Neue Methoden für klassische Grapheinbettungsprobleme Orthogonale Zeichnungen & bedingte Planarität[1]

Thomas Bläsius[2]

Abstract: Möchte man Graphen automatisiert möglichst anschaulich darstellen, so stößt man häufig auf anspruchsvolle Optimierungsprobleme, die diesen Visualisierungsproblemen zugrunde liegen. Ein gutes theoretisches Verständnis dieser Kernprobleme ist essentiell für den Entwurf von Algorithmen, die sowohl performant sind, als auch qualitativ hochwertige Visualisierungen generieren.

Die in meiner Dissertation [Bl15] untersuchten Kernprobleme fallen in zwei Kategorieren und haben gemeinsam, dass sie zwar schon verhältnismäßig intensiv erforscht wurden (und damit als klassisch angesehen werden können), wohingegen gewissen zentrale Fragestellungen noch offen sind bzw. waren. Bei der ersten Kategorie handelt es sich um die Knickminimierung in orthogonalen Zeichnungen. Konkret wird der Fall betrachtet, dass die Topologie der Zeichnung nicht schon in der Eingabe gegeben ist (was mehr Freiheiten und damit bessere Ergebnisse zulässt, das Problem aber signifikant schwerer macht), sowie der Fall, dass die Knotengrade 4 überschreiten dürfen. Die zweite Kategorie beschäftigt sich mit dem Basisfall der Kreuzungsminimierung, also mit der Frage, ob ein Graph ganz ohne Kreuzungen (d.h. planar) gezeichnet werden kann. Dabei werden jedoch Szenarien betrachtet, in denen nicht nur ein einzelner Graph für sich visualisiert werden soll, sondern beispielsweise ein Graph zusammen mit einer Gruppierung (clustering) der Knoten oder zusammen mit einem zweiten Graphen auf der gleichen Knotenmenge (zum Vergleich verschiedener oder einer sich verändernden Relation auf den gleichen Objekten).

Bei all diesen grundlegenden Problemen gehe ich der Frage nach, ob und unter welchen Voraussetzungen sie sich effizient (d.h. in polynomieller Zeit) lösen lassen.

1 Der Nutzen von Graphzeichnungen

In einer Welt, in der Daten im Übermaß verfügbar sind, ist es wichtig, über geeignete Verfahren zu verfügen, um vorhandenen Rohdaten analysieren und interpretieren zu können. In vielen Fällen kann diese Aufgabe von Computern übernommen werden. Das vermutlich populärste Beispiel sind Navigationssysteme, die kürzeste Wege in einem Straßengraphen berechnen können. Ein anderes Beispiel wäre die Berechnung eines minimalen Schnittes zur Lokalisierung von Schwachpunkten in Transport-, Energie- oder Computernetzwerken. Auf der anderen Seite gibt es aber auch Fälle, in denen eine solche direkte algorithmische Lösung nicht anwendbar ist. Dies ist beispielsweise der Fall, wenn die Fragestellung zu wage ist um sie sinnvoll zu formalisieren oder wenn es schlicht keinen Algorithmus gibt, der das zugehörige Problem in angemessener Zeit löst. Eine weitere Schwierigkeit ergibt sich dadurch, dass der Nutzer eine Lösung eventuell nur dann akzeptiert, wenn er sie auch nachvollziehen kann.

[1] Englischer Titel der Dissertation: “New Approaches to Classic Graph-Embedding Problems – Orthogonal Drawings & Constrained Planarity”

[2] Hasso Plattner Institut, thomas.blaesius@hpi.de

Ein Beispiel für eine schwer zu formalisierende Fragestellung ist das auffinden eines Anführers innerhalb einer Gruppe, basierend auf der Kommunikation zwischen den Mitgliedern. Ein möglicher Lösungsansatz besteht darin, möglichst zentrale Knoten in dem Kommunikationsgraphen zu finden. Dabei hängt das Resultat aber stark von dem verwendeten Zentralitätsmaß ab. Darüber hinaus ist eine Lösung für den Benutzer gegebenenfalls nicht akzeptabel, wenn er nicht weiß, wie diese Lösung zustande kommt. Ähnlich verhält es sich, wenn Infrastruktur mit dem Ziel erweitert werden soll, die Ausfallsicherheit unter Einhaltung eines Budgets zu maximieren. Da in solchen Fällen politische Entscheidungen eine wichtige Rolle spielen, muss der Benutzer in der Lage sein, eine potentielle Lösung sowohl verstehen als auch kommunizieren zu können. Darüber hinaus enthalten vieler solcher praxisnaher Fragestellungen grundlegende NP-schwere Probleme, wie zum Beispiel das STEINER BAUM Problem [GJ79], und können daher nicht in polynomieller Zeit gelöst werden (vorausgesetzt $P \neq NP$).

Man kann also nicht jede Fragestellung mithilfe eines Algorithmus direkt beantworten. Das heißt allerdings nicht, dass Computer bei der Suche nach einer Antwort nicht als Werkzeug dienen können. Architekten machen beispielsweise intensiven Gebrauch von Computern, auch wenn eine automatische Generierung eines vollständigen Gebäudeplans nicht realistisch ist. Der Schlüssel besteht dabei darin, dass der Benutzer in den Prozess der Lösungsfindung eingebunden ist. Dabei ist es sogar möglich, dass dem Benutzer erst während dieses Prozesses das genaue Ziel klar wird; eine formale Problemstellung ist also nicht mehr notwendig. Die Beteiligung des Nutzers sorgt darüber hinaus für eine nachvollziehbare Lösung. Schlussendlich kann die menschliche Intuition helfen, die entscheidenden Schritte zu finden, die zur Lösung eines NP-schweren oder sogar unentscheidbaren Problems (z.B. beim maschinengestützten Beweisen) führen.

Da Menschen nicht besonders gut darin sind sehr große abstrakte Datenmengen zu verstehen, müssen diese Rohdaten mit einer visuellen Repräsentation ergänzt werden; siehe auch Abbildung 1. So kann man dem Benutzer beispielsweise eine Zeichnung des Kommunikationsgraphen einer Gruppe von Menschen präsentieren. Anhand dieser Zeichnung kann der Benutzer einen Überblick über die Daten gewinnen, selbstständig entscheidende Personen identifizieren oder für bzw. gegen mögliche Anführer argumentieren, die von einem Algorithmus vorgeschlagen werden. Dieses Beispiel zeigt einen Anwendungsfall für die Visualisierung von Informationen im Allgemeinen sowie für das Zeichnen von Graphen im speziellen: Der Benutzer möchte, basierend auf ihm unbekannten Daten, einen Kenntnisgewinn erzielen.

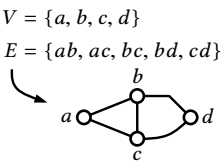

Abbildung 1: Ein Graph bestehend aus Knotenmenge V und Kantenmenge E (abstrakte Rohdaten) sowie eine visuellen Darstellung.

Eine leicht anders gelagerte Anwendung ist die, dass der Benutzer die Daten bereits kennt und zu interpretieren weiß, sein Wissen aber anderen zugänglich machen möchte. Das ist beispielsweise relevant für Lehrer, Museen oder Journalisten, die ihr Wissen an Schüler, Besucher bzw. Leser in anschaulicher Form weitergeben wollen. UML-Diagramme, welche die Struktur eines Softwareprojekts visualisieren, fallen ebenfalls in diese Kategorie.

Das Zeichnen von Graphen mit dem Zweck anderen Menschen Informationen zugänglich zu machen hat lange Tradition. Die ältesten bekannten Zeichnungen von Graphen sind mindestens 900 Jahre alt [Li14]. Abbildung 2 zeigt die Zeichnung eines Baumes (zusammenhängender und kreisfreier Graph) von 1866, der eine biologische Kategorisierung darstell.

Aus algorithmischer Sicht hat das Zeichnen von Graphen zunächst im Zusammenhang mit dem Entwurf von integrierten Schaltkreisen Beachtung erlangt [AGR70]. Dabei besteht das Problem darin, die logische Struktur eines Computerchips auf eine tatsächliche physische Struktur zu übertragen. Dabei müssen Komponenten auf Positionen auf dem Chip und Verbindungen auf Leitungen zwischen den entsprechenden Komponenten abgebildet werden. Man kann also sagen, dass ein gegebener Graph, der die logische Struktur des Computerchips repräsentiert, gezeichnet werden muss. Eine ähnliche aber aktuellere Anwendung ist der Entwurf von Biochips, die im Prinzip ein Miniaturlabor darstellen, auf dem mehrere Reaktionen simultan stattfinden können; siehe Abbildung 3.

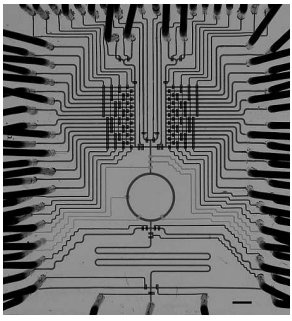

Abbildung 3: Ein Biochip der bis zu 1024 Reaktionen parallel ausführen kann [Wa09].

Zusammenfassend kann man also sagen, dass das Zeichnen von Graphen, neben der Visualisierung von Informationen, auch dazu genutzt werden kann, eine gegebene logische Struktur in eine physisches Objekt zu übertragen. Trotz der unterschiedlichen Anwendungen sind sich die Optimierungsziele oftmals erstaunlich ähnlich. Beispielsweise verschlechtern Kantenkreuzungen die Lesbarkeit von Zeichnungen erheblich [PCJ96]. In Schaltkreisen führen sie dazu, dass die entsprechenden Leitungen auf unterschiedlichen Lagen verlegt werden müssen. Damit ist das Konzept der Planarität (also der kreuzungsfreien Zeichenbarkeit) aus der Sicht verschiedener Anwendungen relevant. Ähnlich verhält es sich bei sogenannten orthogonalen Zeichnungen. Dies sind Zeichnungen, bei denen Kanten durch Sequenzen von horizontalen und vertikalen Strecken dargestellt werden. Diese Art von Zeichnungen werden sehr häufig im Chipentwurf verwendet (die meisten Kanten bei dem Biochip in Abbildung 3 sind orthogonal), sind aber auch bei der Visualisierung von UML-Diagrammen sehr beliebt.

Einige Graphzeichnen-Probleme sind trotz ihrer langen Historie und ihrer grundlegenden Bedeutung bislang ungelöst. Das Ziel meiner Arbeit ist es, die Forschung an solchen klassischen Zeichenproblemen voranzutreiben. Dabei steht die Entwicklung effizienter Algorithmen (mit polynomieller Laufzeit) im Vordergrund. Stellt sich für ein Problem heraus, dass es NP-schwer ist, so stelle ich diesem negativen Resultat immer auch positive Ergebnisse gegenüber. Dazu gebe ich beispielsweise Algorithmen an, deren Laufzeit nur exponentiell bezüglich eines oder mehrerer Parameter ist.

Die Arbeit gliedert sich in zwei Teile. Im ersten Teil werden die bereits erwähnten orthogonale Zeichnungen betrachtet. Im zweiten Teil geht es um die verallgemeinerten Planaritätsbegriffe c-Planarität (*engl.* clustered planarity) sowie simultane Planarität.

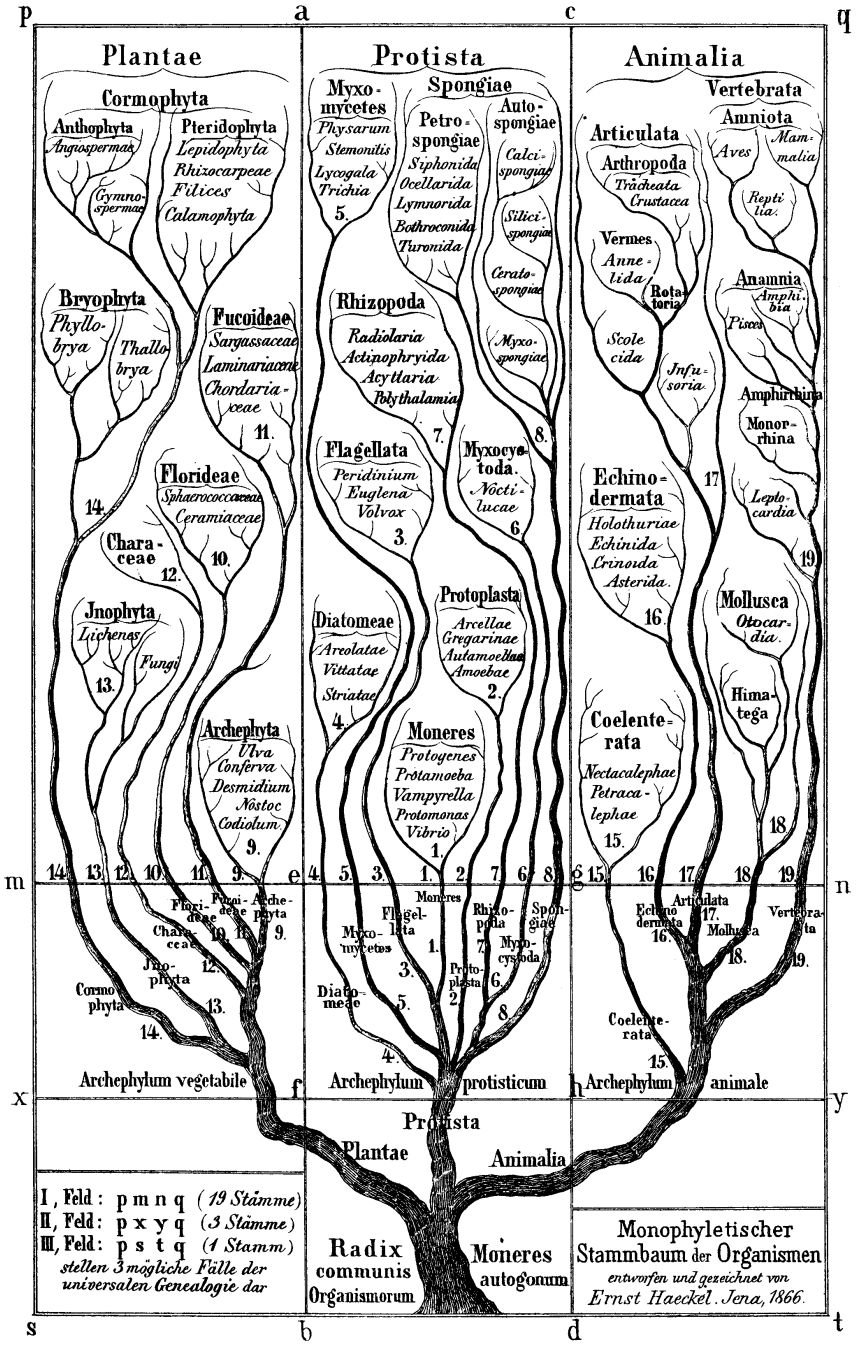

Abbildung 2: Der „Monophyletische Stammbaum der Organismen“ von Ernst Haeckel (1866). Die drei Kategorieren auf dem obersten Level sind Pflanzen (Plantae), Einzeller (Protista) und Tiere (Animalia). In der oberen rechten Ecke sind beispielsweise Vögel (Aves), Reptilien (Reptilia) und Säugetiere (Mammalia).

2 Orthogonale Zeichnungen

Historisch wurde die automatisierte Erstellung von orthogonalen Zeichnungen zunächst im Zusammenhang mit dem Entwurf von integrierten Schaltkreisen erforscht. Die naheliegendsten Optimierungskriterien waren dabei die benötigte Fläche, die sich direkt auf die Größe des Mikrochips überträgt, sowie die Gesamtkantenlänge. Etwas später kam die Knickzahl als mögliches Optimierungskriterium hinzu [St80]. Dies wurde zum einen motiviert durch Kosten, die an Knicken entstehen, wenn Informationen mittels Licht oder Mikrowellen übertragen werden, zum anderen durch „aufgeräumtere“ Zeichnungen.

Der Aspekt der aufgeräumteren Zeichnung erlangt größere Bedeutung, wenn es darum geht, Graphen zum Zweck der Netzwerkanalyse anschaulich zu visualisieren. Dank der klaren und strukturierten Darstellung, die ausschließlich vertikale und horizontale Strecken mit sich bringen, gehören orthogonale Zeichnungen bis heute zu den meistverwendeten Zeichenstilen bei der Visualisierung von kleinen bis mittelgroßen Netzwerken.

Üblicherweise wird der Eingabegraph bei der Erzeugung orthogonaler Zeichnungen als planar vorausgesetzt. Für nicht-planare Graphen wird zunächst eine sogenannte Planarisierung mit möglichst wenigen Kreuzungen berechnet. Diese kann dann wie ein planarer Graph behandelt werden. Da man jeden Gitterpunkt im orthogonalen Gitter nur in vier verschiedene Richtungen verlassen kann, schränkt man sich bei orthogonalen Zeichnungen häufig auf Graphen mit Maximalgrad 4 ein; siehe auch Abbildung 4. Eine Möglichkeit auch mit höhergradigen Knoten umzugehen bietet das sogenannte Kandinskymodell, das in Abschnitt 2.2 näher besprochen wird; siehe auch Abbildung 5.

2.1 Graphen mit Maximalgrad 4

Trotz drei Jahrzehnten intensiver Forschung zu orthogonalen Zeichnungen blieben eine Reihe Fragen lange unbeantwortet. Beispielsweise ist seit 1994 bekannt, dass jeder 4-planare Graph (planar, mit Maximalgrad 4) eine orthogonale Zeichnung mit zwei Knicken pro Kante besitzt [BK98], es jedoch NP-schwer ist zu entscheiden, ob ein gegebener Graph ohne Knicke gezeichnet werden kann [GT01]. Eine naheliegende Frage ist, wie sich die Komplexität zwischen zwei Knicken und keinem Knick pro Kante verhält, also wenn man einen Knick pro Kante erlaubt. Diese Frage konnten wir erst 2010 positiv beantworten, indem wir einen einen effizienten Algorithmus angegeben haben, der entscheidet, ob ein gegebener 4-planarer Graph eine orthogonale Zeichnung besitzt, bei der jede Kante maximal einen Knick enthält [Bl14]. Daran anschließend bieten sich die folgenden zwei Fragestellungen an. Was passiert wenn man nur für manche Kanten fordert, dass diese keinen Knick haben? Falls es keine Zeichnung mit einem Knick pro Kante gibt, kann man dann die Anzahl der Knicke die darüber hinaus gehen effizient minimieren?

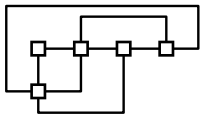

Abbildung 4: Eine orthogonale Zeichnung mit bis zu vier Knicken pro Kante und neun Knicken insgesamt.

Graphen mit starren Kanten. Um die erste dieser beiden Fragen zu beantworten und damit die oben aufgezeigte Komplexitätslücke weiter zu schließen, untersuche ich den Fall, dass manche Kanten nicht geknickt werden dürfen, wohingegen alle anderen mindestens einen Knick erlauben. Kanten die nicht geknickt werden dürfen nenne ich *starr*, die anderen Kanten sind *flexibel*. Beachte, dass diese Problemstellung zwischen dem oben genannten NP-schweren Fall (alle Kanten starr [GT01]) und dem effizient lösbaren Fall (alle Kanten flexibel [Bl14]) liegt. Ich gebe einen parametrisierten Algorithmus an, dessen Laufzeit nur exponentiell in der Anzahl starren Kanten ist. Genauer erhalte ich die Laufzeit $O(2^k \cdot n \cdot T_{\text{flow}}(n))$. Dabei ist k die Anzahl der starren Kanten und $T_{\text{flow}}(n)$ die nötige Zeit um einen maximalen Fluss in einem planaren Flussnetzwerk der Größe n mit mehreren Quellen und mehreren Senken zu berechnen (was in polynomieller Zeit machbar ist). Der Algorithmus ist also ein sogenannter FPT-Algorithmus (*fixed-parameter tractable*). Tatsächlich ist die Laufzeit sogar polynomiell, wenn die Anzahl der starren Kanten in $O(\log n)$ liegt.

Auf der anderen Seite zeige ich, dass das Problem NP-schwer wird, sobald der Graph $O(n^\varepsilon)$ ($\varepsilon > 0$) starre Kanten enthält, selbst wenn diese gleichmäßig über den Graphen verteilt sind, also paarweise Abstand $\Omega(n^{1-\varepsilon})$ haben. Dies beinhaltet beispielsweise den sehr eingeschränkten Fall, dass die starren Kanten ein Matching bilden.

Knickminimierung. Bei den bislang erwähnten Graphzeichen-Problemen handelt es sich um Entscheidungsprobleme, bei denen nur überprüft wird, ob es eine Zeichnung mit den gewünschten Eigenschaften gibt. Existiert beispielsweise keine knickfreie Zeichnung, so würde man stattdessen gerne eine Zeichnung mit möglichst wenigen Knicken ausgeben. Das ist aber leider NP-schwer, da es bereits schwer ist zu testen, ob es ohne Knicke geht. Da man aber effizient testen kann, ob der Graph mit einem Knick pro Kante gezeichnet werden kann, besteht die Hoffnung, dass man auch die Anzahl der Knicke, die über den ersten Knick pro Kante hinausgehen effizient minimieren kann.

Durch den Beweis einiger struktureller Eigenschaften von orthogonalen Zeichnungen mit einem Knick pro Kante gelingt es mir, einen solchen Algorithmus anzugeben. Der Algorithmus funktioniert auch dann noch, wenn man jeder Kante eine individuelle konvexe Kostenfunktion zuweist (vorausgesetzt, der erste Knick ist kostenlos). Damit gebe ich den ersten effizienten Algorithmus zur Knickminimierung in orthogonalen Zeichnungen bei variabler Topologie an, der beliebige 4-planare Graphen als Eingabe erlaubt. Darüberhinaus ist dieser Algorithmus optimal in dem Sinne, dass das Weglassen einer der Forderungen (konvexe Kostenfunktionen und erster Knick pro Kante ist kostenlos) das Problem NP-schwer macht.

2.2 Kandinskyzeichnungen

Im Kandinskymodell werden Knoten als Quadrate fester Größe dargestellt und mehrere Kanten dürfen einen Knoten in dieselbe Richtung verlassen; siehe beispielsweise

Abbildung 5. Die Einschränkung auf Graphen mit Maximalgrad 4 besteht damit nicht mehr. Das Kandinskymodell wurde bereits 1995 vorgestellt [FK95]. Dabei wurde auch ein Algorithmus angegeben, der die Anzahl der Knicke minimiert, unter der Voraussetzung, dass die Topologie der Zeichnung bereits festgelegt ist (d.h. die zyklische Ordnung der Kanten um jeden Knoten ist gegeben). Wenig später stellte sich heraus, dass der Algorithmus einen Fehler enthält [Ei03]. Seither entstanden zahlreiche, meist approximative oder heuristische Verfahren, die Kandinskyzeichnungen generieren oder das Modell, zum Beispiel auf nicht-planare Graphen, erweitern [FK97].

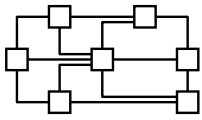

Abbildung 5: Die größeren Knoten im Kandinskymodell erlauben Knoten mit hohem Grad, machen aber die Knickminimierung schwieriger.

Indem ich zeige, dass Knickminimierung im Kandinskymodell NP-schwer ist, beantworte ich die grundlegende Frage nach der Komplexität dieses Problems. Darüberhinaus betrachte ich seine parametrisierte Komplexität bezüglich verschiedener Parameter. Dies liefert insbesondere einen polynomiellen Algorithmus für serien-parallele Graphen (Laufzeit $O(n^3)$), einen subexponentiellen Algorithmus im Allgemeinen (Laufzeit $2^{O(\sqrt{n}\log n)}$), sowie einen FPT-Algorithmus, wobei die Verzweigungsweite des Graphen plus die maximale Facettengröße als Parameter dient. Damit beantworte ich nicht nur eine seit zwanzig Jahren offene Frage, sondern gebe auch neue algorithmische Lösungsansätze für dieses NP-schwere Problem.

3 Planarität mit Nebenbedingungen

In diesem Teil meiner Arbeit geht es um zwei Probleme, bei denen man neben dem Graphen selbst auch zusätzliche Informationen darstellen will. Bei dem Problem der c-Planarität besteht diese zusätzliche Information aus einer Gruppierung (*engl.* clustering) der Knoten. Bei dem Problem der simultanen Planarität soll die Veränderung eines dynamischen Graphen über die Zeit visualisiert werden.

3.1 C-Planarität

Ist zu dem Graphen noch eine Gruppierung der Knoten gegeben (beispielsweise die Einteilung von Klassen eines Softwareprojekts in Pakete), so kann es wünschenswert sein, neben dem Graphen selbst, auch diese Gruppierung mithilfe von Regionen darzustellen, wie es beispielsweise in Abbildung 6 zu sehen ist. Neben Kreuzungen zwischen Kanten, kann es in einer solchen Zeichnung auch zu Kreuzungen zwischen unterschiedlichen Regionen oder Kreuzungen zwischen Kanten und Regionen kommen. Damit erhält man eine Verallgemeinerung des Konzeptes der Planarität: Ein gruppierter Graph heißt *c-planar* (*engl.* clustered planar), wenn

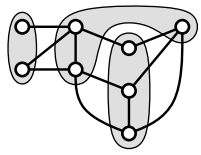

Abbildung 6: Eine c-planare Visualisierung eines Graphen dessen Knoten in drei Gruppen unterteilt sind.

eine Zeichnung existiert, in der keiner dieser drei Kreuzungstypen auftaucht. Das Problem, einen gegebenen gruppierten Graphen auf c-Planarität zu testen, wurde bereits 1986 das erste Mal betrachtet [Le89]. Trotz zahlreicher Arbeiten zu dem Thema konnten seither nur Spezialfälle gelöst werden. Die Komplexität des allgemeinen Problems ist nach wie vor ungeklärt.

Indem ich eine neue Datenstruktur zusammen mit einer Charakterisierung für c-Planarität angebe, kann ich zeigen, dass c-Planarität auf ein bedingtes Einbettungsproblem hinausläuft. Dabei stellt sich die Frage, ob ein gegebener planarer Graph eine planare Zeichnung besitzt, wenn man die möglichen Ordnungen von Kanten um Knoten herum einschränkt. Mithilfe dieser neuen Sichtweise zeige ich, dass sich diverse vorherige, auf den ersten Blick sehr unterschiedliche Resultate, mit ähnlichen Techniken beweisen lassen. Neben der Vereinheitlichung und Vereinfachung existierender Ergebnisse gebe ich effiziente Algorithmen für einige bislang offene Fälle an. Darunter fällt ein effizienter Algorithmus für den Fall, dass jeder Cluster maximal fünf ausgehende Kanten hat, sowie ein Algorithmus dafür, dass jeder Cluster, sowie das Komplement jeden Clusters aus maximal zwei Zusammenhangskomponenten besteht.

3.2 Simultane Planarität

Bei der Visualisierung dynamischer Graphen möchte man neben der Graphstruktur auch die Veränderung dieser Struktur zwischen verschiedenen Zeitpunkten verstehen. Daraus ergibt sich das Problem der simultanen Visualisierung, bei der mehrere Graphen so gezeichnet werden sollen, dass ihr gemeinsamer, unveränderter Teil gleich dargestellt ist; siehe beispielsweise Abbildung 7. Dabei soll weiterhin jeder der Graphen für sich eine möglichst übersichtliche Zeichnung haben. Wählt man die Kreuzungen zwischen Kanten als vorwiegendes Ästhetikkriterium, so erhält man das Konzept der simultanen Planarität. Das Beispiel in Abbildung 7 zeigt einen dynamischen Graphen zu zwei unterschiedlichen Zeitpunkten, wobei die fett gezeichneten Kanten unverändert geblieben sind. Beachte, dass jede Zeichnung für sich gesehen planar ist und der unveränderte Teil in beiden Zeichnungen gleich dargestellt wird. Das Graphenpaar ist also simultan planar. Man beachte, dass die Überlagerung der beiden Zeichnungen nicht planar sein muss (und es in diesem Beispiel auch nicht ist). Eine alternative Formulierung der simultanen Planarität kann damit wie folgt lauten. Gegeben ein Graph in dem jede Kante schwarz (gemeinsame Kante), rot (exklusive Kante des ersten Graphen) oder grün (exklusive Kante des zweiten Graphen) gefärbt ist, gibt es eine Zeichnung, bei der es ausschließlich Kreuzungen zwischen roten und grünen Kanten gibt?

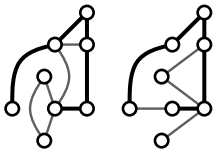

Abbildung 7: Zwei Graphen auf der gleichen Knotenmenge, die sich den fett gezeichneten Subgraph teilen.

Aus algorithmischer Sicht ist die simultane Planarität nah verwandt mit der c-Planarität [AL14, Sc13] und es ist ebenfalls ein offenes Problem, ob man zwei gegebene Graphen effizient auf simultane Planarität testen kann. Für diverse Sonderfälle konnte diese Frage allerdings positiv beantwortet werden.

Dabei wurde bisher meist angenommen, dass der gemeinsame Teilgraph (fett und schwarz in Abbildung 7) zusammenhängend ist. Dies vereinfacht das Problem der simultanen Planarität insofern, dass es genügt, die beiden Graphen so einzubetten, dass die zyklischen Ordnungen der gemeinsamen Kanten um Knoten konsistent sind. Besteht der gemeinsame Graph aus mehreren Zusammenhangskomponenten, so muss man darüber hinaus konsistente relative Positionen der Komponenten zueinander sicherstellen.

Ich gehe zunächst den umgekehrten Weg und löse die Fälle, in denen die zyklischen Ordnungen keine Rolle spielen, man also nur für konsistente relative Lagen sorgen muss. Dies ist dann der Fall, wenn der gemeinsame Graph eine Menge von Kreisen und Pfaden ist oder man die Einbettung jeder Zusammenhangskomponente in der Eingabe festlegt. Die daraus entstehenden Techniken kombiniere ich mit bereits existierenden, sowie neu von mir entwickelten Verfahren zur Sicherstellung konsistenter zyklischer Ordnungen. Daraus ergibt sich insbesondere ein effizienter Algorithmus für den Fall, dass jede Zusammenhangskomponente zweifach zusammenhängend ist, Maximalgrad 3 hat oder außenplanar ist mit Schnittknoten von Grad höchstens 3. Ist jede Zusammenhangskomponente zweifach zusammenhängend, so hat der Algorithmus sogar optimale (lineare) Laufzeit.

Literaturverzeichnis

[AGR70] Akers, Sheldon B.; Geyer, James M.; Roberts, Donald L.: IC Mask Layout with a Single Conductor Layer. In: Proceedings of the 7th Annual Design Automation Conference (DAC'70). ACM Press, S. 7–16, 1970.

[AL14] Angelini, Patrizio; Lozzo, Giordano Da: Deepening the Relationship between SEFE and C-Planarity. Computing Research Repository, abs/1404.6175:1–8, 2014.

[BK98] Biedl, Therese; Kant, Goos: A Better Heuristic for Orthogonal Graph Drawings. Computational Geometry: Theory and Applications, 9(3):159–180, 1998.

[Bl14] Bläsius, Thomas; Krug, Marcus; Rutter, Ignaz; Wagner, Dorothea: Orthogonal Graph Drawing with Flexibility Constraints. Algorithmica, 68(4):859–885, 2014.

[Bl15] Bläsius, Thomas: New Approaches to Classic Graph-Embedding Problems – Orthogonal Drawings & Constrained Planarity. Dissertation, Faculty of Informatics, Karlsruhe Institute of Technology (KIT), 2015.

[Ei03] Eiglsperger, Markus: Automatic Layout of UML Class Diagrams: A Topology-Shape-Metrics Approach. Dissertation, Universität Tübingen, 2003.

[FK95] Fößmeier, Ulrich; Kaufmann, Michael: Drawing High Degree Graphs with Low Bend Numbers. In (Brandenburg, Franz J., Hrsg.): Proceedings of the 3rd International Symposium on Graph Drawing (GD'95). Jgg. 1027 in Lecture Notes in Computer Science. Springer Berlin/Heidelberg, S. 254–266, 1995.

[FK97] Fößmeier, Ulrich; Kaufmann, Michael: Algorithms and Area Bounds for Nonplanar Orthogonal Drawings. In (Di Battista, Giuseppe, Hrsg.): Proceedings of the 5th International Symposium on Graph Drawing (GD'97). Jgg. 1353 in Lecture Notes in Computer Science. Springer Berlin/Heidelberg, S. 134–145, 1997.

[GJ79] Garey, Michael R.; Johnson, David S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1979.

[GT01] Garg, Ashim; Tamassia, Roberto: On the Computational Complexity of Upward and Rectilinear Planarity Testing. SIAM Journal on Computing, 31(2):601–625, 2001.

[Le89] Lengauer, Thomas: Hierarchical Planarity Testing Algorithms. Journal of the ACM, 36(3):474–509, 1989.

[Li14] Lima, Manuel: The Book of Trees: Visualizing Branches of Knowledge. Princeton Architectural Press, 2014.

[PCJ96] Purchase, Helen C.; Cohen, Robert F.; James, Murray: Validating Graph Drawing Aesthetics. In (Brandenburg, Franz J., Hrsg.): Proceedings of the 3rd International Symposium on Graph Drawing (GD'95). Jgg. 1027 in Lecture Notes in Computer Science. Springer Berlin/Heidelberg, S. 435–446, 1996.

[Sc13] Schaefer, Marcus: Toward a Theory of Planarity: Hanani-Tutte and Planarity Variants. Journal of Graph Algorithms and Applications, 17(4):367–440, 2013.

[St80] Storer, James A.: The Node Cost Measure for Embedding Graphs on the Planar Grid (Extended Abstract). In: Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC'80). ACM Press, S. 201–210, 1980.

[Wa09] Wang, Yanju; Lin, Wei-Yu; Liu, Kan; Lin, Rachel J.; Selke, Matthias; Kolb, Hartmuth C.; Zhang, Nangang; Zhao, Xing-Zhong; Phelps, Michael E.; Shen, Clifton K. F.; Faull, Kym F.; Tseng, Hsian-Rong: An Integrated Microfluidic Device for Large-Scale in Situ Click Chemistry Screening. Lab on a Chip, 9(16):2281–2285, 2009.

Thomas Bläsius wurde 1987 in Konstanz geboren und ist in Trier aufgewachsten. Dort ging er auch zur Schule und erlangte im Frühjahr 2006 das Abitur. Bis zum Beginn des Informatikstudiums am Karlsruher Institut für Technologie im Herbst 2006 (damals noch unter dem Namen Universität Karlsruhe) war er als Praktikant im Bereich der Softwareentwicklung in der Mindox GmbH tätig. Das Informatikstudium schloss er im September 2011 mit einem Diplom ab. Ab Oktober 2011 arbeitete er als wissenschaftlicher Mitarbeiter am Lehrstuhl für Algorithmik von Prof. Dorothea Wagner des Karlsruher Institut für Technologie, wo er neben der Forschung an Algorithmen zur Graphvisualisierung, die zu der hier beschriebenen Dissertation führte, auch in der Lehre tätig war. Seine Promotion schloss er im Juli 2015 ab. Seit Oktober 2015 ist er Postdoc am Algorithm Engineering Lehrstuhl von Prof. Tobias Friedrich am Hasso Plattner Institut (Potsdam).