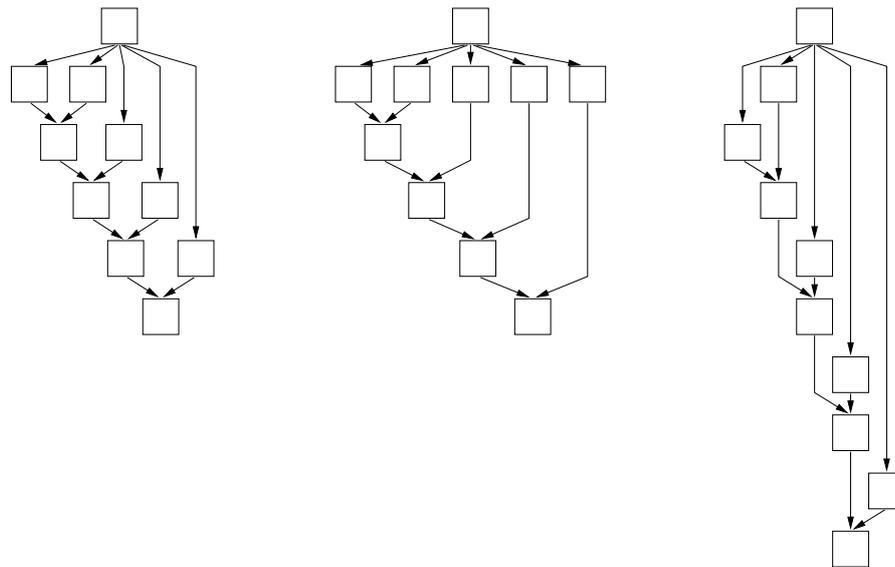


Abbildung 7.14: **Verwendung herkömmlicher Ebeneneinteilungen.** Verschiedene Darstellungen eines Graphen auf der Grundlage unterschiedlicher Ebeneneinteilungen.

7.4 Benennungen von Knoten und Kanten

7.4.1 Einführung

Beim Zeichnen benannter Graphen ist es nötig, die Knoten- und Kantenbenennung sinnvoll zu platzieren. Es werden Positionen für die Benennungen gesucht, die sich gut in die Zeichnung einfügen, die nicht zu Überschneidungen mit anderen Objekten führen und die zugleich das Verstehen der Abbildung fördern, indem sie Benennungen eindeutig ihren Knoten und Kanten zuordnen.

Das Problem des Findens geeigneter Positionen für Benennungen von Objekten kommt ursprünglich aus der Kartographie [Imh62, Yoe72]. Dabei ist das Ziel, räumlich festgelegte Elemente wie Städte, Straßen und Gebiete; also Knoten, Kanten bzw. Polylinien und Flächen mit ihren Namen zu versehen. In der Kartografie ist die Lage der Objekte fixiert. Oft wird zwischen Benennungen von Knoten und von Kanten unterschieden. Für Knotenbenennungen stehen meist nur vier an den Knoten angrenzende Positionen, z. B. oben, unten, links und rechts, zur Verfügung. Aus diesen ist eine so auszuwählen, dass möglichst keine Kollisionen mit anderen verwendeten Positionen auftreten. Bei Kanten ist die mögliche Zahl von Positionen meist größer. Auch hier ist für jede Kante eine Position auszuwählen, die möglichst nicht zu Konflikten führt.

Beide Probleme sind NP-vollständig [FW91, MS91a, KT01]. Das Benennungsproblem lässt sich in einer allgemeinen Form wie folgt definieren:

DEFINITION 7.1 (BENENNUNGSPROBLEM)

Sei O eine endliche Menge von Objekten mit festen Koordinaten und zugehörigen Namen. Jedem Objekt $o \in O$ ist eine endliche Menge Pos von möglichen Namenspositionen zugeordnet. Dabei

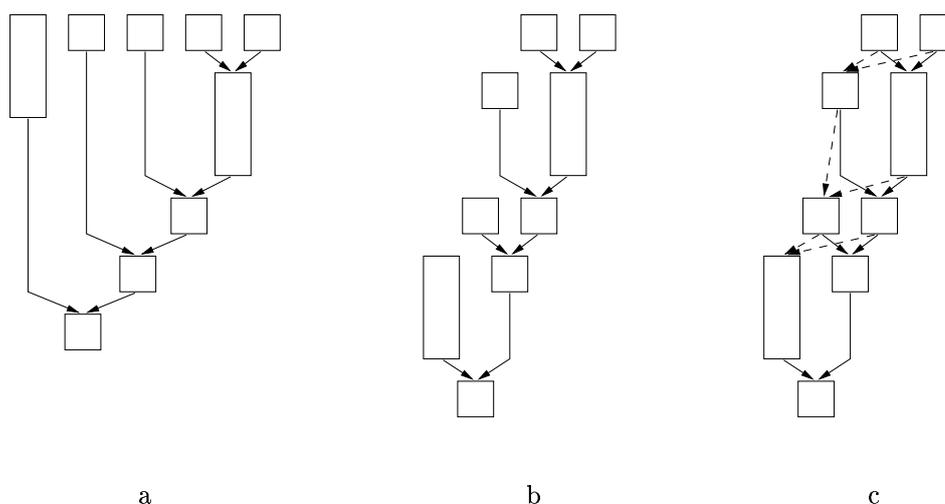


Abbildung 7.15: **Verwendung teilweiser Ebeneneinteilungen.** Visualisierung eines Graphen: (a) Entsprechend dem VGL-Verfahren, (b) unter Verwendung einer traditionellen Ebeneneinteilung und teilweiser Fixierung der Ebeneneinteilung für alle Wurzelknoten, (c) gestrichelt die dafür eingefügten o-u-Lagebeziehungen zu Wurzelknoten.

dürfen sich Namenspositionen gegenseitig beliebig überdecken, sie dürfen jedoch keine Knoten und Kanten überdecken.

Gesucht ist für den Namen eines jeden Objekts eine Position $p \in \text{Pos}$, so dass die Zahl der Überdeckungen von Namen minimal ist.

Überdeckungen von Namen werden auch als *Konflikte* bezeichnet. Teilweise werden die Positionen unterschiedlich stark gewichtet, siehe [KT98]. Damit kann z. B. festgelegt werden, dass ein Name eines Knotens, wenn möglich, rechts neben den Knoten platziert werden soll.

Zur Lösung des BENENNUNGSPROBLEMS wurden viele Heuristiken entwickelt, siehe z. B. die Arbeiten von Christiansen et al. mittels Gradientensteigungsverfahren⁴ [CMS95] und Simulated Annealing⁵ [CMS94, ECMS97] oder von Zoraster mittels ganzzahliger Programmierung [Zor86, Zor90].

Kakoulis und Tollis [KT97] platzieren Kantenbenennungen gleicher Größe auf horizontalen Streifen und verwenden bipartites gewichtetes Matching zur Auflösung von Konflikten. In [KT98] verallgemeinern sie diesen Ansatz zur Platzierung beliebiger Benennungen. Wagner et al. [FW91, WW95] reduzieren das Benennungsproblem auf eine Variante des 2-SAT-Problems. Dabei wird versucht, Benennungen so zu platzieren, dass sie sich nicht überdecken und gleichzeitig die Fläche der Benennungen möglichst groß wird. In diesem, wie auch in ähnlichen Ansätzen [CFMS97, IL97], kann sich also die Größe der Benennungen ändern. Vergleiche einiger Verfahren finden sich in [CMS95]

⁴Bei Gradientensteigungsverfahren handelt es sich um Verfahren, bei denen lokale Verbesserungen zu einem globalen Optimum führen sollen. Da nur Verbesserungen erlaubt sind, können diese Verfahren in lokale Optima laufen.

⁵Simulated Annealing Verfahren sind ähnlich zu Gradientensteigungsverfahren, nur dass mit einer gewissen Wahrscheinlichkeit auch lokale Verschlechterungen erlaubt sind. Diese Wahrscheinlichkeit nimmt dabei im Laufe der Zeit ab.

und [ECMS97].

Während in der Kartographie die Lage der Objekte fixiert ist, sind Benennungen von Knoten und Kanten eines Graphen wesentlich flexibler platzierbar, da hier die Positionen der Objekte nicht zwangsläufig fest sind. In den meisten Ansätzen erfolgt die Platzierung der Benennungen dennoch *nach* der Platzierung der Knoten und Kanten. Es handelt sich dabei um das eben erwähnte Problem. Lediglich Klau und Mutzel [KM99] berechnen die Platzierung der Knoten- und Kantenbenennungen *zusammen mit* der Platzierung der Knoten und Kanten. Sie berücksichtigen die Platzierung der Benennungen dabei während der Kompaktifizierungsphase von orthogonalen Zeichenverfahren.

7.4.2 Benennungen im VGL-Verfahren

Knoten- und Kantenbenennungen lassen sich im VGL-Verfahren einfach berücksichtigen. Für Knotenbenennungen werden in der Vorverarbeitung die zugehörigen Knoten um die Größe der Benennungen vergrößert. Für Kantenbenennungen werden temporäre Knoten mit den für die Benennungen notwendigen Größen in die Kanten eingefügt. Die Positionen der Knotenzentren werden entsprechend angepasst, siehe Abbildung 7.16. In der Nachbearbeitung werden die Knoten wieder auf Originalgröße verkleinert bzw. die temporären Knoten entfernt und die Benennungen an die dafür berechneten Positionen platziert.

Im Gegensatz zu traditionellen Ansätzen aus der Kartographie schränkt dieser Ansatz die Positionen der Namen ein, für Knotenbenennungen muss z. B. bereits in der Vorverarbeitung festgelegt werden, an welcher Position relativ zum Knoten der Name erscheinen soll. Vorteil dieses Ansatzes ist, dass Überdeckungen von Namen immer vermieden werden können, da im VGL-Verfahren keine Überdeckungen zwischen Knoten auftreten.

Da das für die Implementierung von *BioPath* verwendete System Graphlet die notwendige freie Platzierung der Knoten- und Kantenbenennungen nur unzureichend unterstützt, wurde diese Erweiterung nicht realisiert.

7.5 Lokale Zeichenverfahren

7.5.1 Globale Berücksichtigung lokaler Zeichenverfahren

Die Berücksichtigung der Knotengröße im ebenenweisen Zeichenverfahren bietet neue Möglichkeiten, Teilgraphen lokal mit einem anderen Zeichenverfahren zu platzieren und in der Gesamtzeichnung zu berücksichtigen. Dabei ist es unwichtig, wie die Teilgraphen gebildet werden. Sie können beispielsweise durch den Anwender festgelegt werden oder sich aus einer Hierarchisierung des Graphen ergeben.

Im Folgenden wird die lokale Platzierung eines Teilgraphen betrachtet. Das Verfahren lässt sich auch auf eine Menge disjunkter Teilgraphen anwenden. Bei den Teilgraphen handelt es sich stets um induzierte Teilgraphen.

Sei $G = (V, E)$ ein Graph und $G_C \subseteq G$ ein Teilgraph von G mit $G_C = (V_C, E_C)$. Die Knoten aus V_C werden zu Beginn der Vorverarbeitung aus dem Graph G entfernt. Stattdessen wird ein Knoten v_C eingefügt, der zu jenen Knoten der Menge $V \setminus V_C$ adjazent ist, die bereits adjazent zu Knoten aus V_C waren. Der dadurch entstehende Graph sei G' . Die Platzierung des Graphen G (bzw. der Graphen G_C und G') erfolgt nach folgendem Algorithmus (siehe Abb. 7.17):

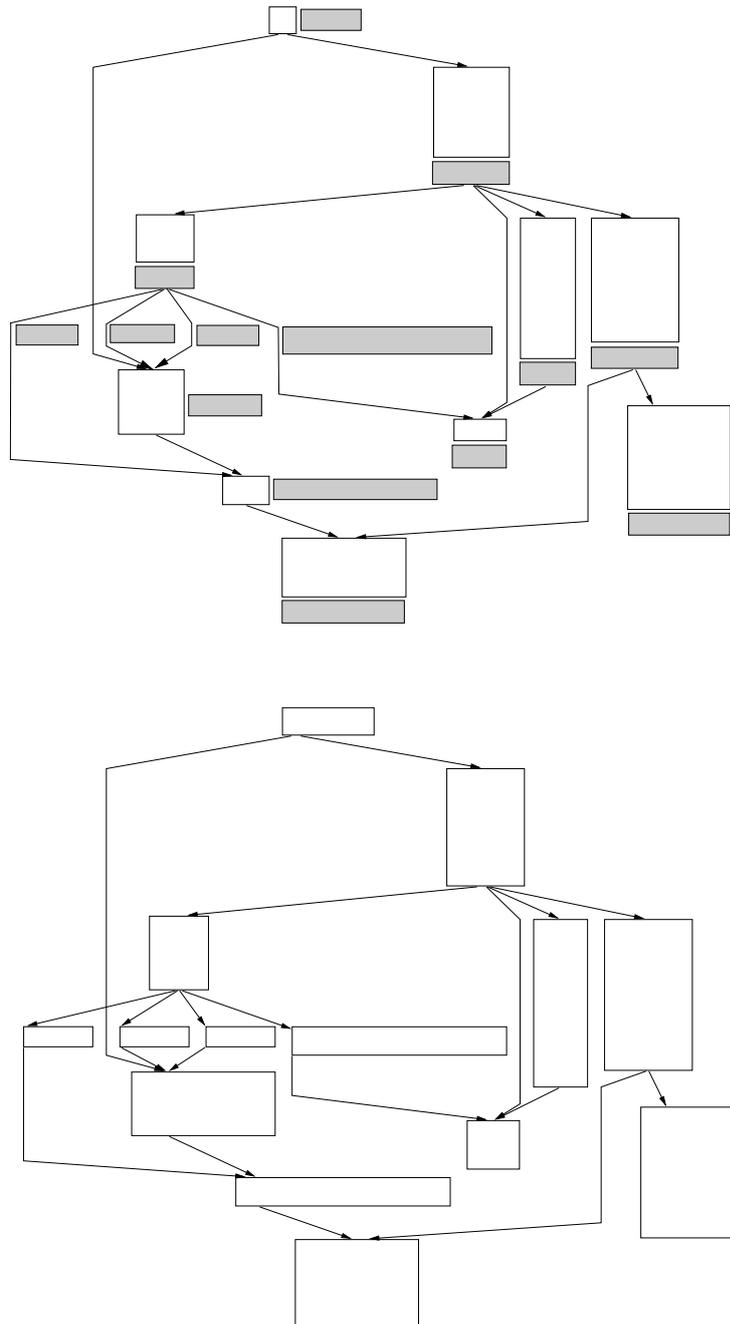


Abbildung 7.16: **Knoten- und Kantenbenennungen.** (oben) Visualisierung eines Graphen mit Knoten- und Kantenbenennung (Benennungen sind grau dargestellt) und (unten) die dafür angepassten bzw. eingefügten temporären Knoten. Knotenbenennungen wurden unterhalb oder rechts neben die Knoten, Kantenbenennungen rechts neben die Kanten platziert.

Zuerst werden auf G' die weitere Vorverarbeitung und der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG angewandt. Die Kantenorientierung legt fest, welche zu v_C inzidenten Kanten eingehende Kanten und welche ausgehende Kanten sind. Diese Information kann verwendet werden, um in der lokalen Zeichnung des Graphen G_C die Verbindungen nach außen zu berücksichtigen und bestimmte Knoten von G_C in der lokalen Zeichnung z. B. oben zu platzieren.

Nun werden Knoten und Kanten von G_C entsprechend dem lokal gewählten Zeichenverfahren platziert. Die dadurch entstehende Zeichnung benötigt eine gewisse Fläche, die durch die Größe des umschließenden Rechtecks gegeben ist. Der Knoten v_C erhält diese Größe als Knotengröße.

Anschließend werden auf G' die weiteren Schritte des VGL-Verfahren angewandt. Am Ende der Nachbearbeitung wird v_C wieder durch die ursprünglichen Knoten und Kanten des Teilgraphen G_C ersetzt, siehe Abbildung 7.17(d). Dabei werden die Stützpunkte der zu v_C inzidenten Kanten auf die originalen Kanten übertragen.

7.5.2 Verbesserung der Zeichnungen durch Ports

Nachteil des eben beschriebenen Verfahrens ist, dass die Anordnung der Knoten des Teilgraphen G_C nicht bei der Platzierung der Knoten des Graphen G' beachtet wird. Dadurch können unnötige Kreuzungen entstehen und die Kantenführung kann unruhig wirken, siehe Abbildung 7.17(d).

Durch die Verwendung von Ports kann die Platzierung verbessert werden. Ausgangspunkt ist, dass das lokale Zeichenverfahren auf den Teilgraph G_C vor der globalen Platzierung des Graphen G' angewandt wird. Zudem ist dann bereits bekannt, welche Kanten zu v_C hinführen und welche Kanten von v_C wegführen. Damit kann die Lage dieser Kanten am Rand des Knotens v_C festgelegt werden. Kanten zu v_C erhalten obere Ports, Kanten von v_C zu anderen Knoten erhalten untere Ports. Die horizontale Position der Ports wird durch die lokale Platzierung des Graphen G_C bestimmt. Die weiteren Schritte entsprechen denen ohne Verwendung von Ports, siehe Abbildung 7.18.

7.5.3 Weitere Verfahren für lokale Zeichnungen

Lokale Zeichenverfahren für Teilgraphen wurden bisher kaum betrachtet. Die folgenden Ansätze ermöglichen zwar, Gruppen von Knoten zusammenzufassen und innerhalb eines umschließenden Rechtecks zu platzieren, dabei werden jedoch alle Knoten (die zusammengefasst wie auch die restlichen Knoten des Graphen) ebenenweise gezeichnet.

So stellen Eades et al. [EFL97] für so genannte c-planare Graphen ein entsprechendes ebenenweises Zeichenverfahren vor. Dabei werden Teilgraphen wie große Knoten betrachtet, jedoch werden alle Knoten, auch die Knoten der Teilgraphen, in einer Ebenenpartitionierung berücksichtigt. Im COMPOSE-Algorithmus von Messinger et al. [MRH91] werden sowohl Teilgraphen wie der entstehende Gesamtgraph ebenenweise platziert. Hier gibt das umschließende Rechteck jeder Teilgraphzeichnung die Knotengröße des zugehörigen Knotens im Gesamtgraphen vor und die Teilgraphzeichnungen werden in Form großer Knoten in der Gesamtzeichnung berücksichtigt. Das Zeichenverfahren berechnet jedoch nur eine größenebenenmaximale Platzierung und verwendet keine anderen Zeichenverfahren zur Platzierung der Teilgraphen. Einen ähnlichen Ansatz wählen Paulisch und Tichy [PT90] im EDGE-System. Die Verwendung anderer Zeichenverfahren ist dabei ebenfalls nicht möglich.

Sanders [San96b] führt verschachtelte Graphen ein. Sein Ziel ist, die Verschachtelungsstruktur durch verschachtelte Begrenzungsrechtecke zu visualisieren. Dabei werden alle Knoten ebenenwei-

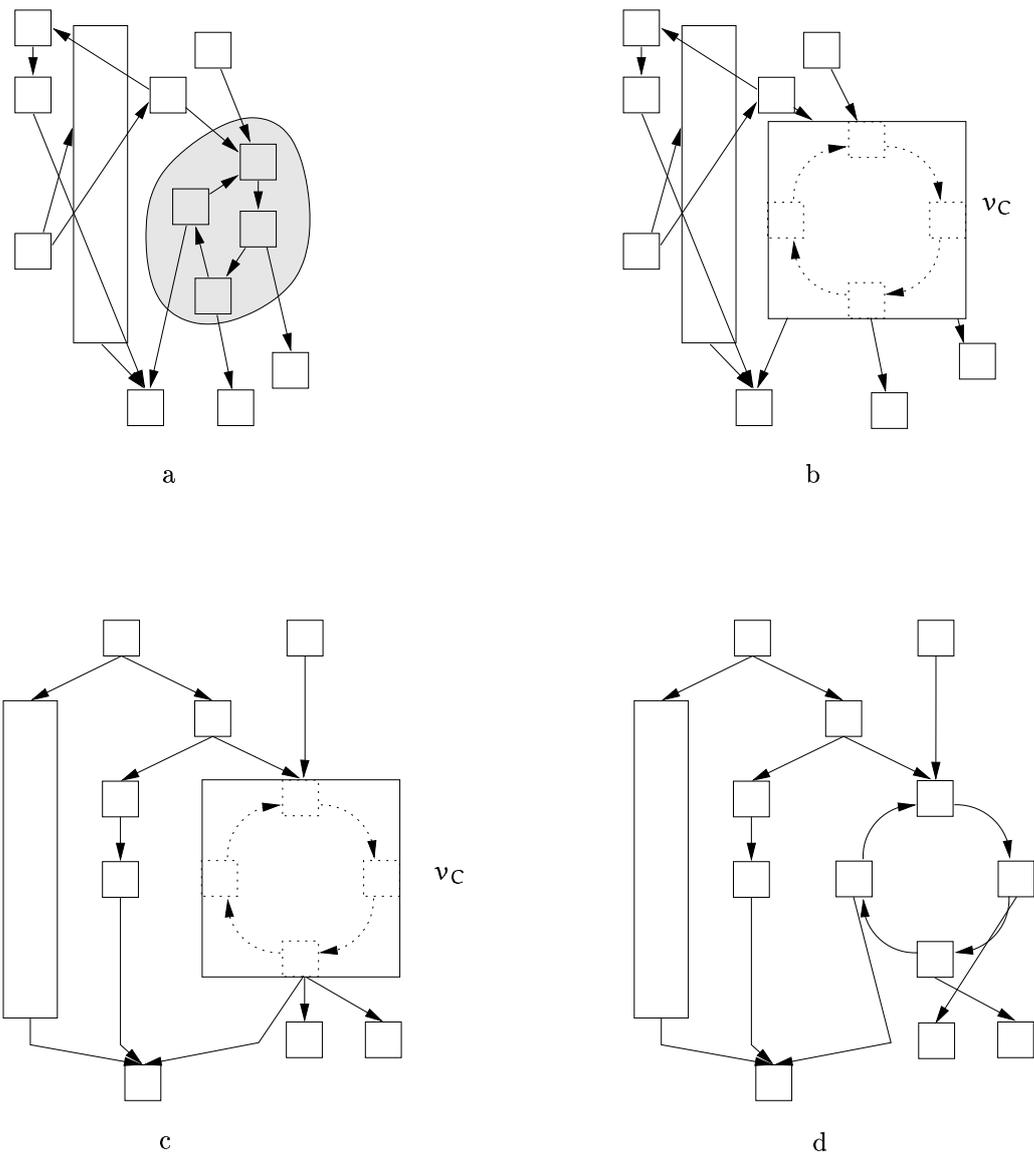


Abbildung 7.17: **Lokale Zeichenverfahren.** (a) Graph $G = (V, E)$ mit Teilgraph G_C (grau hinterlegt). (b) Im Graph G wird G_C durch einen Knoten v_C ersetzt. Nach der Kantenorientierung wird auf den Teilgraph G_C ein lokales Zeichenverfahren angewandt, hier beispielsweise eine Darstellung als Kreis. Die Umrandung gibt die Größe der lokalen Zeichnung und damit die Größe des Knotens v_C an. (c) Platzierung des gesamten Graphen. Die Anordnung der Knoten des Teilgraphen G_C wird dabei nicht berücksichtigt. (d) Visualisierung am Ende des Zeichenverfahrens. Dabei entstehen Kantenkreuzungen (z. B. zwischen der rechten unteren Kurve und einer geraden Kante), die sich durch Verwendung von Ports (siehe Abb. 7.18) vermeiden lassen.

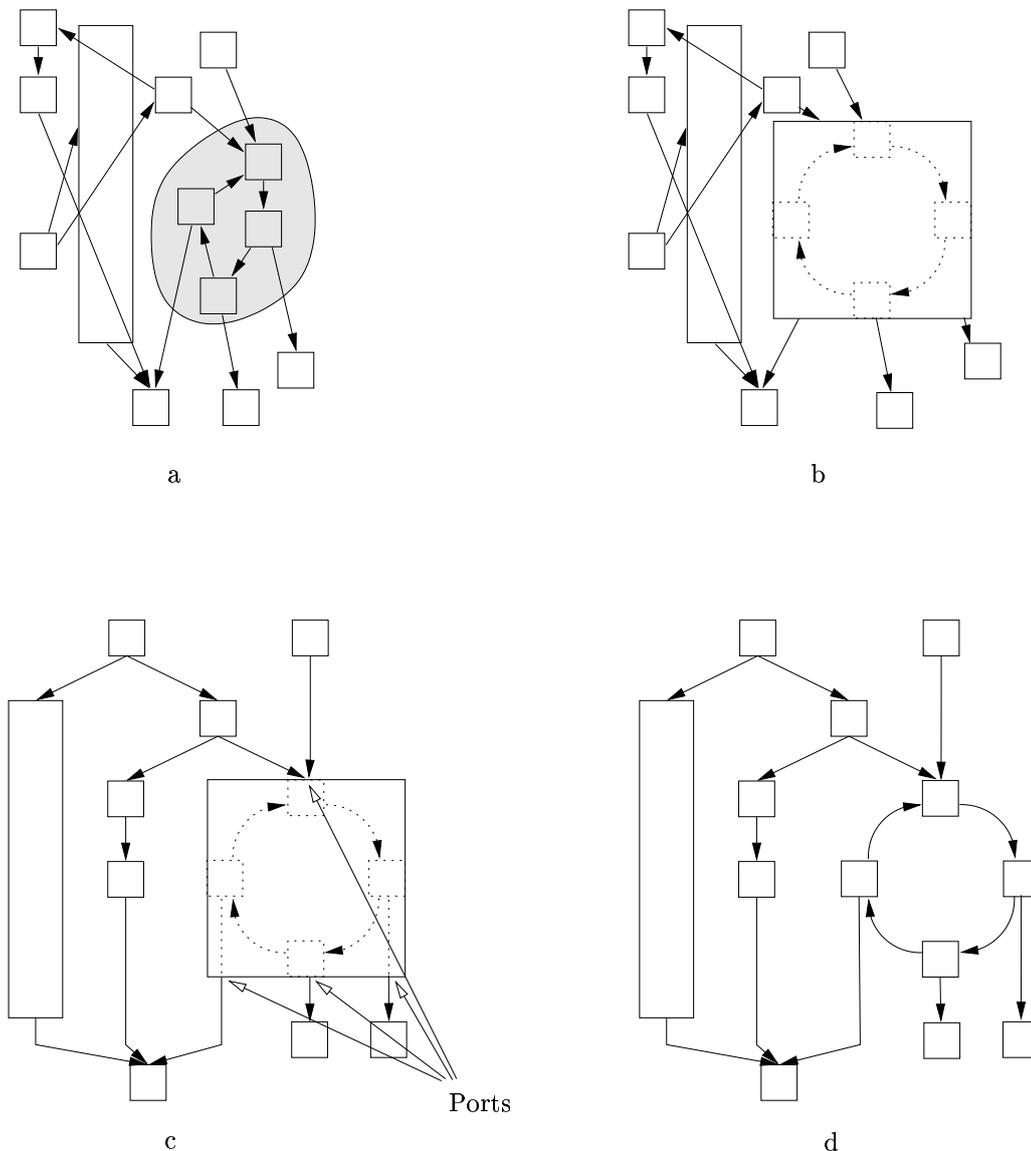


Abbildung 7.18: **Lokale Zeichenverfahren - Verwendung von Ports.** (a,b) Graph $G = (V, E)$ mit Teilgraph G_C , siehe Abbildung 7.17. (c) Nach der Kantenorientierung und der lokalen Platzierung von G_C erhält v_C Ports. Diese hängen davon ab, ob es sich um ein- oder ausgehende Kanten handelt und welche Positionen die zugehörigen inneren Knoten (Knoten aus G_C) besitzen. Hier werden z. B. vertikale Kanten von inneren Knoten zum oberen bzw. unteren Rand gezogen. Durch Verwendung von Ports werden nun unnötige Kantenkreuzungen vermieden. (d) Visualisierung am Ende des Zeichenverfahrens.

se gezeichnet, wobei Kanten zwischen verschiedenen Stufen der Verschachtelungsstruktur bei der Platzierung der Knoten berücksichtigt werden. Die Anwendung lokaler Zeichenverfahren für Teilgraphen ist jedoch nicht möglich. Sugiyama und Misue [SM91] schlagen ein ähnliches Verfahren vor, das ebenfalls auf einem ebenenweisen Zeichenverfahren beruht. Es ist in den Darstellungsmöglichkeiten gegenüber dem Verfahren von Sanders jedoch stärker eingeschränkt und ermöglicht ebenfalls keine Anwendung lokaler Zeichenverfahren für Teilgraphen.

Lediglich Bertault und Miller [BM99] ermöglichen die Anwendung anderer lokaler Zeichenverfahren für Teilgraphen. Auch bei ihrem Ansatz werden Teilgraphen durch große Knoten repräsentiert. Die Knotengröße wird jedoch nicht direkt berücksichtigt, sondern alle Knoten werden als gleich groß betrachtet. Nach Anwendung von bekannten Zeichenalgorithmen auf den Graphen werden die Knoten so verschoben, dass Knotenüberdeckungen beseitigt werden. Für die möglichen Verfahren sei auf Abschnitt 6.1.2.1 auf Seite 102 verwiesen.

7.6 Folgen von Zeichnungen

7.6.1 Einführung

In interaktiven Anwendungen ist man nicht nur an der Visualisierung einzelner Graphen, sondern auch an der Darstellung einer Folge zusammengehöriger Graphen interessiert. Sei (G_1, \dots, G_n) eine Folge zusammengehöriger Graphen und (Z_1, \dots, Z_n) die Folge der zugehörigen Zeichnungen dieser Graphen.

Bei der Erstellung einer Zeichnung Z_i , $i > 1$ unter Verwendung der Zeichnung Z_{i-1} treten neben der Erfüllung der Zeichenkonventionen und Ästhetikkriterien zwei weitere Anforderungen auf: Das Zeichenverfahren ist so zu gestalten, dass

1. die Rechenzeit zur Erstellung von Z_i möglichst gering ist.
2. der Anwender zum Verstehen der Zeichnung Z_i möglichst wenig Zeit benötigt.

Die folgenden Abschnitte behandeln diese zusätzlichen Anforderungen.

7.6.2 Inkrementelle Zeichenverfahren

Das Gewicht inkrementeller Zeichenverfahren liegt auf der Lösung der ersten Anforderung, der Reduktion der Rechenzeit zur Erstellung der Zeichnung Z_i . Dabei geht es besonders um die Entwicklung spezieller Datenstrukturen und von Verfahren, die vorhandene Platzierungsinformationen zur schnelleren Erzeugung der neuen Zeichnung ausnutzen.

Bei inkrementellen Verfahren ist die Folge (G_1, \dots, G_n) der Graphen nicht mehr beliebig, sondern der Graph G_i ergibt sich aus dem Graph G_{i-1} durch Anwenden einer der folgenden Operationen:

- Einfügen einer Kante zwischen zwei Knoten
- Einfügen eines Knotens auf einer Kante
- Einfügen eines Knotens und inzidenter Kanten
- Löschen einer Kante oder eines Knotens
- Verschieben eines Knotens

Inkrementelle Zeichenverfahren werden von verschiedenen Wissenschaftlern untersucht. Am Anfang standen Arbeiten von Tamassia et al. [TDB88] zu inkrementellen orthogonalen Zeichenverfahren. Weitere inkrementelle orthogonale Zeichenverfahren stellen Papakostas et al. [PST97], Biedl und Kaufmann [BK97], Fößmeier [Föß97] sowie Bachl [Bac01] vor. Moen [Moe90] und Cohen et al. [CDTT95] beschäftigen sich mit inkrementellen Zeichenverfahren für Bäume und Serien-Parallel-Graphen. Ein inkrementelles ebenenweises Zeichenverfahren stammt von North [Nor96]. Nachteil dieser Verfahren ist, dass ein Graph nicht beliebig verändert werden kann. Das Einfügen eines Teilgraphen muss in viele Schritte aufgeteilt werden. Deren Ausführung benötigt in der Summe u. U. wesentlich mehr Rechenzeit als ein Zeichnen des neuen Graphen mit herkömmlichen Zeichenverfahren. Zudem ist die Bestimmung der Aufteilung in die verschiedenen Schritte aufwendig und die Reihenfolge der Schritte im inkrementellen Zeichenverfahren kann das Ergebnis beeinflussen.

7.6.3 Stabilität in Zeichenverfahren

Die zweite Anforderung aus Abschnitt 7.6.1 beschäftigt sich mit der Gestaltung der Zeichnung Z_i , so dass der Anwender zum Verstehen der Zeichnung möglichst wenig Zeit benötigt. Zur Lösung dieses Problems werden so genannte *stabile* bzw. *Mental Map erhaltende* Zeichenverfahren verwendet.

7.6.3.1 Mental Map

Bei der Betrachtung von Graph-Visualisierungen erstellt ein Anwender ein mentales Bild der Zeichnung, er entwickelt eine so genannte *Mental Map* der Zeichnung [ELMS91, MELS95]. Der Anwender merkt sich wichtige Aspekte der Darstellung, z. B. die relative Lagebeziehung von Objekten oder Regionen mit vielen Objekten.

Ziel von Zeichnungen aufeinander folgender Graphen ist es, den Graph G_i so zu zeichnen, dass dessen Zeichnung Z_i mit dem mentalen Bild der vorherigen Zeichnung Z_{i-1} möglichst gut übereinstimmt, dass also Lagebeziehungen zwischen Objekten möglichst erhalten bleiben. Bei einer hohen Übereinstimmung findet sich der Anwender sehr schnell in der neuen Zeichnung Z_i zurecht. *Stabile* bzw. *Mental Map erhaltende* Zeichenverfahren liefern solche Zeichnungen.

Unter *Stabilität* des Zeichenverfahrens werden dabei zwei Aspekte verstanden:

1. *Statische Stabilität*
Ein Graph soll bei mehrfachem Anwenden des Zeichenverfahrens stets *gleich* dargestellt werden.
2. *Dynamische Stabilität*
Aufeinander folgende Graphen sollen *ähnlich* dargestellt werden.

7.6.3.2 Statische Stabilität

Die Forderung nach statischer Stabilität wird von vielen Zeichenverfahren nicht erfüllt. So führen kräftebasierte Verfahren bei mehrfacher Anwendung sehr oft zu unterschiedlichen Visualisierungen eines Graphen. Auch traditionelle ebenenorientierte Zeichenverfahren erfüllen diese Forderung nicht, siehe Abbildung 7.19.

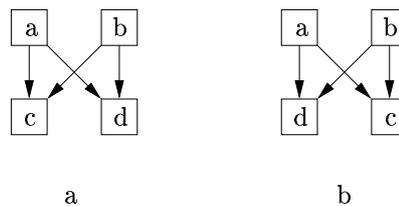


Abbildung 7.19: **Statische Stabilität (Teil 1)**. Ebenenorientierte Zeichenverfahren liefern keine statisch stabilen Visualisierungen. (a) Ein durch ein ebenenorientiertes Zeichenverfahren ZV platzierter Graph G. (b) Wird ZV wiederholt auf G angewandt, so kann diese Zeichnung entstehen. Beide Darstellungen erfüllen die Zeichenkonventionen und Ästhetikkriterien gleich gut, für einen Anwender unterscheiden sie sich jedoch.

Das VGL-Verfahren bietet eine einfache Lösung, um statische Stabilität zu erreichen: Sei $G = (V, E, M)$ ein Graph und $P(G)$ die zugehörige Platzierung. In G werden folgende Lagebeziehungen eingefügt (siehe Abb. 7.20):

1. Für jedes adjazente Knotenpaar $u, v \in V$ mit $y_u(u) < y_o(v)$ wird eine Lagebeziehung $(u, v, o-u, 2)$ eingefügt, die die Reihenfolge der Vorgänger und Nachfolger fixiert. Dadurch wird garantiert, dass die Zyklenbeseitigung in der neuen Zeichnung dieselben Kanten umdreht, die bereits bei Erstellung der Ursprungszeichnung umgedreht wurden.
2. Für jedes Knotenpaar $u, v \in V$ mit einem gemeinsamen y -Bereich (also $\exists y \quad y_o(u) \leq y \leq y_u(u) \wedge y_o(v) \leq y \leq y_u(v)$) und $x_r(u) < x_l(v)$ wird eine Lagebeziehung $(u, v, l-r, 2)$ eingefügt, die die Reihenfolge horizontal benachbarter Knoten erhält. Da wegen der oben eingeführten $o-u$ -Lagebeziehungen dieselbe Ebenenzuordnung wie in der Ursprungszeichnung garantiert wird, sind in der neuen Zeichnung Z_i dieselben Knoten wie in der Zeichnung Z_{i-1} benachbart. Durch die hier eingefügten $l-r$ -Lagebeziehungen wird die Reihenfolge der ebenenweise benachbarten Knoten fixiert.

7.6.3.3 Dynamische Stabilität

Bei ebenenweisen Zeichenverfahren hat es sich durchgesetzt, zusätzliche Lagebeziehungen zur Realisierung der dynamischen Stabilität zu verwenden. Bereits Paulisch et al. [BP90, PT90] schlagen vor, neben der statischen auch die dynamische Stabilität in ebenenweisen Zeichenverfahren durch Lagebeziehungen bzw. Constraints zu realisieren. Ihr EDGE-System unterstützt stabile Zeichnungen, die Realisierung unterscheidet sich jedoch vom VGL-Verfahren: Alle Aspekte der Platzierung, z. B. die Ebeneneinteilung der Knoten, werden durch Constraints repräsentiert. Das System dieser Abhängigkeiten wird anschließend durch einen Constraint-Solver gelöst. Dieser Ansatz ermöglicht zwar die Berücksichtigung verschiedenster Lagebeziehungen, benötigt jedoch sehr viel Rechenzeit und hat sich für ebenenweise Zeichnungen nicht durchgesetzt. Auch North [Nor96] und Waddle [Wad01] verwenden Lagebeziehungen, um dynamische Stabilität zu erreichen. Dabei werden zwei Ziele als wichtig angesehen, um die Mental Map zu erhalten:

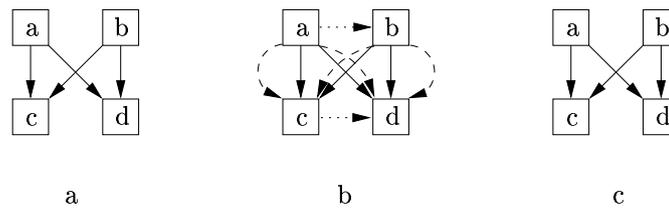


Abbildung 7.20: **Statische Stabilität (Teil 2)**. (a) Ein durch das VGL-Verfahren gezeichneter Graph, (b) die entsprechend Abschnitt 7.6.3.2 eingefügten o-u- und l-r-Lagebeziehungen (o-u-Lagebeziehungen sind gestrichelt, l-r-Lagebeziehungen gepunktet dargestellt). (c) In einer neuen Zeichnung werden die Lagebeziehungen berücksichtigt, es entsteht die gleiche Visualisierung wie in (a).

1. Der Erhalt der vertikalen Anordnung der Knoten.

Viele ebeneiseweise Zeichnungen stellen zeitliche oder hierarchische Abhängigkeiten zwischen Objekten dar. Diese Abhängigkeiten prägen wesentlich das mentale Bild, das ein Anwender von der Zeichnung erstellt. Sie sollen deshalb in einer neuen Darstellung erhalten bleiben.

2. Der Erhalt der horizontalen Anordnung innerhalb der Ebenen.

Bei benachbarten Knoten, insbesondere wenn es sich dabei um direkte Nachfolger eines Knotens handelt, prägt die Reihenfolge der Knoten das mentale Bild des Anwenders, siehe auch Abbildung 7.19. Die Reihenfolge benachbarter Knoten soll deshalb in einer neuen Darstellung erhalten bleiben.

SATZ 7.2

Die beiden Ziele zum Erhalt der Mental Map in ebeneiseweisen Zeichnungen lassen sich durch die Verwendung von Lagebeziehungen erreichen.

Beweis: Zum Erreichen dynamischer Stabilität in ebeneiseweisen Zeichenverfahren werden folgende Lagebeziehungen verwendet:

1. Für jedes Knotenpaar $u, v \in V$ mit $y_u(u) < y_o(v)$ wird eine Lagebeziehung $(u, v, o-u, 2)$ eingefügt.
2. Für jedes Knotenpaar $u, v \in V$ mit einem gemeinsamen y -Bereich (also $\exists y \ y_o(u) \leq y \leq y_u(u) \wedge y_o(v) \leq y \leq y_u(v)$) und $x_r(u) < x_l(v)$ wird eine Lagebeziehung $(u, v, l-r, 2)$ eingefügt.

Die o-u-Lagebeziehungen garantieren wieder, dass die Zyklenbeseitigung in der neuen Zeichnung dieselben Kanten umdreht, die bereits bei Erstellung der Ursprungszeichnung umgedreht wurden. Zusätzlich erhalten o-u-Lagebeziehungen auch die Anordnung nicht adjazenter Knoten. Dadurch haben Knoten in der neuen Zeichnung die gleiche oben-unten-Anordnung bzw. Hierarchie wie in der Ursprungszeichnung. Die zusätzlichen l-r-Lagebeziehungen erhalten die Reihenfolge der ebeneiseweise benachbarten Knoten.

Durch Verwendung von o-u- und l-r-Lagebeziehungen lassen sich also beide Ziele erreichen. \square

Die zum Erreichen der dynamischen Stabilität eingefügten Lagebeziehungen haben ein geringeres Gewicht als die anwendergegebenen Lagebeziehungen. Dadurch hat ein Anwender die Möglichkeit, zusätzliche Lagebeziehungen anzugeben.

Ein Beispiel für dynamische Stabilität mit dem VGL-Verfahren zeigt Abbildung 7.21.

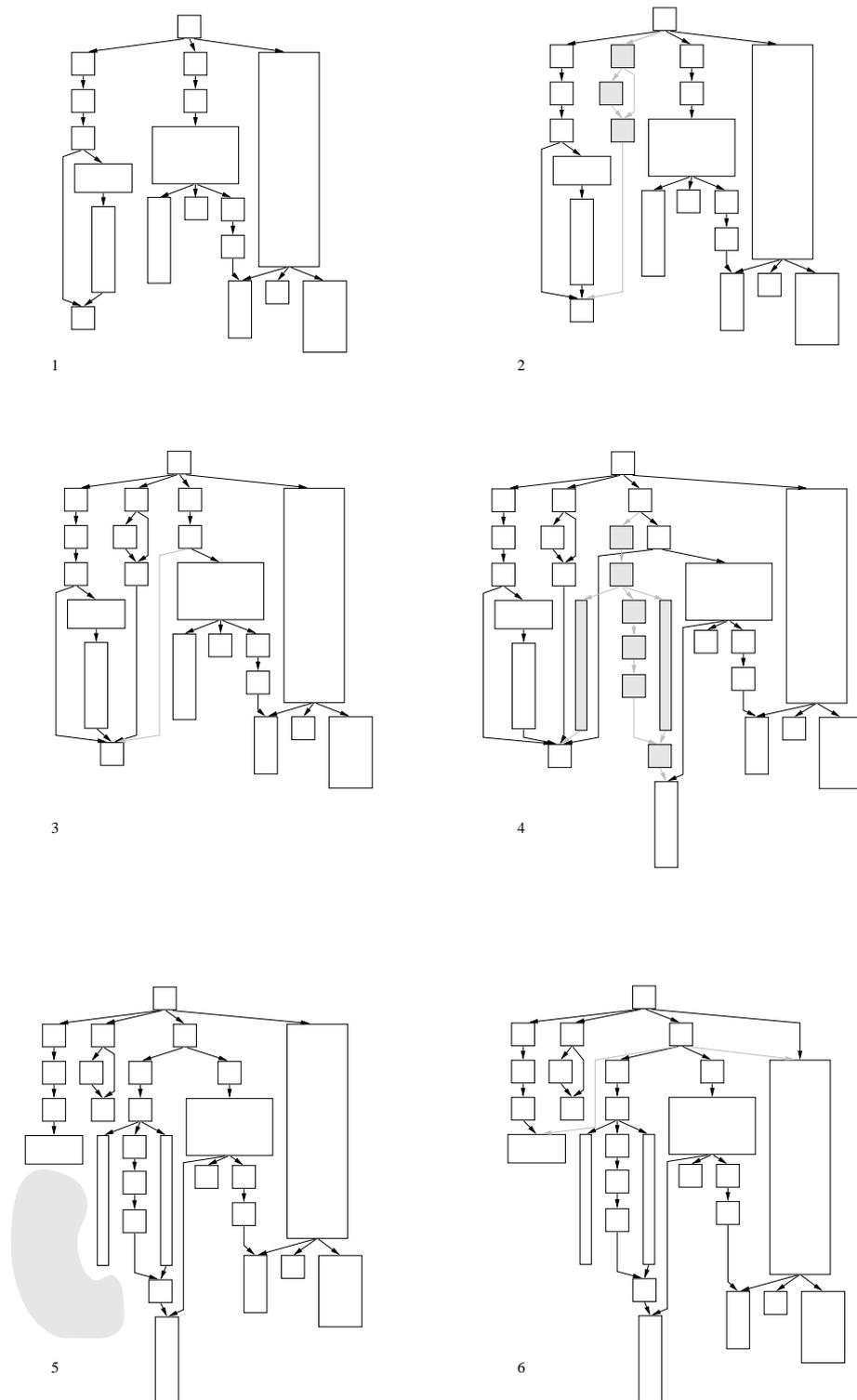
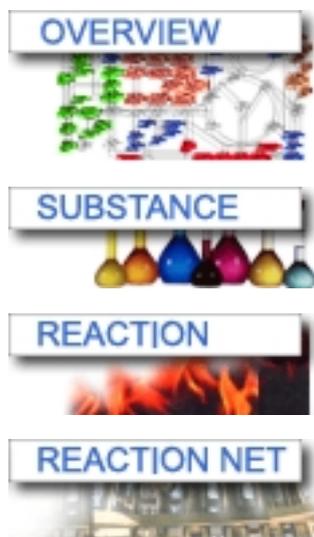


Abbildung 7.21: **Dynamische Stabilität.** Darstellung der zu einer Folge von Graphen (G_1, \dots, G_6) gehörigen Zeichnungen (Z_1, \dots, Z_6) . Änderungen sind jeweils grau markiert.

8 Zeichnungen biochemischer Reaktionsnetze



8.1	Zeichenverfahren für biochemische Reaktionsnetze	208
8.2	Anwendungen	225
8.3	Regulation und deren Darstellung	229

Verschiedene Symbole aus dem *BioPath*-Projekt. Diese repräsentieren Möglichkeiten für die Erkundung biochemischer Reaktionsnetze.

8.1 Zeichenverfahren für biochemische Reaktionsnetze

In den Kapiteln 6 und 7 wurde ein universelles ebenenorientierte Zeichenverfahren für Graphen mit beliebigen Knotengrößen vorgestellt. Für die speziellen Erfordernisse der Biochemie sind einige Anpassungen notwendig.

8.1.1 Reaktionen

Die Visualisierung biochemischer Reaktionen erfordert spezielle Anordnungen der Cosubstanzen und Enzyme einer Reaktion, siehe dazu die Anforderung 3.2 (Seite 43). Diese Anforderung lässt sich durch eine lokale Platzierung der Komponenten erfüllen. Dazu werden Reaktionsknoten, Cosubstanzen und Enzyme jeder Reaktion zu einem großen Knoten wie in Abbildung 8.1 zusammengefasst.

Sei $RG = (K \cup R, A \cup Z, B, T)$ ein Reaktionsgraph und $v_R \in R$ ein Reaktionsknoten. In RG werden die zu v_R adjazenten Knoten für Cosubstanzen und Enzyme temporär entfernt und der Knoten v_R vergrößert. Die Größe des Reaktionsknotens v_R lässt sich aus der Größe der Knoten der zugehörigen Cosubstanzen und Enzyme, der Breite b_R des Reaktionspfeils, den Abständen Enzym-Reaktion a_{ER} , Enzym-Enzym a_{EE} , Cosubstanz-Reaktion a_{CR} , Cosubstanz-Cosubstanz a_{CC} sowie einem zusätzlichen Rand r für oben und unten berechnen. Sei dabei V_C die Menge der zugehörigen Cosubstanzen und V_E die Menge der zugehörigen Enzyme. Es gilt:

$$\begin{aligned} b_l(v_R) &= \max\{b(v) \mid v \in V_E\} + a_{ER} + \frac{1}{2}b_R \\ b_r(v_R) &= \max\{b(v) \mid v \in V_C\} + a_{CR} + \frac{1}{2}b_R \\ h(v_R) &= \max\left\{\left(\sum_{v \in V_E} (h(v) + a_{EE}) - a_{EE}\right), \left(\sum_{v \in V_C} (h(v) + a_{CC}) - a_{CC}\right)\right\} + 2r \end{aligned}$$

Es wird davon ausgegangen, dass alle Cosubstanzen eine Mindesthöhe $h_m > 0$ besitzen, da zumindest ein einzeliger Name in der Darstellung enthalten ist, sowie dass $a_{CR} > h_m$ gilt. Dies ist erforderlich, damit der Kreisbogen zwischen Reaktionskante und Cosubstanz dargestellt werden kann. Für einen Reaktionsknoten v_R ohne adjazente Cosubstanzen und Enzyme ergibt sich dessen Größe zu $b(v_R) = b_R$ und $h(v_R) = b_R$.

Dieses Zusammenfassen der Komponenten einer Reaktion zu einem großen Knoten ist immer möglich, da Enzyme und Cosubstanzen nur einer Reaktion zugeordnet sind. Der Graph bleibt dabei ein Reaktionsgraph, da nur Knoten und inzidente Kanten entfernt, aber keine neuen Kanten eingefügt werden.

Nun wird für den entstehenden Graph eine Platzierung berechnet.¹ Anschließend lassen sich die großen Knoten für Reaktionen auf ihre ursprüngliche Größe verkleinern und die temporär entfernten Knoten wieder in den Reaktionsgraphen einfügen, wie dies in Abbildung 8.2(a) dargestellt ist. Es muss lediglich darauf geachtet werden, dass Reaktionen, die während der Platzierung umgedreht wurden, gesondert behandelt werden. Für diese erfolgt die Zuordnung der Koordinaten für

¹Entscheidend hierbei ist, dass das VGL-Verfahren durch Berücksichtigung beliebiger Knotengrößen ein lokales Zeichnen von Teilen des Graphen im globalen Kontext unterstützt. Für die Berücksichtigung lokaler Zeichenverfahren sei auf Abschnitt 7.5 verwiesen.

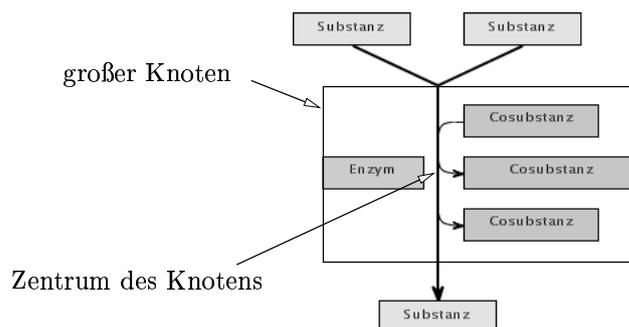


Abbildung 8.1: **Knoten für Reaktion.** Zusammenfassen von Reaktion, Enzym(en) und Cosubstanz(en) zu einem Knoten.

Enzym- und Cosubstanz-Knoten in beiden Ebenen spiegelverkehrt, wie dies Abbildung 8.2(b) zeigt. Das Umdrehen der Richtung einer Reaktion, und damit der Richtung des Reaktionsknotens, wird im nächsten Abschnitt ausführlich betrachtet.

Neben der Erfüllung der Anforderung an die Platzierung von Enzymen und Cosubstanzen hat dieses Zusammenfassen von Knoten einen weiteren Vorteil: Die Zahl der Knoten im Graph wird verringert, was sich positiv auf die Rechenzeit des Zeichenverfahrens auswirkt.

8.1.2 Reaktionsnetze

8.1.2.1 Kantensorientierung

In Abschnitt 6.2 wurde die Kantensorientierung des VGL-Verfahrens behandelt. Diese muss an die Behandlung von Reaktionsgraphen angepasst werden. Zur Begründung betrachte man die Reaktionsfolge $A \rightarrow B \rightarrow A$. Die Anwendung der normalen Kantensorientierung würde eine Darstellung wie in Abbildung 8.3(a) liefern. Diese Darstellung ist falsch: Sie erfüllt nicht die Anforderung 3.2, nach der Edukte und Produkte auf verschiedenen Seiten des Reaktionspfeils liegen sollen und stellt damit den chemischen Sachverhalt verfälscht dar, da die untere Reaktion leicht als eine Reaktion mit zwei Edukten und ohne Produkt fehlgedeutet werden kann.

Ein erster Lösungsansatz wäre, beim Umdrehen einer mit einem Reaktionsknoten v_R inzidenten Kante alle mit v_R inzidenten Kanten umzudrehen, wie dies in Abbildung 8.3(b) dargestellt ist. Dabei wird auch der Knoten v_R als umgedreht markiert und seine Breiten b_r und b_l werden vertauscht, um die im vorherigen Abschnitt beschriebene Anordnung der Komponenten bei umgedrehten Reaktionen zu berücksichtigen.

Dieser Ansatz ist jedoch falsch, wie Abbildung 8.4 zeigt. Dazu sei angenommen, dass die Kantensorientierung bereits die Kanten des Reaktionswegs $A \rightarrow C$ betrachtet hat, siehe Abbildung 8.4(a). Nun wird eine Reaktion $A + C \rightarrow B$ eingefügt. Sei dabei v_R der Knoten der eingefügten Reaktion. Die Reihenfolge des Einfügens der Kanten durch die Kantensorientierung sei (A, v_R) , (C, v_R) , (v_R, B) . Beim Einfügen wird die Kante (v_R, B) umgedreht, um einen Zyklus zu vermeiden (siehe Abb. 8.4(b)). Nach der oben vorgeschlagenen Lösung müssen nun alle zu v_R inzidenten Kanten umgedreht werden. Dies ist jedoch falsch, da dadurch ein neuer Zyklus $B \rightarrow v_R \rightarrow A \rightarrow v_R' \rightarrow B$ entsteht (Abb. 8.4(c)).

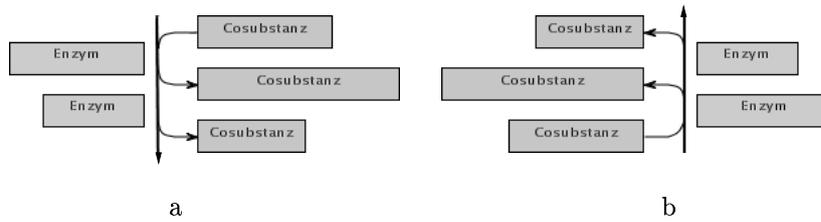


Abbildung 8.2: **Anordnung der Komponenten einer Reaktion.** (a) Am Ende des Zeichenverfahrens werden Enzyme – entsprechend ihrer Reihenfolge und zentriert in y -Richtung – links neben den Reaktionspfeil platziert. Cosubstanzen werden rechts – ebenfalls entsprechend der gegebenen Reihenfolge und in y -Richtung zentriert – platziert. Cosubstanzen werden mit der Reaktionskante² so verbunden, dass das letzte Stück als Kreisbogen mit Radius h_m dargestellt wird. Dabei ist die Position des Mittelpunkts des Kreisbogens abhängig davon, ob die Cosubstanz ein Coedukt oder ein Coprodukt ist.

(b) Falls die Reaktionsrichtung von unten nach oben verläuft, ist die Platzierung an x - und y -Achse gespiegelt.

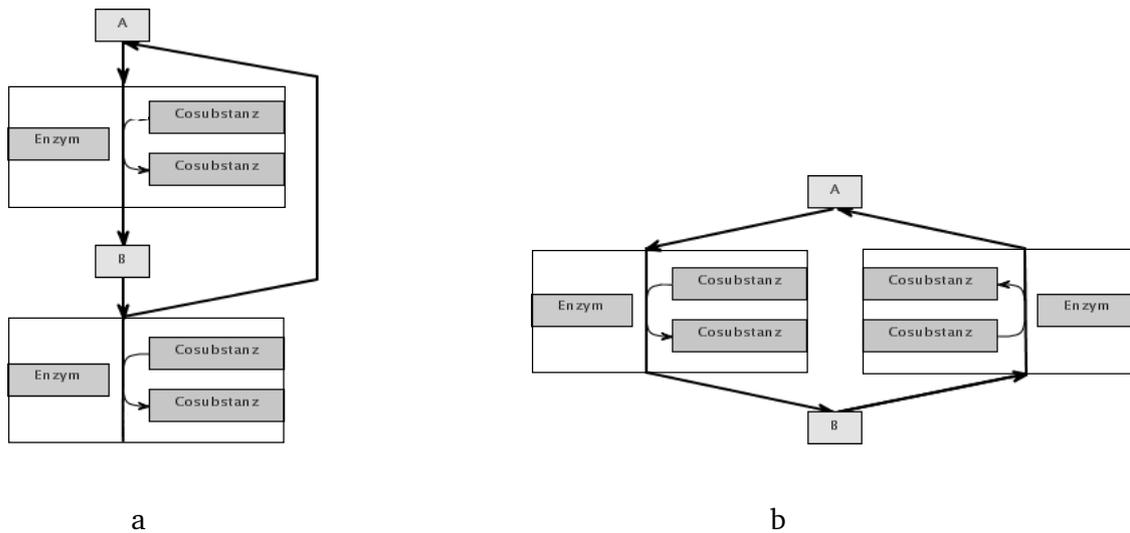


Abbildung 8.3: **Probleme bei der Kantenorientierung.**³ Darstellung der Reaktionsfolge $A \rightarrow B \rightarrow A$. (a) Die Kantenorientierung des VGL-Verfahrens führt zu einer falschen Darstellung, die Reaktion von B nach A ist nicht mehr leicht als solche erkennbar. (b) Ein erster Lösungsansatz: Wird eine zum Reaktionsknoten v_R inzidente Kante umgedreht, so werden alle zu v_R inzidenten Kanten umgedreht und der Knoten v_R als umgedreht markiert.

²Wie in der Modellierung beschrieben, ist die Reaktionskante durch einen Knoten repräsentiert. Dieser Knoten (der vertikale Strich) erscheint in der Visualisierung jedoch als eine Kante.

³In dieser und den folgenden Abbildungen sind zur Verdeutlichung der Kantenrichtung Pfeilspitzen zwischen Substanz- und Reaktionsknoten dargestellt. Zudem ist die Umrandung des vergrößerten Reaktionsknotens eingezeichnet. Zusätzliche Pfeilspitzen und Umrandung sind in der endgültigen Visualisierung des biochemischen Reaktionsnetzes nicht mehr enthalten.

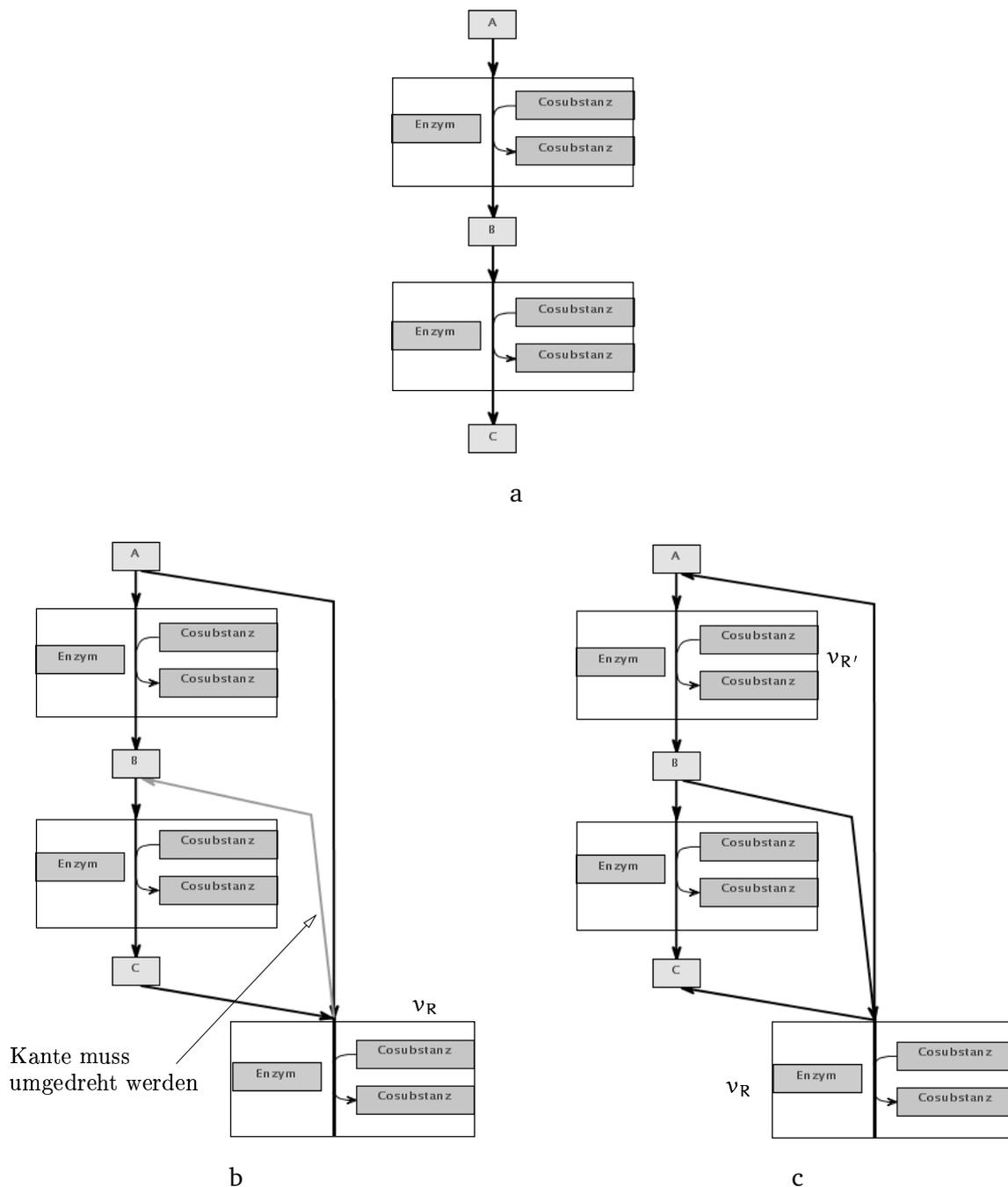


Abbildung 8.4: **Kantenorientierung – erster Ansatz.** (a) Der bereits durch die Kantenorientierung betrachtete Teil eines Reaktionsgraphen. (b) Die zu v_R inzidenten Kanten werden nacheinander durch die Kantenorientierung betrachtet, die Kante (v_R , B) wird als dritte Kante eingefügt und umgedreht. (c) Ein im Lösungsansatz vorgeschlagenes Umdrehen aller zu v_R inzidenten Kanten ist falsch, da es in diesem Beispiel zu einem neuen Zyklus ($B \rightarrow v_R \rightarrow A \rightarrow v_{R'} \rightarrow B$) führt.

Dieses Beispiel motiviert das folgende Verfahren für die Kantenorientierung von biochemischen Reaktionsnetzen.

8.1.2.2 Angepasste Kantenorientierung

Wenn bei der Kantenorientierung von biochemischen Reaktionsnetzen Zyklen erkannt werden, lassen sich zwei Fälle unterscheiden:

1. Situationen, in denen ein Umdrehen des Reaktionsknotens und aller inzidenten Kanten möglich ist, siehe Abbildung 8.5.
2. Situationen, in denen ein Umdrehen des Reaktionsknotens und aller inzidenten Kanten nicht möglich ist. In diesen Fällen müssen deshalb Kanten mit zusätzlichen Stützpunkten versehen werden, um Reaktionen korrekt darzustellen, siehe Abbildung 8.6.

Das Umdrehen eines Reaktionsknotens und aller inzidenten Kanten ist möglich, falls in der Kantenorientierung die aktuell betrachtete Kante e

1. die erste ein- oder ausgehende Kante eines Reaktionsknotens ist und
2. an einer höheren Ebene als die Endknoten aller ausgehenden Kanten beginnt bzw. an einer niedrigeren Ebene als die Startknoten aller eingehenden Kanten endet.

In Abbildung 8.5 (a) ist beispielsweise die Kante (v_R, v_a) die erste ausgehende Kante des Reaktionsknotens v_R . Der Endknoten v_a der Kante (v_R, v_a) liegt auf einer Ebene, die niedriger als die Ebenen aller Startknoten von eingehenden Kanten des Knotens v_R ist.

Die Berechnung der angepassten Kantenorientierung zerfällt in zwei Schritte:

1. Das Umdrehen von Reaktionsknoten und Kanten während der Kantenorientierung. Dazu wird der Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG (Seite 127) erweitert, indem die Zeile 43
„Drehe Richtung von e um; $E_R := E_R \cup \{e\}$;“
durch die Zeilen 142-168 (siehe Abschnitt 8.1.2.2.1) ersetzt wird.
2. Das Einfügen von temporären Knoten im Anschluss an die Kantenorientierung. Es werden temporäre Knoten in Kanten eingefügt, bei denen Stützpunkte für die korrekte Verbindung von Reaktionsknoten und Substanzknoten nötig sind. Dieser Teil wird im Abschnitt 8.1.2.2.2 betrachtet.

8.1.2.2.1 Erweiterung des Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG

Die Erweiterung ersetzt im Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG (Seite 127) den Teil, in welchem nach Erkennen eines Zyklus eine Kante umgedreht wird (Zeile 43 auf Seite 127). Sei e die umzudrehende Kante.

Zuerst wird bestimmt, ob der Reaktionsknoten v_R und die bereits betrachteten inzidenten Kanten umgedreht werden können (Zeilen 142-150). Falls dies möglich ist, werden v_R und diese Kanten umgedreht. Sie werden aus B , der Menge der bereits abgearbeiteten Kanten, entfernt und wieder der Liste L zugefügt. Die Liste L aus dem originalen Algorithmus enthält alle zu bearbeitenden

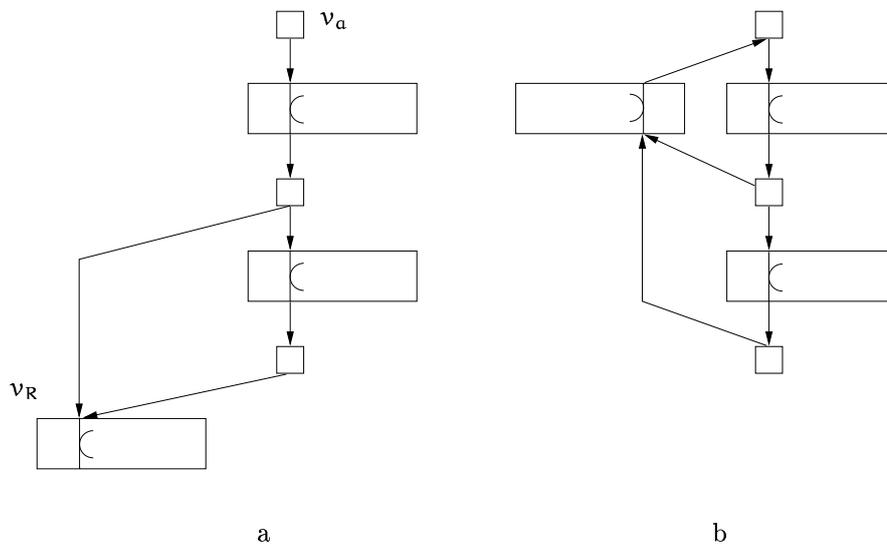


Abbildung 8.5: **Situationen bei der Kantenorientierung (Teil 1).** (a) Ausgangssituation während der Kantenorientierung, die Kante (v_R, v_a) soll eingefügt werden. Dabei werden alle zu v_R inzidenten Kanten umgedreht. (b) Situation am Ende des Gesamtverfahrens, die Kanten sind hier bereits wieder in originaler Richtung orientiert.

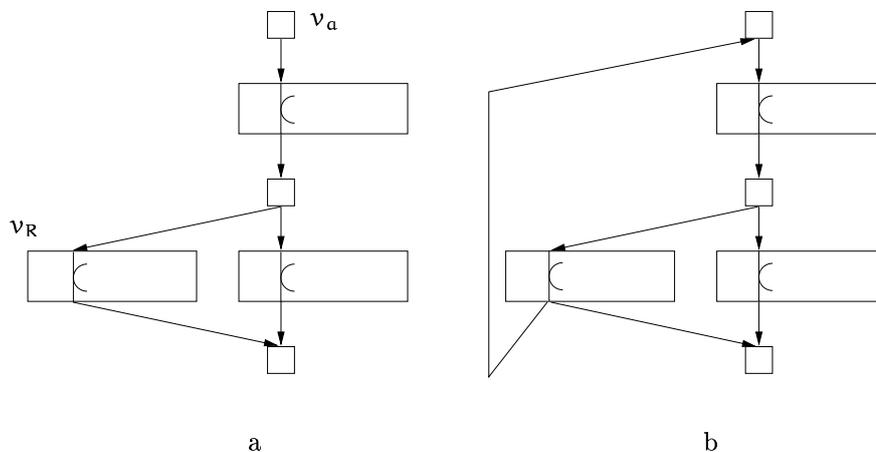


Abbildung 8.6: **Situationen bei der Kantenorientierung (Teil 2).** (a) Ausgangssituation während der Kantenorientierung, die Kante (v_R, v_a) soll eingefügt werden. Dabei muss (v_R, v_a) umgedreht werden. Ein Umdrehen der weiteren zu v_R inzidenten Kanten ist nicht möglich, da dadurch ein neuer Zyklus entstehen würde. (b) Situation am Ende des Gesamtverfahrens nach Einfügen zusätzlicher Stützpunkte, auch hier sind die Kanten bereits wieder in originaler Richtung orientiert.

Kanten entsprechend ihrem Gewicht geordnet. Der Knoten v_R wird aus der Menge M (Menge der bereits betrachteten Knoten) entfernt, außerdem wird in diesem Schleifendurchlauf keine Kante in B und kein Knoten in M eingefügt. Damit kann v_R später nochmals neu auf eine passende Ebene platziert werden.

Falls der Knoten v_R nicht umgedreht werden kann, so wird, wie im originalen Algorithmus, lediglich die aktuelle Kante umgedreht.

Bemerkung: Ersetzt Zeile 43 (Seite 127).

Der Graph $G = (K \cup R, E)$ ist ein Reaktionsgraph, $e = (u, v)$ ist die aktuelle Kante.

```

142 drehe_vR :=false;
143 if (u ∈ R) and (Aus(u) ∩ B = ∅) and (u ist nicht als umgedreht markiert)
144     vR := u;
145     if ebene[v] < min{ebene[v1] | (v1, vR) ∈ B}
146         drehe_vR :=true;
147 if (v ∈ R) and (Ein(v) ∩ B = ∅) and (v ist nicht als umgedreht markiert)
148     vR := v;
149     if ebene[u] > max{ebene[v2] | (vR, v2) ∈ B}
150         drehe_vR :=true;
151
152 if drehe_vR=true
153     forall e' ∈ ((Ein(vR) ∪ Aus(vR)) ∩ B) ∪ {e}
154         Drehe Richtung von e' um;
155         B := B \ {e'}; L := L + {e'};4
156         if (e' ∉ E_R)
157             E_R := E_R ∪ {e'};
158         else
159             E_R := E_R \ {e'};5
160         M := M \ {vR}; Markiere vR als umgedreht;
161         Sortiere L entsprechend den Gewichten bew[] in absteigender Reihenfolge;
162         Überspringe Zeile 44;6
163 else
164     Drehe Richtung von e um;
165     if e ∉ E_R
166         E_R := E_R ∪ {e};
167     else
168         E_R := E_R \ {e};

```

SATZ 8.1

Der wie oben geänderte Algorithmus zur ebenengestützten Kantenorientierung berechnet einen azyklischen Graphen.

⁴ B ist die Menge der bereits betrachteten Kanten, L die Liste der noch zu bearbeitenden Kanten aus dem Algorithmus EBENENGESTÜTZTE KANTENORIENTIERUNG, + fügt ein Element an die Liste an.

⁵Kanten, die bereits umgedreht waren und durch das aktuelle Drehen wieder in Originalrichtung zeigen, werden aus der Menge E_R (Menge der umgedrehten Kanten) entfernt.

⁶Die Zeile 44 ($B := B \cup \{e\}$; $M := M \cup \{u, v\}$;) wird nicht ausgeführt, sie wird in diesem Schleifendurchlauf übersprungen. Dadurch wird keine Kante in die Menge B zugefügt.

Beweis: Es sind zwei Aussagen zu zeigen:

1. Die Invariante $\forall (v_1, v_2) \in B \text{ ebene}[v_1] < \text{ebene}[v_2]$ bleibt gültig.
2. Der Graph wird vollständig betrachtet: Alle Kanten werden in B und alle Knoten in M eingefügt.

Zu 1.: Angenommen, die Invariante gilt vor einem Durchlauf der Zeilen 142 bis 168. Um ihre Gültigkeit danach zu zeigen, sind zwei Fälle zu unterscheiden:

1. Falls in Zeile 152 die Variable `drehe_vR` wahr ist, so werden in den Zeilen 153-162 nur solche Kanten in ihrer Richtung geändert, die auch aus B entfernt werden. Da die anderen Kanten nicht geändert werden und in diesem Schleifendurchlauf in B keine neue Kante zugefügt wird, gilt die Invariante auch nach einem Schleifendurchlauf.
2. Falls in Zeile 152 die Variable `drehe_vR` falsch ist, so wird, wie im originalen Algorithmus, die Kante *e* umgedreht (Zeilen 164-168), es gilt also Lemma 6.18 und damit die Invariante nach dem Schleifendurchlauf.

Zu 2.: Falls ein Knoten $v_R \in R$ umgedreht wird, werden alle seine inzidenten Kanten aus B entfernt. Dies geschieht für jeden Knoten höchstens einmal. Die aus B entfernten Kanten werden wieder in L eingefügt und damit später nochmals betrachtet.

Die aus M entfernten Knoten werden später ebenfalls wieder betrachtet und in M eingefügt, da mit ihnen inzidente Kanten in L enthalten sind und damit noch behandelt werden. $G = (M, B)$ enthält am Ende des Algorithmus jede Kante und jeden Knoten des Graphen, dieser wird also vollständig betrachtet.

Auch der geänderte Algorithmus berechnet einen azyklischen Graphen. □

8.1.2.2.2 Stützknoten

Als *Stützknoten* werden temporäre Knoten bezeichnet, die später zu zusätzlichen Stützpunkten der Kanten werden. Diese dienen dazu, die Kanten zwischen Reaktionsknoten und Substanzknoten korrekt zu platzieren, siehe Abbildung 8.8.

Das Einfügen von Stützknoten lässt sich anhand der Abbildung 8.7 erklären. Abbildung 8.7(a) stellt die Situation an normalen Reaktionsknoten, Abbildung 8.7(b) die an umgedrehten Reaktionsknoten dar. Dabei treten für Kanten folgende Fälle auf:

Fall	Kante nach Kantenorientierung	Aktion
1	Kante (u, v_R) , also $\text{ebene}[u] < \text{ebene}[v_R]$	
1.1	Knoten v_R <i>normal</i>	
1.1.1	$(u, v_R) \notin E_R$	
1.1.2	$(u, v_R) \in E_R$	Ersetze (u, v_R) durch $v_{\text{neu}}, (u, v_{\text{neu}}), (v_R, v_{\text{neu}})$
1.2	Knoten v_R <i>umgedreht</i>	
1.2.1	$(u, v_R) \notin E_R$	Ersetze (u, v_R) durch $v_{\text{neu}}, (u, v_{\text{neu}}), (v_R, v_{\text{neu}})$
1.2.2	$(u, v_R) \in E_R$	
2.	Kante (v_R, u) , also $\text{ebene}[v_R] < \text{ebene}[u]$	
2.1	Knoten v_R <i>normal</i>	
2.1.1	$(v_R, u) \notin E_R$	
2.1.2	$(v_R, u) \in E_R$	Ersetze (v_R, u) durch $v_{\text{neu}}, (v_{\text{neu}}, u), (v_{\text{neu}}, v_R)$
2.2.	Knoten v_R <i>umgedreht</i>	
2.2.1	$(v_R, u) \notin E_R$	Ersetze (v_R, u) durch $v_{\text{neu}}, (v_{\text{neu}}, u), (v_{\text{neu}}, v_R)$
2.2.2	$(v_R, u) \in E_R$	

8.1.2.2.3 Reversible Reaktionen

Reversible Reaktionen sind im Reaktionsgraph als einfache Reaktionen mit einer Richtung und einem entsprechendem Attribut zur Kennzeichnung der Reversibilität gegeben. Im Unterschied zu nicht reversiblen Reaktionen ist es bei reversiblen Reaktionen nicht nötig, den Reaktionsknoten v_R umzudrehen. Es macht keinen Unterschied, ob die reversible Reaktion $A \leftrightarrow B$ in Richtung $A \rightarrow v_R \rightarrow B$ oder in Richtung $B \rightarrow v_R \rightarrow A$ betrachtet wird. Das Verfahren zum Einfügen von Stützpunkten bleibt dagegen gleich. Die Pfeilspitzen bei reversiblen Reaktionen sind grafische Attribute, die bereits im Reaktionsgraphen gegeben und unabhängig vom Platzierungsalgorithmus sind.

8.1.2.3 Spezielle Reaktionswege

Offene und geschlossene Zyklen sollen entsprechend Anforderung 3.4 (Seite 46) dargestellt werden. Dazu müssen diese speziell behandelt werden.

8.1.2.3.1 Offene Zyklen

Vorausgesetzt sei ein Reaktionsweg RW , der als offener Zyklus Teil eines Reaktionsnetzes ist und für den gilt:

1. RW ist unterteilt in die aufeinander folgenden Teilwege RW_1, \dots, RW_n , die alle gleiche Länge haben.
2. Kein Teil von RW ist in einem anderen offenen oder geschlossenen Zyklus enthalten.⁷

⁷Diese Voraussetzung ist für die Verwendung von ver-Lagebeziehungen nötig und entspricht auch den Gegebenheiten in biochemischen Reaktionsnetzen.

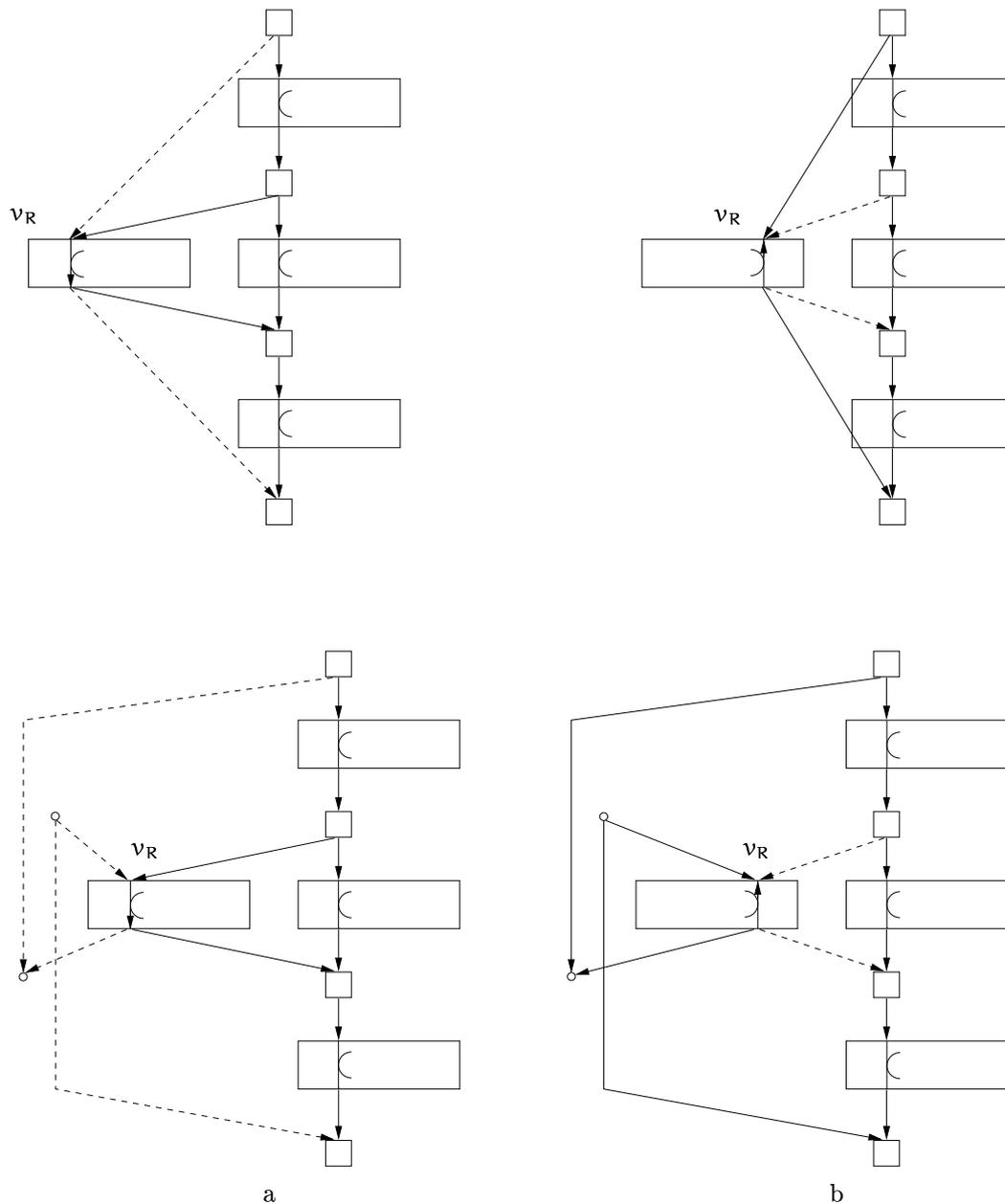


Abbildung 8.7: **Einfügen von Stützknoten (Teil 1)**. Situation (oben) vor Einfügen von Stützknoten, (unten) nach Einfügen von Stützknoten. (a) Behandlung normaler Reaktionsknoten v_R , (b) Behandlung umgedrehter Reaktionsknoten v_R .

Kanten, die vor dem Einfügen temporärer Knoten umgedreht waren (also Kanten aus E_R), sind jeweils gestrichelt dargestellt. Die Richtung der Reaktionsknoten v_R ist durch einen Pfeil hervorgehoben. Abbildung 8.8 zeigt die fertige Visualisierung.

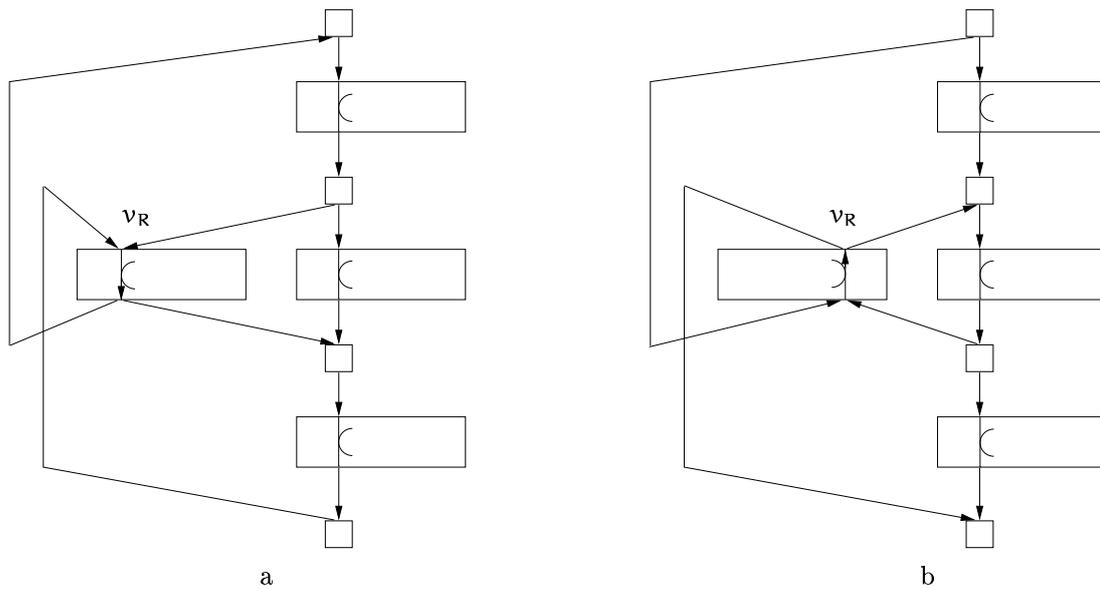


Abbildung 8.8: **Einfügen von Stützknoten (Teil 2)**. Visualisierung nach Ersetzen der temporären Knoten durch Stützpunkte und Orientierung der Kanten in originaler Richtung.

Für Teilwege RW_i und RW_{i+1} mit $1 \leq i < n$ sei v_e der letzte Knoten des Teilwegs RW_i und v_a der erste Knoten des Teilwegs RW_{i+1} .

Das Einfügen zusätzlicher Knoten und spezieller Lagebeziehungen für offene Zyklen geschieht wie folgt (siehe auch Abb. 8.9), dabei haben alle hier eingefügten Lagebeziehungen ein höheres Gewicht als die vom Anwender zusätzlich vorgegebenen Lagebeziehungen:

1. Ersetze die Kante (v_e, v_a) durch Knoten v_1, v_2, v_3 , die wie folgt verbunden sind: (v_e, v_1) , (v_2, v_1) , (v_3, v_2) , (v_3, v_a) (siehe Abb. 8.9 (a)).
2. Füge o-u-Lagebeziehungen zwischen aufeinander folgenden Knoten desselben Teilwegs sowie zwischen v_3 und v_a , v_3 und v_2 , v_2 und v_1 sowie v_e und v_1 ein (siehe Abb. 8.9(b), o-u-Lagebeziehungen sind gestrichelt dargestellt). Dies garantiert, dass auch bei der Berücksichtigung weiterer Reaktionen, an denen Teile des offenen Zyklus beteiligt sind, für die Teilwege des Zyklus die vorgegebene Richtung von oben nach unten erhalten bleibt.
3. Füge ver-Lagebeziehungen zwischen aufeinander folgenden Knoten desselben Teilwegs ein (siehe Abb. 8.9(b), ver-Lagebeziehungen sind mit durchgezogenen Pfeile dargestellt). Dies garantiert, dass die Knoten der Teilwege untereinander platziert werden, auch wenn auf Grund weiterer Reaktionen mehr als eine Kante zu einer Substanz oder Reaktion eines Teilwegs führt.
4. Füge l-r- und hor-Lagebeziehungen zwischen Knoten gleicher Tiefe⁸ in benachbarten Teilwegen sowie zwischen den temporären Knoten entsprechend Abbildung 8.9(c) ein (l-r- und hor-

⁸Seien RW_1 und RW_2 zwei Teilwege mit v_{a_1} ist der erste Knoten von RW_1 und v_{a_2} ist der erste Knoten von RW_2 . Zwei Knoten v_1 und v_2 , $v_1 \in RW_1, v_2 \in RW_2$ haben gleiche Tiefe, wenn $|v_{a_1} \rightarrow^* v_1| = |v_{a_2} \rightarrow^* v_2|$ gilt, wobei $|u \rightarrow^* w|$ die Zahl der Knoten des Pfads $u \rightarrow^* w$ angibt.

Lagebeziehungen sind als gemeinsame Kante gestrichelt dargestellt). Dadurch werden gleiche Schritte verschiedener Durchläufe des offenen Zyklus auf gleichen Ebenen und in einer fortlaufenden Reihenfolge von links nach rechts platziert.

Nach Ende des VGL-Verfahrens werden die temporären Knoten, wie in Abbildung 8.9(d) dargestellt, durch eine Kante mit Kreisbögen ersetzt.

Man könnte den Knoten v_e jedes Teilwegs auch auf der aufsteigenden Kante platzieren. Dies lässt sich durch entsprechende Änderungen der temporären Knoten und Lagebeziehungen einfach erhalten. Die hier gewählte Darstellung wurde jedoch von Anwendern aus der Biochemie bevorzugt, da, abgesehen von den Rückkanten zwischen den Teilwegen, alle Reaktionen von oben nach unten verlaufen und die Darstellung leicht zu lesen ist.

Die Struktur von offenen Zyklen bleibt in einer Visualisierung auch dann erhalten, wenn der offene Zyklus Teil eines größeren Reaktionsnetzes ist, siehe Abbildung 8.10.

8.1.2.3.2 Geschlossene Zyklen

Vorausgesetzt sei ein Reaktionsweg RW, der als geschlossener Zyklus Teil eines Reaktionsnetzes ist und für den gilt:

1. RW umfasst mindestens zwei aufeinander folgende Reaktionsschritte und ist in zwei Teilwege RW_1 und RW_2 unterteilt, wobei RW_1 mit einer Substanz endet.
2. Kein Teil von RW ist in einem anderen offenen oder geschlossenen Zyklus enthalten.⁹

Sei v_a der erste und v_e der letzte Knoten des Teilwegs RW_1 , der Knoten v_a sei zugleich der letzte Knoten von RW_2 . Sei v_m der erste Knoten des Teilwegs RW_2 . Das Einfügen zusätzlicher Knoten und spezieller Lagebeziehungen für geschlossene Zyklen geschieht wie folgt (siehe auch Abb. 8.11), dabei haben wieder alle hier eingefügten Lagebeziehungen ein höheres Gewicht als die vom Anwender zusätzlich vorgegebenen Lagebeziehungen:

1. Füge o-u-Lagebeziehungen zwischen allen aufeinander folgenden Knoten in RW_2 in entgegengesetzter Richtung sowie zwischen v_m und v_e ein. Markiere alle Reaktionsknoten aus RW_2 als umgedreht. Dies entspricht einem Umdrehen der Reaktionen des aufwärts gerichteten Teilwegs RW_2 . Füge o-u-Lagebeziehungen zwischen aufeinander folgenden Knoten von RW_1 ein (siehe Abb. 8.11(a), o-u-Lagebeziehungen sind gestrichelt dargestellt). Durch die Lagebeziehungen wird garantiert, dass auch bei Berücksichtigung weiterer Reaktionen, an denen Teile des geschlossenen Zyklus beteiligt sind, für die Teilwege des geschlossenen Zyklus die vorgegebene Richtung erhalten bleibt.
2. Füge ver-Lagebeziehungen zwischen aufeinander folgenden Knoten in $RW_1 \setminus \{v_a, v_e\}$ und in umgekehrter Richtung zwischen aufeinander folgenden Knoten in $RW_2 \setminus \{v_a\}$ ein (siehe Abb. 8.11(b), ver-Lagebeziehungen sind als durchgezogene Pfeile dargestellt). Wie in offenen Zyklen wird dadurch garantiert, dass die Knoten der Teilwege untereinander platziert werden, auch wenn auf Grund weiterer Reaktionen mehr als eine Kante zu einer Substanz oder Reaktion eines Teilwegs inzident ist.

⁹Wie bei offenen Zyklen ist diese Voraussetzung für die Verwendung von ver-Lagebeziehungen nötig und entspricht den Gegebenheiten in biochemischen Reaktionsnetzen.

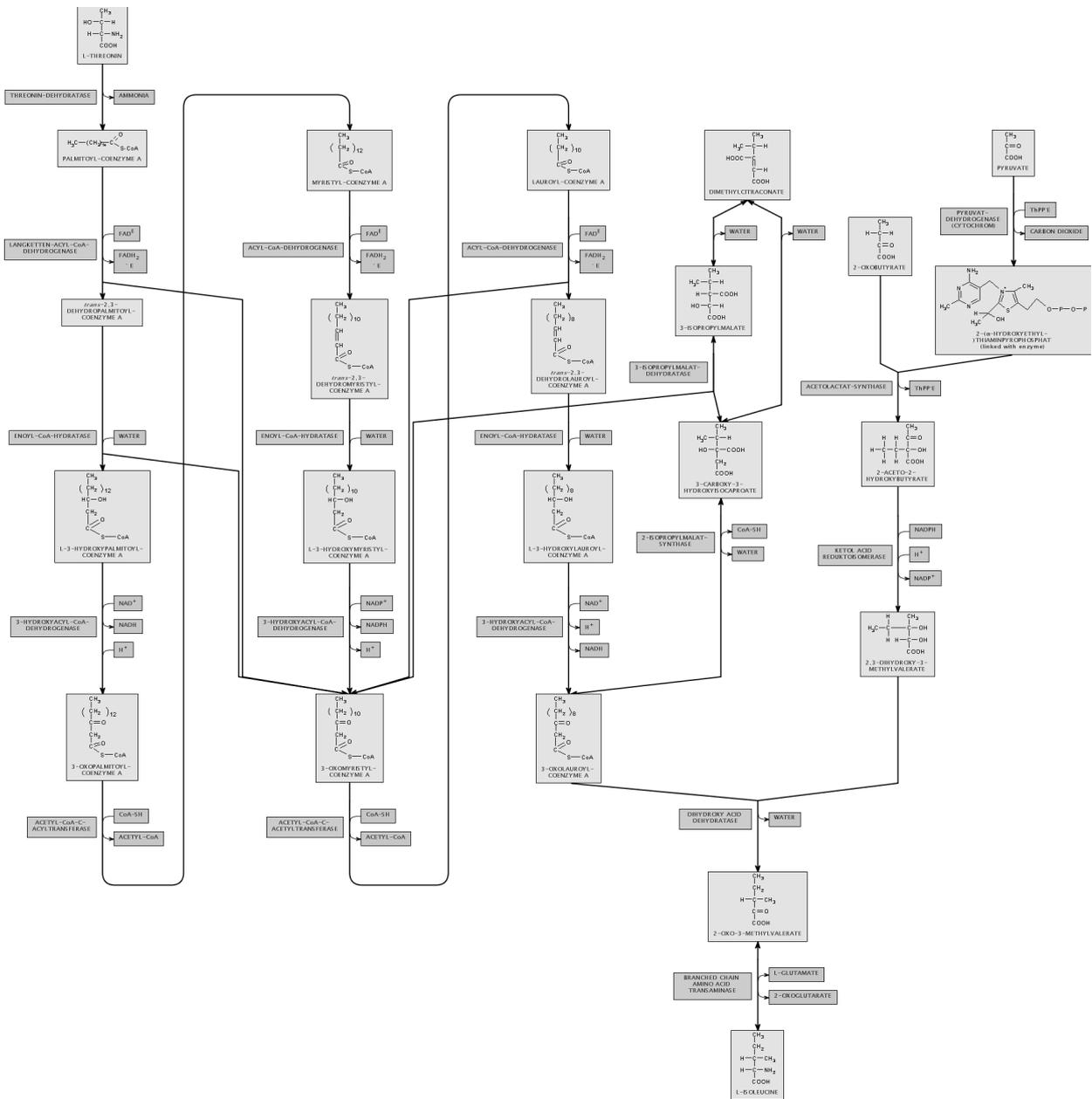


Abbildung 8.10: Offene Zyklen als Teil von Reaktionsnetzen. Darstellung des offenen Zyklus aus Abbildung 8.9 als Teil eines (in der Natur nicht existierenden) biochemischen Reaktionsnetzes.

8 Zeichnungen biochemischer Reaktionsnetze

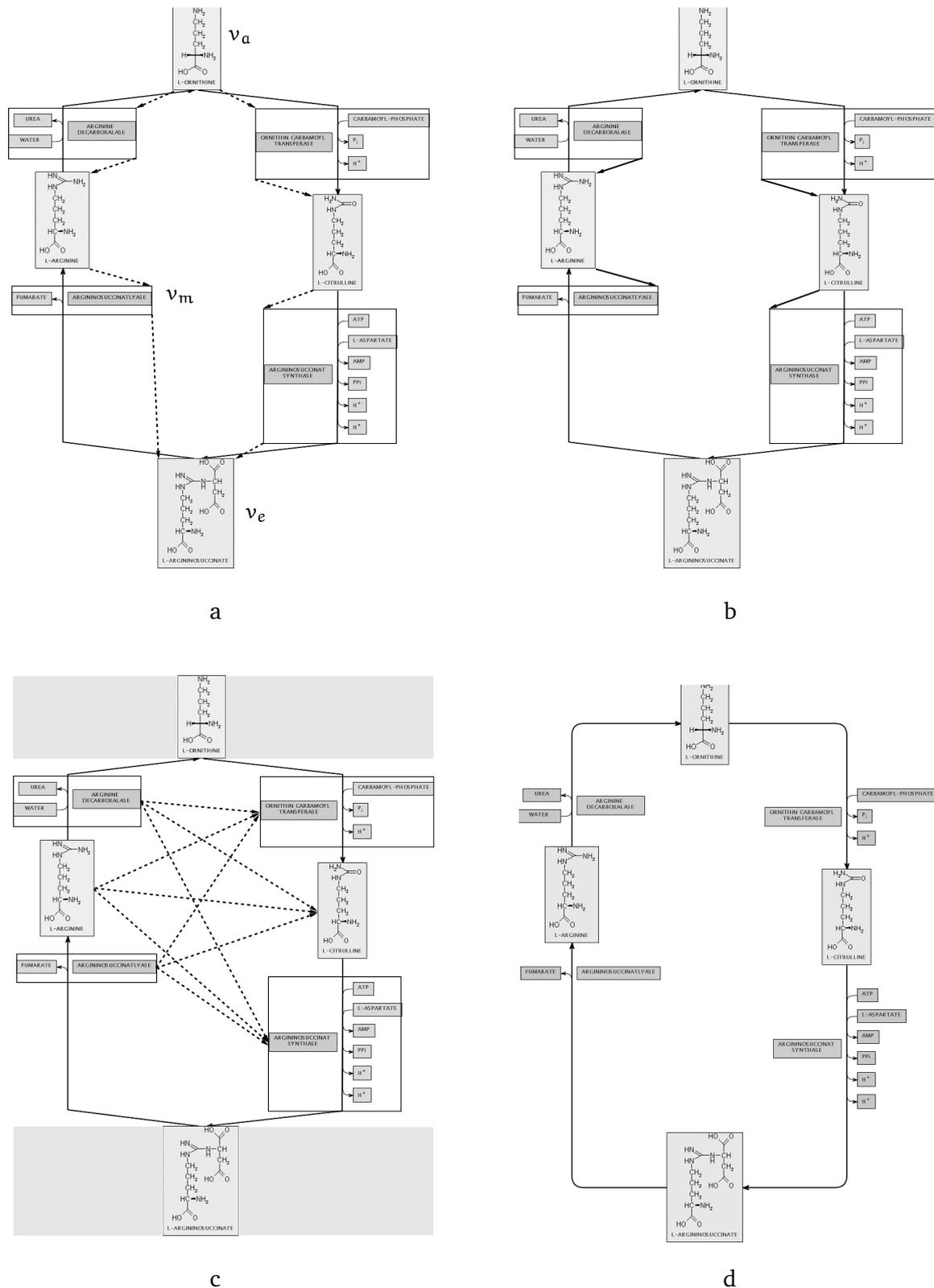


Abbildung 8.11: **Behandlung geschlossener Zyklen.** Die Erklärung der in (a) bis (d) dargestellten Schritte erfolgt im Abschnitt 8.1.2.3.2.

3. Füge l-r-Lagebeziehungen von allen Knoten aus $RW_2 \setminus \{v_a\}$ zu allen Knoten aus $RW_1 \setminus \{v_a, v_e\}$ ein (siehe Abb. 8.11(c)). Dies garantiert die Richtung des Zyklus im Uhrzeigersinn.¹⁰ Da bei geschlossenen Zyklen keine gleichen Ebenen gefordert werden, ist wegen der unterschiedlichen Knotengröße das Einfügen aller l-r-Lagebeziehungen nötig.

Zusätzlich wird die Berechnung der x -Koordinaten angepasst: Bei der Bestimmung der x -Koordinaten von v_a bzw. v_e beeinflussen jeweils nur die beiden Kanten zu Knoten der Reaktionswege RW_1 und RW_2 die Positionierung, alle anderen Kanten inzident zu v_a und v_e werden nicht beachtet. Dadurch wird die Zentrierung der Knoten v_a und v_e im Bezug zu den Teilwegen garantiert. Zudem wird der seitliche Mindestabstand der beiden Knoten v_a und v_e so angepasst, dass er jeweils bis zur aktuellen minimalen x_1 -Koordinate von Knoten aus RW_2 bzw. zur maximalen x_r -Koordinate von Knoten aus RW_1 geht (siehe Abb. 8.11(c) der graue Bereich). Horizontale Kantenstücke überlappen später nicht mit Knoten, da bei der Berechnung der x -Koordinaten keine Knoten in die grauen Bereiche platziert werden.

Nach Ende des VGL-Verfahrens werden einige Kanten durch Kanten mit Kreisbögen ersetzt, siehe Abbildung 8.11(d). Wie bei offenen Zyklen bleibt auch bei geschlossenen Zyklen deren Struktur erhalten, wenn der geschlossene Zyklus Teil eines größeren Reaktionsnetzes ist, siehe Abbildung 8.12.

8.1.3 Kontexterhaltende Navigation und Sichten

Verfeinerungen und Vergrößerungen von biochemischen Reaktionsnetzen und spezielle Sichten auf hierarchische Reaktionsgraphen lassen sich durch die in Kapitel 5 beschriebenen Operationen erhalten. Bei der Visualisierung der dabei entstehenden Reaktionsgraphen sollen neu eingefügte Teile im Kontext des umgebenden Reaktionsnetzes dargestellt werden.

Kontexterhaltende Navigation und Sichten erreicht man durch stabile Zeichenverfahren, also durch Verwendung von Lagebeziehungen wie in Abschnitt 7.6 beschrieben.

Ein Beispiel soll dies demonstrieren, siehe Abbildungen 8.13-8.15. Sei dazu $HRG = (RG, B)$ der in Abbildung 8.13 dargestellte hierarchische Reaktionsgraph. In Abbildung 8.14 ist eine Knotenmenge U markiert und in Abbildung 8.15(a) der durch die Menge U gegebene Ausschnitt $Ausschnitt(U)$ von HRG dargestellt. Die Visualisierung in Abbildung 8.15(a) ist zugleich die Ursprungszeichnung für weitere, durch Navigation oder Sichten erhaltene Darstellungen.

Durch Vergrößerung und Verfeinerung werden neue Reaktionsgraphen bestimmt. In ihnen werden zwischen Knoten, die schon in der Ursprungszeichnung enthalten sind, entsprechend Abschnitt 7.6.3 o-u- und l-r-Lagebeziehungen eingefügt. Diese dienen dem Erhalt der relativen Lagebeziehungen zwischen den Knoten.

Hier wird auf den Beispielgraphen in Abbildung 8.15(a) die Verfeinerung des Knotens v (siehe Abb. 8.14) angewandt, das Ergebnis der Visualisierung ist in Abbildung 8.15(b) dargestellt. Eine anschließend durch Anwenden einer einheitlichen Sicht (Darstellung aller Reaktionen mit Enzymen, aber ohne Cosubstanzen) erreichte Visualisierung zeigt Abbildung 8.15(c). Sichten entstehen durch eine Bearbeitung des Reaktionsgraphen, bei der vor dem Berechnen der Platzierung bestimmte Knoten, z. B. alle Knoten vom Typ *Cosubstanz*, entfernt werden. Auch hier wird die Ähnlichkeit zur Ursprungszeichnung durch Verwendung von o-u- und l-r-Lagebeziehungen erreicht.

¹⁰Falls stattdessen eine entgegengesetzte Richtung bevorzugt wird, sind die l-r-Lagebeziehungen in entgegengesetzter Richtung einzufügen und in Punkt 1 die o-u-Lagebeziehungen entsprechend anzupassen.

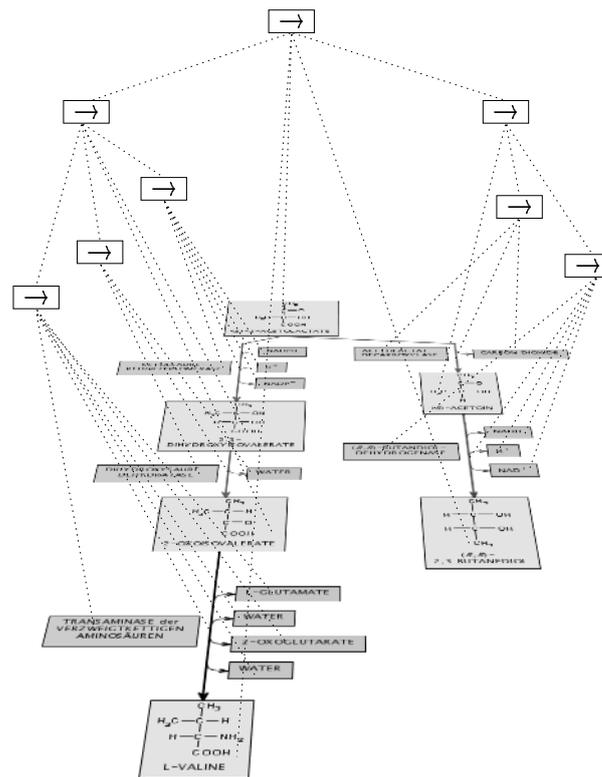


Abbildung 8.13: Kontexterhaltende Navigation und Sichten (Teil 1). Beispiel eines hierarchischen Reaktionsgraphen.

8.2 Anwendungen

8.2.1 Erkundung des Stoffwechsels

Das Visualisierungsverfahren wird zur Darstellung von einzelnen biochemischen Reaktionen, von Stoffwechselwegen (z. B. Glycolyse) und von biochemischen Reaktionsnetzen (z. B. der Gesamtheit der Reaktionswege zwischen zwei Substanzen) verwendet. Diese Anwendungen sind auch im *Bio-Path*-Projekt [Bio01] realisiert. Darstellungen von Reaktionsnetzen sowie interaktives Erkunden des Stoffwechsels in Zellen sind die zentralen Anwendungen dieses Zeichenverfahrens.

Daneben ist eine Reihe weiterer Anwendungen denkbar, bei denen jedoch oft spezielle Anordnungen von Teilen eines Reaktionsnetzes gewünscht sind. Viele dieser speziellen Anwenderwünsche, z. B. andere zeitliche oder links-rechts-Anordnungen von Reaktionswegen, die Platzierung ausgewählter Substanzen aus benachbarten Reaktionswegen auf derselben Ebene oder die Platzierung von Substanzen auf der ersten oder letzten Ebene, lassen sich mittels Lagebeziehungen realisieren. Hier werden drei weitere Anwendungen vorgestellt: Der Vergleich von Reaktionswegen, die Verfolgung von Atomen und quantitative Darstellungen.

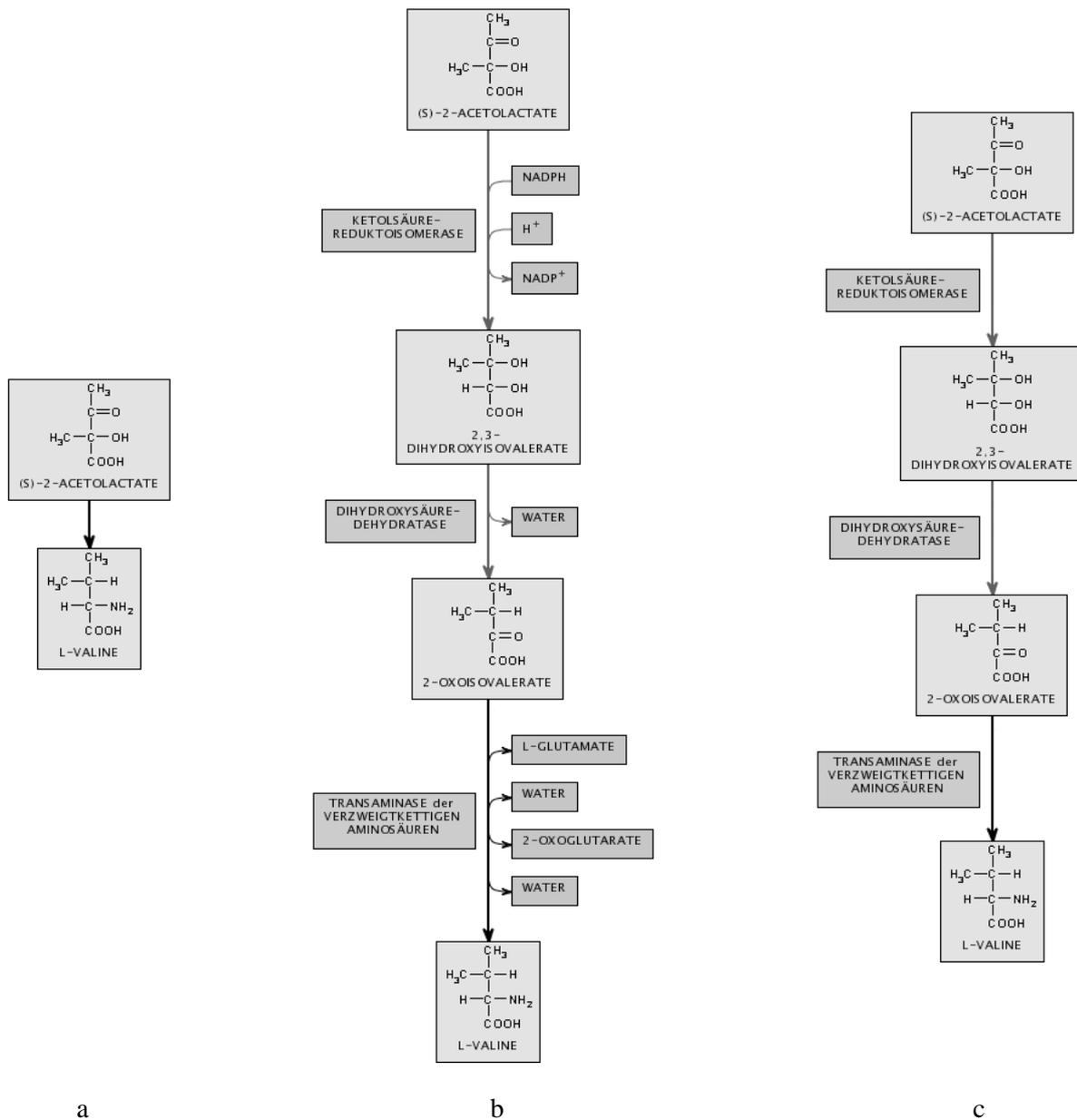


Abbildung 8.15: **Kontexterhaltende Navigation und Sichten (Teil 3)**. (a) Der durch die Menge U gegebene Ausschnitt aus dem hierarchischen Reaktionsgraphen aus Abbildung 8.14. (b) Reaktionsgraph, der durch Verfeinerung des Knotens v (Abb. 8.14) entsteht. Die Anordnung der in der originalen Zeichnung vorhandenen Objekte bleibt durch Verwendung von o - u - und l - r -Lagebeziehungen erhalten. (c) Eine spezielle Sicht, die durch temporäres Entfernen aller Knoten mit Typ *Cosubstanz* entsteht. Auch bei Sichten dienen o - u - und l - r -Lagebeziehungen zum Erhalt der Anordnung. Eine kontexterhaltende Erweiterung der Visualisierung ist in Abbildung 3.9 auf Seite 53 dargestellt.

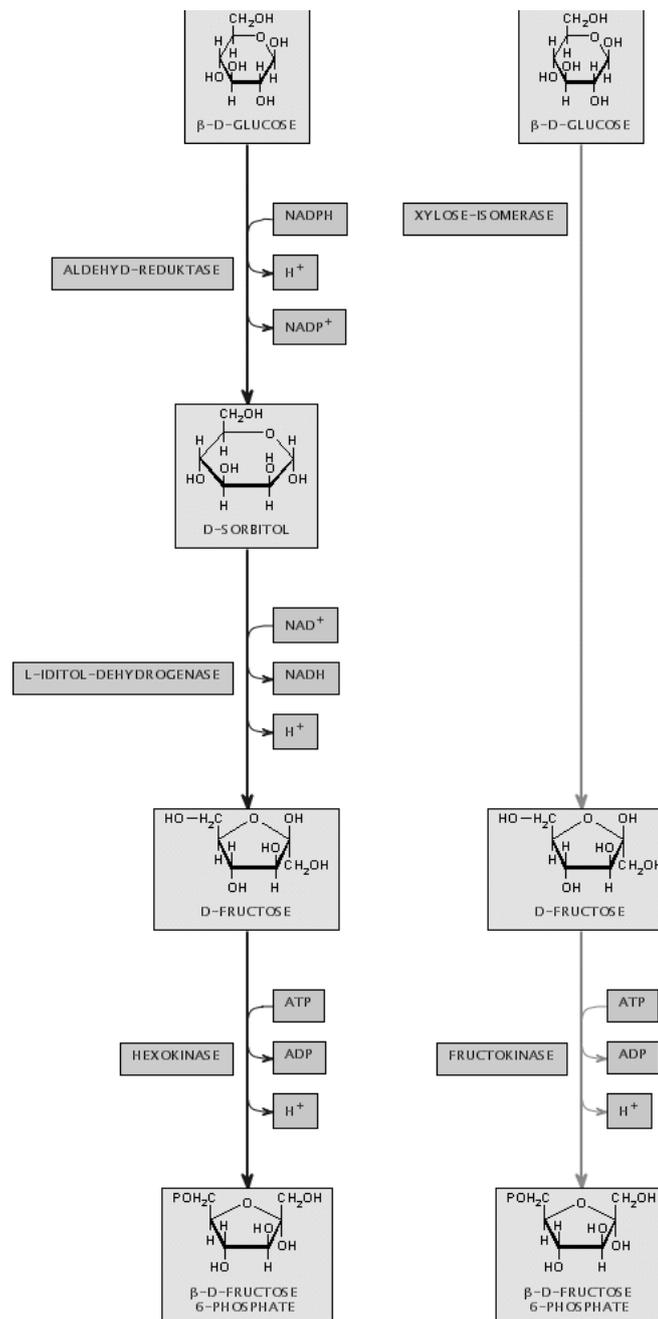


Abbildung 8.16: **Vergleich von Reaktionswegen.** Umwandlung von *Glucose* in *Fructose-6-Phosphat*, links der Weg in Tieren, rechts der Weg in Pflanzen. Beide Reaktionswege wurden gleichzeitig durch das Zeichenverfahren bearbeitet. Gleiche Substanzen, beispielsweise *Fructose*, wurden durch Verwendung von hor-Lagebeziehungen auf derselben Ebene platziert.

tersuchung unterstützen, indem Substanzen, welche das zu verfolgende Atom enthalten, sowie die verbindenden Reaktionen mittels ver-Lagebeziehungen untereinander platziert werden (siehe Abb. 8.17).

8.2.4 Quantitative Darstellungen

Bereits in der Einleitung wurden DNA-Chips als wichtiges Mittel zur Untersuchung des Zellstoffwechsels beschrieben (siehe Seite 7). Dabei werden durch Bestimmung der Menge der aktuell transkribierten Gene Aussagen über den Stoffwechsel getroffen. Dies wird beispielsweise verwendet, um den Stoffwechsel von Zellen unter verschiedenen Umweltbedingungen miteinander zu vergleichen. Grundlage der Darstellung ist die Beziehung, dass viele Gene Enzyme kodieren und Enzyme Reaktionen katalysieren. Eine aussagekräftige Darstellung der Ergebnisse erreicht man, wenn die Breite der Reaktionskante an die Stärke der Aktivität des korrespondierenden Gens angepasst wird, siehe Abbildung 8.18. Breite Reaktionen bzw. Reaktionswege entsprechen besonders wichtigen Reaktionen bzw. Stoffwechselwegen im aktuellen Zellstoffwechsel. Man könnte den Hauptweg zusätzlich durch Verwendung von ver-Lagebeziehungen hervorheben.

8.3 Regulation und deren Darstellung

8.3.1 Einführung

Zellen können nicht nur viele verschiedene Produkte gleichzeitig synthetisieren, sie produzieren diese auch in den Mengenverhältnissen, in denen einzelne Moleküle gebraucht werden. Fast alle biochemischen Reaktionen in Zellen werden von Enzymen katalysiert. Da Enzyme durch die Reaktion nicht verändert werden und nach der Reaktion im Allgemeinen sofort eine weitere Reaktion katalysieren können, muss ihre Aktivität entsprechend den Bedürfnissen der Zellen reguliert, also erhöht oder verringert werden. Dies wird als *Regulation* von Reaktionen bzw. von Reaktionsnetzen bezeichnet.

Regulation geht weit über die bisher betrachteten biochemischen Reaktionsnetze hinaus. Im Folgenden werden deshalb die Grundlagen der Regulation biochemischer Reaktionen vorgestellt, anschließend werden Erweiterungen zu deren Darstellung untersucht.

Bei der Regulation werden zwei wesentliche Mechanismen unterschieden: Bei der *schnellen Regulation* wird das Enzym, welches die Reaktion katalysiert, durch Aktivierung oder Inhibition direkt beeinflusst. *Langsame Regulation* verändert dagegen die Konzentration des Enzyms in der Zelle durch Beeinflussung seines Auf- und Abbaus.

Die Regulation der Enzymaktivität erfolgt auf mehreren Ebenen (siehe auch Abb. 8.19). Der erste, mit einer Reaktionszeit von Stunden bis Tagen aber auch langsamste, Angriffsort findet sich beim Abschreiben der Geninformation, die das Enzym kodiert (Transkription) und während der Bildung des Enzyms (Translation). Diese auf der so genannten Transkriptions- und Translationsebene ablaufende langsame Regulation beeinflusst die Menge des Enzyms in der Zelle. Der nächste Angriffsort ist das Enzym selbst, dessen Aktivität erhöht oder verringert werden kann. Die dabei wirkenden Mechanismen gehören zur schnellen Regulation und beeinflussen die Enzymaktivität innerhalb von Sekunden bis Minuten. Der letzte Angriffsort ist die Regulation der Abbaurate des Enzyms. Dieser Mechanismus zählt zu den langsamen Regulationsmechanismen und beeinflusst, wie der erste

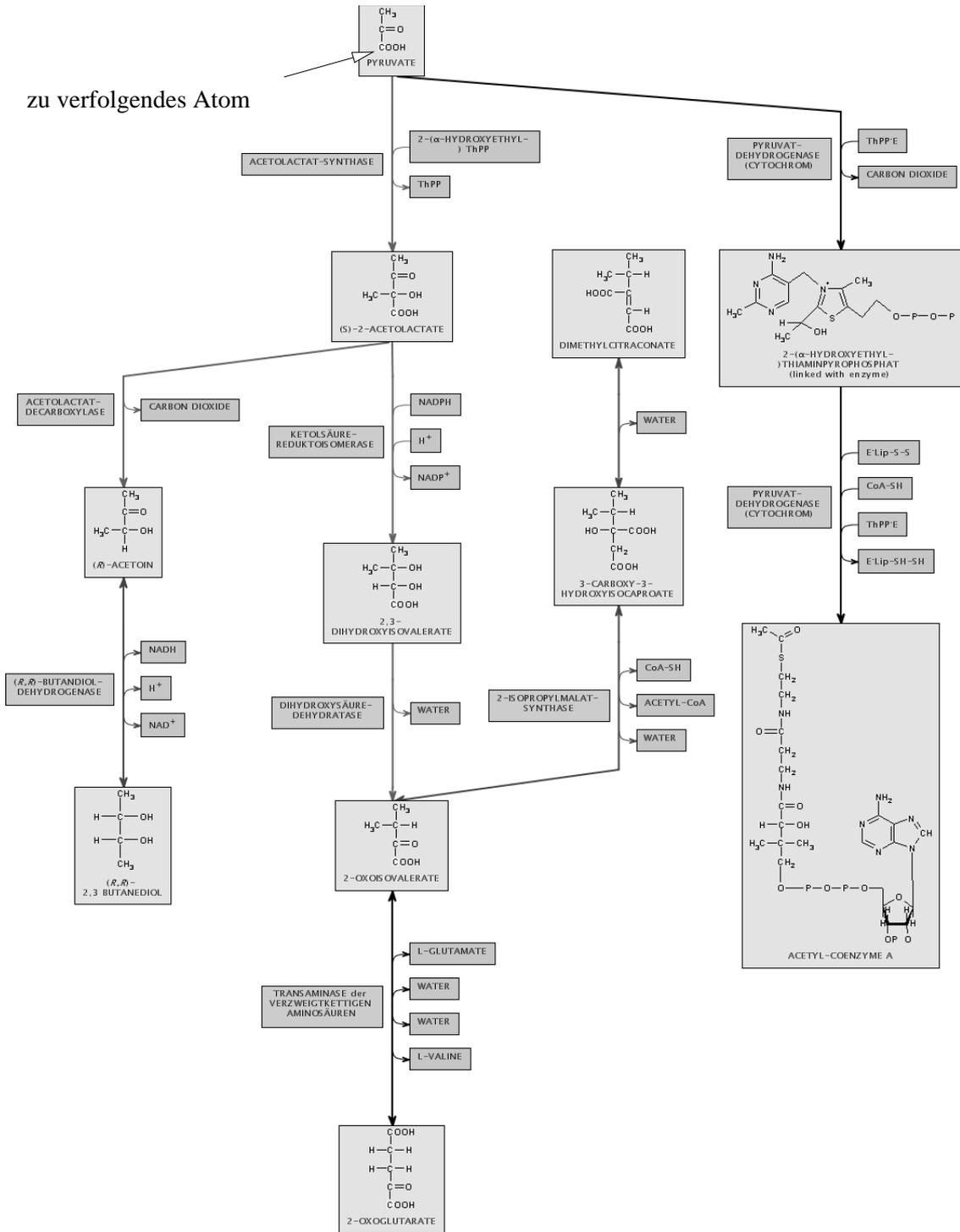


Abbildung 8.17: **Verfolgung von Atomen.** Darstellung eines Reaktionsnetzes, das zu verfolgende Kohlenstoff-Atom der COOH-Gruppe ist in den Molekülen weiß markiert (siehe z. B. Pyruvat). Substanzen, welche das Atom enthalten, und deren verbindende Reaktionen sind untereinander platziert.

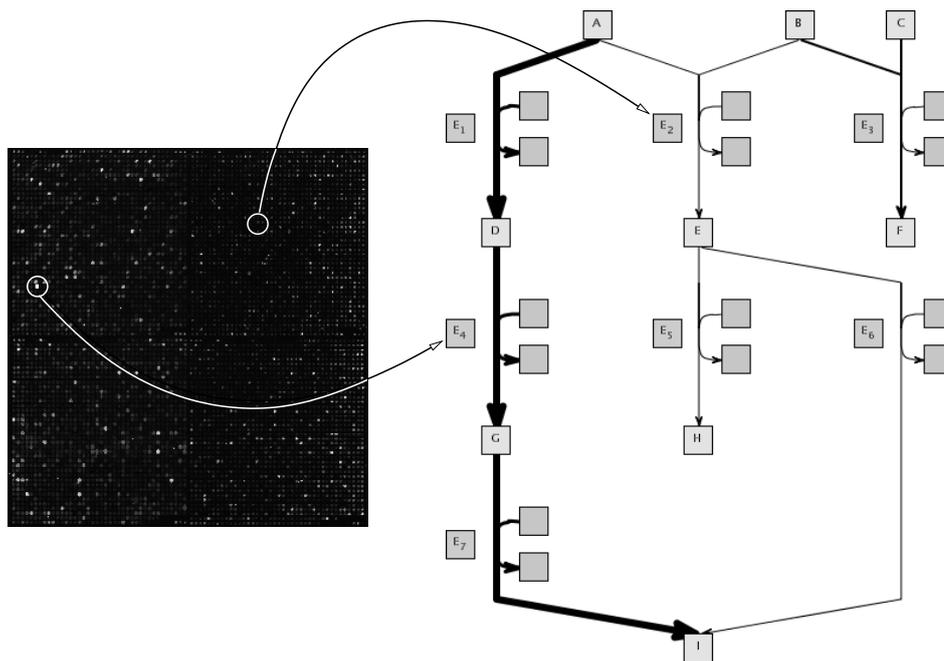


Abbildung 8.18: **Quantitative Darstellungen.** (links) DNA-Chip, helle Bereiche weisen indirekt auf eine hohe, dunkle Bereiche auf eine geringe Enzymkonzentration hin. (rechts) Die Enzymkonzentration wirkt sich auf den Stoffumsatz der katalysierten Reaktion aus, dieser wird durch die Dicke der Kanten repräsentiert.

Schritt, die Menge des Enzyms in der Zelle.¹¹

Besondere Bedeutung in biochemischen Reaktionsnetzen hat die schnelle Regulation, bei der eine Substanz, der *Regulator*, direkt auf das Enzym wirkt und dadurch zu einer Erhöhung oder Verringerung der Enzymaktivität führt. Es gibt verschiedene Mechanismen zur direkten Regulation der Enzymaktivität, beispielsweise durch reversible Bindung des Regulators an das aktive Zentrum des Enzyms (*kompetitive Hemmung*). Im Rahmen dieser Betrachtungen wird keine Unterscheidung zwischen den verschiedenen Mechanismen der schnellen Regulation getroffen, sondern alle Mechanismen werden in der Visualisierung gleichartig dargestellt. Unterschieden wird nur zwischen Erhöhung (*Förderung*) und Verringerung (*Hemmung*) der Enzymaktivität. Die Angriffsorte der Regulation werden als *Regulationsstellen* bezeichnet.

Besonders oft wird die Regulation für einzelne Stoffwechselwege untersucht. Um zu verhindern, dass sich bei der Synthese von Substanzen die Zwischenprodukte der Stoffwechselwege unnötig anhäufen, wird durch die Natur im Allgemeinen die erste Reaktion eines Stoffwechselwegs reguliert. Oft hemmt dabei das Produkt einer späteren oder der letzten Reaktion die Aktivität des Enzyms der ersten Reaktion dieses Stoffwechselwegs; dieser Mechanismus wird als *Rückkopplungs-*

¹¹Neben diesen allgemeinen Regulationsmechanismen gibt es weitere, die jedoch an spezielle Zellen oder Enzyme gebunden sind. Ein Beispiel ist die Zymogenaktivierung: Zymogene sind inaktive Vorstufen von Enzymen, die erst, nach gesteuerter Freisetzung, am Wirkort in Enzyme umgebaut werden.

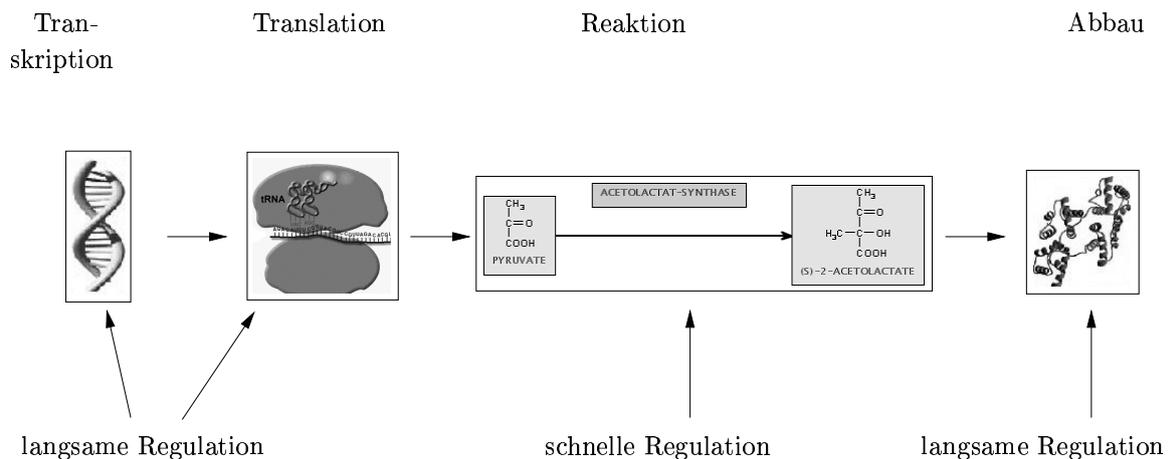


Abbildung 8.19: **Mechanismen der Regulation.** Weg eines Enzyms von der Entstehung (links) bis zum Abbau (rechts) mit verschiedenen Regulationsmechanismen.

hemmung bezeichnet. Abbildung 8.20(a) stellt eine solche Rückkopplungshemmung im Stoffwechselweg *Synthese von Valin* dar. Einige Reaktionen unterliegen zudem auch der *Vorwärtsförderung*, z. B. beschleunigt *Cholesterin* seinen eigenen Abbau.

Bisher gibt es nur wenige Daten über die Regulation biochemischer Reaktionen. Doch bereits jetzt ist absehbar, dass die überwiegende Zahl der Verzweigungsstellen in Reaktionsnetzen reguliert wird [ABL⁺97, Mic99], so dass man von relativ komplexen Regulationen ausgehen kann, wie dies in Abbildung 8.20(b) angedeutet wird.

8.3.2 Darstellung von Regulation

Die Regulation biochemischer Reaktionen ist ein Aspekt, der für das Verständnis und die Beeinflussung biochemischer Reaktionsnetze zunehmend an Bedeutung gewinnt. Nicht die möglichen Reaktionswege entscheiden letztendlich über den Stoffwechsel in einer Zelle, sondern die aktiven Reaktionen. Reaktionsregulation wird nur in wenigen der in Kapitel 4 untersuchten Visualisierungen dargestellt und ist kein allgemeiner Bestandteil von Visualisierungen biochemischer Reaktionsnetze. Auf Grund der wachsenden Bedeutung soll hier die Erweiterung des Visualisierungsverfahrens betrachtet werden. Dabei werden zwei Ansätze unterschieden:

1. Eine Darstellung der lokalen Wirkung des Regulators, bei der die regulierende Substanz neben das regulierte Enzym (bzw. die regulierte Reaktion) platziert wird. Diese Darstellung ist ähnlich zu [Mic93, Mic99].
2. Eine Visualisierung der globalen Wirkung des Regulators, bei der eine zusätzliche Kante von der Substanz zum regulierten Enzym (bzw. zur regulierten Reaktion) eingefügt wird. Diese Darstellung wird oft für Regulationsdarstellungen innerhalb eines Stoffwechselwegs, insbesondere für die Darstellung von Rückkopplungshemmung verwendet und orientiert sich an einigen Darstellungen in [Mic99, ABL⁺97].

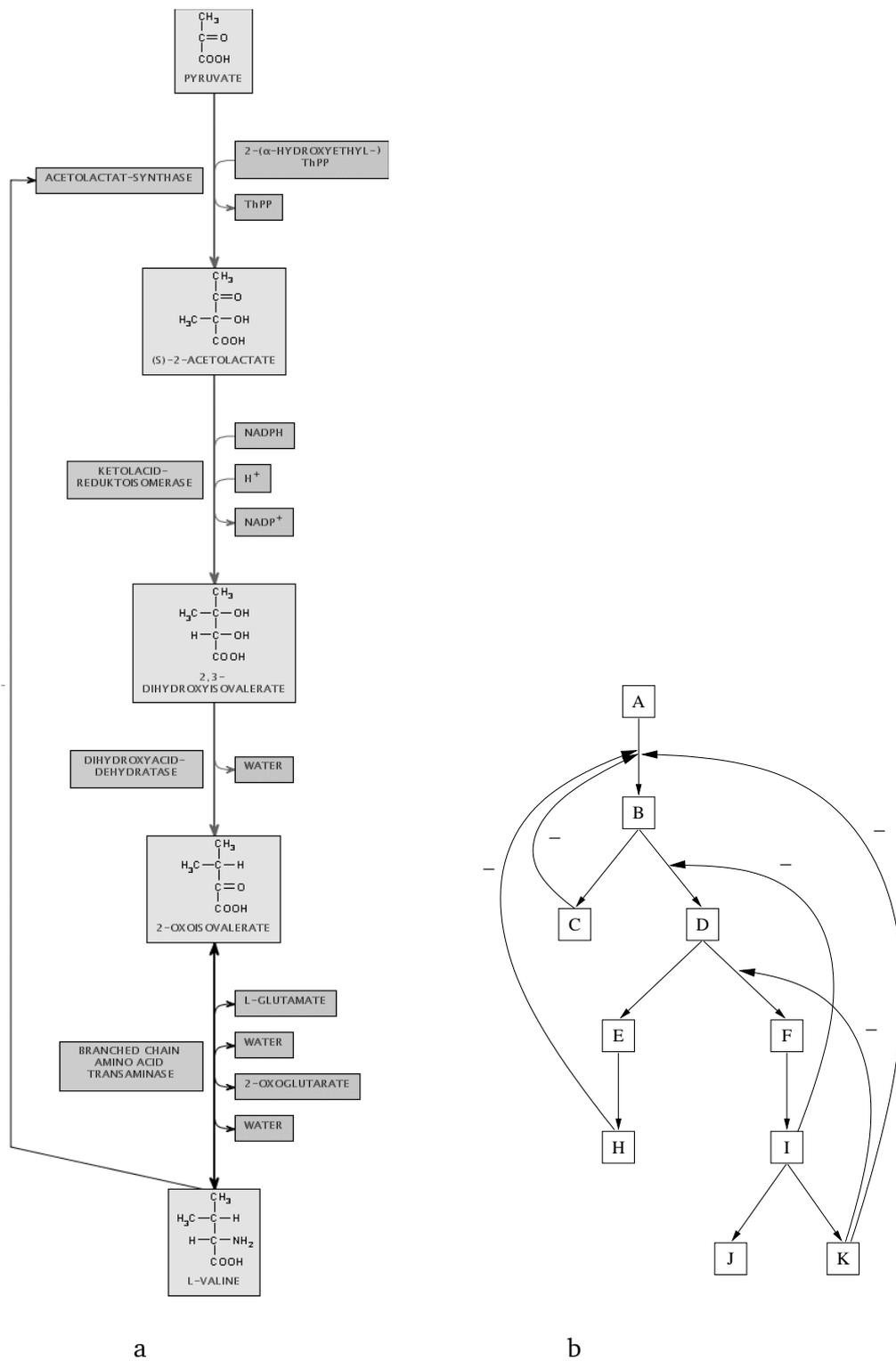


Abbildung 8.20: **Rückkopplungshemmung.** (a) Der Stoffwechselweg *Synthese von Valin* von *Pyruvat* zu *Valin* unterliegt der Rückkopplungshemmung. (b) Multiple Rückkopplungshemmung von Reaktionen in einem Reaktionsnetz, ähnlich zu [ABL⁺97].

In beiden Fällen wird für die Kante, welche die regulierende Substanz und den Angriffsort verbindet, die Konvention aus [Mic93, Mic99] verwendet: Durchgezogene Linie für schnelle Regulation, gestrichelte Linie für langsame Regulation, Kantenbenennung + für Aktivierung, Kantenbenennung – für Inhibitierung. Auch wenn die langsame Regulation nicht direkt am Enzym angreift, sondern nur indirekt dessen Menge beeinflusst, enden in Darstellungen die Regulationskanten oft am Enzym oder der Reaktion.

8.3.2.1 Lokale Darstellung von Regulation

Bei dieser Art der Visualisierung werden die regulierenden Substanzen wie in Abbildung 8.21(a) als zusätzliche Knoten neben dem Enzym dargestellt. Bei der Berechnung der Platzierung werden diese, wie Cosubstanzen und Enzyme, in dem vergrößerten Knoten für die Reaktion berücksichtigt. Die Größe dieses Knotens wird entsprechend angepasst.

Die Abbildung 8.21(b) zeigt einen regulierten Reaktionsweg mit lokaler Darstellung der Regulation. Auch andere Sichten sind möglich, beispielsweise können im Falle einer Sicht ohne Enzyme die Pfeile direkt auf die Reaktion führen, wie dies in Abbildung 8.21(c) dargestellt ist.

8.3.2.2 Globale Darstellung von Regulation

Besonders Rückkopplungshemmung innerhalb eines Stoffwechselwegs wird manchmal wie in Abbildung 8.20(a) dargestellt. Dies lässt sich mittels der in Abschnitt 7.1.5 eingeführten seitlichen Ports realisieren. Die Kante von der Substanz zum Enzym endet an einem seitlichen Port des Enzym-Knotens. Für Regulatoren, die nicht als Substanz im Reaktionsnetz vorkommen, wird eine lokale Darstellung wie in Abschnitt 8.3.2.1 mit Visualisierung des Regulators direkt neben dem Enzym gewählt. Auch hier können andere Sichten gewählt werden, beispielsweise eine Sicht ohne Enzyme, die Kante endet dann am Reaktionsknoten.

Mit den beiden beschriebenen Möglichkeiten lassen sich sowohl lokale wie globale Darstellungen der Regulation erhalten. Das Modell aus Kapitel 5 kann allerdings nicht mehr verwendet werden, da der bei der globalen Darstellung entstehende Graph nicht mehr bipartit ist. Es gibt nun Kanten zwischen Substanzen und Enzymen, die beide im Reaktionsgraph $RG = (K \cup R, A \cup Z, B, T)$ Elemente der Menge K sind. In *BioPath* sind Regulationsdarstellungen ebenfalls nicht realisiert, da dort die nötigen Daten über Regulation biochemischer Reaktionen nicht vorhanden sind.

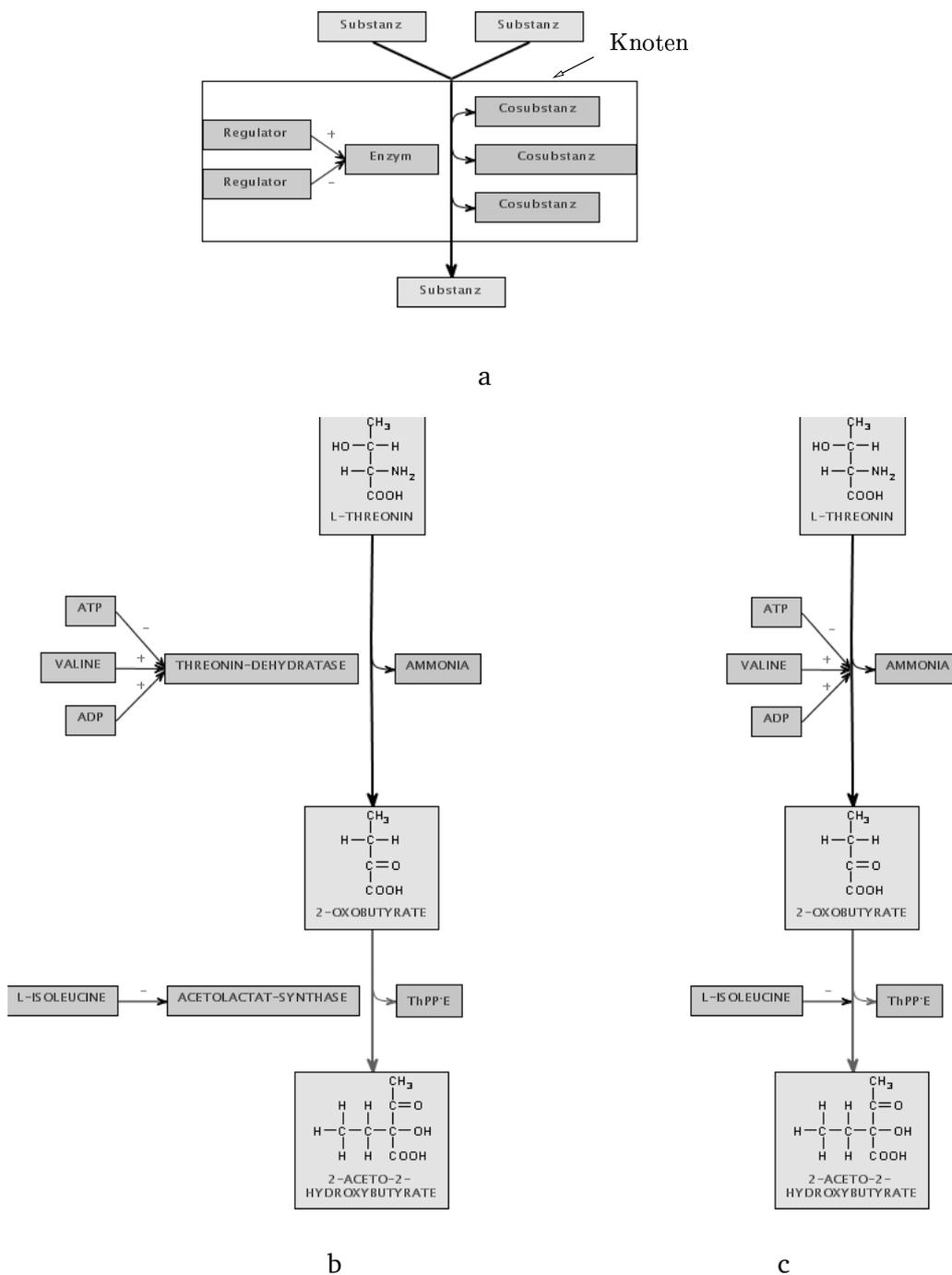
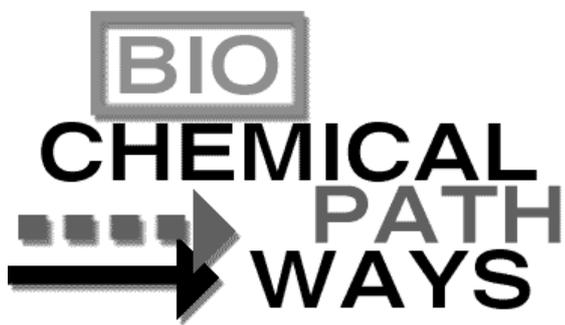


Abbildung 8.21: **Lokale Darstellung der Regulation.** (a) Die regulierenden Substanzen sind weitere Knoten, die bei der Bildung des temporären großen Knotens berücksichtigt werden. (b) Darstellung eines regulierten Reaktionswegs, (c) Sicht ohne Enzyme, die Regulationskanten enden an der Reaktion.

9 Zusammenfassung und Ausblick



9.1 Zusammenfassung	238
9.2 Ausblick	240

Das Logo des *BioPath*-Projekts, der Realisierung der in dieser Arbeit beschriebenen Verfahren zur Visualisierung biochemischer Reaktionsnetze (siehe [Bio01]).

9.1 Zusammenfassung

9.1.1 Inhalt der Arbeit

In dieser Arbeit wurde die Visualisierung biochemischer Reaktionsnetze behandelt. Die Arbeit ist dazu in zwei Teile gegliedert: Die Analyse der Darstellungsanforderungen und die Modellierung biochemischer Reaktionsnetze im ersten Teil und die Entwicklung eines Zeichenverfahrens sowie dessen Anwendung zur Visualisierung der Reaktionsnetze im zweiten Teil.

9.1.1.1 Einführung und Teil I

Die Bedeutung der Visualisierung biochemischer Reaktionsnetze wurde in Kapitel 1 betrachtet. Dabei wurde festgestellt, dass Darstellungen dieser Netze entscheidend für das Verständnis biochemischer Prozesse sind. Interaktives Erkunden der Information und der ständige Wissenszuwachs erfordern automatische Visualisierungsverfahren.

Nach Einführung biochemischer und informatischer Grundlagen in Kapitel 2 beschäftigte sich Kapitel 3 mit Darstellungsanforderungen. Diese beruhen im Wesentlichen auf der Auswertung traditioneller Darstellungen in Büchern und elektronischen Systemen, auf Untersuchungen zur Eignung verschiedener Graph-Zeichenverfahren und auf Befragungen von Anwendern. Trotz der Vielzahl verschiedenster traditioneller Darstellungsformen gelang es, die Anforderungen an die Visualisierungen zu vereinheitlichen. Im Ergebnis konnten die Visualisierungsanforderungen in drei Punkten zusammengefasst werden: Anforderungen an die lokale Darstellung von Reaktionen, Anforderungen an die globale Darstellung von Reaktionen und Anforderungen an kontexterhaltende Navigation und Sichten.

In Kapitel 4 wurde untersucht, wie existierende Zeichenverfahren die Anforderungen erfüllen. Fazit der Analyse war, dass traditionelle Verfahren nicht ausreichend sind; die Entwicklung eines neuen Zeichenverfahrens ist also notwendig.

Der erste Teil der Arbeit schließt mit Betrachtungen zur Modellierung biochemischer Reaktionsnetze in Kapitel 5. Es wird eine Modellierung vorgestellt, die die Struktur der Netze widerspiegelt und die Unterteilung der Reaktionskomponenten repräsentiert. Aus einem Reaktionsnetz mit zugehöriger Hierarchisierung der Reaktionen können zudem beliebige Ausschnitte sowie Verfeinerung und Vergrößerung der Ausschnitte erzeugt werden. Die Modellierung ist Grundlage des Visualisierungsverfahrens und von Mechanismen für kontexterhaltende Navigation und Sichten.

9.1.1.2 Teil II

Der zweite Teil beschäftigte sich mit einem ebenenweisen Zeichenverfahren für Graphen und dessen Anwendung zur Visualisierung biochemischer Reaktionsnetze. In Kapitel 6 wurden die Berücksichtigung von echten Knotengrößen und von Lagebeziehungen als neue Anforderungen an ebenenweise Zeichenverfahren betrachtet und das Gesamtverfahren skizziert. Anschließend wurden die grundlegenden Schritte des Zeichenverfahrens vorgestellt. Zur Lösung der bei Kantenorientierung, Partitionierung in Ebenen und Kreuzungsreduzierung auftretenden Probleme wurden Algorithmen entwickelt und analysiert. Das Ergebnis sind neue Verfahren zur Lösung der Teilprobleme, die Knotengrößen und Lagebeziehungen berücksichtigen.

In Kapitel 7 wurden Erweiterungen des Zeichenverfahrens betrachtet, die dessen Anwendbarkeit in der Praxis verbessern. Dazu gehören die Berücksichtigung von Ports und von Benennungen der Knoten und Kanten sowie lokale Zeichenverfahren.

Schließlich beschäftigte sich Kapitel 8 mit der Anwendung des universellen Verfahrens zur Visualisierung biochemischer Reaktionsnetze. Neben den dabei nötigen Anpassungen, beispielsweise der Umsetzung von Darstellungsanforderungen bei offenen und geschlossenen Zyklen mittels Lagebeziehungen, wurden auch Anwendungen für Visualisierungen solcher Netze vorgestellt. Das entwickelte Zeichenverfahren erfüllt dabei alle drei Anforderungen an die Visualisierung biochemischer Reaktionsnetze.

9.1.2 Ergebnisse der Arbeit

In dieser Arbeit konnte gezeigt werden, dass sich komplexe biochemische Reaktionsnetze durch automatische Zeichenverfahren so gut visualisieren lassen, dass die erhaltenen Darstellungen die Anforderungen der Biologen erfüllen. Dieses Ergebnis war keineswegs zu erwarten, da sich bereits andere Gruppen mit der automatischen Darstellung solcher Netze beschäftigt haben, dabei jedoch nur unzulängliche Lösungen erreichten.

Großen Einfluss auf die Ergebnisse der Arbeit hatte die umfangreiche Analyse der Darstellungsanforderungen. Hier wurden erstmals konkrete Qualitätsanforderungen an Visualisierungen biochemischer Reaktionsnetze formuliert. Die herausgearbeiteten Anforderungen und das Visualisierungsverfahren stießen auch beim BioPathways Konsortium [BPC01], das sich mit Standards für Informationsdarstellungen in der Biochemie beschäftigt, auf großes Interesse und beeinflussen die aktuelle Diskussion [Sch01].

Das Visualisierungsverfahren ist Bestandteil des *BioPath*-Projekts, eines Informationssystems für biochemische Reaktionswege. Die dort erreichten Darstellungen zeichnen sich dadurch aus, dass die Qualität manuell erstellter Zeichnungen, wie sie in Büchern vorkommen, mit der Flexibilität dynamischer Verfahren vereint werden konnte. Von Biochemikern wurde das Verfahren als neue Qualität der Darstellung gewertet. So schreibt Michal in [Mic01]

„Die (...) gezeigte graphische Darstellung verschiedener Stoffwechselwege ist sehr übersichtlich und kann auch schwierige Prinzipien, wie unterschiedliche Verzweigungen und Parallelläufe, repräsentieren. Ebenso können Stoffwechselcyclen und wiederholtes Durchlaufen einer Sequenz von mehreren Reaktionen, wie sie z. B. bei der Fettsäuresynthese auftritt, angemessen dargestellt werden.“

Das im zweiten Teil vorgestellte ebenenweise Zeichenverfahren berücksichtigt Lagebeziehungen und stellt einen neuen Mechanismus zur Behandlung beliebiger Knotengrößen zur Verfügung. Lagebeziehungen dienen nicht nur der Anpassung der Visualisierung an spezielle Erfordernisse der Anwendung, wie dies hier z. B. bei der Darstellung offener und geschlossener Zyklen erfolgte. Sie sind auch bei der Erstellung schöner Zeichnungen hilfreich, da der Anwender dem Zeichenverfahren Verbesserungen der Platzierung vorschlagen kann.

Die Berücksichtigung der Knotengröße verbessert ebenfalls die Qualität der Zeichnungen im Gegensatz zu traditionellen ebenenweisen Zeichenverfahren. Wesentlich dabei ist die Verwendung der hier entwickelten lokalen Ebeneneinteilung, die kein herkömmliches Zeichenverfahren bietet. Der vorgestellte Mechanismus wurde so flexibel gestaltet, dass damit Knotengrößen sowohl im Sinne

lokaler Ebeneneinteilungen wie auch im Sinne traditioneller, globaler Ebeneneinteilungen berücksichtigt werden können.

9.2 Ausblick

Die Abläufe in Zellen sind sehr komplex. Biochemische Reaktionsnetze sind nur eine Komponente des Netzes aller Wechselwirkungen. Sie hängen mit Signalübertragungsnetzen¹ und genregulatorischen Netzen² zusammen. Während biochemische Reaktionsnetze schon seit langem untersucht werden, beschäftigt man sich mit den anderen Netzen noch nicht so lange. Insbesondere existieren für diese Netze Darstellungskonventionen bisher nur in Ansätzen.

Signalübertragungsnetze und genregulatorische Netze unterscheiden sich von biochemischen Reaktionsnetzen, so dass das hier entwickelte Visualisierungsverfahren nicht direkt angewandt werden kann. Eine Herausforderung liegt in der Untersuchung der Darstellungsanforderungen für diese Netze, um in einem zweiten Schritt das Zeichenverfahren entsprechend anpassen zu können. Da diese Netze ebenfalls gerichtete Abhängigkeiten enthalten und da Bereiche von Signalübertragungsnetzen biochemischen Reaktionen entsprechen, besteht begründete Hoffnung, dass sich das Zeichenverfahren zur Darstellung dieser Netze verwenden lässt.

Mit den Prozessen in Zellen sind weitere Informationen verknüpft. Ein Beispiel ist die Lokalisation von Reaktionen in speziellen Zellen oder Zell-Kompartimenten. Hier bleibt zu untersuchen, wie sich diese Informationen in Zeichnungen darstellen lassen, beispielsweise die Lokalisation durch entsprechende farbliche Kodierung oder durch spezielle Platzierung der Komponenten.

In der Arbeit wurde ein universelles ebenenweises Zeichenverfahren für Graphen entwickelt. Es bleibt zu untersuchen, wie sich das Zeichenverfahren für andere Anwendungen außerhalb der Biologie verwenden lässt. Ein Beispiel sind UML-Diagramme [BRJ98, FS98], die in der Softwareentwicklung eine große Rolle spielen. Die bisher vorgeschlagenen Verfahren zur Visualisierung [Eic00, GRR99, Mar98, See97] haben sich nicht durchgesetzt. Dies liegt unter anderem daran, dass es bisher keine umfassende Untersuchung der Darstellungsanforderungen gibt. Auch hier ist die Herausforderung, die Darstellungsanforderungen zu analysieren, um darauf aufbauend das Zeichenverfahren zielgerichtet anwenden zu können.

¹Signalübertragung ist ein Vorgang, durch den eine Zelle ein extrazelluläres Signal in eine Reaktion (im Sinne einer Antwort auf das Signal) umwandelt. Wird beispielsweise eine Leberzelle einem Glucocorticoid-Hormon ausgesetzt, so wird als Folge die Produktion mehrerer spezifischer Proteine drastisch erhöht. Ausführliche Informationen zur Signalübertragung finden sich in [Kra97].

²Genregulatorische Netze beschreiben Wechselwirkungen bei der Transkription von Genen.

Literaturverzeichnis

- [ABH94] APPEL, R. D., A. BAIROCH und D. F. HOCHSTRASSER: *A new generation of information retrieval tools for biologists: the example of the ExPASy WWW server*. Trends in Biochemical Sciences, 19:258–260, 1994.
- [ABL⁺97] ALBERTS, B., D. BRAY, J. LEWIS, M. RAFF, K. ROBERTS und J. D. WATSON: *Molekularbiologie der Zelle*. VCH Verlagsgesellschaft, Weinheim, 1997.
- [Adl94] ADLEMAN, L. M.: *Molecular Computation of Solutions to Combinatorial Problems*. Science, 266:1021–1024, 1994.
- [Adl96] ADLEMAN, L. M.: *On Constructing a Molecular Computer*. In: BAUM, E. B. und R. J. LIPTON [BL96], Seiten 1–22.
- [AHU74] AHO, A. V., J. E. HOPCROFT und J. D. ULLMAN: *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, 1974.
- [AHU83] AHO, A. V., J. E. HOPCROFT und J. D. ULLMAN: *Data Structures and Algorithms*. Addison-Wesley, Reading, 1983.
- [ARWS94] ARNDT, S., A. RIECK, A. WILL und H. SCHUMANN: *Visualisierung multivarianter Daten aus der Biologie*. In: HAUBENSACK, F. und J. SÜHNEL [HS94], Seiten 127–130.
- [Ata99] ATALLAH, M. J. (Herausgeber): *Algorithms and Theory of Computation Handbook*. CRC Press, 1999.
- [Bac97] BACHL, S.: *Graphlet's Tree Drawing*. Manuskript, Universität Passau, 1997.
- [Bac01] BACHL, W.: *Interaktives Orthogonales Zeichnen von Graphen*. Dissertation, Universität Passau, 2001. in Vorbereitung.
- [Bah00] BAHNSEN, U.: *Im Dickicht der Proteine*. Die Zeit Nr. 29, 14. Juli, Seiten 33–34, 2000.
- [Bau90] BAUMGARTEN, B.: *Petri-Netze*. BI Verlag, Mannheim, 1990.
- [BCE⁺92] BRODLIE, K. W., L. A. CARPENTER, R. A. EARNSHAW, J. R. GALLOP, R. J. HUBBOLD, A. M. MUMFORD, C. D. OSLAND und P. QUARENDON (Herausgeber): *Scientific Visualisation*. Springer Verlag, Heidelberg, 1992.
- [BE99] BASALAJ, W. und K. EILBECK: *Straight-line drawings of protein interactions*. In: KRATOCHVIL, J. [Kra99], Seiten 259–266.

- [BEMW00] BUSATTO, G., G. ENGELS, K. MEHNER und A. WAGNER: *A Framework for Adding Packages to Graph Transformation Approaches*. In: EHRIG, H., G. ENGELS, H. J. KREOWSKI und G. ROZENBERG (Herausgeber): *Selected Papers of the 8th International Workshop on Theory and Application of Graph Transformation (TAGT'00)*, Band 1764 der Reihe *Lecture Notes in Computer Science*, Seiten 352–367. Springer Verlag, 2000.
- [BFN85] BATINI, C., L. FURLANI und E. NARDELLI: *What is a Good Diagram? A Pragmatic Approach*. In: *Proceedings of the 4th International Conference on Entity-Relationship Approach*, Seiten 312–319, 1985.
- [BGHS98] BRANDENBURG, F. J., B. GRUBER, M. HIMSOLT und F. SCHREIBER: *Automatische Visualisierung biochemischer Information*. In: HOFESTÄDT, R. [Hof98], Seiten 24–38.
- [Bio00] *BioJAKE*. <http://sandstorm.bic.nus.edu.sg/BioJAKE/>, 2000.
- [Bio01] *BioPath: Biochemical Pathways*. <http://biopath.fmi.uni-passau.de>, 2001.
- [BJG01] BANG-JENSEN, J. und G. GUTIN: *Digraphs: Theory, Algorithms and Applications*. Springer Verlag, London, 2001.
- [BJL01] BUCHHEIM, C., M. JÜNGER und S. LEIPERT: *A Fast Layout Algorithm for k-Level Graphs*. In: MARKS, J. [Mar01], Seiten 229–240.
- [BJM97] BRANDENBURG, F. J., M. JÜNGER und P. MUTZEL: *Algorithmen zum automatischen Zeichnen von Graphen*. *Informatik Spektrum*, 27(4):199–207, 1997.
- [BJM⁺99] BRANDENBURG, F. J., M. JÜNGER, J. MARKS, P. MUTZEL und F. SCHREIBER: *Graph-Drawing Contest Report*. In: KRATOCHVIL, J. [Kra99], Seiten 400–409.
- [BJT92] BANG-JENSEN, J. und C. THOMASSEN: *A Polynomial Algorithm for the 2-Path Problem for Semicomplete Digraphs*. *SIAM Journal on Discrete Mathematics*, 5(3):366–376, 1992.
- [BK94] BIEDL, T. und G. KANT: *A Better Heuristic for Orthogonal Graph Drawings*. In: LEEUWEN, J. VAN (Herausgeber): *Proceedings of the 2nd European Symposium on Algorithms (ESA'94)*, Band 855 der Reihe *Lecture Notes in Computer Science*, Seiten 124–135. Springer Verlag, 1994.
- [BK97] BIEDL, T. C. und M. KAUFMANN: *Area-efficient Static and Incremental Graph Drawings*. In: *Proceedings of the 5th European Symposium on Algorithms (ESA'97)*, Band 1284 der Reihe *Lecture Notes in Computer Science*, Seiten 37–52. Springer Verlag, 1997.
- [BKW99] BRANDES, U., P. KENIS und D. WAGNER: *Centrality in Policy Network Drawings*. In: KRATOCHVIL, J. [Kra99], Seiten 250–258.
- [BL95] BODENDIEK, R. und R. LANG: *Lehrbuch der Graphentheorie*. Spektrum Akademischer Verlag, Heidelberg, 1995. Bände 1-2.
- [BL96] BAUM, E. B. und R. J. LIPTON (Herausgeber): *DNA Based Computers*, Band 27 der Reihe *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

-
- [Blo93] BLOESCH, A.: *Aesthetic Layout of Generalized Trees*. Software - Practice and Experience, 23(8):817–827, 1993.
- [BM99] BERTAULT, F. und M. MILLER: *An Algorithm for Drawing Compound Graphs*. In: KRATOCHVIL, J. [Kra99], Seiten 197–204.
- [BMT97] BIEDL, T. C., B. P. MADDEN und I. G. TOLLIS: *The Three-Phase Method: A Unified Approach to Orthogonal Graph Drawing*. In: DI BATTISTA, G. [Di 97], Seiten 391–402.
- [BNT86] BATINI, C., E. NARDELLI und R. TAMASSIA: *A Layout Algorithm for Data Flow Diagrams*. IEEE Transactions on Software Engineering, SE-12(4):538–546, 1986.
- [BP90] BÖHRINGER, K.-F. und F. N. PAULISCH: *Using Constraints to Achieve Stability in Automatic Graph Layout Algorithms*. In: *Proceedings of the ACM SIGCHI'90 Conference on Human Factors in Computing Systems, Constraint Based UI Tools*, Seiten 43–51, 1990.
- [BPC01] *BPC: The BioPathways Consortium*. <http://www.biopathways.org/>, 2001.
- [Bra90] BRANDENBURG, F. J.: *Layout Graph Grammars: the Placement Approach*. In: EHRIG, H., H.-J. KREOWSKI und G. ROZENBERG (Herausgeber): *Proceedings of the 4th International Workshop on Graph Grammars and their Application to Computer Science*, Band 532 der Reihe *Lecture Notes in Computer Science*, Seiten 144–156. Springer Verlag, 1990.
- [Bra95] BRANDENBURG, F. J.: *Designing Graph Drawings by Layout Graph Grammars*. In: TAMASSIA, R. und I. G. TOLLIS [TT95], Seiten 416–427.
- [Bra96] BRANDENBURG, F. J. (Herausgeber): *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, Band 1027 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1996.
- [Bra97a] BRANDENBURG, F. J.: *Graph Clustering I: Cycles of Cliques*. In: DI BATTISTA, G. [Di 97], Seiten 158–168.
- [Bra97b] BRAUN, H.: *Neuronale Netze*. Springer Verlag, Berlin, 1997.
- [Bra99] BRANDENBURG, F. J.: *Drawing Decorated Graphs*. Manuskript, Universität Passau, 1999.
- [BRJ98] BOOCH, G., J. RUMBAUGH und I. JACOBSON: *UML: The Unified Modeling Language User Guide*. Addison-Wesley, Reading, 1998.
- [BS89] BRONSTEIN, I. N. und K. A. SEMENDJAJEW: *Taschenbuch der Mathematik*. Gemeinschaftsausgabe Verlag Nauka und BSB B.G. Teubner Verlagsgesellschaft, Leipzig, 1989.
- [BS90] BERGER, B. und P. SHOR: *Approximation Algorithms for the Maximum Acyclic Subgraph Problem*. In: *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'90)*, Seiten 236–243, 1990.
- [BSR92] BOTAFOGO, R., B. SHNEIDERMAN und E. RIVLIN: *Structural Analysis of Hypertext: Identifying Hierarchies and Useful Metrics*. ACM Transactions on Information Systems, 10(2):142–180, 1992.

- [BT99] BRANDEN, C. und J. TOOZE: *Introduction to Protein Structures*. Garland Science Publishing, London, 1999.
- [BTT83] BATINI, C., M. TALAMO und R. TAMASSIA: *Computer Aided Layout of Entity Relationship Diagrams*. The Journal of Systems and Software, 4(2-3):163-173, 1983.
- [BW98] BRANDES, U. und D. WAGNER: *Using Graph Layout to Visualize Train Interconnection Data*. In: WHITESIDES, S. H. [Whi98], Seiten 44-56.
- [BW00] BUCHSBAUM, A. L. und J. R. WESTBROOK: *Maintaining Hierarchical Graph Views*. In: *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA'00)*, Seiten 566-575, 2000.
- [Cam98] CAMPBELL, N. A.: *Biologie*. Spektrum Akademischer Verlag, Heidelberg, 1998.
- [Car80] CARPANO, M.-J.: *Automatic Display for Hierarchized Graphs for Computer-Aided Decision Analysis*. IEEE Transactions on Systems, Man and Cybernetics, 10(11):705-715, 1980.
- [Cat95] CATARCI, T.: *The Assignment Heuristic for Crossing Reduction*. IEEE Transactions on Systems, Man, and Cybernetics, 25(3):515-521, 1995.
- [CDTT95] COHEN, R. F., G. DI BATTISTA, R. TAMASSIA und I. G. TOLLIS: *Dynamic Graph Drawings: Trees, Series-Parallel Digraphs, and Planar ST-Digraphs*. SIAM Journal on Computing, 24(5):970-1001, 1995.
- [CFMS97] CHRISTENSEN, J., S. FRIEDMAN, J. MARKS und S. SHIEBER: *Empirical Testing of Algorithms for Variable-sized Label Placement*. In: *Proceedings of the 13th ACM Symposium on Computational Geometry (CG'97)*, Seiten 415-417, 1997.
- [CG72] COFFMAN, E. G. und R. L. GRAHAM: *Optimal Scheduling for Two-Processor Systems*. Acta Informatica, 1(3):200-213, 1972.
- [CH94] CLARK, J. und D. A. HOLTON: *Graphentheorie: Grundlagen und Anwendungen*. Spektrum Akademischer Verlag, Heidelberg, 1994.
- [Che76] CHEN, P. P.-S.: *The Entity-Relationship Model - Towards a Unified View of Data*. ACM Transactions on Database Systems, 1(1):9-36, 1976.
- [Che99] CHEN, C.: *Information Visualisation and Virtual Environments*. Springer Verlag, London, 1999.
- [CHMM98] COLLADO-VIDES, J., R. HOFESTÄDT, M. MAVROVOUNIOTIS und G. MICHAL (Herausgeber): *Workshop Modelling and Simulation of Gene and Cell Regulation and Metabolic Pathways (Schloss Dagstuhl, Seminar No. 98251)*, 1998.
- [CLR90] CORMEN, T. H., C. E. LEISERSON und R. L. RIVEST: *Introduction to Algorithms*. McGraw-Hill, New York, 1990.
- [CM93] CONSENS, M. P. und A. O. MENDELZON: *Hy⁺: a Hygraph-based Query and Visualization System*. SIGMOD Record (ACM Special Interest Group on Management of Data), 22(2):511-516, 1993.

-
- [CMS94] CHRISTENSEN, J., J. MARKS und S. SHIEBER: *Placing Text Labels on Maps and Diagrams*. In: *Graphics Gems IV*, Seiten 497–504. Academic Press, 1994.
- [CMS95] CHRISTENSEN, J., J. MARKS und S. SHIEBER: *An Empirical Study of Algorithms for Point-Feature Label Placement*. *ACM Transactions on Graphics*, 14(3):203–232, 1995.
- [Cod68] CODD, E. F.: *Cellular Automaton*. Academic Press, New York, 1968.
- [CT99] CRUZ, I. F. und R. TAMASSIA: *Graph Drawing Tutorial*. <http://www.cs.brown.edu/people/rt/papers/gd-tutorial/>, 1999.
- [Dat00] *Datenmodell BioPath*. <http://pi3.informatik.uni-mannheim.de/biopath/datenmodell.html>, 2000.
- [DDPP99] DI BATTISTA, G., W. DIDIMO, M. PATRIGNANI und M. PIZZONIA: *Orthogonal and Quasi-Upward Drawings with Vertices of Arbitrary Size*. In: KRATOCHVIL, J. [Kra99].
- [DETT94] DI BATTISTA, G., P. EADES, R. TAMASSIA und I. G. TOLLIS: *Algorithms for Drawing Graphs: An Annotated Bibliography*. *Computational Geometry*, 4(5):235–282, 1994.
- [DETT99] DI BATTISTA, G., P. EADES, R. TAMASSIA und I. G. TOLLIS: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, New Jersey, 1999.
- [DFM93] DENGLER, E., M. FRIEDEL und J. MARKS: *Constraint-driven Diagram Layout*. In: *Proceedings of the IEEE Symposium on Visual Languages (VL'93)*, Seiten 330–335, 1993.
- [DGKN97] DOBKIN, D. P., E. R. GANSNER, E. KOUTSOFIOS und S. C. NORTH: *Implementing a General-Purpose Edge Router*. In: DI BATTISTA, G. [Di 97], Seiten 262–271.
- [DHS00] DAYTON, L., S. HUNT und S. STEUER: *CoreDRAW Wow!* Addison-Wesley, München, 2000.
- [Di 97] DI BATTISTA, G. (Herausgeber): *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, Band 1353 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1997.
- [Die96] DIESTEL, R.: *Graphentheorie*. Springer Verlag, Berlin, 1996.
- [DM90] DING, C. und P. MATETI: *Framework for the Automated Drawing of Data Structure Diagrams*. *IEEE Transactions on Software Engineering*, 16(5):543–557, 1990.
- [DPTT89] DI BATTISTA, G., E. PIETROSANTI, R. TAMASSIA und I. G. TOLLIS: *Automatic Layout of PERT Diagrams with X-PERT*. In: *Proceedings of the IEEE Workshop on Visual Languages (VL'89)*, Seiten 171–176, 1989.
- [Dre95] DRESBACH, S.: *A New Heuristic Layout Algorithm for Directed Acyclic Graphs*. In: DERIGS, U., A. BACHEM und A. DREXL (Herausgeber): *Proceedings of the International Conference on Operations Research (OR'94)*, Seiten 121–126. Springer Verlag, 1995.
- [Dro96] DROSDOWSKI, G. (Herausgeber): *Duden - Deutsches Universalwörterbuch*. Duden Verlag, Mannheim, 1996.

- [DT88] DI BATTISTA, G. und R. TAMASSIA: *Algorithms for Plane Representations of Acyclic Digraphs*. Theoretical Computer Science, 61(2-3):175–198, 1988.
- [Ead84] EADES, P.: *A Heuristic for Graph Drawing*. Congressus Numerantium, 42:149–160, 1984.
- [ECMS97] EDMONDSON, S., J. CHRISTENSEN, J. MARKS und S. SHIEBER: *A General Cartographic Labeling Algorithm*. Cartographica, 33(4):13–23, 1997.
- [Eco00] *EcoCyc and MetaCyc WWW Server*. <http://ecocyc.panbio.com/ecocyc/>, 2000.
- [EF96] EADES, P. und Q.-W. FENG: *Orthogonal Grid Drawing for Clustered Graphs*. Technischer Bericht, Department of Computer Science, The University of Newcastle, 1996.
- [EFL97] EADES, P., Q.-W. FENG und X. LIN: *Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs*. In: NORTH, S. C. [Nor97], Seiten 113–128.
- [EHW99] ELLIS, L. B., C. D. HERSHBERGER und L. P. WACKETT: *The University of Minnesota Biocatalysis/Biodegradation Database: Specialized Metabolism for Functional Genomics*. Nucleic Acid Research, 27(1):373–376, 1999.
- [EHW00] ELLIS, L. B., C. D. HERSHBERGER und L. P. WACKETT: *The University of Minnesota Biocatalysis/Biodegradation Database: Microorganisms, Genomics and Prediction*. Nucleic Acids Research, 28(1):377–379, 2000.
- [Eic00] EICHELBERGER, H.: *Automatisches Zeichnen von UML Klassendiagrammen mit dem Sugiyama-Algorithmus*. In: *Tagungsband des GI-Workshop Softwarevisualisierung 2000, Technischer Bericht Universität des Saarlandes A/01/2000*, 2000.
- [EK86] EADES, P. und D. KELLY: *Heuristics for Reducing Crossings in 2-Layered Networks*. Ars Combinatoria, 21:89–98, 1986.
- [EL00] EADES, P. und X. LIN: *Spring Algorithms and Symmetry*. Theoretical Computer Science, 240(2):379–405, 2000.
- [ELMS91] EADES, P., W. LAI, K. MISUE und K. SUGIYAMA: *Preserving the Mental Map of a Diagram*. In: *Proceedings of COMPUGRAPHICS '91*, Band I, Seiten 34–43, 1991.
- [ELS93] EADES, P., X. LIN und W. F. SMYTH: *A Fast and Effective Heuristic for the Feedback Arc Set Problem*. Information Processing Letters, 47(6):319–323, 1993.
- [ELT96] EADES, P., X. LIN und R. TAMASSIA: *An Algorithm for Drawing a Hierarchical Graph*. International Journal of Computational Geometry and Applications, 6(2):145–155, 1996.
- [EM96] EADES, P. und J. MARKS: *Graph-Drawing Contest Report*. In: BRANDENBURG, F. J. [Bra96], Seiten 224–233.
- [EM99] EADES, P. und P. MUTZEL: *Graph Drawing Algorithms*. In: ATALLAH, M. J. [Ata99]. Kapitel 9.

-
- [EMP00] *EMP - Enzymes and Metabolic Pathways database.* <http://www.empproject.com/about/>, 2000.
- [ENSS95] EVEN, G., J. NAOR, B. SCHIEBER und M. SUDAN: *Approximating Minimum Feedback Sets and Multi-Cuts in Directed Graphs.* In: *Proceedings of the 4th International Conference on Integer Programming and Combinatorial Optimization*, Band 920 der Reihe *Lecture Notes in Computer Science*, Seiten 14–28. Springer Verlag, 1995.
- [ENZ00] *ENZYME - Enzyme nomenclature database.* <http://www.expasy.ch/enzyme/>, 2000.
- [ES90] EADES, P. und K. SUGIYAMA: *How to Draw a Directed Graph.* *Journal of Information Processing*, 13(4):424–437, 1990.
- [Eul36] EULER, L.: *Solutio problematis ad geometriam situs pertinentis.* *Commentarii Academiae Scientiarum Petropolitanae*, 8:128–140, 1736. (*Opera Omnia*, 1, vol. 7 (1911–1956) 1–10).
- [EW86] EADES, P. und N. C. WORMALD: *The Median Heuristic for Drawing 2-Layers Networks.* *Technischer Bericht 69*, Department of Computer Science, University of Queensland, 1986.
- [EW92] EARNSHAW, R. A. und N. WISEMAN: *An Introductory Guide to Scientific Visualization.* Springer Verlag, Heidelberg, 1992.
- [EW94] EADES, P. und N. C. WORMALD: *Edge Crossings in Drawings of Bipartite Graphs.* *Algorithmica*, 11(4):379–403, 1994.
- [Exp00] *ExpASY Molecular Biology Server.* <http://www.expasy.ch/>, 2000.
- [FC84] FRANCO, R. und E. I. CANELA: *Computer Simulation of Purine Metabolism.* *European Journal of Biochemistry*, 144:305–315, 1984.
- [FK97] FÖSSMEIER, U. und M. KAUFMANN: *Algorithms and Area Bounds for Nonplanar Orthogonal Drawings.* In: DI BATTISTA, G. [Di 97], Seiten 134–145.
- [Flo90] FLOOD, M. M.: *Exact and Heuristic Algorithms for the Weighted Feedback Arc Set Problem.* *Networks*, 20:1–23, 1990.
- [FM99] FALLERT-MÜLLER, A. (Herausgeber): *Lexikon der Biochemie.* Spektrum Akademischer Verlag, Heidelberg, 1999. Bände 1-2.
- [For99] FORSTER, M.: *Zeichnen ungerichteter Graphen mit gegebenen Knotengrößen durch ein Springembedder Verfahren.* Diplomarbeit, Universität Passau, 1999.
- [Föß97] FÖSSMEIER, U.: *Interactive Orthogonal Graph Drawing: Algorithms and Bounds.* In: DI BATTISTA, G. [Di 97], Seiten 111–123.
- [FR91] FRUCHTERMAN, T. und E. REINGOLD: *Graph Drawing by Force-directed Placement.* *Software - Practice and Experience*, 21(11):1129–1164, 1991.

- [Fri95] FRIEDRICH, C.: *The ffggraph Library*. Technischer Bericht MIP-9520, Universität Passau, 1995.
- [FS98] FOWLER, M. und K. SCOTT: *UML Distilled*. Addison-Wesley, Reading, 1998.
- [Fuk99] FUKUYAMA, F.: *Der programmierte Unmensch*. Serie: Die Gegenwart der Zukunft, Feuilleton-Beilage der Süddeutschen Zeitung, 7./8. August, 1999.
- [FvFH90] FOLEY, J. D., A. VAN DAM, S. K. FEINER und J. F. HUGHES: *Computer Graphics, Principles and Practice*. Addison-Wesley, Reading, 1990.
- [FW91] FORMANN, M. und F. WAGNER: *A Packing Problem with Applications in Lettering of Maps*. In: *Proceedings of the 7th ACM Symposium on Computational Geometry (CG'91)*, Seiten 128–144, 1991.
- [Gar94] GARLAND, K.: *Mr Beck's Underground Map*. Capitel Transport Publishing, Harrow Weald, 1994.
- [GBO⁺97] GOTO, S., H. BONO, H. OGATA, W. FUJIBUCHI, T. NISHIOKA und M. KANEHISA: *Organizing and Computing Metabolic Pathway Data in Terms of Binary Relations*. In: *Proceedings of the 2nd Pacific Symposium on Biocomputing*, Seiten 175–186, 1997.
- [GBS98] GHOSH, D., F. BRGLEZ und M. STALLMANN: *First Steps Towards Experimental Design in Evaluating Layout Algorithms: Wire Length versus Wire Crossing in Linear Placement Optimization*. Technischer Bericht TR-CBL-11-Ghosh, North Carolina State University, 1998.
- [GJ79] GAREY, M. R. und D. S. JOHNSON: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.
- [GJ83] GAREY, M. R. und D. S. JOHNSON: *Crossing number is NP-complete*. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.
- [GKHK86] GELLERT, W., H. KÜSTNER, M. HELLWICH und H. KÄSTNER (Herausgeber): *Kleine Enzyklopädie Mathematik*. VEB Bibliographisches Institut, Leipzig, 1986.
- [GKNV93] GANSNER, E. R., E. KOUTSOFIOS, S. C. NORTH und K. P. VO: *A Technique for Drawing Directed Graphs*. *IEEE Transactions on Software Engineering*, 19(3):214–230, 1993.
- [Glo89] GLOVER, F.: *Tabu Search - Part 1*. *ORSA Journal on Computing*, 1(3):190–206, 1989.
- [GMS94] GOOSSENS, M., F. MITTELBACH und A. SAMARIN: *Der \LaTeX -Begleiter*. Addison-Wesley, Bonn, 1994.
- [GN98] GANSNER, E. R. und S. C. NORTH: *Improved Force-Directed Layouts*. In: WHITESIDES, S. H. [Whi98], Seiten 364–373.
- [GN00] GANSNER, E. R. und S. C. NORTH: *An Open Graph Visualization System and its Applications to Software Engineering*. *Software - Practice and Experience*, 30(11):1203–1233, 2000.

-
- [GNV88] GANSNER, E. R., S. C. NORTH und K. P. VO: *DAG - A Program that Draws Directed Graphs*. *Software - Practice and Experience*, 17(1):1047–1062, 1988.
- [Gra01a] *Graphlet: A Toolkit for Implementing Graph Editors and Graph Drawing Algorithms*. <http://www.infosun.fmi.uni-passau.de/Graphlet/>, 2001.
- [Gra01b] *Graphviz*. <http://www.research.att.com/sw/tools/graphviz/>, 2001.
- [Gre98] GREULICH, W. (Herausgeber): *Lexikon der Physik*. Spektrum Akademischer Verlag, Heidelberg, 1998. Bände 1-6.
- [Gro96] GROPP, H.: *The Drawing of Configurations*. In: BRANDENBURG, F. J. [Bra96], Seiten 267–276.
- [GRR99] GOGOLLA, M., O. RADFELDER und M. RICHTERS: *Towards Three-Dimensional Representation and Animation of UML Diagrams*. In: FRANCE, R. und B. RUMPE (Herausgeber): *Proceedings of the 2nd International Conference of the Unified Modeling Language (UML99)*, Band 1723 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1999.
- [GSBM01] GÜNTHER, W., R. SCHÖNFELD, B. BECKER und P. MOLITOR: *k-Layer Straightline Crossing Minimization by Speding up Sifting*. In: MARKS, J. [Mar01], Seiten 253–258.
- [GTL01] *GTL: Graph Template Library*. <http://www.infosun.fmi.uni-passau.de/GTL/>, 2001.
- [Har95] HAREL, D.: *On Visual Formalisms*. In: CHANDRASEKARAN, B., J. GLASGOW und H. NARAYANAN (Herausgeber): *Diagrammatic Reasoning*, Seiten 235–271. AAAI Press, 1995.
- [Har96] HAREL, D.: *Statecharts: Past, Present and Future*. In: *Proceedings of the 23rd Seminar on Current Trends in Theory and Practice of Informatics*, Band 1175 der Reihe *Lecture Notes in Computer Science*, Seite 285, 1996.
- [Hei94] HEISTERMANN, J.: *Genetische Algorithmen*. Teubner Verlag, Stuttgart, 1994.
- [HG95] HOWER, W. und W. H. GRAF: *Research in Constraint-Based Layout, Visualization, CAD, and Related Topics: A Bibliographical Survey*. Technischer Bericht RR-95-12, DFKI, 1995.
- [Him93] HIMSOLT, M.: *Konzeption und Implementierung von Grapheditoren*. Dissertation, Universität Passau, 1993.
- [Him97] HIMSOLT, M.: *The Graphlet System*. In: NORTH, S. C. [Nor97], Seiten 233–240.
- [Him00] HIMSOLT, M.: *Graphlet: Design and Implementation of a Graph Editor*. *Software - Practice and Experience*, 30(11):1303–1324, 2000.
- [HIMF98] HAYASHI, K., M. INOUE, T. MASUZAWA und H. FUJIWARA: *A Layout Adjustment Problem for Disjoint Rectangles Preserving Orthogonal Order*. In: WHITESIDES, S. H. [Whi98], Seiten 183–197.

- [HM95] HOFESTÄDT, R. und F. MEINEKE: *Interactive Modelling and Simulation of Biochemical Networks*. Computers in Biology and Medicine, 25(3):312–334, 1995.
- [HM97] HE, W. und K. MARRIOTT: *Constrained Graph Layout*. In: NORTH, S. C. [Nor97], Seiten 217–232.
- [Hof98] HOFESTÄDT, R. (Herausgeber): *Proceedings of the Workshop Molekulare Bioinformatik, GI Jahrestagung*. Shaker Verlag, 1998.
- [Hof99] HOFESTÄDT, R.: *Bioinformatik 2000 - Von der Molekularen Biologie zum Metabolic Engineering*. Informatik Spektrum, 22(5):385–390, 1999.
- [HR94] HASSIN, R. und S. RUBINSTEIN: *Approximations for the Maximum Acyclic Subgraph Problem*. Information Processing Letters, 51:133–140, 1994.
- [HS94] HAUBENSACK, F. und J. SÜHNEL (Herausgeber): *Proceedings of the Workshop Bioinformatik '94*, 1994.
- [HT98] HOFESTÄDT, R. und S. THELEN: *Qualitative Modeling of Biochemical Networks*. Silico Biology, 1:39–53, 1998.
- [HU90] HOPCROFT, J. E. und J. D. ULLMAN: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Addison-Wesley, Bonn, 1990.
- [IL97] ITURRIAGA, C. und A. LUBIW: *Elastic Labels: The Two-Axis Case*. In: DI BATTISTA, G. [Di 97], Seiten 181–192.
- [Imh62] IMHOF, E.: *Die Anordnung der Namen in der Karte*. Internationales Jahrbuch für Kartographie, 2:93–129, 1962.
- [INK95] IGARASHI, T., Y. NADAOKA und T. KAMINUMA: *CSNDB - A Data and Knowledge Base for Cell Signaling Networks*. In: *Proceedings of the Genome Informatics Workshop*, Seiten 49–56, 1995.
- [Int92] INTERNATIONAL UNION OF BIOCHEMISTRY AND MOLECULAR BIOLOGY, NOMENCLATURE COMMITTEE: *Enzyme Nomenclature*. Academic Press, 1992.
- [IR99] ITALIANO, G. F. und R. RAMAN: *Topics in Data Structures*. In: ATALLAH, M. J. [Ata99]. Kapitel 5.
- [JLMO97] JÜNGER, M., E. K. LEE, P. MUTZEL und T. ODENTHAL: *A Polyhedral Approach to the Multi-Layer Crossing Number Problem*. In: DI BATTISTA, G. [Di 97], Seiten 13–24.
- [JM96a] JÜNGER, M. und P. MUTZEL: *Exact and Heuristic Algorithms for 2-Layer Straightline Crossing Minimization*. In: BRANDENBURG, F. J. [Bra96], Seiten 337–348.
- [JM96b] JÜNGER, M. und P. MUTZEL: *Maximum Planar Subgraphs and Nice Embeddings: Practical Layout Tools*. Algorithmica, 16(1):33–59, 1996.

-
- [JM97] JÜNGER, M. und P. MUTZEL: *2-Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms*. *Journal of Graph Algorithms and Applications*, 1(1):1–25, 1997.
- [Kaa81] KAAS, R.: *A Branch and Bound Algorithm for the Acyclic Subgraph Problem*. *European Journal of Operations Research*, 8:335–362, 1981.
- [Kah62] KAHN, A. B.: *Topological Sorting of Large Networks*. *Communications of the ACM*, 5:558–562, 1962.
- [Kan96] KANEHISA, M.: *Toward Pathway Engineering: a New Database of Genetic and Molecular Pathways*. *Science & Technology Japan*, 59:34–38, 1996.
- [Kan99] KANEHISA, M.: *KEGG: From genes to biochemical pathways*. In: LETOVSKY, S. (Herausgeber): *Bioinformatics: Databases and Systems*, Seiten 63–76. Kluwer Academic Publishers, 1999.
- [Kan01] KANNE, C.-C.: *Persönliche Kommunikation*, 2001.
- [Kar72] KARP, R. M.: *Reducibility Among Combinatorial Problems*. In: MILLER, R. E. und J. W. THATCHER (Herausgeber): *Complexity of Computer Computations*, Seiten 85–103. Plenum Press, 1972.
- [KEG00] *KEGG: Kyoto Encyclopedia of Genes and Genomes*. <http://www.genome.ad.jp/kegg/>, 2000.
- [KG00] KANEHISA, M. und S. GOTO: *KEGG: Kyoto Encyclopedia of Genes and Genomes*. *Nucleic Acid Research*, 28(1):27–30, 2000.
- [Kin94] KINNEBROCK, W.: *Neuronale Netze*. Oldenbourg Verlag, München, 1994.
- [KJ37] KREBS, H. A. und W. A. JOHNSON: *The Role of Citric Acid in Intermediate Metabolism in Animal Tissues*. *Enzymologia*, 4:148–156, 1937.
- [KK89] KAMADA, T. und S. KAWAI: *An Algorithm for Drawing General Undirected Graphs*. *Information Processing Letters*, 31(1):7–15, 1989.
- [KLMS95] KLEMETTI, H., I. LAPINLEIMU, E. MÄKINEN und M. SIERANTA: *Trimming the Spring Algorithm for Drawing Hypergraphs*. *SIGCSE Bulletin*, 27(3):34–38, 1995.
- [KLSW94] KARP, P. D., J. LOWRANCE, T. STRAT und D. WILKINS: *The Grasper-CL Graph Management System*. *LISP and Symbolic Computation*, 7:245–282, 1994.
- [KM94] KARP, P. D. und M. L. MAVROVOUNIOTIS: *Representing, Analyzing, and Synthesizing Biochemical Pathways*. *IEEE Expert*, 9(2):11–21, 1994.
- [KM99] KLAU, G. und P. MUTZEL: *Combining Graph Labeling and Compaction*. In: KRATOCHVIL, J. [Kra99], Seiten 27–37.
- [KN93] KOUTSOFIOS, E. und S. C. NORTH: *Drawing Graphs with Dot*. Technischer Bericht, AT&T Bell Laboratories, Murray Hill NJ, 1993.

- [Koh92] KOHN, M. C.: *Propagation of Information in MetaNet Graph Models*. Journal Theoretical Biology, 154:505–317, 1992.
- [KOP96] KARP, P. D., C. OUZOUNIS und S. M. PALEY: *HinCyc: A Knowledge Base of the Complete Genome and Metabolic Pathways of H. influenzae*. In: STATES, D. J., P. AGARWAL, T. GAASTERLAND, L. HUNTER und R. F. SMITH (Herausgeber): *Proceedings of the 4th International Conference on Intelligent Systems for Molecular Biology (ISMB'96)*, Seiten 116–124, 1996.
- [KP94a] KARP, P. D. und S. M. PALEY: *Automated Drawing of Metabolic Pathways*. In: LIM, H., C. CANTOR und R. BOBBINS (Herausgeber): *Proceedings of the 3rd International Conference on Bioinformatics and Genome Research*, Seiten 225–238, 1994.
- [KP94b] KARP, P. D. und S. M. PALEY: *Representations of Metabolic Knowledge: Pathways*. In: ALTMAN, R., D. BRUTLAG, P. D. KARP, R. LATHROP und D. SEARLS (Herausgeber): *Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology (ISMB'94)*, Seiten 203–211, 1994.
- [KP96] KARP, P. D. und S. M. PALEY: *Integrated Access to Metabolic and Genomic Data*. Journal of Computational Biology, 3(1):191–212, 1996.
- [KR97] KARP, P. D. und M. RILEY: *Guide to the EcoCyc Schema*. <ftp://ftp.ai.sri.com/pub/papers/karp-ecocyc-schema.ps>, 1997.
- [Kra97] KRAUSS, G.: *Biochemie der Regulation und Signaltransduktion*. Wiley-VCH, Weinheim, 1997.
- [Kra99] KRATOCHVIL, J. (Herausgeber): *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*, Band 1731 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1999.
- [KRPPT96] KARP, P. D., M. RILEY, S. M. PALEY und A. PELLEGRINI-TOOLE: *EcoCyc: Electronic Encyclopedia of E. coli Genes and Metabolism*. Nucleic Acids Research, 24(1):32–40, 1996.
- [KRS⁺00] KARP, P. D., M. RILEY, M. SAIER, I. PAULSEN, S. M. PALEY und A. PELLEGRINI-TOOLE: *The EcoCyc and MetaCyc database*. Nucleic Acid Research, 28(1):56–59, 2000.
- [KS97] KNOOP, J. und F. SCHREIBER: *Analysing and Optimizing Strongly Typed Object-Oriented Languages*. In: SMOLYANINOV, A. V. und A. S. SHESTIALTYNOV (Herausgeber): *Proceedings of the 2nd International Conference on Object-Oriented Technology (WOON'97)*, Seiten 252–266, 1997.
- [KST99a] KANNE, C.-C., F. SCHREIBER und D. TRÜMBACH: *Electronic Biochemical Pathways*. In: KRATOCHVIL, J. [Kra99], Seiten 418–419.
- [KST99b] KANNE, C.-C., F. SCHREIBER und D. TRÜMBACH: *Interactive Biochemical Pathways*. In: *Proceedings of the German Conference on Bioinformatics (GCB'99)*, Seiten 204–205, 1999.

-
- [KT97] KAKOULIS, K. G. und I. G. TOLLIS: *An Algorithm for Labeling Edges of Hierarchical Drawings*. In: DI BATTISTA, G. [Di 97], Seiten 169–180.
- [KT98] KAKOULIS, K. G. und I. G. TOLLIS: *A Unified Approach to Labeling Graphical Features*. In: *Proceedings of the 14th ACM Symposium on Computational Geometry (CG'98)*, Seiten 347–356, 1998.
- [KT01] KAKOULIS, K. G. und I. G. TOLLIS: *On the Edge Label Placement Problem*. *Computational Geometry*, 18:1–17, 2001.
- [LC79] LOWRANCE, J. D. und D. D. CORKILL: *The Design of GRASPER 1.0: A Programming Language Extension for Graph Processing*. Technischer Report 79-6, University of Massachusetts, 1979.
- [LED00] *LEDA: A Library of the Data Types and Algorithms of Combinatorial Computing*. <http://www.mpi-sb.mpg.de/LEDA/>, 2000.
- [Len96] LENGAUER, T.: *Molekulare Bioinformatik*. In: WEGENER, I. (Herausgeber): *Highlights aus der Informatik*, Seiten 83–111. Springer Verlag, 1996.
- [Len97] LENGAUER, T.: *DFG-Schwerpunkt: Informatikmethoden zur Analyse und Interpretation großer genomischer Datenmengen*. In: JARKE, M. (Herausgeber): *28. Jahrestagung der Gesellschaft für Informatik (Informatik '97)*, Seiten 76–84. Springer Verlag, 1997.
- [Lev93] LEVY, S.: *KL- Künstliches Leben aus dem Computer*. Droemer Knaur, München, 1993.
- [Lin94] LINOS, P.: *Visualizing Program Dependencies*. *Software - Practice and Experience*, 24(4):387–403, 1994.
- [LIO01] *LION Bioscience*. <http://www.lionbioscience.com>, 2001.
- [Lip95] LIPTON, R. J.: *Speeding up Computation via Molecular Biology*. Technischer Bericht, Princeton University, 1995. <ftp://ftp.cs.princeton.edu/pub/people/rjl/bio.ps>.
- [LMV97] LAGUNA, M., R. MARTI und V. VALLS: *Arc Crossing Minimization in Hierarchical Digraphs with Tabu Search*. *Computers and Operations Research*, 24(12):1175–1186, 1997.
- [LNC94] LEHNINGER, A. L., D. L. NELSON und M. M. COX: *Prinzipien der Biochemie*. Spektrum Akademischer Verlag, Heidelberg, 1994.
- [LR96] LAMPING, J. und R. RAO: *The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies*. *Journal of Visual Languages and Computing*, 7:33–55, 1996.
- [LS77] LAM, S. und R. SETHI: *Worst Case Analysis of Two Scheduling Algorithms*. *SIAM Journal on Computing*, 6(3):518–536, 1977.
- [LW00] LOCKHART, D. J. und E. A. WINZELER: *Genomics, Gene Expression and DNA Arrays*. *Nature*, 405:827–836, 2000.

- [Mäk90a] MÄKINEN, E.: *Experiments on Drawing 2-Level Hierarchical Graphs*. International Journal Computer Mathematics, 37:129–135, 1990.
- [Mäk90b] MÄKINEN, E.: *How to Draw a Hypergraph*. International Journal Computer Mathematics, 34:177–185, 1990.
- [Mar98] MARKWITZ, S.: *Layoutalgorithmen für die UML*. OBJEKTSpektrum, 6:56–61, 1998.
- [Mar01] MARKS, J. (Herausgeber): *Proceedings of the 8th International Symposium on Graph Drawing (GD'00)*, Band 1984 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 2001.
- [Meh84a] MEHLHORN, K.: *Data Structures and Algorithms: 1. Searching and Sorting*. Springer Verlag, Heidelberg, 1984.
- [Meh84b] MEHLHORN, K.: *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. Springer Verlag, Heidelberg, 1984.
- [MELS95] MISUE, K., P. EADES, W. LAI und K. SUGIYAMA: *Layout Adjustment and the Mental Map*. Journal of Visual Languages and Computing, 6:183–210, 1995.
- [Men00] MENDES, P.: *Advanced Visualization of Metabolic Pathways in PathDB*. In: *Proceedings of the 8th Conference on Plant and Animal Genome*, 2000.
- [MH99] MELANCON, G. und I. HERMAN: *DAG Drawing from an Information Visualization Perspective*. Technischer Bericht INS-R9915, Centrum voor Wiskunde en Informatica, Amsterdam, 1999.
- [Mic93] MICHAL, G.: *Biochemical Pathways (Poster)*. Boehringer Mannheim, Penzberg, 1993.
- [Mic98a] MICHAL, G.: *On Representation of Metabolic Pathways*. BioSystems, 47:1–7, 1998.
- [Mic98b] MICHAL, G.: *Stoffwechselwege in der Biologie und ihre Darstellung*. In: HOFESTÄDT, R. [Hof98], Seiten 19–23.
- [Mic99] MICHAL, G.: *Biochemical Pathways*. Spektrum Akademischer Verlag, Heidelberg, 1999.
- [Mic00] MICHAL, G.: *Persönliche Kommunikation*, 2000.
- [Mic01] MICHAL, G.: *Briefwechsel zwischen G. Michal und F. J. Brandenburg*, 2001.
- [Min98] MINAS, M.: *Hypergraph Representation of Diagrams in Diagram Editors*. In: *Proceedings of the AAAI Fall Symposium, Workshop on Formalizing Reasoning with Visual and Diagrammatic Representations*, Seiten 79–85, 1998.
- [ML97] MARTÍ, R. und M. LAGUNA: *Heuristics and Meta-Heuristics for 2-Layer Straight Line Crossing Minimization*. Technischer Bericht, Department of Statistics and Operations Research, University of Valencia, 1997.
- [Moe90] MOEN, S.: *Drawing Dynamic Trees*. IEEE Software, Seiten 21–28, 1990.

-
- [MP43] McCULLOCH, W. S. und W. PITTS: *A Logical Calculus of the Ideas Immanent in Nervous Activity*. Bulletin Mathematical Biophysics, Seiten 115–133, 1943.
- [MP70] MODER, J. J. und C. R. PHILLIPS: *Project Management with CPM and PERT*. Reinhold, New York, 1970.
- [MRH91] MESSINGER, E. B., L. A. ROWE und R. R. HENRY: *A Divide-and-Conquer Algorithm for the Automatic Layout of Large Directed Graphs*. IEEE Transactions on Systems, Man, and Cybernetics, 21(1):1–11, 1991.
- [MS91a] MARKS, J. und S. SHIEBER: *The Computational Complexity of Cartographic Label Placement*. Technischer Bericht 05-91, Harvard University, 1991.
- [MS91b] MISUE, K. und K. SUGIYAMA: *Multi-Viewpoint Perspective Display Methods: Formulation and Application to Compound Graphs*. In: *Proceedings of the 4th International Conference on Human-Computer Interaction*, Band 2 der Reihe *Congress II: Design and Implementation of Interactive Systems*, Seiten 834–838, 1991.
- [MS94] MÄKINEN, E. und M. SIERANTA: *Genetic Algorithms for Drawing Bipartite Graphs*. Technischer Bericht A-1994-1, Department of Computer Science, University of Tampere, 1994.
- [MS96] MUSSER, D. R. und A. SAINI: *STL Tutorial and Reference Guide*. Addison-Wesley, Reading, 1996.
- [MSM99] MATUSZEWSKI, C., R. SCHÖNFELD und P. MOLITOR: *Using Sifting for k-Layer Straightline Crossing Minimization*. In: KRATOCHVIL, J. [Kra99], Seiten 217–224.
- [MST01] MARRIOTT, K., P. STUCKEY und V. TAM: *Removing Node Overlapping in Graph Layout Using Constrained Optimization*. Manuskript, erscheint im Journal *Constraint*, 2001.
- [Mut97] MUTZEL, P.: *An Alternative Method to Crossing Minimization on Hierarchical Graphs*. In: NORTH, S. C. [Nor97], Seiten 318–333.
- [Näh93] NÄHER, S.: *LEDA - a Library of Efficient Data Types and Algorithms*. In: ENJALBERT, P., A. FINKEL und K. W. WAGNER (Herausgeber): *Proceedings of the 10th Symposium on the Theoretical Aspects of Computer Science (STACS'93)*, Band 665 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1993.
- [Nic97] NICHOLSON, D. E.: *Metabolic Pathways Map (Poster)*. Sigma Chemical Co., St. Louis, 1997.
- [NK94] NORTH, S. C. und E. KOUTSOFIOS: *Applications of Graph Visualization*. In: *Proceedings of Graphics Interface*, Seiten 235–245, 1994.
- [Nor96] NORTH, S. C.: *Incremental Layout in DynaDAG*. In: BRANDENBURG, F. J. [Bra96], Seiten 409–418.
- [Nor97] NORTH, S. C. (Herausgeber): *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*, Band 1190 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1997.

- [OGS⁺99] OGATA, H., S. GOTO, K. SATO, W. FUJIBUCHI, H. BONO und M. KANEHISA: *KEGG: Kyoto Encyclopedia of Genes and Genomes*. Nucleic Acid Research, 27(1):29–34, 1999.
- [PA95] PACH, J. und P. K. AGARWAL: *Combinatorial Geometry*. John Wiley & Sons, New York, 1995.
- [Par00] *Parteienfinanzierung*. http://people.freenet.de/parteienfinanzierung/pf_diag.htm, 2000.
- [Pat00] *PathDB: Metabolic Pathways Database*. <http://www.ncgr.org/software/pathdb/>, 2000.
- [PCJ96] PURCHASE, H. C., R. F. COHEN und M. JAMES: *Validating Graph Drawing Aesthetics*. In: BRANDENBURG, F. J. [Bra96], Seiten 435–446.
- [PFB01] *PFBP: Protein Function and Biochemical Pathways*. <http://www.ebi.ac.uk/research/pfmp/>, 2001.
- [Phi00] PHIMISTER, B.: *The Nature of the Number*. Nature Genetics, 25(2):127–128, 2000.
- [PRS98] PÄUN, G., G. ROZENBERG und A. SALOMAA: *DNA Computing: New Computing Paradigms*. Springer Verlag, Berlin, 1998.
- [PST97] PAPAKOSTAS, A., J. M. SIX und I. G. TOLLIS: *Experimental and Theoretical Results in Interactive Orthogonal Graph Drawings*. In: NORTH, S. C. [Nor97], Seiten 371–386.
- [PT90] PAULISH, F. N. und W. F. TICHY: *EDGE: An Extendible Graph Editor*. Software - Practice and Experience, 20(S1):63–88, 1990.
- [PT95] PAPAKOSTAS, A. und I. G. TOLLIS: *Improved Algorithms and Bounds for Orthogonal Drawings*. In: TAMASSIA, R. und I. G. TOLLIS [TT95], Seiten 40–51.
- [PT97] PAPAKOSTAS, A. und I. G. TOLLIS: *A Pairing Technique for Area-Efficient Orthogonal Drawings*. In: NORTH, S. C. [Nor97], Seiten 355–370.
- [Pur97] PURCHASE, H. C.: *Which Aesthetic Has the Greatest Effect on Human Understanding?* In: DI BATTISTA, G. [Di 97], Seiten 248–261.
- [Pur98] PURCHASE, H. C.: *Performance of Layout Algorithms: Comprehension, not Computation*. Journal of Visual Languages and Computing, 9:647–657, 1998.
- [Ram98] RAMSAY, G.: *DNA Chips – States-of-the-Art*. Nature Biotechnology, 16(1):40–44, 1998.
- [RCG97] RYSER, S., B. D. CARBONARE und E. GWINNER: *Streiflichter der Gentechnik*. La Roche, Basel, 1997.
- [RDM⁺87] ROWE, L. A., M. DAVIS, E. MESSINGER, C. MEYER, C. SPIRAKIS und A. TUAN: *A Browser for Directed Graphs*. Software - Practice and Experience, 17(1):61–76, 1987.
- [Rei86] REISIG, W.: *Petri Netze*. Springer Verlag, Heidelberg, 1986.

-
- [Rei95] REISS, S. P.: *3-D Visualization of Program Information*. In: TAMASSIA, R. und I. G. TOLLIS [TT95], Seiten 12–23.
- [REM99] REM 99. <http://www.zoologie.sbg.ac.at/rem99/home.htm>, 1999.
- [Ric91] RICHARDS, F. M.: *The Protein Folding Problem*. Scientific American, 54(1):34–41, 1991.
- [RL00] RUBEN, A. J. und L. F. LANDWEBER: *The Past, Present and Future of Molecular Computing*. Nature Reviews: Molecular Cell Biology, 1:69–72, 2000.
- [RML93] REDDY, V. N., M. L. MAVROVOUNIOTIS und M. N. LIEBMAN: *Petri Net Representations of Metabolic Pathways*. In: HUNTER, L., D. SEARLS und J. SHAVLIK (Herausgeber): *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB'93)*, Seiten 328–336, 1993.
- [RMS97] RYALL, K., J. MARKS und S. SHIEBER: *An Interactive Constraint-Based System for Drawing Graphs*. In: *Proceedings of the ACM Symposium on User Interface Software and Technology*, Seiten 97–104, 1997.
- [Rud93] RUDELL, R.: *Dynamic Variable Ordering for Ordered Binary Decision Diagrams*. In: *Proceedings of the IEEE/ACM International Conference on CAD*, Seiten 42–47, 1993.
- [San95] SANDER, G.: *Graph Layout through the VCG Tool*. Technischer Bericht Sep26-34, Technische Universität München, 1995.
- [San96a] SANDER, G.: *A Fast Heuristic for Hierarchical Manhattan Layout*. In: BRANDENBURG, F. J. [Bra96], Seiten 447–458.
- [San96b] SANDER, G.: *Visualisierungstechniken für den Compilerbau*. Dissertation, Universität Saarbrücken, 1996.
- [San99] SANDER, G.: *Graph Layout for Applications in Compiler Construction*. Theoretical Computer Science, 217(2):175–214, 1999.
- [SBG⁺96] SELKOV, E., S. BASMANOVA, T. GAASTERLAND, I. GORYANIN, Y. GRETCHKIN, N. MALTSEV, V. NENASHEV, R. OVERBEEK, E. PANYUSHKINA, L. PRONEVITCH, E. SELKOV, JR und I. YUNUS: *The Metabolic Pathway Collection from EMP: The Metabolic Pathways Database*. Nucleic Acids Research, 24(1):26–29, 1996.
- [SBG99] STALLMANN, M., F. BRGLEZ und D. GHOSH: *Heuristics and Experimental Design for Bi-graph Crossing Number Minimization*. In: GOODRICH, M. T. und C. C. MCGEOCH (Herausgeber): *Proceedings of the 1st Workshop on Algorithm Engineering and Experimentation (ALENEX'99)*, Band 1619 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1999.
- [Sch96] SCHIRMER, R.: *Eine Erweiterung des Algorithmus von Fruchterman und Reingold zum Zeichnen von Graphen um die Berücksichtigung von Constraints*. Diplomarbeit, Universität Passau, 1996.
- [Sch01] SCHACHERER, F.: *Persönliche Kommunikation*, 2001.
-

- [Sed92] SEDGEWICK, R.: *Algorithmen in C*. Addison-Wesley, Bonn, 1992.
- [See97] SEEMANN, J.: *Extending the Sugiyama Algorithm for Drawing UML Class Diagrams*. In: DI BATTISTA, G. [Di 97], Seiten 415–424.
- [SFM99] STOREY, M.-A. D., F. D. FRACCHIA und H. A. MÜLLER: *Customizing a Fisheye View Algorithm to Preserve the Mental Map*. *Journal of Visual Languages and Computing*, 10:245–267, 1999.
- [SGI⁺97] SELKOV, E., M. GALIMOVA, N. IVANOVA, I. GORYANIN, Y. GRETCHKIN, Y. KOMAROV, N. MALTSEV, V. NENASHEV, R. OVERBEEK, N. MIKHAILOVA, E. PANYUSHKINA, L. PRONEVITCH und E. SELKOV, JR: *The Metabolic Pathway Collection: an Update*. *Nucleic Acids Research*, 25(1):37–38, 1997.
- [SGMS98] SELKOV, JR, E., Y. GRECHKIN, N. MIKHAILOVA und E. SELKOV: *MPW: The Metabolic Pathways Database*. *Nucleic Acids Research*, 26(1):43–45, 1998.
- [SHF94] SCHÖNEBURG, E., F. HEINZMANN und S. FEDDERSEN: *Genetische Algorithmen und Evolutionsstrategien*. Addison-Wesley, Bonn, 1994.
- [SK96] SCHULZE-KREMER, S.: *Molecular Bioinformatics: Algorithms and Applications*. Walter de Gruyter, Berlin, 1996.
- [SM91] SUGIYAMA, K. und K. MISUE: *Visualization of Structural Information: Automatic Drawing of Compound Digraphs*. *IEEE Transactions on Systems, Man and Cybernetics*, 21(4):876–892, 1991.
- [SM95] SUGIYAMA, K. und K. MISUE: *A Simple and Unified Method for Drawing Graphs: Magnetic-Spring Algorithm*. In: TAMASSIA, R. und I. G. TOLLIS [TT95], Seiten 364–375.
- [SMKS99] SALAMONSEN, W., K. Y. MOK, P. KOLATKAR und S. SUBBIAH: *BioJAKE: A Tool for the Creation, Visualization and Manipulation of Metabolic Pathways*. In: *Proceedings of the 4th Pacific Symposium on Biocomputing*, Seiten 392–400, 1999.
- [SRS00] *SRS: Sequence Retrieval System*. <http://srs.ebi.ac.uk/>, 2000.
- [SSSV97] SHAHROKHI, F., O. SÝKORA, L. A. SZÉKELY und I. VRT’O: *On Bipartite Drawings and the Linear Arrangement Problem*. In: *Proceedings of the International Workshop in Implementation of Functional Languages (IFL97)*, Band 1467 der Reihe *Lecture Notes in Computer Science*, Seiten 55–68. Springer Verlag, 1997.
- [ST99] SIX, J. M. und I. G. TOLLIS: *A Framework for Circular Drawings of Networks*. In: KRATOCHVIL, J. [Kra99], Seiten 107–116.
- [STL00] *STL: Standard Template Library*. <http://www.sgi.com/Technology/STL/>, 2000.
- [Str95] STRYER, L.: *Biochemie*. Spektrum Akademischer Verlag, Heidelberg, 1995.

-
- [STT81] SUGIYAMA, K., S. TAGAWA und M. TODA: *Methods for Visual Understanding of Hierarchical System Structures*. IEEE Transactions on Systems, Man and Cybernetics, SMC-11(2):109–125, 1981.
- [Stü97] STÜBINGER, ANDREAS: *Eine algebraische Beschreibung planarer Graphen und ihre Anwendung auf Petri-Netze*. Dissertation, Universität Passau, 1997.
- [Tam87] TAMASSIA, R.: *On Embedding a Graph in the Grid with the Minimum Number of Bends*. SIAM Journal on Computing, 16(3):421–444, 1987.
- [Tam98] TAMASSIA, R.: *Constraints in Graph Drawing Algorithms*. Constraints, 3(1):87–120, 1998.
- [Tam99a] TAM, V.: *Removing Node and Edge Overlapping in Graph Layouts by A Modified EGENET Solver*. In: *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'99)*, Seiten 698–703, 1999.
- [Tam99b] TAMASSIA, R.: *Advances in the Theory and Practice of Graph Drawing*. Theoretical Computer Science, 217(2):235–254, 1999.
- [Tar72] TARJAN, R. E.: *Depth First Search and Linear Graph Algorithms*. SIAM Journal of Computing, 1(2):146–160, 1972.
- [TDB88] TAMASSIA, R., G. DI BATTISTA und C. BATINI: *Automatic Graph Drawing and Readability of Diagrams*. IEEE Transactions on System, Man, and Cybernetics, 18(1):61–79, 1988.
- [TG00] TRÜMBACH, D. und J. GASTEIGER: *Interaktive biochemische Stoffwechselwege*. In: *Proceedings of the Workshop Informationssysteme in der Biotechnologie*, Seiten 7–8, 2000.
- [The01] THE GENOME INTERNATIONAL SEQUENCING CONSORTIUM: *Initial Sequencing and Analysis of the Human Genome*. Nature, 409:860–921, 2001.
- [Thr94] THRO, E.: *Künstliches Leben*. Addison-Wesley, Bonn, 1994.
- [TINK98] TAKAI-IGARASHI, T., Y. NADAOKA und T. KAMINUMA: *A Database for Cell Signaling Networks*. Journal of Computational Biology, 5:747–754, 1998.
- [Tom00] Tom Sawyer Software. <http://www.tomsawyer.com>, 2000.
- [TT95] TAMASSIA, R. und I. G. TOLLIS (Herausgeber): *Proceedings of the 2nd International Symposium on Graph Drawing (GD'94)*, Band 894 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1995.
- [UMB00] *The University of Minnesota Biocatalysis/Biodegradation Database*. <http://www.labmed.umn.edu/umbdb/>, 2000.
- [Ven01] VENTER, J. C. ET AL.: *The Sequence of the Human Genome*. Science, 291:1304–1351, 2001.

- [VML96] VALLS, V., R. MARTI und P. LINO: *A Branch and Bound Algorithm for Minimizing the Number of Crossing Arcs in Bipartite Graphs*. Journal of Operational Research, 90:303–319, 1996.
- [vNM⁺00] VAN HELDEN, J., A. NAIM, R. MANCUSO, M. ELDRIDGE, L. WERNISCH, D. GILBERT und S. WODAK: *Representing and Analysing Molecular and Cellular Function in the Computer*. Biological Chemistry, 381(9–10):921–935, 2000.
- [Voi00] VOIT, E. O.: *Computational Analysis of Biochemical Systems*. Cambridge University Press, Cambridge, 2000.
- [Wad01] WADDLE, V.: *Graph Layout for Displaying Data Structures*. In: MARKS, J. [Mar01], Seiten 241–252.
- [War77] WARFIELD, J.: *Crossing Theory and Hierarchy Mapping*. IEEE Transactions on Systems, Man and Cybernetics, SMC-7(7):503–523, 1977.
- [WC53] WATSON, J. D. und F. H. C. CRICK: *Molecular Structure of Nucleic Acids: A Structure of Desoxynucleic Acids*. Nature, 171:738, 1953.
- [WE96] WACKETT, L. P. und L. B. ELLIS: *The University of Minnesota Biocatalysis/Biodegradation Database: A Novel Microbiological Method on the World Wide Web*. Journal of Microbiological Methods, 25:91–93, 1996.
- [WGKG83] WASER, M., L. GARFINKEL, M. C. KOHN und K. GARFINKEL: *Computer Modeling of Muscle Phosphofructokinase Kinetics*. Journal Theoretical Biology, 103:195–312, 1983.
- [Whi98] WHITESIDES, S. H. (Herausgeber): *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, Band 1547 der Reihe *Lecture Notes in Computer Science*. Springer Verlag, 1998.
- [Wie94] WIECHERT, W.: *Diskrete Analyse von Stoffwechselnetzwerken*. In: HAUBENSACK, F. und J. SÜHNEL [HS94], Seiten 100–111.
- [Wil86] WILSON, R. J.: *An Eulerian Trail Through Königsberg*. Journal of Graph Theory, 10(3):265–275, 1986.
- [WIT00] WIT - *What Is There. Interactive Metabolic Reconstruction on the WEB*. <http://wit.mcs.anl.gov/>, 2000.
- [WM96] WANG, X. und I. MIYAMOTO: *Generating Customized Layouts*. In: BRANDENBURG, F. J. [Bra96], Seiten 504–515.
- [Won00] WONG, M.: *Persönliche Kommunikation*, 2000.
- [Won01] WONG, L.: *PIES: A Protein Interaction Extraction System*. In: *Proceedings of the 6th Pacific Symposium on Biocomputing*, Seiten 520–531, 2001.
- [WR96] WILHELM, P. und K. ROTHEMUND: *A DNA and Restriction Enzyme Implementation of Turing Machines*. In: BAUM, E. B. und R. J. LIPTON [BL96], Seiten 75–120.

-
- [WTd98] WIECHERT, W., R. TAKORS und A. A. DE GRAAF: *Daten und Modelle im Metabolic Engineering*. In: HOFESTÄDT, R. [Hof98], Seiten 39–53.
- [Wuß89] WUSSING, H.: *Vorlesungen zur Geschichte der Mathematik*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1989.
- [WW95] WAGNER, F. und A. WOLFF: *An Efficient and Effective Approximation Algorithm for the Map Labeling Problem*. In: *Proceedings of the 3rd European Symposium on Algorithms (ESA'95)*, Band 979 der Reihe *Lecture Notes in Computer Science*, Seiten 420–433. Springer Verlag, 1995.
- [yFi00] *yFiles*. http://www.yworks.de/ger/products_yfiles_about.htm, 2000.
- [Yoe72] YOELI, P.: *The Logic of Automated Map Lettering*. *The Cartographic Journal*, 9(2):99–108, 1972.
- [YS99] YAMAGUCHI, A. und A. SUGIMOTO: *An Approximation Algorithm for the Two-Layered Graph Drawing Problem*. In: ASANO, T., H. IMAI, D. T. LEE, S. NAKANO und T. TOKUYAMA (Herausgeber): *Proceedings of the 5th International Conference on Computing and Combinatorics (COCOON'99)*, Band 1627 der Reihe *Lecture Notes in Computer Science*, Seiten 81–91. Springer Verlag, 1999.
- [Zor86] ZORASTER, S.: *Integer Programming Applied to the Map Label Placement Problem*. *Cartographica*, 23(3):16–27, 1986.
- [Zor90] ZORASTER, S.: *The Solution of Large 0-1 Integer Programming Problems Encountered in Automated Cartography*. *Operations Research*, 38(5):752–759, 1990.

Symbolverzeichnis

Biochemie

\rightarrow	Reaktionspfeil
R	biochemische Reaktion
$RW = (R_1, \dots, R_n)$	biochemischer Reaktionsweg
RN	biochemisches Reaktionsnetz

Mengen und Folgen

\mathbb{N}	Menge der natürlichen Zahlen (einschließlich 0)
\mathbb{R}	Menge der reellen Zahlen
$M = \{x_1, \dots, x_n\}$	Menge
$F = (x_1, \dots, x_n)$	Folge (geordnete Folge von Elementen einer Menge)
$x \in M, x \in F$	Element einer Menge, Element einer Folge
$ M $	Kardinalität (Mächtigkeit) einer Menge oder Folge
\cup	Vereinigung zweier Mengen
\cap	Durchschnitt zweier Mengen
\setminus	Differenz zweier Mengen
\subseteq	Teilmengenrelation
\emptyset	leere Menge
$M \times N$	kartesisches Produkt
$\mathcal{P}(M)$	Potenzmenge einer Menge
$\max(M)$	maximales Element einer Menge oder Folge
$\min(M)$	minimales Element einer Menge oder Folge
$A < B$	Vergleich zweier Folgen

Graphen

$G = (V, E)$	Graph, Hypergraph
V	Knotenmenge eines Graphen
E	Kantenmenge eines Graphen
$G = (V, E, B)$	benannter Graph
$G = (V, E, T)$	getypter Graph
$G = (V, E, M)$	geometrischer Graph
$G = (V, E, W)$	gewichteter Graph, Lagebeziehungsgraph
$G = (V, E, LB)$	Graph mit Lagebeziehungen
(u, v)	Kante von u nach v
$(v_1, \dots, v_k), v_1 \rightarrow^* v_k$	Weg von v_1 nach v_k
$\text{Ein}(v)$	Menge der eingehenden Kanten eines Knotens
$\text{Aus}(v)$	Menge der ausgehenden Kanten eines Knotens
$\text{Vor}(v)$	Menge der Vorgänger eines Knotens
$\text{Nach}(v)$	Menge der Nachfolger eines Knotens
$B(v), B(e)$	Benennung eines Knotens oder einer Kante
$T(v)$	Typ eines Knotens
$W(e)$	Gewicht einer Kante
$\Delta(G)$	maximaler Knotengrad
\subseteq	Teilgraphrelation
$G_{V'}$	durch die Knotenmenge V' induzierter Teilgraph
$\text{EK}(v)$	Menge der von v erreichbaren Knoten
$\text{ZK}(v)$	Menge der mit v zusammenhängenden Knoten
$\text{SZK}(v)$	Menge der mit v stark zusammenhängenden Knoten
$G_{\text{EK}(v)}$	Erreichbarkeitskomponente eines Knotens
$G_{\text{SZK}(v)}$	starke Zusammenhangskomponente eines Knotens
$G_{\text{ZK}(v)}$	Zusammenhangskomponente eines Knotens
$\delta(G)$	Dichte eines Graphen
$\text{TS}(v)$	topologische Sortierung eines Knotens
$\text{RK}(G, E_R)$	Graph, in dem Kanten aus E_R umgedreht sind

Reaktionsgraphen

$\text{RG} = (K \cup R, A \cup Z, B, T)$	Reaktionsgraph
K	Menge der Komponenten eines Reaktionsnetzes (<i>Substanzen, Cosubstanzen, Enzyme</i>)
R	Menge der Reaktionen eines Reaktionsnetzes
$B(v)$	Name einer Komponente
$T(v)$	Typ einer Komponente

Rand(G')	Rand des Teilgraphen G'
Bl(v)	Blätter eines Knotens (im zugehörigen Baum)
Bl(U)	Blätter einer Menge von Knoten (im zugehörigen Baum)
HRG = (RG, B)	hierarchischer Reaktionsgraph
Ausschnitt(U)	durch U gegebener Ausschnitt eines hierarchischen Reaktionsgraphen

Zeichnungen

$\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$	Ebene
$(x, y) \in \mathbb{R}^2$	Punkt der Ebene
Rechteck($h, b, (x, y)$)	Rechteck mit Höhe h , Breite b am Punkt (x, y)
Strecke($(x_1, y_1), (x_2, y_2)$)	Strecke von (x_1, y_1) nach (x_2, y_2)
Polygonzug($((x_1, y_1), \dots, (x_n, y_n))$)	Polygonzug von (x_1, y_1) über \dots nach (x_n, y_n)
$P(v), P(e)$	Platzierung (Koordinaten für Knoten und Stützpunkte für Kanten)
$Z(G, P)$	Zeichnung eines geometrischen Graphen mit Platzierung P
$x(v)$	x -Koordinate eines Knotens
$y(v)$	y -Koordinate eines Knotens
$y_o(v)$	obere Begrenzung eines Rechtecks/Knotens
$y_u(v)$	untere Begrenzung eines Rechtecks/Knotens
$x_l(v)$	linke Begrenzung eines Rechtecks/Knotens
$x_r(v)$	rechte Begrenzung eines Rechtecks/Knotens
Höhe(v)	Höhe eines Rechtecks/Knotens
Breite(v)	Breite eines Rechtecks/Knotens

Lagebeziehungen

(u, v, t, b)	Lagebeziehung von Knoten u zu Knoten v mit Typ t und Gewicht b
A(l)	Anfangsknoten einer Lagebeziehung
Z(l)	Zielknoten einer Lagebeziehung
T(l)	Typ einer Lagebeziehung
W(l)	Gewicht einer Lagebeziehung
o-u	oben-unten (Lagebeziehungstyp)
hor	horizontal (Lagebeziehungstyp)
l-r	links-rechts (Lagebeziehungstyp)
ver	vertikal (Lagebeziehungstyp)

Ebenen

$G = (L_1 \cup \dots \cup L_l,$ $E_1 \cup \dots \cup E_{l-1})$	l-Ebenenpartitionierung mit fester Reihenfolge der Knoten pro Ebene
$\text{Rang}(v)$	Rang eines Knotens
$\text{Spanne}(e)$	Spanne einer Kante
$L_i(v)$	relative Position eines Knotens auf einer Ebene
$R(v)$	Repräsentanten eines großen Knotens
$k_{u,v}$	paarweise Kreuzungszahl
$K(L_1, L_2)$	Zahl der Kreuzungen einer 2-Ebenenpartitionierung

Abbildungsverzeichnis

1.1	Biochemisches Reaktionsnetz	4
2.1	Strukturformel	16
2.2	Biochemische Reaktion	16
2.3	Mehrkomponentenreaktionen	16
2.4	Stoffwechselweg	19
2.5	Stoffwechselwege - Überblick	20
2.6	Spezielle Reaktionswege	21
2.7	Hierarchie biochemischer Reaktionen	23
2.8	Visualisierung	28
2.9	Visualisierungstechniken	29
2.10	Sichten	29
2.11	Widersprüchliche Ästhetikkriterien	33
3.1	Darstellung von Molekülen	41
3.2	Analyse herkömmlicher Darstellungen (Teil 1)	42
3.3	Darstellung von Reaktionen	44
3.4	Analyse herkömmlicher Darstellungen (Teil 2)	45
3.5	Darstellung des Reaktionspfeils und Komponentenanzordnung	47
3.6	Spezielle Reaktionswege	48
3.7	Analyse herkömmlicher Darstellungen (Teil 3)	49
3.8	Darstellungsformen für offene Zyklen	51
3.9	Navigation	53
3.10	Navigation - Verdeutlichung der Schritte	54
3.11	Sichten	55
3.12	Kontexterhaltende Darstellung	57
4.1	Statische Visualisierung biochemischer Reaktionsnetze (Teil 1)	61
4.2	Statische Visualisierung biochemischer Reaktionsnetze (Teil 2)	64
4.3	Statische Visualisierung biochemischer Reaktionsnetze (Teil 3)	66
4.4	Zeichenverfahren für Graphen (Teil 1)	70
4.5	Zeichenverfahren für Graphen (Teil 2)	72
4.6	Dynamische Visualisierung biochemischer Reaktionsnetze (Teil 1)	74
4.7	Dynamische Visualisierung biochemischer Reaktionsnetze (Teil 2)	76
4.8	Dynamische Visualisierung biochemischer Reaktionsnetze (Teil 3)	77
4.9	Dynamische Visualisierung biochemischer Reaktionsnetze (Teil 4)	80

4.10 Graph Drawing Contest '99	81
5.1 Hierarchieebenen	85
5.2 Reaktionsgraph	86
5.3 Rand, Blätter und Ebeneneindeutigkeit	87
5.4 Hierarchischer Reaktionsgraph	90
5.5 Ausschnitte hierarchischer Reaktionsgraphen (Teil 1)	92
5.6 Ausschnitte hierarchischer Reaktionsgraphen (Teil 2)	93
5.7 Hierarchie mit Teilreaktionen	94
6.1 Ebenenweise Zeichnungen	101
6.2 Nachbearbeitung einer Zeichnung	103
6.3 Geometrische topologische Platzierungen	105
6.4 Kompaktheit von Zeichnungen	106
6.5 Knoten-Kanten-Überdeckungen	107
6.6 Ebenenweises Zeichnen von Graphen (Gesamtverfahren)	112
6.7 Ebenenweises Zeichnen von Graphen (Teil 1)	114
6.8 Ebenenweises Zeichnen von Graphen (Teil 2)	115
6.9 Ebenenweises Zeichnen von Graphen (Teil 3)	116
6.10 Ansätze bei widersprüchlichen o-u-Lagebeziehungen	120
6.11 Orientierung der Kanten	126
6.12 Rechenzeiten des Algorithmus KANTENORIENTIERUNG	126
6.13 Algorithmen zur Kantensorientierung - Rechenzeiten (Teil 1)	130
6.14 Algorithmen zur Kantensorientierung - Rechenzeiten (Teil 2)	131
6.15 Algorithmen zur Kantensorientierung - Rechenzeiten (Teil 3)	132
6.16 Algorithmen zur Kantensorientierung - Rechenzeiten (Teil 4)	133
6.17 Ebenenübergreifende Auswirkung einer Knotenplatzierung	139
6.18 Algorithmen GRÖSSENECHTE PLATZIERUNG, EBENENAUFTEILUNG und EBENENEIN- FACHHEIT	140
6.19 Ebenen und temporäre Knoten	145
6.20 Entstehung vieler Ebenen	145
6.21 Rastergestützte Ebeneneinteilung	146
6.22 Spalten von Knoten	149
6.23 Auswirkungen der Rasterhöhe (Teil 1)	151
6.24 Auswirkungen der Rasterhöhe (Teil 2)	151
6.25 Auswirkungen der Rasterhöhe (Teil 3)	152
6.26 Auswirkungen der Rasterhöhe (Teil 4)	153
6.27 Kreuzungsreduzierung und große Knoten	157
6.28 Kreuzungen von Kanten	160
6.29 Algorithmus KREUZUNGSREDUZIERUNG (Teil 1)	163
6.30 Algorithmus KREUZUNGSREDUZIERUNG (Teil 2)	163
6.31 Algorithmus KREUZUNGSREDUZIERUNG (Teil 3)	163
6.32 Kreuzungsreduzierung mit großen Knoten	164
6.33 Vergleiche zur Kreuzungsreduzierung (Teil 1)	169
6.34 Vergleiche zur Kreuzungsreduzierung (Teil 2)	170

6.35	Vergleiche zur Kreuzungsreduzierung (Teil 3)	171
6.36	Vergleiche zur Kreuzungsreduzierung - Rechenzeit (Teil 1)	174
6.37	Vergleiche zur Kreuzungsreduzierung - Rechenzeit (Teil 2)	175
7.1	Knotenzentrum	181
7.2	Individuelle Abstände zwischen Knoten	183
7.3	Kantenhöhe	183
7.4	Kantenbreite	184
7.5	Ports	185
7.6	Obere und untere Ports	186
7.7	Seitliche Ports	186
7.8	Reihenfolge der Ports	187
7.9	Behandlung reflexiver Kanten	189
7.10	Mehrfache reflexive Kanten	189
7.11	Behandlung paralleler und antiparalleler Kanten	189
7.12	Gewichtete Kreuzungszahl	191
7.13	Globale l-r-Lagebeziehungen	193
7.14	Verwendung herkömmlicher Ebeneneinteilungen	194
7.15	Verwendung teilweiser Ebeneneinteilungen	195
7.16	Knoten- und Kantenbenennungen	197
7.17	Lokale Zeichenverfahren	199
7.18	Lokale Zeichenverfahren - Verwendung von Ports	200
7.19	Statische Stabilität (Teil 1)	203
7.20	Statische Stabilität (Teil 2)	204
7.21	Dynamische Stabilität	206
8.1	Knoten für Reaktion	209
8.2	Anordnung der Komponenten einer Reaktion	210
8.3	Probleme bei der Kantenorientierung	210
8.4	Kantenorientierung – erster Ansatz	211
8.5	Situationen bei der Kantenorientierung (Teil 1)	213
8.6	Situationen bei der Kantenorientierung (Teil 2)	213
8.7	Einfügen von Stützknoten (Teil 1)	217
8.8	Einfügen von Stützknoten (Teil 2)	218
8.9	Behandlung offener Zyklen	219
8.10	Offene Zyklen als Teil von Reaktionsnetzen	221
8.11	Behandlung geschlossener Zyklen	222
8.12	Geschlossene Zyklen als Teil von Reaktionsnetzen	224
8.13	Kontexterhaltende Navigation und Sichten (Teil 1)	225
8.14	Kontexterhaltende Navigation und Sichten (Teil 2)	226
8.15	Kontexterhaltende Navigation und Sichten (Teil 3)	227
8.16	Vergleich von Reaktionswegen	228
8.17	Verfolgung von Atomen	230
8.18	Quantitative Darstellungen	231
8.19	Mechanismen der Regulation	232

8.20 Rückkopplungshemmung	233
8.21 Lokale Darstellung der Regulation	235

Definitionsverzeichnis

Chemische Grundlagen	14
Biochemische Reaktion	15
Enzym	15
Biochemischer Reaktionsweg	18
Substanz, Cosubstanz	18
Geschlossene und offene Zyklen	20
Biochemisches Reaktionsnetz, Stoffwechsel	21
Graph	24
Eigenschaften von Graphen	24
Hypergraph	24
Benannter, getypter und gewichteter Graph (Hypergraph)	24
Pfad und Zyklus	25
Zusammenhang und Erreichbarkeit	25
Mengen zusammenhängender und erreichbarer Knoten	25
Spezielle Graphen	25
(starke) Zusammenhangskomponente, Erreichbarkeitskomponente	26
Topologische Sortierung	26
Ebene und Objekte	30
Platzierung von Graphen	31
Geometrischer Graph	31
Zeichnung geometrischer Graphen	31
Kardinalitätsgleiche Graphen	35
Isomorphe Graphen	35
Dichte eines Graphen	35
Reaktionsgraph	84
Rand, Blätter eines Knotens bzw. einer Menge, Ebeneneindeutigkeit	86
Hierarchischer Reaktionsgraph	89
Ausschnitt eines hierarchischen Reaktionsgraphen	89
Verfeinerung eines Ausschnitts, Vergrößerung eines Ausschnitts	91
Geometrische topologische Platzierung	103
Minimale geometrische topologische Platzierung	104
Graph mit Lagebeziehungen	108
Einfacher Graph	118
Umdrehen von Kanten	118
Erfüllte o-u-Lagebeziehung	119
Lagebeziehungsgraph	120

Globales Kantenorientierungs-Problem (GKP)	121
Lokales Kantenorientierungs-Problem (LKP)	121
Feedback Set Problem (FSP)	121
Feedback Set Entscheidungsproblem (FSEP)	121
Lokales Kantenorientierungs-Entscheidungsproblem (LKEP)	122
L-Ebenenpartitionierung	125
Eigenschaften von l-Ebenenpartitionierungen	137
Repräsentanten großer Knoten	157
Erfüllte l-r-Lagebeziehung in 2-Ebenenpartitionierung	158
Zulässige 2-Ebenenpartitionierung	158
2-LRG-Ebenenpartitionierung	159
Kreuzungszahl	160
2-LR-Ebenenpartitionierung	161
Kreuzungsreduzierungs-Problem (KRP)	161
Crossing Problem (CP)	161
Benennungsproblem	194

Index

- adjazent, *siehe* Knoten
Ästhetikkriterien, 32–33, 110, 111
aktives Zentrum, *siehe* Enzym
Algorithmus
 VGL, 112–113, 118, 137, 156, 177
 EBENENAUFTEILUNG, 142
 EBENENEINFACHHEIT, 142
 EBENENGESTÜTZTE KANTENORIENTIERUNG, 117, 127
 EBENENPARTITIONIERUNG, 180
 GLOBALE KREUZUNGSREDUZIERUNG, 156
 GRÖSSENECHTE PLATZIERUNG, 141
 GRÖSSENECHTE EBENENEINTEILUNG, 137
 KANTENORIENTIERUNG, 117, 124, 214
 KREUZUNGSREDUZIERUNG, 165
 RASTERGESTÜTZTE EBENENEINTEILUNG, 137
 RASTERGESTÜTZTE EBENENPARTITIONIERUNG, 147, 180
 Sugiyama, 100, 180, 193
Anfangsknoten, *siehe* Knoten
Anforderungen, 38–58
 Darstellung offener und geschlossener Zyklen, 46–50
 Darstellung von Reaktionen, 43–46
 Darstellung von Reaktionswegen und -netzen, 46–50
 Darstellung von Substanzen, Cosubstanzen und Enzymen, 40–43
 globale, 58
 Grundlagen, 39–40
 kontexterhaltende Navigation und Sichten, 52–56, 58
 lokale, 58
 Visualisierung, 38
 Ziele, 38–39
Angriffsort, *siehe* Regulationsstelle
Anker, 184
Atlas *Biochemical Pathways*, 18, 22, 39, 56, 60
Atom, 14
 Verfolgung, 226–229
Ausgangsgrad, *siehe* Knoten
Ausgangsstoff, *siehe* Edukt
Ausgangssubstanz, *siehe* Edukt
Ausschnitt, 7
 hierarchischer Reaktionsgraphen, *siehe* Reaktionsgraph
azyklischer Graph, *siehe* Graph
Baum, 26
Benennung von Knoten und Kanten, *siehe* Knoten bzw. Kante
Benennungsproblem, 194
Beseitigung von Zyklen, *siehe* Kantensorientierung
Bindung, *siehe* chemische Bindung
biochemische Reaktion, *siehe* Reaktion
biochemischer Reaktionsweg, *siehe* Reaktionsweg
biochemisches Reaktionsnetz, *siehe* Reaktionsnetz
Bioinformatik, 2, 3
BioJake, 75
BioPath, 8
bipartiter Graph, *siehe* Graph
Blätter, 26
 einer Menge, 87
 eines Knotens, 87
chemische Bindung, 14
chemische Verbindung, *siehe* Molekül
chemisches Element, 14
Citratzyklus, 18, 63, 65
Coedukt, *siehe* Edukt
Coenzym, *siehe* Enzym
Cofaktor, 17
Constraint, *siehe* Lagebeziehung
Coproduct, *siehe* Produkt

- Cosubstanz, *siehe* Substanz
Crossing Problem, 161
- DAG, 26
Dichte, 35
DNA, 2, 3
DNA-Chip, 7, 229
DNA-Computer, 2
DNA-Mikroarray, *siehe* DNA-Chip
dynamische Visualisierung, *siehe* Visualisierung
- Ebene, 30, 112
 i-te, 137
 makroskopische, 14
 mikroskopische, 14
 Objekt, 30
 Punkt, *siehe* Punkt
- Ebeneneinteilung
 Arten, 105
 globale, 104
 lokale, 106
- Ebenenpartitionierung, 112, 136–154
 1-, 125, 137, 156
 1-, einfache, 137, 143
 2-, 157
 2-, zulässige, 158
 2-LR-, 161
 2-LRG-, 159
 weitere, 192–193
- EC-Nomenklatur, 22–23
EC-Nummer, *siehe* EC-Nomenklatur
EcoCyc, 71–73
Edukt, 15, 18
 Co-, 18, 210
Edukt-Enzym-Komplex, 17
Eingangsgrad, *siehe* Knoten
Einkomponentenreaktion, *siehe* Reaktion
EMP, *siehe* MPW
Endknoten, *siehe* Knoten
Endstoff, *siehe* Produkt
Endsubstanz, *siehe* Produkt
Enzym, 3–6, 15, 84, 208, 229
 aktives Zentrum, 15, 231
 Aktivität, 229
 Anforderungen an die Darstellung, *siehe*
 Anforderungen
 Bild, 41
 Co-, 17
 Hierarchie, 22–23
 Klassifikation, *siehe* EC-Nomenklatur
 Name, 40
 Regulation der Aktivität, *siehe* Regulation
- ER-Diagramm, 7
Erreichbarkeit, 25
Erreichbarkeitskomponente, 26
Erweiterung, *siehe* Navigation
ExPASy, 39, 62
Experiment, 35–36
- Feedback Arc Set Entscheidungsproblem, 121
Feedback Set Problem, 121
Fisheye-View, 6
Fläche einer Zeichnung, *siehe* Zeichnung
Folgen von Zeichnungen, 201–205
funktionelle Gruppe, 23
- Gen, 3–6, 14
genetische Algorithmen, 2
Genom, 3
geometrischer Graph, *siehe* Graph
Glycolyse, 18, 62, 66
Grad, *siehe* Knoten
Graph, 24
 azyklischer, 25, 112, 119
 benannter, 24
 bewerteter, 25
 bipartiter, 26
 Dichte, *siehe* Dichte
 dichter, 35
 einfacher, 118
 geometrischer, 31
 gerichteter, 24
 gerichteter azyklischer, *siehe* DAG
 getypter, 25
 gewichteter, 25
 hierarchischer, 86
 kantenorientierter, 117
 lichter, 35
 mit Lagebeziehungen, *siehe* Lagebeziehung

-
- planarer, 69
 - Reaktions-, *siehe* Reaktionsgraph
 - stark zusammenhängender, 25
 - Teil-, *siehe* Teilgraph
 - Visualisierung, *siehe* Visualisierung
 - Zeichnung, *siehe* Zeichnung
 - zusammenhängender, 25
- Graphen
- isomorphe, 35
 - kardinalitätsgleiche, 35
- Heuristik, 35–36, 117
- Hierarchie, 56
- biochemischer Reaktionen, *siehe* Reaktion der Enzyme, *siehe* Enzym
 - eindeutige, 22
 - vollständige, 22
- hierarchischer Reaktionsgraph, *siehe* Reaktionsgraph
- hor-Lagebeziehung, *siehe* Lagebeziehung
- Hypergraph, 24
- benannter, 24
 - gerichteter, 24
 - getypter, 25
 - gewichteter, 25
- Hyperkante, 24, *siehe* Kante
- inkrementelle Zeichenverfahren, *siehe* Zeichenverfahren
- inzident, *siehe* Kante, Knoten
- Ion, 14, 15
- Kante, 24
- antiparallele, 188
 - ausgehende, 24
 - Benennung, 25, 194–196
 - Breite, 183
 - eingehende, 24
 - Hyper-, *siehe* Hyperkante
 - Höhe, 181
 - innere, 185
 - inzidente, 24
 - parallele, 188–190
 - reflexive, 188
 - Schlinge, 24
 - Spanne, 137
 - Stützpunkte, 113
 - temporäre, 113
- Kanten, 24
- Kreuzungen, *siehe* Kreuzungen
 - orthogonales Routing, 184
 - Routing, 183–184
 - Umdrehen, 118
- Kantenorientierung, 112, 117–135, 209–216
- globale, 117, 121
 - lokale, 117, 121, 122
 - lokales Entscheidungsproblem, 122
- Katalyse, 15
- KEGG, 39, 56, 62–63
- Knoten, 24
- Abstände, 180–181
 - adjazenter, 24
 - Anfangs-, 24
 - Ausgangsgrad, 24
 - Benennung, 25, 194–196
 - Breite, 31
 - Eingangsgrad, 24
 - End-, 24
 - erreichbarer, 25
 - große, 101, 157
 - Höhe, 31
 - innere, 26
 - inzidenter, 24
 - Nachfolger, 24
 - Position, 137
 - Rang, 137
 - stark zusammenhängende, 25
 - temporärer, 113
 - Vorgänger, 24
 - Zentrum, 180
 - zusammenhängende, 25
- Komplexitätstheorie, 35
- Kontext, 6
- kontexterhaltende Visualisierung, *siehe* Visualisierung
- Koordinate, 30
- x, 31, 113, 176–177, 181
 - y, 31, 136–154
- Kreuzungsreduzierung, 113, 155–173
- globale, 156–157
- Kreuzungsreduzierungs-Problem, 161
-

- Kreuzungszahl, 160
 - gewichtete, 190–191
- künstliches Leben, 2
- Lagebeziehung, 108–110
 - hor-, 108, 191
 - l-r-, 108, 109
 - l-r-, erfüllte, 158
 - l-r-, globale, 192
 - o-u-, 108, 134
 - o-u-, erfüllte, 119
 - ver-, 108, 192
- Lagebeziehungen
 - gewichtete, 119
 - Graph mit, 108
 - widersprüchliche, 109
- Lagebeziehungsgraph, 120
- lexikografische Ordnung, 118
- lineare Segmente, 176
- l-r-Lagebeziehung, *siehe* Lagebeziehung

- maximaler Knotengrad, 24
- Mehrfachkante, *siehe* Kante, parallele
- Mehrkomponentenreaktion, *siehe* Reaktion
- Menge
 - ebeneneindeutige, 87
- Mental Map, 202
- Metabolismus, *siehe* Stoffwechsel
- Metabolite, 18
- MetaCyc, *siehe* EcoCyc
- Mindestabstand, *siehe* Knoten, Abstände
- Modellierung von Reaktionen, *siehe* Reaktion
- Molekül, 14
 - Struktur, 15
- MPW, 56, 63–65

- Nachfolger, *siehe* Knoten
- Name, 25
- Navigation, 6, 22, 50–53
 - Erweiterung, 50, 52
 - kontexterhaltende, 52–56, 223
 - Reduktion, 50, 52
 - Ursprungszeichnung, *siehe* Zeichnung
 - Verfeinerung, 50, 52, 56, 91, 223
 - Vergrößerung, 50, 52, 56, 91, 223
- neuronale Netze, 2

- O-Notation, 35
- Objekt, *siehe* Ebene
- Orientierung der Kanten, *siehe* Kantenorientierung
- o-u-Lagebeziehung, *siehe* Lagebeziehung

- PathDB, 76
- PERT, 181
- PERT-Diagramm, 7
- Pfad, 25
- PFBP, 73–75
- Platzierung, 31
 - geometrische topologische, 103
 - größenebenenmaximale, 104
 - größenechte, 104, 137, 138, 141
 - größenfreie, 103
 - größenmaximale, 104
 - minimale geometrische topologische, 104
- Polygonzug, 30–32
- Polylinie, *siehe* Polygonzug
- Port, 184–188
 - oberer, 185
 - seitlicher, 185–187
 - unterer, 185
- Poster *Biochemical Pathways*, 4, 18, 22, 39, 56, 62
- Produkt, 15, 18
 - Co-, 18, 210
- Produkt-Enzym-Komplex, 17
- Protein, 3, 5
 - faltung, 3
- Proteom, 4
- Punkt, 30

- Rand, 87
- Rang, *siehe* Knoten
- Rasterhöhe, 144
- Reaktant, *siehe* Edukt
- Reaktion, 3–6, 15–18, 84, 208–209
 - abstrakte, 89
 - Anforderungen an die Darstellung, *siehe* Anforderungen
 - Einkomponenten-, 17
 - Hierarchie, 6, 22, 56
 - Komponenten, 15, 16, 40

- Mehrkomponenten-, 16, 17
Modellierung, 68
reversible, 17, 216
Richtung, 17, 43
Teil-, 17, 22
Visualisierung, 40
- Reaktionsgleichgewicht, 17
- Reaktionsgraph, 84–86, 208
Ausschnitt, 89
hierarchischer, 84, 86–91, 223
Verfeinerung eines Ausschnitts, 91
Vergrößerung eines Ausschnitts, 91
- Reaktionsnetz, 3, 5–7, 21–22, 209–223
Analyse und Modellierung, 3
Anforderungen an die Darstellung, *siehe*
Anforderungen
dynamische Visualisierung, 68–79
Navigation, *siehe* Navigation
Sichten, *siehe* Sichten
statische Visualisierung, 60–67
Visualisierung, 5–7, 38
- Reaktionspfeil
gemeinsamer, 43
Gestalt, 43
Richtung, 43
- Reaktionsprodukt, *siehe* Produkt
- Reaktionsweg, 3, 18–21, 216–223
Anforderungen an die Darstellung, *siehe*
Anforderungen
geschlossener Zyklus, 20–21, 46, 220–223
offener Zyklus, 20–21, 46, 216–220
- Reaktionswege
Vergleich von, 226
- Rechteck, 30–32, 109
Begrenzung, 31
- Regulation, 17, 229–234
Angriffsort, *siehe* Regulationsstelle
Ebenen, 229
Förderung, 231
Hemmung, 231
langsame, 229
Rückkopplungshemmung, 232
schnelle, 229, 231
Vorwärtsförderung, 232
- Regulationsstelle, 229, 231
- Regulator, 231
- Repräsentant für großen Knoten, 157
- Routing, *siehe* Kanten
- Schlinge, *siehe* Kante
- Schlüsselsubstanz, 18
- Scrollen, 6
- Segmentordnungsgraph, 177
- Sequenzvergleich, 3
- Sicht, 6, 27, 29, 52
kontexterhaltende, 52–56, 223
- Splines, 183–184
- Stabilität, 202–205, 223
dynamische, 202–205
statische, 202–203
- statische Visualisierung, *siehe* Visualisierung
- Stoff, 14
- Stoffwechsel, 22, 225, 229
- Stoffwechselweg, 18, 22
amphiboler, 22
anaboler, 22
kataboler, 22
- Strecke, 30
- Struktur eines Moleküls, *siehe* Molekül
- Strukturformel, 15, 16
vereinfachte, 15, 16
- Stützknoten, 215
- Substanz, 15, 18, 84
Anforderungen an die Darstellung, *siehe*
Anforderungen
Bild, 41
Co-, 18, 84, 208
Name, 40
- Substrat, *siehe* Edukt
- Teilgraph, 26
induzierter, 26
- Teilreaktion, *siehe* Reaktion
- temporäre Kante, *siehe* Kante
- temporärer Knoten, *siehe* Knoten
- Tiefensuche, 134
- topologische Sortierung, 26
- Transkription, 229
- Translation, 229
- Übersichtsreaktion, *siehe* Stoffwechselweg

- UM-BBD, 39, 65, 76–77
- UML, 7, 101
- Ursprungszeichnung, *siehe* Zeichnung

- ver-Lagebeziehung, *siehe* Lagebeziehung
- Verfeinerung, *siehe* Navigation
- Vergößerung, *siehe* Navigation
- Visualisierung, 7, 26–27
 - Anforderungen, *siehe* Anforderungen
 - biochemischer Information, 3
 - biochemischer Reaktionsnetze, *siehe* Reaktionsnetz
 - Erzeugung, 27
 - kontexterhaltende, 6
 - Sichten, *siehe* Sicht
 - Techniken, 27, 29, 32
- Vorgänger, *siehe* Knoten

- WIT, 39, 63–65
- Wurzel, 26

- yFiles, 76–77

- Zeichenkonventionen, 32–33, 110–111
- Zeichenverfahren
 - ebenenweise, 8, 34, 100–113
 - inkrementelle, 201–202
 - Klassifikation, 33–34
 - kräftebasierte, 34, 101
 - lokale, 196–201
 - orthogonale, 34, 101
 - Stabilität, *siehe* Stabilität
- Zeichnung
 - Berechnung, 32
 - Erzeugung, 32
 - Fläche, 32
 - geometrischer Graphen, 31
 - Ursprungs-, 52
 - von Graphen, 30–34
- Zelle, 3, 13
- Zellkompartiment, 22
- zelluläre Automaten, 2
- Zitronensäurezyklus, *siehe* Citratzyklus
- Zoomen, 6
- Zuordnung der Knoten zu Ebenen, *siehe* Ebenenpartitionierung

- Zusammenhang, 190
- Zusammenhangskomponente, 26
 - starke, 26
- Zyklus, 25
 - geschlossener, *siehe* Reaktionsweg
 - offener, *siehe* Reaktionsweg