Visualisierung von Netzen: Algorithmen, Anwendungen und Komplexität

Martin Nöllenburg

Institut für Theoretische Informatik
Karlsruher Institut für Technologie (KIT)
noellenburg@kit.edu

Abstract: Netze sind physische oder virtuelle Strukturen, die aus Objekten und Verknüpfungen dieser Objekte bestehen. Für menschliche Betrachter werden Netze und ihre Eigenschaften meist erst durch geeignete Visualisierungen zugänglich. In dieser Arbeit geht es darum, praktische und theoretische Visualisierungsprobleme für unterschiedliche Arten von Netzen aus algorithmischer Sicht zu untersuchen. Die betrachteten Anwendungen stammen aus dem Bereich der Kartografie und der Biologie. Darüberhinaus werden auch theoretisch motivierte Fragestellungen untersucht. Ziel ist in der Regel Algorithmen zu finden, die beweisbar effizient oder – im Falle von NP-schweren Problemen – zumindest beweisbar gut sind. In einigen Fällen werden auch theoretisch langsame, aber exakte Algorithmen oder schnelle Heuristiken entwickelt und durch experimentelle Untersuchungen gerechtfertigt.

1 Einführung

Es gibt die unterschiedlichsten Arten von Netzen: Kommunikations- und Datennetze, Verkehrs- und Versorgungsnetze, soziale Netze, Netze in den Naturwissenschaften (z. B. Protein-Interaktionsnetze und Evolutionsbäume), Netze in den Wirtschaftswissenschaften (z. B. Handelsbeziehungen und Firmenbeteiligungen) und Netze in den Ingenieurswissenschaften (z. B. Schaltnetze in der Elektrotechnik oder Klassenhierarchien in der Softwaretechnik). Die Liste ließe sich mit vielen Beispielen aus Gesellschaft, Natur und Technik fortsetzen. Gemeinsam ist all diesen Netzen die mathematische Struktur eines *Graphen*. Ein solcher Graph G besteht aus einer Menge von Knoten V und einer Menge von Kanten E, wobei eine Kante jeweils zwei Knoten verbindet. Die oben genannten Beispiele lassen sich direkt in einen Graph übersetzen. Betrachtet man beispielsweise ein öffentliches Nahverkehrsnetz, so werden die Haltestellen zu Knoten; direkt hintereinander von einer Bus- oder Bahnlinie bediente Haltestellen werden durch eine Kante verbunden.

Nun sind Graphen in der Informatik, insbesondere in der Algorithmik, weit verbreitete Datenstrukturen und Gegenstand unzähliger Veröffentlichungen. Für Graphen lassen sich viele Eigenschaften effizient berechnen, z. B. kürzeste Wege zwischen zwei Knoten zum Routing in Daten- und Verkehrsnetzen oder Kapazitätsengpässe in Versorgungsnetzen. Hierzu benötigt ein Rechner lediglich eine geeignete interne Darstellung der Daten, z. B. in Form einer Adjazenzliste oder -matrix.

Ganz anders sieht die Situation aus, wenn man sich als menschlicher Betrachter für ein Netz oder einen Graph interessiert. Ein Mensch ist in langen Listen oder großen Matrizen schnell verloren. Einen Ausweg bietet die Visualisierung von Netzen, also die grafische Repräsentation der Knoten und Kanten des zugrundeliegenden Graphen. Ziel ist dabei in der Regel, eine möglichst *effektive* Darstellung eines Graphen in der Ebene zu finden, also eine Zeichnung, die für den Betrachter leicht zu lesen ist und gleichzeitig die zu vermittelnde Information objektiv wiedergibt, sei es zur Exploration von unbekannten Zusammenhängen oder zur Präsentation von bekanntem Wissen. Meist werden Knoten als Punkte und Kanten als Kurven zwischen den jeweiligen Endpunkten dargestellt. In einer guten Zeichnung können viele strukturelle Eigenschaften eines Graphen oft schon auf einen Blick erfasst werden. Gerade wenn es darum geht, Netze zu untersuchen und verborgene Eigenschaften zu entdecken, sind Visualisierungen ein unerlässliches Werkzeug. Auch zur Beschreibung von Daten, die als Graph vorliegen, ist eine Zeichnung oft das einfachste und effizienteste Mittel. Allerdings ist eine gute Visualisierung weit mehr als ein eilig produziertes buntes Bild [Tuf90]. Übersichtliche Zeichnungen von Hand zu erstellen ist sehr zeitintensiv und für große Graphen fast unmöglich. Der Ausweg liegt im automatischen Zeichnen von Graphen durch geeignete Algorithmen.

Mit grundlegenden theoretischen Fragen und Algorithmen zur Visualisierung von Graphen befasst sich das Forschungsgebiet des *Graphenzeichnens*. Dabei geht es meist darum, unter bestimmten vorgegebenen Einschränkungen (z. B. Knotenpositionen auf einem Gitter und geradlinig gezeichnete Kanten) ein Gütemaß für die Zeichnung zu optimieren (z. B. die benötigte Fläche oder die Anzahl von Kantenkreuzungen). Beim Graphenzeichnen handelt es sich inzwischen um ein etabliertes Teilgebiet der Informatik. So findet seit 18 Jahren jährlich das "International Symposium on Graph Drawing[1]" statt, und die drei Monografien [dBETT99, KW01, NR04] behandeln das Thema ausführlich.

In meiner Dissertation betrachte ich mehrere Probleme, die aus Anwendungen heraus motiviert sind. Das hat einen guten Grund. So gibt es zwar Algorithmen, die Zeichnungen für beliebige Graphen berechnen können, aber oft genügen diese Zeichnungen nicht den spezifischen Anforderungen der Anwendung. Je nach Einsatzzweck sind verschiedene Zeichenstile mit unterschiedlichen geometrischen Eigenschaften und dementsprechend verschiedene Algorithmen gefragt. Ein bestimmter Stil mag gut geeignet sein, um soziale Netze oder UML-Klassendiagramme darzustellen. Für die Visualisierung von Verkehrsnetzen oder Stoffwechselwegen ist der gleiche Stil dagegen vermutlich nicht geeignet. Darüberhinaus werfen Visualisierungsprobleme aus den Anwendungen oft auch interessante theoretische Fragen auf.

In den folgenden Abschnitten skizziere ich die konkreten Problemstellungen und Ergebnisse meiner Dissertation. Die ersten drei Themenfelder entstammen der Kartografie und Biologie: schematische Verkehrslinienpläne, dynamische Landkarten und phylogenetische Bäume. Die beiden weiteren Themen sind im Gegensatz dazu eher theoretisch motiviert und beschäftigen sich zum Einen mit der Darstellbarkeit von Graphen als Berührgraphen bestimmter geometrischer Objekte und zum Anderen mit der Konsistenz von Strahlen im Pixelgitter. Weitere Details zu den einzelnen Themen und eine Einführung in die verwendeten Konzepte finden sich direkt in der Dissertation [Nöl09].

[1] siehe `http://www.graphdrawing.org`

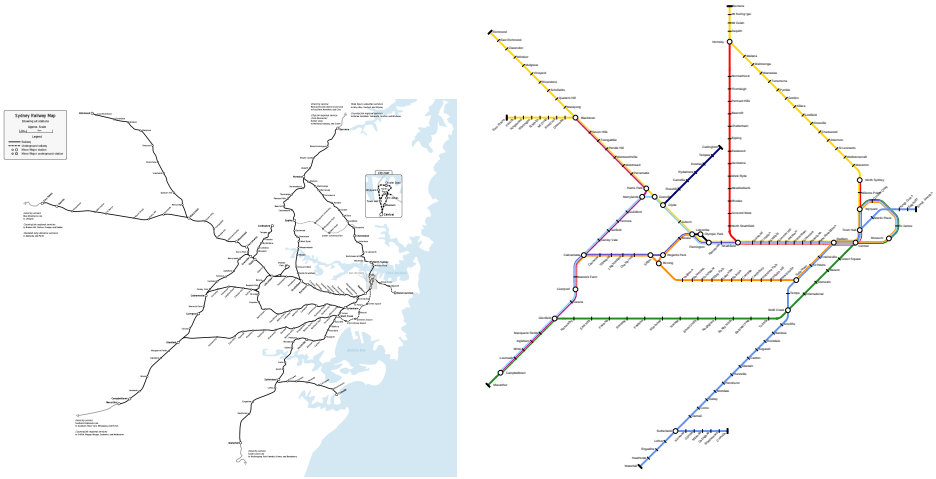
2 Schematische Linienpläne

Die erste Anwendung ist das Erstellen von schematischen Linienplänen, wie sie vorwiegend zur Darstellung von öffentlichen Verkehrsnetzen eingesetzt werden. Hier geht es darum, einen durch die geografische Lage der einzelnen Stationen und ihrer Verbindungen bereits eingebetteten Graph so zu schematisieren, dass typische visuelle Navigationsaufgaben ("Wie komme ich von A nach B? Wann muss ich aus- oder umsteigen?") möglichst effizient und fehlerfrei durchgeführt werden können. Ausgehend von einer Vielzahl existierender (und manuell entworfener!) Linienpläne konnte ich sieben (fast) universelle Entwurfsregeln identifizieren. Dementsprechend werden in meinem Ansatz die zulässigen Kantenrichtungen auf acht sogenannte *oktilineare* Richtungen (horizontal, vertikal, und 45°-diagonal) eingeschränkt, Knicke entlang des Verlaufs von Verkehrslinien minimiert und Kantenlängen maßstabsunabhängig vereinheitlicht – ohne allerdings das ursprüngliche Layout zu sehr zu verzerren. Es ist offensichtlich, dass die Bewertungsfunktion für die Qualität eines Linienplans mehrere sich widersprechende Kriterien enthält (z. B. Knickminimierung vs. Verzerrung) und daher sorgfältig ausbalanciert werden muss. Ein besonderes Augenmerk liegt auch auf dem Erstellen von beschrifteten Linienplänen, also Zeichnungen, die ausreichend Platz bieten, um alle Stationen überlappungsfrei mit ihrem Namen zu beschriften.

Sowohl das Zeichnen von unbeschrifteten Linienplänen als auch die reine Beschriftung von gegebenen Landkarten sind NP-vollständige Probleme. In meiner Arbeit betrachte ich das kombinierte Zeichen- und Beschriftungsproblem und modelliere es als gemischtganzzahliges lineares Programm. Hierbei handelt es sich um ein diskretes mathematisches Optimierungsverfahren, das zwar auch NP-vollständig ist, aber in der Praxis trotzdem in vielen Fällen erfolgreich eingesetzt wird. Einige der oben genannten Entwurfsregeln für Linienpläne werden als Nebenbedingungen in lineare Ungleichungen übersetzt, deren Erfüllung dafür sorgt, dass Kanten oktilinear sind, die Eingabetopologie des Graphen erhalten bleibt und ausreichend Platz für die Haltestellennamen ist. Diese Regeln sind *harte* Anforderungen an die Zeichnung. Die verbleibenden Regeln werden in die lineare Zielfunktion übersetzt, deren Optimierung dann dafür sorgt, dass Linienknicke, Verzerrung und Flächenbedarf gemeinsam minimiert werden. Letztere Kriterien werden *weich* genannt, da sie zwar optimiert, aber nicht erzwungen werden.

Das vorgestellte lineare Programm wurde implementiert und die Ergebnisse anhand dreier Fallstudien im Vergleich zu existierenden Ansätzen [HMdN06, SRMOW10] und offiziellen Plänen evaluiert. Abbildung 1 zeigt den Eingabegraphen und die Ausgabezeichnung für das Beispiel Sydney. In Bezug auf die Ergebnisqualität und die Einhaltung der Entwurfsregeln ist meine Methode bisherigen Ansätzen weit überlegen. Zu diesem Schluss kommt auch die Auswertung einer aktuellen Expertenbefragung [NW10].

Ein sekundäres Problem, das häufig bei der Visualisierung von Linienplänen auftritt, ist die Darstellung von parallel verlaufenden Verkehrslinien entlang gemeinsamer Kanten. Üblicherweise wird jede Bus- oder Bahnlinie stetig und in einer eindeutigen Farbe als Pfad entlang der benutzten Kanten des zugrundeliegenden Netzes gezeichnet. Dabei kann es – trotz der Planarität des Netzes – zu Kreuzungen kommen. Nur manche dieser Kreuzungen sind durch die Topologie des Netzes unvermeidlich. In meiner Dissertation betrachte ich



(a) Geographische Eingabe. (b) Beschriftete schematische Zeichnung.

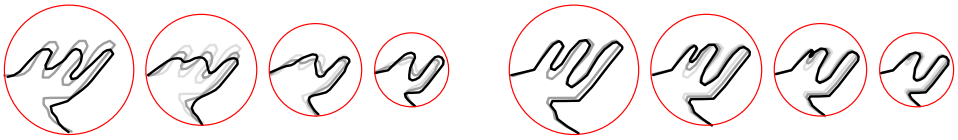
Abbildung 1: Linienplan des CityRail-Netzes in Sydney.

erstmals die Minimierung solcher Linienkreuzungen in einer gegebenen Einbettung des zugrundeliegenden Graphen.

Zunächst wird das Kreuzungsminimierungsproblem eingeschränkt auf eine einzelne Kante des Graphen optimal mittels dynamischer Programmierung gelöst. Der Algorithmus benötigt dazu quadratische Laufzeit. In einer weiteren interessanten Variante geht man davon aus, dass die Linien nicht völlig beliebig platziert werden können, sondern in ihren Anfangs- und Endknoten bereits mit einer relativen vorgegebenen Position starten. Ohne diese Einschränkung weiß man, dass das Problem NP-schwer ist [BKPS08]. In vielen Fällen in der Praxis hat man die relativen Positionen jedoch tatsächlich gegeben; alternativ lassen sie sich mit Hilfe eines ganzzahligen linearen Programms bestimmen [AGM08]. Geht man also davon aus, dass die Linienpositionen in den Endhaltestellen bereits gegeben sind, so benötigt der neue Algorithmus $O(\ell^2 n)$ Zeit, wobei ℓ die Anzahl der Verkehrslinien ist und n die Anzahl der Haltestellen. Der Algorithmus geht in zwei Schritten vor. Zunächst wird für alle Linienpaare, die auf einem Teilpfad gemeinsam verlaufen, bestimmt, in welcher Konstellation sie sich am Ende trennen. Anschließend werden die einzelnen Linien nacheinander in konsistenter Weise dem Layout hinzugefügt, so dass am Ende in jedem Knoten eine vollständige Ordnung aller vorhandenen Linien entsteht.

3 Dynamische Landkarten

Der zweite Anwendungsbereich entstammt wiederum der Kartografie, diesmal geht es jedoch um die Darstellung dynamischer, interaktiver Karten, in denen der Nutzer konti-



(a) Lineare Interpolation.

(b) Interpolation mit optimaler Zuordnung.

Abbildung 2: Ausschnitt der Animation eines Straßenverlaufs beim Zoomen zwischen zwei Maßstäben. Der Verlauf in vorangehenden Einzelbildern ist durch graue Kurven angedeutet.

nuierlich den Kartenausschnitt und -maßstab entsprechend seiner Anforderungen wählen kann. Im Gegensatz zur Visualisierung statischer Karten sind hier viele typische Fragestellungen noch offen. In meiner Dissertation beschäftige ich mich mit zwei Problemen: der Beschriftung von Punkten und der Generalisierung von Polygonzügen.

Unter dem Begriff der *Generalisierung* versteht man in der Kartografie die Anpassung der Darstellungskomplexität von Objekten in der Karte an deren Maßstab. Das Generalisieren von Polygonzügen (z. B. in Straßen- oder Flussnetzen) für einen festen Maßstab ist algorithmisch recht gut verstanden [MRS07]. In Bezug auf das kontinuierliche Zoomen in interaktiven Karten ergibt sich jedoch das Problem, dass es mit den bisherigen Methoden zu Diskontinuitäten kommen kann, z. B. wenn eine Gebirgsstraße beim Herauszoomen für die Darstellung ab einem gewissen Maßstab aufgrund des eingeschränkten Platzes plötzlich eine Serpentine weniger aufweisen soll. Um solche Diskontinuitäten zu vermeiden, stelle ich einen Algorithmus vor, der, ausgehend von zwei gegebenen Darstellungen sich entsprechender Polygonzüge in unterschiedlichen Maßstäben, zunächst die beiden Polygonzüge in charakteristische Abschnitte unterteilt. Danach wird mit dynamischer Programmierung eine Zuordnung der Abschnitte bestimmt, die bezüglich eines geeigneten Abstandsmaßes optimal ist. Dadurch wird erreicht, dass bei der Interpolation semantisch äquivalente Abschnitte während des Zoomens zwischen den Eingabedarstellungen mit möglichst geringen Formveränderungen ineinander übergehen. Um nicht nur einzelne Polygonzüge sondern ganze Straßennetze dynamisch darstellen zu können, genügt es jeden Straßenabschnitt zwischen zwei Kreuzungen einzeln zu interpolieren.

Anhand mehrerer Beispiele von Straßen, Flüssen und Landesgrenzen vergleiche ich die Interpolation entsprechend der berechneten optimalen Zuordnung der Abschnitte mit einer gewöhnlichen linearen Interpolation; dabei zeigt sich in den Experimenten, dass die Verfälschung der Polygonzüge bei der neu vorgestellten Interpolation weit geringer ausfällt als bei einer linearen Interpolation. Abbildung 2 zeigt, dass sich der Kurvenverlauf bei einer linearen Interpolation in Zwischenmaßstäben stark verändern kann, während die Interpolation entsprechend der optimalen Zuordnung der Abschnitte die grobe Form der Kurven beibehält und damit die *mentale Karte* des Betrachters weitestgehend respektiert.

Ein zweites Problem im Zusammenhang mit dynamischen Landkarten ist das konsistente Beschriften von Kartenobjekten bei Nutzerinteraktion. Betrachtet man das Beschriften in statischen Karten, so müssen die Ortsnamen eindeutig und überlappungsfrei angeordnet werden. In dynamischen Karten gelten zusätzliche Anforderungen für die Animation beim

Zoomen und Verschieben der Karte. Beim Herauszoomen aus einem Kartenausschnitt haben die vorhandenen Ortsnamen bei konstanter Schriftgröße auf dem Bildschirm einen wachsenden Platzbedarf in der Karte, so dass es zu Verdrängungseffekten kommt und bestimmte Namen ausgeblendet werden müssen. Eine dynamische Beschriftung heißt *konsistent*, wenn jeder Ortsname in höchstens einem Intervall von Maßstäben sichtbar ist. Das bedeutet, dass es während eines monotonen Zoomvorgangs nicht zu Flackereffekten kommt, die durch das wiederholte Ein- und Ausblenden des Namens entstehen. Weiterhin dürfen die Namen in einer konsistenten Beschriftung nicht sprunghaft ihre Position in der Karte ändern.

In Anlehnung an statische Varianten des Beschriftungsproblems betrachte ich die Maximierung der Anzahl der sichtbaren Namen, allerdings integriert über alle möglichen Maßstäbe. Dieses Optimierungsproblem wird dreidimensional modelliert: Die ersten beiden Dimensionen entsprechen der eigentlichen Landkarte und die dritte Dimension entspricht dem Maßstab der Karte. So bilden die Rechteckboxen, die die Namen enthalten, über alle Maßstäbe betrachtet eine Menge von auf der Spitze stehenden Pyramiden, siehe Abbildung 3. Denn damit die Schriftgröße in der Karte konstant bleibt, nimmt die Größe der Rechtecke mit fallendem Maßstab relativ zum Kartenausschnitt zu. Ziel ist es nun, für jede solche Pyramide genau ein (evtl. leeres) *aktives* Maßstabsintervall zu bestimmen, innerhalb dessen der entsprechende Name angezeigt wird. Damit es nicht zu Überschneidungen der Namen kommt, dürfen sich allerdings keine zwei der dadurch induzierten Pyramidenstümpfe schneiden. Das dynamische Beschriftungsproblem erweist sich schon für sehr einfache Varianten als NP-vollständig. Diesmal liegt der Ausweg darin, effiziente Approximationsalgorithmen zu entwerfen, die zwar keine optimalen Lösungen, aber zumindest Lösungen mit fester Gütegarantie berechnen. Für verschiedene Varianten, abhängig von der Pyramidenform, ergeben sich unterschiedliche Approximationsfaktoren.

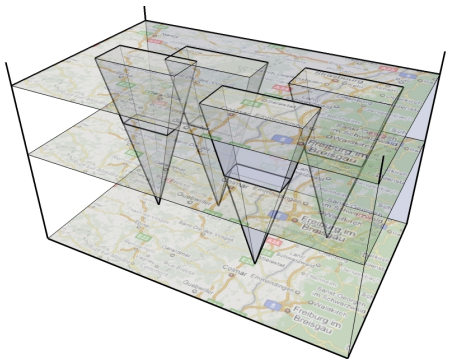


Abbildung 3: Dreidimensionale Modellierung des dynamischen Beschriftungsproblems.

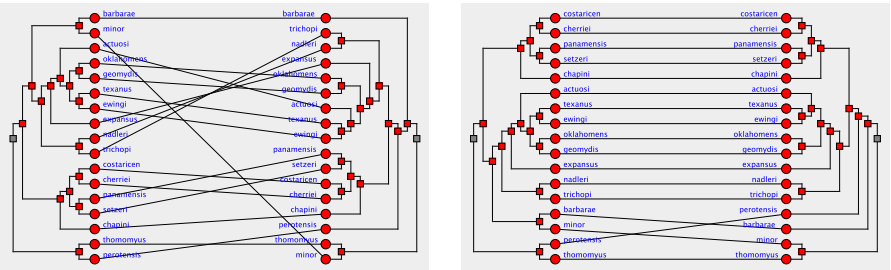
4 Phylogenetische Bäume

Phylogenetische Bäume sind Binärbäume, die in der Biologie die evolutionären Verwandtschaftsbeziehungen zwischen verschiedenen Spezies darstellen. Sie stehen jedoch meist nur für Hypothesen der realen Abstammungsverhältnisse. So lassen sich für die gleiche Menge von Spezies je nach Parameterwahl und Rekonstruktionsmethode unterschiedliche phylogenetische Bäume erstellen, die anschließend auf ihre Gemeinsamkeiten untersucht

werden sollen. Ein visueller Vergleich zweier Bäume ist mit einem sogenannten *Tanglegram* möglich, einer Gegenüberstellung der beiden planar gezeichneten Binärbäume, so dass die Blätter (die den Spezies entsprechen) auf zwei parallelen Geraden angeordnet sind und je zwei sich entsprechende Blätter miteinander durch eine *Zwischenbaumkante* verbunden werden. Zwei Beispiele von Tanglegrams sind in Abbildung 4 zu sehen. Im Allgemeinen kreuzen sich diese Interbaumkanten. Je mehr Kreuzungen das Tanglegram hat, umso schlechter ist es lesbar. Das algorithmische Problem hinter der Optimierung der Lesbarkeit ist also, zwei planare Baumlayouts zu finden, die die Kreuzungsanzahl der Interbaumkanten minimieren.

Das Problem ist für allgemeine Binärbäume als NP-vollständig bekannt [FKP05]. Durch einen neuen Beweis konnte ich die NP-Vollständigkeit auch auf den Spezialfall von vollständigen Binärbäumen ausdehnen. Im Gegensatz zum allgemeinen Tanglegram-Problem, das, wie ich zeigen konnte, unter einer weit verbreiteten Annahme (*Unique Games Conjecture*) keine Approximationsalgorithmen mit konstantem Approximationsfaktor zulässt, bieten vollständige Binärbäume mehr Angriffspunkte. So beschreibe ich in der Dissertation einen Algorithmus, der höchstens um einen Faktor 2 vom Optimum abweicht und dazu $O(n^3)$ Zeit für einen Baum mit n Blättern benötigt. Darüberhinaus lässt sich das Problem mit einem parametrisierten Algorithmus exakt in Laufzeit $O(4^k n^2)$ lösen, wobei k die Anzahl der Kreuzungen in einer optimalen Lösung ist.

Auch für den praktisch relevanteren Fall von allgemeinen Binärbäumen stelle ich einerseits ein sehr einfaches ganzzahliges lineares Programm und andererseits einen exakten branch-and-bound Algorithmus vor. Der branch-and-bound Algorithmus durchläuft die inneren Knoten der beiden Bäume in einer geschickten Reihenfolge. In jedem Knoten wird entschieden, wie die beiden Teilbäume der Kindknoten lokal optimal angeordnet werden. So gelangt man sehr schnell zu erstaunlich guten Lösungen. Im schlechtesten Fall benötigt das Verfahren zwar exponentielle Laufzeit, als Heuristik lässt sich aber auch die bereits in $O(n^2)$ Zeit gefundene erste Lösung verwenden. Die Heuristik hat zudem die Eigenschaft, garantiert die optimale Lösung zu finden, falls diese kreuzungsfrei ist. In einer umfangreichen experimentellen Evaluation der vorgestellten Algorithmen und anderer



(a) Zufälliges Tanglegram.

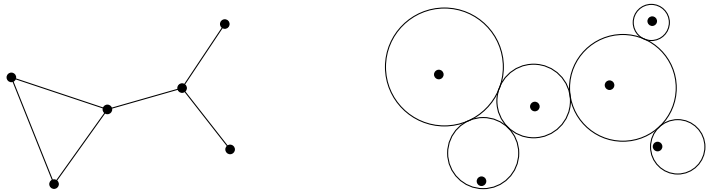
(b) Tanglegram mit minimaler Kreuzungszahl.

Abbildung 4: Visueller Vergleich zweier phylogenetischer Bäume für Parasitenstämme von Taschenratten.

existierender Verfahren mit generierten und realen Eingabeinstanzen zeigt sich, dass die genannte Heuristik den anderen Verfahren in Bezug auf die Lösungsqualität weit überlegen ist. Für die meisten Instanzen findet sie sogar optimale Lösungen und ist gleichzeitig noch schnell genug für den Einsatz in der Praxis.

5 Überdeckungskontaktgraphen

Ein klassisches Resultat in der Graphentheorie besagt, dass sich jeder planare Graph als Kontaktgraph von Kreisen in der Ebene darstellen lässt [Koe36]. Hierbei repräsentieren Kreise die Knoten des Graphen und sich berührende Kreise jeweils eine Kante. Andererseits geht es in vielen geometrischen Optimierungsproblemen darum, beispielsweise Punkte durch andere geometrische Objekte (z. B. Kreise oder Polygone) in einer gewissen Weise optimal zu überdecken. Überdeckungskontaktgraphen kombinieren die beiden Fragestellungen: Gegeben ist eine Menge S von zu überdeckenden Objekten (z. B. Punkte) und eine Klasse von Überdeckungselementen (z. B. Kreise). Gesucht ist eine Überdeckung von S, so dass jedes Objekt in S von genau einem Element überdeckt wird und die Überdeckungselemente sich höchstens berühren. Dadurch wird der Überdeckungskontaktgraph (ÜKG) mit der Knotenmenge S definiert, in dem eine Kante zwischen zwei Knoten genau dann besteht, wenn sich die zugehörigen Überdeckungselemente berühren. Abbildung 5 zeigt ein Beispiel.



(a) Eingabegraph G. (b) Realisierung von G als ÜKG.

Abbildung 5: Realisierung eines Graphen als Überdeckungskontaktgraph.

Die beiden in der Arbeit betrachteten Fragen sind, ob zu gegebenem S ein zusammenhängender ÜKG existiert und ob sich ein gegebener Graph als ÜKG auf S realisieren lässt. Ich zeige, dass sich Punkte in der Ebene immer so mit Kreisen bzw. Dreiecken überdecken lassen, dass der entstehende ÜKG ein- oder zweifach zusammenhängend ist. Die Frage nach der Realisierbarkeit eines gegebenen Graphen ist dagegen NP-schwer. Lediglich für Bäume und n Punkte auf einer Geraden, die durch Kreise bzw. Dreiecke überdeckt werden sollen, lässt sich die Realisierbarkeit unter bestimmten Voraussetzungen in $O(n \log n)$ Zeit entscheiden. Betrachtet man statt Punkten Kreise als zu überdeckende Objekte, ist selbst die Frage nach einem zusammenhängenden ÜKG NP-schwer.

6 Konsistente Strahlen im Raster

Der letzte Teil meiner Arbeit beschäftigt sich mit einem sehr grundlegenden Visualisierungsproblem von gerasterten oder *digitalen* Strecken in der Gittergeometrie. Übliche Rastermethoden, die auf dem Schnitt von euklidischen Strecken mit einem Pixelgitter beruhen, weisen Konsistenzprobleme auf, wenn mehrere Strecken gleichzeitig betrachtet werden. Beispielsweise ist der Schnitt zweier solcher Strecken nicht immer eine zusammenhängende Pixelmenge, wie Abbildung 6 zeigt. Um die Konsistenz digitaler Strecken analog zu Strecken in der euklidischen Geometrie zu gewährleisten, stelle ich vier Konsistenzaxiome auf, die erfüllt werden müssen. Andererseits soll aber die Ähnlichkeit der digitalen Strecken zu ihren euklidischen Gegenstücken möglichst hoch sein. Auf der einen Seite hat man die herkömmlichen Rasterstrecken, die eine hohe Ähnlichkeit aufweisen, aber nicht konsistent sind. Auf der anderen Seite lassen sich sehr einfache konsistente Strecken definieren, die sich allerdings stark von den zugehörigen euklidischen Strecken unterscheiden. Die Frage ist also, konsistente digitale Strecken zu finden, die trotzdem noch eine hohe Ähnlichkeit zu den entsprechenden euklidischen Strecken haben.

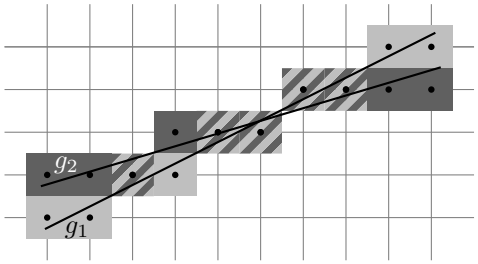


Abbildung 6: Beispiel zweier digitaler Strecken, deren Schnitt (schraffiert) nicht zusammenhängt.

Für den Spezialfall von Strahlen in einem Gitter der Größe $n \times n$, die von einem gemeinsamen Ursprung ausgehen, beweise ich eine untere Schranke von $\Omega(\log n)$ für den Hausdorff-Abstand als Ähnlichkeitsmaß zwischen euklidischem Strahlensegment und zugehörigem konsistentem digitalen Strahlensegment. Gleichzeitig gebe ich ein System von konsistenten Strahlen mit oberer Schranke $O(\log n)$ für den Hausdorff-Abstand an; das System ist in diesem Sinne also asymptotisch optimal. Zudem lässt es sich auf den d-dimensionalen Fall verallgemeinern.

Digitale Strahlen können genutzt werden, um sternförmige Gebiete in einem Gitter zu definieren. Damit lässt sich z. B. das sogenannte *mountain approximation problem* aus der Bildverarbeitung lösen, das in einem Grauwertbild nach sternförmigen Gebieten mit maximalem Gewicht sucht.

Literatur

[AGM08] Matthew Asquith, Joachim Gudmundsson und Damian Merrick. An ILP for the Metro-Line Crossing Problem. In J. Harland und P. Manyem, Hrsg., *Proc. 14th Computing: The Australasian Theory Symp. (CATS'08)*, Jgg. 77 of *CRPIT*, Seiten 49–56. Australian Comput. Soc., 2008.

[BKPS08] Michael A. Bekos, Michael Kaufmann, Katerina Potika und Antonios Symvonis. Line Crossing Minimization on Metro Maps. In S.-H. Hong, T. Nishizeki und W. Quan,

Hrsg., *Proc. 15th Internat. Symp. Graph Drawing (GD'07)*, Jgg. 4875 of *Lecture Notes Comput. Sci.*, Seiten 231–242. Springer-Verlag, 2008.

[dBETT99] Giuseppe di Battista, Peter Eades, Roberto Tamassia und Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[FKP05] Henning Fernau, Michael Kaufmann und Mathias Poths. Comparing Trees Via Crossing Minimization. In R. Ramanujam und S. Sen, Hrsg., *Proc. 25th Internat. Conf. Found. Softw. Techn. Theoret. Comput. Sci. (FSTTCS'05)*, Jgg. 3821 of *Lecture Notes Comput. Sci.*, Seiten 457–469. Springer-Verlag, 2005.

[HMdN06] Seok-Hee Hong, Damian Merrick und Hugo A. D. do Nascimento. Automatic Visualization of Metro Maps. *J. Visual Languages and Computing*, 17(3):203–224, 2006.

[Koe36] Paul Koebe. Kontaktprobleme der konformen Abbildung. *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Klasse*, 88:141–164, 1936.

[KW01] Michael Kaufmann und Dorothea Wagner, Hrsg. *Drawing Graphs: Methods and Models*, Jgg. 2025 of *Lecture Notes Comput. Sci.* Springer-Verlag, 2001.

[MRS07] William A. Mackaness, Anne Ruas und L. Tiina Sarjakoski, Hrsg. *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Elsevier, 2007.

[Nöl09] Martin Nöllenburg. *Network Visualization: Algorithms, Applications, and Complexity*. Dissertation, Fakultät für Informatik, Universität Karlsruhe (TH), Februar 2009.

[NR04] Takao Nishizeki und Md. Saidur Rahman. *Planar Graph Drawing*, Jgg. 12 of *Lecture Notes Series on Computing*. World Scientific, 2004.

[NW10] Martin Nöllenburg und Alexander Wolff. Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *IEEE Trans. Visualization and Computer Graphics*, 2010. Accepted for publication.

[SRMOW10] Jonathan Stott, Peter Rodgers, Juan Carlos Martinez-Ovando und Stephen G. Walker. Automatic Metro Map Layout Using Multicriteria Optimization. *IEEE Trans. Visualization and Computer Graphics*, 2010. Preprint available online.

[Tuf90] Edward R. Tufte. *Envisioning Information*. Graphics Press, 1990.

Martin Nöllenburg wurde am 28. Juni 1979 in Heilbronn geboren. Er studierte von 1999 bis 2005 Informatik an der Universität Karlsruhe (TH). Gefördert durch ein Stipendium des DAAD führte er 2002/03 einen Studienaufenthalt an der McGill Universität in Montreal, Kanada durch. Sein Studium beendete er mit einer Diplomarbeit über das automatische Zeichnen von U-Bahnlinienplänen. Die Arbeit wurde mit dem NRW Undergraduate Science Award 2005 und dem Absolventenpreis der Fakultät für Informatik ausgezeichnet. Im Anschluss promovierte er bei Prof. Alexander Wolff über die Visualisierung von Netzen und legte seine Promotionsprüfung am 13. Februar 2009 mit Auszeichnung ab. Inzwischen ist er als Leiter einer Young Investigator Group am Karlsruher Institut für Technologie (KIT) tätig und befindet sich aktuell, gefördert durch ein Stipendium der DFG, zu einem achtmonatigen Forschungsaufenthalt an der University of California, Irvine.