Geometrische Netzwerke und ihre Visualisierung

Schriftliche Habilitationsleistung

Alexander Wolff

Institut für Theoretische Informatik
Fakultät für Informatik
Universität Karlsruhe

Eingereicht am 14. Juni 2005,
überarbeitete endgültige Fassung vom 19. Februar 2007

Inhaltsverzeichnis

Im Anschluss folgen die vorgelegten Arbeiten.

Verzeichnis der vorgelegten Arbeiten

[1] **Facility location and the geometric minimum-diameter spanning tree**. Computational Geometry: Theory and Applications, 27(1):87–106, 2004. In Zusammenarbeit mit Joachim Gudmundsson, Herman Haverkort, Sang-Min Park und Chan-Su Shin.

[2] **Optimal spanners for axis-aligned rectangles**. Computational Geometry: Theory and Applications, 30(1):59–77, 2005. In Zusammenarbeit mit Tetsuo Asano, Mark de Berg, Otfried Cheong, Hazel Everett, Herman Haverkort und Naoki Katoh.

[3] **The minimum Manhattan network problem: Approximations and exact solutions**. Computational Geometry: Theory and Applications, 35(3):188–208, 2006. In Zusammenarbeit mit Marc Benkert, Florian Widmann und Takeshi Shirabe.

[4] **Constructing interference-minimal networks**. In: J. Wiedermann, J. Stuller, G. Tel, J. Pokorný, and M. Bieliková (Herausgeber): *Proc. 32nd Int. Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM'06)*, Band 3831 der Reihe *Lecture Notes in Computer Science*, Seiten 166–176. Springer-Verlag, 2006. In Zusammenarbeit mit Marc Benkert, Joachim Gudmundsson und Herman Haverkort.

[5] **Configurations with few crossings in topological graphs**. Computational Geometry: Theory and Applications, 37(2). Erscheint Juli 2007. In Zusammenarbeit mit Christian Knauer, Étienne Schramm und Andreas Spillner.

[6] **Constructing the city Voronoi diagram faster**. International Journal of Computational Geometry and Applications, 2007. Erscheint in Bälde. In Zusammenarbeit mit Robert Görke und Chan-Su Shin.

[7] **A mixed-integer program for drawing high-quality metro maps**. In: P. Healy and N. S. Nikolov (Herausgeber): *Proc. 13th Int. Symposium on Graph Drawing (GD'05)*, Band 3843 der Reihe *Lecture Notes in Computer Science*, Seiten 321–333. Springer-Verlag, 2006. In Zusammenarbeit mit Martin Nöllenburg.

[8] **Boundary labeling: Models and efficient algorithms for rectangular maps**. Computational Geometry: Theory and Applications, 36(3):215–236, 2007. In Zusammenarbeit mit Michael A. Bekos, Michael Kaufmann und Antonios Symvonis.

[9] **Labeling points with weights**. Algorithmica, 38(2):341–362, 2003. In Zusammenarbeit mit Sheung-Hung Poon, Chan-Su Shin, Tycho Strijk und Takeaki Uno.

[10] **New algorithms for two-label point labeling**. In: M. Paterson (Herausgeber): *Proc. 8th Annu. Europ. Symp. on Algorithms (ESA'00)*, Band 1879 der Reihe *Lecture Notes in Computer Science*, Seiten 368–379. Springer-Verlag, 2000. In Zusammenarbeit mit Zhongping Qin, Yinfeng Xu und Binhai Zhu.

[11] **A simple factor-2/3 approximation algorithm for two-circle point labeling**. International Journal of Computational Geometry and Applications, 12(4):269–281, 2002. In Zusammenarbeit mit Michael Thon und Yinfeng Xu.

Statt der Arbeit [BGHW06] liegt eine längere Version bei, die zur Veröffentlichung bei einem Journal eingereicht wurde.

Kapitel 1

Einleitung

Geometrische Netzwerke sind Graphen, deren Knoten Punkten in einem metrischen Raum entsprechen. Wenn zusätzlich die Kanten solcher Netzwerke geradlinig sind, d.h. wenn ihre Länge dem Abstand ihrer Endpunkte entspricht, reden wir von geometrischen Graphen. Geometrische Netzwerke sind das Rückgrat bei der Modellierung von Verkehrs-, Waren- und Informationsströmen – ob in der Eisenbahnnetzplanung, dem VLSI-Layout oder der Analyse des Internets. Ich habe unter anderem mit Methoden der algorithmischen Geometrie und Graphentheorie sowie der linearen Optimierung zwei Teilgebiete bearbeitet.

Das erste Gebiet ist die *Konstruktion und Analyse geometrischer Netzwerke*, die Objekte mit gegebener Lage im Raum verbinden und dabei gewisse positive Eigenschaften haben – wie etwa geringen Durchmesser. Von besonderem Interesse sind *geometrische Spanner*, also Netzwerke, in denen jeder Weg im Netzwerk im Vergleich zur „Luftlinie“ nur um einen konstanten Faktor, den Streckfaktor, länger ist. Für die praktische Anwendung ist wichtig, dass ein Spanner möglichst dünn besetzt ist. Solche Spanner finden Verwendung in verteilten Systemen, in Kommunikationsnetzwerken, in der Robotik, in der Mustererkennung, bei der Kompression von Daten sowie in der Biologie und bieten noch viele interessante offene Fragen.

Das zweite Gebiet ist die *Visualisierung geometrischer Netzwerke*. Da bei solchen Netzwerken die Geometrie schon vorgegeben ist, scheint eines der Hauptprobleme beim Zeichnen von Graphen zu entfallen, nämlich die Knoten des Graphen so in die Ebene einzubetten, dass die resultierende Zeichnung möglichst gut lesbar ist. Das Beispiel U-Bahn-Linienplan zeigt jedoch stellvertretend für viele Arten von technischen Zeichnungen die Schwierigkeit: wie nämlich die zugrunde liegende Geometrie zugunsten einer klareren Darstellung des Netzwerks verzerrt, aber nicht aufgehoben wird.

Im vorliegenden Übersichtsartikel gehe ich zuerst in Kapitel 2 auf die Konstruktion und Analyse geometrischer Netzwerke ein. In Kapitel 3 folgt die Visualisierung geometrischer Netzwerke.

Ich möchte mich an dieser Stelle bei allen bedanken, mit denen ich in den letzten Jahren zusammen gearbeitet habe: an erster Stelle bei Dorothea Wagner, die mir immer mit Rat und Tat zur Seite gestanden hat und immer zur richtigen Zeit gedrängt hat, damit es wieder ein Stückchen vorwärts geht... Herzlichen Dank meinen Koautoren, meinen Doktoranden, Diplomanden, StudienarbeiterInnen, allen Kollegen vom Korridor, Bernd und Lilian: die Arbeit mit Euch hat Spaß gemacht – (fast) jeden Tag! All das wäre aber nur halb so viel wert, wenn Sveta nicht wäre. Und Sofia!

Kapitel 2

Konstruktion und Analyse geometrischer Netzwerke

Kriterien für die Konstruktion von Netzwerken sind viele vorgeschlagen worden: Größe (also Kantenanzahl), Gewicht (also Gesamtlänge der Kanten), Maximalgrad, Durchmesser, Zusammenhang, Planarität oder maximale Kantenlast (für eine gegebene Menge von Knotenpaaren). In einem geometrischen Kontext ist darüber hinaus der Streckfaktor von besonderem Interesse, also der maximale Faktor, um den der kürzeste Weg zwischen zwei Knoten im Netzwerk länger ist als die „Luftlinie“.

Geometrische Spanner sind geometrische Netzwerke mit konstantem Streckfaktor. Sie werden in verteilten Systemen und Kommunikationsnetzwerken [Awe85, PU89], in der Robotik (Bewegungsplanung), in der Mustererkennung und Datenkompression sowie in der Biologie [BD86, LAP03] angewendet. Dünn besetzte Spanner, also solche linearer Größe, sind ein sehr kompaktes Modell eines metrischen Raums und daher ideale Strukturen, wenn es um die Speicherung großer Datenmengen geht. Viele Algorithmen, deren Zielfunktion Längen oder Abstände sind, liefern t-Approximationen, wenn sie statt auf den Originalgraphen auf einen t-Spanner (d.h. einen Spanner mit Streckfaktor t) angewendet werden. Auch das motiviert die Beschäftigung mit Spannern.

Geometrische Spanner (auch *euklidische Spanner* genannt) wurden für punktförmige Objekte schon gründlich erforscht. Um Spanner linearer Größe zu konstruieren, wurden sehr verschiedene Methoden entwickelt. Die erste Methode geht auf Yao [Yao82] zurück. Er schlug vor, für einen gegebenen Winkel θ den Raum um jeden Eingabepunkt in Keile dieser Größe aufzuteilen und dann jeden Punkt mit den Punkten durch Kanten zu verbinden, die ihm in irgendeinem Keil am nächsten sind. Später wurde gezeigt, dass diese *Yao- (oder θ-) Graphen* tatsächlich konstanten Streckfaktor (in Abhängigkeit von θ) haben [RS91]. Yao-Graphen lassen sich zwar schnell, d.h. in $O(n \log n)$ Zeit berechnen, garantieren aber weder kleinen Grad noch geringes Gewicht. Die zweite Methode benützt hierarchische Aufteilungen des Raums. Unter diesen hat besonders die *Well-Separated Pair-Decomposition* von Callahan und Kosaraju [CK95] große Auswirkungen auf das Gebiet gehabt [ADM$^+$95, GHP$^+$02, GLNS02]. Die dritte Methode ist ein Greedy-Algorithmus, der an Kruskals Algorithmus für die Berechnung minimal spannender Bäume erinnert: man fügt in den anfangs leeren Graphen nacheinander Kanten mit aufsteigendem Gewicht ein, falls es noch keinen genügend kurzen Pfad gibt, der ihre Endpunkte miteinander verbindet. Das Gewicht des so konstruierten Spanners ist nur ein konstantes Vielfaches des Gewichts eines minimal spannenden Baums. Das schnellste

Implementierung des Greedy-Algorithmus' benötigt ebenfalls $O(n \log n)$ Zeit [GLN02].

Bald wurde untersucht, wie man geringe Größe mit anderen vorteilhaften Merkmalen kombinieren kann. Der *Link-Durchmesser* eines t-Spanners ist definiert als die kleinste Zahl D, so dass es für jedes Paar von Knoten einen Weg aus höchstens D Kanten gibt, dessen Streckfaktor t nicht übersteigt. In [AMS99] wird ein effizienter randomisierter Algorithmus angegeben, der es erlaubt, einen t-Spanner mit logarithmisch beschränktem Link-Durchmesser zu konstruieren und dynamisch aufrechtzuerhalten, wenn Punkte eingefügt und gelöscht werden (was jeweils nur polylogarithmische Zeit benötigt).

In [ADM+95] werden Algorithmen für die Konstruktion dünn besetzter geometrischer Spanner angegeben, die von den Kriterien Maximalgrad, Gewicht oder Link-Durchmesser jeweils eines, simultan zwei dieser Kriterien oder gar alle optimieren. Die Algorithmen für Maximalgrad und Gewicht sind zeitoptimal, die für die simultane Optimierung zweier Kriterien sind optimal, was den Kompromiss in der jeweiligen Kombination von Zielfunktionen betrifft. Ein Fehler bei der Kombination von Grad und Gewicht wird in [GLN02] durch einen neuen Algorithmus behoben. In [CDS01] werden entsprechende untere Zeitschranken etabliert, auch für die Berechnung von Spannern, die Hindernisse vermeiden, was in [ACC+96] untersucht wurde.

Außer den genannten Arbeiten widmen sich Übersichtsartikel im *Handbook of Computational Geometry* von Eppstein [Epp00], Mitchell [Mit00] und Smid [Smi00] dem Thema der geometrischen Netzwerke. In der Datenbank `geom.bib`[1], die leider seit Ende 2002 nicht mehr auf dem Laufenden gehalten wird, haben rund 50 Artikel das Stichwort „spanner“ im Titel, 20 davon außerdem entweder „Euclidean“, „geometry“ oder „geometric“. Es liegen mehrere Dissertationen zum Thema vor; eine Monographie [NS07] ist vor kurzem erschienen.

Im Rest dieses Kapitels gehe ich auf meine Arbeiten zum Thema der Konstruktion und Analyse geometrischer Netzwerke ein. In Abschnitt 2.1 befasse ich mich mit schnellen Approximationsalgorithmen für die Konstruktion von Spannbäumen minimalen Durchmessers. Um eine neue Art von Spannern, nämlich um Spanner für achsenparallele Rechtecke, geht es in Abschnitt 2.2. Ich zeige, dass es NP-schwer ist, für diese Objekte bei gegebener Netzwerk-Topologie Spanner mit minimalem Streckfaktor zu konstruieren, gebe aber effiziente Konstruktionsalgorithmen für den Fall an, dass der zugrunde liegende Graph ein Baum oder – besser noch – ein Pfad ist. Ebenfalls um die Konstruktion von Spannern geht es in Abschnitt 2.3, nämlich um gewichtsminimale Spanner mit Streckfaktor 1 bezüglich der Manhattan-Metrik, so genannte minimale Manhattan-Netzwerke. Während immer noch offen ist, wie schwer die Konstruktion solcher Spanner ist, gebe ich eine schnelle 3-Approximation an. Mit Kommunikationsnetzwerken befasse ich mich in Abschnitt 2.4. Ein wichtiges Ziel bei der Konstruktion von Netzwerken für die drahtlose Kommunikation ist die Minimierung der Interferenz, die misst, wie viele andere Knoten potentiell gestört werden, wenn ein bestimmter Knoten sendet. Ich gebe Algorithmen an, die für ein etabliertes Interferenz-Maß Netzwerke konstruieren, die unter allen Netzwerken einer gegebenen Eigenschaften (etwa für ein gegebenes t ein t-Spanner zu sein) minimale Interferenz besitzen.

In den beiden letzten Abschnitten dieses Kapitels beschäftige ich mich mit der Analyse von Netzwerken. In Abschnitt 2.5 geht es um Spannbäume und andere Konfigurationen mit wenigen Kreuzungen in geometrischen Graphen. Es ist bekannt, dass es schwer ist, in einem geometrischen Graphen einen kreuzungsminimalen Spannbaum zu finden. Ich zeige, dass es auch schwer ist, gute Approximationsalgorithmen für dieses Problem zu finden. Stattdessen gebe ich Fest-Parameter-Algorithmen an, die das Problem

[1] siehe `ftp://ftp.cs.usask.ca/pub/geometry/geombib.tar.gz`

effizient lösen, wenn die Anzahl der Kreuzungen in dem gegebenen Graphen konstant ist. Außerdem gebe ich eine gemischt-ganzzahlige Formulierung und eine einfache, aber effektive Heuristik für das Problem an. In Abschnitt 2.6 geht es um das so genannte City-Voronoi-Diagramm, das einem dabei hilft, in einer orthogonalen Umgebung mit Transportfließbändern und gleichartigen Basisstationen (etwa Postämtern) effizient den schnellsten Weg zur nächsten Basisstation zu finden. Ich gebe einen neuen Algorithmus für die Konstruktion des City-Voronoi-Diagramms an.

2.1 Spannbäume minimalen Durchmessers

Wir benötigen die folgenden Definitionen. Für ein geometrisches Netzwerk $G(V, E)$ mit $V \subset \mathbb{R}^d$ und Knoten s und t sei der *Netzwerkabstand* $\delta_G(s, t)$ von s und t die Länge eines kürzesten Weges von s nach t in N. Dabei sei die Länge einer Kante gerade der euklidische Abstand ihrer Endpunkte. Der *Durchmesser* von G ist dann $\max_{s,t \in V} \delta_G(s, t)$.

Wenn man für eine Menge von Punkten ein kreisfreies Netzwerk sucht, das für weit entfernte Punkte den Abstand im Netzwerk möglichst gering hält, so sind *Spannbäume minimalen Durchmessers* (SBMD) interessant. Eine mögliche Anwendungen sind Kommunikationsnetzwerke, in denen die maximale Zeitdauer beim Zustellen einer Nachricht minimiert werden soll, vorausgesetzt, dass die Zustellungszeit entlang einer Kante durch den euklidischen Abstand modelliert werden kann. Ho et al. [HLCW91] haben gezeigt, dass immer ein *monopolarer* oder ein *dipolarer* SBMD existiert, also ein SBMD, der einen bzw. zwei Knoten mit Grad größer 1 hat. Außerdem haben Ho et al. gezeigt, wie man im monopolaren Fall einen SBMD in $O(n \log n)$ und im dipolaren Fall in $O(n^3)$ Zeit berechnet. Die Lösung für den dipolaren Fall wurde sehr viel später von Chan [Cha03] durch die Verwendung von so genannten *halbdynamischen* Datenstrukturen verbessert. Sein Algorithmus benötigt $\tilde{O}(n^{3-c_d})$ Zeit, wobei c_d eine Konstante ist, die mit wachsender Dimension d des Raum gegen 0 konvergiert. Im planaren Fall, der uns hier hauptsächlich interessiert, gilt $c_2 = 1/6$. Die $\tilde{O}$-Notation versteckt Faktoren, die für ein beliebiges $\varepsilon > 0$ in $o(n^\varepsilon)$ liegen.

Mit Joachim Gudmundsson, Herman Haverkort, Sang-Min Park und Chan-Su Shin [GHP+02, GHP+04] habe ich nach schnellen Approximationen für den SBMD gesucht. Ein *starkes Linearzeit-Approximationsschema* (LTAS) der Ordnung r für das Problem SBMD ist ein Algorithmus, der, gegeben eine Menge P von n Punkten und einen Parameter $\varepsilon > 0$, in $O^*(n + \varepsilon^{-r})$ Zeit einen Baum berechnet, der P aufspannt und dessen Durchmesser höchstens um einen Faktor von $(1+\varepsilon)$ größer ist als der eines SBMD. Dabei versteckt die O^*-Notation Terme der Größenordnung $O(\text{polylog}\, \varepsilon^{-1})$.

Es ist nicht schwer, ein starkes LTAS der Ordnung 6 anzugeben: Sei Q ein kleinstes achsenparalleles Quadrat, das P enthält. Zerlege Q in $O(\varepsilon^{-1}) \times O(\varepsilon^{-1})$ gleichgroße quadratische Gitterzellen. Wähle dann für jeden Punkt in P einen nächsten Gitterpunkt, seinen *Repräsentanten*. Berechne für die $O(\varepsilon^{-2})$ Repräsentanten mit einem exakten Algorithmus einen SBMD $\mathcal{T}_\varepsilon$, der ohne Beschränkung der Allgemeinheit höchstens zwei Pole habe. Konstruiere nun aus $\mathcal{T}_\varepsilon$ einen Baum $\mathcal{T}$ für die Punktmenge P. Dabei werden die Punkte grob so untereinander verbunden wie ihre Repräsentanten in $\mathcal{T}_\varepsilon$. Man sieht schnell, dass der Durchmesser von $\mathcal{T}$ nur unwesentlich größer ist als der von $\mathcal{T}_\varepsilon$, der wiederum höchstens so groß ist wie der eines SBMD *aller* Punkte in P. Verwendet man als exakten Algorithmus den von Ho et al. [HLCW91], so erhält man ein starkes LTAS der Ordnung 6. Mit dem exakten Algorithmus von Chan [Cha03] reduziert sich die Ordnung geringfügig auf $5\frac{2}{3}$.

Nun haben wir uns gefragt, wie man die Ordnung des LTAS weiter vermindern kann.

Wir geben ein Approximationsschema an, das die Well-Separated Pair-Decomposition (WSPD) von Callahan und Kosaraju [CK95] benützt, eine geometrische Datenstruktur, die bereits am Anfang dieses Kapitels angesprochen wurde. Die WSPD hilft uns dabei, im dipolaren Fall die Suche nach den beiden Polen auf eine *lineare* Anzahl von Punktepaaren zu beschränken. Unser Approximationsschema benötigt $O(\varepsilon^{-3}n + \varepsilon^{-3}n \log n)$ Zeit und ist somit *kein* LTAS. Allerdings können wir es anstelle eines exakten Algorithmus' auf die oben definierten $O(\varepsilon^{-2})$ vielen Repräsentanten anwenden und bekommen so ein LTAS der Ordnung 5. Unabhängig von uns ist es Spriggs et al. [SKB+04] gelungen, ein starkes LTAS der Ordnung 3 für die Berechnung des SBMD anzugeben.

Die folgende Überlegung soll den Zusammenhang des SBMD zu anderen Problemen der Standortoptimierung (*facility location*) herausarbeiten. Der dipolare Spannbaum minimalen Durchmessers (dSBMD) hat zwei Pole x und y, die die Funktion $r_x + |xy| + r_y$ minimieren, wobei $|xy|$ der euklidische Abstand von x und y ist und r_x und r_y die Radien von Kreisscheiben mit Mittelpunkten x bzw. y bezeichnen, deren Vereinigung die ganze Punktmenge P enthält. (Man beachte, dass der dSBMD auch für Punktmengen definiert ist, die keinen SBMD haben, der dipolar ist.) Ein anderes Problem der Standortoptimierung ist das diskrete 2-Zentren-Problem (2ZP), in dem es darum geht, zwei Punkte x und y aus P so auszuwählen, dass $\max\{r_x, r_y\}$ minimiert wird. Für dieses Problem konnten Agarwal et al. [ASW98] den ersten subquadratischen Algorithmus angeben. Seine Laufzeit beträgt $O(n^{4/3} \log^5 n)$. (Falls die Anzahl der Zentren Teil der Eingabe ist, ist das Problem NP-schwer [GJ79].) Für das 2ZP lässt sich eine exakte Lösung also deutlich schneller bestimmen als für den dSBMD. Allerdings wird der Abstand zwischen den Zentren in der Zielfunktion überhaupt nicht berücksichtigt. Wenn die beiden Zentren aber miteinander in Kontakt stehen, etwa um Waren oder Personal auszutauschen, dann sollte ihr Abstand berücksichtigt werden.

Deshalb haben wir den *dipolaren Spannbaum minimaler Abstandssumme* (dSBMA) definiert, der die Funktion $|xy| + \max\{r_x, r_y\}$ minimiert. Dies stellt einen Kompromiss zwischen den beiden obigen Zielfunktionen dar. Wie sich herausstellt, gilt das auch für den Konstruktionsaufwand: wir geben einen Algorithmus zur Berechnung des dSBMA an, der für n Punkte in der Ebene $O(n^2 \log n)$ Zeit benötigt. Damit ist er asymptotisch langsamer als der 2ZP-Algorithmus von Agarwal et al., aber schneller als der SBMD-Algorithmus von Chan. Letzterer kann auch an unsere neue Zielfunktion angepasst werden, ist allerdings erst ab Dimension 5 schneller. Außerdem zeigen wir, dass der Durchmesser einer Variante des dSBMA, die innerhalb der gleichen Zeit berechnet werden kann, höchstens um ein Drittel länger ist als der Durchmesser eines SBMD (falls dieser nicht monopolar ist, d.h. sowieso schnell berechnet werden kann). Schließlich gilt, dass unser oben skizziertes LTAS für die Berechnung des SBMD modifiziert werden kann, so dass es stattdessen das 2ZP oder den dSBMA approximiert.

2.2 Spanner für achsenparallele Rechtecke

Am Anfang dieses Kapitels haben wir Spanner eingeführt. Ein t-Spanner ist ein geometrisches Netzwerk, dessen Streckfaktor durch t beschränkt ist. In einem t-Spanner gilt also für jedes Paar von (punktförmigen) Knoten, dass ihr Netzwerkabstand höchstens um einen Faktor von t länger ist als ihr euklidischer Abstand. Mit Tetsuo Asano, Mark de Berg, Otfried Cheong, Hazel Everett, Herman Haverkort und Naoki Katoh [AdBC+05] haben wir das Konzept von Spannern verallgemeinert, indem wir statt punktförmiger Knoten achsenparallele Rechtecke in der Ebene als Knoten zulassen. Es ist die erste Arbeit, die Spanner für nicht-punktförmige Knoten untersucht.

Wir nehmen an, dass die Netzwerkkanten achsenparallele Strecken sind, die Paare von Rechtecken miteinander verbinden. Diese Strecken nennen wir *Brücken*. Entsprechend messen wir den Abstand sowohl im Netzwerk als auch außerhalb mit der Manhattan- (oder L_1-) Norm. Dann ist der Streckfaktor für ein Paar von Punkten in der Vereinigung aller Rechtecke wieder die Länge eines kürzesten Weges im Netzwerk geteilt durch ihren Abstand. Der Streckfaktor des Netzwerks wird nun definiert als Supremum der Streckfaktoren über alle Paare von Punkten in der Vereinigung der gegebenen Rechtecke. Anders als bei Spannern für Punkte nehmen wir an, dass wir außer den Rechtecken auch einen Graphen bekommen, den so genannten *Brückengraphen*, der angibt, welche Rechtecke miteinander verbunden werden sollen. Das Problem besteht nun darin, die Brücken gemäß dem Brückengraphen so zu platzieren, dass der Streckfaktor des resultierenden Netzwerks minimiert wird. Als Anwendung wäre ein achsenparallel angelegter Gebäudekomplex denkbar, dessen Teile durch Fußgängerbrücken oder Tunnel miteinander verbunden werden sollen, etwa so wie der *Skywalk* Gebäude in den Innenstädten Minneapolis und St. Paul miteinander verbindet.

Wir zeigen zuerst, dass das Problem für allgemeine Brückengraphen NP-schwer ist. Dies beweisen wir durch Reduktion von dem NP-schweren Problem PARTITION [GJ79]. Für kreisfreie Brückengraphen (also Bäume) können wir das Problem mit linearem Programmieren (LP) lösen. Unsere LP-Formulierung benötigt $O(n^2)$ Variable und Beschränkungen. Falls der Brückengraph ein Pfad ist, können wir natürlich auch das LP verwenden. Allerdings haben wir für diesen Fall einen effizienteren Algorithmus, der in $O(n^3 \log n)$ Zeit läuft und auf Megiddos parametrischer Suche [Meg83] beruht. Falls zusätzlich gilt, dass alle Brücken vertikal sind, vermindert sich die Laufzeit weiter auf $O(n^2)$. In diesem Fall lässt sich das Problem auch besonders gut approximieren: wir können in $O(n \log \varepsilon^{-1})$ Zeit eine Brückenplatzierung berechnen, deren Streckfaktor höchstens um einen Faktor von $(1+\varepsilon)$ größer ist als der kleinstmögliche Streckfaktor. Das ist ein starkes LTAS der Ordnung 0.

2.3 Minimale Manhattan-Netzwerke

In der Einleitung zu diesem Kapitel war schon die Rede von der Bedeutung von euklidischen Spannern, also Spannern, bei denen der Streckfaktor eines Knotenpaares definiert ist als Quotient des Abstands der Knoten im Netzwerk und des euklidischen Abstands. Es ist klar, dass für eine gegebene Punktmenge P der einzige Graph mit Knotenmenge P und euklidischem Streckfaktor 1 der vollständige Graph ist. Im euklidischen Fall ist die Lage jeder Kante eindeutig bestimmt. Anders ist dies im Fall der Manhattan- (oder L_1-) Metrik, wo die Kanten eines 1-Spanners *Manhattan-Pfade* sind, also x- und y-monotone Polygonzüge aus achsenparallelen Strecken. Ein 1-Spanner für die Manhattan-Metrik heißt kurz *Manhattan-Netzwerk*. Es ist eine interessante Frage, wie sich der Freiheitsgrad beim Kantenlayout dazu benutzen lässt, so genannte *minimale* Manhattan-Netzwerke (MMN) zu konstruieren, also Manhattan-Netzwerke minimaler Gesamtlänge.

MMN könnten von Bedeutung in der Stadtplanung oder im VLSI-Layout sein. Eine ganz konkrete Anwendung gibt es jedoch in der algorithmischen Biologie. Lam et al. [LAP03] beschreiben einen dreistufigen Prozess zum Ausrichten von Gensequenzen. Im zweiten Schritt dieses Prozesses lösen sie eine Variante des Manhattan-Netzwerk-Problems, bei dem jeder Punkt nur mit den Punkten durch Manhattan-Pfade verbunden werden muss, die sowohl eine größere x- als auch y-Koordinate haben. In ihrer Anwendung entspricht ein Manhattan-Pfad einer Folge von Einfüge-, Lösch- und (Miss-) Match-Operationen, die man braucht, um zwei Gensequenzen ineinander zu überführen.

Lam et al. zeigen, dass eine Modifikation eines Algorithmus' von Gudmundsson et al. [GLN01] eine 2-Approximation für ihre Problemvariante liefert. In ihren Experimenten habe die Reduktion des Suchraums auf dieses Manhattan-Netzwerk die Berechnung einer guten Ausrichtung im dritten Schritt ihres Prozesses deutlich beschleunigt, berichten sie.

Interessanterweise ist die Komplexität des MMN-Problems immer noch nicht bekannt, obwohl es schon einige Approximationsalgorithmen dafür gibt. Gudmundsson et al. [GLN01] beschreiben eine 8-Approximation, die $O(n \log n)$ Zeit und linearen Speicher benötigt, sowie eine 4-Approximation, die kubische Zeit und quadratischen Platz braucht. Die gleiche Zeit- und Speicherplatzkomplexität besitzt eine 2-Approximation von Kato et al. [KIA02], bei der jedoch sowohl die Beschreibung des Algorithmus' als auch sein Korrektheitsbeweis unvollständig sind.

Mit meinem Doktoranden Marc Benkert, unserem Studienarbeiter Florian Widmann und mit Takeshi Shirabe [BWW05, BWWS06] haben wir die bisherigen Resultate verbessert und erweitert. Wir geben eine schnelle 3-Approximation an, die mit $O(n \log n)$ Zeit und linearem Speicher auskommt. Neu an unserer Herangehensweise ist, dass wir die Ebene in zwei Regionen aufteilen, in denen wir die Teile unseres Manhattan-Netzwerks getrennt berechnen. Auch unsere Analyse benutzt diese Aufteilung und vergleicht die Längen der Teilnetze getrennt mit den entsprechenden Teilen eines MMN. Die eine der beiden Regionen ist die Vereinigung von disjunkten Treppenstufenpolygonen, die wir jeweils in Pseudo-Rechtecke zerlegen. Dafür genügt uns eine (schnelle) 2-Approximation. Sie funktioniert ähnlich wie die 2-Approximation, die Gudmundsson et al. [GLN01] dazu verwenden, Treppenstufenpolygone in Rechtecke zu zerlegen.

Des Weiteren geben wir eine Formulierung des MMN-Problems als gemischt-ganzzahliges Programm (MIP) an. Dadurch konnten wir unsere 3-Approximation auf Beispielen kleiner bis mittlerer Größe mit exakten Lösungen vergleichen. Wir haben unseren Algorithmus implementiert und mit der 4-Approximation von Gudmundsson et al. auf zwei verschiedenen Klassen von zufälligen Beispielen verglichen. Dabei hat sich herausgestellt, dass unser Algorithmus im Allgemeinen Netzwerke berechnet die höchstens 50% länger sind als die entsprechenden minimalen. Während unser Algorithmus nur auf einer der beiden Beispielklassen besser abschneidet als der von Gudmundsson et al., liefert eine Variante unseres Algorithmus in unseren Experimenten deutlich bessere Ergebnisse. Allerdings konnten wir nicht beweisen, dass dies immer der Fall ist. Noch bessere Resultate lieferte allerdings eine sehr einfache LP-Heuristik, die unsere MIP-Formulierung relaxiert, alle nicht-ganzzahligen Variablen aufrundet und das entsprechende Manhattan-Netzwerk zurückgibt. Auf unseren zufälligen Beispielen lieferte diese Heuristik im schlechtesten Fall ein Manhattan-Netzwerk, das um nur 7.8% länger ist als das entsprechende MMN. Unsere 3-Approximation sowie die 4- und die 8-Approximation von Gudmundsson et al. können mithilfe eines Java-Applets im Internet getestet werden, siehe *http://i11www.iti.uni-karlsruhe.de/manhattan*.

Chepoi et al. [CNV05] haben später ebenfalls unsere MIP-Formulierung relaxiert. Durch cleveres Runden gelingt es den Autoren, einen Approximationsfaktor von 2 zu beweisen. Da ihre 2-Approximation vor dem Runden ein LP (mit kubisch vielen Variablen und Beschränkungen) lösen muss, ist sie natürlich bedeutend langsamer als unsere 3-Approximation. Allerdings scheinen ihre praktischen Resultate qualitativ mit denen unserer trivialen LP-Heuristik vergleichbar zu sein.

Chepoi et al. schlagen eine Verallgemeinerung des MMN-Problems vor, bei der der Benutzer angeben kann, welche Paare von Punkten durch Manhattan-Pfade miteinander verbunden werden sollen. Dieses Problem verallgemeinert aber auch das NP-schwere rektilineare Steiner-Arboreszenz-Problem [SS05], womit es selbst ebenfalls NP-schwer ist. Offen bleibt die Frage nach der Komplexität des ursprünglichen MMN-Problems.

2.4 Interferenz-minimale Netzwerke

Drahtlose Ad-hoc-Netzwerke bestehen aus miteinander kommunizierenden Geräten, die sich innerhalb einer bestimmten geografischen Region aufhalten und die typischerweise mit geringer Sendeleistung auskommen müssen. Daher ist es sinnvoll, die Kommunikation der Geräte untereinander so zu beschränken, dass nur solche Paare direkt miteinander kommunizieren können, bei denen der Energieaufwand klein ist und die geringe Interferenz verursachen, d.h. deren Kommunikation möglichst wenige andere Geräte stört. Da das tatsächliche Kommunikationsverhalten vorab unbekannt ist, versucht man, ein Netzwerk zu finden, das einen Parameter optimiert, der ein guter Indikator dafür ist, dass im späteren Betrieb keine oder nur geringe Interferenz auftritt.

Das folgende Interferenz-Modell wurde von Burkhart et al. [BvRWZ04] vorgeschlagen und auch von Moaveni-Nejad und Li [ML05] verwendet. Es sieht vor, dass jede Kante des Netzwerks (also jede Direktverbindung) einen Interferenzwert besitzt, der der Anzahl der anderen Teilnehmer entspricht, die beim Benutzen dieser Direktverbindung potentiell gestört werden. Wenn ein Knoten u mit einem Knoten v kommunizieren will, so stört er alle Knoten, die von u nicht weiter entfernt sind als v – und umgekehrt. Da laut Prakash [Pra99] Einweg-Kommunikation ohne Bestätigungsmeldung problematisch ist, wird davon ausgegangen, dass der Kommunikationsgraph ungerichtet ist und der Interferenzwert der Kante $\{u, v\}$ gerade der Anzahl aller Knoten entspricht, die sich in der Vereinigung der beiden Kreise mit Radius $|uv|$ und Mittelpunkten u und v befinden, siehe Abbildung 2.1. Die Interferenz eines Netzwerks wird als maximale Kanteninterferenz modelliert.

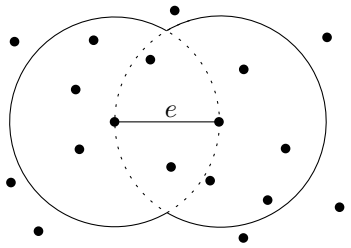


Abb. 2.1: Kante e hat Interferenz 9.

Nun könnte man ein Netzwerk berechnen, das sowohl die maximale Kanteninterferenz als auch die Summe der Kanteninterferenzen minimiert, indem man mit Prims Algorithmus [Pri57] in $O(m+n \log n)$ Zeit einen minimalen Spannbaum (MST) für die Menge der Knoten berechnet, wobei $m \in \binom{n}{2}$ die Anzahl der zur Verfügung stehenden Kanten ist. Allerdings müssten Signale in so einem Baum-Netzwerk unter Umständen große Umwege zurücklegen. Um das zu verhindern, kann man verlangen, dass das Netzwerk ein geometrischer Spanner ist. Burkhart et al. [BvRWZ04] haben den ersten Algorithmus angegeben, der für ein gegebenes $t > 1$ einen t-Spanner mit minimaler Interferenz liefert, ohne auf dessen Laufzeit einzugehen. Ihr Ergebnis wurde später von Moaveni-Nejad und Li [ML05] verbessert. Wie Burkhart et al. gehen sie davon aus, dass die Interferenz jeder Kante vorab bekannt ist. Unter dieser Voraussetzung benötigt ihr Algorithmus $O(n \log n(m + n \log n))$ Zeit. Falls alle $\binom{n}{2}$ Kanten betrachtet werden, benötigt ihr Algorithmus also $O(n^3 \log n)$ Zeit.

Mit Marc Benkert, Joachim Gudmundsson und Herman Haverkort [BGHW06] habe ich die Resultate von Burkhart et al. sowie die von Moaveni-Nejad und Li verbessert und erweitert. Erstens untersuchen wir außer Spannbäumen und t-Spannern auch so genannte d-Hop-Netzwerke, in denen es für jedes Paar von Knoten eine Verbindung gibt, die aus höchstens d Kanten (oder „Hops“) besteht. Dieses Modell ist sinnvoll, da Signale typischerweise den größten Teil ihrer Reisezeit beim Empfangen und Weiterversenden verbringen.

Zweitens beseitigen wir die Annahme, dass die Interferenzen der Kanten vorab gegeben sind. Wir geben für jede der drei oben erwähnten Eigenschaften (Spannbaum, t-

Spanner, d-Hop-Netzwerk) Entscheidungsalgorithmen an, die testen, ob es ein Netzwerk mit Interferenz k gibt, das die gegebene Eigenschaft besitzt, sowie darauf aufbauende Optimierungsalgorithmen.

	exaktes Modell (erwartet)	Fuzzy-Modell (determ.)
Spannbaum	$\min\{n^{9/4}\operatorname{polylog} n, nk^2 + nk\log n\}$	$\frac{n}{\varepsilon^2}(\frac{1}{\varepsilon} + \log n)$
t-Spanner	$n^2 \log k(k + \log n)$	$n^2 \log k(\frac{1}{\varepsilon^2} + \log n)$
d-Hop-Netzwerk	$\min\{n^{9/4}\operatorname{polylog} n, nk^2\} + n^2 \log n \log k$	$\frac{n}{\varepsilon^2}(\frac{1}{\varepsilon} + n \log k)$

Tab. 2.1: Laufzeiten (in Groß-Oh-Notation) unserer Algorithmen zur Berechnung von Netzwerken, deren Interferenz minimal ist unter aller Netzwerken mit der angegebenen Eigenschaft. Dabei ist n die Anzahl der Knoten, k die Interferenz des berechneten Netzwerks und ε gibt die relative Ungenauigkeit an, mit der die Reichweite eines Signals (sowie, im Fall des Spanners, der Streckfaktor) bekannt ist.

Die Entscheidungsalgorithmen lassen sich auf zwei Arten benützen. In Alternative (A) berechnen wir alle $\binom{n}{2}$ Interferenzwerte im Voraus durch *Kreisanfragen* (*circular range searching*) in $O(n^{9/4}\operatorname{polylog} n)$ Zeit. Dann führen wir eine kombinierte exponentielle und binäre Suche durch, die den entsprechenden Entscheidungsalgorithmus $O(\log k)$ mal aufruft, um das kleinste k zu bestimmen, für das es ein Netzwerk mit Interferenz k und der gewünschten Eigenschaft gibt. In Alternative (B) berechnen wir keine Interferenzwerte vorab. Stattdessen berechnen wir in jedem Schritt der gerade erwähnten Suche eine Obermenge der Kanten, deren Interferenz durch das aktuelle k beschränkt ist. Diese Obermenge besteht aus den Kanten der *Delaunay-Triangulierung der Ordnung k*. Diese Zerlegung der Ebene hat $O(nk)$ Kanten und kann in (erwartet) $O(nk \log k + n \log n)$ Zeit berechnet werden [GHvK02]. Alternative (B) ist besonders schnell für kleine Werte von k. Beide Alternativen sind (im Gegensatz zu den bisherigen Algorithmen) in dem Sinne ausgabesensitiv, dass ein Netzwerk der Interferenz k höchstens $O(nk)$ Kanten besitzt. Außerdem sind sie für alle $k \in o(n)$ schneller als der Algorithmus von Moaveni-Nejad und Li [ML05], siehe mittlere Spalte von Tabelle 2.1. Die Worst-Case-Laufzeiten von deterministischen Algorithmen für das exakte Modell sind etwas schlechter als die in der Tabelle angegebenen erwarteten Laufzeiten.

Die obigen Algorithmen nehmen an, dass das Signal, das von Knoten u an v gesandt wird, *genau* die Knoten erreicht und somit stört, die nicht weiter von u entfernt sind als v. Neben diesem *exakten* Modell stellen wir ein *Fuzzy-Modell* vor, wo wir die realistische Annahme machen, dass nicht genau bekannt ist, wie weit das Signal reicht. Statt also davon auszugehen, dass das Signal genau die Knoten mit Abstand höchstens $|uv|$ von u erreicht, sagen wir im Fuzzy-Modell, dass alle Knoten erreicht werden, die höchstens Abstand $|uv|$ von u haben und kein Knoten erreicht wird, der mindestens Abstand $(1 + \varepsilon)|uv|$ von u hat. Über Knoten dazwischen machen wir keine Aussage. Dieses Modell erlaubt es besonders für große Werte von k zu schnelleren Algorithmen zu kommen, siehe die Worst-Case-Laufzeiten in der rechte Spalte von Tabelle 2.1. Wie in Abschnitt 2.1 machen wir uns bei diesen Algorithmen die Well-Separated Pair-Decomposition [CK95] zunutze.

2.5 Konfigurationen mit wenigen Kreuzungen

Eine natürliche Verallgemeinerung von geometrischen Graphen, bei denen Kanten geradlinig gezeichnet sein müssen, sind *topologische* Graphen, die so in die Ebene eingebettet sein müssen, dass sich keine zwei Kanten in einer unbeschränkte Zahl von Punkten schneiden. Es ist schon länger bekannt, dass es NP-schwer ist, in topologischen Graphen Subgraphen zu finden, die unter allen Subgraphen einer gewissen Konfiguration kreuzungsminimal sind. Kratochvíl et al. [KLN91] haben das für die Konfigurationen Spannbaum, s-t-Pfad, Kreis, Matching sowie 1- und 2-Faktor gezeigt. (Zur Erinnerung: ein r-Faktor ist ein Teilgraph, in dem alle Knoten des Ausgangsgraphen genau Grad r haben.) Später haben Jansen und Woeginger [JW93] einige dieser Aussagen verschärft, indem sie gezeigt haben, dass sie für die Konfigurationen Spannbaum, 1- und 2-Faktor auch in geometrischen Graphen gelten, und zwar sogar dann, wenn die Kanten des Graphen nur zwei verschiedene Längen oder zwei verschiedene Steigungen haben.

Mit Christian Knauer, Andreas Spillner und meinem Doktoranden Étienne Schramm [KSSW05, KSSW07] haben wir diese Negativ-Resultate weiter verschärft, andererseits aber auch durch positive Ergebnisse ergänzt. Wir haben zuerst gezeigt, dass keines der Probleme gut approximierbar ist. Genauer: ist G ein geometrischer Graph, k die Anzahl der Kreuzungen von G, $\phi(G)$ die minimale Anzahl der Kreuzungen in einem Spannbaum von G und $\varepsilon > 0$ beliebig, dann ist es NP-schwer, $\phi(G)$ besser zu approximieren als mit einem Faktor von $k^{1-\varepsilon}$. Entsprechende Aussagen gelten für die anderen Konfigurationen.

Auf der positiven Seite konnten wir für den Parameter k sehr einfache Fest-Parameter-Algorithmen angeben, die entscheiden, ob ein Graph einen kreuzungsfreien Subgraphen einer gewissen Konfiguration besitzt. Sie basieren auf der trivialen Beobachtung, dass höchstens $2k$ Kanten an einer Kreuzung beteiligt sind. Also genügt es, alle 4^k Teilmengen dieser Kanten durchzugehen und zu prüfen, ob eine dieser Teilmengen kreuzungsfrei ist und zusammen mit den verbliebenen Kanten die gewünschte Konfiguration besitzt. Die Basis des exponentiellen Teils der Laufzeit lässt sich verbessern, wenn man sich bei dieser Aufzählung auf die höchstens 2^k *inklusionsmaximalen* planaren Teilmengen beschränkt. Wir wissen nicht, wie sich dieses Ergebnis für Matchings, 1- oder 2-Faktoren verbessern lässt. Aber auch für die Konfiguration Spannbaum ist es erstaunlich schwer, den 2^k-Term zu verringern. Wir konnten schließlich einen Algorithmus angeben, der das Entscheidungsproblem in $O(m + k1.9999992^k)$ Zeit löst, wobei m die Anzahl der Kanten von G ist. Etwas besser wird die Laufzeit, wenn wir den Zufall zur Hilfe nehmen: es gibt einen Monte-Carlo-Algorithmus mit einseitigem Fehler und Erfolgswahrscheinlichkeit größer $1/2$, der das Entscheidungsproblem in $O(m + k \cdot 1.968^k)$ Zeit löst. Hinter beiden Algorithmen steckt die Beobachtung, dass man für eine Zusammenhangskomponente K nicht alle 2^d Möglichkeiten betrachten muss, wie man K mit dem Restgraphen verbindet, sondern nur $2^d - 1$ Möglichkeiten, wobei d die Anzahl der zu K inzidenten Kanten ist. Grund: die Möglichkeit, keine inzidenten Kanten zu wählen, scheidet aus.

Für die Konfigurationen s-t-Pfad und Kreis beträgt der exponentielle Teil jeweils β^k, wobei $\beta = \sqrt{3} \approx 1.73$. Für alle Konfigurationen lässt sich das Optimierungsproblem durch wiederholtes Aufrufen des Entscheidungsalgorithmus' lösen. Dadurch erhöht sich die Basis im exponentiellen Teil der Laufzeit um 1.

Für große Werte von k sind obige Fest-Parameter-Algorithmen zu langsam. Daher haben wir die Optimierungsprobleme für alle Konfigurationen auch als gemischt-ganzzahlige Programme formuliert. Der „Trick“ besteht darin, etwa für die Konfigurationen Spannbaum und s-t-Pfad den Zusammenhang geschickt als ganzzahligen Fluss zu modellieren. Für größere Graphen ist aber auch dieser Ansatz zu langsam. Deshalb haben wir für die Konfiguration Spannbaum eine einfache und relativ schnelle Heuristik angegeben

[KSSW05]. Wir haben sowohl die Heuristik als auch das entsprechende MIP implementiert und auf zufälligen und echten Daten getestet. Zu unserer Überraschung hatten die von der Heuristik in Sekundenbruchteilen berechneten Spannbäume für zufällige Graphen mit 20 Knoten und 25–60 Kanten im Durchschnitt weniger als eine Kreuzung mehr als die vom MIP berechneten optimalen Spannbäume. Aber auch auf echten Daten und größeren Graphen lieferte die Heuristik Ergebnisse nicht weit vom Optimum.

2.6 Das City-Voronoi-Diagramm

Das *Voronoi-Diagramm* ist eine bekannte geometrische Datenstruktur, neben konvexer Hülle und Delaunay-Triangulierung ein echter Klassiker der algorithmischen Geometrie. Das Voronoi-Diagramm einer Punktmenge P zerlegt einen zugrunde liegenden metrischen Raum, etwa die euklidische Ebene, derart in Regionen, dass all die Punkte in einer Region zusammengefasst werden, für die die Antwort auf die Frage nach den nächsten Nachbarn in P gleich ist. Aus Anwendersicht liefert das Voronoi-Diagramm die Antwort auf das Postamt-Problem. Wenn P eine Menge von Postämtern in meiner Umgebung ist und ich mich im Voronoi-Diagramm von P lokalisieren kann, dann weiß ich, welches Postamt für mich das nächste ist. In der euklidischen Ebene hat das Voronoi-Diagramm lineare Komplexität, seine Regionen sind konvex, und es kann für n Postämter in $O(n \log n)$ Zeit [dBvKOS00] berechnet werden. Wie in allen ebenen Partitionen lässt sich ein Punkt im Voronoi-Diagramm in logarithmischer Zeit lokalisieren [dBvKOS00].

Interessant wird das Problem, wenn statt Entfernungen Reisezeiten gemessen werden und wenn diese Reisezeiten durch Transportnetzwerke verkürzt werden. Im euklidischen Fall wird zum Beispiel das *Airlift-Voronoi-Diagramm* betrachtet, bei dem der Zugang zum Transportnetzwerk auf s Stationen beschränkt wird. In dieser Situation sind die Voronoi-Regionen nicht mehr unbedingt zusammenhängend. Trotzdem kann es in $O((n+s)\log(n+s)+c)$ Zeit [OB05] berechnet werden, wobei c die Anzahl der Transportlinien ist. Aichholzer et al. [AAP02] betrachten eine andere Art von Transportnetzwerk, nämlich achsenparallele Strecken, die man sich am besten als bi-direktionale Fließbänder vorstellt, da sie überall betreten und verlassen werden können. Auf ihnen bewegt man sich mit einer konstanten Geschwindigkeit fort, die höher ist als außerhalb des Netzwerks. Außerdem nehmen sie an, dass Entfernungen mithilfe der Manhattan- (oder L_1-) Metrik gemessen werden, was die Situation in amerikanischen Großstädten gut modelliert. Deshalb nennen sie die Metrik, die durch das Transportnetzwerk induziert wird, *City-Metrik* und das resultierende Voronoi-Diagramm *City-Voronoi-Diagramm*. Sie zeigen, dass dieses Diagramm lineare Komplexität in der Anzahl n der Postämter und der Anzahl c der Fließbänder besitzt, obwohl seine Regionen nicht konvex sind. Außerdem geben sie einen Algorithmus an, der in $O(n \log n + c^2 \log c)$ Zeit eine Verfeinerung des City-Voronoi-Diagramms berechnet, eine so genannte *quickest Path-Map* (QPM). Mithilfe der QPM kann man in $O(L + \log(n + c))$ Zeit für einen beliebigen Anfragepunkt einen kürzesten Weg zu seinem nächsten Postamt berechnen, wobei L die Komplexität des Weges ist.

Mit meinem Diplomanden Robert Görke und mit Chan-Su Shin [GW05, GSW07] habe ich einen anderen Algorithmus zur Berechnung einer QPM angegeben. Seine Laufzeit beträgt $O((n + c)\ \mathrm{polylog}(n + c))$ und hängt damit weniger stark von der Anzahl c der Fließbänder ab als die Laufzeit des Algorithmus' von Aichholzer et al.. Um genau zu sein, ist unser Algorithmus schneller, wenn $c \in \Omega(\sqrt{n}\log^3 n)$. Obwohl wir die QPM von Aichholzer et al. weiter verfeinern, hat sie immer noch lineare Komplexität. Am Zeitaufwand von $O(L + \log(n + c))$ für Schnellste-Wege-Anfragen ändert sich nichts.

Wie Aichholzer et al. verwenden auch wir eine Wellenfront, die sich ab dem Zeitpunkt

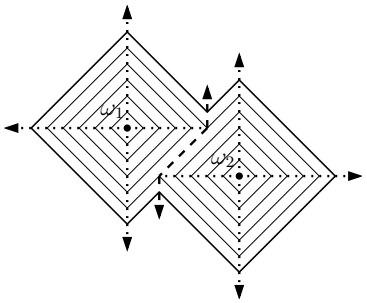


Abb. 2.2: Dort wo sich die Wellenfrontteile von zwei Postämtern treffen, verläuft die Grenze ihrer Voronoi-Regionen.

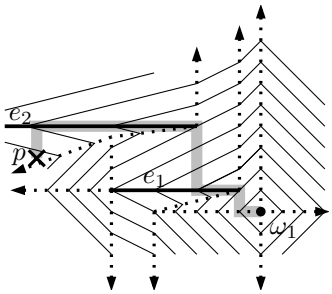


Abb. 2.3: Die Wellenfrontausdehnung liefert für jeden Anfragepunkt p den schnellsten Weg zum nächsten Postamt.

$t = 0$ gleichzeitig von allen Postämtern ausgehend über die ganze Ebene ausbreitet. Die Wellenfront zum Zeitpunkt $t \geq 0$ besteht aus allen Punkten, die vom nächsten Postamt Abstand t in der City-Metrik haben. Immer dort, wo Teile der Wellenfront von verschiedenen Postämtern aufeinander prallen, verläuft die Grenze zwischen den Voronoi-Regionen der beteiligten Postämter, siehe die Wellenfront, die in Abbildung 2.2 von den Postämtern ω_1 und ω_2 ausgeht. Die Wellenfront verändert immer dann ihre Form, wenn ein so genanntes *Ereignis* eintritt, etwa wenn die Wellenfront auf ein Fließband stößt, siehe die Fließbänder e_1 und e_2 in Abbildung 2.3. Die ganze Schwierigkeit der Konstruktion der QPM besteht in der effizienten Vorhersage solcher Ereignisse, denn die Spuren, die die Ecken der Wellenfront hinterlassen, liefern die Gebiete der QPM. In Abbildung 2.3 werden die Grenzen dieser Gebiete durch die gepunkteten Pfeile markiert. In jedem Gebiet breitet sich die Wellenfront monoton aus. Jedes Gebiet besitzt einen ausgezeichneten Startpunkt, der zuerst von der Wellenfront erreicht wird. Wenn man nun von einem Anfragepunkt p aus entgegen der Wellenfrontausbreitung von Startpunkt zu Startpunkt springt, so gelangt man auf einem schnellsten Weg zum nächsten Postamt (ω_1 in Abbildung 2.3).

Zur effizienten Vorhersage der Ereignisse unterteilen wir die QPM in Regionen konstanter Komplexität, die wir *Streifen* nennen. Ein Streifen ist der Teil eines Gebietes, der von der Wellenfront zwischen zwei aufeinander folgenden, das Gebiet betreffenden Ereignissen überstrichen wird. Wir heben die Ausbreitung der Wellenfront in den dreidimensionalen Raum, wobei die dritte Achse die Zeit modelliert. Dort betrachten wir die bereits erwähnten Streifen sowie *Strahlen*, auf denen sich die Ecken der Wellenfront fortbewegen. Es stellt sich heraus, dass sich alle Typen von Ereignissen auf Kollisionen zwischen Streifen und Strahlen zurückführen lassen. Diese Kollisionen lassen sich mithilfe einer Technik von Eppstein und Erickson [EE99] effizient vorhersagen, wenn man Datenstrukturen für so genannte *Minimierungsanfragen* zur Verfügung stellt. Aufgrund der konstanten Komplexität unserer Streifen lassen sich diese Minimierungsanfragen nach einer geometrischen Transformation der beteiligten Objekte mittels orthogonaler Bereichsanfragen [dBvKOS00] implementieren. Die Verwendung der orthogonalen Bereichsanfragen ruft den polylogarithmischen Anteil an der Laufzeit unseres Konstruktionsalgorithmus' hervor.

Unsere Technik kann als Instanz einer typischen Herangehensweise an gewisse geometrische Probleme betrachtet werden. Wenn mit komplexen geometrischen Objekten (hier den Regionen des City-Voronoi-Diagramms) gearbeitet wird, hilft es, wenn die fol-

genden drei Anforderungen erfüllt werden. Erstens ist es wichtig, die Objekte in einfache Teile (möglichst konstanter Komplexität) zu zerlegen, so dass sie schnell verarbeitet werden können. Zweitens sollte die Zerlegung nicht zu einer substantiellen Erhöhung der Anzahl der Objekte führen. Und drittens müssen die vereinfachten Objekte helfen, das Problem auf den ursprünglichen Objekten effizient(er) zu lösen. Ein weiteres Beispiel für diese Herangehensweise sind *Trapezzerlegungen*, eine bekannte Datenstruktur zum Lokalisieren von Anfragepunkten in ebenen Unterteilungen. Eine Trapezzerlegung zerlegt die im Allgemeinen sehr komplizierten Facetten einer ebenen Unterteilung in Trapeze, also in Objekte konstanter Komplexität. Aufgrund dessen kann man über diesen einfachen Objekten dann effiziente Suchstrukturen konstruieren, zum Beispiel mit randomisierten Algorithmen [dBvKOS00]. Wenn man mit diesen Strukturen für einen Anfragepunkt das Trapez bestimmt, das ihn enthält, so hat man automatisch auch die Facette bestimmt, die den Anfragepunkt enthält.

Robert Görke wurde für seinen Vortrag über unsere Arbeit [GW05] beim *International Symposium on Voronoi Diagrams in Science and Engineering* ausgezeichnet mit dem *Preis für die beste Präsentation eines jungen Vortragenden.*

Kapitel 3

Visualisierung geometrischer Netzwerke

Die Visualisierung von geometrischen Netzwerken ist ein Spezialgebiet des Graphenzeichnens, ein noch relativ junges Gebiet zwischen Graphentheorie, Algorithmik, Geometrie und Kognitionswissenschaften, das sich aber mit zwei Monographien [DETT99, KW01] und einer jährlichen internationalen Konferenz (mit Tagungsbänden in der Serie *Lecture Notes in Computer Science*) schon etabliert hat. Die Bedeutung des Gebiets wird noch weiter wachsen mit der Menge des sehr komplexen Informationsmaterials, das in unserer Medienwelt sowie in allen Wissenschaften anfällt und das man visualisieren muss, um es verständlicher zu machen.

Anders als beim klassischen Graphenzeichnen sind die Knoten geometrischer Netzwerke bereits mit einem Ort etwa in der Ebene assoziiert. Das Hauptproblem des Graphenzeichnens, also die Einbettung der Knoten in die Zeichenebene, scheint also zu entfallen. Falls der Ort der Knoten nicht verändert werden darf, bleibt die Frage nach der Art und Weise, wie die Kanten gezeichnet werden sollen, die diese Knoten miteinander verbinden. Mit dem Fall, dass auch diese eine vorgegebene Lage besitzen, beschäftigt sich die Kartografie beim Erstellen von Straßen-, Bahn- oder Gewässerkarten. Dabei stellt sich das überaus wichtige Problem der *Generalisierung* [MLW95]: wie kann man aus Datenmaterial eines großen Maßstabs übersichtliche Karten kleinerer Maßstäbe erzeugen? Dabei müssen Objekte vereinfacht, vergrößert, verschoben, gelöscht, verschmolzen oder auch getrennt werden, um Information lesbar zu kodieren – anders als etwa beim kontinuierlichen Herauszoomen am Bildschirm. Dies ist Thema einer großen Menge kartografischer Literatur, die sich natürlich auch mit der Automatisierung dieses Prozesses befasst, etwa beim Zeichnen von Straßennetzwerken [MM97].

Schematische Karten spielen eine wichtige Rolle im öffentlichen Nah- oder Fernverkehr, in der Modellierung von Verkehrs- oder Warenströmen sowie bei der Planung und Verwaltung von Leitungssystemen etwa von Elektrizität oder (Ab-) Wasser. Abgesehen von [CdBvK05] verwenden alle bisherigen Verfahren (etwa [AM00]) lokale Operatoren, die iterativ angewendet werden, bis gewisse Abbruchkriterien erfüllt sind und das Resultat (hoffentlich) zufrieden stellend aussieht. In [CdBvK05] werden die Knoten des gegebenen Netzwerks als fix angesehen. Der Benutzer kann nun Anfragen der Art stellen: „Gibt es eine Schematisierung mit Verbindungen vom Typ achsenparallel-diagonal-achsenparallel?“ Wenn ja, so liefert der Algorithmus (in $O(n \log^3 n)$ Zeit) eine solche, wenn nein, so „beweist“ er, dass sie nicht existiert. Eine Erweiterung dieses Algorithmus' [CvK02], die

Benutzerinteraktion erlaubt, wurde implementiert und kann online ausprobiert werden[1].

Wie das Zeichnen von Graphen ist auch das Platzieren von Beschriftungen im allgemeinen ein schwieriges Problem [FW91, KR92, Wol00]. Um das Interesse an einer Automatisierung nachvollziehen zu können, muss man sich vergegenwärtigen, dass das manuelle Beschriften für schätzungsweise 50% der Produktionszeit einer Landkarte verantwortlich ist [Mor80]. Zirka 200 wissenschaftliche Artikel beschäftigen sich mit der Beschriftungsproblematik [WS96], außerdem fast 20 Dissertationen – darunter meine eigene [Wol99].

Beim allgemeinen Beschriftungsproblem ist eine Menge von Objekten gegeben, die beschriftet werden sollen, und zu jedem Objekt eine Menge möglicher Beschriftungskandidaten mit einer gewissen räumlichen Ausdehnung. Nun geht es meist darum, entweder eine möglichst große Teilmenge der Objekte ohne Überlappung zu beschriften oder alle Objekte mit möglichst großen Beschriftungen zu versehen. Aufgrund seiner Komplexität gibt es für dieses allgemeine Beschriftungsproblem nur Heuristiken. Während einige das Problem als maximale unabhängige Menge darstellen [WWKS01], greifen die meisten auf allgemeine Optimierungsverfahren wie Simulated Annealing [ECMS97] oder genetischen Algorithmen [Rai98, vD01] zurück. Andererseits existieren für viele interessante Spezialfälle beweisbar gute Algorithmen, so genannte *Approximationsalgorithmen*, zum Beispiel für das Beschriften von Punkten mit möglichst großen kongruenten Rechtecken [FW91]. Auch der Spezialfall, dass Punkte auf einer Geraden (z.B. einer U-Bahn-Linie) liegen, wurde schon untersucht [GIM+01].

Bei allen bisherigen Ansätzen zur Schematisierung wird die Beschriftung von Knoten und Kanten nicht beim Zeichnen der Netzwerke berücksichtigt, sondern höchstens im Nachhinein. Im Gegensatz dazu gibt es beim orthogonalen Zeichnen von Graphen schon Bestrebungen, die beiden Arbeitsschritte zu verknüpfen, und zwar mittels gemischt-ganzzahligem linearem Programmieren [KM03, BDLN05].

Im Rest dieses Kapitels fasse ich meine Arbeiten zur Visualisierung geometrischer Netzwerke zusammen. In Abschnitt 3.1 zeige ich, wie man mithilfe von gemischt-ganzzahliger Programmierung die Erstellung qualitativ hochwertiger U-Bahn-Linienpläne und anderer technischer Zeichnungen automatisieren kann. Die folgenden Abschnitte befassen sich mit Beschriftungsproblemen. In Abschnitt 3.2 gehe ich darauf ein, wie man dichte Punktwolken beschriftet. Dabei werden die Beschriftungen, im folgenden auch *Label* genannt, durch Pfeile mit den Punkten verbunden, die sie beschriften. Durch die Kombination von Labeln und Pfeilen entstehen interessante Probleme im Grenzbereich von Beschriftungsplatzierung, Graphenzeichnen und VLSI-Layout. In Abschnitt 3.3 beschäftige ich mich mit der Beschriftung von gewichteten Punkten. Das ist natürlich von großer praktischer Bedeutung, führt allerdings zu schwierigen theoretischen Problemen, deren Komplexität (genauer Approximierbarkeit) immer noch nicht bekannt ist. In Abschnitt 3.4 beschäftige ich mich mit Approximationsalgorithmen für die Beschriftung von Punkten mit je zwei Labeln, was etwa für Wetterkarten eine Rolle spielt.

3.1 U-Bahn-Linienpläne

U-Bahn-Linienpläne sind ein verbreiteter Spezialfall schematischer Karten. Wie bei schematischen Straßenkarten möchte man durch die Einschränkung auf wenige Zeichenrichtungen (horizontal, vertikal und die beiden Diagonalen) erreichen, dass die Pläne gut lesbar sind. Es gibt allerdings eine Reihe von Besonderheiten, die das Spezialproblem

[1] siehe `www.cs.uu.nl/archive/sw/schematic-map`

interessant machen: zum Beispiel gibt es zwischen Umsteigebahnhöfen oft viele Grad-2-Knoten, die meist äquidistant gezeichnet werden. Außerdem spielen die meist farbig hervorgehobenen U-Bahn-Linien eine wichtige Rolle: sie sollen in Umsteigebahnhöfen möglichst nicht abknicken, damit es Betrachter leicht haben, ihrem Verlauf zu folgen. Aufgrund der vielen (auch lokalen) Besonderheiten sind alle heute existierenden U-Bahn-Linienpläne von Grafikdesignern gezeichnet worden. Das Buch von Ovenden [Ove03] gibt einen schönen Überblick. Die Mutter aller modernen U-Bahn-Linienpläne, die sich von der zugrunde liegenden Geometrie gelöst haben, ist die Karte der Londoner *Tube* von Henry Beck [Gar94] aus dem Jahre 1933.

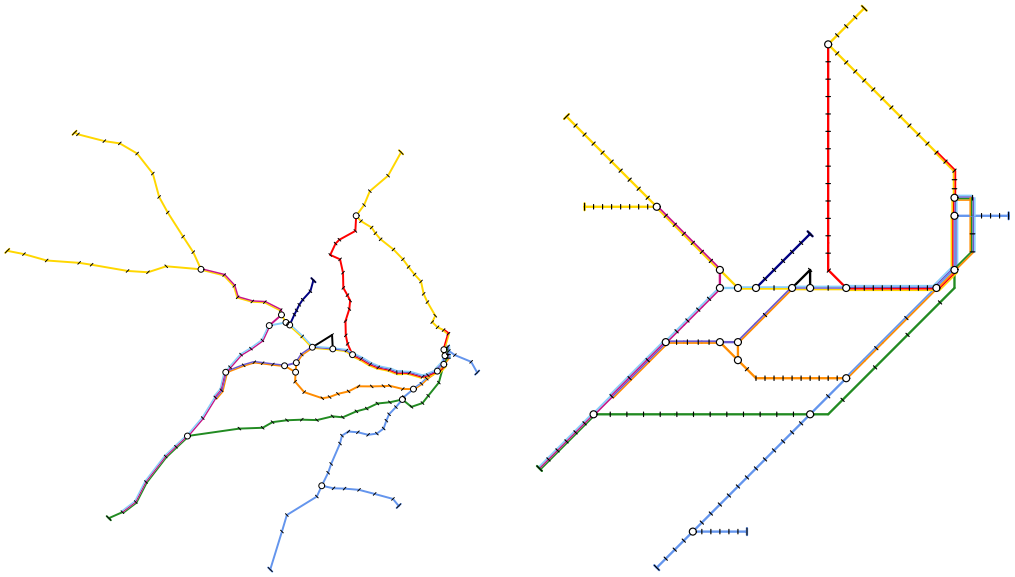
Die drei bisherigen Versuche, das Zeichnen von U-Bahn-Linienpläne zu automatisieren, basieren auf iterativer lokaler Optimierung: Barkowsky et al. [BLR00] verwenden ein allgemeines Verfahren zur Vereinfachung von Linienzügen. Hong et al. [HMdN05] verwenden einen Spring-Embedder, also ein physikalisches Modell, in dem sich adjazente Knoten anziehen und nicht-adjazente Knoten abstoßen. Stott und Rodgers [SR04] schließlich verwenden Hill-Climbing, um multikriteriell zu optimieren. Abbildung 3.1 zeigt am Beispiel des CityRail-Netzwerks von Sydney die Vorzüge und Nachteile der beiden letztgenannten Arbeiten.

(a) Ausschnitt von Abb. 7(b) in [HMdN05]. **(b)** Abb. 15 in [SR04].

Abb. 3.1: Verschiedene Zeichnungen des CityRail-Netzwerks von Sydney.

Mit meinem Diplomanden Martin Nöllenburg [NW06] habe ich mich ebenfalls mit dem Zeichnen von U-Bahn-Linienplänen beschäftigt. Unser Hauptanliegen war es, wenn möglich die Oktilinearität zu garantieren, d.h. dass alle Kanten parallel zu einer Koordinatenachse oder zu einer Winkelhalbierenden der Achsen verlaufen müssen. Allerdings ist es im oktilinearen (im Gegensatz zum orthogonalen) Fall möglich, dass zu einer oktilinearen Einbettung keine kreuzungsfreie Zeichnung existiert [BT04].

Martin Nöllenburg konnte in seiner Diplomarbeit [Nöl05] zeigen, dass eine wesentlich stärkere Aussage gilt: es ist nämlich NP-schwer zu entscheiden, ob ein eingebetteter planarer Graph eine oktilineare Zeichnung besitzt. Dies ist erstaunlich, weil Tamassia in einer schon klassischen Arbeit [Tam87] durch Reduktion auf ein Flussproblem zeigen



(a) Tatsächlicher Linienverlauf. **(b)** Ausgabe unseres MIPs.

Abb. 3.2: Das CityRail-Netzwerk von Sydney.

konnte, dass man im orthogonalen Fall (d.h. wenn man Diagonalen ausschließt) sogar knickminimale Zeichnungen effizient finden kann.

Es hat sich herausgestellt, dass die Hauptarbeit bei der praktischen Lösung des Problems darin bestand, das Problem gut zu modellieren. Nachdem wir viele echte U-Bahn-Linienpläne [Gar94, Ove03] studiert hatten, haben wir Listen von harten und weichen Anforderungen an gute Pläne formuliert. Die harten Anforderungen umfassen die Erhaltung der gegebenen Einbettung, die Oktilinearität der Kanten und die Einhaltung gewisser Minimalabstände zwischen Knoten. Zu den weichen Anforderungen gehört, dass U-Bahn-Linien möglichst gerade durch Umsteigebahnhöfen gehen sollten, dass die Gesamtkantenlänge klein sein sollte und dass die *relative Lage* adjazenter Knoten so gut wie möglich erhalten werden sollte. Das Konzept der relativen Lage haben wir modelliert, in dem wir den Winkelraum um jeden Knoten v in acht gleichgroße Sektoren aufteilen und verlangen, dass jeder zu v adjazente Knoten im gleichen Sektor bezüglich v liegt wie im gegebenen Layout – oder in einem der beiden benachbarten Sektoren.

Es ist uns dann gelungen, ein gemischt-ganzzahliges lineares Programm (MIP) für das Zeichnen von U-Bahn-Linienplänen anzugeben. Falls eine Zeichnung existiert (und das war bei unseren Beispielen immer der Fall), die alle harten Anforderungen erfüllt, dann liefert unser MIP eine solche. Außerdem optimiert unser MIP die gewichtete Summe von drei Kostenfunktionen, von denen jede eine der weichen Anforderungen modelliert. Unser MIP ist die erste Methode, die Oktilinearität garantiert, was für die Übersichtlichkeit von U-Bahn-Linienplänen essentiell ist. Dadurch dass wir global optimieren, können wir nicht in lokale Minima geraten und vermeiden so die Probleme, mit denen die bisherigen Methoden zu kämpfen hatten. Wir haben unser MIP implementiert. Die experimentellen Resultate sind vielversprechend, siehe Abbildung 3.2.

Wir zeigen, wie unser MIP dahingehend erweitert werden kann, dass U-Bahn-Stationen beschriftet werden. Der „Trick“ besteht darin, Beschriftungen als spezielle kleine U-Bahn-

Linien zu modellieren. Außerdem machen wir Vorschläge, wie unser MIP beschleunigt werden kann. Dies erreichen wir dadurch, dass wir heuristisch die Anzahl der notwendigen linearen Beschränkungen vermindern, insbesondere für die aufwändige Modellierung der Planarität.

Unser MIP hat gegenüber anderen Verfahren den Vorteil, dass leicht zusätzliche Anforderungen (etwa ein vorgegebenes Papierformat) einbezogen werden können. Auch Benutzerinteraktion („zeichne diese Kante horizontal“) ist kein Problem. Schließlich eignet es sich auch für alle anderen Arten von technischen Zeichnungen, bei denen die Anzahl der erlaubten Kantenrichtungen beschränkt ist.

Interessanterweise sind die Layout-Prinzipien von U-Bahn-Linienplänen nicht nur in Anwendungen mit geometrischem Hintergrund eingesetzt worden. Sandvad et al. [SGSK01] und Nesbitt [Nes04] verwenden die so genannte *Metro-Map-Metapher*, um abstrakte Information zu visualisieren. Ein besonders hübsches Beispiel, das auf dieser Metapher basiert, ist eine Karte [O'R03], die die Produktlinien des Verlegers O'Reilly zum Thema Open Source zeigt, siehe Abbildung 3.3.

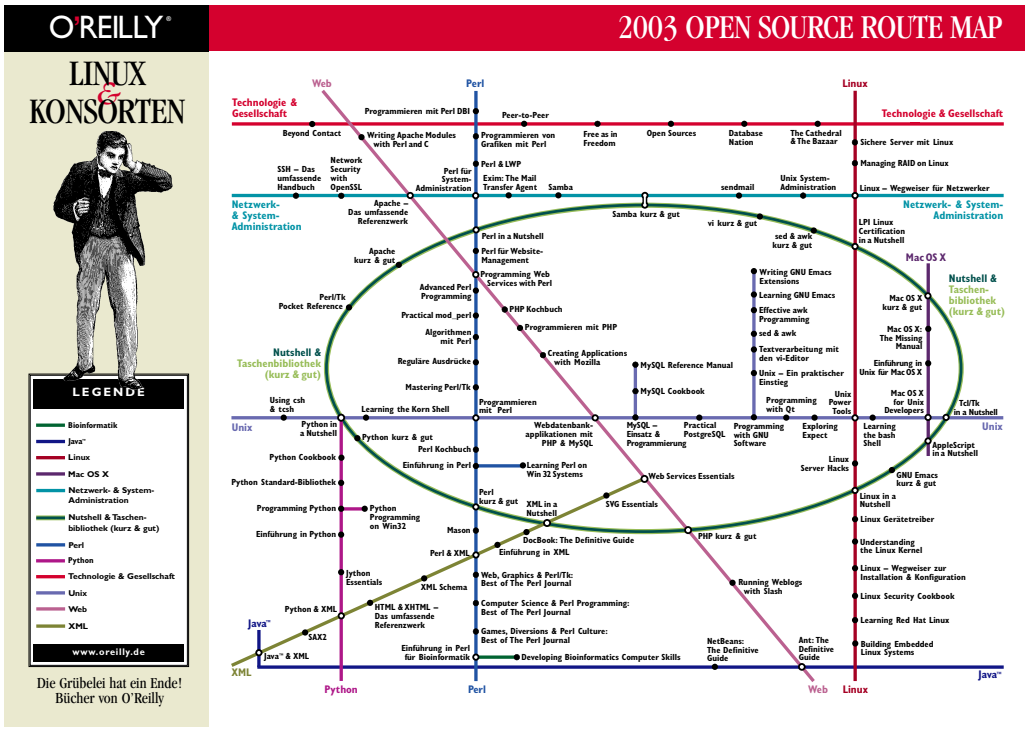


Abb. 3.3: Produktlinien des Verlegers O'Reilly [O'R03].

Über unsere Arbeit [NW06] ist ein zweiseitiger Artikel [Pol06] in der Frankfurter Allgemeinen Sonntagszeitung erschienen. Martin Nöllenburg wurde für die Arbeit ausgezeichnet mit dem *NRW Undergraduate Science Award 2005* in der Sparte Informatik. Außerdem bekam er für seinen Studienabschluss den *Absolventenpreis* der Fakultät für Informatik an der Universität Karlsruhe.

3.2 Beschriften am Kartenrand

In medizinischen Atlanten und manchen Arten von technischen Zeichnungen werden Punkte durch große Textblöcke beschriftet. Dies geschieht meistens dadurch, dass die Textblöcke um die eigentliche Abbildung herum gruppiert werden und dann mittels Pfeilen auf die Punkte verweisen, die sie beschriften. Es gibt schon eine Reihe von Arbeiten, die sich mit der Platzierung von Beschriftungen unter Zuhilfenahme von Pfeilen beschäftigt. So benützt Zoraster [Zor97] Simulated Annealing, um Punkte und Strecken in Karten für seismische Untersuchungen zu beschriften. Freeman et al. [FMC96] verwenden eine iterative rasterbasierte Methode, um Regionen in bodenkundlichen Karten zu beschriften. Schließlich erweitern Fekete und Plaisant [FP99] das *Infotipp-Paradigma*, um in dichten Punktwolken interaktiv zu beschriften, d.h. immer die Punkte, die gerade von einem kreisförmigen Cursor erfasst werden. Mit der Platzierung von Textblöcken festen Flächeninhalts (aber ohne Pfeile) beschäftigen sich Iturriaga und Lubiw [IL03]. Ihre so genannten *elastischen* Label müssen innerhalb eines gegebenen Rechtecks platziert werden und dabei Punkte auf dem Rand des Rechtecks beschriften. Sie geben einen Algorithmus an, der in $O(n^4)$ Zeit entscheidet, ob eine Platzierung existiert, und wenn ja, diese ausgibt.

Mit Michael Bekos, Michael Kaufmann und Antonios Symvonis [BKSW05, BKSW07] habe ich ein neues Modell für die Beschriftung von Punkten entworfen, in dem die Beschriftungen durch Pfeile mit den jeweiligen Punkten verbunden werden und in dem es uns dann möglich war, effiziente Algorithmen anzugeben.

Unser Modell ist das folgende. Wir gehen davon aus, dass die zu beschriftenden Punkte in einem Rechteck R enthalten sind und dass jeder Punkt durch ein achsenparalleles Rechteck gegebener Größe, seinen Label, beschriftet werden soll. Von den Labeln fordern wir, (a) dass sie einander nicht überlappen, (b) dass sie außerhalb von R liegen, aber R berühren und (c) dass sie durch einen Pfeil mit dem Punkt verbunden sind, den sie berühren. Weiterhin soll gelten, (d) dass Pfeile keine anderen Pfeile, Punkte oder Label schneiden. Wir betrachten zwei Möglichkeiten, Pfeile zu zeichnen: geradlinig oder rektilinear, d.h. als zusammenhängende Folge von achsenparallelen Strecken. Im rektilinearen Fall unterscheiden wir zwei Modi, in denen jeder Pfeil entweder aus maximal zwei oder aus maximal drei Segmenten bestehen darf. Entsprechend sagen wir, dass im geradlinigen Fall jeder Pfeil aus genau einem (natürlich nicht notwendigerweise achsenparallelen) Segment bestehen muss.

Uns interessieren zulässige Pfeil-Label-Platzierungen, aber auch solche, die entweder die Gesamtpfeillänge oder (im rektilinearen Fall) die Knickanzahl minimieren. Dies sind beim Graphenzeichnen übliche Zielfunktionen, die die Übersichtlichkeit einer Zeichnung quantifizieren. Wir gehen davon aus, dass die Zuordnung der Label zu den Rechtecksseiten gegeben ist. Der Grund für diese Annahme ist, dass man das NP-schwere Problem Partition [GJ79] auf dieses Zuordnungsproblem reduzieren kann.

Wir stellen eine Reihe effizienter Algorithmen vor, die für verschiedene Anzahlen von Pfeilsegmenten zulässige, längen- und knickminimale Pfeil-Label-Platzierungen berechnen, wobei die Label auf einer, zwei oder vier Seiten des Rechtecks platziert werden, siehe Tabelle 3.1. Bei der Längenminimierung unterscheiden wir noch zwischen fest vorgegebenen und frei wählbaren Punkten, an denen die Pfeile an die Beschriftungen angeknüpft werden müssen. Die Optimierungsalgorithmen benützen entweder dynamisches Programmieren oder geometrisches bipartites Matching, um eine längenminimale Pfeilplatzierung zu berechnen. Der Trick beim geometrischen Matching ist, dass ein kostenminimales Matching kreuzungsfrei ist. Einige der Algorithmen haben wir implementiert. Abbildung 3.4 zeigt die Ausgabe eines dieser Algorithmen.

Pfeil-segmente	Rechtecksseiten mit Beschriftung	zulässige Lösung	knickminimale Lösung	längenminimale Lsg.	
				feste Anknüpfungspunkte	freie Anknüpfungspunkte
1	1	$n \log n$	—	$n^{2+\varepsilon}$	n^3
1	4	$n \log n$	—	$n^{2+\varepsilon}$	n^3
2	1	n^2	offen	n^2	n^2
2	2	n^2	offen	n^2	n^2
3	1	$[n \log n]$	$[n^2]$	$n \log n$	$[n^2]$
3	2	n^2	offen	n^2	n^2
3	4	$n \log n$	offen	$n^2 \log^3 n$	n^3

Tab. 3.1: Laufzeiten unserer Algorithmen (in Groß-Oh-Notation) für verschiedene Varianten von Randbeschriftung. Dabei ist ε eine beliebig kleine positive Konstante. Die Laufzeiten in eckigen Klammern beziehen sich auf Beschriftungen verschiedener Größe. Die mit „—“ markierten Kombinationen sind nicht sinnvoll.

3.3 Beschriftung gewichteter Punkte

Da häufig nicht genug Platz vorhanden ist, um alle gegebenen Objekte mit Labeln der gewünschten Größe zu beschriften, besteht eine natürliche Zielfunktion darin, die Anzahl der beschrifteten Objekte zu maximieren. In vielen Situationen, z.B. in der Kartografie, sind die Objekte jedoch unterschiedlich wichtig. Um diesen Sachverhalt zu modellieren, kann man das Problem verallgemeinern, indem man die Objekte gewichtet und entsprechend die Summe der Gewichte der beschrifteten Objekte zur Zielfunktion macht. Dies lässt sich in allgemeine Optimierungsverfahren wie Simulated Annealing oder in genetische Algorithmen relativ problemlos einbauen.

Mit Sheung-Hung Poon, Chan-Su Shin, Tycho Strijk und Takeaki Uno [PSSW01, PSS+03] habe ich nach *beweisbar* guten Algorithmen gesucht, die gewichtete Punkte durch achsenparallele Rechtecke gleicher Höhe beschriften. Wir haben zuerst eine Nomenklatur von Beschriftungsmodellen erweitert, die van Kreveld et al. vorgeschlagen haben [vKSW99]. Wie van Kreveld et al. vergleichen wir unsere Modelle untereinander. Im gewichteten Fall bedeutet das, jeweils für Paare von Modellen das Verhältnis der Gewichte optimaler Beschriftungen zu untersuchen. Da wir auch Label unterschiedlicher Länge zulassen, tauchen dabei zum ersten Mal logarithmische Verhältnisse auf.

Wichtig bei der Nomenklatur ist insbesondere die Unterscheidung zwischen *Fest-Positionen-Modellen*, bei denen nur eine konstante Anzahl von Label-Positionen pro Punkt zulässig ist, und *Schiebemodellen*, bei denen ein Label mit einer oder mehrerer seiner vier Seiten an dem Punkt entlang geschoben werden kann. Der Begriff Schiebemodell (engl. *slider modell*) wurde von van Kreveld et al. [vKSW99] geprägt und hat viele Veröffentlichungen nach sich gezogen [ZQ02, GIM+01, KSY01, EKW05, ZH06, SvK02]. Man sieht relativ leicht, dass sich die bekannten Algorithmen [AvKS98] für Fest-Positionen-Modelle vom ungewichteten auf den gewichteten Fall übertragen lassen [Itu99].

Schwieriger wird es bei den Schiebemodellen. Zuerst konnten wir zeigen, dass im Gegensatz zum ungewichteten Fall erstaunlicherweise schon die eindimensionale Version des Problems NP-schwer ist, indem wir das NP-schwere Problem SubsetSum [GJ79] darauf reduziert haben. Also stellt sich die Frage nach der Existenz von Approximati-

Abb. 3.4: Ein Skelett mit Originalbeschriftung (rechts) und einer Beschriftung, die unser Algorithmus für rektilineare Pfeile mit maximal drei Segmenten berechnet hat (links).

onsalgorithmen, die wir als erste positiv beantworten konnten. Geholfen hat uns dabei, dass wir den engen Zusammenhang zwischen unserem eindimensionalen Beschriftungsproblem und einem Problem der Ablaufplanung gefunden haben, und zwar dem Problem *Durchsatzmaximierung mit beschränktem Streckfaktor auf einer Maschine*. Zu diesem Problem gab es schon einen Approximationsalgorithmus [BD00], ε-2PA, der für unseren Spezialfall ein voll-polynomielles Approximationsschema (FPTAS) mit Laufzeit und Speicherplatzbedarf $O(n^2/\varepsilon)$ liefert. Für diesen Spezialfall konnten wir jedoch den Korrektheitsbeweis des Algorithmus vereinfachen und seinen Speicherplatzbedarf auf $O(n/\varepsilon)$ reduzieren. Das zweidimensionale Problem kann man nun dadurch lösen, dass man es in viele eindimensionale Teilprobleme zerlegt, diese mit obigem Algorithmus löst und geeignet zusammensetzt. So erhält man einen Faktor-$(2+\varepsilon)$-Approximationsalgorithmus mit den gleichen Zeit- und Platzschranken wie im eindimensionalen Fall.

Wir untersuchen auch einige Spezialfälle. Besonders interessant sind dabei quadratische Label. Das eindimensionale Problem kann man lösen, indem man die schiebbaren Beschriftungen von n Punkten in höchstens $n' \leq \binom{n}{2}$ diskrete Beschriftungen, also Intervalle, verwandelt. Nun muss man nur noch eine *unabhängige Menge maximalen Ge-*

wichts in einem Intervallschnittgraphen finden, was mit dynamischem Programmieren in $O(n' \log n')$ Zeit möglich ist [HTC92]. Wenn man hinter die Diskretisierung ein dynamisches Programm [AvKS98] schaltet, erhält man für den zweidimensionalen Fall ein Polynomialzeitapproximationsschema (PTAS).

Folglich ist eine gewichtsmaximale Beschriftung von Punkten in allen Fällen *beliebig gut* approximierbar, in denen ein „Freiheitsgrad“ eingeschränkt ist: im eindimensionalen Fall (obiges FPTAS), im Fall von Fest-Positionen-Modellen [AvKS98], im Fall von konstantem Gewicht [vKSW99] und im Fall von konstanter Labellänge, was äquivalent zum oben diskutierten Fall der quadratischen Label ist. Offen bleibt die brennende Frage nach der Komplexität des allgemeinen Problems. Seither ist nur ein geringer Fortschritt erzielt worden: mit Thomas Erlebach, Torben Hagerup, Klaus Jansen und meinem Studienarbeiter Moritz Minzlaff konnten wir eine Faktor-$(2/3 - \varepsilon)$-Approximation angeben [EHJ$^+$06].

Alle unsere Algorithmen für die Beschriftung gewichteter Punkte lassen sich auch für die Beschriftung von Knoten in Graphzeichnungen oder Punkten in anderen Diagrammen einsetzen, indem man vorab für die „Schiebeintervalle“ jedes Labels prüft, welcher Teil davon durch andere Elemente des (feststehenden) Diagramms beansprucht wird. Nachdem man die Schiebeintervalle entsprechend gestutzt hat, kann man dann jeden unserer Algorithmen benutzen, um die Label innerhalb ihrer verkleinerten Schiebeintervalle zu platzieren. Die Kanten von Graphzeichnungen können beschriftet werden, indem man auf jeder Kante einen Punkt festlegt, der beschriftet werden soll. Im Fall von orthogonal gezeichneten Graphen kann ein Label durch diese Hilfskonstruktion wie gewünscht parallel zur Kantenrichtung verschoben werden, allerdings im Allgemeinen nicht über die ganze Kantenlänge. Um das zu erreichen, könnte man die allgemeine Version von ε-2PA verwenden. Deren Approximationsfaktor nimmt zwar mit wachsendem Quotienten von Schiebeintervall- und Labellänge ab, für praktische Belange sollte das aber nicht sehr ins Gewicht fallen.

3.4 Beschriftung von Punkten mit je zwei Beschriftungen

In Wetterkarten kommt es oft vor, dass Städte außer mit ihrem Namen mit weiteren Daten, etwa der dort gemessenen Temperatur, beschriftet werden. Das Problem der Mehrfachbeschriftung wurde zuerst von Kakoulis und Tollis [KT98] betrachtet, die zwei Heuristiken für die Beschriftung von Knoten und Kanten von Graphzeichnungen angeben und dabei auch mehrere Label für ein Objekt zulassen. Ihre Zielfunktion ist die Maximierung der (vollständig) beschrifteten Objekte. Einer ihrer Algorithmen ist iterativ, der andere benutzt *maximales bipartites Matching*. Die Autoren geben weder Laufzeitschranken noch Approximationsgarantien an. Soweit nicht anders vermerkt, haben alle anderen Beschriftungsalgorithmen in diesem Abschnitt eine Laufzeit von $O(n \log n)$ und benötigen linearen Speicher.

Die ersten Approximationsalgorithmen wurden von Zhu und Poon [ZP99, ZP01] angegeben. Sie beschriften eine gegebene Menge von Punkten mit Labeln gleicher Größe, zwei pro Punkt. Alle Label müssen ihre Punkte berühren und dürfen sich gegenseitig nicht schneiden. Dabei soll die Labelgröße maximiert werden. Für achsenparallele quadratische Labelpaare erzielen sie eine 1/4-Approximation, für kreisförmige Labelpaare eine 1/2-Approximation. Beide Algorithmen beruhen darauf, dass man disjunkte Regionen angeben kann, eine pro Punkt (bzw. Punktepaar), so dass die Punkte innerhalb ihrer

Region mit Labelpaaren hinreichender Größe beschriftet werden können. Im Falle der kreisförmigen Labelpaare sind diese Regionen einfach auf die Eingabepunkte zentrierte Kreisscheiben mit Durchmesser D, wobei D der kleinste euklidische Abstand zwischen zwei Eingabepunkten ist. Offensichtlich kann man in diese Kreisscheiben beliebig orientierte Doppelkreise mit Durchmesser $D/2$ platzieren. Andererseits sieht man leicht, dass D eine obere Schranke für den maximalen Labelradius ist. Der Approximationsfaktor für quadratische Beschriftungen wurde später von Zhu und Qin [ZQ02] auf 1/3 verbessert.

In Zusammenarbeit mit Zhongping Qin, Yinfeng Xu und Binhai Zhu [QWXZ00] konnte ich diese Ergebnisse weiter verbessern. Die Approximationsfaktoren, die wir erzielt haben, sind 1/2 für das Beschriften von Punkten mit Quadratpaaren und $1/(1+\cos 18°) \approx 0.513$ für den Fall von Kreispaaren.

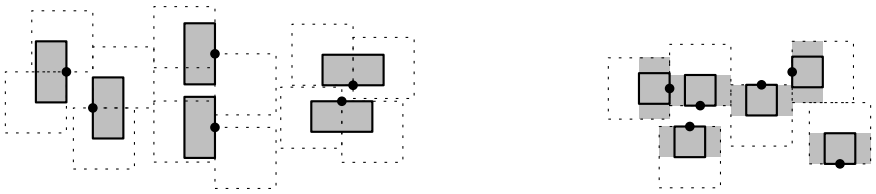
Kreisförmige Label. Während die Verbesserung im Fall der kreisförmigen Label gering zu sein scheint, gilt es folgende Schwierigkeit zu bedenken. Der Schnittgraph der oben angesprochenen (offenen) Kreisscheiben, die auf die Eingabepunkten zentriert sind und die Labelpaaren enthalten, ändert sich schlagartig, wenn man ihre Durchmesser von D aus leicht erhöht: der Maximalgrad steigt von 0 auf 6. Entsprechend schwierig und technisch wurde der Beweis des Approximationsfaktors. Der Algorithmus basiert auf den zueinander dualen geometrischen Datenstrukturen *Delaunay-Triangulierung* und *Voronoi-Diagramm* [AK00]. Die Kanten der Delaunay-Triangulierung benützen wir, um zu entscheiden, wie wir unsere Labelpaare platzieren; die paarweise disjunkten Regionen des Voronoi-Diagramms, um zu zeigen, dass sich keine zwei Labelpaare schneiden.

Unser Ergebnis wurde bald von Spriggs und Keil [SK02] verbessert. Ihr Algorithmus hat einen Approximationsfaktor von $4/(1+\sqrt{33}) \approx 0.593$. Auch ihr Algorithmus basiert darauf, disjunkte Regionen zu berechnen und Labelpaare darin zu platzieren.

Schließlich konnten Michael Thon, einer meiner Studenten, Yinfeng Xu und ich auch dieses Resultat verbessern [WTX02]. Unser Algorithmus hat einen Approximationsfaktor von 2/3. Dazu haben wir die Idee mit dem Voronoi-Diagramm konsequent zu Ende gedacht. Es lag auf der Hand, dass man obigen Algorithmus [QWXZ00] heuristisch verbessern kann, indem man Labelpaare maximaler Größe in jede Voronoi-Region platziert[2]. Dabei entstehen natürlich verschieden große Labelpaare, also muss man anschließend alle auf die Größe d des kleinsten Labelpaares schrumpfen. Die geschrumpften Labelpaare können sich nicht schneiden, da die Voronoi-Regionen konvex sind. Die Schwierigkeit bestand also hauptsächlich in der Analyse des Approximationsfaktors, also dem Vergleich der Labelgröße d mit der maximalen Labelgröße d_{opt}. Dabei ergibt sich d als die Nullstelle der Ableitung einer gebrochen rationalen Funktion, die den Abstand eines Labelmittelpunktes zu einer parametrisierten Geraden ausdrückt. Es stellte sich heraus, dass unsere Abschätzung scharf ist, das heißt, dass es tatsächlich Punktmengen gibt, für die die Gleichheit in $d \geq 2d_{\text{opt}}/3$ gilt.

Vor kurzem wurde auch dieses Ergebnis noch leicht verbessert – wenn auch ein Stück weit zu Lasten der Laufzeit. Jiang et al. [JBQZ04] skizzieren einen Algorithmus mit Approximationsfaktor $(0.671-\varepsilon)$ und Laufzeit $O(n\log(n/\varepsilon))$. Außerdem zeigen sie, dass das Problem NP-schwer ist und sich nicht besser als mit einem Faktor von zirka 0.966 approximieren lässt. Zum Vergleich: auch das Beschriften von Punkten mit je einem Kreis ist NP-schwer und lässt sich nicht beliebig gut approximieren [SW01]. Den augenblicklich besten Approximationsfaktor von $(0.336-\varepsilon)$ erzielt ein Algorithmus von Jiang et al. [JBQZ04], der n Punkte in $O(n\log n + n(1/\varepsilon)^{O(1/\varepsilon^2)})$ Zeit beschriftet.

[2]Diesen Algorithmus haben wir implementiert und als Java-Applet im Internet zur Verfügung gestellt, siehe *http://i11www.iti.uni-karlsruhe.de/map-labeling/points/two-circles/*



(a) Quadratpaar-Beschriftung **(b)** Metafont-Beschriftung

Abb. 3.5: Quadratpaar- und Metafont-Beschriftungen (gepunktet) können so auf Rechtecksbeschriftungen (grau schattiert) abgebildet werden, dass die Rechtecke dabei Quadratpaare bzw. Metafont-Quadrate der halben ursprünglichen Seitenlänge (dick gezeichnet) enthalten.

Quadratische Label. Im Gegensatz dazu ist unser Faktor-1/2-Approximationsalgorithmus [QWXZ00] für die Beschriftung von Punkten mit achsenparallelen Quadratpaaren nicht weiter verbessert worden. Bei diesem Problem gibt es wie bei der Beschriftung mit Kreispaaren für jeden Punkt überabzählbar viele erlaubte Labelpositionen. Anders als bei Kreispaaren ist jedoch die Position eines Labels nicht durch die des anderen festgelegt. Unser Algorithmus löst eine diskrete Version des Problems. Bei dieser Version gibt es für jeden Punkt nur vier zulässige Labelpositionen, nämlich die, wo das Quadratpaar ein Rechteck mit Seitenverhältnis 2:1 bildet, das den Punkt im Mittelpunkt einer der beiden längeren Seiten berührt, siehe die Rechtecke in Abbildung 3.5a. Natürlich ist jede Lösung des diskreten Problems auch eine Lösung des ursprünglichen. Wir haben gezeigt, dass man das diskrete Problem *optimal* lösen kann und dass man wie in Abbildung 3.5a jede optimale Lösung des ursprünglichen Problems auf eine des diskreten Problems abbilden kann, bei der die Label halb so groß sind. Mit anderen Worten: unser optimaler Algorithmus für das diskrete Problem hat einen Approximationsfaktor von 1/2 für das ursprüngliche Problem.

Die Tatsache, dass sich das Punktbeschriftungsproblem für die oben beschriebenen Rechtecke mit Seitenverhältnis 2:1 exakt lösen lässt, ist insofern erstaunlich, als zu diesem Zeitpunkt nur ein exakt lösbares ebenes Beschriftungsproblem bekannt war. Wenn für jedes zu beschriftende Objekt nur zwei Labelpositionen erlaubt sind, kann man das Entscheidungsproblem („Gibt es für die gegebene Labelgröße eine Beschriftung aller Punkte?“) mit einer booleschen Variablen pro Punkt als 2-SAT-Formel kodieren und in Linearzeit auf Erfüllbarkeit testen. Der „Trick“ bei obigem Rechtecksbeschriftungsproblem bestand darin, pro Punkt *zwei* boolesche Variable zu benützen. Interessanterweise klappt dieser Trick beim klassischen Vier-Positionen-Modell (4P) nicht, bei dem jeder rechteckige Label den zu beschriftenden Punkt mit einer *Ecke* berühren muss. Formann und Wagner [FW91] haben gezeigt, dass 4P selbst für Quadrate NP-schwer ist und sich nicht besser als bis auf einen Faktor von 1/2 approximieren lässt. Andererseits konnten sie einen Algorithmus angeben, der gerade diesen Approximationsfaktor erreicht.

Wenn man das Modell leicht abändert und für quadratische Label, einer pro Punkt, fordert, dass sie die zu beschriftenden Punkte mit dem Mittelpunkt einer Seite berühren, erhält man das *Metafont-Beschriftungsproblem*, siehe die kleinen und großen quadratischen Label in Abbildung 3.5b. Metafont ist ein Werkzeug, das Knuth zum Schriftdesign entwickelt hat und das dabei gewisse Punkte in Diagrammen beschriften muss. Knuth und Raghunathan [KR92] haben bewiesen, dass diese Variante ebenfalls NP-schwer ist. Ihr Beweis zeigt implizit, dass sich die maximale Labelgröße nicht beliebig gut approxi-

mieren lässt. Formann und Wagner [FW93] konnten einen Algorithmus für das Metafont-Beschriftungsproblem mit Approximationsfaktor 1/3 angeben. Interessanterweise liefert unser optimaler Rechtecksbeschriftungsalgorithmus auch für dieses Problem einen Approximationsfaktor von 1/2. Der Grund ist der gleiche wie beim Beschriften mit Quadratpaaren: jede optimale Lösung des Metafont-Beschriftungsproblems lässt sich auf eine Lösung des Rechtecksproblems abbilden, bei der die Rechtecke halb so groß wie die Metafont-Quadrate sind, siehe Abbildung 3.5b. Bei den Quadratpaaren ist der Beweis komplizierter und technischer, da die Rechtecke nicht immer in den entsprechenden Quadratpaaren enthalten sind, siehe Abbildung 3.5a.

Das Gebiet der Beschriftung von Punkten mit mehreren Quadraten wurde später durch eine Arbeit von Duncan et al. [DQVZ03] abgerundet, in der gezeigt wurde, dass man n Punkte mit je *drei* Quadraten maximaler Größe in $O(n^2 \log n)$ Zeit beschriften kann. Das Beschriften mit vier Quadraten ist trivial, denn es genügt, das Punktepaar mit kleinstem Abstand d in der L_∞-Norm zu finden. Dann ist eine Beschriftung aller Punkte mit Quadraten der Seitenlänge $d/2$ möglich und optimal.

Folglich zeigt sich beim Beschriften von Punkten mit quadratischen Labeln derselbe Trend, der für kreisförmige Label gilt: je mehr Label pro Punkt, desto besser lässt sich das Problem approximieren oder desto schneller lässt es sich optimal lösen.

Literaturverzeichnis

[AAP02] Aichholzer, Oswin, Franz Aurenhammer und Belén Palop del Río: *Quickest paths, straight skeletons, and the city Voronoi diagram.* In: *Proc. 18th Annu. ACM Symp. Comput. Geom. (SoCG'02)*, Seiten 151–159, 2002.

[ACC+96] Arikati, Srinivasa Rao, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid und Christos D. Zaroliagis: *Planar spanners and approximate shortest path queries among obstacles in the plane.* In: *Proc. European Symposium on Algorithms (ESA'96)*, Band 1136 der Reihe *Lecture Notes Comput. Sci.*, Seiten 514–528. Springer-Verlag, 1996.

[AdBC+05] Asano, Tetsuo, Mark de Berg, Otfried Cheong, Hazel Everett, Herman Haverkort, Naoki Katoh und Alexander Wolff: *Optimal spanners for axis-aligned rectangles.* Comput. Geom. Theory Appl., 30(1):59–77, 2005.

[ADM+95] Arya, Sunil, Gautam Das, David M. Mount, Jeffrey S. Salowe und Michiel Smid: *Euclidean spanners: Short, thin, and lanky.* In: *Proc. 27th Annu. ACM Sympos. Theory Comput. (STOC'95)*, Seiten 489–498, 1995.

[AK00] Aurenhammer, Franz und Rolf Klein: *Voronoi diagrams.* In: J.-R. Sack and J. Urrutia (Herausgeber): *Handbook of Computational Geometry*, Kapitel 5, Seiten 201–290. Elsevier, 2000.

[AM00] Avelar, Silvania und Matthias Müller: *Generating topologically correct schematic maps.* In: *Proc. 9th Int. Symp. on Spatial Data Handling (SDH'00)*, Seiten 4a.28–4a.35, 2000.

[AMS99] Arya, Sunil, David M. Mount und Michiel Smid: *Dynamic algorithms for geometric spanners of small diameter: Randomized solutions.* Comput. Geom. Theory Appl., 13:91–107, 1999.

[ASW98] Agarwal, Pankaj K., Micha Sharir und Emo Welzl: *The discrete 2-center problem.* Discrete Comput. Geom., 20(3):287–305, 1998.

[AvKS98] Agarwal, Pankaj K., Marc van Kreveld und Subhash Suri: *Label placement by maximum independent set in rectangles.* Comput. Geom. Theory Appl., 11:209–218, 1998.

[Awe85] Awerbuch, Baruch: *Complexity of network synchronization.* J. ACM, 32(4):804–823, 1985.

[BD86] Bandelt, Hans-Jürgen und Andreas W. M. Dress: *Reconstructing the shape of a tree from observed dissimilarity data.* Adv. Appl. Math., 7:309–343, 1986.

[BD00] Berman, Piotr und Bhaskar DasGupta: *Multi-phase algorithms for throughput maximization for real-time scheduling.* J. Combin. Optim., 4(3):307–323, 2000.

[BDLN05] Binucci, Carla, Walter Didimo, Giuseppe Liotta und Maddalena Nonato: *Orthogonal drawings of graphs with vertex and edge labels.* Comput. Geom. Theory Appl., 32(2):71–114, 2005.

[BGHW06] Benkert, Marc, Joachim Gudmundsson, Herman Haverkort und Alexander Wolff: *Constructing interference-minimal networks.* In: J. Wiedermann, J. Stuller, G. Tel, J. Pokorný, and M. Bieliková (Herausgeber): *Proc. 32nd Int. Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM'06)*, Band 3831 der Reihe *Lecture Notes Comput. Sci.*, Seiten 166–176. Springer-Verlag, 2006.

[BKSW05] Bekos, Michael A., Michael Kaufmann, Antonios Symvonis und Alexander Wolff: *Boundary labeling: Models and efficient algorithms for rectangular maps.* In: J. Pach (Herausgeber): *Proc. 12th Int. Symposium on Graph Drawing (GD'04)*, Band 3383 der Reihe *Lecture Notes Comput. Sci.*, Seiten 49–59. Springer-Verlag, 2005.

[BKSW07] Bekos, Michael A., Michael Kaufmann, Antonios Symvonis und Alexander Wolff: *Boundary labeling: Models and efficient algorithms for rectangular maps.* Comput. Geom. Theory Appl., 36(3):215–236, 2007.

[BLR00] Barkowsky, Thomas, Longin Jan Latecki und Kai-Florian Richter: *Schematizing maps: Simplification of geographic shape by discrete curve evolution.* In: C. Freksa, W. Brauer, C. Habel, and K. F. Wender (Herausgeber): *Proc. Spatial Cognition II—Integrating abstract theories, empirical studies, formal models, and practical applications*, Band 1849 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 41–53, 2000.

[BT04] Bodlaender, Hans L. und Gerard Tel: *A note on rectilinearity and angular resolution.* J. Graph Algorithms Appl., 8(1):89–94, 2004.

[BvRWZ04] Burkhart, Martin, Pascal von Rickenbach, Roger Wattenhofer und Aaron Zollinger: *Does topology control reduce interference?* In: *Proc. 5th ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc'04)*, Seiten 9–19, 2004.

[BWW05] Benkert, Marc, Florian Widmann und Alexander Wolff: *The minimum Manhattan network problem: A fast factor-3 approximation.* In: J. Akiyama, M. Kano, and X. Tan (Herausgeber): *Proc. 8th Japanese Conf. on Discrete and Computational Geometry (JCDCG'04)*, Band 3742 der Reihe *Lecture Notes Comput. Sci.*, Seiten 16–28. Springer-Verlag, 2005.

[BWWS06] Benkert, Marc, Alexander Wolff, Florian Widmann und Takeshi Shirabe: *The minimum Manhattan network problem: Approximations and exact solutions.* Comput. Geom. Theory Appl., 35(3):188–208, 2006.

[CdBvK05] Cabello, Sergio, Mark de Berg und Marc van Kreveld: *Schematization of networks.* Comput. Geom. Theory Appl., 30(3):223–238, 2005.

[CDS01] Chen, Danny Z., Gautam Das und Michiel Smid: *Lower bounds for computing geometric spanners and approximate shortest paths.* Discrete Appl. Math., 110:151–167, 2001.

[Cha03] Chan, Timothy M.: *Semi-online maintenance of geometric optima and measures.* SIAM J. Computing, 32(3):700–716, 2003.

[CK95] Callahan, Paul B. und S. Rao Kosaraju: *A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields.* J. ACM, 42(1):67–90, Januar 1995.

[CNV05] Chepoi, Victor, Karim Nouioua und Yann Vaxés: *A rounding algorithm for approximating minimum Manhattan networks.* In: C. Chekuri, K. Jansen, J. D. P. Rolim, and L. Trevisan (Herausgeber): *Proc. 8th Intern. Workshop Approx. Algorithms for Combinatorial Optimization Problems (APPROX'05)*, Band 3624 der Reihe *Lecture Notes Comput. Sci.*, Seiten 40–51. Springer-Verlag, 2005.

[CvK02] Cabello, Sergio und Marc van Kreveld: *Schematic networks: an algorithm and its implementation.* In: D. Richardson and P. van Oosterom (Herausgeber): *Proc. 10th Int. Symp. Spatial Data Handling (SDH'02)*, Advances in Spatial Data Handling, Seiten 475–486, 2002.

[dBvKOS00] Berg, Mark de, Marc van Kreveld, Mark Overmars und Otfried Schwarzkopf: *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, zweite Auflage, 2000.

[DETT99] Di Battista, Guiseppe, Peter Eades, Roberto Tamassia und Ioannis G. Tollis: *Graph Drawing.* Prentice Hall, Upper Saddle River, NJ, 1999.

[DQVZ03] Duncan, Rob, Jianbo Qian, Antoine Vigneron und Binhai Zhu: *Polynomial time algorithms for three-label point labeling.* Theoret. Comput. Sci., 296(1):75–87, 2003.

[ECMS97] Edmondson, Shawn, Jon Christensen, Joe Marks und Stuart Shieber: *A general cartographic labeling algorithm.* Cartographica, 33(4):13–23, 1997.

[EE99] Eppstein, David und Jeffrey G. Erickson: *Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions.* Discrete Comput. Geom., 22(4):569–592, 1999.

[EHJ+06] Erlebach, Thomas, Torben Hagerup, Klaus Jansen, Moritz Minzlaff und Alexander Wolff: *A new approximation algorithm for labeling weighted points with sliding labels.* In: *Proc. 22nd European Workshop on Computational Geometry (EWCG'06)*, Seiten 137–140, Delphi, 2006.

[EKW05] Ebner, Dietmar, Gunnar W. Klau und René Weiskircher: *Label number maximization in the slider model.* In: J. Pach (Herausgeber): *Proc. 12th Internat. Symp. on Graph Drawing (GD'04)*, Band 3383 der Reihe *Lecture Notes Comput. Sci.*, Seiten 144–154. Springer-Verlag, 2005.

[Epp00] Eppstein, David: *Spanning trees and spanners.* In: J.-R. Sack and J. Urrutia (Herausgeber): *Handbook of Computational Geometry*, Seiten 425–461. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[FMC96] Freeman, Herbert, Sean Marrinan und Hitesh Chitalia: *Automated labeling of soil survey maps.* In: *Proc. ASPRS-ACSM Annual Convention, Baltimore*, Band 1, Seiten 51–59, 1996.

[FP99] Fekete, Jean-Daniel und Catharine Plaisant: *Excentric labeling: Dynamic neighborhood labeling for data visualization.* In: *Proc. Conference on Human Factors in Computer Systems (CHI'99)*, Seiten 512–519. ACM New York, 1999.

[FW91] Formann, Michael und Frank Wagner: *A packing problem with applications to lettering of maps.* In: *Proc. 7th Annuual ACM Symposium on Computational Geometry (SoCG'91)*, Seiten 281–288, 1991.

[FW93] Formann, Michael und Frank Wagner: *An efficient solution to Knuth's METAFONT labeling problem.* Manuskript siehe `http://i11www.ira.uka.de/map-labeling/papers/fw-eskml-93.ps.gz`, 1993. Fachbereich Informatik, Freie Universität Berlin.

[Gar94] Garland, Ken: *Mr Beck's Underground Map.* Capital Transport Publishing, 1994.

[GHP+02] Gudmundsson, Joachim, Herman Haverkort, Sang-Min Park, Chan-Su Shin und Alexander Wolff: *Facility location and the geometric minimum-diameter spanning tree.* In: K. Jansen, S. Leonardi, and V. Vazirani (Herausgeber): *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'02)*, Band 2462 der Reihe *Lecture Notes Comput. Sci.*, Seiten 146–160. Springer-Verlag, 2002.

[GHP+04] Gudmundsson, Joachim, Herman Haverkort, Sang-Min Park, Chan-Su Shin und Alexander Wolff: *Facility location and the geometric minimum-diameter spanning tree.* Comput. Geom. Theory Appl., 27(1):87–106, 2004.

[GHvK02] Gudmundsson, Joachim, Mikael Hammar und Marc van Kreveld: *Higher order delaunay triangulations.* Comput. Geom. Theory Appl., 23(1):85–98, 2002.

[GIM+01] Garrido, Mari Ángeles, Claudia Iturriaga, Alberto Márquez, José Ramon Portillo, Pedro Reyes und Alexander Wolff: *Labeling subway lines.* In: P. Eades and T. Takaoka (Herausgeber): *Proc. 12th Annu. Int. Symp. Algorithms and Computation (ISAAC'01)*, Band 2223 der Reihe *Lecture Notes Comput. Sci.*, Seiten 649–659. Springer-Verlag, 2001.

[GJ79] Garey, Michael R. und David S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[GLN01] Gudmundsson, Joachim, Christos Levcopoulos und Giri Narasimhan: *Approximating a minimum Manhattan network*. Nordic J. Comput., 8:219–232, 2001.

[GLN02] Gudmundsson, Joachim, Christos Levcopoulos und Giri Narasimhan: *Improved greedy algorithms for constructing sparse geometric spanners*. SIAM J. Comput., 31(5):1479–1500, 2002.

[GLNS02] Gudmundsson, Joachim, Christos Levcopoulos, Giri Narasimhan und Michiel Smid: *Approximate distance oracles for geometric graphs*. In: *Proc. 13th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA'02)*, Seiten 828–837, 2002.

[GSW07] Görke, Robert, Chan-Su Shin und Alexander Wolff: *Constructing the city Voronoi diagram faster*. Internat. J. Comput. Geom. Appl., 2007. Erscheint in Bälde.

[GW05] Görke, Robert und Alexander Wolff: *Constructing the city Voronoi diagram faster*. In: *Proc. 2nd Int. Symp. on Voronoi Diagrams in Science and Engineering (VD'05)*, Seiten 162–172, Seoul, 2005.

[HLCW91] Ho, Jan-Ming, D. T. Lee, Chia-Hsiang Chang und C. K. Wong: *Minimum diameter spanning trees and related problems*. SIAM Journal on Computing, 20(5):987–997, Oktober 1991.

[HMdN05] Hong, Seok-Hee, Damian Merrick und Hugo A. D. do Nascimento: *The metro map layout problem*. In: J. Pach (Herausgeber): *Proc. 12th Int. Symp. on Graph Drawing (GD'04)*, Band 3383 der Reihe *Lecture Notes Comput. Sci.*, Seiten 482–491. Springer-Verlag, 2005.

[HTC92] Hsiao, Ju Yuan, Chuan Yi Tang und Ruay Shiung Chang: *An efficient algorithm for finding a maximum weight 2-independent set on interval graphs*. Inform. Process. Lett., 43(5):229–235, Oktober 1992.

[IL03] Iturriaga, Claudia und Anna Lubiw: *Elastic labels around the perimeter of a map*. J. Algorithms, 47(1):14–39, 2003.

[Itu99] Iturriaga, Claudia: *Map Labeling Problems*. Doktorarbeit, School of Computer Science, University of Waterloo, 1999.

[JBQZ04] Jiang, Minghui, Sergey Bereg, Zhongping Qin und Binhai Zhu: *New bounds on map labeling with circular labels*. In: R. Fleischer and G. Trippen (Herausgeber): *Proc. 15th Annual International Sympo sium on Algorithms and Computation (ISAAC'04)*, Band 3341 der Reihe *Lecture Notes Comput. Sci.*, Seiten 606–617. Springer-Verlag, 2004.

[JW93] Jansen, Klaus und Gerhard J. Woeginger: *The complexity of detecting crossingfree configurations in the plane*. BIT, 33:580–595, 1993.

[KIA02] Kato, Ryo, Keiko Imai und Takao Asano: *An improved algorithm for the minimum Manhattan network problem*. In: P. Bose and P. Morin (Herausgeber): *Proc. 13th Annual International Symposium on Algorithms and Computation (ISAAC'02)*, Band 2518 der Reihe *Lecture Notes Comput. Sci.*, Seiten 344–356. Springer-Verlag, 2002.

[KLN91] Kratochvíl, Jan, Anna Lubiw und Jaroslav Nešetřil: *Noncrossing subgraphs in topological layouts*. SIAM J. Disc. Math., 4(2):223–244, 1991.

[KM03] Klau, Gunnar W. und Petra Mutzel: *Automatic layout and labelling of state diagrams*. In: W. Jäger and H.-J. Krebs (Herausgeber): *Mathematics—Key Technology for the Future*, Seiten 584–608. Springer-Verlag, Berlin, 2003.

[KR92] Knuth, Donald E. und Arvind Raghunathan: *The problem of compatible representatives*. SIAM J. Discr. Math., 5(3):422–427, 1992.

[KSSW05] Knauer, Christian, Étienne Schramm, Andreas Spillner und Alexander Wolff: *Configurations with few crossings in topological graphs*. In: X. Deng and D.-Z. Du (Herausgeber): *Proc. 16th Annu. Int. Symp. Algorithms Comput. (ISAAC'05)*, Band 3827 der Reihe *Lecture Notes Comput. Sci.*, Seiten 604–613. Springer-Verlag, 2005.

[KSSW07] Knauer, Christian, Étienne Schramm, Andreas Spillner und Alexander Wolff: *Configurations with few crossings in topological graphs*. Comput. Geom. Theory Appl., 37(2). Erscheint Juli 2007.

[KSY01] Kim, Sung Kwon, Chan-Su Shin und Tae-Cheon Yang: *Labeling a rectilinear map with sliding labels*. Internat. J. Comput. Geom. Appl., 11(2):167–179, 2001.

[KT98] Kakoulis, Konstantinos G. und Ioannis G. Tollis: *On the multiple label placement problem*. In: *Proc. 10th Canadian Conf. Computational Geometry (CCCG'98)*, Seiten 66–67, Montréal, 1998.

[KW01] Kaufmann, Michael und Dorothea Wagner (Herausgeber): *Drawing Graphs: Methods and Models*, Band 2025 der Reihe *Lecture Notes Comput. Sci.* Springer-Verlag, 2001.

[LAP03] Lam, Fumei, Marina Alexandersson und Lior Pachter: *Picking alignments from (Steiner) trees*. Journal of Computational Biology, 10:509–520, 2003.

[Meg83] Megiddo, N.: *Applying parallel computation algorithms in the design of serial algorithms*. J. ACM, 30(4):852–865, 1983.

[Mit00] Mitchell, Joseph S. B.: *Geometric shortest paths and network optimization*. In: J.-R. Sack and J. Urrutia (Herausgeber): *Handbook of Computational Geometry*, Seiten 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[ML05] Moaveni-Nejad, Kousha und Xiang-Yang Li: *Low-interference topology control for wireless ad hoc networks*. Internat. J. Ad Hoc & Sensor Wireless Networks, 1(1–2):41–64, 2005.

[MLW95] Müller, Jean-Claude, Jean-Philippe Lagrange und Robert Weibel (Herausgeber): *GIS and Generalization—Methodology and Practice*, Band I der Reihe *GISDATA*. Taylor & Francis, London, 1995.

[MM97] Mackaness, William und Gordon A. Mackechnie: *Detection and simplification of road junctions in automated map generalization.* In: *ACSM/ASPRS Annual Convention & Exposition, Technical Papers Volume 1: Surveying & Cartography*, Seiten 72–82, Bethesda, MD, 1997.

[Mor80] Morrison, Joel L.: *Computer technology and cartographic change.* In: D. Taylor (Herausgeber): *The Computer in Contemporary Cartography.* Johns Hopkins University Press, 1980.

[Nes04] Nesbitt, Keith V.: *Getting to more abstract places using the metro map metaphor.* In: *Proc. 8th International Conference on Information Visualisation (IV'04)*, Seiten 488–493. IEEE Computer Society, 2004.

[Nöl05] Nöllenburg, Martin: *Automated drawing of metro maps.* Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, August 2005.

[NS07] Narasimhan, Giri und Michiel Smid: *Geometric Spanner Networks.* Cambridge University Press, 2007.

[NW06] Nöllenburg, Martin und Alexander Wolff: *A mixed-integer program for drawing high-quality metro maps.* In: P. Healy and N. S. Nikolov (Herausgeber): *Proc. 13th Int. Symposium on Graph Drawing (GD'05)*, Band 3843 der Reihe *Lecture Notes Comput. Sci.*, Seiten 321–333. Springer-Verlag, 2006.

[OB05] Ostrovsky-Berman, Yaron: *Computing transportation Voronoi diagrams in optimal time.* In: *Proc. 21st European Workshop on Computational Geometry (EWCG'05)*, Seiten 159–162, Eindhoven, 2005.

[O'R03] O'Reilly: *Open source route map.* `http://www.oreilly.de/artikel/routemap.pdf`, 2003.

[Ove03] Ovenden, Mark: *Metro Maps of the World.* Capital Transport Publishing, 2003.

[Pol06] Polatschek, Klemens: *Die Schönheit des Untergrundes.* Frankfurter Allgemeine Sonntagszeitung 28, 16 Juli 2006.

[Pra99] Prakash, Ravi: *Unidirectional links prove costly in wireless ad-hoc networks.* In: *Proc. 3rd Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM'99)*, Seiten 15–22. ACM Press, 1999.

[Pri57] Prim, R. C.: *Shortest connection networks and some generalisations.* Bell Systems Technical Journal, Seiten 1389–1410, 1957.

[PSS+03] Poon, Sheung-Hung, Chan-Su Shin, Tycho Strijk, Takeaki Uno und Alexander Wolff: *Labeling points with weights.* Algorithmica, 38(2):341–362, 2003.

[PSSW01] Poon, Sheung-Hung, Chan-Su Shin, Tycho Strijk und Alexander Wolff: *Labeling points with weights.* In: P. Eades and T. Takaoka (Herausgeber): *Proc. 12th Annu. Int. Symp. Algorithms Comput. (ISAAC'01)*, Band 2223 der Reihe *Lecture Notes Comput. Sci.*, Seiten 610–622. Springer-Verlag, 2001.

[PU89] Peleg, David und Jeffrey D. Ullman: *An optimal synchronizer for the hypercube.* SIAM J. Comput., 18(4):740–747, August 1989.

[QWXZ00] Qin, Zhongping, Alexander Wolff, Yinfeng Xu und Binhai Zhu: *New algorithms for two-label point labeling.* In: M. Paterson (Herausgeber): *Proc. 8th Annu. Europ. Symp. on Algorithms (ESA'00)*, Band 1879 der Reihe *Lecture Notes Comput. Sci.*, Seiten 368–379. Springer-Verlag, 2000.

[Rai98] Raidl, Günther: *A genetic algorithm for labeling point features.* In: *Proc. Internat. Conf. Imaging Science, Systems, and Technology (CISST'98)*, Seiten 189–196, Las Vegas, NV, 1998.

[RS91] Ruppert, Jim und Raimund Seidel: *Approximating the d-dimensional complete Euclidean graph.* In: *Proc. 3rd Canad. Conf. Comput. Geom. (CCCG'91)*, Seiten 207–210, 1991.

[SGSK01] Sandvad, Elmer S., Kaj Grønbæk, Lennert Sloth und Jørgen Lindskov Knudsen: *A metro map metaphor for guided tours on the Web: the Webvise Guided Tour System.* In: *Proc. 10th Int. World Wide Web Conference (WWW'01)*, Seiten 326–333. ACM Press, 2001.

[SK02] Spriggs, Michael J. und J. Mark Keil: *A new bound for map labeling with uniform circle pairs.* Inform. Process. Lett., 81(1):47–53, 2002.

[SKB+04] Spriggs, Michael J., J. Mark Keil, Sergei Bespamyatnikh, Michael Segal und Jack Snoeyink: *Computing a* $(1+\epsilon)$*-approximate geometric minimum-diameter spanning tree.* Algorithmica, 38(4):577–589, 2004.

[Smi00] Smid, Michiel: *Closest point problems in computational geometry.* In: J.-R. Sack and J. Urrutia (Herausgeber): *Handbook of Computational Geometry*, Seiten 877–935. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[SR04] Stott, Jonathan und Peter Rodgers: *Metro map layout using multicriteria optimization.* In: *Proc. 8th International Conference on Information Visualisation (IV'04)*, Seiten 355–362. IEEE Computer Society, 2004.

[SS05] Shi, Weiping und Chen Su: *The rectilinear Steiner arborescence problem is NP-complete.* SIAM J. Comput., 35(3):729–740, 2005.

[SvK02] Strijk, Tycho und Marc van Kreveld: *Practical extensions of point labeling in the slider model.* GeoInformatica, 6(2):181–197, 2002.

[SW01] Strijk, Tycho und Alexander Wolff: *Labeling points with circles.* Internat. J. Comput. Geom. Appl., 11(2):181–195, April 2001.

[Tam87] Tamassia, Roberto: *On embedding a graph in the grid with the minimum number of bends.* SIAM J. Comput., 16(3):421–444, 1987.

[vD01] Dijk, Steven van: *Genetic Algorithms for Map Labeling.* Doktorarbeit, Department of Computer Science, Utrecht University, November 2001.

[vKSW99] Kreveld, Marc van, Tycho Strijk und Alexander Wolff: *Point labeling with sliding labels.* Comput. Geom. Theory Appl., 13(1):21–47, 1999.

[Wol99] Wolff, Alexander: *Automated Label Placement in Theory and Practice.* Doktorarbeit, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Mai 1999.

[Wol00] Wolff, Alexander: *A simple proof for the NP-hardness of edge labeling.* Technischer Bericht 11/2000, Institut für Mathematik und Informatik, Universität Greifswald, September 2000.

[WS96] Wolff, Alexander und Tycho Strijk: *The Map-Labeling Bibliography.* `http://i11www.ira.uka.de/map-labeling/bibliography`, 1996.

[WTX02] Wolff, Alexander, Michael Thon und Yinfeng Xu: *A simple factor-2/3 approximation algorithm for two-circle point labeling.* Internat. J. Comput. Geom. Appl., 12(4):269–281, 2002.

[WWKS01] Wagner, Frank, Alexander Wolff, Vikas Kapoor und Tycho Strijk: *Three rules suffice for good label placement.* Algorithmica, 30(2):334–349, 2001.

[Yao82] Yao, Andrew Chi Chih: *On constructing minimum spanning trees in k-dimensional spaces and related problems.* SIAM J. Comput., 11(4):721–736, 1982.

[ZH06] Zhang, Qingnian und Lars Harrie: *Placing text and icon labels simultaneously: A real-time method.* Cartography and Geographic Information Science, 33(1):53–64, 2006.

[Zor97] Zoraster, Steven: *Practical results using simulated annealing for point feature label placement.* Cartography and GIS, 24(4):228–238, 1997.

[ZP99] Zhu, Binhai und Chung Keung Poon: *Efficient approximation algorithms for multi-label map labeling.* In: A. Aggarwal and C. P. Rangan (Herausgeber): *Proc. 10th Annual International Symposium on Algorithms and Computation (ISAAC'99)*, Band 1741 der Reihe *Lecture Notes Comput. Sci.*, Seiten 143–152. Springer-Verlag, 1999.

[ZP01] Zhu, Binhai und Chung Keung Poon: *Efficient approximation algorithms for two-label point labeling.* Internat. J. Comput. Geom. Appl., 11(4):455–464, August 2001.

[ZQ02] Zhu, Binhai und Zhongping Qin: *New approximation algorithms for map labeling with sliding labels.* Journal of Combinatorial Optimization, 6(1):99–110, 2002.

Lebenslauf

Ausbildung

16.01.2006	Abschluss der Habilitation mit einem Kolloquiumsvortrag; Thema: „Rekonstruktion von Sprachstammbäumen“
14.06.2005	Einreichung der Habilitationsschrift (kumulativ); Titel: „Geometrische Netzwerke und ihre Visualisierung“. Referenten: Prof. Dr. Dorothea Wagner und Prof. Dr. Emo Welzl.
28.05.1999	Disputation; Vortrag über „Parametrisierte Komplexität – eine neue Herangehensweise für schwere Probleme“.
10.02.1999	Einreichung der Promotionsschrift; Titel: „Automated Label Placement in Theory and Praxis“. Referenten: Dr. F. Wagner, Dr. M. van Kreveld und Prof. Ch. Jones.
19.12.1995	Diplom an der Freien Universität Berlin; Titel der Diplomarbeit: „Map Labeling“. Betreuer: Dr. Frank Wagner
02.06.1987	Abitur am Mörike-Gymnasium Ludwigsburg

Beruflicher Werdegang

04.2003 –	Nachwuchsgruppenleiter (BAT Ia) des Projekts „Geometrische Netzwerke und ihre Visualisierung“ gefördert durch die DFG im Rahmen des Aktionsplans Informatik, Universität Karlsruhe
10.2002 – 02.2003	Vertretung der Professur für praktische Informatik (C4) an der Universität Konstanz
06.1999 – 09.2002	Wissenschaftlicher Assistent (C1) von Prof. Peter Schreiber, Universität Greifswald
06.1996 – 05.1999	Wissenschaftlicher Mitarbeiter im Rahmen des DFG-Projekts „Effiziente Algorithmen zur Beschriftung von Landkarten“ unter der Leitung von Dr. Frank Wagner an der Freien Universität Berlin.
07.1987 – 02.1989	Zivildienst