OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

Fakultät für Informatik
Institut für Simulation und Graphik

DIPLOMARBEIT

Layout hierarchisch strukturierter Graphen am Beispiel von Daten aus der Systembiologie

Erik Aschenbrenner

OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG
Fakultät für Informatik
Institut für Simulation und Graphik

MAX-PLANCK-INSTITUT MAGDEBURG
Arbeitsgruppe Systembiologie

DIPLOMARBEIT

Layout hierarchisch strukturierter Graphen am Beispiel von Daten aus der Systembiologie

von:	ERIK ASCHENBRENNER
geboren am:	26.06.1981
in:	Halberstadt
Matrikelnummer:	162772
Studiengang:	Computervisualistik

1. Gutachter:	Prof. Dr.-Ing. BERNHARD PREIM
2. Gutachter:	Prof. Dr. rer. nat. FALK SCHREIBER
Betreuer:	Prof. Dr.-Ing. BERNHARD PREIM
	Dipl.-Ing. SEBASTIAN MIRSCHEL

Bearbeitungszeit: 14.09.2007 – 14.02.2008

ERKLÄRUNGEN:

Der Autor dieser Arbeit verwendet für Erläuterungen, die Personen einbeziehen, die männliche Form der Beschreibung, wie *Anwender* oder *Nutzer*. Das schließt natürlich weibliche *Anwender*- und *Nutzerinnen* nicht aus.

Zu nicht allgemein bekannten Begriffen und Abkürzungen, wie z.B. *EGF*, die im Text nicht weiter erklärt werden, befinden sich Erläuterungen im Begriffs- und Abkürzungsverzeichnis dieser Arbeit.

Selbständigkeitserklärung

Hiermit versichere ich, Erik Aschenbrenner (Matrikel-Nr. 162772), die vorliegende Arbeit allein und nur unter Verwendung der angegebenen Quellen angefertigt zu haben.

Erik Aschenbrenner

Magdeburg, den 14.02.2008

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die mir beim Erstellen dieser Arbeit durch ihre Unterstützung zur Seite standen.

Zu allererst sei meinem Betreuer Sebastian Mirschel gedankt. Dieser war mir durch seine motivierende Art und seine Hilfe bei organisatorischen, technischen und inhaltlichen Fragen eine große Unterstützung bei der Umsetzung und dem Schreiben dieser Arbeit.

Ebenso danke ich Professor Dr. Bernhard Preim für die Betreuung und die nützlichen Ratschläge, die zur Verbesserung dieser Arbeit beitragen konnten.

Bei Professor Dr. Falk Schreiber bedanke ich mich für die hilfreichen Hinweise im Vorfeld dieser Arbeit und dafür, dass er sich spontan als Gutachter zur Verfügung stellte.

Weiterer Dank geht an das Max-Planck-Institut für Dynamik komplexer technischer Systeme in Magdeburg, das mir die Anfertigung dieser Diplomarbeit in einer idealen Arbeitsumgebung ermöglichte. Besonders bei den Mitarbeitern der Arbeitsgruppe Systembiologie bedanke ich mich für die freundliche Aufnahme ins Team und die gute Zusammenarbeit.

Meiner Freundin Franzi danke ich für ihre Liebe und ihre Hilfe bei der Korrektur von Rechtschreibung und Kommasetzung.

Zu guter Letzt möchte ich mich herzlich bei meiner gesamten Familie für die Unterstützungen in jeglicher Hinsicht während der Dauer der Diplomarbeit und des Studiums bedanken.

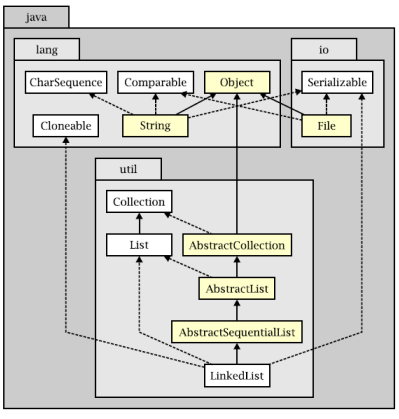
Inhaltsverzeichnis

1 Einleitung

In vielen Bereichen des täglichen Lebens und der Wissenschaft entstehen und existieren große und komplexe Datenmengen, welche in Textform oder anhand von Tabellen schwer zu überschauen sind. Eine graphische Darstellung erleichtert in vielen Fällen den Umgang mit diesen Daten. Sie kann dabei helfen einen Überblick über die Datenmenge zu bekommen, komplexe Fragestellungen zu beantworten oder bisher nicht bekannte Zusammenhänge in den Daten zu erkennen. Bestehen zwischen den Datenelementen Beziehungen, so werden häufig Graphen zur Visualisierung dieser Daten benutzt [HMM00]. Anwendungsbeispiele hierfür sind Darstellungen biochemischer Reaktionsnetzwerke, Visualisierungen von Programmabläufen in der Softwaretechnik oder Karten von U-Bahn-Netzen.

Sollen Inklusionsinformationen dargestellt werden, reichen einfache (flache) Graphen jedoch nicht aus. In diesem Fall werden oft hierarchisch strukturierte Graphen, sogenannte *Compound-Graphen* verwendet. Abbildung 1.1 zeigt zwei Anwendungsbeispiele solcher Graphen. In Abbildung 1.1(a) ist ein Diagramm aus der Softwaretechnik abgebildet, welches Verbindungen zwischen Klassen der Programmiersprache JAVA und die Einordnung der Klassen in Pakete, welche wiederum zu weiteren Paketen zugeordnet werden, darstellt. Auch in der Biologie ist das Wissen zu großen Teilen hierarchisch gegliedert: Organismen beinhalten Organe, welche wiederum aus Zellen aufgebaut sind. Zellen bestehen, wie in Abbildung 1.1(b) dargestellt, aus biochemischen Komponenten, wie Proteinen oder Proteinkomplexen, zwischen denen Reaktionen ablaufen.

(a)



(b)

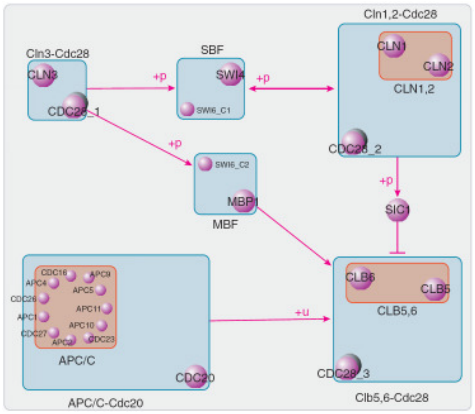


Abbildung 1.1: Beispiele für die Anwendung von Compound-Graphen: (a) Verschachtelung und Abhängigkeiten unter Klassen der JAVA-*API* aus [For04] und (b) Zusammenfassung von Proteinen zu Proteinkomplexen [HMW+07] zur Darstellung von Vorgängen während des Zellzyklus der Backhefe.

Weiterer Einsatzbedarf für Compound-Graphen besteht dort, wo große, komplexe Systeme nachgebildet und modelliert werden. Besonders Modelle aus dem Bereich Systembiologie zeichnen sich durch eine hohe und ständig wachsende Komplexität aus. Um diese komplexen Modelle dennoch analysieren zu können, ist oft eine Strukturierung der Modelle in Module und Hierarchien hilfreich. Compound-Graphen ermöglichen es hier, diese Informationen sinnvoll zu strukturieren. Dadurch kann sowohl die Analyse als auch die Visualisierung des Modells, aus Sicht des Benutzers und in Hinblick auf die technischen Ressourcen, effizienter gestaltet werden. Abbildung 1.2 zeigt zwei Darstellungen eines komplexen Modells aus der Systembiologie. In Abbildung 1.2(a) ist dieses durch einen einfachen, unstrukturierten Graphen visualisiert. Abbildung 1.2(b) stellt das gleiche Modell hierarchisch strukturiert dar, wodurch die Übersichtlichkeit im Vergleich zur unstrukturierten Darstellung deutlich erhöht wird.

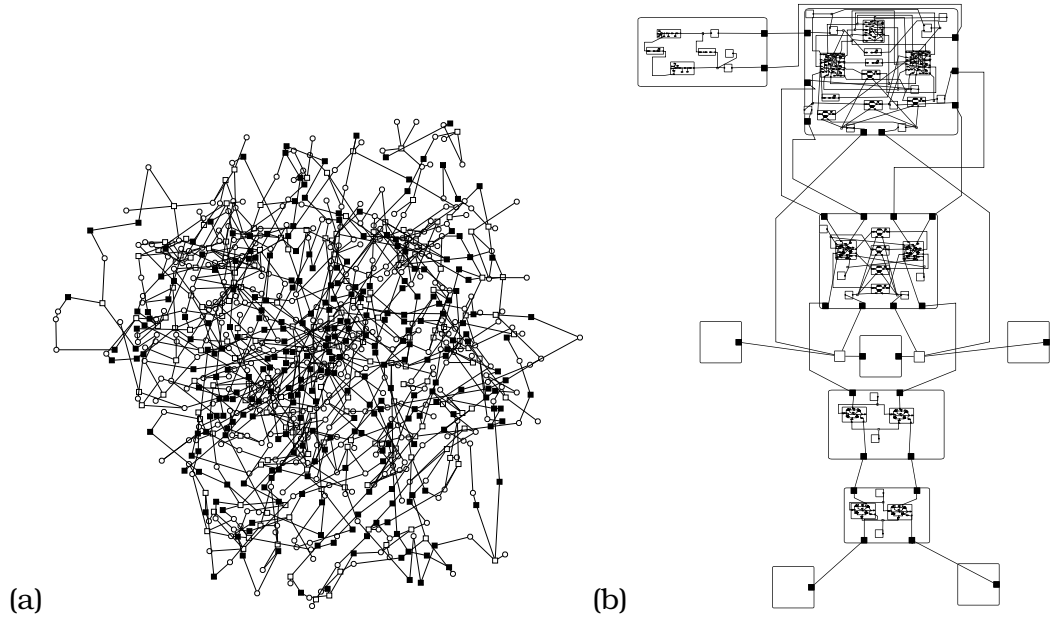


Abbildung 1.2: Verschiedene Darstellungen eines Modells des *EGF*-Rezeptor-Netzwerks: (a) Unstrukturiert und (b) nach Strukturierung und manuellem Layout. Durch die Strukturierung wird die Übersichtlichkeit der Darstellung deutlich erhöht.

1.1 Motivation

Biologische Netzwerke[1] werden traditionell als Graphen modelliert und visualisiert. Diese Art der graphischen Darstellung hilft Biologen beim Verständnis des Aufbaus biologischer Netzwerke und der innerhalb dieser Netzwerke ablaufenden Prozesse. Für eine aussagekräftige und intuitiv verständliche visuelle Darstellung der durch einen Graphen repräsentierten Informationen ist sein *Layout*, also die Anordnung seiner Elemente, von großer Bedeutung. Hierzu ist es wichtig, dass das Layout so-

1 Im Kontext dieser Arbeit werden molekularbiologische und biochemische Netzwerke zur Vereinfachung mit dem Begriff „biologische Netzwerke“ [BBS08] bezeichnet.

wohl allgemeine Anforderungen (z.B. keine Knotenüberlappungen) als auch anwendungsspezifische Anforderungen (z.B. Darstellung von Reaktionsrichtungen) erfüllt.

In der Systembiolgie ist es üblich, dass Modelle biologischer Netzwerke oder auch ganzer zellulärer Systeme textuell, auf Basis mathematischer Gleichungen, beschrieben werden. Diese Beschreibungen enhalten in der Regel keine Layoutinformationen. Moderne Modellierungswerkzeuge erlauben dem Benutzer die graphische Erstellung und Manipulation von Modellen in Form von Reaktionsgraphen. Für den Austausch der Modelle zwischen verschiedenen Modellierungswerkzeugen wird zunehmend das Standard-Austauschformat *SBML* eingesetzt, welches aber neben der Speicherung der mathematischen Beschreibung eines Modells, die Speicherung von Layoutinformationen nicht vorsieht. Sind keine Layoutinformationen vorhanden, ist eine graphische Darstellung des Modells ohne Weiteres nicht möglich. Existiert keine Möglichkeit zur automatischen Erstellung eines geeigneten Layouts, muss dieses durch den Benutzer manuell erstellt werden. Dies ist gerade in Hinblick auf die Größe und Komplexität existierender Modelle eine sehr zeitaufwändige, mühselige und fehleranfällige Aufgabe.

Aus diesem Grund ist ein automatisches Verfahren zur Erstellung eines aussagekräftigen Layouts notwendig. Da angesichts der Größe und Komplexität existierender Modelle in der Systembiologie Compound-Graphen zur Darstellung der Modelle genutzt werden, sollte hierbei die inhärente Struktur, d.h. die vorhandene Strukturierung der Modelle in Module, in den Layoutprozess integriert werden. Es existieren bereits Verfahren zur automatischen Generierung eines Layouts von Compound-Graphen. Diese beachten jedoch weder die besondere Anforderung an das Layout von *Terminalknoten*, welche die Schnittstellen zwischen Modulen darstellen und daher immer am Rand eines Moduls platziert werden müssen, noch eine Skalierung von Elementen aus verschiedenen Hierarchiebenen, welche zur verbesserten Darstellung der Hierarchie benutzt wird (siehe Abbildung 1.2(b)).

1.2 Ziel

Das Ziel dieser Diplomarbeit ist die Entwicklung eines Algorithmus zum automatischen Layout der beschriebenen Compound-Graphen. Dabei sind folgende Anforderungen zu beachten:

- Da biologische Netzwerke häufig als ungerichtete Graphen modelliert werden, muss der Algorithmus auf diese Graphen anwendbar sein.
- Werden zur verbesserten Visualisierung der Hierarchie die Elemente entsprechend ihrer Hierarchiebene skaliert dargestellt (siehe Abbildung 1.2(b)), so muss diese Skalierung durch den Algorithmus beachtet werden.
- Zur Herstellung der Verbindungen zwischen Knoten aus verschiedenen Modulen werden Terminalknoten benutzt. Diese müssen immer am Rand eines Moduls platziert werden.

Da der Algorithmus im Bereich der Systembiologie eingesetzt werden soll, ist ein weiteres Ziel, anwendungsspezifische Informationen, soweit vorhanden, herauszu-

arbeiten und die Beachtung dieser in den Algorithmus einzubinden. Der Algorithmus soll in das Modellierungswerkzeug PROMOT integriert und an Realmodellen getestet werden.

1.3 Gliederung

Diese Arbeit ist in folgende Kapitel gegliedert:

Kapitel 2 gibt eine Einführung in die Grundlagen aus dem Forschungsgebiet des Graphzeichnens (engl. *Graph Drawing*), die für das Verständnis dieser Arbeit relevant sind. Es werden zunächst Anforderungen an Zeichnungen von allgemeinen Graphen erläutert und die verbreitetsten Verfahren zum Zeichnen von Graphen vorgestellt. Anschließend werden Anforderungen an die Darstellung biologischer Netzwerke genannt. Das Kapitel endet mit der Auswahl eines Zeichenverfahrens, welches als Basis für das in dieser Arbeit entwickelte Zeichenverfahren für Compound-Graphen dient.

Kapitel 3 gibt einen Überblick über existierende Verfahren zum Zeichnen von Compound-Graphen. Weiterhin werden Techniken zur Integration biologischer Informationen, in das in Kapitel 2 ausgewählte Zeichenverfahren, vorgestellt.

Kapitel 4 beschäftigt sich mit dem Entwurf des in dieser Arbeit entwickelten Zeichenverfahrens. Zu Beginn des Kapitels werden spezielle Anforderungen an die Darstellung von Compound-Graphen erläutert. Im Anschluss daran, werden das Zeichenverfahren und seine Unterschiede gegenüber existierenden Verfahren vorgestellt und diskutiert. Es wird gezeigt, wie zur Verbesserung der Qualität der erzeugten Layouts, biologische Informationen in das Zeichenverfahren integriert werden können. Das Kapitel endet mit der Betrachtung von Möglichkeiten, Anwender in den automatischen Layoutprozess einzubeziehen.

Kapitel 5 beschreibt die Integration des Zeichenverfahrens in das Modellierungswerkzeug PROMOT. Hierzu wird zunächst der Aufbau von PROMOT vorgestellt. Anschließend werden notwendige Erweiterungen zur Umsetzung des Zeichenverfahrens sowie bei der Implementierung aufgetretende Probleme erläutert.

Kapitel 6 beschäftigt sich mit der Auswertung des Zeichenverfahrens. Hierzu werden Experimente zum Vergleich des Zeichenverfahrens mit existierenden Verfahren beschrieben und durchgeführt. Des Weiteren werden die erzielten Ergebnisse in Bezug auf Laufzeit und Qualität bewertet.

Kapitel 7 fasst die Erkenntnisse über das in dieser Arbeit entwickelte Zeichenverfahren zusammen. Es werden die Eigenschaften des Zeichenverfahrens diskutiert und die erzielten Ergebnisse mit den Zielen dieser Arbeit verglichen. Weiterhin werden Möglichkeiten zur Optimierung und Erweiterung des Zeichenverfahrens genannt, die im zeitlichen Rahmen dieser Arbeit nicht umgesetzt werden konnten.

2 Grundlagen

In diesem Kapitel werden zunächst (die für das Verständnis der Arbeit benötigten) Grundlagen aus dem Bereich des Graphzeichnens erläutert, grundlegende Anforderungen an das Layout allgemeiner Graphen genannt sowie die verbreitetsten Zeichenverfahren für Graphen vorgestellt. Anschließend wird ein kurzer Überblick über die Visualisierung biologischer Netzwerke gegeben und Anforderungen erläutert, die bei der Visualisierung biologischer Netzwerke zu beachten sind. Das Kapitel endet mit der Auswahl eines Zeichenverfahrens, welches als Basis für den Algorithmus zum Zeichen von hierarchisch strukturierten biologischen Netzwerken dient.

2.1 Zeichnen von Graphen

Wie in der Einleitung erläutert, werden abstrakte, relationale Daten, häufig mit Hilfe der *Graph-Metapher* dargestellt. In dieser werden Objekte als Knoten und die Beziehungen zwischen den Objekten als Kanten eines Graphen oder Netzwerks[1] angesehen. Die durch den Graphen repräsentierten Informationen können dabei entweder textuell, in Tabellenform (z.B. als Adjazenzmatrix) oder grafisch dargestellt werden.

Der im folgenden Beispiel betrachtete Graph stellt einen Teil der Vorgänge während des Zellzyklus der Backhefe dar. Proteine oder Proteinkomplexe werden hierbei durch die Knoten des Graphen, Reaktionen oder Übergänge zwischen Proteinen durch die Kanten des Graphen, repräsentiert. In Tabelle 2.1 ist dieser Graph als Adjazenzmatrix dargestellt. Bei dieser Art der Darstellung wird die Existenz einer Kante zwischen zwei Knoten mit 1 und die Nichtexistenz einer Kante zwischen zwei Knoten mit 0 gekennzeichnet. In Abbildung 2.1 ist der gleiche Graph in zwei unterschiedlichen graphischen Darstellungen abgebildet.

	Cln3-Cdc28	SBF	Cln1,2-Cdc28	SIC1	Clb5,6-Cdc28	APC/C,Cdc20	MBF
Cln3-Cdc28	0	1	0	0	0	0	1
SBF	1	0	1	0	0	0	0
Cln1,2-Cdc28	0	1	0	1	0	0	0
SIC1	0	0	1	0	1	0	0
Clb5,6-Cdc28	0	0	0	1	0	1	1
APC/C,Cdc20	0	0	0	0	1	0	0
MBF	1	0	0	0	1	0	0

Tabelle 2.1: Darstellung eines Teils des Zellzyklus der Backhefe als Adjazenzmatrix

1 Im Kontext dieser Arbeit werden die Begriffe *Graph* und *Netzwerk* als Synonyme verwendet.

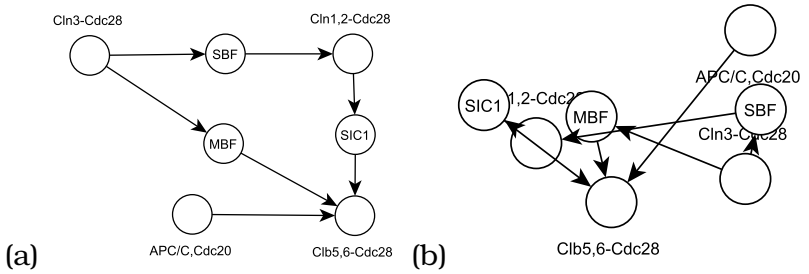


Abbildung 2.1: Zwei unterschiedliche graphische Darstellungen eines Teils des Zellzyklus der Backhefe. (a) Geeignete Visualisierung (adaptiert aus [HMW+07]) und (b) ungeeignete Visualisierung, da die Struktur des Graphen nicht erkennbar ist (siehe auch Abschnitt 2.1.2).

Beide Arten der Darstellung enthalten die gleichen Informationen, jedoch ist die Darstellung in Tabellenform im Vergleich zu Abbildung 2.1(a) unübersichtlicher und schwerer zu interpretieren. Auch eine ungeeignete Visualisierung wie in Abbildung 2.1(b) erschwert die Interpretation der durch den Graphen repräsentierten Informationen, da hier die Knoten und Kanten so angeordnet sind, dass die Struktur des Graphen (und damit die dargestellten Informationen) nicht erkennbar ist. Die Positionierung der Knoten und Kanten in der Zeichnung eines Graphen ist also ein entscheidender Faktor zur Bestimmung seiner Lesbarkeit.

Der Forschungsbereich des Graphzeichnens ist ein Teilgebiet der Informationsvisualisierung, welche nach [GEC98] folgendermaßen beschrieben wird: *„Information visualization is the process of transforming data, information and knowledge into visual form making use of humans' natural visual capabilities."*

Graphzeichnen beschäftigt sich mit dem Problem, die Knoten und Kanten eines Graphen, welche in den meisten Fällen abstrakte Daten repräsentieren, automatisch so zu zeichnen, dass die dargestellten Informationen leicht verständlich sind. Parameter wie Form oder Farbe der Objekte werden hierbei, anders als bei der allgemeinen Informationsvisualisierung, nicht betrachtet. Im weiteren Verlauf des Kapitels werden einige grundlegende Definitionen aus dem Gebiet des Graphzeichnens genannt und Methoden zur automatischen Zeichnung von Graphen vorgestellt. Um eine detailiertere Übersicht zum Thema Graphzeichnen zu erhalten, wird auf [BETT99] oder [KW01] verwiesen.

2.1.1 Definitionen

Ein *Graph* ist ein Paar $G(V, E)$, hierbei ist V eine endliche Menge an Knoten und $E \subseteq V \times V$ eine Relation auf V, die die Menge der Kanten zwischen den Knoten beschreibt.

Ein Graph wird als *ungerichteter Graph* (siehe Abbildung 2.2(a)) bezeichnet, wenn durch seine Kantenrelation jeder Kante $e \in E$ ein ungeordnetes Paar (u, v) zugeordnet wird, so dass gilt: $(u, v) = (v, u)$. Wird jeder Kante e ein geordnetes Paar (u, v) zugeordnet, wird er als *gerichteter Graph* (siehe Abbildung 2.2(b)) bezeichnet. Die Richtung einer Kante wird in Abbildungen üblicherweise durch einen Pfeil veran-

schaulicht. Enthält ein Graph sowohl gerichtete als auch ungerichtete Kanten, so wird er als *Mixed-Graph* (siehe Abbildung 2.2(c)) bezeichnet.

Ein *gerichteter azyklischer Graph* (*DAG*) ist ein gerichteter Graph, der keine Kantenfolge $p = (u_0,v_0)\dots(u_{m-1},v_{m-1})$ mit $m \in \mathbb{N}_0$ und $v_{i-1} = u_i$ für alle $i \in 1,\dots,m-1$ enthält, bei der Start- und Endknoten identisch sind, d.h. $v_{m-1} \neq u_0$ (siehe Abbildung 2.2(d)).

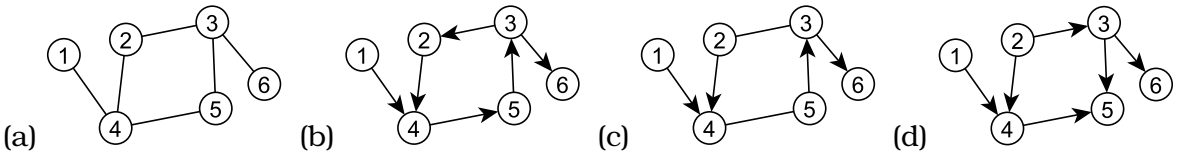


Abbildung 2.2: Verschiedene Arten von Graphen: (a) Ungerichteter Graph, (b) gerichteter Graph, (c) Mixed-Graph und (d) DAG.

Ein *Compound-Graph* $C(V,E,F)$ [SM91] ist definiert durch einen Graphen $G(V,E)$ und einen gewurzelten Baum $T(V,F)$, welcher als *Inklusionsbaum* (siehe auch Abbildung 2.3(b)) bezeichnet wird. Die Menge der Kanten E wird als *Adjazenzkanten*, die Menge an Kanten F als *Inklusionskanten* bezeichnet. Die Inklusionskanten bestimmen eine Inklusionsbeziehung zwischen den Knoten.

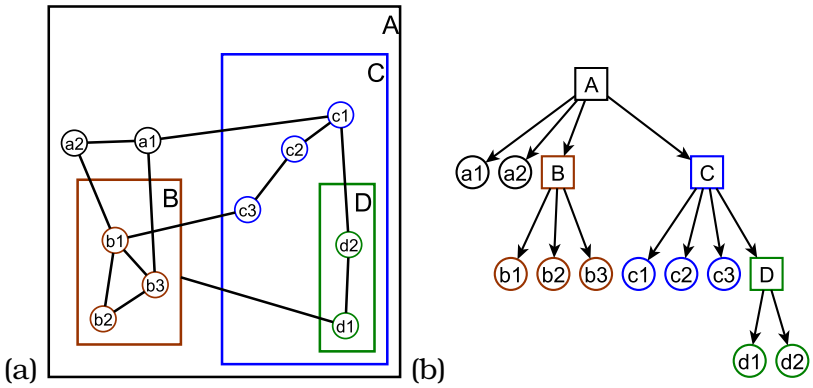


Abbildung 2.3: (a) Ein Compound-Graph und (b) der dazugehörige Inklusionsbaum.

Als *Subgraph*, *komplexer Knoten* oder auch *Elternknoten* wird ein Knoten bezeichnet, der im Inklusionbaum einen Nachfolger besitzt. Nach dieser Definition stellen die Knoten A, B, C und D aus dem Beispielgraphen in Abbildung 2.3 jeweils einen Subgraphen dar. Ein Subgraph v beinhaltet alle Knoten, die direkte Nachfolger von v im Inklusionsbaum sind.

Die direkten Nachfolger eines Knotens im Inklusionsbaum werden als *Kindknoten* des Knotens bezeichnet. Knoten, die im Inklusionsbaum keine Nachfolger besitzen, werden als *einfache Knoten* oder auch als *Blattknoten* bezeichnet.

Ein *Cluster-Graph* [EF96] ist ein spezieller Compound-Graph, bei dem Adjazenzkanten aus E nur zwischen Blattknoten des Inklusionsbaums existieren.

Ein Graph wird als *flacher Graph* oder als *einfacher Graph* bezeichnet, wenn alle seine Knoten einfache Knoten sind.

Die Tiefe des Inklusionsbaumes ohne Blattknoten wird als *Verschachtelungstiefe* des Compound-Graphen bezeichnet. Bei dem in Abbildung 2.3 dargestellten Compound-Graphen, beträgt die Verschachtelungstiefe beispielsweise 3.

Der Endknoten einer Kante, welche den Elternknoten des Endknotens verlässt, wird als *Terminalknoten* (schwarze Knoten in Abbildung 2.4(a)) bezeichnet, wenn dieser sich in der gleichen oder einer tieferen Hierarchieebene wie der andere Endknoten der Kante befindet. Terminalknoten sind dementsprechend die Verbindungsstellen eines Subgraphen nach außen und werden immer am inneren Rand des Elternknotens positioniert.

Als *Port* wird im Graphzeichnen der Andockpunkt einer Kante an einen Knoten bezeichnet. Ein Port kann entweder Start- oder Endpunkt einer Kante sein. Im Unterschied zu Terminalknoten werden Ports jedoch nicht als echte Knoten angesehen.

Ein *Terminal-Cluster-Graph* ist ein spezieller Cluster-Graph, bei dem Terminalknoten auftreten. Durch die Verwendung eines Terminal-Cluster-Graphen wird es möglich, jeden Subgraphen ohne Beachtung der Beziehung zu Knoten außerhalb des Subgraphen zu erstellen und zu bearbeiten (siehe Abschnitt 2.2.2.3). Jeder Compound-Graph kann in einen Terminal-Cluster-Graphen umgewandelt werden, indem jeweils an dem Schnittpunkt einer Kante mit der Begrenzung eines Subgraphen ein Terminalknoten eingefügt wird (vergleiche Abbildungen 2.3(a) und 2.4(a)).

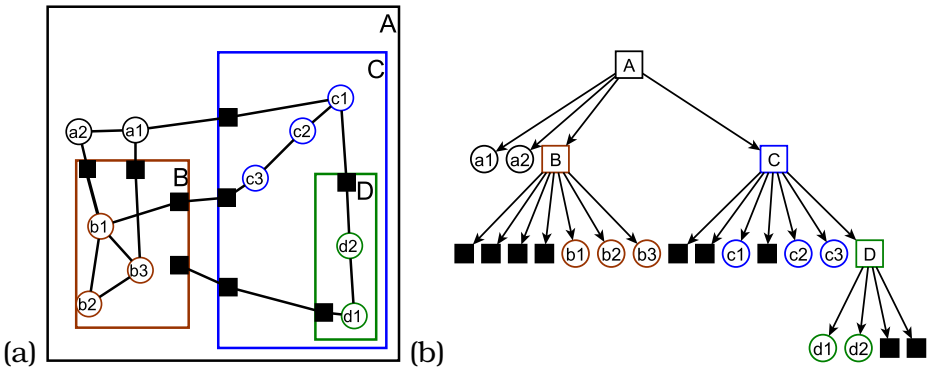


Abbildung 2.4: (a) Der Terminal-Cluster-Graph des Compound-Graphen aus Abbildung 2.3 und (b) der dazugehörige Inklusionsbaum.

Als *Layout* oder Zeichnung wird die graphische Repräsentation eines Graphen G, welche durch die Abbildung $G \rightarrow (\mathbb{R}^2$ oder $\mathbb{R}^3)$ erzeugt wird, bezeichnet. Durch diese werden den Knoten und Kanten Koordinaten zugeordnet. In dieser Arbeit wird sich beim Zeichnen auf den zweidimensionalen Raum ($\mathbb{R}^2$) beschränkt.

Durch die Betrachtung der graphischen Repräsentation eines Graphen entwickelt der Anwender ein mentales Bild, die sogenannte *Mental Map* [MELS95], von dieser graphischen Repräsentation. Die Mental Map hilft dem Anwender, sich wichtige

Aspekte des Layouts eines Graphen, wie z.B. die Positionierung bestimmter Knoten zueinander, einzuprägen und diese wiederzuerkennen. Werden Änderungen am Graphen, z.B. durch Einfügen von Knoten, vorgenommen und das Layout des Graphen anschließend neu bestimmt, kann es sein, dass das neue Layout des Graphen vom alten Layout erheblich abweicht. Dadurch wird die Mental Map zerstört und der Anwender findet sich nicht mehr in der neuen graphischen Repräsentation zurecht. Aus diesem Grund ist die Wahrung der ursprünglichen graphischen Repräsentation und damit die Wahrung der Mental Map nach einer Neuberechnung des Layouts eines Graphen wichtig.

2.1.2 Ästhetikkriterien

Das Ziel jedes Algorithmus zum Zeichnen von Graphen ist immer ein „schönes“ und für den Anwender leicht verständliches Layout eines Graphen zu erzeugen[2]. Die Frage, was die Schönheit, die Qualität oder die Lesbarkeit des Layouts eines Graphen ausmacht, kann jedoch nicht eindeutig beantwortet werden. Aus diesem Grund wurden Ästhetikkriterien [BETT99, KW01] festgelegt, die als Maßstab zur Beurteilung der Qualität eines Layouts herangezogen werden können. Im weiteren Verlauf werden einige Ästhetikkriterien erläutert, die allgemein als vorteilhaft angesehen werden, um die Lesbarkeit des Layouts eines Graphen zu erhöhen:

- *Minimierung von Kantenkreuzungen:* Überschneidungen zwischen Kanten erschweren es, dem Verlauf der Kanten zu folgen und machen ein Layout unübersichtlich. Daher ist es notwendig, die Anzahl an Kantenkreuzungen zu minimieren.
- *Minimierung von Kantenknicken:* Je mehr Knicke eine Kante enthält, desto schwieriger ist es, ihrem Verlauf zu folgen. Daher sollte die Zahl der Kantenknicke klein gehalten werden.
- *Abstand der Knoten:* Knotenüberlappungen behindern die Lesbarkeit und sind somit zu vermeiden. Andererseits ist jedoch darauf zu achten, dass Knoten nicht zu weit auseinander platziert werden, da dies zu langen Kanten und hohem Platzbedarf führt.
- *Abstand von Knoten und Kanten:* Um die Lesbarkeit eines Layouts nicht zu verringern, sollten Kanten nicht zu nah an Knoten vorbeilaufen und diese nach Möglichkeit nicht schneiden.
- *Gleichlange, kurze Kanten:* Die Kanten sollten nicht unnötig lang sein, da Layouts mit kurzen Kanten in der Regel einfacher zu lesen sind. Ebenso ist es vorteilhaft, wenn die Kantenlängen sich innerhalb eines Layouts nicht zu stark unterscheiden.
- *Minimierung der Fläche:* Die Fläche, die durch das Layout eingenommen wird, sollte so klein wie möglich sein, um Platz auf dem Bildschirm zu sparen.

2 Ausnahme bilden hier spezielle Anwendungsgebiete, wie z.B. der Entwurf von Schaltkreisen (*VLSI-Design*), in denen nach anderen Kriterien optimiert wird.

- *Minimierung des Seitenverhältnisses:* Das Seitenverhältnis des Layouts, welches durch das Verhältnis der Länge der längsten Seite zur Länge der kürzesten Seite bestimmt wird, sollte möglichst klein sein, um die Fläche auf dem Bilschirm optimal auszunutzen.
- *Maximierung des Winkels zwischen zwei Kanten:* Der Winkel zwischen zwei in einem Knoten beginnenden oder endenden Kanten sollte möglichst groß sein, um eine Unterscheidung der Kanten zu vereinfachen.
- *Symmetrie:* Je symmetrischer das Layout eines Graphen ist, desto einfacher ist es die dargestellten Informationen zu erfassen. Daher ist es sinnvoll, die Symmetrie des Layouts zu optimieren.
- *Richtung:* Existieren Richtungsinformationen, wie hierarchische Beziehungen zwischen dargestellten Objekten, so sollten diese auch im Layout des Graphen hervorgehoben werden.

Um weitere Anforderungen an des Layout umzusetzen, lassen sich eine Reihe von Einschränkungen (engl. *Constraints*) definieren, welche sich jedoch im Gegensatz zu Ästhetikkriterien nicht auf den gesamten Graphen sondern immer nur auf eine Teilmenge von Knoten beziehen. Beispiele für Constraints sind:

- *Innen oder Außen:* Platzierung eines Knotens in der Mitte oder am Rand der Zeichnung des Graphen.
- *Cluster:* Platzierung einer Auswahl von Knoten nah beieinander, um ihre Zusammengehörigkeit hervorzuheben.
- *Subgraphen:* Platzierung der Kindknoten eines Subgraphen innerhalb der Grenzen des Subgraphen.
- *Links-rechts oder auch oberhalb-unterhalb:* Platzierung von zwei Knoten in einer bestimmten Lage zueinander.

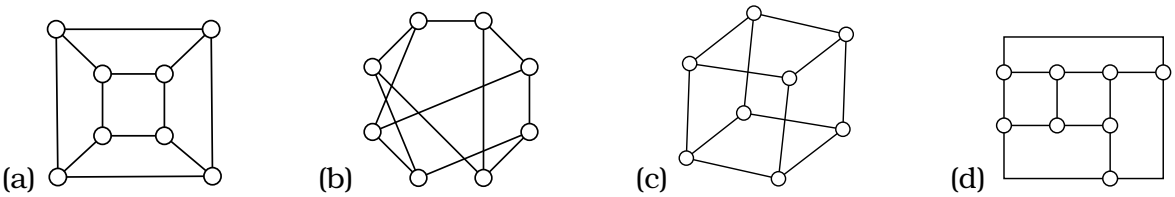


Abbildung 2.5: Optimierung verschiedener Qualitätsparameter und die daraus resultierenden Layouts des Würfel-Graphen: (a) Minimierung von Kantenkreuzungen und Kantenknicken, (b) Platzierung aller Knoten am Rand, (c) Einhaltung der Anforderung gleicher Kantenlängen und (d) Verwendung orthogonaler Kanten.

Das Problem bei der Erzeugung eines Layouts ist jedoch, dass es nicht möglich ist, alle Qualitätsparameter gleichzeitig zu optimieren, da diese sich gegenseitig behindern können. Abbildung 2.5 zeigt vier verschiedene Layouts des Würfel-Graphen, welche auf Basis unterschiedlicher Qualitätskriterien erzeugt wurden. Welches der

dargestellten Layouts das intuitivere oder qualitativ bessere ist, lässt sich nicht eindeutig feststellen: Abbildungen 2.5(a) und 2.5(d) stellen beispielsweise ein planares Layout des Würfel-Graphen dar. In Abbildung 2.5(a) wurde jedoch zusätzlich die Zahl der Kantenknicke minimiert, wogegen in Abbildung 2.5(d) die Konvention der orthogonalen Kanten eingehalten wurde und dafür Kantenknicke in Kauf genommen werden mussten. In Abbildung 2.5(b) wurde das Außen-Constraint optimiert und alle Knoten am Rand des Layouts platziert. Dadurch entstehen jedoch Kantenkreuzungen, die die Lesbarkeit des Layouts erschweren. Abbildung 2.5(c) enhält zwar auch Kantenkreuzungen, jedoch ist die Struktur des Graphen gut erkennbar, da sie wie die Projektion eines dreidimensionalen Würfels aussieht.

Die Frage, welches der genannten Ästhetikkriterien wichtiger ist, ist zum einen abhängig vom Anwendungsgebiet in dem der Graph benutzt wird (und den dort geltenden Konventionen bei der Darstellung von Graphen) und zum anderen von der individuellen Wahrnehmung jedes einzelnen Anwenders. Bei einer Untersuchung von Darstellungen von *UML*-Diagrammen wurde beispielsweise herausgefunden, dass es am wichtigsten ist, Kantenkreuzungen zu vermeiden und Kanten immer orthogonal (entweder horizontal oder vertikal) zu zeichnen [PAC02]. Allgemeinere empirische Studien haben ergeben, dass wenige Kantenkreuzungen das wichtigste Merkmal zum Verständis von Graphenzeichnungen sind, gefolgt von wenigen Kantenknicken, kurzen Kantenverläufen, der Vermeidung von spitzen Winkeln zwischen Kanten und der Darstellung von Symmetrien [PCJ96].

2.1.3 Zeichenverfahren

Wie in Abschnitt 2.1.2 erläutert, gibt es das „perfekte“ Layout für einen Graphen nicht. Aus diesen Grund existieren verschiedene Methoden um ein Layout nach bestimmten Anforderungen zu erzeugen. Im Folgenden wird eine Auswahl der verbreitetsten Zeichenverfahren vorgestellt.

2.1.3.1 Kräftebasierte Verfahren

Kräftebasierte Verfahren [Ead84, KK89, DH96] eignen sich zum Zeichnen allgemeiner (ungerichteter) Graphen. Diese benutzen zur Bestimmung des Layouts eine Analogie zur Physik, in dem der Graph als System aus Objekten (Knoten) und denen zwischen diesen Objekten wirkenden Kräften betrachtet wird. Kräftebasierte Verfahren suchen nach einer Konfiguration der Objekte, so dass die Summe der auf jedes Objekt wirkenden Kräfte unter einem Grenzwert liegt. Diese Konfiguration stellt dann die Zeichnung (oder das Layout) des Graphen dar. Abbildungen 2.7(a) und 2.8(a) zeigen typische Ergebnisse kräftebasierter Zeichenverfahren.

Ein wichtiger Vertreter kräftebasierter Verfahren ist der *Spring-Embedder* von Eades [Ead84]. Dieser wird, wie in Abschnitt 2.3 erläutert, als Grundlage für den in dieser Arbeit vorgestellten Algorithmus ausgewählt. Aus diesem Grund wird die Funktionsweise des Spring-Embedders im Folgenden näher beschrieben. Beim Spring-Embedder werden die Knoten des Graphen als sich abstoßende, elektrisch geladene Knoten und die Kanten als die Knoten zusammenhaltende *Federn* (engl. Springs) modelliert, welche je nach Abstand der Knoten eine anziehende bzw. abstoßende

Kraft erzeugen. Die Anziehungskraft FS zwischen zwei adjazenten Knoten wird mit

$$FS = c_1 * \log(\frac{l}{c_2}) \tag{2.1}$$

und die Abstossungskraft FR zwischen zwei Knoten mit

$$FR = \frac{c_3}{\sqrt{d}} \tag{2.2}$$

berechnet. c_1, c_2 und c_3 bezeichnen hierbei Konstanten, l bezeichnet die Federlänge und d ist der Abstand zwischen zwei Knoten. Die Positionen der Knoten werden dann basierend auf ihren Ausgangspositionen iterativ in Richtung der wirkenden Kräfte verändert, bis sich ein Kräftegleichgewicht einstellt (siehe auch Algorithmus 2.1).

Algorithmus 2.1 Spring-Embedder nach Eades

Input: Graph G(V, E)
Output: A layout of G
1: place nodes of G in random locations
2: **while** $k < maxIterations$ **do**
3: calculate the force on each node
4: move the node in direction of the calculated force
5: **end while**

Abbildung 2.6 zeigt schematisch die iterative Entwicklung des Layouts eines Graphen durch das Spring-Embedder-Verfahren.

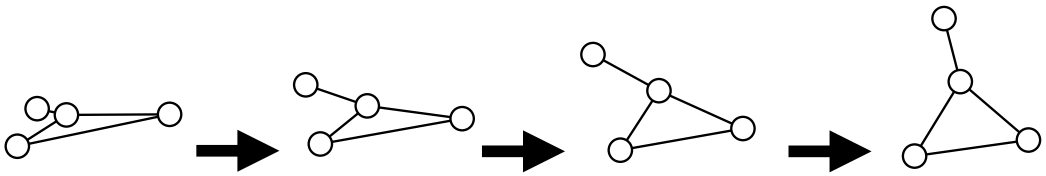


Abbildung 2.6: Entwicklung des Layouts beim einfachen Spring-Embedder.

Die Laufzeit des Spring-Embedder-Algorithmus ist im Allgemeinen $\mathcal{O}(k*n^2)$, wobei k die Anzahl der benötigten Iterationen und n die Anzahl an Knoten bezeichnet. Aufgrund dieses hohen Berechnungsaufwandes, welcher durch die Tatsache bedingt ist, dass zwischen jedem Knotenpaar Abstoßungskräfte berechnet werden, ist der ursprüngliche Spring-Embedder nur für kleinere Graphen (bis 50 Knoten) geeignet. Es existieren mitterweile jedoch Techniken, die die Verwendung des Spring-Embedders auch für größere Graphen (mit mehr als 10000 Knoten) ermöglichen [Wal00]. Weiterhin existieren für den Spring-Embedder verschiedene Erweiterungen zur Verbesserung der Qualität [FR91, FLM94, LEN05] und zur Umsetzung anwendungsbezogener Constraints [RMS97, WM95]. So können, beispielsweise durch das Konzept der magnetischen Kräfte [SM94], gerichtete Kanten von gerichteten oder Mixed-Graphen in eine zuvor definierte Richtung ausgerichtet werden (siehe Abschnitt 3.2.1).

Ein Vorteil kräftebasierter Verfahren ist die konzeptionelle Einfachheit. Die grundlegenden Algorithmen sind dadurch mit relativ geringem Implementierungsaufwand umsetzbar und erweiterbar. Eine weitere positive Eigenschaft sind die symmetrischen, organisch wirkenden Layouts. Da kräftebasierte Verfahren meist iterativ ablaufen, eignen sie sich zur Wahrung der Mental Map.

Nachteile kräftebasierter Verfahren sind die schlechte Nachprüfbarkeit der Ergebnisse, die Abhängigkeit vom Ausgangslayout und die fehlende Minimierung von Kantenkreuzungen. Ein weiterer Nachteil ist, dass die Abstimmung der einzelnen Kräfte aufgrund der vielen Parameter, welche auch durch den Anwender eingestellt werden können (siehe Abschnitt 4.3.2), relativ schwierig ist.

2.1.3.2 Schichtenbasierte Verfahren

Schichtenbasierte Verfahren ordnen die Knoten schichtweise an. Abbildungen 2.7(b) und 2.8(b) zeigen typische Ergebnisse schichtenbasierter Zeichenverfahren. Das am häufigsten verwendete Verfahren für DAGs basiert auf dem in [STT81] vorgestellten Verfahren. Dieses läuft in drei Schritten ab:

- Schritt 1: Zuordnung jedes Knotens zu einer Schicht, so dass alle Kanten in eine bestimmte Richtung zeigen (z.B. von oben nach unten).
- Schritt 2: Ordnung der Knoten innerhalb der Schichten, so dass die Zahl der Kantenkreuzungen minimiert wird.
- Schritt 3: Zuordnung von konkreten x-Koordinaten zu den Knoten, auf Basis der in Schritt 2 berechneten Knotenordnungen.

Der Vorteil schichtenbasierter Verfahren ist die sehr gute Darstellung der Kantenrichtungen bei DAGs. Ein weiterer Vorteil ist die Vermeidung von Kantenkreuzungen und Knotenüberlappungen.

Der Nachteil schichtenbasierter Verfahren ist die Beschränkung auf die Darstellung von DAGs. Es ist zwar möglich, auch ungerichtete oder zyklische Graphen durch Hinzufügen oder Ändern der Kantenrichtungen in einen DAG umzuwandeln und diesen dann mit einem schichtenbasierten Verfahren zu zeichnen, jedoch führt in diesem Fall die Optimierung des primären Ziels (Darstellung der Richtung) schichtenbasierter Verfahren oft zu qualitativ unbefriedigenden Ergebnissen. Ein weiterer Nachteil ist das oft ungünstige Seitenverhältnis.

2.1.3.3 Orthogonale Verfahren

Bei orthogonalen Verfahren werden die Knoten auf virtuelle Gitterpunkte platziert und die Kanten auf Kantenfolgen abgebildet, deren Teilstücke entweder horizontal oder vertikal verlaufen (siehe auch Abbildungen 2.7(c) und 2.8(c)). Ein häufig angewendetes orthogonales Verfahren ist der *Topology-Shape-Metrics*-Ansatz [Tam87]. Dieser besteht aus drei Schritten:

- Schritt 1 (*Topology*): Veränderung der Topologie der Elemente des Graphen, so dass dieser planar (d.h. auf der Ebene überkreuzungsfrei zeichenbar) ist.

Falls der Graph nicht planar ist, werden Kantenkreuzungen durch temporäre Knoten ersetzt.

- Schritt 2 (*Shape*): Bestimmung der Form des Layouts, indem Kantenknicke und Winkel zwischen den Kanten festgelegt werden.
- Schritt 3 (*Metric*): Veränderung der Länge der Kanten, so dass das resultierende Layout möglichst kompakt ist.

Vorteil orthogonaler Verfahren ist die Optimierung vieler Ästhetikkriterien (Kantenknicke, Kantenkreuzungen) und die ausschließliche Verwendung von horizontalen oder vertikalen Kanten, was in bestimmten Anwendungsgebieten wie dem Schaltkreisentwurf besonders wichtig ist.

Der Nachteil orthogonaler Verfahren ist der im Vergleich zu anderen Zeichenverfahren um ein vielfaches größere Implementierungsaufwand. Des Weiteren werden Kantenrichtungen meist nicht beachtet. Da die Kanten nur horizontal oder vertikal verlaufen, entstehen viele Kantenknicke, wodurch es besonders bei Graphen mit zahlreichen Kanten schwierig wird, dem Kantenverlauf zu folgen.

2.1.3.4 Zirkuläre Verfahren

Bei zirkulären Verfahren werden die Knoten des Graphen kreisförmig angeordnet. Bei den meisten Verfahren wird der Graph dafür zunächst strukturell untersucht. Dann werden die Knoten so auf einem Kreis positioniert, dass möglichst wenig Kantenkreuzungen entstehen. Abbildungen 2.7(d) und 2.8(d) zeigen typische Ergebnisse zirkulärer Zeichenverfahren.

Vorteil zirkulärer Verfahren ist, dass sie relativ einfach zu implementieren sind und auch größere Graphen mit einer geringen Laufzeit zeichnen können. Sie eignen sich besonders gut zur Darstellung von Graphen mit zyklischen Strukturen.

Nachteilig ist, dass sie meist auf bestimmte Graphen, welche zyklische Strukturen beinhalten, optimiert sind. Bei allgemeinen Graphen, wird die Struktur schlecht „herausgearbeitet“ und es entstehen viele Kantenkreuzungen. Ein weiterer Nachteil zirkulärer Verfahren ist der meist hohe Platzbedarf der erstellten Layouts.

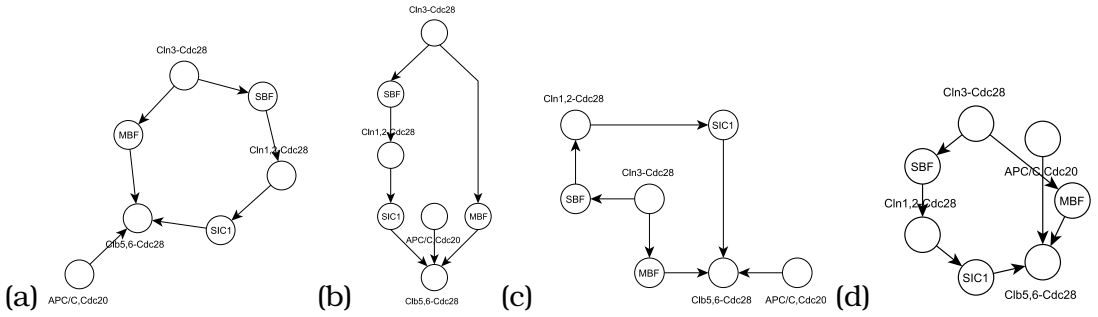


Abbildung 2.7: Ergebnisse verschiedener Zeichenverfahren am Beispiel des Graphen aus Abbildung 2.1: (a) Kräftebasiertes Verfahren, (b) schichtenbasiertes Verfahren, (c) orthogonales Verfahren und (d) zirkuläres Verfahren.

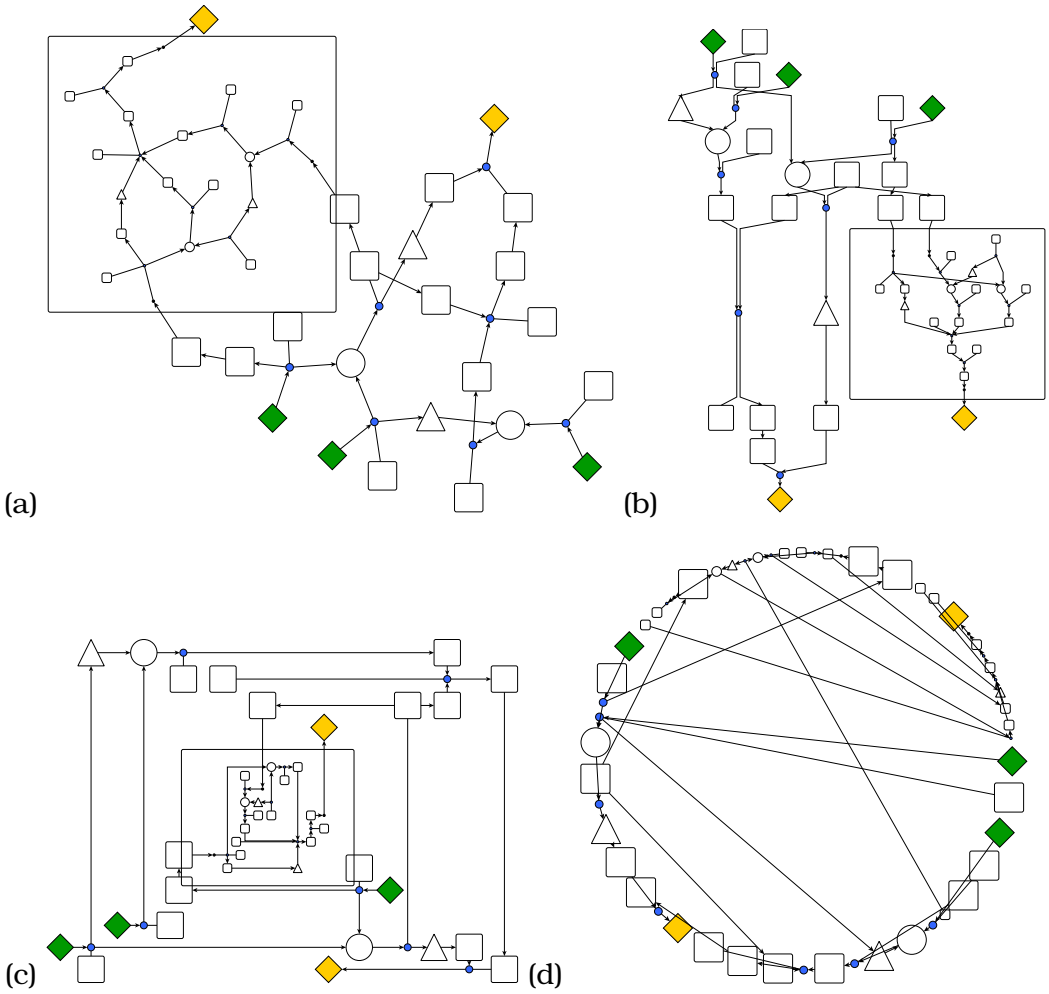


Abbildung 2.8: Ergebnisse verschiedener Zeichenverfahren am Beispiel des Testmodells aus Abbildung 4.9: (a) Kräftebasiertes Verfahren, (b) schichtenbasiertes Verfahren, (c) orthogonales Verfahren und (d) zirkuläres Verfahren.

2.2 Visualisierung biologischer Netzwerke

2.2.1 Grundlagen

Um Biologen das Verständnis und die Analyse von biologischen Netzwerken zu erleichtern, werden diese traditionell als Graphen modelliert und visualisiert. Das Anwendungsfeld der dargestellten Netzwerke reicht hierbei von einfachen Netzwerken, wie Beschreibungen einzelner Stoffwechselreaktionen, bis hin zu sehr großen Netzwerken, welche ganze zelluläre Systeme beschreiben.

Früher waren solche Visualisierungen ausschließlich in biologischen Lehrbüchern zu finden. Sie wurden manuell erzeugt und stellten typischerweise einfache, kleinere Netzwerke dar. Mit dem Anstieg an Wissen über Aufbau und Ablauf biologischer Prozesse stieg auch die Komplexität in der Darstellung kontinuierlich an. Als Beispiel hierfür ist in Abbildung 2.9 ein Ausschnitt aus der Online-Version [Exp08] des Posters *Biochemical Pathways* von Michal [Mic98] abgebildet, welches die wesentlichen biochemischen Stoffwechselwege der Zelle darstellt.

Wie auch die Abbildungen in Lehrbüchern, wurde diese Darstellung manuell und lange vor der eigentlichen Verwendung erzeugt. Solche Darstellungen werden daher auch als *statische Darstellungen* bezeichnet [BGHS98]. Statische Darstellungen haben den Nachteil, dass ihre Erzeugung (zeit-)aufwändig ist und dass sie schwer zu aktualisieren sind (bei Änderung des Modells müssen diese komplett neu erzeugt werden, da sich in den meisten Fällen auch das Layout ändert). Ein weiterer Nachteil statischer Darstellungen ist, dass diese unflexibel sind und keine Filterung oder alternative Darstellungen der Daten ermöglichen. Beispielsweise kommen in internetbasierten Systemen wie der KEGG-Datenbank [Keg08] statische Darstellungen in Form von Bilddateien zum Einsatz. Problematisch ist hierbei, dass für jede mögliche Datenbankabfrage eine vorberechnete Graphik vorhanden sein muss und bei Änderungen im Datenbestand Graphiken entsprechend manuell aktualisiert werden müssen.

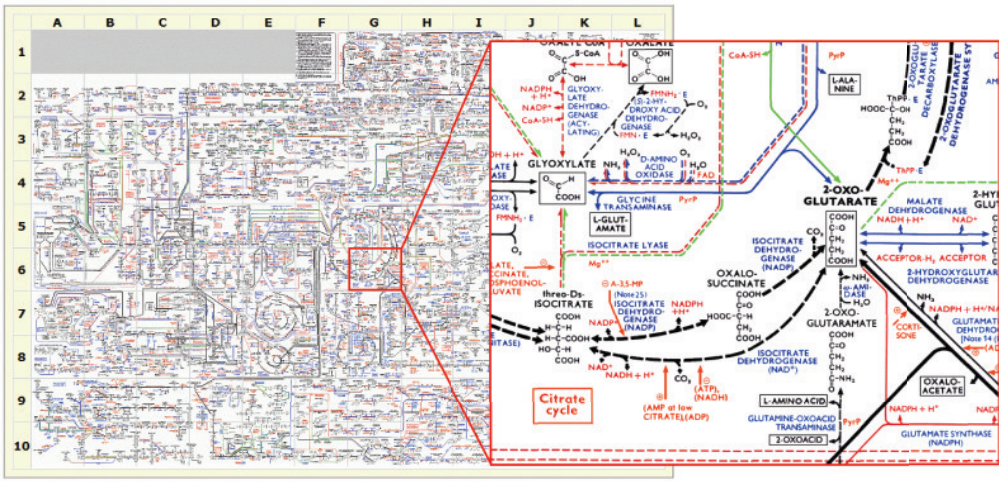


Abbildung 2.9: Ausschnitt aus der Online-Version [Exp08] des Posters *Biochemical Pathways* von Michal [Mic98]

Zur Vermeidung der Nachteile statischer Darstellungen kommen vermehrt *dynamische Darstellungen* [BGHS98] zum Einsatz. Hierbei werden automatische Methoden zur Erzeugung der Darstellungen von biologischen Netzwerken eingesetzt, um diese, genau zu dem Zeitpunkt zu dem sie benötigt werden, zu erzeugen. Hierdurch wird es möglich, dem sich ständig ändernden und immer größer werdenden Wissenstand in Molekularbiologie und Biochemie gerecht zu werden.

Tabelle A.2 im Anhang enthält eine Übersicht von Programmen, die in der Lage sind automatisch Darstellungen biologischer Netzwerke zu erzeugen. Viele dieser Programme setzen Standard-Zeichenverfahren zum Erstellen der Darstellung biologischer Netzwerke ein. Diese Verfahren dienen hauptsächlich zur Visualisierung der Topologie des Netzwerks. Dadurch können z.B. besonders stark vernetzte Elemente (sogenannte *Hubs*), welche häufig wichtige Elemente in biologischen Netzwerken darstellen, hervorgehoben werden. Standard-Zeichenverfahren folgen jedoch nicht den Konventionen zum Zeichnen biologischer Netzwerke, welche traditionell in Lehrbüchern verwendet werden. Deshalb ist es für die effiziente Visualisierung von biologischen Netzwerken wichtig, ihre besonderen Eigenschaften zu

beachten. Wichtige Ziele der Visualisierung von biologischen Netzwerken sind beispielsweise das Verständnis der Regulation zellulärer Prozesse durch Signale oder neue Erkenntnisse über den Verlauf von Signalen innerhalb der Netzwerke zu erhalten [BBS08]. Einige der in Tabelle A.2 genannten Programme erweitern dazu Standard-Zeichenverfahren (siehe auch Abschnitt 3.2). Im Folgenden werden Ziele und Anforderungen erläutert, die beim Zeichnen biologischer Netzwerke zu beachten sind.

2.2.2 Anforderungen an das Layout biologischer Netzwerke

Die im Abschnitt 2.1.2 genannten allgemeinen Ästhetikkriterien werden auch als *syntaktische* Ästhetikkriterien bezeichnet und dienen zur Bewertung der Lesbarkeit des Layouts eines Graphen unabhängig von seinem Anwendungsgebiet. Im Folgenden werden anwendungsspezifische Anforderungen, sogenannte *semantische* Ästhetikkriterien, beschrieben, die beim Zeichnen von biologischen Netzwerken beachtet werden sollten [BBS08].

2.2.2.1 Darstellung der Richtung von Reaktionspfaden

Viele biologische Prozesse, wie z.B. Signaltransduktionen, Genregulationen oder Stoffwechselvorgänge, laufen in bestimmten Richtungen ab. Existiert eine Hauptreaktionsrichtung, so sollte diese klar erkennbar sein (in vielen Fällen von oben nach unten), um die zeitliche Reihenfolge von Prozessen und Reaktionen deutlich zu machen. In Abbildung 2.10 sind die Reaktionspfade des EGF-Netzwerks schematisch dargestellt. Durch die dargestellte Richtung der Reaktionspfade ist der Verlauf der Signale beginnend am Rezeptor in der extrazellulären Schicht der Zelle bis hin zum Zellkern gut zu erkennen.

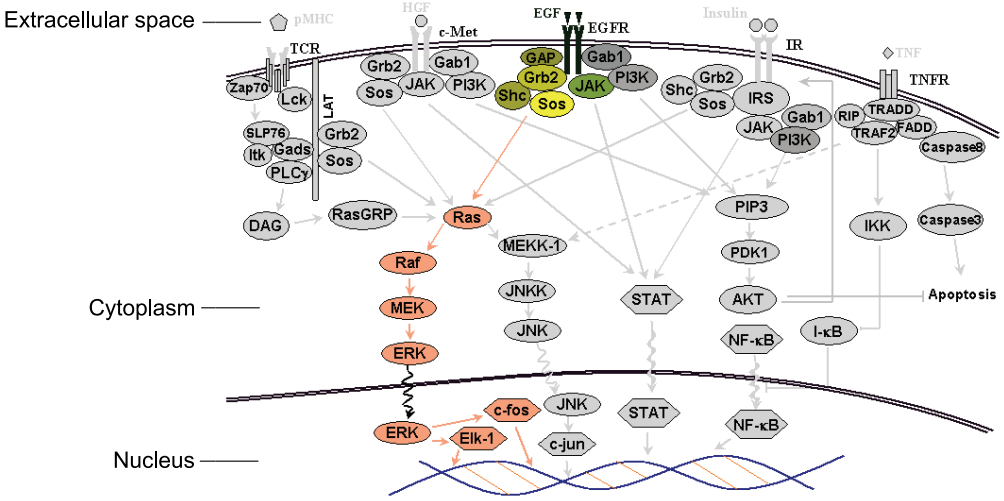


Abbildung 2.10: Manuell erzeugte Darstellung von Signalverläufen innerhalb einer Zelle. Das EGF-Netzwerk ist farblich hervorgehoben (Graphik mit freundlicher Genehmigung von Julio Saez-Rodriguez).

2.2.2.2 Darstellung der subzellulären Lokalisation

Die einzelnen Komponenten biologischer Netzwerke befinden sich in verschiedenen zellulären Kompartimenten[3], wie Zellmembran, Zellkern und Mitochondrium. Diese Zuordnung sollte (falls bekannt) in der Visualisierung erkenntlich sein, da sie hilft, das Verständnis über Aufbau und Funktionsweise von zellulären Systemen zu erleichtern. In Abbildung 2.10 ist die subzelluläre Lokalisation durch eine schichtweise Darstellung des Aufbaus einer Zelle abgebildet. Ein weiteres Beispiel für die Darstellung der subzellulären Lokalisation ist Abbildung 2.11. Diese stellt die beim Alterungsprozess ablaufenden Interaktionen zwischen Komponenten aus verschiedenen Kompartimenten (subzellulär, zellulär und extrazellulär) dar.

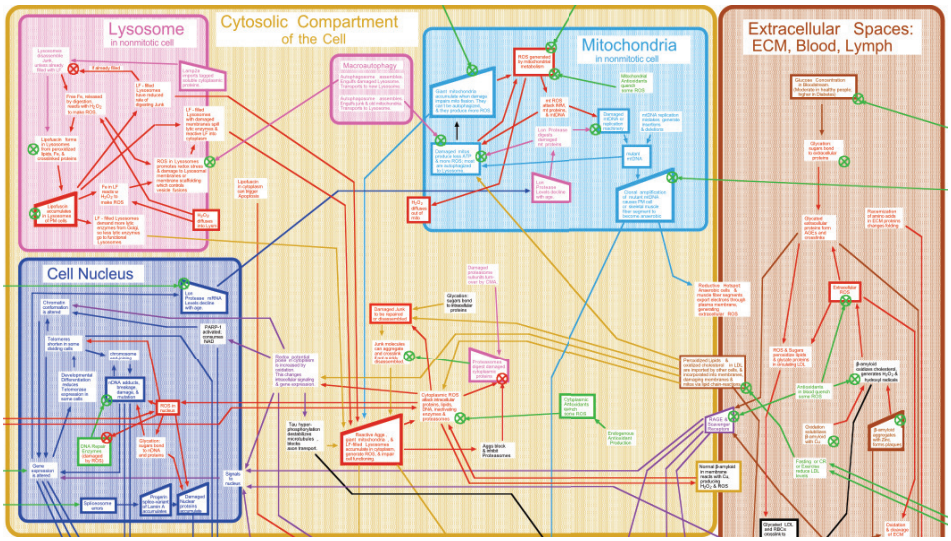


Abbildung 2.11: Auschnitt aus einer manuell erstellten Darstellung der beim Alterungsprozess ablaufenden Interaktionen zwischen Komponenten aus verschiedenen zellulären Kompartimenten [Fur08].

2.2.2.3 Darstellung hierarchisch strukturierter Modelle

Die Systembiologie hat die einheitliche Erforschung der Vorgänge innerhalb zelluläre Systeme zum Ziel [Rei02]. Aufgrund der Größe und Komplexität zellulärer Systeme sind flache Graphen zur Modellierung dieser Systeme jedoch nicht geeignet, da hierdurch sowohl die Analyse als auch die effektive Visualisierung des kompletten Systems kaum möglich sind. Aus diesem Grund ist es sinnvoll, zelluläre Systeme als hierarchisch strukturierte Modelle bzw. als Compound-Graphen zu modellieren [GKN+03, BUS04, HMW+07].

Neben der erläuterten biologisch motivierten Zusammengehörigkeit von Komponenten durch subzelluläre Lokalisation oder der Zugehörigkeit zu verschiedenen Organisationsebenen (Makro- und Mikrolevel), werden zur Strukturierung die Komponenten biologischer Netzwerke auch durch ihre funktionale Zusammengehörigkeit

3 Zellkompartimente sind Reaktionsräume, in denen biochemische Reaktionen ablaufen. Zwischen den Zellkompartimenten findet aktiver und passiver Stoffaustausch durch Membranen statt [BGHS98].

zu Komplexen oder Modulen zusammengefasst [HHLM99, SRKC+04]. Beispiele für Module in der Systembiologie sind funktionale Einheiten in Stoffwechselvorgängen (z.B. Glykolyse), Verarbeitungseinheiten (z.B. Signalkaskaden) oder auch elementare biologische Prozesse (z.B. Reaktionen). Im Allgemeinen repräsentieren Module zusammenhängende Teile eines Systems in unterschiedlichen Hierarchieebenen. Sind die einzelnen Module durch Kapselung von ihrer Umgebung separiert wird auch von *modularer Modellierung* [GKN+03] gesprochen. Zum Aufbau von komplexen Modellen aus den einzelnen Modulen, können diese über Schnittstellen nach außen, welche durch Terminalknoten (schwarze Knoten in Abbildung 2.12) modelliert werden, miteinander verbunden werden.

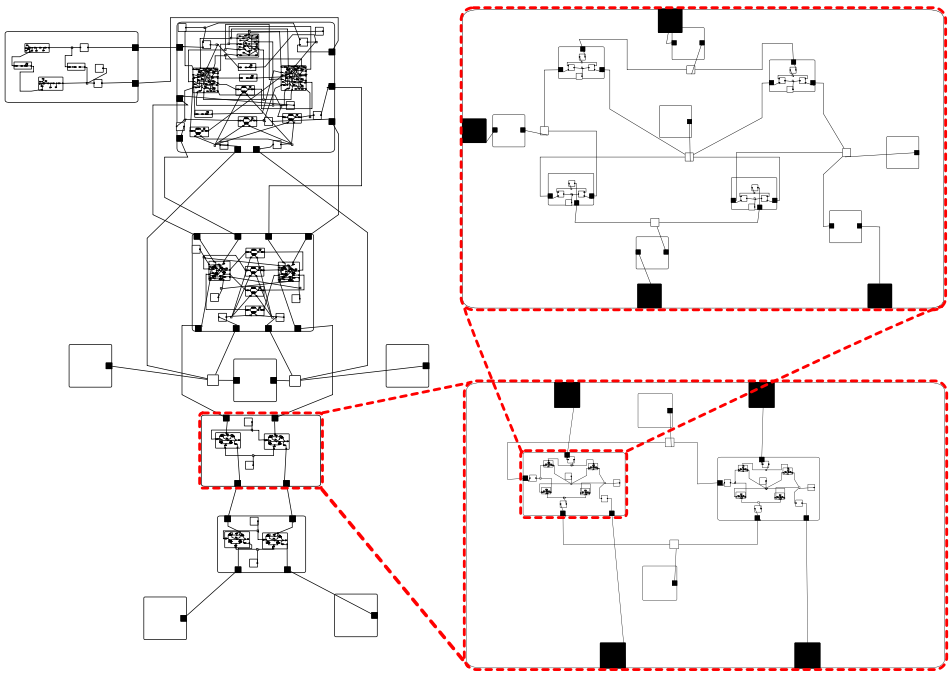


Abbildung 2.12: Ein hierarchisch strukturiertes Modell des EGF-Rezeptor-Netzwerks: Das Modell ist in 73 Module unterteilt und besteht aus mehr als 900 Knoten. Zwei skalierte Module/Subgraphen wurden zur besseren Darstellung vergrößert. Terminalknoten sind schwarz dargestellt.

Das Problem bei der Arbeit mit großen, komplexen Modellen ist, dass diese nicht ohne Weiteres so auf dem Bildschirm darstellbar sind, dass Details des Modells betrachtet werden können, ohne die Orientierung im Modell zu verlieren. Aus diesem Grund ist es notwendig, bei der Darstellung und Exploration großer, komplexer Modelle das Informationsvisualisierungs-Mantra [Shn96]: *„Overview first, zoom and filter, then details-on-demand."* zu beachten.

Eine Möglichkeit den Überblick (*Overview*) über ein hierarchisch strukturiertes Modell zu erhalten, ist es, die Komponenten des Modells entsprechend ihrer Verschachtelungstiefe um einen konstanten Faktor zu skalieren (siehe Abbildung 2.12). Die Exploration der so dargestellten Modelle kann dann, z.B. auf Basis des Konzepts des *ZUI*, unter Anwendung verschiedener Interaktionstechniken (wie z.B. *Fisheye-View* [Fur86] oder *Semantic Zooming* [PF93]) erfolgen. Der Anwender kann mit Hilfe dieser Techniken interaktiv bestimmte Teile des Modells vergrößert darstellen

und so das Modell in verschiedenen Detailierungsgraden betrachten (*details-on-demand*). In Bezug auf die Berechnung des Layouts existieren hierbei zwei Herangehensweisen:

- *Exploration auf Basis eines vorberechneten Layouts*: Bei dieser Art der Exploration ist die Berechnung des Layouts des kompletten Modells vor dem Beginn der Exploration notwendig. Sie wird von der Mehrzahl der Programme zur Exploration von hierarchisch strukturierten Modellen eingesetzt. Der Vorteil dieser Technik ist, dass die Layoutberechnung nur einmal durchgeführt wird und dadurch während der Exploration keine Rechenzeit mehr aufgewendet werden muss. Da sich das Layout des Modells während der Exploration nicht ändert, ist der Erhalt der Mental Map ein weiterer Vorteil. Der Nachteil dieser Variante ist, dass die Berechnung des kompletten Layouts eines größeren Graphen sehr zeitaufwändig sein kann, so dass bei Nichtvorhandensein eines intuitiven initialen Layouts, der Anwender zunächst auf das Ergebnis des Zeichenverfahrens warten muss.

- *Steerable Exploration*: Bei dieser Art der Exploration wird das Layout immer nur für die aktuell sichtbaren Komponenten des Modells bestimmt. Diese Variante wird in wenigen Programmen, z.B. in [AMA07] eingesetzt. Vorteil dieser Variante ist, dass sofort mit der Exploration begonnen werden kann, da zu Beginn nicht das Layout des kompletten Modells bestimmt wird. Der Nachteil ist, dass während der Exploration des Modells Teile des Layouts immer wieder neu berechnet werden müssen, was besonders bei größeren Graphen zum Wartezeiten führen kann. Des Weiteren kann die Mental Map des Anwenders durch Änderungen am Layout während der Exploration zerstört werden.

Der in dieser Arbeit vorgestellte Algorithmus soll hauptächlich zum Erstellen eines Layouts für die erste der beiden Explorationstechniken verwendet werden. Ein intuitives Initiallayout ist hierfür essentiell, da nur dieses eine effektive Exploration des Modells ermöglicht. Aufgrund der hierarchischen Struktur der Modelle ergeben sich hierdurch zusätzlich folgende Anforderungen an das Layout:

- Darstellung von Modulen/Subgraphen,
- Skalierung von Knoten entsprechend ihre Verschachtelungstiefe und
- Platzierung von Terminalknoten am Rand eines Moduls/Subgraphen.

2.3 Auswahl eines Zeichenverfahrens

Es wurden Verfahren zum Zeichnen von Graphen vorgestellt und sowohl allgemeine als auch anwendungsspezifische Kriterien erläutert, die wichtig für ein verständliches und intuitives Layout sind. Darauf aufbauend wird im Folgenden ein Zeichenverfahren ausgewählt, welches als Basis für den zu entwickelnden Algorithmus zum Zeichnen von hierarchisch strukturierten biologischen Netzwerken dienen soll. Wichtig für die Auswahl des Zeichenverfahrens waren dabei folgende Kriterien:

- Anwendbarkeit auf allgemeine Graphen,

- Anwendbarkeit auf Compound-Graphen,
- Implementierungsaufwand,
- Laufzeit und
- Eignung zur Darstellung biologischer Netzwerke.

Alle vorgestellten Zeichenverfahren können prinzipiell zur Darstellung *allgemeiner* Graphen, d.h. Graphen mit gerichteten und ungerichteten Kanten, Graphen ohne besondere Strukturen (mit und ohne Zyklen, beliebiges Verhältnis von Kanten und Knoten) eingesetzt werden. Einige Zeichenverfahren sind jedoch auf bestimmte Arten von Graphen optimiert, so dass die Qualität der Darstellungen von allgemeinen Graphen durch diese nicht optimal ist. Schichtenbasierte Verfahren eigenen sich beispielsweise am besten für DAGs und zirkuläre Verfahren für Graphen mit zyklischen Strukturen.

Mit allen vorgestellten Verfahren ist es möglich, Compound-Graphen zu zeichnen. Konkrete Umsetzungen für kräftebasierte und schichtenbasierte Verfahren sind beispielsweise in [WM95, EH00, DGC+04] (siehe auch Abschnitt 3.1.2) bzw. [SM91, San96] beschrieben. Selbst wenn keine optimalen Erweiterungen der Verfahren zum Layout von Compound-Graphen bestehen, existiert noch die Möglichkeit ein lokales Zeichenverfahren (siehe Abschnitt 3.1.1) für diese Aufgabe zu nutzen.

Die Laufzeit eines Zeichenverfahrens, welches in interaktiven Anwendungen eingesetzt wird, sollte möglichst gering sein. Dies ist wichtig, damit Anwender nicht zu lange auf das Ergebnis warten müssen. In der in [Him95] durchgeführten Untersuchung wurden die Laufzeiten der vorgestellten Zeichenverfahren am Beispiel von Graphen mit bis zu 100 Knoten, verglichen. Kräftebasierte Verfahren wurden hierbei als „langsam“ (Ergebnis in einigen Sekunden bis mehreren Minuten) und schichtenbasierte und orthogonale Verfahren als „mittelschnell“ (Ergebnis in wenigen Sekunden) eingestuft. Zirkuläre Verfahren gelten allgemein als „sehr schnell“ (Ergebnis sofort oder in wenigen Sekunden). Die eigentliche Laufzeit eines Zeichenverfahrens ist jedoch stark abhängig von der konkreten Implementierung, der Nutzung von Optimierungstechniken und der Art der Graphen. So sind, wie bereits erläutert, mittlerweile selbst kräftebasierte Verfahren in der Lage größere Graphen in wenigen Sekunden zu zeichnen.

Der Aufwand zur theoretischen und praktischen Umsetzung der vorgestellten Zeichenverfahren wird im Allgemeinen bei kräftebasierten Verfahren als „niedrig“, bei schichtenbasierten Verfahren als „mittelschwer“ und bei orthogonalen Verfahren als „hoch“ angesehen [Mut99]. Wie auch bei der Laufzeit, ist der tatsächliche Implementierungsaufwand jedoch abhängig von der konkreten Implementierung und den umgesetzten Erweiterungen und Optimierungen.

Zur Darstellung biologischer Netzwerke werden hauptsächlich kräftebasierte und schichtenbasierte Zeichenverfahren verwendet. Orthogonale und zirkuläre Verfahren werden zur Darstellung von biologischen Netzwerken gar nicht bzw. selten eingesetzt [BBS08] (siehe auch Tabelle A.2 im Anhang). Die Nichtanwendung von

orthogonalen Verfahren ist hierbei zum Teil auf den hohen Implementierungsaufwand zurückzuführen. Andererseits entsprechen die „schaltplanartigen“ Layouts nicht den Anforderungen an die Darstellung biologischer Netzwerke. Der Vorteil kräftebasierter Verfahren sind die symmetrischen, organisch wirkenden Layouts, durch die beispielsweise Hubs gut hervorgehoben werden. Aus diesem Grund werden sie oft zur Darstellung von Genregulationsnetzwerken oder Protein-Protein-Interaktionsnetzwerken eingesetzt. Schichtenbasierte Verfahren eignen sich besonders, um qualitativ hochwertige Darstellungen von metabolischen Reaktionsnetzwerken mit definierten Richtungen zu erzeugen. Diese entsprechen dann oft traditionellen Darstellungen [Sch01]. Sie werden z.B. in den Programmen BIOPATH [BFP+04] und BIOMAZE [ZSdG07] eingesetzt.

Aufgrund des Verbreitungsgrades bei der Darstellung biologischer Netzwerke erfolgt die Auswahl des Zeichenverfahrens zwischen kräftebasierten und schichtenbasierten Zeichenverfahren. Letztere eignen sich eher zur Darstellung von DAGs. Da es bei vielen biologischen Netzwerken jedoch vorkommt, dass Reaktionsrichtungen reversibel oder unbekannt sind und daher a priori keine Kantenrichtung festgelegt werden können, muss von ungerichteten oder Mixed-Graphen ausgegangen werden. Da der zu entwickelnde Algorithmus auch auf diese Graphen anwendbar sein soll, wird daher das kräftebasierte Spring-Embedder-Verfahren als Basis für den zu entwickelnden Algorithmus ausgewählt.

Weitere Vorteile des Spring-Embedders sind der im Vergleich zu anderen Zeichenverfahren geringe Implementierungsaufwand und die gute Erweiterbarkeit. Das Problem der hohen Laufzeit kann, wie erläutert, mit Hilfe verschiedener Optimierungstechniken gelöst werden. Wobei eine geringe Wartezeit bis zur Berechnung des Layouts hinnehmbar ist, wenn dieses nur einmal berechnet werden muss und es sich während der Exploration des Netzwerks nicht ändert.

In Tabelle 2.2 wird die Bewertung der vorgestellten Zeichenverfahren in Bezug auf die erläuterten Kriterien zusammengefasst.

	kräftebasiert	schichtenbasiert	orthogonal	zirkulär
Allgemeine Graphen	+	-	+	-
Compound-Graphen	+	+	+	-
Biologische Netzwerke	+	+	--	-
Implementierungsaufwand	+	-	--	+
Laufzeit	-	+	+	++

Tabelle 2.2: Bewertung der Eignung verschiedener Zeichenverfahren in Bezug auf die dargestellten Kriterien: „++“ bedeutet sehr gut geeignet und „--“ nicht geeignet.

2.4 Zusammenfassung

Im ersten Teil dieses Kapitels wurden Grundlagen aus dem Bereich des Graphzeichnens erläutert. Hierzu wurden, die für das Verständnis dieser Arbeit notwendigen

Begriffe eingeführt, allgemeine Kriterien zur Bewertung der Qualität von Darstellungen von Graphen erläutert und die verbreitetsten Zeichenverfahren für Graphen vorgestellt. Anschließend wurde in die Visualisierung biologischer Netzwerke eingeführt und einige besondere Anforderungen an das Layout biologischer Netzwerke beschrieben. Am Ende des Kapitels wurden verschiedene Zeichenverfahren, bezüglich ihrer Eignung zur Darstellung von Graphen aus dem Anwendungsgebiet der Systembiolgie, verglichen und das kräftebasierte Spring-Embedder-Verfahren als Basis, für den in dieser Arbeit zu entwickelnden Algorithmus, ausgewählt.

3 Verwandte Arbeiten

Nachdem im vorigen Kapitel das Spring-Embedder-Verfahren als Grundlage für den zu entwickelnden Algorithmus zum Zeichnen von Compound-Graphen ausgewählt wurde, werden im ersten Teil dieses Kapitels auf dem Spring-Embedder basierende Verfahren zum Zeichnen von Compound-Graphen beschrieben. Im zweiten Teil werden Techniken vorgestellt, mit denen die Integration der in Abschnitt 2.2.2 genannten semantischen Anforderungen an das Layout von biologischen Netzwerken in ein kräftebasiertes Verfahren, wie dem Spring-Embedder, ermöglicht werden kann.

3.1 Verfahren zum Zeichnen von Compound-Graphen

Existierende Methoden zum Zeichnen von Compound-Graphen lassen sich in lokale und globale Verfahren einteilen. Im Folgenden wird eine Auswahl der wichtigsten, auf dem Spring-Embedder-Modell basierenden, lokalen und globalen Verfahren vorgestellt.

3.1.1 Lokale Verfahren

Lokale Zeichenverfahren, wie beispielsweise in [BM99, PV06], berechnen das Layout eines Compound-Graphen nach dem *Divide&Conquer*-Prinzip. Bei diesem werden die Layouts der einzelnen Subgraphen zunächst unabhängig voneinder bestimmt. Anschließend werden die Subgraphen zum Gesamtgraphen zusammengesetzt.

Abbildung 3.1 veranschaulicht die Vorgehensweise lokaler Layoutverfahren am Beispiel des Graphen aus Abbildung 2.3. Im einfachsten Fall wird der Inklusionsbaum ebenenweise, beginnend in der vorletzten Ebene, von unten nach oben (*Bottom-up*) traversiert und das Layout der einzelnen Subgraphen nacheinander bestimmt. Die Subgraphen aus Ebene n werden dabei in Ebene $n-1$ als einfache Knoten betrachtet. Die Berechnung des inneren Layouts eines Subgraphen geschieht hierbei unabhängig von anderen Subgraphen, d.h. Kanten, die den Subgraphen verlassen, werden nicht in die Berechnung einbezogen.

Der Vorteil lokaler Zeichenverfahren besteht darin, dass die Layouts der einzelnen Subgraphen unabhängig voneinander berechnet werden und somit die Kräfte nur lokal verrechnet werden müssen. Aus diesem Grund kommt es zu einer Verringerung des Implementierungsaufwandes, da keine komplexen Datenstrukturen zur Verwaltung eines kompletten Compound-Graphen notwendig sind. Des Weiteren kommt es durch das Divide&Conquer-Prinzip lokaler Verfahren zu einer Verringerung der Laufzeit gegenüber globalen Verfahren (siehe folgenden Abschnitt). Ein

weiterer Vorteil lokaler Verfahren ist die Möglichkeit, unterschiedliche Zeichenverfahren für die einzelnen Subgraphen zu benutzen, da die Subgraphen unabhängig betrachtet werden.

Nachteilig ist die Nichtbeachtung der Beziehungen zwischen Knoten aus verschiedenen Subgraphen, welche zum Entstehen von langen Kanten, Kanten-Kanten- und Kanten-Knoten-Kreuzungen führt (siehe dazu auch Abbildung 3.1(d)). Hierdurch kommt es zu einer Verringerung der Darstellungsqualität.

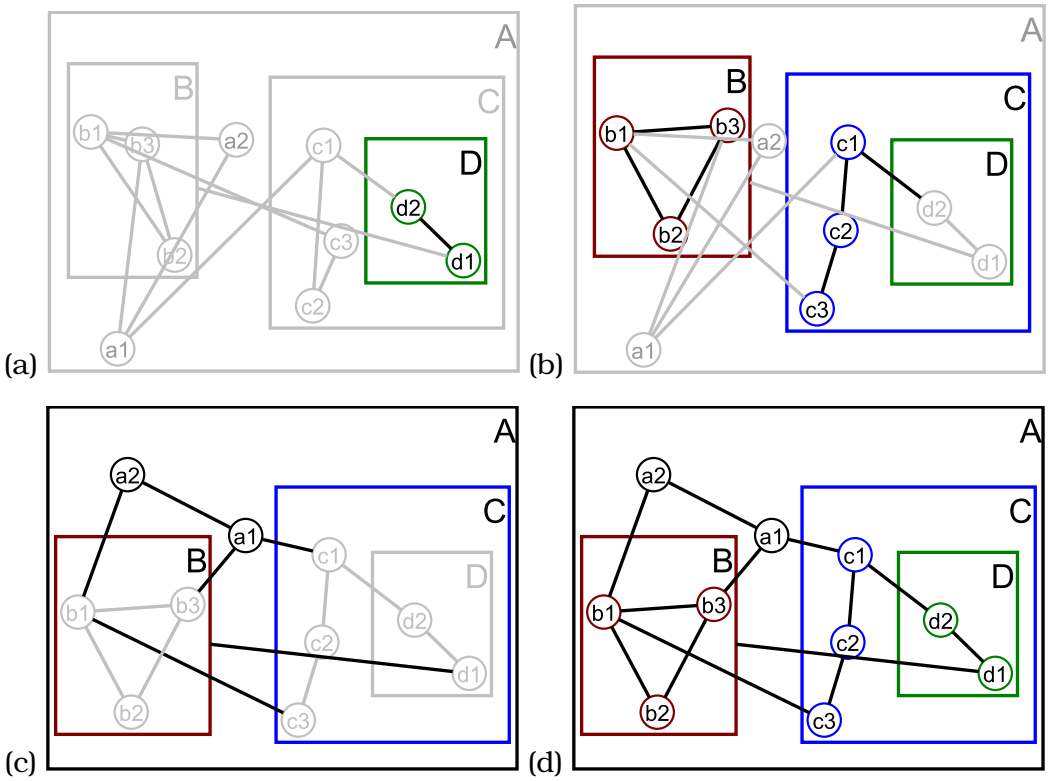


Abbildung 3.1: Vorgehensweise lokaler Layoutverfahren. Kanten und Knoten, die grau dargestellt sind, werden im jeweiligen Schritt nicht betrachtet: (a) Beginnend in der vorletzten Ebene des Inklusionsbaums wird zuerst das Layout von Subgraph D bestimmt. (b) Im nächsten Schritt werden Subgraphen B und C gezeichnet. Subgraph D wird hierbei als einfacher Knoten betrachtet und sein inneres Layout nicht mehr verändert. (c) Im letzten Schritt wird das Layout von A bestimmt. (d) Das finale Layout des Graphen.

Zur Vermeidung der Nachteile lokaler Verfahren wird in [Ker05] ein lokales Verfahren beschrieben, welches auch globale Zusammenhänge in die Layoutberechnung mit einbezieht. Hierbei wird zunächst, wie erläutert, der Compound-Graph Bottom-up gezeichnet. In einem zweiten Layoutdurchgang werden die einzelnen Subgraphen des Compound-Graphen noch einmal, diesmal jedoch von oben nach unten (*Top-down*), gezeichnet. Dieser zweite Layoutdurchgang hat zum Ziel, die Zahl an Kreuzungen von Kanten, welche zwischen Knoten aus verschiedenen Subgraphen verlaufen, zu verringern. Dazu werden die Subgraphen unter Beachtung der relativen Position von Knoten aus höheren Hierarchiebenen, welche durch eine Kante

mit Knoten aus dem aktuell betrachtetem Subgraphen verbunden sind, erneut gezeichnet. Da die Position von Knoten aus höheren Hierarchieebenen hierbei jedoch nicht verändert wird, sind die erzielten Verbesserungen nur minimal.

3.1.2 Globale Verfahren

Globale Zeichenverfahren betrachten den kompletten Compound-Graphen und damit auch die Beziehungen zwischen Objekten aus verschiedenen Hierarchieebenen während der Berechnung des Layouts. Im Folgenden werden drei der bekanntesten globalen Zeichenverfahren vorgestellt.

Das in [WM95] beschriebene Verfahren eignet sich zum Zeichnen von Cluster-Graphen mit einer Verschachtelungstiefe < 3. Hierbei werden die Kanten in drei Klassen unterteilt:

- *Intrakanten* (Kanten zwischen Knoten aus demselben Subgraphen),
- *Interkanten* (Kanten zwischen Knoten aus verschiedenen Subgraphen) und
- *Metakanten* (virtuelle Kanten zwischen Subgraphen).

Das Verfahren läuft in drei Phasen ab und verwendet für jeden Kantentyp unterschiedliche Kantenkräfte (Intra-, Inter- und Metakräfte), deren Stärke sich im Laufe des Verfahrens ändert. In Phase 1 wird das Layout des Graphen ohne Betrachtung der Strukturierung bestimmt (keine Metakräfte). Anschließend werden in Phase 2 die Interkräfte verringert und die Metakräfte erhöht und die Strukturierung herausgearbeitet. Phase 3 ähnelt der Komposition von bereits layouteten Subgraphen in lokalen Verfahren, da sich das innere Layout der Subgraphen nicht mehr ändert, keine Interkräfte mehr wirken und die Positionierung der Subgraphen nur noch durch die Metakräfte bestimmt wird (siehe auch Abbildung 3.2).

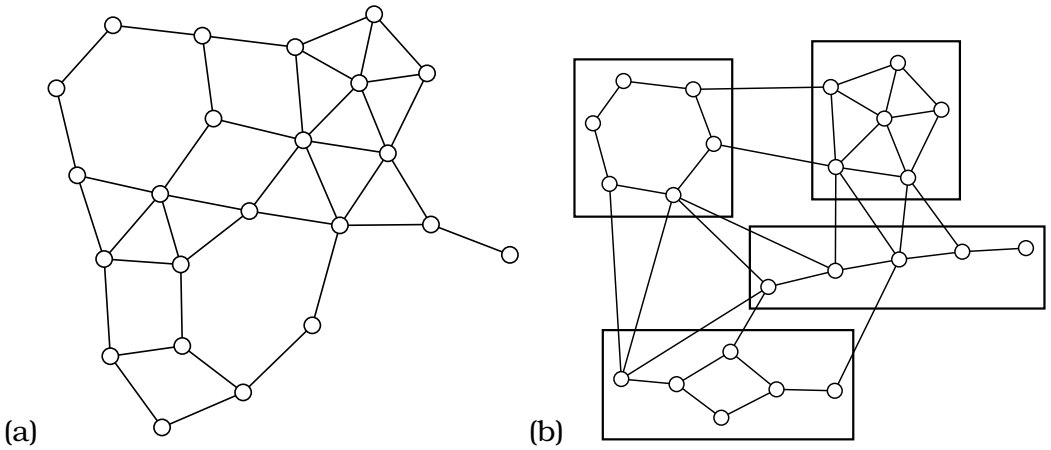


Abbildung 3.2: (a) Layout des Graphen nach Phase 1 (ohne Beachtung der Strukturierung) und (b) das fertige Layout, nachdem Metakräfte zur Herausarbeitung der Strukturierung hinzugefügt wurden (aus [WM95]).

In [EH00] wird ebenfalls eine Erweiterung des Spring-Embedder-Modells zum Layout von Cluster-Graphen präsentiert, bei dem verschiedene Kräfte für die unter-

schiedlichen Kantentypen benutzt werden. Um die Knoten innerhalb eines Subgraphen zusammenzuhalten werden, im Gegensatz zum Konzept der Metakräfte bei [WM95], sogenannte virtuelle Kräfte verwendet. Dazu wird im Zentrum jedes Subgraphen ein virtueller Knoten eingefügt und über virtuelle Kanten mit allen Knoten des Subgraphen verbunden (siehe Abbildung 3.3).

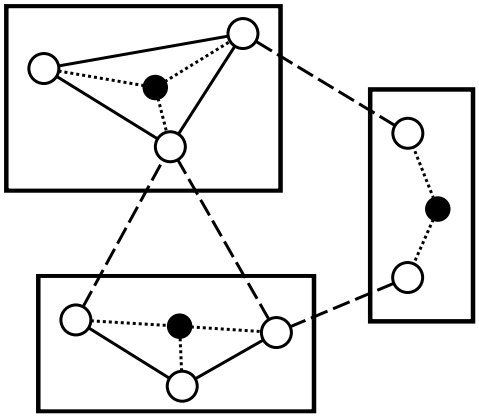


Abbildung 3.3: Verschiedenen Kantentypen aus [EH00]: Intrakanten sind mit durchgezogener, Interkanten mit gestrichelter und virtuelle Kanten mit gepunkteter Linie gekennzeichnet.

Das am weitest entwickelte globale Verfahren wird in [DGC+04] vorgestellt. Es eignet sich, im Unterschied zu den beiden zuvor vorgestellten Verfahren, zum Layout allgemeinen Compound-Graphen mit beliebiger Verschachtelungstiefe und wird in PATIKA, einem Programm zur Visualisierung biologischer Reaktionsnetzwerke, eingesetzt.

Das Verfahren verwendet verschiedene Optimierungen zur Erhöhung der Qualität und zur Verbesserung der Laufzeit:

- Um die Knoten eines Subgraphen zusammenzuhalten werden, anstatt virtueller Kräfte oder Metakräfte, Gravitationskräfte benutzt, welche auf alle Knoten eines Subgraphen eine Kraft in Richtung des Massenschwerpunktes des Subgraphen ausüben.
- Für die Behandlung beliebig verschachtelteter Compound-Graphen wird die Länge von Interkanten proportional zur Differenz zwischen der Tiefe der beiden Endknoten erhöht.
- Als erstes globales Verfahren nutzt es die hierarchische Strukturierung der Graphen zur Verringerung der Komplexität. Die aufwändige Berechnung der Abstoßungskräfte zwischen allen Knoten erfolgt dabei nur zwischen Knoten aus demselben Subgraphen.

Ein weiterer Unterschied gegenüber den vorher genannten Verfahren ist die Nutzung einer zweiphasigen Layoutberechnung unter Verwendung einer Skelettierungstechnik: Zunächst wird in Phase 1 das Skelett des Graphen erzeugt, indem alle Knoten, die nicht zu Subgraphen gehören und Teil eines Baumes sind, bis zur Wurzel temporär gelöscht werden (siehe Abbildung 3.4). Dieser Skelettgraph wird

dann durch den beschriebenen Algorithmus gezeichnet. In Phase 2 werden die temporär gelöschten Knoten wieder eingefügt und das Layout des Graphen auf Basis des Skelettgraphen berechnet. Diese Technik bringt in den meistem Fällen einen Geschwindigkeitsvorteil, da dadurch, dass der Skelettgraph weniger Knoten als der Originalgraph enthält, die Berechnung seines Layouts schneller möglich ist. Das Layout des Skelettgraphen stellt hierbei die grobe Struktur des Graphen dar und ist daher ein günstiges Ausgangslayout zur Bestimmung des Layouts des Originalgraphen. Ein weiterer Vorteil, der durch die Verwendung dieser Skeletttechnik entsteht, ist die teilweise Vermeidung der bei kräftebasierten Verfahren auftretenden lokalen Minima[1].

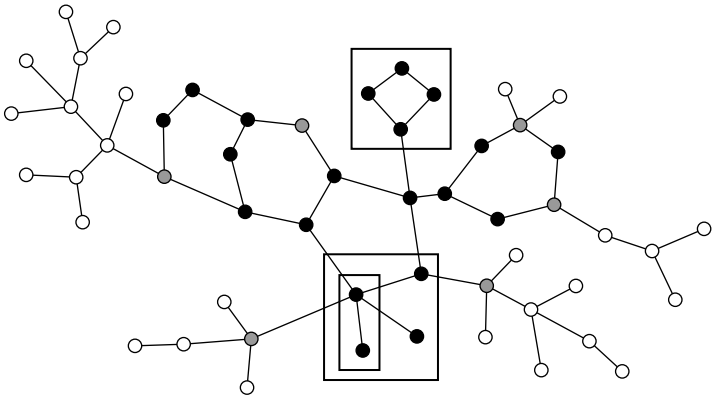


Abbildung 3.4: Skelettierungstechnik aus [DGC+04]: Der Skelettgraph besteht aus schwarzen und grauen Knoten. Temporär entfernte Detailknoten sind weiß dargestellt.

Globale Verfahren lösen die Nachteile lokaler Methoden (lange Interkanten, Kanten-Kanten- und Kanten-Knoten-Kreuzungen) durch gleichzeitige Betrachtung der Beziehungen zwischen Knoten aus allen Hierarchieebenen des Compound-Graphen. Hierdurch wird eine globale Optimierung des Layouts ermöglicht, wodurch die Darstellungsqualität im Vergleich zu lokalen Verfahren verbessert wird.

Globale Verfahren nutzen die hierarchische Struktur eines Compound-Graphen nicht oder nur wenig zur Optimierung der Berechnung des Layouts. Ihr Nachteil ist daher die hohe Komplexität, die aufgrund der gleichzeitigen Betrachtung aller Knoten des Compound-Graphen entsteht. Dies führt zum einen zu einem höheren Berechnungsaufwand im Vergleich zu lokalen Verfahren. Zum anderen sind hierdurch komplexe Methoden und Datenstrukturen zur Verwaltung des kompletten Compound-Graphen notwendig, wodurch der Implementierungsaufwand erhöht wird.

3.2 Techniken zur Integration biologischer Informationen

Es existieren eine Vielzahl an Programmen zum Erstellen und Visualisieren von biologischen Netzwerken (siehe Tabelle A.2 im Anhang). In den meisten Program-

1 Lokale Minima sind in kräftebasierten Verfahren auftretenden Knotenkonfigurationen die zwar energetisch günstig sind (und so zum Abbruch des Algorithmus führen) jedoch nach ästhetischen Kriterien nicht optimal sind.

me kommen die in Abschnitt 2.1.3 beschriebenen Standard-Zeichenverfahren zum Einsatz, entweder implementiert oder unter Verwendung kommerzieller Bibliotheken, wie yFILES [WEK04] und TOMSAWYER LAYOUT [DFM+02] oder frei verfügbarer Bibliotheken, wie GRAPHVIZ [EGK+01] und GRAVISTO [BBF+04]. Einige Programme erweitern diese Standard-Zeichenverfahren zur Beachtung der in Abschnitt 2.2.2 genannten Anforderungen an das Layout biologischer Netzwerke. Im Folgenden wird eine Auswahl dieser Erweiterungen, welche sich zur Integration in ein kräftebasiertes Zeichenverfahren eignen, vorgestellt.

3.2.1 Integration der Richtung von Reaktionspfaden

Wie in Abschnitt 2.2.2.1 beschrieben, ist die Darstellung der Richtung von Reaktionen und Reaktionspfaden wichtig um diese finden und analysieren zu können. Die in den Programmen SBMLLAYOUT [DBS06] und PATIKA [GD03] verwendeten Zeichenverfahren benutzen das Konzept der magnetischen Kräfte [SM94] um Kanten entsprechend der durch sie dargestellten Reaktionsrichtung zu orientieren. Die Kanten werden hierzu als magnetische Federn modelliert, die durch ein Magnetfeld ausgerichtet werden. Die durch das Magnetfeld auf eine Kante wirkende magnetische Kraft FM wird mit folgender Formel berechnet:

$$FM = m * d^{\alpha} * \phi^{\beta} \tag{3.1}$$

Die Parameter m, d, ϕ, α und β bezeichen hierbei die Stärke des Magnetfeldes, die Länge der Kante, den Winkel zwischen Kante und Magnetfeld, den Einfluss der Kantenlänge bzw. den Einfluss des Winkels zwischen Kante und Magnetfeld auf die Stärke der magnetischen Kraft. Zur Ausrichtung der Kante werden ihre Endknoten, entprechend der durch FM dargestellten Kraft, in entgegengesetzter Richtung verschoben. Abbildung 3.5 veranschaulicht die Wirkungsweise der Ausrichtung einer Kante durch magnetische Kräfte.

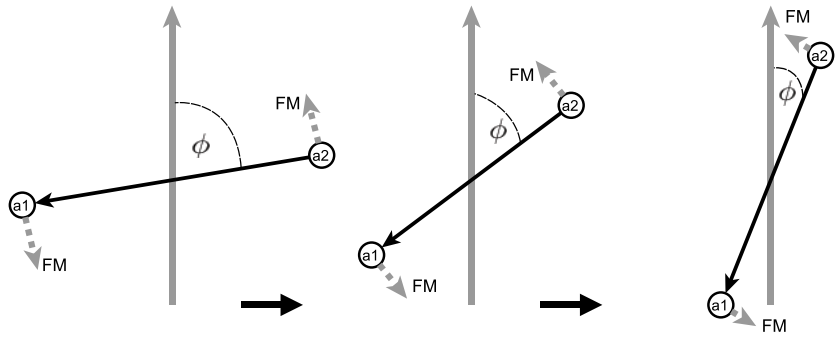


Abbildung 3.5: Wirkungsweise des Konzepts der magnetischen Kräfte. Die gerichtete Kante zwischen Knoten $a1$ und $a2$ wird durch das nach Norden ausgerichtete Magnetfeld (grauer Pfeil) nach unten gedreht. Die auf die Knoten wirkenden magnetischen Kräfte FM sind durch die grauen, gepunkteten Pfeile dargestellt.

Da die Orientierung einer Kante durch die Richtung des Magnetfeldes bestimmt wird, ist es möglich, unterschiedlich gerichtete Magnetfelder zu nutzen, um Kanten unterschiedlich auszurichten. In den meisten Fällen werden alle Kanten jedoch nur

in eine Richtung (meist von oben nach unten) ausgerichtet.

Die besten Ergebnisse der Darstellung von Kantenrichtungen durch magnetische Kräfte werden, wie bei schichtenbasierten Verfahren auch, für DAGs erzielt. Daher ist es sinnvoll einen Graphen vor der Ausrichtung der Kanten in einen DAG umzuwandeln. Effiziente Verfahren, die genau dieses Problem lösen (z.B. [ELS93], sind vorhanden.

3.2.2 Integration der subzellulären Lokalisation

Die Darstellung der subzellulären Lokalisation trägt wie in Abschnitt 2.2.2.2 ebenfalls zum Verständnis der Vorgänge innerhalb zellulärer Netzwerke bei. Abbildung 3.6 zeigt die Darstellung der subzellulären Lokalisation in den Programmen CEREBRAL [BGHM07] und PATIKA.

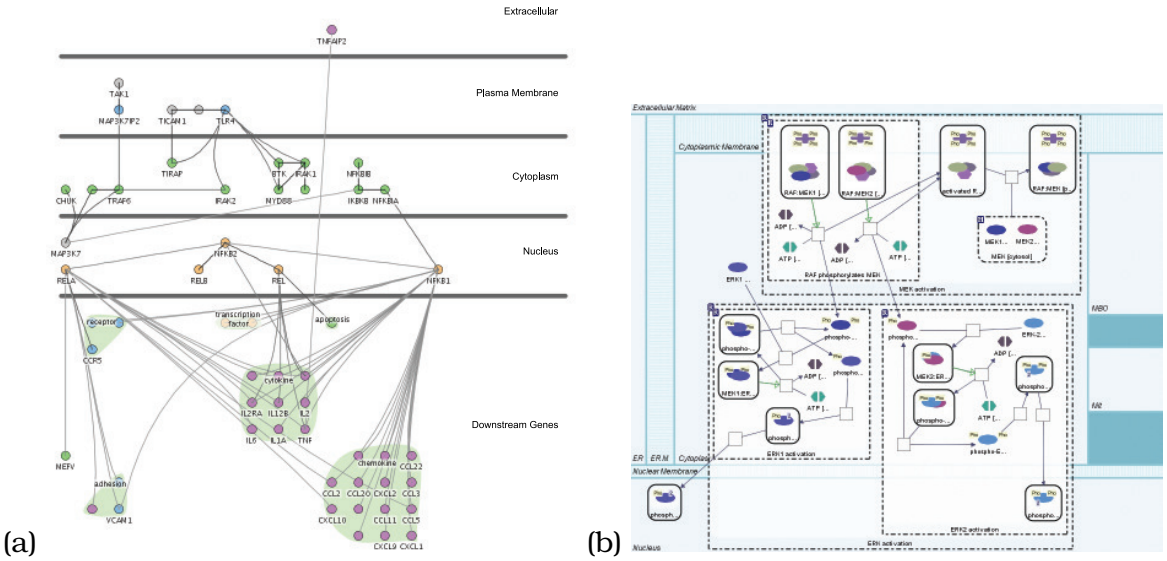


Abbildung 3.6: Darstellung der subzellulären Lokalisation: (a) Schichtweise Anordnung der Zellkompartimente in CEREBRAL und (b) Anordnung von Zellkompartimenten in PATIKA übereinander und nebeneinander.

In CEREBRAL werden zur Darstellung der subzellulären Lokalisation Knoten mit bestimmten Attributen entweder automatisch oder durch den Anwender in schichtweise angeordnete Zellkompartimente eingeordnet (siehe Abbildung 3.6(a)). Das verwendete Zeichenverfahren, welches auf *Simulated Annealing* [DH96] basiert, stellt dann mit Hilfe einer Bewertungsfunktion sicher, dass die Knoten in diesen Schichten verbleiben. Da das in CEREBRAL verwendete Zeichenverfahren auf einem anderen Ansatz basiert, ist die Integration mit Hilfe einer Bewertungsfunktion in ein Spring-Embedder-basiertes Verfahren schwer umsetzbar. Jedoch wäre es möglich, die Separatoren, die die zellulären Schichten trennen, wie in dem im Folgenden beschrieben Ansatz von PATIKA in ein Spring-Embedder-basiertes Verfahren zu integrieren. Dadurch können die Knoten daran gehindert werden, die ihnen zugeordneten Schichten zu verlassen. Des Weiteren wäre es sinnvoll, die Länge der Kanten entsprechend der zwischen ihren Endknoten verlaufenden Schichten anzupassen, da wie in Abbildung 3.6(a) dargestellt, zwischen Knoten aus verschiedenen Schich-

ten längere Kanten notwendig sind.

In PATIKA werden im Gegensatz zu CEREBRAL die Zellkompartimente nicht schichtweise angeordnet, da hier die Separatoren nicht nur horizontal, sondern auch vertikal verlaufen (siehe Abbildung 3.6(b)). Die Separatoren, die die Grenzen der Zellkompartimente beschreiben, werden als Teile des physikalischen Systems angesehen und entsprechend der auf sie wirkenden Kräfte vertikal oder horizontal, wie in Abbildung 3.7 dargestellt, bewegt.

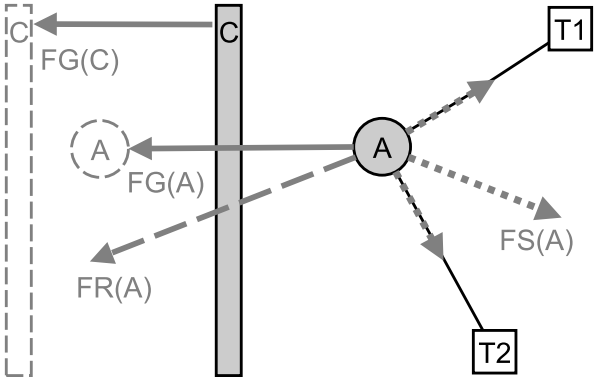


Abbildung 3.7: Verschiedene Kräfte des Kräftemodells in PATIKA [GD03], die auf den Knoten A wirken (Federkräfte $FS(A)$ und Abstoßungskräfte $FR(A)$). Als Resultat werden Knoten A sowie der Kompartiment-Separator C, durch die Summe der auf sie wirkenden Kräfte ($FG(A)$ bzw. $FG(C)$), nach links verschoben.

Eine weitere Möglichkeit zur Behandlung von Zellkompartimenten in PATIKA wird in [DGC+04] beschrieben. Bei dieser werden die Zellkompartimente, wie in Abbildung 3.6(b) dargestellt, ähnlich wie Subgraphen als rechteckige Bereiche behandelt. Im Unterschied zu Subgraphen grenzen benachbarte Zellkompartimente direkt aneinander. Daher ist es bei einer Änderung der Geometrie eines Zellkompartimentes notwendig die Geometrie aller benachbarten Zellkompartimente entsprechend anzupassen.

3.3 Zusammenfassung

In diesem Kapitel wurden existierende Verfahren zum Zeichnen von Compound-Graphen auf Basis des Spring-Embedders vorgestellt. Diese Verfahren lassen sich grundsätzlich in lokale und globale Verfahren unterscheiden und besitzen verschiedene Eigenschaften (lokal: schnell, schlechte Qualität, geringer Implementierungsaufwand und global: langsam, gute Qualität, hoher Implementierungsaufwand). Weiterhin wurden Techniken vorgestellt mit denen semantische Informationen, wie die Beachtung der Reaktionsrichtung, in ein Spring-Embedder-basiertes Verfahren integriert werden können.

4 Konzept

Dieses Kapitel befasst sich mit der Entwicklung eines neuen Spring-Embedder-basierten Verfahrens zum Zeichnen von Compound-Graphen. Im ersten Teil des Kapitels werden Anforderungen an ein effektives Layout eines hierarchisch strukturierten Modells aus der Systembiologie erläutert. Anschließend erfolgt eine Bewertung der Eignung existierender Verfahren zur Darstellung dieser Modelle. Der zweite Teil des Kapitels beschäftigt sich mit der Vorstellung eines neuen Algorithmus, der eine Mischform der beiden existierenden Ansätze zum Zeichnen von Compound-Graphen darstellt. Im weiteren Verlauf des Kapitels werden Erweiterungen dieses Algorithmus zur Erhöhung der Qualität der erzeugten Darstellungen beschrieben. Das Kapitel endet mit der Betrachtung von Interaktionsmöglichkeiten von Anwendern mit einem automatischen Zeichenverfahren.

4.1 Anforderungsanalyse

Wie in Kapitel 2 erläutert wurde, ist das Layout eines Graphen wichtig für die verständliche Darstellung der durch ihn repräsentierten Informationen. Werden Modellbeschreibungen ohne Layoutinformationen erzeugt oder in Austauschformaten, wie z.B SBML, abgelegt, enthalten diese keine Layoutinformationen. Öffnet ein Benutzer innerhalb eines Modellierungswerkzeugs ein Modell ohne Layoutinformationen, wird dem Anwender unter Umständen, die in Abbildung 4.1 dargestellte Visualisierung des Modells präsentiert. Möchte sich der Anwender nun einen Überblick

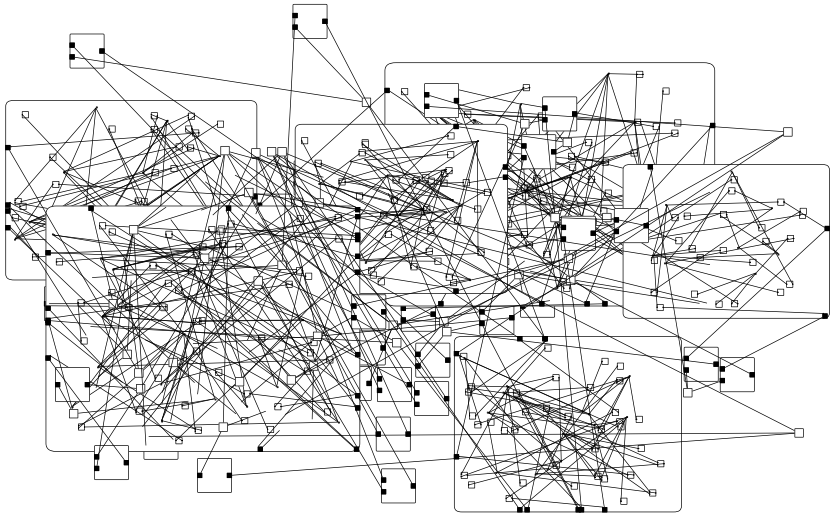


Abbildung 4.1: Graph ohne Layoutinformation. Für einen ersten Überblick über die Komplexität des Graphen, wurden die Knoten zufällig platziert.

über Aufbau des Modells und Beziehungen zwischen Elementen des Modells, mit Hilfe verschiedener Interaktionstechniken verschaffen, ist ein so dargestelltes Modell aufgrund der in Kapitel 2 erläuterten Ästhetikkriterien und Anforderungen relativ ungeeignet. Besteht keine Möglichkeit der (semi-)automatischen Generierung eines verständlichen Layouts, bleibt dem Anwender nur die Option, die Knoten und Kanten des Graphen manuell zu positionieren, um so ein verständliches Layout zu erzeugen. Dies kann bei großen und komplexen Modellen eine sehr zeitaufwändige Aufgabe sein. Gerade bei stark strukturierten und verschachtelten Modellen ist die manuellen Erstellung eines angemessenen Layouts daher fast unmöglich.

4.1.1 Anforderungen

Das Ziel ist es daher, für die beschriebenen Modelle ohne Layoutinformationen, ein geeignetes Layout zu finden, um dem Anwender die visuelle Analyse des Modells zu ermöglichen. Ziele der visuellen Analyse sind hier beispielsweise das Finden von besonders wichtigen Elementen innerhalb des biologischen Netzwerks oder das Erkennen von Reaktionspfaden. Zur Bewertung der Darstellungsqualität der genannten Modelle wurden die folgenden Ästhetikkriterien ausgewählt (Ästhetikkriterien 1-5 wurden aus [OS07] adaptiert, Ästhetikkriterium 6 ergibt sich durch anwendungsspezifische Anforderungen aus der Systembiolgie):

1. *Darstellung der Inklusion:* Kindknoten eines Subgraphen sollen innerhalb und andere Knoten außerhalb des Subgraphen platziert werden. Terminalknoten sollen am Rand des sie beinhaltenden Subgraphen platziert werden.

2. *Vermeidung von Knotenüberlappungen:* Knotenüberlappungen sollen vermieden werden.

3. *Gleichmäßige Knotenverteilung:* Knoten sollen auf der vorhandenen Zeichenfläche möglichst gleichmäßig verteilt werden.

4. *Kurze Kanten:* Durch eine Kante verbundene Knoten sollen nah, jedoch nicht zu nah, zueinander platziert werden.

5. *Minimierung von Kantenkreuzungen:* Kantenkreuzungen sollen vermieden werden.

6. *Beachtung biologischer Informationen:* Existieren semantische Informationen, wie definierte Kantenrichtungen oder Konventionen zur Platzierung von Knoten zueinander, sollen diese beim Layout beachtet werden.

Da die Modelle in der Systembiologie sich in Eigenschaften wie Struktur (Zahl an Submodulen, Verschachtelungstiefe), Größe (wenige Knoten und Kanten bis mehrere tausend) und vorhandenen semantischen Informationen (Richtungsinformationen, Platzierungskonventionen) stark unterscheiden, muss das Layoutverfahren auf diese unterschiedlichen Eigenschaften anpassbar sein. Eine weitere Begründung für diese Anforderung ist die Tatsache, dass die individuellen Vorlieben für die Anordnung einzelner Elemente des Modells von Anwender zu Anwender verschieden sein können. Während ein Anwender beispielsweise Kanten-Knoten-Kreuzungen weniger störend, Kantenknicke jedoch als sehr störend empfindet, kann bei einem

anderen Anwender das Gegenteil der Fall sein. Aus diesem Grund müssen entsprechende Interaktionsmöglichkeiten vorhanden sein. Hierzu gehören:

- Die Anpassung der Parameter des Layoutverfahrens.
- Die Einbeziehung des Anwenders in den Layoutprozess, beispielsweise durch die Möglichkeit eines manuellen Layouts ausgewählter Knoten.
- Die Mitteilung des Status des Layoutprozesses und die Möglichkeit, diesen gegenenfalls abzubrechen.

4.1.2 Fazit aus verwandten Arbeiten

Wie in Abschnitt 4.1.2 erläutert, verfolgen Verfahren zum Layout von Compound-Graphen entweder einen lokalen oder einen globalen Ansatz. Der Vorteil des lokalen Ansatzes ist die durch das verwendete Divide&Conquer-Prinzip geringere Laufzeit, da so jeder Subgraph unabhängig betrachtet werden kann. Die daraus resultierenden Probleme (z.B. Kantenkreuzungen zwischen Interkanten) können durch Verwendung des globalen Ansatzes, auf Kosten einer höheren Komplexität und damit höheren Laufzeit, vermieden werden. Wünschenswert wäre es hier, Vorteile beider Ansätze verbinden zu können, um so einen besseren Trade-off zwischen Laufzeit und Darstellungsqualität zu erhalten.

Keines der in Abschnitt betrachteten Verfahren beachtet die Skalierung von Subgraphen. Die Integration der Beachtung skalierter Subgraphen in lokale Verfahren stellt kein Problem dar, da hierbei jeder Subgraph unabhängig betrachtet wird. Zur Integration in globale Verfahren ist jedoch eine Anpassung der auf die inneren Knoten wirkenden Kräfte notwendig. Dies kann für stark verschachtelte Graphen mit tiefen Hierarchien sehr komplex werden, da hier die Kräfte zwischen unterschiedlich skalierten Knoten aus beliebigen Hierarchieebenen verrechnet werden müssen und es so zu unerwünschten numerischen Effekten kommen kann.

Ein weiterer Punkt der in keinem der vorgestellten Verfahren betrachtet wird, ist das Konzept der Terminalknoten und damit eine explizite Nutzung der Eigenschaften des Terminal-Cluster-Graphen. Um die Anforderungen an die Positionierung der Terminalknoten in die vorgestellten Verfahren zu integrieren, könnten diese einfacherweise am Schnittpunkt zwischen Interkante und Grenzen des Subgraphen platziert werden.

4.2 Ein hybrides Zeichenverfahren für Compound-Graphen

Der Terminal-Cluster-Graph, der der Darstellung modular modellierter Modelle zugrunde liegt, hat die Eigenschaft, dass Kanten immer nur über maximal zwei benachbarten Hierarchieebenen verlaufen. Wie in Abschnitt 2.1.1 erläutert, kann durch das Einfügen von Terminalknoten an den Schnittpunkten einer Interkante mit der Begrenzung eines Subgraphen, jeder allgemeine Compound-Graph in einen Terminal-Cluster-Graphen überführt werden (vergleiche Abbildungen 2.3(a) und 2.4(a)). Somit wird es nun möglich, unter Ausnutzung der Eigenschaften eines Terminal-Cluster-Graphen, lokale und globale Verfahren zum Zeichnen von

Compound-Graphen zu einem hybriden Verfahren zu verbinden und so die Vorteile beider Verfahren zu nutzen. Dieses Verfahren wird im weiteren Verlauf als HyCoGLa (Hybrid Compound-Graph Layout) bezeichnet und im Folgenden vorgestellt.

Die Grundidee von HyCoGLa ist es, wie bei lokalen Verfahren auch, immer nur eine Teilmenge des Graphen gleichzeitig zu betrachten. Im Gegensatz zu lokalen Verfahren besteht hierbei allerdings diese Teilmenge nicht nur aus Kindknoten des aktuell betrachteten Subgraphen, sondern zusätzlich aus allen Kindknoten der Subgraphen des aktuellen Subgraphen.

Der Vorteil dieser Herangehensweise ist, dass so die Berechnung des Layouts des Graphen nun unter Beachtung der benötigten globalen Zusammenhänge zwischen zwei Hierarchieebenen geschehen kann, da davon ausgegangen wird, dass Subgraphen die miteinander verbunden sind, auch einen engen Bezug zueinander haben. Da jedoch immer nur eine Teilmenge der Knoten und Kanten betrachtet wird, kann gleichzeitig auch die Strukturierung des Graphen zur Verringerung der Laufzeit genutzt werden. Somit wird es prinzipiell möglich, einen besseren Trade-off zwischen Laufzeit und Darstellungsqualität als bei rein lokalen oder rein globalen Verfahren zu erzielen. Ein weiterer Vorteil ist, dass die Skalierung immer nur zwischen zwei Hierarchieebenen betrachtet werden muss. Dadurch wird die Komplexität der Berechnung der Kräfte zwischen Knoten aus verschiedenen Hierarchieebenen verringert.

In Algorithmus 4.1 ist der Ablauf von HyCoGLa dargestellt. Der Inklusionsbaum wird beginnend in der dritttiefsten Ebene, ebenenweise (*levelorder*), Bottom-up traversiert. Ist der aktuell besuchte Knoten ein Subgraph, werden seine Kindknoten zusammen mit den Kindknoten der Subgraphen des aktuellen Subgraphen, mit Hilfe des in Algorithmus 4.2 beschriebenen Erweiterten Spring-Embedders gezeichnet.

Algorithmus 4.1 HyCoGLa

Input: Graph $G(V,E)$
Output: A layout of G

```
1: depth ← inclusionTree.depth − 2
2: while depth > −1 do
3:    subgraphsList ← getSubgraphsOfLevel(depth)
4:    for all subgraph ∈ subgraphsList do
5:       ExtendedSpringEmbedder(subgraph)
6:    end for
7:    depth ← depth − 1
8: end while
```

In Abbildung 4.2 ist die Funktionsweise von HyCoGLa am Beispiel des Terminal-Cluster-Graphen aus Abbildung 2.4(a) illustriert. Die Laufzeit des Algorithmus kann verbessert werden, wenn immer nur Subgraphen, welche wiederum Subgraphen enthalten, mit Hilfe des Erweiterten Spring-Embedders gezeichnet werden. Im Beispiel in Abbildung 4.2 würde Subgraph B zunächst bei der Traversierung übersprungen werden. Erst bei Betrachtung von Subgraph A würde B gezeichnet werden.

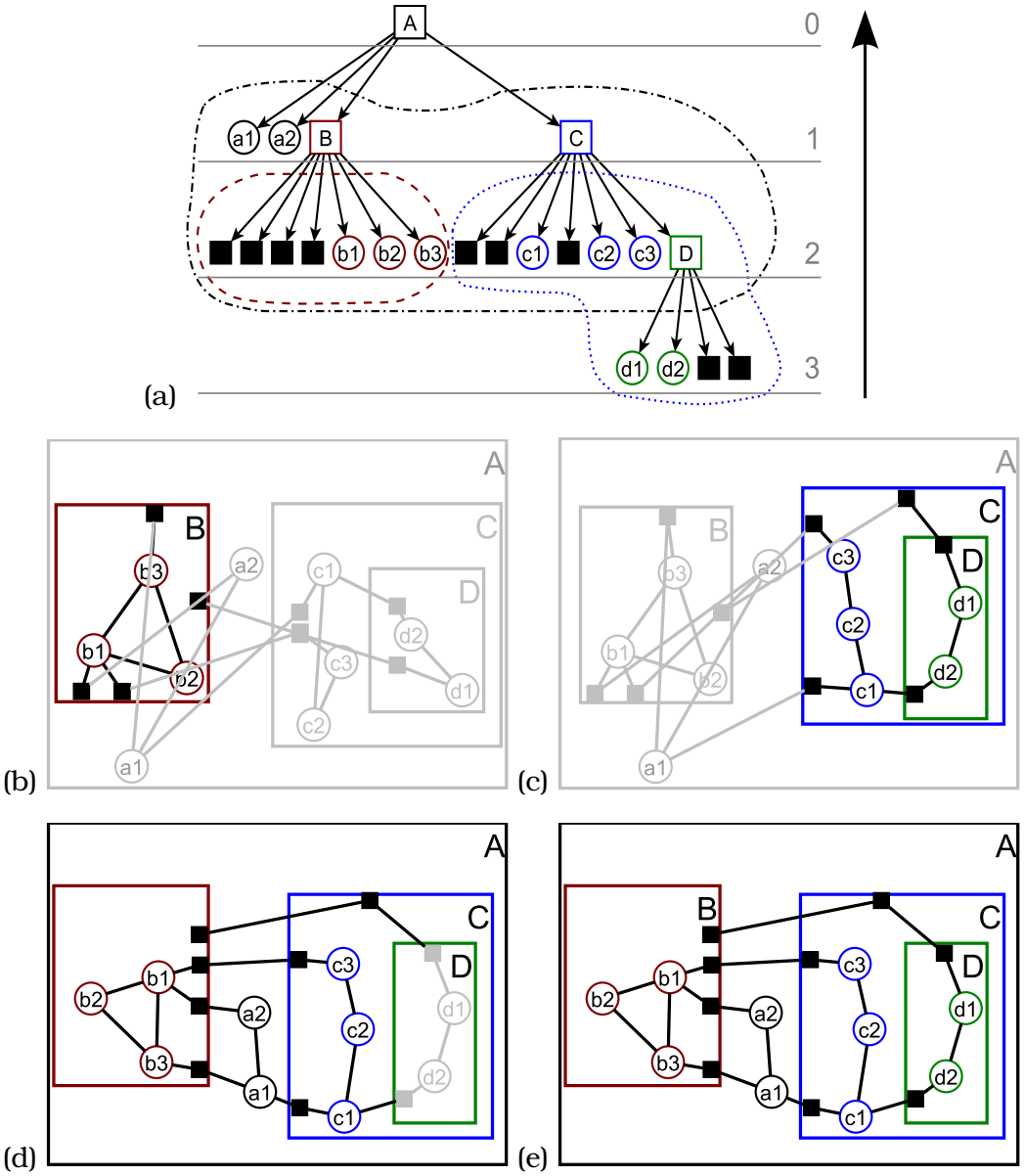


Abbildung 4.2: Funktionsweise von HyCoGLa: (a) Traversierung des Inklusionsbaums. (b) Beginnend in Ebene 1 wird zuerst Subgraph B (Knoten innerhalb gestrichelter Linie) gezeichnet. (c) Im nächsten Schritt wird dann Subgraph C zusammen mit Subgraph D (Knoten innerhalb gepunkteter Linie) gezeichnet. Hierbei werden die Positionen der Knoten aus D unter Beachtung der Beziehung zu den einfachen Knoten aus C berechnet. (d) Schließlich wird im letzten Schritt Subgraph A zusammen mit B und C (Knoten innerhalb gepunktstrichelter Linie) gezeichnet. Die Positionen der inneren Knoten der Subgraphen B und C werden hierbei unter Beachtung der einfachen Knoten aus A Layouts angepasst. Subgraph D wird im letzten Schritt nur als einfacher Kindknoten von C betracht und sein inneres Layout nicht mehr verändert. (e) Das finale Layout des Graphen.

4.2.1 Erweiterter Spring-Embedder

Der Erweiterte Spring-Embedder ist in Algorithmus 4.2 dargestellt. Er erhält als Eingabe einen Graphen mit eventuell vorhandenen Subgraphen. Für diesen Graphen wird dann das Layout in zwei Phasen, ähnlich wie bei dem in Abschnitt 3.1.2 vorgestellten Verfahren von [DGC+04] berechnet.

Algorithmus 4.2 ExtendedSpringEmbedder

Input: Graph $G(V,E)$ with (scaled) subgraphs
Output: A layout of G and its subgraphs

```
1:  phase ← 1
2:  while phase ≤ 2 do
3:    if phase < 2 then
4:      details ← details(graph)
5:      currentGraph ← skeleton(graph)
6:    else
7:      placeDetails()
8:      currentGraph ← graph
9:    end if
10:   initialize(currentGraph)
11:   i ← 0
12:   while i ≤ maxIterations and checkStoppingConditions() = false do
13:     calculateSpringForces()
14:     calculateRepulsiveForces()
15:     calculateGravitationalForces()
16:     if considerEdgeDirections then
17:       calculateMagneticForces()
18:     end if
19:     updateNodes()
20:     i ← i + 1
21:   end while
22:   phase ← phase + 1
23: end while
24: adjustTerminals()
25: removeNodeOverlappings()
```

In Phase 1 wird zunächst das Skelett des Graphen bestimmt, indem alle Knoten, die weniger als zwei Nachbarn besitzen (Detailknoten) temporär gelöscht werden und das Layout des Skeletts berechnet. Prinzipiell wäre hier auch eine andere Art der Skelettierung denkbar. Diese Variante ist jedoch einfach zu realisieren. Weiterhin stellt das Layout des so erzeugten Skeletts ein günstiges Ausgangslayout für den kompletten Graphen dar, da die entfernten Detailknoten das spätere Gesamtlayout nicht signifikant beeinflussen.

In Phase 2 wird basierend auf dem Layout des Skeletts das Layout des kompletten Graphen bestimmt. Dazu werden die in Phase 1 gelöschten Knoten an die Positionen ihrer im Skelett verbliebenen Nachbarknoten gesetzt und anschließend der komplette Graph gezeichnet.

Die Hauptarbeit des Erweiterten Spring-Embedders wird in den Zeilen 12-21 in Algorithmus 4.2 beschrieben. Zunächst werden die auf die Knoten wirkenden Kräfte berechnet. Eine detailierte Beschreibung hierzu findet sich im folgenden Abschnitt. Anschließend werden die Kräfte kombiniert und die Knoten in Richtung der Kräfte bewegt. Dies wird solange wiederholt, bis entweder die maximale Anzahl an Iterationen erreicht ist oder sich die Standardabweichung s der auf die Knoten wirkenden Kräfte x_i vom Idealwert 0 unter einem bestimmten Schwellwert befindet. Die Standardabweichung wird hierbei anstatt des Mittelwertes benutzt, da so Ausreißer unter den Knoten, auf die im Gegensatz zu anderen Knoten viel größere Kräfte wirken, besser ausgeglichen werden können. Sie wird mit folgender Formel berechnet:

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i)^2} \tag{4.1}$$

Nach dem alle Berechnungen zur Bestimmung des Layouts abgeschlossen sind, werden im letzten Schritt des Algorithmus die Positionen der Terminalknoten noch einmal verändert. Durch die Verbindungen der Terminalknoten zu anderen Subgraphen werden diese zwar nach außen gezogen, jedoch befinden sie sich nicht immer am Rand des Subgraphen. Daher werden diese in einem Nachbearbeitungsschritt am Rand des Subgraphen platziert. Eine genauere Beschreibung über die verschiedenen Platzierungsmöglichkeiten wird im weiteren Verlauf des Kapitels gegeben. Da das Spring-Embedder-Verfahren nur Knotenüberlappungen vermeiden, jedoch nicht komplett ausschließen kann, ist es sinnvoll, nach der Berechnung des Layouts in einem Nachbearbeitungsschritt zusätzlich eine Methode zur Entfernung von Knotenüberlappungen durchzuführen [LEN05].

4.2.1.1 Berechnung der Kräfte unter Beachtung der Subgraphen

In Abbildung 4.3 ist das Kräftemodell dargestellt, welches dem Erweiterten Spring-Embedder zugrunde liegt. Die verschiedenen auf die Knoten wirkenden Kräfte, werden, wie in den Algorithmen 4.3 bis 4.6 dargestellt, berechnet.

Algorithmus 4.3 beschreibt die Berechnung der auf die Knoten wirkenden Federkräfte. Im Gegensatz zum einfachen Spring-Embedder, bei dem die Länge d einer Kante *e(u,v)* gleich der Distanz zwischen den Knotenzentren $u.center$ und $v.center$ entspricht, wird die wahre Länge der Kante hierbei als Abstand zwischen den Ports der beiden Knoten bestimmt. Hierdurch werden die Knotengrößen in die Berechnung einbezogen und so Knotenüberlappungen vermieden [LEN05]. Um Knoten mit vielen Nachbarknoten mehr Platz zu verschaffen, wird die Federkraft zusätzlich umgekehrt proportional zur Anzahl der Nachbarknoten skaliert. Dies kann die Layoutqualität besonders bei Subgraphen mit hoher Konnektivität in einigen Fällen erhöhen. Da die Kanten zwischen den Knoten aus skalierten Subgraphen ebenfalls skaliert werden müssen, wird dies in die Berechnung der idealen Kantenlänge l mit einbezogen. Da ein einfaches Skalieren der Federlänge immer noch zu zu stark wirkenden (inneren) Kräften führt, ist es weiterhin notwendig, die berechneten Kräfte vor der Zuweisung zu skalieren (Zeilen 10 und 11).

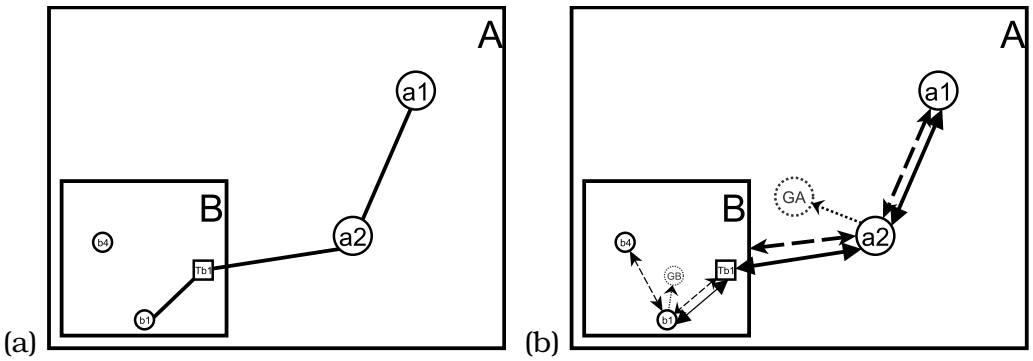


Abbildung 4.3: Kräftemodell des erweiterten Spring-Embedders: (a) Compound-Graph und (b) Darstellung der verwendeten Kräfte am Beispiel von Knoten a2 und b1. Dünne Linien bedeuten skalierte Kräfte. Federkräfte *FS* sind mit durchgezogener, Abstoßungskräfte *FR* mit gestrichelter und Gravitationskräfte *FG* mit gepunkteter Linie gekennzeichnet.

Algorithmus 4.3 calculateSpringForces

Input: Set of edges E of a graph $G(V,E)$
Output: A spring force for each node from V

for all $e = (u, v) \in E$ **do**
  $d \leftarrow$ distance($u.port$, $v.port$)
  if u is inner node and v is inner node **then**
    $l \leftarrow defaultSpringLength * scalingFactor$
  else
    $l \leftarrow defaultSpringLength$
  end if
  $spring.x \leftarrow k * \log(d/l) * (u.port.x - v.port.x)/d$
  $spring.y \leftarrow k * \log(d/l) * (u.port.y - v.port.y)/d$
  $FS(u) \leftarrow spring * u.scalingFactor$
  $FS(v) \leftarrow -spring * v.scalingFactor$
end for

Algorithmus 4.4 beschreibt die Berechnung der zwischen den Knoten wirkenden Abstoßungskräfte. Im Gegensatz zum einfachen Spring-Embedder, bei dem der ideale Abstand zwischen zwei Knoten gleich der Standard-Federlänge ist, muss hierzu ebenfalls die Größe und Skalierung der Knoten in die Berechnung des idealen Abstandes einbezogen werden. Daher wird der ideale Abstand k zwischen zwei Knoten u und v durch die Standard-Federlänge plus die Radien $u.radius$ und $v.radius$ der Umkreise der beiden Knoten berechnet. Zur Beschleunigung der Berechnung werden Abstoßungskräfte immer nur zwischen Knoten mit demselben Elternknoten bestimmt.

Algorithmus 4.4 calculateRepulsiveForces

Input: Set of nodes V of a graph $G(V,E)$
Output: A repulsion force for each node from V

```
1: isFinished ← {}
2: for all u ∈ V do
3:    isFinished add u
4:    for all v ∈ V − isFinished do
5:       if u.parent == v.parent then
6:          d ← distance(u.center, v.center)
7:          if d − u.radius − v.radius < repulsionRange then
8:             k ← defaultSpringLength ∗ u.scalingFactor + u.radius + v.radius
9:             repulsion.x ← e ∗ (k/d) ∗ (u.center.x − v.center.x)/d
10:            repulsion.y ← e ∗ (k/d) ∗ (u.center.y − v.center.y)/d
11:            FR(u) ← repulsion ∗ u.scalingFactor
12:            FR(v) ← −repulsion ∗ v.scalingFactor
13:         end if
14:      end if
15:   end for
16: end for
```

Die auf die Knoten wirkenden Gravitationskräfte dienen dazu, ein Auseinanderdriften nicht verbundener Teile des Graphen zu verhindern. Das heißt, die Knoten werden durch die Gravitationskräfte in Richtung des Massezentrums des Graphen bewegt. Weiterhin sorgen sie dafür, die inneren Knoten von Subgraphen zusammenzuhalten. Hierdurch wird die, für das innere Layout der Subgraphen, benötigte Fläche und damit die Fläche des gesamten Layouts des Graphen minimiert. Daher ist es vorteilhaft, den Parameter g für die auf die inneren Knoten wirkenden Gravitationskräfte zu verdoppeln. Algorithmus 4.5 zeigt die Berechnung der Gravitationskräfte. Der Punkt $globalGravityCenter$ beschreibt hierbei das Massenzentrum aller Kindknoten des aktuell betrachteten Subgraphen. Der Punkt $gravityCenter$ beschreibt das Massenzentrum aller Kindknoten eines im aktuell betrachteten Subgraphen enthaltenen Subgraphen.

Algorithmus 4.5 calculateGravitationalForces

Input: Set of nodes V of a graph $G(V,E)$
Output: A gravity force for each node from V

```
1: for all v ∈ V do
2:    if v is child of the parent graph then
3:       gravity.x ← (globalGravityCenter.x − v.center.x) ∗ g
4:       gravity.y ← (globalGravityCenter.y − v.center.y) ∗ g
5:    else
6:       gravity.x ← (v.parent.gravityCenter.x − v.center.x) ∗ g
7:       gravity.y ← (v.parent.gravityCenter.y − v.center.y) ∗ g
8:    end if
9:    FG(v) ← gravity ∗ v.scalingFactor
10: end for
```

4.2.1.2 Update der Knoten

Die für einfache äußere Knoten (z.B. Knoten a1 und a2 aus Abbildung 4.3(a)) berechneten Kräfte werden direkt benutzt, um deren neue Positionen zu bestimmen. Das Zusammenwirken der Kräfte wird in Gleichung 4.2 gezeigt. (Die Berechnung der magnetischen Kräfte *FM* wird in Abschnitt 4.2.2 erläutert.)

$$F(v) = FS(v) - FR(v) - FG(v) + FM(v) \tag{4.2}$$

Den inneren Knoten (z.B. Knoten b1, b2 und Tb1 aus Abbildung 4.3(a)) werden zusätzlich die auf ihren Elternknoten wirkenden Knotenabstoßungs- und Gravitationskräfte hinzugefügt. Hierbei ist zu beachten, dass die von den Elternknoten hinzugefügten Kräfte skaliert werden. Anschließend werden die Positionen der inneren Knoten ebenfalls aktualisiert. Nachdem alle inneren Knoten aktualisiert wurden, wird die neue Größe und Position des Elternknotens mit der *Bounding-Box* der inneren Knoten gleichgesetzt.

Die Aktualisierung der Knotenpositionen findet hierbei unter Nutzung des Konzepts der lokalen Temperatur [FLM94] statt. Hierdurch ist es möglich, Oszillationen von Knoten zwischen zwei optimalen Positionen zu verhindern, die Bewegung von Knoten in die richtige Richtung zu verstärken und somit die Berechnung des Layouts zu beschleunigen.

4.2.1.3 Behandlung der Terminalknoten

Die Positionierung der Terminalknoten kann auf verschiedene Arten erfolgen. Im Folgenden werden zwei mögliche Varianten vorgestellt. Die einfachste Art der Behandlung von Terminalknoten ist in Abbildung 4.4 dargestellt. Hierbei erfolgt die Positionierung des Terminalknotens am Schnittpunkt der Subgraphenbegrenzung mit der Interkante. Hierdurch können die Terminalknoten auch in Verfahren integriert werden, die auf allgemeinen Compound-Graphen basieren und Terminalknoten nicht in den Layoutprozess mit einbeziehen.

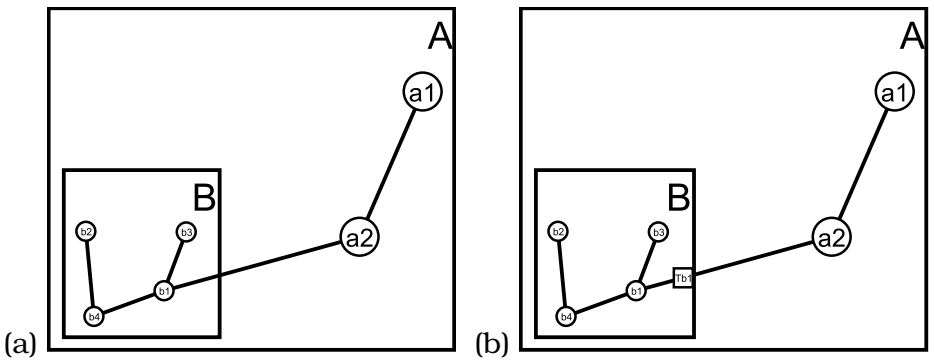


Abbildung 4.4: Variante 1 der Platzierung der Terminalknoten: (a) Comnpound-Graph ohne Terminalknoten, (b) Platzierung des Terminalknotens Tb1 an Schnittpunkt von Subgraph B und der Interkante zwischen a2 und b1.

Eine andere Variante ist es, die Terminalknoten, wie die anderen Knoten auch, als Teil des physikalischen Systems zu betrachten, sie also entsprechend der auf sie

wirkenden Kräfte zu bewegen und nach Abschluss der Layoutberechnung dann an den ihnen am nächstgelegenen Rand des Subgraphen zu platzieren. Diese Platzierung kann sowohl unter Nichtbeachtung als auch unter Beachtung der zuletzt auf den Terminalknoten wirkenden Kräfte geschehen. In Abbildung 4.5 ist diese Variante unter Nichtbeachtung der zuletzt auf den Terminalknoten wirkenden Kräfte dargestellt.

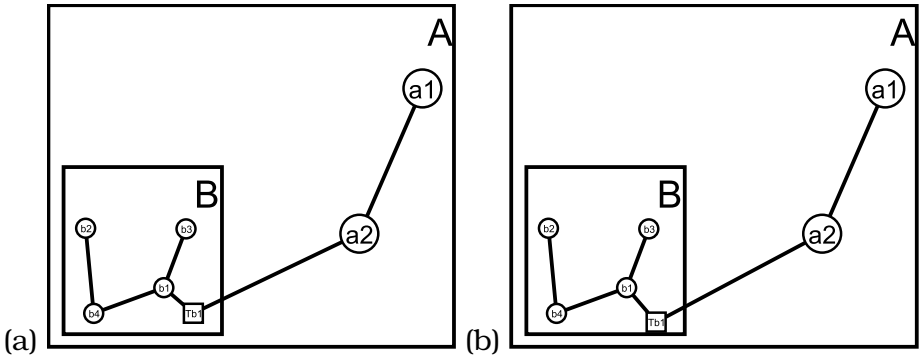


Abbildung 4.5: Variante 2 der Platzierung der Terminalknoten: (a) Platzierung des Terminalknotens Tb1 aufgrund der auf ihn wirkenden Kräfte, (b) Verschiebung von Tb1 an den ihm am nächstgelegenen Rand des Subgraphen.

In HyCoGLa wird Variante 2 zur Platzierung der Terminalknoten benutzt. Die Behandlung der Terminalknoten in Variante 2 ist zwar relativ einfach, der dort verwendete Ansatz ist jedoch ein guter Ausgangspunkt für weitere Untersuchungen zur optimalen Platzierung von Terminalknoten.

4.2.1.4 Behandlung mehrfach vorkommender Subgraphen

Bei den in der Systembiologie verwendeten Modellen kann es auftreten, dass die gleichen Subgraphen innerhalb eines Modells mehrfach verwendet werden. Des Weiteren wird die Wiederverwendung von Subgraphen/Modulen durch das Prinzip der modularen Modellierung explizit unterstützt. Zur Bestimmung des Layouts mehrfach vorkommender Subgraphen bestehen verschiedene Möglichkeiten. Im weiteren Verlauf werden diese Möglichkeiten vorgestellt und ihre Vor- und Nachteile diskutiert.

In Variante 1 (siehe Abbildung 4.6(a)) wird nur das Layout der ersten auftretenden Instanz eines Subgraphen berechnet. Den anderen Instanzen wird das berechnete Layout dieser ersten Instanz zugewiesen. Der Vorteil dieser Variante ist, dass das Layout nur für einen Subgraphen berechnet werden muss. Ein weiterer Vorteil ist, dass hierdurch alle Instanzen eines Subgraphen gleich dargestellt werden und damit mehrfach auftretende Muster oder Netzwerkmotive leichter entdeckt werden können. Der Nachteil dieser Variante ist, dass ähnlich wie bei lokalen Zeichenverfahren, das innere Layout der Subgraphen unabhängig voneinander bestimmt wird und somit unnötige Kantenkreuzungen und lange Kanten entstehen können, die die Qualität des Layouts verschlechtern.

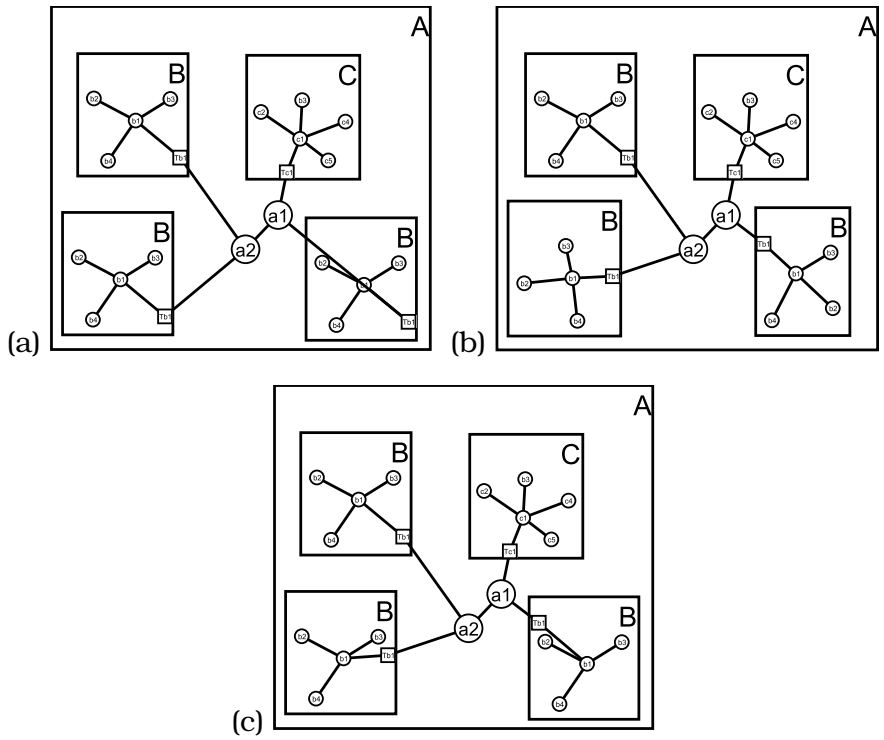


Abbildung 4.6: Verschiedene Möglichkeiten der Behandlung mehrfach vorkommender Subgraphen. (a) Variante 1: gleiches Layout aller Instanzen eines Subgraphen, (b) Variante 2: einzelnes Layout für jeden Subgraphen und (c) Variante 3: gleiches Layout aller Instanzen eines Subgraphen und individuelle Anpassung der Positionen der Terminalknoten.

Bei Variante 2 (Abbildung 4.6(b)) wird das Layout jeder Instanz eines Subgraphen entsprechend der wirkenden Kräfte individuell berechnet, wodurch eine qualitativ bessere Platzierung der Knoten und Kanten ermöglicht wird. Der Nachteil dieser Variante ist, dass sie das Finden von Netzwerkmotiven erschwert.

Variante 3 (Abbildung 4.6(c)) versucht die Vorteile der zuvor genannten Varianten zu verbinden. Wie bei Variante 1 wird hierbei zunächst nur das Layout der ersten vorkommenden Instanz eines Subgraphen berechnet. Dieses Layout erhalten dann ebenfalls die weiteren Instanzen des Subgraphen. Um die Probleme, die durch die Nichtbeachtung der globalen Verbindungen der einzelnen Subgraphen auftreten, zu verringern, werden die Positionen der Terminalknoten unter Beachtung der globalen Verbindungen angepasst. In Abbildung 4.6(c) ist das Ergebnis dieser Variante zu sehen. Die Qualität des globalen Layouts des Graphen ist etwas besser als in Abbildung 4.6(a), da die Interkanten durch die Platzierung der Terminalknoten nicht unnötig lang sind. Im Vergleich zu Abbildung 4.6(b) ist die Gleichheit der Subgraphen jedoch einfacher zu erkennen.

In der Implementierung von HyCoGLa wird die erste Variante zum Layout von Instanzen des gleichen Subgraphen benutzt.

4.2.1.5 Beachtung der Lesbarkeit von Labels

Zur Beschriftung von Knoten werden in vielen Anwendungen Labels benutzt. Die Lesbarkeit dieser Labels ist wichtig für eine verständliche Darstellung des gesamten Graphen. Aus diesem Grund ist darauf zu achten, dass Labels und Knoten so platziert werden, dass sie sich nicht überlappen.

In Anwendungen, in denen die Labels fest mit den zu beschriftenden Knoten verbunden sind, ist es sinnvoll, zur Vermeidung von Überlappungen von Knoten und Labels, anstatt der wahren Knotengröße, die Größe der Bounding-Box, welche den Knoten und das Label beinhaltet, als Berechnungsgrundlage zu wählen. Als Länge der Kante, wird dann der Abstand der Schnittpunkte der Kante, mit den Bounding-Boxen ihrer beiden Endknoten, betrachtet. Die Berechnung der Abstoßungs- und Gravitationskräfte sollte ebenfalls analog dazu, unter Beachtung der Bounding-Box, berechnet werden. Abbildung 4.7(b) zeigt das Ergebnis dieser Methode zur Beachtung von Labels. Können Labels frei um den zu beschriftenden Knoten platziert werden, sind erweiterte Platzierungsstrategien für die Labels, wie beispielweise die in [DKMT07] beschriebene Technik, notwendig.

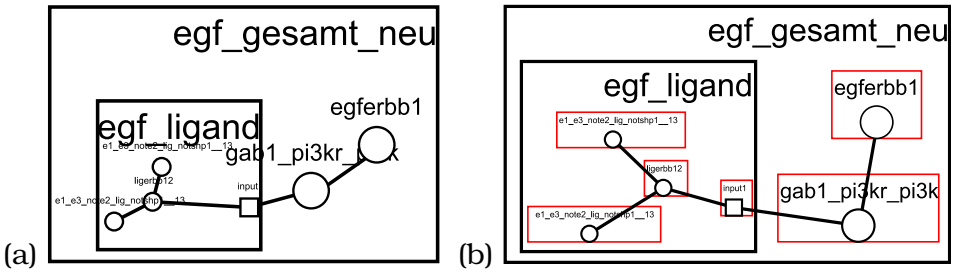


Abbildung 4.7: Beachtung der Lesbarkeit von Labels: (a) Nichtbeachtung der Labels und (b) die Bounding-Box von Knoten und Label (rot markiert) wird als Grundlage zur Berechnung genutzt, um Überlappungen von Knoten und Labels zu vermeiden.

4.2.2 Integration semantischer Informationen

Durch das bisher beschriebene Verfahren wurden nur die in Abschnitt 4.1.1 unter den Punkten 1 bis 5 genannten syntaktischen Anforderungen an das Layout ungerichteter Compound-Graphen optimiert. Abbildungen 4.8 und 4.9 zeigen zwei Graphen, deren Layouts unter Beachtung dieser Anforderungen erzeugt wurde. Bei beiden Darstellungen ist sowohl die globale Struktur des Graphen als auch die innere Struktur der Subgraphen gut erkennbar.

Dar in Abbildung 4.8 abgebildete Graph stellt ein Realmodell aus der Systembiologie dar. Diesem können ohne Weiteres keine semantischen Informationen, wie z.B. definierte Reaktionsrichtungen, zugeordnet werden, da diese entweder reversibel oder unbekannt sind. In Abbildung 4.9 ist der Graph eines Testmodells dargstellt. Bei diesem Modell sind die Reaktionsrichtungen gut ablesbar und es liegen zusätzlich zu den strukturellen Informationen semantische Informationen vor. An diesem Modell wird daher im weiteren Verlauf des Kapitels gezeigt, wie die Beachtung

semantischer Anforderungen (siehe Abschnitt 4.1.1, Punkt 6) an die Darstellung biologischer Netzwerke, in die Layoutberechnung von HyCoGLa, integriert werden können. Zu diesen semantischen Anforderungen gehören beispielsweise die Darstellung von Reaktionsrichtungen, die Positionierung bestimmter Knoten zueinander und die subzelluläre Lokalisation von Komponenten.

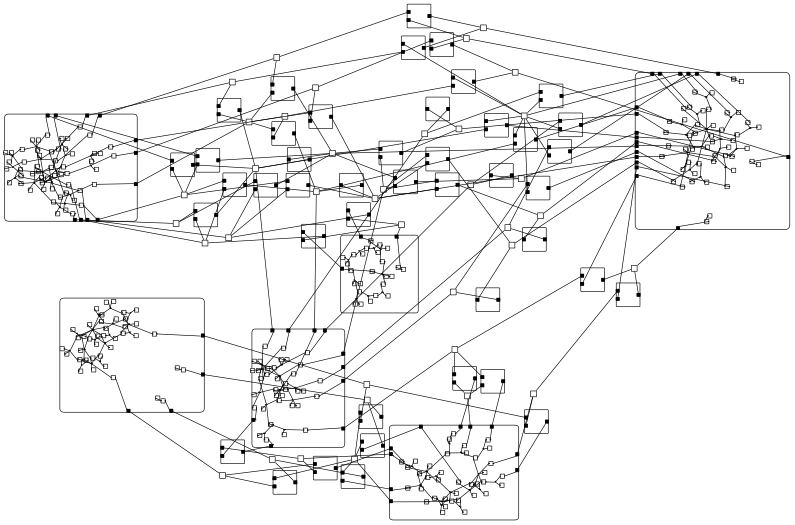


Abbildung 4.8: Dasselbe Modell wie in Abbildung 4.1. Das Layout wurde mit HyCoGLa erzeugt.

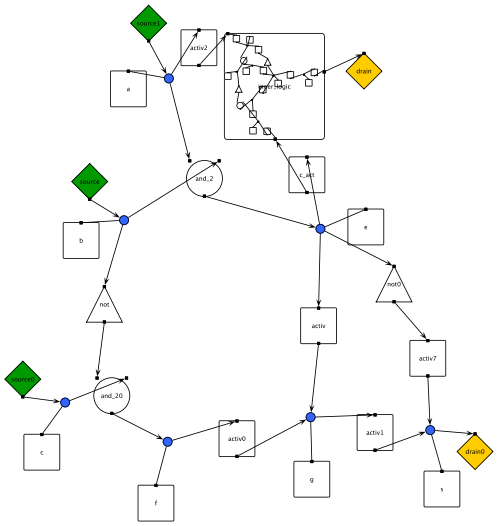


Abbildung 4.9: Mit HyCoGLa erstelltes Layout eines Testmodells. Bei diesem Modell sind semantischen Informationen vorhandenen, die in Abbildung 4.10 in das Layout integriert werden.

4.2.2.1 Reaktionsrichtungen

Die erste semantische Anforderung ist die Darstellung von Reaktionsrichtungen. Diese sollten, wie in Abschnitt 2.2.2 erläutert, in eine Richtung gezeichnet werden, um den zeitlichen Ablauf aufeinanderfolgender Prozesse darzustellen. Hierdurch

wird das Finden und das Verfolgen des Verlaufes von Reaktionspfaden vereinfacht. Dies ermöglicht es Biologen, wichtige Zusammenhänge zwischen den Komponenten eines biologischen Netzwerks zu erkennen. Die Ausrichtung der Kanten kann in HyCoGLa mit Hilfe der in Abschnitt 3.2.1 erläuterten magnetischen Kräfte erfolgen. Algorithmus 4.6 beschreibt die Berechnung der auf die Knoten wirkenden magnetischen Kräfte. Die Stärke der auf die Knoten wirkenden magnetischen Kräfte ist abhängig von der Länge d der Kante und dem Winkel ϕ zwischen dem Magnetfeld und der Kante. In HyCoGLa wird nur ein Magnetfeld mit der Richtung $(0, 1)$ eingesetzt, um die Kanten von oben nach unten auszurichten. Prinzipiell ist es jedoch auch möglich, die Kanten mit verschiedenen Richtungstypen unterschiedlich (z.B. von oben nach unten oder von links nach rechts) auszurichten, indem jeweils ein entsprechend ausgerichtetes Magnetfeld eingesetzt wird.

Algorithmus 4.6 calculateMagneticForces

Input: Set of edges E of a graph $G(V,E)$
Output: A magnetic force for each node from V

1: $magneticFieldDirection \leftarrow (0, 1)$
2: **for all** $e = (u, v) \in E$ **do**
3: $d \leftarrow$ distance($u.port$, $v.port$)
4: **if** edge direction is from u to v **then**
5: $directedEdge \leftarrow (u.x - v.x, u.y - v.y)$
6: **else**
7: $directedEdge \leftarrow (v.x - u.x, v.y - u.y)$
8: **end if**
9: $n \leftarrow$ normal($directedEdge$)
10: $\phi \leftarrow$ angle($directedEdge$, $magneticFieldDirection$)
11: $magnetic.x \leftarrow m * d^{\alpha} * \phi^{\beta} * normal.x$
12: $magnetic.y \leftarrow m * d^{\alpha} * \phi^{\beta} * normal.y$
13: $FM(u) \leftarrow magnetic * u.scalingFactor$
14: $FM(v) \leftarrow -magnetic * v.scalingFactor$
15: **end for**

In Abbildung 4.10(a) ist das Layout des Modells aus Abbildung 4.9, berechnet unter zusätzlicher Nutzung der magnetischen Kräfte zur Ausrichtung der Kanten, dargestellt. Die Richtungen der Kanten sind durch Pfeile gekennzeichnet. Der Verlauf der Reaktionspfade, beginnend an den gelb gezeichneten *Input*-Knoten im oberen Teil des Netzwerks zu den grün gezeichneten *Output*-Knoten im unteren Teil des Netzwerks, ist gut zu erkennen.

4.2.2.2 Positionierung bestimmter Knoten zueinander

Sollen ausgewählte Knoten in einer bestimmten Lage zueinander positioniert werden, so ist dies durch Formulierung von Constraints umsetzbar. Die Beachtung der Kantenrichtung ist in diesem Sinne ein Beispiel für solch ein Constraint, welches jedoch nur zwischen adjazenten Knoten wirkt. In [RMS97, WM95] wird gezeigt wie verschiedene Constraints in ein kräftebasiertes Verfahren wie HyCoGLa integriert werden können. Das Problem bei der Verwendung solcher Constraints ist jedoch, dass diese sich gegenseitig und auch die Umsetzung der grundlegenden Anforde-

rungen (siehe Abschnitt 4.1.1) behindern können und sich somit negativ auf die Qualität des Layouts auswirken. Daher sollten diese nur dann benutzt werden, wenn durch sie keine anderen Anforderungen verletzt werden.

Ein Beispiel für ein relativ einfach umzusetzendes Constraint ist die Platzierung von bestimmtem Knotenpaaren in einer festen Position zueinander, wie z.B. die Platzierung beider Knoten auf der gleichen Position. In HyCoGLa kann dies relativ einfach umgesetzt werden, indem diese Knotenpaare als ein einzelner Knoten betrachtet werden. Der durch diese Platzierungsstrategie enstehende Vorteil ist zum einen, die geringere visuelle Komplexität des Layouts. Zum anderen verringert sich der Aufwand der Berechnung des Layouts, da anstelle von beiden Knoten nur ein Knoten in die Berechnung mit einbezogen werden muss und dem anderen Knoten dann die gemeinsame Position zugewiesen werden kann.

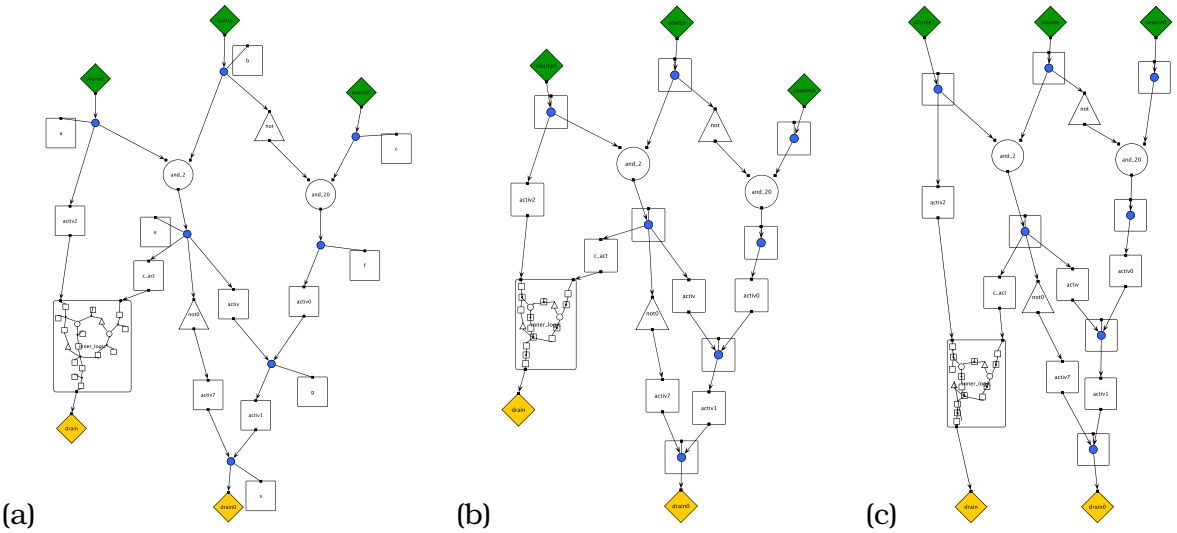


Abbildung 4.10: Das Modell aus Abbildung 4.9. Diesmal wurden in das Layout jedoch verschiedene semantische Informationen integriert: (a) Kantenrichtungen, (b) Kantenrichtungen und Positionierung bestimmter Knoten aufeinander und (c) Kantenrichtungen, Positionierung bestimmter Knoten aufeinander und Beachtung der Lokalisation.

4.2.2.3 Subzelluläre Lokalisation

Lassen sich die einzelnen Komponenten des Netzwerks zu zellulären Kompartimenten oder verschiedenen systemtheoretischen Schichten (z.B. Eingabe-, Zwischen- und Ausgabeschicht) zuordnen, ist es für das Verständnis des Netzwerks hilfreich, die Anordnung der Komponenten innerhalb dieser Schichten darzustellen (siehe Abbildung 2.10). Dies kann dadurch geschehen, dass die subzelluläre Lokalisation der Knoten mit Hilfe der in Abschnitt 3.2.2 beschriebenen Techniken in HyCoGLa integriert wird.

Eine weitere Möglichkeit, die Knoten bestimmten Schichten zuzuordnen, stellt die Interaktion durch den Anwender dar. Wie in Abbildung 4.10(c) dargestellt, sind die

Knoten drei Schichten zugeordnet. Die Zuordnung der Knoten zur Eingabe- (grünen Knoten) bzw. Ausgabeschicht (gelbe Knoten) wurde hierbei auf Basis des zuvor berechneten Layouts unter Beachtung der Kantenrichtungen durch den Anwender durchgeführt. Dies kann komplett manuell oder mit Unterstützung einer automatischen Anordnungsfunktion geschehen. Die Knoten in der Zwischenschicht werden dann unter Beachtung der manuell gesetzten Positionen der Knoten aus der Eingabe und Ausgabeschicht angepasst. Die Beachtung manuell positionierter Knoten ist in Abschnitt 4.3.1 beschrieben.

4.3 Benutzerinteraktion bei automatischen Zeichenverfahren

Ein Zeichenverfahren wird dann als automatisch bezeichnet, wenn es das Layout eines Graphen ohne Eingriff des Benutzers, abgesehen von der Auswahl des zu verwendenden Zeichenverfahrens, komplett selbstständig erledigt. In manchen Fällen wird sogar die Auswahl des optimalen Zeichenverfahrens automatisch durchgeführt. Die meisten verwendeten Zeichenverfahren sind jedoch semiautomatisch, da sie dem Anwender Möglichkeiten bieten, das Ergebnis des Zeichenverfahrens interaktiv zu beeinflussen. Dies kann beispielsweise durch Auswahl von Parametern oder durch ein manuelles Layout einzelner Teile des Graphen geschehen. Im Folgenden wird daher beschrieben, wie diese Interaktion des Benutzers mit dem Zeichenverfahren unterstützt werden kann.

4.3.1 Beachtung manuell positionierter Knoten

Eine Möglichkeit des Eingriffs in das Zeichenverfahren durch den Benutzer ist, dass dieser die Positionierung von Knoten per Hand, vor oder nach dem eigentlichen Zeichenverfahren, zur Verbesserung des Layouts durchführt. Bei iterativen Verfahren wie dem Spring-Embedder ist ein Eingriff des Anwenders sogar während der Berechnung des Layouts möglich.

Werden Knoten durch den Anwender vor der Ausführung des Zeichenverfahrens platziert, so soll die Positionierung dieser Knoten in der Regel nicht oder nur minimal durch das Zeichenverfahren geändert werden. Um die Neupositionierung ausgewählter Knoten komplett zu verhindern, gibt es bei der Verwendung eines kräftebasierten Zeichenverfahrens im Prinzip zwei Alternativen. Zum einen können diese Knoten aus der Berechnung der Kräfte ausgeschlossen werden, was jedoch dazu führen würde, dass es bei der Platzierung der nicht manuell positionierten Knoten zu Überlappungen oder ungünstigen Positionierungen zwischen ihnen und den nicht manuell platzierten Knoten kommen kann (siehe Abbildung 4.11(b)). Die andere Möglichkeit ist, die Knoten, deren Position durch das Zeichenverfahren nicht mehr verändert werden soll „festzunageln“ und sie trotzdem in die Berechnung der Kräfte mit einzubeziehen (siehe Abbildung 4.11(c)). Die für die „festgenagelten“ Knoten berechneten Kräfte können dann entweder vernachlässigt oder zu den anderen Knoten hinzugefügt werden.

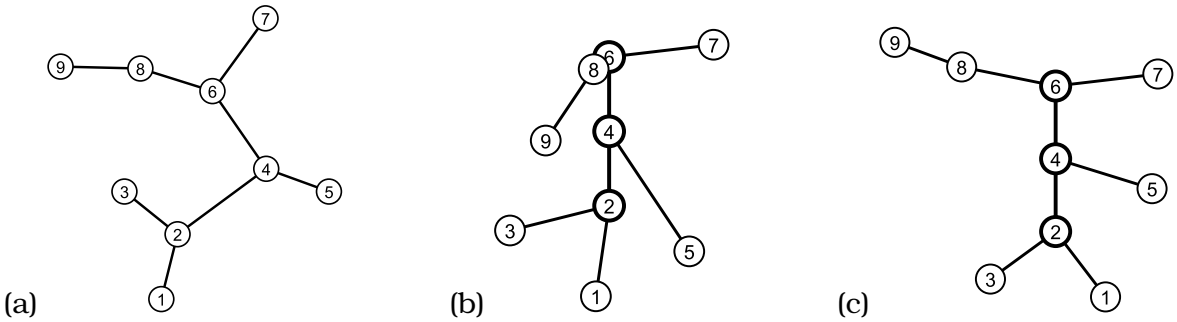


Abbildung 4.11: Manuelles Layout von Knoten: (a) Ausgangslayout, (b) Überlappungen von Knoten durch Nichtbeachtung von manuell positionierten Knoten (2, 4 und 6) während einer erneuten Layoutberechnung und (c) „Festnageln“ von Knoten (2, 4 und 6) und Beachtung dieser während der Berechnung des Layouts.

Sollen die Positionen der Knoten durch das Layoutverfahren nur minimal verändert werden, so können diese, wie in Abbildung 4.12 dargestellt, mit virtuellen Knoten, welche an die Ausgangspositionen der Knoten eingefügt und über virtuelle Kanten mit den Knoten verbunden werden, in der Nähe ihrer Ausgangspositionen gehalten werden. Diese Option wäre beispielsweise nach dem Ändern eines Modells durch Hinzufügen oder Entfernen von Knoten geeignet, um eine zu starke Änderung des Layouts nach einer erneuten Ausführung des Zeichenverfahrens zu verändern und so die Mental Map des Anwenders zu bewahren.

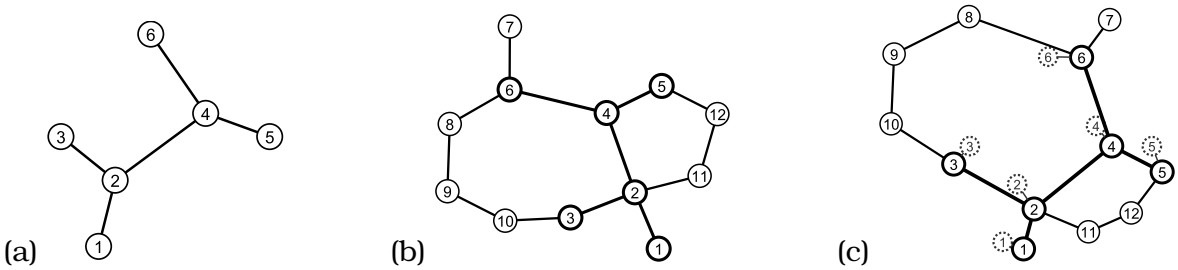


Abbildung 4.12: Bewahrung des ursprünglichen Layouts durch virtuelle Knoten: (a) Ausgangslayout, (b) Layout des Graphen nach dem Einfügen neuer Knoten und (c) erneutes Layout unter Beachtung der alten Knotenpositionen durch die grau, gepunktet dargestellten virtuelle Knoten. Ursprüngliche Kanten und Knoten sind fett dargestellt.

4.3.2 Parameterauswahl

Wie in der Anforderungsanalyse in Abschnitt 4.1.1 erläutert, unterscheiden sich biologische Netzwerke in vielen Faktoren voneinander. Des Weiteren existieren von Anwender zu Anwender verschiedene Anforderungen an das Layout. Um eine geeignete Darstellung eines biologischen Netzwerks zu erhalten, ist es daher notwendig, dem Anwender die Parametrisierung des Zeichenverfahrens zu ermöglichen. Zur Beeinflussung des Layouts existieren in HyCoGLa folgende Parameter und Einstellungsmöglichkeiten:

- Die *Kantenlänge*, die den idealen Knotenabstand bestimmt.

- Die *Federhärte*, die die Stärke der Federkraft bestimmt.
- Die *elektrische Abstoßung*, die die Stärke der Abstoßung zwischen zwei Knoten bestimmt.
- Der *Bereich der Abstoßungskräfte* definiert den maximalen Abstand von Knoten bis zu dem Abstoßungskräfte berechnet werden. Dieser Parameter hilft, unnötige Berechnungen der Abstoßungskräfte zwischen weit entfernten Knoten einzusparen.
- Die *äußere Gravitationsstärke*, die bestimmt wie stark die auf die äußeren Knoten wirkende Gravitationskraft ist.
- Die *innere Gravitationsstärke*, die bestimmt wie stark die auf die inneren Knoten wirkende Gravitationskraft ist.
- Die *Stärke des magnetischen Feldes* zur Bestimmung der Stärke der magnetischen Kräfte.
- Die *Auswirkung der Kantenlänge* α und die *Auswirkung des Winkels zwischen Kante und Magnetfeld* β auf die Stärke der magnetischen Kräfte.
- Die *Richtung des magnetischen Feldes*, die die Ausrichtung der Kanten bestimmt.
- Die *maximale Anzahl an Iterationen* bis zum Abbruch der Berechnung.
- Der *Schwellwert*, den die auf die Knoten wirkenden Kräfte unterschreiten müssen, damit die Berechnung abgebrochen wird.
- Die *initiale Temperatur*, die die Stärke der Bewegung jedes Knotens pro Iteration beeinflusst.
- Die *Lesbarkeit von Labels*, bewirkt die Berechnung des Layouts unter der in Abschnitt 4.2.1.5 beschriebenen Beachtung der Lesbarkeit von Labels.
- Das *Layout selektierter Knoten* bewirkt, dass das Layout nur auf die aktuell selektierten Knoten angewendet wird (siehe Abschnitt 4.3.1).
- Die *Platzierung von bestimmten Knotenpaaren zueinander* bewirkt, dass bestimmte Knotenpaare, wie in Abschnitt 4.2.2 erläutert, platziert werden.
- Die *Verwendung eines Zufallslayouts als Ausgangslayout* bewirkt, dass vor der eigentlichen Berechnung des Layouts dem Graphen ein Zufallslayout zugewiesen wird. Dies kann sinnvoll sein, um ein geeignetes Ausgangslayout zu erzeugen, wenn z.B. beim Ausgangslayout alle Knoten im Ursprung angeordnet sind, was laufzeittechnisch ungünstig ist. Existiert bereits ein Layout, verhindert die Verwendung eines Zufallslayouts als Ausgangslayout jedoch die Wahrung der Mental Map, da das eigentliche Ausgangslayout nicht beachtet wird.

Die manuelle Einstellung aller Parameter durch den Anwender kann aufgrund der großen Anzahl an Parametern und der daraus resultierenden Kombinationsmöglichkeiten eine sehr aufwändige Aufgabe sein. Des Weiteren besitzt nicht jeder Anwender die gleiche Erfahrung in der Bedienung eines Zeichenverfahrens. Aus diesem Grund ist es notwendig, den Anwender bei der Parametrisierung zu unterstützen. Folgende Punkte sollten hierzu beachtet werden:

- *Sinnvolle Standardwerte und Wertebereiche:* Die Standardwerte und Wertebereiche nummerischer Parameter wie der Kantenlänge oder der maximalen Anzahl an Iterationen sollten sinnvoll ausgewählt werden. Zur Präsentation in einem Benutzerdialog eignet sich beispielsweise die Darstellung durch Slider. In den meisten Fällen sind die konkreten Werte der Parameter sehr unterschiedlich und für den Anwender nicht sofort interpretierbar. Daher ist es sinnvoll, zur Vereinheitlichung bei der Beschriftung der Slider anstatt der konkreten Minimal- und Maximalwerte, normierte Werte (z.B. 0 und 1) oder Beschriftungen wie "-" und "+" zu verwenden.

- *Intuitive Benennung der Parameter:* Das Konzept kräftebasierter Zeichenverfahren ist zwar intuitiv, dennoch sind der Zweck und die Auswirkungen von Parametern wie "Federhärte", "Gravitationsstärke" und "Stärke des magnetischen Feldes", besonders aus Sicht von unerfahrenen Anwendern, nicht immer direkt ersichtlich. Daher ist es sinnvoll, diese Parameter so zu benennen, dass dem Anwender sofort klar wird, was diese bewirken. Die Bezeichnung "Stärke des magnetischen Feldes" könnte beispielsweise mit '"Hervorhebung von Kantenrichtungen" ersetzt werden.

- *Gruppierung von Parametern:* Die Parameter einen Zeichenverfahrens lassen sich nach ihrem Zweck ordnen. So dienen Parameter wie die Kantenlänge, die Gravitationsstärke oder die Richtung des magnetischen Feldes zur Beeinflussung der visuellen Eigenschaften des Layouts und Parameter wie die maximale Anzahl an Iterationen, der der Schwellwert zum Abbruch oder die initiale Temperatur zur Beeinflussung der Terminierung des Zeichenverfahrens. Daher ist es sinnvoll, die Parameter nach ihrem Zweck geordnet darzustellen.

- *Kombination von Parametern:* Besitzen Parameter eine gleiche oder ähnlich Wirkung, so können diese zur Vereinfachung zusammengefasst werden. Die Parameter Kantenlänge, Federhärte und elektrische Abstoßung können z.B. zu einem Parameter *Skalierung des Layouts* zusammengefasst werden, da sie alle die Größe des Layouts, d.h. die durch das Layout eingenommene Fläche beinflussen.

- *Abgestufte Einstellungen:* Aufgrund der unterschiedlichen Erfahrung von Anwendern in der Bedienung von Zeichenverfahren ist es sinnvoll, die Einstellungsmöglichkeiten in verschiedene Stufen zu unterteilen. Unerfahrenen Anwendern sollten hierbei nur die wichtigsten Parameter, wie z.B. Skalierung des Layouts, Hervorhebung von Kantenrichtungen, die Beachtung von Labels oder die Verwendung eines Zufallslayouts als Ausgangslayout präsentiert werden. Die restlichen Parameter sollten dem Anwender verborgen und durch sinnvolle Standardwerte belegt werden. Falls möglich könnten die Standardwerte auch unter Beachtung der Eigenschaften des aktuellen Graphen angepasst werden.

Erfahrenen Anwendern sollten alle verfügbaren Einstellungen präsentiert werden, damit diese in der Lage sind, das Zeichenverfahren so genau wie möglich an ihre Anforderungen und die Eigenschaften des zu zeichnenden Graphen anzupassen.

4.4 Zusammenfassung

Zu Beginn dieses Kapitels wurden die Anforderungen an ein Verfahren zum Layout von hierarchisch strukturierten Modellen aus der Systembiologie, welche durch Compound-Graphen repräsentiert werden, erläutert. Hieraus folgte, dass neben der Beachtung allgemeiner Anforderungen an die Darstellung von Compound-Graphen, auch die Beachtung von speziellen Anforderungen an die Darstellung biologischer Netzwerke notwendig ist. Weiterhin wurde festgestellt, dass aufgrund der großen Unterschiede systembiologischer Modelle (in Größe, Strukturierung und Vorhandensein von semantischen Informationen) Interaktionsmöglichkeiten für den Benutzer zur Anpassung des Zeichenverfahrens notwendig sind.

Im Anschluss daran wurde das Konzept von HyCoGLa, eines neuartigen Spring-Embedder-basierten Verfahrens zum Zeichnen von Compound-Graphen vorgestellt. HyCoGLa ist eine Mischform existierender lokaler und globaler Techniken zum Zeichnen von ungerichteten Compound-Graphen. Es nutzt dabei die Eigenschaften des Terminal-Cluster-Graphen, eines speziellen Compound-Graphen, bei dem Interkanten immer über Terminalknoten verlaufen. Hierdurch wird es möglich, globale Zusammenhänge beim Zeichnen zu betrachten ohne dafür alle Knoten und Kanten des Compound-Graphen auf einmal betrachten zu müssen. Daher sollte es prinzipiell möglich sein, den Trade-off zwischen existierenden Techniken (lokal: schnell, schlechte Qualität und global: langsam, gute Qualität) zu verbessern. Es wurden verschiedene Ansätze zur Platzierung von Terminalknoten und dem Umgang mit mehrfach vorkommenden Subgraphen erläutert, sowie auf die Beachtung der Lesbarkeit von Labels eingegangen. Des Weiteren wurden Techniken zur Integration vorhandener semantischer Informationen in HyCoGLa beschrieben, um die besonderen Anforderungen an das Layout biologischer Netzwerke umzusetzen.

Das Kapitel endete mit einer Betrachtung der Unterstützung möglicher Interaktionen von Anwendern mit automatischen, kräftebasierten Zeichenverfahren. Die erste Interaktionsmöglichkeit, bei der Anwender unterstützt werden können, ist die Auswahl der Parameter des Zeichenverfahrens, welche aufgrund der typischerweise hohen Anzahl an Paramtern, sehr komplex ist. Weiterhin wurden beschrieben, wie durch den Anwender manuell positionierte Knoten, durch das Zeichenverfahren berücksichtigt werden können.

5 Implementierung

Dieses Kapitel befasst sich mit der Integration des entwickelten Zeichenverfahrens in das Modellierungswerkzeug ProMoT. Im ersten Teil dieses Kapitels werden zunächst ProMoT und die Technologien vorgestellt, auf deren Basis die Implementierung erfolgte. Im zweiten Teil wird die eigentliche Implementierung, inklusive der durch ProMoT bedingten Anpassungen, beschrieben.

5.1 ProMoT

ProMoT ist ein frei verfügbares Programm[1] zur Modellierung und Visualisierung komplexer technischer und biologischer Systeme. Zum Einsatz im Bereich der Systembiologie, ist es aufgrund folgender Eigenschaften besonders geeignet:

- Unterstützung modularer Modellierung,
- Verwendung spezialisierter Bibliotheken zur Modellierung und
- Unterstützung des Modellaustauschformats SBML.

ProMoT besteht aus zwei Hauptkomponenten: dem in Lisp programmierten *Kern* und der in Java programmierten Benutzeroberfläche (*GUI*). Die Kommunikation zwischen Kern und GUI wird durch *CORBA* ermöglicht. Der Kern dient zum Lesen, Speichern und Verwalten der Modelle. Die GUI beinhaltet u.a. Komponenten zum visuellen Erstellen, Editieren (Visual Editor) und Explorieren (Visual Explorer) von Modellen. Eine ausführlichere Beschreibung von ProMoT ist in [GKN+03, SRMH+06] zu finden.

5.1.1 Visual Editor

Der Visual Editor wird zum Erstellen und Editieren von Modellen benutzt. In Abbildung 5.1 ist ein hierarchisch strukturiertes Modell des EGF-Rezeptor-Netzwerks im Visual Editor dargestellt. Im Editierbereich des Visual Editor wird immer nur die oberste Hierarchieebene eines Moduls (Subgraphen) eines hierarchisch strukturierten Modells (Compound-Graphen) dargestellt. Für die Betrachtung des Inhalts eines Submoduls eines Moduls, ist es notwendig, dieses in einem neuen Visual Editor zu öffnen. Submodule werden im Visual Editor immer um einen konstanten Faktor skaliert dargestellt, um den Platz auf dem Bildschirm optimal auszunutzen.

1 ProMoT ist erhältlich unter: http://www.mpi-magdeburg.mpg.de/projects/promot/

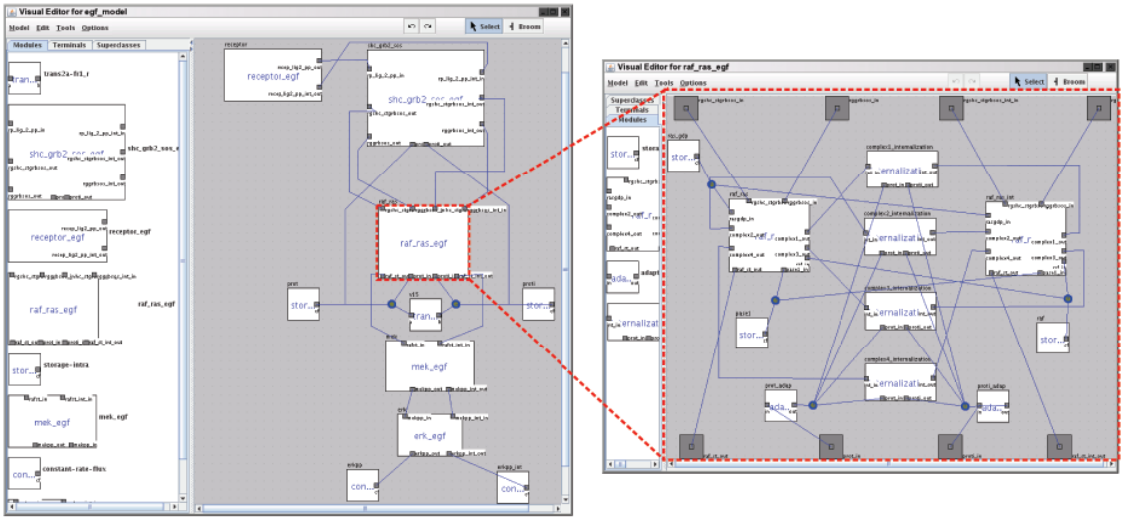


Abbildung 5.1: Darstellung der obersten Hierarchieebene eines Modells des EGF-Rezeptor-Netzwerks im VISUAL EDITOR (linke Abbildung): Der Editierbereich ist die graue Fläche, in der das Modell dargestellt ist. Im linken Bereich des VISUAL EDITOR befindet sich die Komponentenpalette. In der rechten Abbildung ist die Darstellung eines geöffneten Submoduls des Modells in einem weiteren VISUAL EDITOR abgebildet.

Der VISUAL EDITOR basiert auf dem *Graph Editing Framework* (GEF) [Tig08] und nutzt dessen Funktionalität zur graphischen Repräsentation der Modelle durch Graphen. Abbildung 5.2 stellt die wichtigsten Komponenten des VISUAL EDITOR und den Zusammenhang zwischen der logischen und graphischen Repräsentation des Graphen, eines im VISUAL EDITOR geöffneten Modells, vereinfacht dar. Das Haupt-

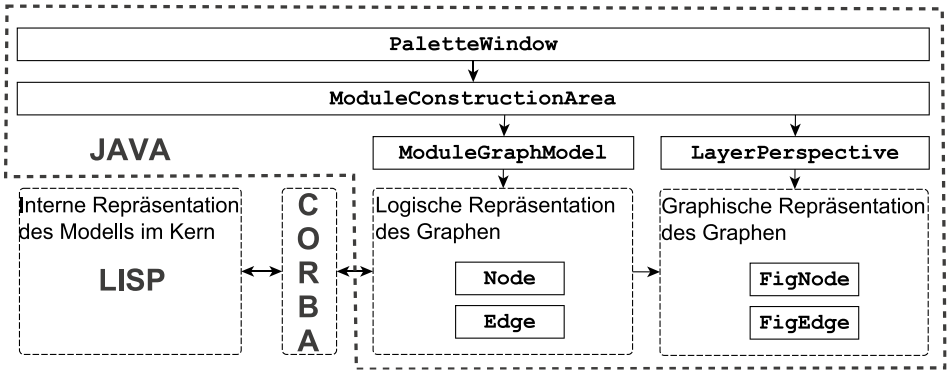


Abbildung 5.2: Vereinfachte Darstellung des Aufbaus des VISUAL EDITOR und des Zusammenhangs zwischen logischer und graphischer Repräsentation des Graphen, eines im VISUAL EDITOR geöffneten Modells.

fenster des VISUAL EDITOR ist in der Klasse `PaletteWindow` implementiert. Dieses beinhaltet im oberen Bereich Menü und Toolbar und im linken Bereich die Komponentenpalette, in der die Klassen aller im aktuell geöffneten Modul enhaltenen Submodule und Komponenten aufgelistet sind (siehe Abbildung 5.1). Der Editierbereich nimmt den größten Bereich des VISUAL EDITOR ein. Dieser ist in der Klasse `ModuleConstructionArea` implementiert. Die Klasse `ModuleConstructionArea` verwaltet das `ModuleGraphModel`, welches den aktuell dargestellten Graphen und

seine Elemente (`Node`, `Edge`) beschreibt. Des Weiteren verwaltet sie den Layer (Klasse `LayerPerspective`), auf dem sich die graphischen Repräsentationen der Elemente des Graphen (`FigNode`, `FigEdge`) befinden. Die logischen und graphischen Repräsentationen der Elemente des Graphen verfügen über eine Verbindung zueinander. Durch Änderungen der Position der graphischen Repräsentation eines Knotens ändert sich beispielsweise auch die Position seiner logischen Repräsentation und umgekehrt. Des Weiteren ist das `ModuleGraphModel` auch mit der internen Repräsentation des Modells im Kern verbunden und aktualisiert diese beim Speichern des Modells im Visual Editor entsprechend.

5.1.2 Visual Explorer

Zur Exploration der Modelle in ProMoT wird der Visual Explorer verwendet, welcher unter Nutzung der Piccolo-API [BGM04], erweiterte Visualisierungstechniken (siehe Abbildung 5.3), auf Basis des Konzepts des ZUI bietet. Der Visual Explorer übernimmt das Layout eines Modells aus dem Visual Editor. Im Unterschied zur Darstellung im Visual Editor, wird hierbei das komplette Modell, d.h. die Elemente aus allen Hierarchieebenen, in einem Fenster visualisiert. Die dargestellten Elemente werden, wie bei der Darstellung im Visual Editor, entsprechend ihrer Verschachtelungstiefe, mit einem konstanten Faktor skaliert dargestellt.

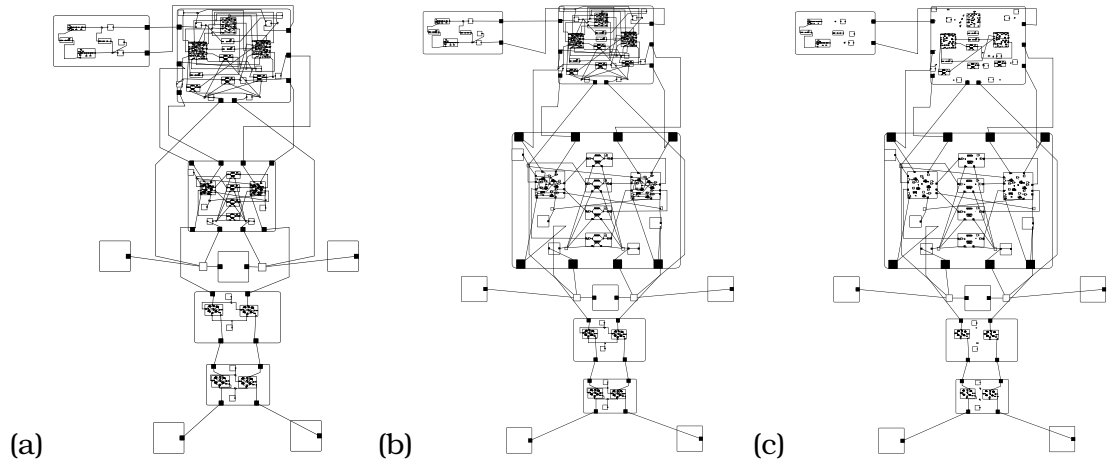


Abbildung 5.3: Visualisierungstechniken im Visual Explorer: (a) Darstellung eines kompletten Modells des EGF-Rezeptor-Netzwerks, (b) Hervorhebung eines Moduls durch die Fisheye-View-Technik [Fur86] und (c) Anwendung von Fisheye-View und Filtertechniken (Semantic Zooming [PF93]) zur Ausblendung unnötiger Details.

5.2 Integration von HyCoGLa in ProMoT

Die Entwicklung eines Zeichenverfahrens für hierarchisch strukturierte Graphen, ist das Ziel dieser Arbeit. Wurde ein intuitives Layout für ein Modell erstellt, so ist es sinnvoll, dieses zusammen mit den Modelldaten abzuspeichern, um später mit diesem Modell arbeiten zu können, ohne eine erneute Berechnung des Layouts vornehmen zu müssen. Dauerhafte Änderungen am Layout der Modelle, sind in

ProMoT nur über den Visual Editor möglich, da nur dieser mit dem Kern, welcher die Datenhaltung der Modelle verwaltet, synchronisiert ist. Aus diesem Grund wurde HyCoGLa in den Visual Editor integriert. Da der Visual Editor, wie die gesamte GUI von ProMoT, in Java geschrieben ist, wurde Java als Programmiersprache gewählt.

HyCoGLa wurde in der Klasse `CompoundLayoutManager` implementiert. Die Umsetzung der im Konzept dieser Arbeit vorgestellten Algorithmen, konnte im Prinzip ohne größere Anpassungen erfolgen. Dennoch waren einige Erweiterungen notwendig, um Datenstrukturen, die von den Algorithmen als gegeben vorrausgesetzt werden, zu erzeugen. Diese Erweiterungen werden im Folgenden beschrieben.

5.2.1 Aufbau des Inklusionsbaums in HyCoGLa

Wie in Algorithmus 4.1 beschrieben, ist zur Berechnung des Layouts durch HyCoGLa, eine Bottom-up Traversierung des Inklusionsbaums notwendig. Ein Subgraph wird im Visual Editor durch die Klasse `ModuleGraphModel` repräsentiert, welche nur über Informationen zu den Kindknoten des Subgraphen verfügt. Daher sind im Visual Editor immer nur Elemente aus einer Hierarchieebene des Inklusionsbaums präsent und darstellbar (siehe Abbildung 5.1). Da jedoch zur Traversierung der komplette Inklusionsbaums benötigt wird, wurde eine Methode zum Aufbau des Inklusionsbaums in die Klasse `CompoundLayoutManager` integriert. Da für die Traversierung des Inklusionsbaums nur wichtig ist, in welcher Verschachtelungstiefe sich ein Subgraph befindet, wurde zur Vereinfachung eine *verschachtelte Liste* als Datenstruktur zur Darstellung des Inklusionsbaums benutzt. In dieser werden die Subgraphen, wie in Abbildung 5.4 dargestellt, entsprechend ihrer Verschachtelungstiefe abgespeichert. In Algorithmus 5.1 und Funktion 5.2 ist die Verfahrensweise zum Aufbau dieser Datenstruktur dargestellt.

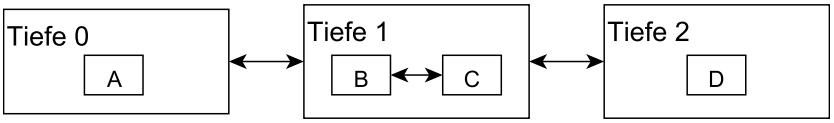


Abbildung 5.4: Verschachtelte Liste zur Darstellung der Ebenen des Inklusionsbaums am Beispiel des Compound-Graphen aus Abbildung 2.3.

Algorithmus 5.1 Aufbau des Inklusionsbaums

Input: Graph $G(V,E)$
Output: A nested list $inclusionTree$, building the inclusion tree

```
1: inclusionTree add v to list 0
2: d ← 0
3: for all v ∈ V do
4:    if v is subgraph then
5:       d ← d + 1
6:       inclusionTree add v at list d
7:       FindSubgraphs(inclusionTree, v, d)
8:    end if
9: end for
```

Funktion 5.2 FindSubgraphs

Input: Nested list $inclusionTree$, subgraph $G(V,E)$, nesting depth d of G
Output: A classification of the elements of G in $inclusionTree$

```
1: for all v ∈ V do
2:   if v is subgraph then
3:     d ← d + 1
4:     inclusionTree add v to list d
5:     FindSubgraphs(v, d)
6:   end if
7: end for
```

5.2.2 Layoutobjekte im Erweiterten Spring-Embedder

Die Implementierung des Erweiterten Spring-Embedders (siehe Abschnitt 4.2.1), welcher immer zwei Hierarchieebenen eines Compound-Graphen gleichzeitig behandelt, ist in der Klasse `CompoundSpringLayout` zu finden. Die beschriebenen Algorithmen zur Berechnung der Kräfte, wurden als Methoden in diese Klasse integriert. Bei der Implementierung des Erweiterten Spring-Embedders stellten sich jedoch einige Nachteile der im VISUAL EDITOR verwendeten, auf GEF basierenden, Datenstrukturen zur Darstellung der Knoten und Kanten des Graphen heraus:

- Jede Veränderung der Position eines Knotens (`Node`), bewirkt auch eine sofortige Veränderung der Position der graphischen Repräsentation des Knotens `FigNode` im VISUAL EDITOR. Besonders bei iterativen Zeichenverfahren, wie HyCoGLa, wirkt sich die ständige Aktualisierung der Grafik negativ auf die Laufzeit aus.

- Der weitaus größere Nachteil ist jedoch, dass immer nur sichtbare Knoten und Kanten, d.h. nur Elemente innerhalb eines aktuell geöffneten Fensters, bearbeitet und neu positioniert werden können. Wird ein Fenster geöffnet oder geschlossen, wird sein Inhalt mit der internen Struktur des Modells im Kern synchronisiert, was wiederum zu Laufzeiteinbußen führt. Da durch den Erweiterten Spring-Embedder Knoten und Kanten aus verschiedenen Subgraphen und damit aus verschiedenen VISUAL EDITOREN, gleichzeitig betrachtet werden, ist hierdurch ein aufwändiges Fenstermanagement notwendig.

Aus diesem Grund wurden neue Datenstrukturen zur Präsentation der Elemente des Graphen implementiert, die anstatt der ursprünglichen Datenelemente (`Node` und `Edge`), während der Berechnung des Layouts verwendet werden. Diese *Layoutobjekte* werden vor Beginn der Layoutberechnung mit den Werten der ursprünglichen Datenelemente des Graphen, welche sich in verschiedenen VISUAL EDITOREN befinden, initialisiert. Nach der Layoutberechnung werden die verschiedenen VISUAL EDITOREN erneut geöffnet, um die ursprünglichen Datenelemente mit den Werten der Layoutobjekte (und damit auch die graphischen Repräsentationen der Datenelemente) zu aktualisieren. Tabelle 5.1 stellt die Beziehung zwischen den Datenelementen des Graphen und den Layoutobjekten in einer Übersicht dar. Ein weiterer Vorteil der Verwendung von Layoutobjekten ist, dass diese genau die Variablen enthalten, die zur Berechnung des Layouts benötigt werden. Anstatt die auf die Knoten wirkenden Kräfte, Temperaturen und anderen Parameter für jeden

Knoten in zusätzliche Datenstrukturen, wie z.B. *HashMaps* abzuspeichern, können diese direkt als Parameter der Layoutobjekte gespeichert werden. Wie erläutert, be-

Graphentheorie	VE logisch	VE graphisch	Layoutobjekt
Knoten	`Node`	`FigNode`	`LayoutNode`
Subgraph	`ModuleNode`	`FigModuleNode`	`LayoutModuleNode`
Kante	`Edge`	`FigEdge`	`LayoutEdge`
Port	`TerminalPort`	-	`LayoutPort`

Tabelle 5.1: Gegenüberstellung von Bezeichnungen aus Graphentheorie mit den Bezeichnungen der Datenelemente im VISUAL EDITOR (*VE*)

finden sich die Elemente aus verschiedenen Subgraphen in verschiedenen VISUAL EDITOREN. Jeder VISUAL EDITOR, und damit jeder Subgraph, verfügt über ein eigenes lokales Koordinatensystem, in welchem die globale Skalierung der Elemente nicht betrachtet wird (siehe Abbildung 5.1). Für die Berechnung des Layouts ist es jedoch notwendig, dass die Koordinaten aller Elemente in einem globalen Koordinatensystem (dem des Elternknotens) vorliegen und dass die Skalierung der Elemente aus den Subgraphen beachtet wird. Aus diesem Grund müssen die Koordinaten der Elemente der Subgraphen, nach dem Auslesen aus den VISUAL EDITOREN, zunächst in das globale Koordinatensystem des Elternknotens umgewandelt werden. Die Umwandlung von lokalen in globale Koordinaten, ist in Funktion 5.3 dargestellt. Damit die Layoutobjekte der Subgraphen wieder den Graphelementen, und damit ihren graphischen Repräsentationen in den VISUAL EDITOREN, zugeordnet werden können, müssen nach Abschluss der Berechnung des Layouts, ihre globalen Koordinaten wieder in lokale Koordinaten umgewandelt werden. Die Umwandlung von globalen in lokale Koordinaten, ist in Funktion 5.4 dargestellt.

Funktion 5.3 TransformFromLocalToGlobal

Input: The node n to transform, global scaling factor s, center pcg, width pw and height ph of the nodes parent in global coordinates

Output: n, scaled and transformed into global coordinates

1: {1. Calculate the center pcl of the parent of n in local coordinates:}
2: $pcl.x \leftarrow pw/(2 * s)$
3: $pcl.y \leftarrow ph/(2 * s)$
4: {2. Find the distance d between the center of n and pcl:}
5: $d.x \leftarrow n.x - pcl.x$
6: $d.y \leftarrow n.y - pcl.y$
7: {3. Scale d:}
8: $d.x \leftarrow d.x * s$
9: $d.y \leftarrow d.y * s$
10: {4. Find the global coordinates of n by adding d to pcg:}
11: $n.x \leftarrow pcg.x + d.x$
12: $n.y \leftarrow pcg.y + d.y$
13: {5. Scale the size of n:}
14: $n.width \leftarrow n.width * s$
15: $n.height \leftarrow n.height * s$

Funktion 5.4 TransformFromGlobalToLocal

Input: The node n to transform, global scaling factor s, center pcg, width pw and height ph of the nodes parent in global coordinates

Output: n, scaled and transformed into local coordinates

1: {1. Calculate the center pcl of the parent of n in local coordinates:}
2: $pcl.x \leftarrow pw * s/2)$
3: $pcl.y \leftarrow ph * s/2)$
4: {2. Find the distance d between the center of n and pcg:}
5: $d.x \leftarrow n.x - pcg.x$
6: $d.y \leftarrow n.y - pcg.y$
7: {3. Scale d:}
8: $d.x \leftarrow d.x/s$
9: $d.y \leftarrow d.y/s$
10: {4. Find the local coordinates of n by adding d to pcl:}
11: $n.x \leftarrow pcl.x + d.x$
12: $n.y \leftarrow pcl.y + d.y$
13: {5. Scale the size of n:}
14: $n.width \leftarrow n.width/s$
15: $n.height \leftarrow n.height/s$

5.2.3 Probleme

HyCoGLa wurde wie beschrieben implementiert und in ProMoT integriert. Beim Testen kam es jedoch zu verschiedenen Problemen, welche die Anwendbarkeit von HyCoGLa beeinträchtigen. Einige dieser Probleme wurden beseitigt, andere konnten jedoch bis zum Zeitpunkt der Fertigstellung dieser Arbeit nicht behoben werden. Die noch bestehenden Probleme sind im Folgenden aufgelistet:

- Bei größeren Modellen kommt es vor, dass die Verbindung zu dem in Lisp geschriebenen Kern, der die Modelle verwaltet, abbricht. Dies führt dazu, dass die Layoutobjekte nicht mehr mit den ursprünglichen Datenelementen synchronisiert werden können und damit zum Abbruch der Layoutberechnung. Dieser Fehler tritt jedoch nicht immer an der gleichen Stelle der Layoutberechnung auf und lässt sich daher schwer zurückverfolgen.

- Zur Vermeidung eines aufwändigen Fenstermanagements werden, wie beschrieben, Layoutobjekte anstatt der Datenelemente des Graphen im Visual Editor verwendet. Dennoch ist es notwendig, zu Beginn und nach Abschluss der Layoutberechnung durch den Erweiterten Spring-Embedder, die aktuell zu bearbeitenden Subgraphen in einem neuen Visual Editor zu öffnen, um entweder die enthaltenen Elemente auszulesen oder diese, entsprechend der berechneten neuen Positionen, zu aktualisieren. Das Öffnen der Visual Editoren ist jedoch zeitaufwändig und wirkt sich negativ auf die Laufzeit aus.

- Die Aktualisierung der Knoten mit den berechneten Werten ist nicht immer konsistent. Es kommt vor, dass die Positionen von gerade positionierten Knoten, kurz vor dem Schließen und Speichern des aktualisierten Visual Editors, noch einmal verändert werden und somit das berechnete Layout teilweise zerstört wird. Hier wird ein Synchronisationsproblem zwischen der gra-

phischen Darstellung des Graphen im VISUAL EDITOR und der internen Repräsentation im Kern vermutet.

5.3 Zusammenfassung

In diesem Kapitel wurde die Implementierung von HYCOGLA und die Integration in das Modellierungsprogramm PROMOT beschrieben. Hierzu wurde PROMOT vorgestellt und dessen Aufbau erläutert. Da es nur über die Editierkomponente von PROMOT, dem VISUAL EDITOR, möglich ist, Änderungen am Layout der Modelle abzuspeichern, wurde HYCOGLA in diese integriert.

Für die Umsetzung der im Konzept dieser Arbeit entwickelten Algorithmen, waren einige Anpassungen und Erweiterungen notwendig. Es mussten Datenstrukturen und Methoden implementiert werden, welche im Konzept als gegeben vorausgesetzt wurden, in der Implementierung von PROMOT jedoch nicht vorhanden waren. Eine dieser Erweiterungen war der Aufbau einer Datenstruktur zur Verwaltung des Inklusionsbaums eines Compound-Graphen. Eine weitere notwendige Erweiterung war die Implementierung sogenannter Layoutobjekte, welche anstatt der ursprünglichen Graphelemente im VISUAL EDITOR bei der Berechnung des Layouts verwendet werden. Der Vorteil der Verwendung der Layoutobjekte liegt zum einen in einer Verminderung der Gesamtlaufzeit, da die ständige Aktualisierung der graphischen Repräsentationen der Elemente des Graphen vermieden wird, zum anderen vereinfacht sie die Berechnung, da sich alle Layoutobjekte im gleichen, globalen Koordinatensystem befinden.

Im letzten Teil des Kapitels wurde auf die bei der Implementierung aufgetretenden Probleme eingegangen. Einige der Probleme konnten gelöst werden, andere Probleme, die auf ein Synchronisationsproblem zwischen der in JAVA geschriebenen GUI und dem in LISP geschriebenen Kern zurückzuführen sind, konnten jedoch bis zum Zeitpunkt der Abgabe dieser Arbeit nicht behoben werden.

6 Ergebnisse

In diesem Kapitel wird die Auswertung des entwickelten Zeichenverfahrens erläutert. Hierzu werden zunächst Experimente zur Bewertung von HyCoGLa durchgeführt und erste Ergebnisse, in Bezug auf Laufzeit und Darstellungsqualität, mit einem lokalen Zeichenverfahren verglichen. Anschließend erfolgt eine Bewertung der mit HyCoGLa erzielbaren Ergebnisse anhand von Beispielgraphen.

6.1 Experimente

6.1.1 Aufbau

Zeichenverfahren lassen sich an genau zwei Größen messen: ihrer Laufzeit und ihrer Darstellungsqualität. Da HyCoGLa eine Mischform zwischen lokalen und globalen Verfahren zum Zeichnen von Compound-Graphen ist, müsste es zur Einordnung der Ergebnisse mit diesen beiden Verfahren verglichen werden. Zu erwarten wäre hier, dass HyCoGLa eine schlechtere Laufzeit und eine bessere Darstellungsqualität als rein lokale Methoden und eine bessere Laufzeit und eine schlechtere Darstellungsqualität als rein globale Methoden erzielt.

Für einen aussagekräftigen Vergleich der Zeichenverfahren ist es notwendig, dass alle Verfahren dieselben Optimierungen und Erweiterungen (Skelettierungstechnik, Beachtung der Skalierung von Subgraphen, Beachtung von Terminalknoten) wie HyCoGLa verwenden. Ein rein lokales Zeichenverfahren, welches diese Anforderungen erfüllt, wurde implementiert. Im zeitlichen Rahmen dieser Arbeit war es jedoch nicht möglich, ebenso ein globales Zeichenverfahren mit diesen Anforderungen, zu implementieren. Aus diesem Grund wurden die Ergebnisse von HyCoGLa nur mit den Ergebnissen eines lokalen Zeichenverfahrens verglichen.

Als Testgraphen wurden Graphen verwendet, die sich in Knotenanzahl, Kantenanzahl, der Anzahl an Subgraphen und der Verschachtelungstiefe unterscheiden (siehe Tabelle A.1). Testgraphen 8-11 sind Realmodelle aus der Systembiologie. Da keine Möglichkeit zur automatischen Generierung von Testgraphen bestand und ein automatisiertes Testverfahren ebenfalls nicht implementiert wurde, ist die Anzahl an Testgraphen zu gering, um ein repräsentatives Ergebnis zu erhalten. Weiterhin konnten nur Terminal-Cluster-Graphen verwendet und untersucht werden, da in ProMoT nur diese Art von Graphen unterstützt wird. Compound-Graphen mit Interkanten zwischen Knoten aus Hierarchieebenen, die sich um mehr als eine Verschachtelungstiefe unterscheiden, wurden nicht betrachtet. Daher sollten die durchgeführten Experimente eher als erste Einordnung und Bewertung von HyCoGLa angesehen werden, mit dem Hinweis, dass umfangreichere Experimente zur

endgültigen Bewertung notwendig sind.

Bei beiden Verfahren fand die Berechnung der Layouts unter den gleichen Voraussetzungen (gleiche Parameter, 100 Iterationen, gleiches Zufallslayout als Ausgangslayout) statt. Die Experimente wurden auf einem Standard-PC (Intel Pentium 4, 3GHz CPU, 2GB RAM, Suse Linux 10.1) durchgeführt.

6.1.2 Messwerte

Bei der Untersuchung der Gesamtlaufzeit, wurden die Laufzeiten der einzelnen Teilschritte der implementierten Zeichenverfahren erfasst, um einen besseren Einblick in die Verteilung der Laufzeiten zu erhalten. Diese Teilschritte sind:

- Aufbau des Inklusionsbaums,
- Erzeugung der Layoutobjekte,
- Berechnung des Layouts,
- Entfernung von verbliebenen Knotenüberlappungen in einem Nachbearbeitungsschritt und
- Aktualisierung der graphischen Repräsentationen der Knoten und Kanten.

Die schlechtere Qualität lokaler Zeichenverfahren, ist auf die Nichtbeachtung der Beziehungen zwischen Knoten aus verschiedenen Subgraphen und die hierdurch ungünstig verlaufenden Kanten zurückzuführen. Aus diesem Grund wurden zum Vergleich der Darstellungsqualität der beiden Verfahren, folgende Eigenschaften der erzeugten Layouts gemessen:

- Anzahl von Kanten-Kanten-Kreuzungen,
- Anzahl von Kanten-Knoten-Kreuzungen (Als Kanten-Knoten-Kreuzung wird eine Überschneidung einer Kante mit einem Knoten, welcher nicht Endknoten dieser Kante ist, gewertet. Zusätzlich werden unter Umständen auch die Überschneidungen einer Kante mit einem ihren Endknoten als Kanten-Knoten-Kreuzungen gewertet (siehe Abbildung 6.1).) und
- Abweichung der Kantenlängen von der idealen Kantenlänge.

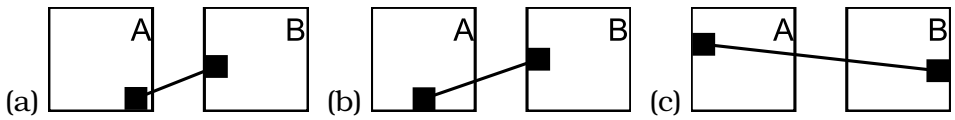


Abbildung 6.1: Wertung der Kanten-Knoten-Kreuzung einer Kante mit ihren Endknoten: (a) Keine Kanten-Knoten-Kreuzung, da Länge der Kante innerhalb der Knoten kleiner als ein definierter Schwellwert (z.B. wurde bei den Experimenten, die dreifache Seitenlänge eines Ports benutzt (Erfahrungswert)), (b) Wertung als eine Kanten-Knoten-Kreuzung (nur in Knoten A) und (c) Wertung als zwei Kanten-Knoten-Kreuzungen (A und B), da Länge der Kante innerhalb der Knoten größer als der Schwellwert.

6.2 Auswertung

Dieser Abschnitt beschäftig sich mit der Auswertung der durchgeführten Experimente. Hierzu werden die Ergebnisse von HyCoGLa und dem lokalen Vergleichsverfahren in Bezug auf Laufzeit und Qualität verglichen und diskutiert. In Tabelle A.1 im Anhang befindet sich die textuelle Darstellung der Ergebnisse.

6.2.1 Laufzeit

Das Diagramm in Abbildung 6.2 stellt für die beiden verglichenen Verfahren die kumulierten Laufzeiten jedes Teilschritts gegenüber. Wie angenommen, ist die Gesamtlaufzeit von HyCoGLa höher als die, des lokalen Zeichenverfahrens. Die Laufzeit zum Aufbau des Inklusionsbaums ist bei beiden Verfahren gleich. Auch die Entfernung von Knotenüberlappungen benötigt bei beiden Verfahren die gleiche Zeit, da hier jeder Subgraph einzeln betrachtet wird. Die benötigte Zeit zur Erzeugung der Layoutobjekte und zur Aktualisierung der Knotenpositionen ist bei HyCoGLa, durch das aufwändige Fenstermanagment, deutlich höher als beim lokalen Zeichenverfahren. Auch die Laufzeit zur eigentlichen Layoutberechnung ist, aufgrund der in HyCoGLa betrachteten globalen Zusammenhänge, etwa um ein vierfaches höher als die, des lokalen Zeichenverfahrens. Wie erwartet, ist die Laufzeit abhän-

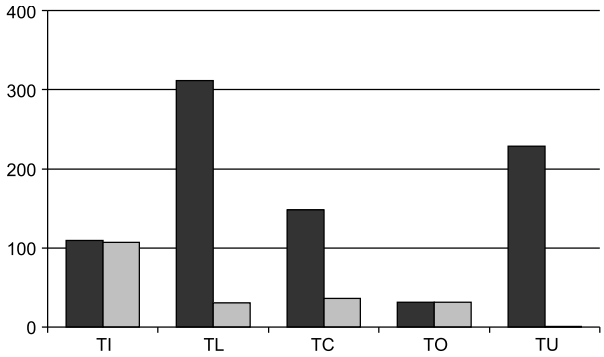


Abbildung 6.2: Gegenüberstellung der Laufzeiten der einzelnen Teilschritte (Aufbau des Inklusionsbaums (*TI*), Erzeugung der Layoutobjekte (*TL*), eigentlichen Berechnung des Layouts (*TC*), Entfernung von Knotenüberlappungen (*TO*) Aktualisierung (*TU*)) von HyCoGLa (schwarz) und einem lokalen Verfahren (grau).

gig von der Struktur des Graphen. Bei Graphen mit einer Verschachtelungstiefe < 3 ist die theoretische Laufzeit gleich der Laufzeit anderer Spring-Embedder-basierter Verfahren ($\mathcal{O}(k * n^2)$). In der Praxis lässt sich die Laufzeit, wie auch bei dem in [DGC+04] beschriebenen globalen Verfahren, durch Nutzung der Skelettierungstechnik und durch die ausschließliche Berechnung der Abstoßungskräfte zwischen Knoten mit demselben Elternknoten deutlich verbessern. Bei Graphen mit einer Verschachtelungstiefe > 2 kommt der Vorteil des Divide&Conquer-Ansatzes des Algorithmus zum Tragen, da nicht mehr das Layout des gesamten Graphen berechnet werden muss, sondern immer nur das Layout einzelner Teilgraphen unabhängig voneinander berechnet werden kann. Die in in Tabelle A.1 dargestellten Laufzeiten zur Berechnung der Layouts der Testgraphen 9 und 11, können als Beispiel für diesen Effekt herangezogen werden: Während für die Berechnung des Layouts

des wenig strukturierten Testgraphen 9, mit 469 Knoten, 57 Sekunden benötigt wurden, wurden für den stark strukturierten Testgraphen 11, bei doppelter Knotenzahl, nur 9 Sekunden benötigt [1].

In der Praxis bewegt sich die aktuelle Gesamtlaufzeit von 3 bis 4 Minuten bei größeren Realmodellen mit ca. 500-1000 Knoten jedoch in einem akzeptablen Rahmen, wenn berücksichtigt wird, dass das Layout nur einmal berechnet werden muss. Die größten Schwachstellen in der aktuellen Implementierung sind hierbei die zeitaufwändige Erzeugung der Layoutobjekte und die Aktualisierung der Knotenpositionen, welche einen Großteil der Rechenzeit in Anspruch nehmen. Ein weiterer Ansatzpunkt zur Optimierung, wäre die Nutzung einer erweiterten Multileveltechnik, wie z.B. in [Wal00] beschrieben, anstelle der verwendeten einfachen Skelettierungstechnik.

6.2.2 Darstellungsqualität

Die Experimente zur Darstellungsqualität brachten ebenfalls die zu erwartenden Resultate. HyCoGLa schneidet aufgrund der Betrachtung der globalen Zusammenhänge bei allen gemessenen Kriterien besser ab, als das lokale Zeichenverfahren. Kanten-Kanten- und Kanten-Knoten-Kreuzungen und die Abweichung der mittleren Kantenlänge von der idealen Kantenlänge konnten verringert werden. Das Diagramm in Abbildung 6.3 stellt die Ergebnisse der Messungen der Werte beider Verfahren gegenüber. Beim Vergleich von Laufzeit und Darstellungsqualität ist jedoch

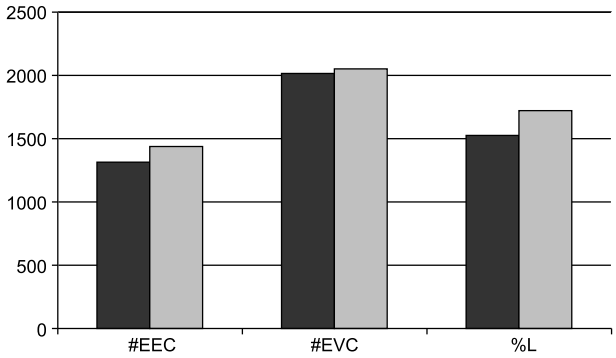


Abbildung 6.3: Gegenüberstellung der Darstellungsqualität von HyCoGLa (schwarz) und einem lokalen Verfahren (grau) in Bezug auf Kanten-Kanten-Kreuzungen (*#EEC*), Kanten-Knoten-Kreuzungen (*#EVC*) und die Summe der Abweichungen der Kantenlängen von der idealen Kantenlänge (*%L*)

festzustellen, dass der Vorteil von HyCoGLa bezüglich der Darstellungsqualität, nicht so groß ist, wie der Vorteil des lokalen Verfahrens in puncto Laufzeit. Ein Grund hierfür ist, dass durch HyCoGLa Knotenüberlappungen, besonders bei größeren Subgraphen, nicht komplett ausgeschlossen werden können und der Nachbearbeitungsschritt diese ohne Berücksichtigung der Kantenverläufe entfernt, was

1 Zum Vergleich mit einem globalen Verfahren, könnten die Werte aus [DGC+04] herangezogen werden. Bei diesem, wurden für Graphen mit ca. 500 Knoten, Laufzeiten für die Berechnung des Layouts von ca. 20 Sekunden ermittelt. Dieser Vergleich ist jedoch mit Vorsicht zu betrachten, da die Implementierung und die Experimenten des globalen Verfahrens, nicht unter den gleichen Voraussetzungen, wie bei HyCoGLa, stattfanden.

negative Auswirkungen auf die optimierten Kriterien hat. Ein weiterer Grund hierfür ist, dass ausschließlich Terminal-Cluster-Graphen in den Experimenten untersucht wurden. Da Terminal-Cluster-Graphen nur Interkanten besitzen, welche über maximal zwei Hierarchieebenen verlaufen, sind diese mit lokalen Verfahren ebenfalls gut darstellbar.

Nachdem der Vergleich der Darstellungsqualität der beiden gegenübergestellten Verfahren, anhand bloßer Zahlen durchgeführt wurde, wird sie im Folgenden am Beispiel von zwei Testgraphen aus Tabelle A.1 verglichen. In den Abbildungen 6.4 und 6.5 sind die erstellten Layouts der Testgraphen 1 und 10 dargestellt[2]. Die Struktur der inneren Layouts der Subgraphen ist bei allen Darstellungen gut erkennbar. Der Unterschied zwischen den lokalen Layouts (Abbildungen 6.4(a) und 6.5(a)) und dem durch HyCoGLa erzeugten Layouts (Abbildungen 6.4(b) und 6.5(b)) ist jedoch die Darstellung der Interkanten. Da das lokale Verfahren die einzelnen Subgraphen unabhängig voneinander betrachtet und keine Optimierung hinsichtlich der globaler Beziehungen zwischen Knoten durchführt, kommt es bei der Komposition der Subgraphen zum Gesamtlayout zu unerwünschten Effekten, wie Kanten-Kanten- und Kanten-Knoten-Kreuzungen. Diese können durch die Betrachtung der globalen Zusammenhänge in HyCoGLa verringert werden, wodurch es einfacher wird, dem Verlauf von Interkanten zu folgen.

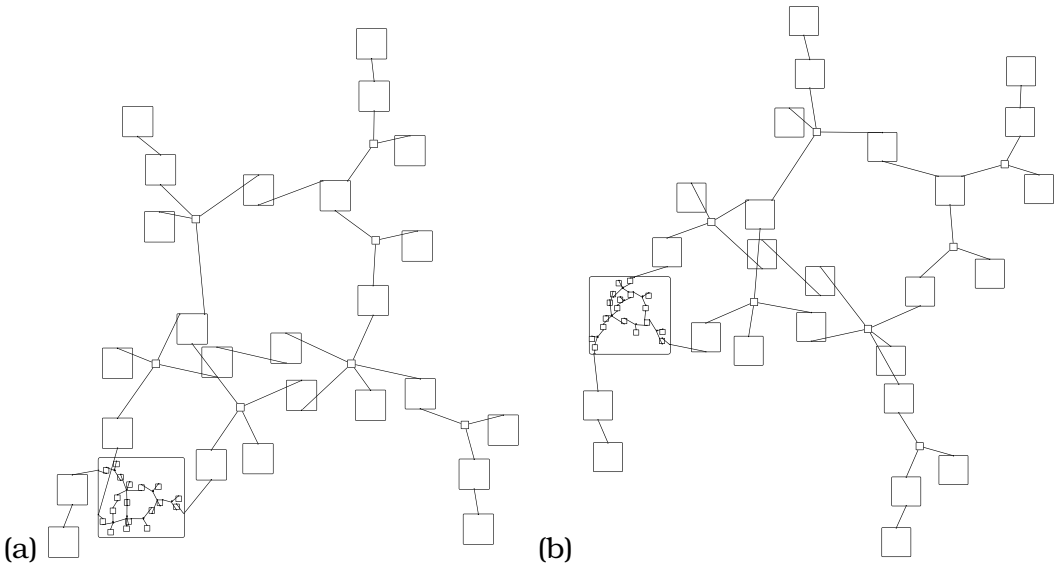


Abbildung 6.4: Vergleich der Layouts des Testgraphen 1: (a) Lokales Verfahren und (b) HyCoGLa. Durch die Beachtung globaler Beziehungen in HyCoGLa können adjazente Knoten aus verschiedenen Subgraphen günstiger zueinander platziert werden. Hierdurch wird die Qualität des erzeugten Layouts im Vergleich zum lokalen Verfahren verbessert (weniger Kanten-Kanten und Kanten-Knoten-Kreuzungen).

2 Weitere Darstellungen der Testgraphen aus Tabelle A.1 sind in Abbildung 4.8 sowie in Abbildungen A1-A4 im Anhang zu finden.

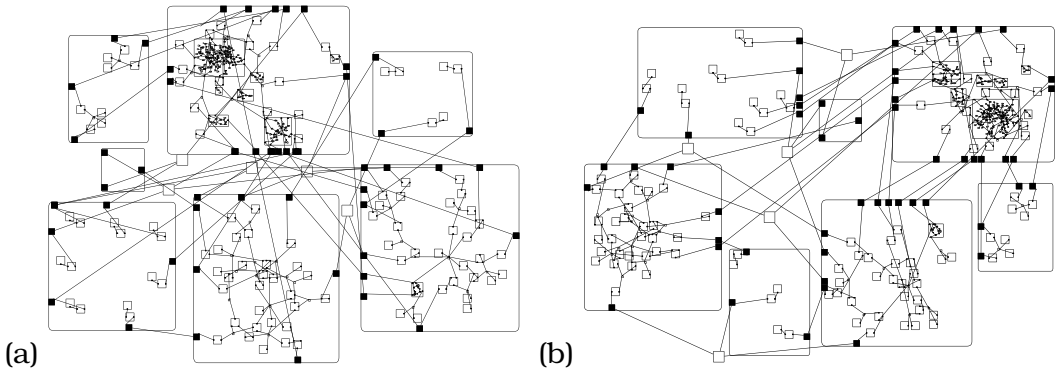


Abbildung 6.5: Vergleich der Layouts des Testgraphen 10, welcher ein Realmodell aus der Systembiologie repräsentiert: (a) Lokales Verfahren und (b) HyCoGLa. Bei beiden Verfahren werden die inneren Layouts der Subgraphen gut dargestellt. Durch die Beachtung globaler Beziehungen zwischen Knoten aus verschiedenen Subgraphen in HyCoGLa, kommt es gegenüber dem lokalen Verfahren zu einer deutlichen Reduzierung von Kantenkreuzungen.

6.3 Zusammenfassung

In diesem Kapitel wurden Experimente zur Bestimmung der Laufzeit und Darstellungsqualität von HyCoGLa beschrieben und durchgeführt. Hierzu wurden passende Messwerte und Testgraphen ausgewählt und die durch HyCoGLa erzielten Ergebnisse mit einem lokalen Zeichenverfahren verglichen. Ein aussagekräftiger Vergleich mit einem globalen Verfahren, konnte aufgrund der fehlenden Zeit nicht durchgeführt werden. Des Weiteren war die Anzahl an getesteten Graphen zu gering, um eine repräsentative Bewertung von HyCoGLa durchzuführen. Erste Ergebnisse konnten jedoch zeigen, dass die Erwartungen an HyCoGLa (bessere Darstellungsqualität, bei längerer Laufzeit im Vergleich zu dem lokalen Zeichenverfahren) erfüllt werden konnten.

7 Zusammenfassung

Hierarchisch strukturierte Graphen, sogenannte Compound-Graphen, werden in vielen Anwendungsgebieten eingesetzt, um komplexe und große Datenmengen und die Beziehungen zwischen den Datenelementen strukturiert darzustellen. In der Systembiologie werden beispielsweise Modelle zellulärer Systeme durch Compound-Graphen abstrahiert und graphisch dargestellt. Für das Verständnis, der durch diese Graphen repräsentierten Informationen, ist ein intuitiv verständliches Layout dieser Graphen erforderlich. Da die manuelle Erstellung eines aussagekräftigen Layouts aufgrund der Größe und Komplexität der dargestellten Modelle oft nicht möglich ist, wurde in dieser Arbeit HyCoGLa, ein automatisches Verfahren zum Zeichnen von Compound-Graphen, entwickelt.

HyCoGLa basiert auf dem Spring-Embedder-Verfahren und stellt eine Mischform existierender Techniken (lokal und global) zum Zeichnen ungerichteter Compound-Graphen dar. Es ist der erste Ansatz, der versucht, den Trade-off zwischen rein lokalen Techniken (schlechte Qualität, schnell) und rein globalen Techniken (gute Qualität, langsam) zu verbessern. Hierzu wird im Gegensatz zu lokalen und globalen Techniken weder jeder Subgraph des Compound-Graphen einzeln, noch der komplette Compound-Graph auf einmal betrachtet, sondern immer eine Teilmenge von Subgraphen und deren Kindknoten über zwei Hierarchieebenen hinweg. Da aus diesem Grund keine komplexe Datenstruktur zur Verwaltung des kompletten Compound-Graphen notwendig ist, kann auch der Implementierungsaufwand von HyCoGLa im Vergleich zu globalen Verfahren verringert werden.

Des Weiteren beachtet HyCoGLa verschiedene Besonderheiten, die von existierenden Verfahren zum Zeichnen von Compound-Graphen nicht beachtet werden. Diese sind:

- Die Skalierung von Elementen des Graphen entsprechend ihrer Verschachtelungstiefe. Dadurch, dass in HyCoGLa immer nur zwei Hierarchieebenen gleichzeitig betrachtet werden, ist dies leicht umsetzbar.
- Die Beachtung der Anforderungen an das Layout von Terminalknoten.

Es wurden Techniken beschrieben und in HyCoGLa integriert, um Anforderungen an das Layout biologischer Netzwerke, bei Vorhandensein entsprechender semantischer Informationen, zu beachten. Hierzu gehören beispielsweise die Beachtung der Richtung von Kanten oder die Darstellung der subzellulären Lokalisation. Weiterhin wurde erläutert, wie die Interaktion von Anwendern mit automatischen Zeichenverfahren, wie HyCoGLa, umgesetzt und verbessert werden kann.

HyCoGLa wurde in das Modellierungswerkzeug ProMoT integriert und an typischen und teilweise sehr komplexen Modellen aus der Systembiologie getestet. Die erzielten Ergebnisse wurden in Hinblick auf die Laufzeit und die Qualität der erzeugten Darstellung als gut eingestuft.

Da sich HyCoGLa zum Zeichnen der beschriebenen Modelle aus der Systembiologie eignet und alle gestellten Anforderungen umgesetzt wurden, konnte das Ziel der Arbeit erreicht werden.

7.1 Ausblick

Das in dieser Arbeit entwickelte Verfahren, eignet sich in seinem aktuellen Zustand zur Erzeugung einer angemessenen Darstellung von Compound-Graphen. Dennoch existieren einige Ansatzpunkte zur Erweiterung und Verbesserung des Verfahrens, die jedoch aufgrund des beschränkten Zeitrahmens in dieser Arbeit nicht umgesetzt werden konnten. Diese werden im Folgenden erläutert.

Die in Abschnitt 5.2.3 beschriebenen Synchronisationsprobleme beeinträchtigen die Praxistauglichkeit von HyCoGLa. Da diese Probleme zu Fehlern im Layout oder zum Abbruch des Layoutprozesses führen können, ist es als erstes notwendig, diese Probleme zu beheben.

Die Qualität der mit HyCoGLa erzielbaren Ergebnisse ist zwar gut, kann aber durch verschiedene Erweiterungen verbessert werden. Hierzu gehören:

- Verringerung von Kanten-Knoten-Kreuzungen: In der aktuellen Version von HyCoGLa erfolgt keine explizite Vermeidung von Kanten-Knoten-Kreuzungen. Zu diesem Zweck ist es möglich, ähnlich wie bei der Vermeidung von Knotenüberlappungen, zusätzlich Abstoßungskräfte zwischen Kanten und Knoten zu definieren [RMS97]. Eine weitere Möglichkeit zur Verhinderung von Kanten-Knoten-Kreuzungen ist die Durchführung eines Kantenroutings, als Nachbearbeitungsschritt nach der eigentlichen Layoutberechnung [Fre01]. Beide Techniken zur Verringerung von Kanten-Knoten-Kreuzungen haben allerdings den Nachteil, dass durch sie die Gesamtlaufzeit der Layoutberechnung erhöht wird.
- Verbesserung der Platzierung der Terminalknoten: Bisher werden die Terminalknoten nach Abschluss der Layoutberechnung an den ihnen am nächstgelegenen Rand platziert. Dies ist nicht immer die beste Variante, da durch diese „0-1“-Entscheidung oft harte Knicke im Kantenverlauf enstehen, die die Verfolgung des Kantenverlaufs erschweren. Aus diesem Grund ist es von Vorteil, eine erweiterte Platzierungstechnik für die Terminalknoten, beispielsweise unter Beachtung der zuletzt auf einen Terminalknoten wirkenden Kräfte, zu verwenden.
- Beachtung des größeren Platzbedarfs von Subgraphen: Bei den durch HyCoGLa erzeugten Layouts kommt es trotz Beachtung der Größe der Knoten zu Überlappungen und Clusterbildungen von Subgraphen. Daher ist es notwendig, die optimalen Kantenlängen für Interkanten und Abstände zwischen

Subgraphen zu vergrössern, um diesen mehr Platz als einfachen Knoten zu lassen.

Auch die Laufzeit von HyCoGLa bietet noch einige Ansatzpunkte zur Verbesserung:

- Die benötigte Zeit zur Aktualisierung der graphischen Repräsentationen der Knoten in der GUI von ProMoT nimmt einen großen Anteil an der Gesamtlaufzeit ein. Daher ist eine Beschleunigung der Aktualisierung notwendig.
- Da das Ergebnis von HyCoGLa vom Ausgangslayout des Graphen abhängt, kann die Nutzung eines optimaleren Ausgangslayouts, anstelle des verwendeten Zufallslayouts, die Laufzeit der Layoutberechnung verringern. Eine Technik zur Erzeugung eines optimalen Ausgangslayouts für Spring-Embedder-basierte Verfahren, ist beispielweise in [MR02] beschrieben.
- Die eigentliche Laufzeit zur Berechnung des Layouts ist besonders bei großen, wenig strukturierten Graphen verbesserungswürdig. Zur Beschleunigung der Berechnung wäre die Nutzung einer erweiterten Multilevel-Technik, wie sie beispielsweise in [Wal00] eingesetzt wird, sinnvoll. Mittlerweile existieren auch Ansätze zur effizienten Layoutberechnung unter Nutzung moderner Graphik-Hardware [FT07]. Da die hierfür benötigte Graphik-Hardware zur Zeit jedoch noch wenig verbreitet ist (zumindest nicht dort, wo keine 3D-Graphik benötigt wird), ist diese Optimierungstechnik in der aktuellen Praxis kaum einsetzbar.

Aktuell betrachtet HyCoGLa maximal zwei Hierarchieebenen gleichzeitig. An dieser Stelle wäre es interessant zu untersuchen, inwieweit die Steuerung des Trade-offs zwischen Geschwindigkeit und Darstellungsqualität, durch die gleichzeitige Betrachtung von mehr als zwei Hierarchieebenen, möglich ist.

Ein letzter Ansatzpunkt für weiterführende Arbeiten ist eine ausführlichere Evaluierung von HyCoGLa. In Kapitel 6 wurden bereits Experimente beschrieben, die dazu verwendet werden können. Da die Anzahl der durchgeführten Experimenten zu gering war, um ein repräsentatives Ergebnis zu erhalten, ist die Durchführung weiterer Experimente notwendig. Des Weiteren wurde HyCoGLa aufgrund des Fehlens eines globalen Vergleichsverfahrens, nur mit einem lokalen Verfahren verglichen. Daher konnte die angestrebte Verbesserung des Trade-offs zwischen lokalen und globalen Verfahren nur teilweise bestätigt werden. Als Testumgebung zur Umsetzung weiterführender Experimente und zum Vergleich mit einem globalen Verfahren bietet sich Chisio [CKG08], ein frei verfügbares und erweiterbares Framework zum Erstellen und Modifizieren von Compound-Graphen, an. In diesem ist bereits ein globales Zeichenverfahren [DGC+04] implementiert. Für einen Vergleich müssten dann nur noch ein lokales Verfahren und HyCoGLa integriert werden.

Literaturverzeichnis

[AMA07] ARCHAMBAULT, DANIEL, TAMARA MUNZNER und DAVID AUBER: *Grouse: Feature-Based, Steerable Graph Hierarchy Exploration*. In: *Eurographics/ IEEE-VGTC Symposium on Visualization*, S. 67–74. Eurographics Association, 2007.

[BBF+04] BACHMAIER, CHRISTIAN, FRANZ-JOSEF BRANDENBURG, MICHAEL FORSTER, PAUL HOLLEIS und MARCUS RAITNER: *Gravisto: Graph Visualization Toolkit*. In: *Graph Drawing*, S. 502–503. Springer-Verlag, 2004.

[BBS08] BACHMAIER, CHRISTIAN, ULRIK BRANDES und FALK SCHREIBER: *Handbook of Graph Drawing and Visualization*, Kapitel 5: Biological Networks. Chapman & Hall/CRC Press, voraussichtliches Erscheinungsdatum 2008.

[BETT99] BATTISTA, GIUSEPPE DI, PETER EADES, ROBERTO TAMASSIA und IOANNIS G. TOLLIS: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

[BFP+04] BRANDENBURG, FRANZ J., MICHAEL FORSTER, ANDREAS PICK, MARCUS RAITNER und FALK SCHREIBER: *BioPath: Exploration and Visualization of Biochemical Pathways*. In: *Graph Drawing Software*, S. 215–236. Springer-Verlag, 2004.

[BGHM07] BARSKY, AARON, JENNIFER L. GARDY, ROBERT E. W. HANCOCK und TAMARA MUNZNER: *Cerebral: a Cytoscape plugin for layout of and interaction with biological networks using subcellular localization annotation.* Bioinformatics, 23:1040–1042, 2007.

[BGHS98] BRANDENBURG, FRANZ J., BERNHARD GRUBER, MICHAEL HIMSOLT und FALK SCHREIBER: *Automatische Visualisierung biochemischer Information*. In: *Workshop Molekulare Bioinformatik, 28th Annual Conference German CS Society*, S. 24–38. Shaker Verlag, 1998.

[BGM04] BEDERSON, BENJAMIN B., JESSE GROSJEAN und JON MEYER: *Toolkit Design for Interactive Structured Graphics*. In: *IEEE Transactions on Software Engineering*, S. 535–546. IEEE Computer Society Press, 2004.

[BM99] BERTAULT, FRANÇOIS und MIRKA MILLER: *An Algorithm for Drawing Compound Graphs*. In: *Graph Drawing*, S. 197–204. Springer-Verlag, 1999.

[BUS04] BIERMANN, SUSANNE, ADELINDE M. UHRMACHER und HEIDRUN SCHUMANN: *Supporting Multi-Level Models in Systems Biology by Visual Methods*. European Simulation Multiconference, 2004.

[CKG08] CIHAN KUCUKKECECI, UGUR DOGRUSOZ, M.ESAT BELVIRANLI ALPTUG DILEK und ERHAN GIRAL: *Chisio: Compound or Hierarchical Graph Visualization Tool*. `http://www.cs.bilkent.edu.tr/~ivis/chisio.html`, 2008. Website, zuletzt besucht am 14.02.2008.

[DBS06] DECKARD, ANASTASIA, FRANK T. BERGMANN und HERBERT M. SAURO: *Supporting the SBML layout extension*. Bioinformatics, 22:2966–2967, 2006.

[DFM+02] DOGRUSOZ, UGUR, QING-WEN FENG, BRENDAN MADDEN, MICHAEL DOORLEY und ARNE FRICK: *Graph Visualization Toolkits*. IEEE Computer Graphics and Applications, 22:30–37, 2002.

[DGC+04] DOGRUSOZ, UGUR, ERHAN GIRAL, AHMET CETINTAS, ALI CIVRIL und EMEK DEMIR: *A Compound Graph Layout Algorithm for Biological Pathways*. In: *Graph Drawing*, S. 442–447. Springer-Verlag, 2004.

[DH96] DAVIDSON, RON und DAVID HAREL: *Drawing Graphs Nicely using Simulated Annealing*. ACM Transactions on Graphics, 15:301–331, 1996.

[DKMT07] DOGRUSOZ, UGUR, KONSTANTINOS G. KAKOULIS, BRENDAN MADDEN und IOANNIS G. TOLLIS: *On labeling in graph visualization*. Information Sciences, 177:2459–2472, 2007.

[Ead84] EADES, PETER: *A Heuristic for Graph Drawing*. Congressus Numerantium, 42:149–160, 1984.

[EF96] EADES, PETER und QING-WEN FENG: *Multilevel Visualization of Clustered Graphs*. S. 101–112. Springer-Verlag, 1996.

[EGK+01] ELLSON, JOHN, EMDEN R. GANSNER, ELEFTHERIOS KOUTSOFIOS, STEPHEN C. NORTH und GORDON WOODHULL: *Graphviz - Open Source Graph Drawing Tools*. In: *Graph Drawing*, S. 483–484. Springer-Verlag, 2001.

[EH00] EADES, PETER und MAO LIN HUANG: *Navigating Clustered Graphs using Force-Directed Methods*. Journal of Graph Algorithms and Applications, 4:157–181, 2000.

[ELS93] EADES, PETER, XUEMIN LIN und W. F. SMYTH: *A Fast and Effective Heuristic for the Feedback Arc Set Problem*. Information Processing Letters, 47:319–323, 1993.

[Exp08] EXPASY.ORG: *Metabolic Pathways*. `http://www.expasy.ch/cgi-bin/show_thumbnails.pl`, 2008. Website, zuletzt besucht am 14.02.2008.

[FLM94] FRICK, ARNE, ANDREAS LUDWIG und HEIKO MEHLDAU: *A Fast Adaptive Layout Algorithm for Undirected Graphs.* In: *Graph Drawing*, S. 388–403. Springer-Verlag, 1994.

[For04] FORSTER, MICHAEL: *Crossings in Clustered Level Graphs*. Dissertation, Universität Passau, 2004.

[FR91] FRUCHTERMAN, THOMAS M. J. und EDWARD M. REINGOLD: *Graph Drawing by Force-directed Placement*. Software - Practice and Experience, 21:1129–1164, 1991.

[Fre01] FREIVALDS, KARLIS: *Curved Edge Routing*. In: *International Symposium on Fundamentals of Computation Theory*, S. 126–137. Springer-Verlag, 2001.

[FT07] FRISHMAN, YANIV und AYELLET TAL: *Online Dynamic Graph Drawing.* In: *Eurographics/ IEEE-VGTC Symposium on Visualization*, S. 75–82. Eurographics Association, 2007.

[Fur86] FURNAS, GEORGE W.: *Generalized fisheye views*. In: *SIGCHI conference on Human factors in computing systems*, S. 16–23. ACM Press, 1986.

[Fur08] FURBER, JOHN D.: *The 2007 Network of Biological Interactions in Human Aging*. `http://www.LegendaryPharma.com/chartbg.html`, 2008. Website, zuletzt besucht am 14.02.2008.

[GD03] GENC, BURKAY und UGUR DOGRUSOZ: *A Constrained, Force-Directed Layout Algorithm for Biological Pathways*. In: *Graph Drawing*, S. 314–319. Springer-Verlag, 2003.

[GEC98] GERSHON, NAHUM D., STEPHEN G. EICK und STUART K. CARD: *Information visualization*. Interactions, 5:9–15, 1998.

[GKN+03] GINKEL, MARTIN, ANDREAS KREMLING, TORSTEN NUTSCH, ROBERT REHNER und ERNST DIETER GILLES: *Modular Modeling of Cellular Systems with ProMoT/Diva*. Bioinformatics, 19:1169–1176, 2003.

[HHLM99] HARTWELL, LELAND H., JOHN J. HOPFIELD, STANISLAS LEIBLER und ANDREW W. MURRAY: *From molecular to modular cell biology*. Nature, 402:C47–C52, 1999.

[Him95] HIMSOLT, MICHAEL: *Comparing and Evaluating Layout Algorithms within GraphEd*. Journal of Visual Languages and Computing, 6:255–273, 1995.

[HMM00] HERMAN, IVAN, GUY MELANÇON und M. SCOTT MARSHALL: *Graph Visualization and Navigation in Information Visualization: A Survey*. In: *IEEE Transactions on Visualization and Computer Graphics*, S. 24–43. IEEE Computer Society Press, 2000.

[HMW+07] HU, ZHENJUN, JOE MELLOR, JIE WU, MINORU KANEHISA, JOSHUA M. STUART und CHARLES DELISI: *Towards zoomable multidimensional maps of the cell*. Nature Biotechnology, 25:547–554, 2007.

[Keg08] KEGG.ORG: *KEGG: Kyoto Encyclopedia of Genes and Genomes*. `http://www.genome.jp/kegg`, 2008. Website, zuletzt besucht am 14.02.2008.

[Ker05] KERN, ACHIM: *Layoutalgorithmen für hierarchische Graphen*. Diplomarbeit, Universität Stuttgart, 2005.

[KK89] KAMADA, TOMIHISA und SATORU KAWAI: *An Algorithm for Drawing General Undirected Graphs*. Information Processing Letters, 31:7–15, 1989.

[KW01] KAUFMANN, MICHAEL und DOROTHEA WAGNER: *Drawing Graphs: Methods and Models*. Springer-Verlag, 2001.

[LEN05] LI, WANCHUN, PETER EADES und NIKOLA NIKOLOV: *Using Spring Algorithms to Remove Node Overlapping*. In: *Asia Pacific Symposium on Information Visualisation*, Band 45, S. 131–140. ACS, 2005.

[MELS95] MISUE, KAZUO, PETER EADES, WEI LAI und KOZO SUGIYAMA: *Layout Adjustment and the Mental Map*. Journal of Visual Languages and Computing, 6:183–210, 1995.

[Mic98] MICHAL, GERHARD: *Biochemical Pathways: An Atlas of Biochemistry and Molecular Biology*. Wiley-Spektrum, 1998.

[MR02] MUTTON, PAUL und PETER RODGERS: *Spring Embedder Preprocessing for WWW Visualization*. In: *Information Visualization*. IEEE Computer Society Press, 2002.

[Mut99] MUTZEL, PETRA: *Zeichnen von Diagrammen - Theorie und Praxis*. Dissertation, Universität des Saarlandes, 1999.

[OS07] OMOTE, HIROKI und KOZO SUGIYAMA: *Method for Visualizing Complicated Structures Based on Unified Simplification Strategy*. IEICE Transactions on Information and Systems, E90-D:1649–1656, 2007.

[PAC02] PURCHASE, HELEN C., JO-ANNE ALLDER und DAVID A. CARRINGTON: *Graph Layout Aesthetics in UML Diagrams: User Preferences.* Journal of Graph Algorithms and Applications, 6(3):255–279, 2002.

[PCJ96] PURCHASE, HELEN C., ROBERT F. COHEN und MURRAY JAMES: *Validating Graph Drawing Aesthetics*. In: *Graph Drawing*, S. 435–446. Springer-Verlag, 1996.

[PF93] PERLIN, KEN und DAVID FOX: *Pad: An Alternative Approach to the Computer Interface*. In: *Annual Conference on Computer Graphics*, S. 57–72. ACM Press, 1993.

[PV06] PAPADOPOULOS, CHARIS und CONSTANTINOS VOGLIS: *Drawing Graphs using Modular Decomposition*. In: *Graph Drawing*, S. 343–354. Springer-Verlag, 2006.

[Rei02] REISS, THOMAS: *Systeme des Lebens - Systembiologie*. Broschüre, Bundesministerium für Bildung und Forschung (BMBF), 2002.

[RMS97] RYALL, KATHY, JOE MARKS und STUART M. SHIEBER: *An Interactive Constraint-Based System for Drawing Graphs*. In: *Symposium on User Interface Software and Technology*, S. 97–104. ACM Press, 1997.

[San96] SANDER, GEORG: *Layout of Compound Directed Graphs*. Technischer Bericht, Universität des Saarlandes, 1996.

[Sch01] SCHREIBER, FALK: *Visualisierung biochemischer Reaktionsnetze*. Dissertation, Universität Passau, 2001.

[Shn96] SHNEIDERMAN, BEN: *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations*. In: *IEEE Symposium on Visual Languages*, S. 336–343. IEEE Computer Society, 1996.

[SM91] SUGIYAMA, KOZO und KAZUO MISUE: *Visualization of Structural Information: Automatic Drawing of Compound Digraphs*. IEEE Transactions on Systems, Man, And Cybernetics, 21:876–892, 1991.

[SM94] SUGIYAMA, KOZO und KAZUO MISUE: *A Simple and Unified Method for Drawing Graphs: Magnetic-Spring Algorithm*. In: *Graph Drawing*, S. 364–375. Springer-Verlag, 1994.

[SRKC+04] SAEZ-RODRIGUEZ, JULIO, ANDREAS KREMLING, HOLGER CONZELMANN, KATJA BETTENBROCK und ERNST D. GILLES: *Modular analysis of signal transduction networks*. IEEE Control Systems Magazine, 24:35–52, 2004.

[SRMH+06] SAEZ-RODRIGUEZ, JULIO, SEBASTIAN MIRSCHEL, REBECCA HEMENWAY, STEFFEN KLAMT, ERNST DIETER GILLES und MARTIN GINKEL: *Visual setup of logical models of signaling and regulatory networks with ProMoT*. BMC Bioinformatics, 7:506, 2006.

[STT81] SUGIYAMA, K., S. TAGAWA und M. TODA: *Methods for Visual Understanding of Hierarchical System Structures*. IEEE Transactions on Systems, Man, And Cybernetics, 11:109–125, 1981.

[Tam87] TAMASSIA, ROBERTO: *On embedding a graph in the grid with the minimum number of bends*. SIAM Journal on Computing, 16:421–444, 1987.

[Tig08] TIGRIS.ORG: *Java Graph Editing Framework*. `http://gef.tigris.org/`, 2008. Website, zuletzt besucht am 14.02.2008.

[Wal00] WALSHAW, CHRIS: *A Multilevel Algorithm for Force-Directed Graph Drawing*. In: *Graph Drawing*, Band 1984, S. 171–182. Springer-Verlag, 2000.

[WEK04] WIESE, ROLAND, MARKUS EIGLSPERGER und MICHAEL KAUFMANN: *yfiles - Visualization and automatic layout of graphs*. In: *Graph Drawing Software*, S. 173–191. Springer-Verlag, 2004.

[WM95] WANG, XIAOBO und ISAO MIYAMOTO: *Generating Customized Layouts*. In: *Graph Drawing*, S. 504–515. Springer-Verlag, 1995.

[ZSdG07] ZIMÁNYI, ESTEBAN und SABRI SKHIRI DIT GABOUJE: *A New Constraint-based Compound Graph Layout Algorithm for Drawing Biochemical Networks*. In: *Visual Languages for Interactive Computing: Definitions and Formalizations*, S. 407–424. IGI Global, 2007.

Begriffs- und Abkürzungsverzeichnis

API Die *Application Programming Interface* eines Softwaresystems, ist eine Schnittstelle, die anderen Programmen die Anbindung an das System auf Quelltextebene ermöglicht.

Bounding-Box Die *Bounding-Box*, einer Menge von Objekten, ist das kleinste Rechteck, dass alle Objekte dieser Menge beinhaltet.

CORBA Die *Common Object Request Broker Architecture* ist eine Spezifikation, die plattformübergreifende Protokolle und Dienste definiert. CORBA stellt eine Schnittstelle dar, durch die Anwendungen über das Netzwerk zusammenarbeiten können.

DAG Ein *Directed Acyclic Graph* ist ein gerichteter Graph, welcher keine Zyklen enthält.

EGF Der *Epidermal-Growth-Factor* ist ein Protein, welches bei der Einleitung der Mitose als Signalmolekül auftritt. Es stimuliert die Ausbildung einer Reihe von Zelltypen, weshalb es auch in der Zellkultur verwendet wird.

GUI Die *Graphical User Interface* ist eine graphische Benutzeroberfläche, über die ein Anwender mit einem Computerprogramm interagieren kann.

HyCoGLa Das *Hybrid Compound-Graph Layout* ist das in dieser Arbeit entwickelte Zeichenverfahren für Compound-Graphen.

ProMoT Das *Process Modelling Tool* ist ein Programm zur Konstruktion und Manipulation von komplexen technischen und biologischen Systemen.

SBML Die *Systems Biology Markup Language* ist ein, auf XML (Extensible Markup Language) basierendes, Datenaustauschformat zur logischen Auszeichnung biochemischer Modelle.

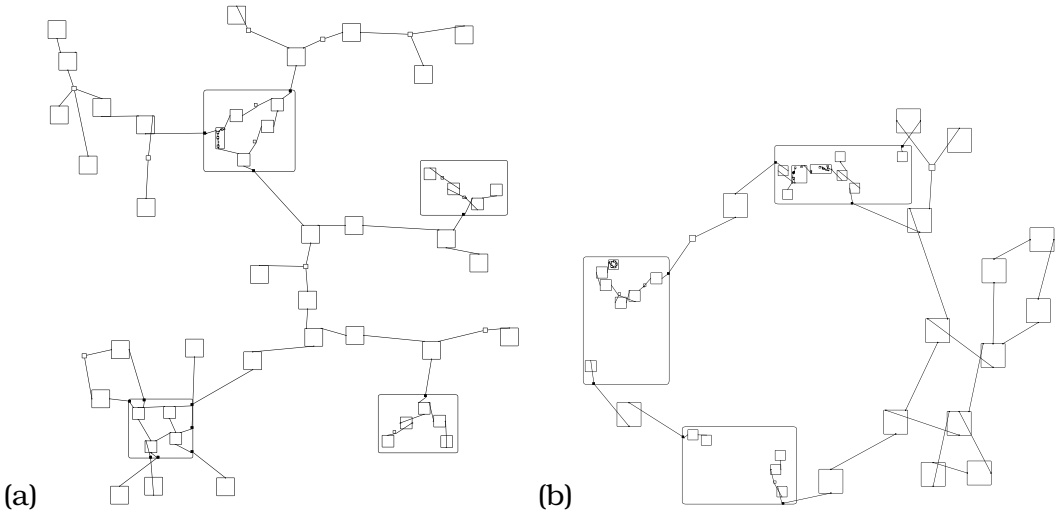
UML Die *Unified Modelling Language* ist eine standardisierte Sprache zur Spezifikation, Visualisierung, Konstruktion und Dokumentation von Software.

ZUI Die *Zoomable User Interface* ist eine GUI, welche dem Benutzer die Möglichkeit bietet, auf einzelne Elemente oder die gesamte Oberfläche zu zoomen. Sie eignet sich daher sehr gut zur Visualisierung komplexer und strukturierter Daten.

Anhang A

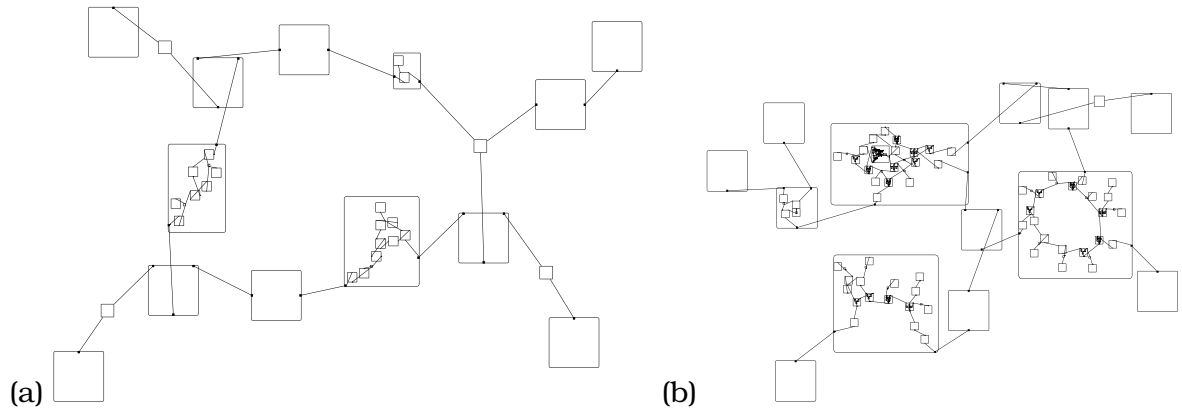
Abbildungen und Tabellen

A.1 Layouts verschiedener Testgraphen



(a) (b)

Abbildung A.1: Mit HyCoGLa erzeugte Layouts der Testgraphen (a) 2 und (b) 3 aus Tabelle A.1 . Beide Testgraphen wurden aus [DGC+04] adaptiert.



(a) (b)

Abbildung A.2: Mit HyCoGLa erzeugte Layouts der Testgraphen (a) 6 und (b) 7 aus Tabelle A.1.

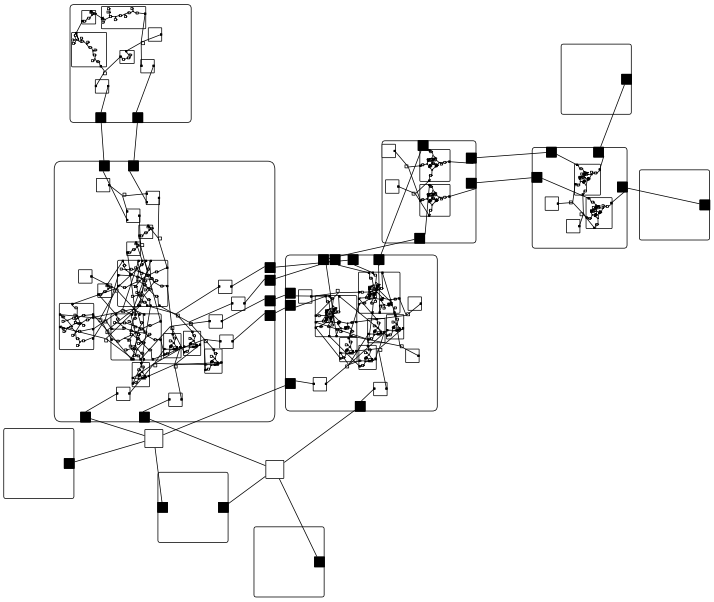


Abbildung A.3: Mit HyCoGLa erzeugtes Layout des Modells eines EGF-Rezeptor-Netzwerks (Testgraph 11 aus Tabelle A.1).

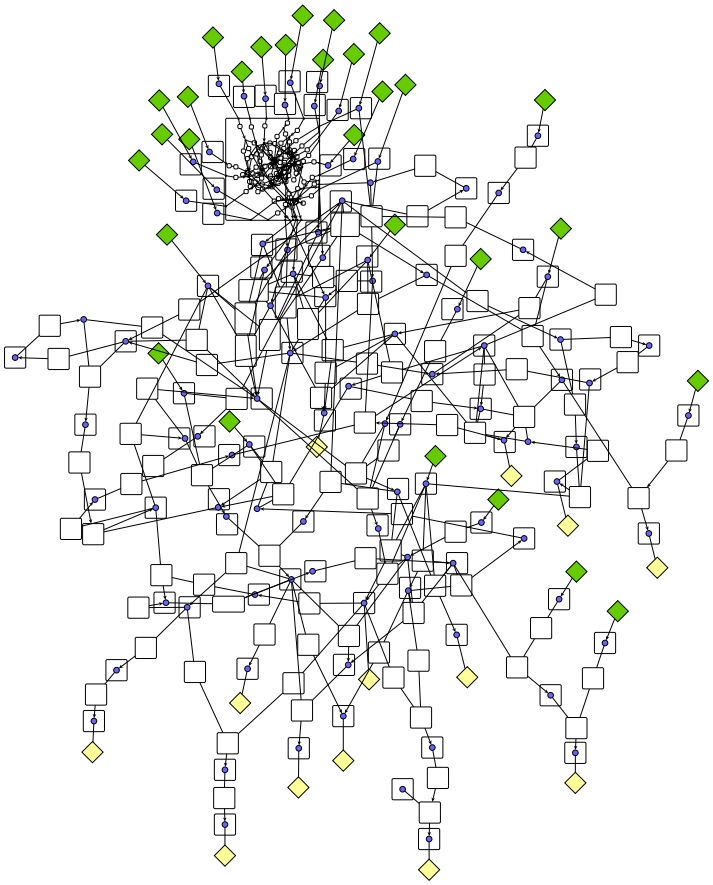


Abbildung A.4: Mit HyCoGLa, unter Beachtung semantischer Informationen, wie z.B. Reaktionsrichtungen, erzeugtes Layout des EGFR/ErbB-Netzwerks (Testgraph 8 aus Tabelle A.1).

A.2 Ergebnisse der durchgeführten Experimente

Graph	#V	#E	#SG	D	TI	TL	TC	TO	TU	T	#EEC	#EVC	%L
1	61	64	1	2	0 ,9	1,9	0,6	0,1	1,1	4,7	2	28	135,3
					1,0	*0,6*	*0,4*	*0,1*	*0,0*	*4*	*3*	*38*	*156,3*
2	84	81	5	3	1,5	4,3	1,0	0,1	3,1	10,1	4	21	73,5
					1,4	*1,3*	*0,4*	*0,2*	*0,1*	*3,4*	*3*	*22*	*77,6*
3	79	73	7	4	1,4	5,8	0,8	0,1	4,1	6,5	6	43	66,4
					1,7	*1,9*	*0,2*	*0,0*	*0,0*	*3,9*	*5*	*40*	*119,4*
4	26	27	2	2	0,5	1,6	0,2	0,0	1,0	3,3	2	5	58,0
					0,4	*0,5*	*0,1*	*0,0*	*0,1*	*1,2*	*0*	*1*	*48,8*
5	27	26	2	3	0,6	2,1	0,1	0,0	1,3	4,1	2	8	72,8
					0,6	*0,6*	*0,1*	*0,0*	*1,4*	*0,0*	*2*	*8*	*124,5*
6	44	43	3	2	0,8	2,4	0,4	0,0	1,5	5,1	1	29	114,1
					0,9	*0,9*	*0,1*	*0,0*	*0,0*	*2,0*	*5*	*20*	*144,6*
7	117	133	4	3	4,2	11,7	1,9	0,3	7,7	25,9	12	81	128,3
					5,7	*2,2*	*0,6*	*0,2*	*0,1*	*8,7*	*18*	*73*	*170,1*
8	480	612	1	2	6,9	46,1	49,0	17,0	35,6	154,5	605	602	319,4
					5,5	*1,6*	*24,0*	*23,9*	*0,1*	*55,0*	*594*	*604*	*312,7*
9	469	593	6	2	33,8	64,5	56,9	4,8	50,1	210,2	302	494	279,4
					27,8	*5,6*	*4,2*	*3,5*	*0,1*	*41,2*	*363*	*495*	*269,4*
10	420	491	16	3	38,2	87,6	28,3	6,1	39,2	199,3	220	399	213,0
					37,0	*7,6*	*3,6*	*2,7*	*0,1*	*50,9*	*280*	*402*	*231,0*
11	937	994	73	4	20,7	83,2	8,9	2,9	83,6	199,4	158	297	41,9
					25,2	*8,2*	*2,2*	*0,9*	*0,1*	*36,6*	*166*	*351*	*53,1*

Tabelle A.1: Ergebnisse der Experimente zu Laufzeit und Darstellungsqualität von HyCoGLa im Vergleich mit einem lokalen Zeichenverfahren: Es wurden 11 Testgraphen ausgewählt, die sich in Knotenanzahl (#*V*), Kantenanzahl (#*E*), der Anzahl an Subgraphen (#*SG*) und der Verschachtelungstiefe (*D*) unterscheiden. Zur Bewertung der Laufzeit wurden die benötigten Zeiten (in Sekunden) zum Aufbau des Inklusionsbaums (*TI*), zur Erzeugung der Layoutobjekte (*TL*), zur eigentlichen Berechnung des Layouts (*TC*), zur Entfernung von verbliebenen Knotenüberlappungen (*TO*) und zur Aktualisierung der graphischen Repräsentationen der Knoten und Kanten (*TU*) gemessen. Die Gesamtlaufzeit der Verfahren ist mit *T* bezeichnet. Zur Bewertung der Darstellungsqualität wurde die Anzahl von Kanten-Kanten-Kreuzungen (#*EEC*), die Anzahl von Kanten-Knoten-Kreuzungen (#*EVC*) und die Abweichung der Summe der Kantenlängen von der Summe der idealen Kantenlängen (in Prozent) (%*L*) der berechneten Layouts bestimmt. Die Werte für das lokale Verfahren sind kursiv unter den Werten von HyCoGLa eingetragen.

A.3 Programme zur Modellierung und Visualisierung biologischer Netzwerke

Programm	**URL**	**Zeichenverfahren**	**Bemerkungen**
BioLayout	http://cgg.ebi.ac.uk/services/biolayout	- kräftebasiert	
Biological Networks	http://biologicalnetworks.net/	- kräftebasiert - schichtenbasiert - zirkulär	
BioMaze	http://cs.ulb.ac.be/research/biomaze/vbm	- kräftebasiert - schichtenbasiert (C2GL-Verfahren)	- erweiterbares auf Eclipse basierendes Framework - folgt biochemischen Konventionen - Behandlung von gerichteten Compound-Graphen
BioPath	http://www.molecular-networks.com/databases/biopath.html	- schichtenbasiert	- inspiriert vom Poster „Biochemical Pathways“ - stellt Ergebnisse von Datenbankabfragen dynamisch dar - beachtet Konventionen aus der Biochemie
CADLIVE	http://www.cadlive.jp	- kräftebasiertes (Simulated Annealing) Grid-Layout	- Darstellung von Clustern
Cell Designer	http://www.celldesigner.org	- siehe yEd	- verwendet yFiles
Cytoscape	http://www.cytoscape.org	- siehe yEd - siehe Osprey	- verschiedene Plugins wie BubbleRouter oder Cerebral zur Erweiterung der Layoutmöglichkeiten und Interaktion - verwendet yFiles
JDesigner	http://www.sys-bio.org/software/jdesigner.htm	kräftebasiert	
Osprey	http://biodata.mshri.on.ca/osprey	- kräftebasiert - verschiedene zirkuläre Techniken	
Pathway Studio	http://www.ariadnegenomics.com/products/pathway-studio	- kräftebasiert - schichtenbasiert	
Patika	http://www.cs.bilkent.edu.tr/patikaweb	- kräftebasiert	- Darstellung von Compound-Graphen - Darstellung der subzellulären Lokalisation - verwendet TomSawyer Layout
SimViz	http://projects.eml.org/bcb/software/software/Ulla/SimWiz	- Mischung aus zirkulär und schichtenbasiert	- Identifikation von Subgraphen - Layout der Subgraphen mit geeignetem Verfahren - Kompositon der Subgraphen - verwendet yFiles
Vanted	http://vanted.ipk-gatersleben.de	- kräftebasiert - zirkulär - schichtenbasiert	- experimentelle Daten sind auf Elemente der dargestellten Graphen abbildbar - verwendet Graphviz
VisAnt	http://visant.bu.edu	- kräftebasiert - zirkulär	- unterstützt die Darstellung von Compound-Graphen
yEd	http://www.yworks.com/en/products/yed/about.html	- kräftebasiert - zirkulär - schichtenbasiert - orthogonal - Baumlayout	- Graph-Editor von yWorks - für alle Arten von Graphen geeignet - basiert auf yFiles

Tabelle A.2: Gegenüberstellung einer Auswahl von Programmen zum Erstellen, Modifizieren und Visualisieren von Modellen biologischer Netzwerke.