Beschriftungsalgorithmen in Theorie & Praxis

Blockseminar in Trassenheide, 16./17. Juni 2001

Katharina Bach, Kristina Hanig, Tim Hoffmann, Wolfgang Kresse, Julia Löcherbach, Paul Rosenthal, Steffen Rudnick, Peter Schreiber, Michael Thon, Alexander Wolff

Ernst-Moritz-Arndt-Universität Greifswald
Institut für Mathematik und Informatik

Preprint-Reihe Mathematik 13/2002

Zweite, leicht geänderte Auflage
Juni 2002

Inhaltsverzeichnis

Einleitung

Die Idee zu einem Seminar über Beschriftungsalgorithmen entstand nach meinen ersten Semestern als Assistent in Greifswald, als ich mich an den hiesigen Lehrbetrieb gewöhnt hatte. Ich war vorher im Rahmen eines Forschungsprojekts mit dem Titel „Effiziente Algorithmen zur Beschriftung von Landkarten“ an der FU Berlin beschäftigt und hatte dort deshalb nur wenig mit der Lehre zu tun. Allerdings hatte ich kurz vor meinem Wechsel nach Greifswald geholfen, an der FU ein Seminar zum Thema „Algorithmen für die Linguistik“ zu veranstalten. Das Seminar hatte einen fächerübergreifenden Charakter und wurde als Blockseminar an einem Wochenende vor den Toren der Stadt durchgeführt. Beides, sowohl die Interdisziplinarität als auch die Veranstaltungsform, kam bei den Studenten gut an und führte zu deutlich mehr Diskussionen als ich es von rein mathematischen Seminaren in der normalen Uniumgebung kannte.

Das waren die Gründe, warum ich auch in Greifswald das Experiment Blockseminar wagen wollte – das erste seiner Art hier am Institut für Mathematik und Informatik, soweit ich weiß. Das Thema zwischen Mathematik, Informatik und Geografie war durch meine Projektarbeit in Berlin vorgezeichnet. Bei der Visualisierung von Information, zum Beispiel auf Landkarten (Abbildung 1), in Grafiken (Abbildung 2) oder Elektrophorese-Gels (Abbildung 3), spielt die Beschriftung von Gestaltungselementen wie Punkten, Kreisen oder Flächen eine große Rolle. Besonders durch die Verbreitung des Internets ist in den letzten Jahren auch die Menge an zu visualisierender Information stark angewachsen – und damit der Bedarf an guten und schnellen Beschriftungsalgorithmen.

Abbildung 1: Ausschnitt aus einer Internetkarte der Firma MapQuest.

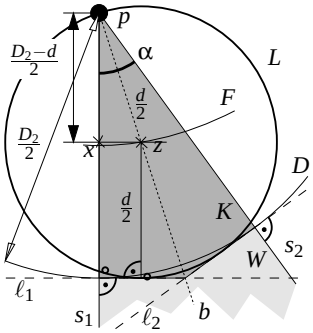


Abbildung 2: Illustration aus einer mathematischen Veröffentlichung [QWXZ00].

Abbildung 3: Ausschnitt aus einem Elektrophorese-Gel. Quelle: Internet.

Schon ein Jahr bevor es überhaupt stattfand, bekam der interdisziplinäre Charakter des Seminars jedoch einen kleinen Dämpfer. Am Lehrstuhl für Geografische Informationssysteme des benachbarten Instituts für Geografie fand sich niemand, der Interesse an einer gemeinsamen Ausrichtung des Seminars hatte. Deshalb war ich sehr froh, dass ich Prof. Wolfgang Kresse

vom Fachbereich Bauingenieur- und Vermessungswesen der Fachhochschule Neubrandenburg als Vortragenden gewinnen konnte und mir auch mein Chef Prof. Peter Schreiber seine Unterstützung zusagte.

Obwohl ich den Annahmeschluss fürs Vorlesungsverzeichnis des Sommersemesters 2001 mit meiner Seminarankündigung verpasst hatte, haben sich für Greifswalder Verhältnisse erstaunlich viele Studenten von den entprechenden Aushängen zum Mitmachen locken lassen. Zu meiner Freude waren neben Diplommathematikern und Lehrämtlern auch Geografen und eine Biomathematikerin sowie ein ehemaliger Student unseres Instituts dabei, der in der Zwischenzeit in Göttingen studiert. Die Themen waren schnell vergeben und entsprachen jeweils der mathematischen Vorbelastung.

Die Themenkreise, die wir behandeln wollten, waren die folgenden.

1. Was ist (gute) Landkartenbeschriftung und wo außerhalb der Kartografie spielen Beschriftungsprobleme noch eine Rolle?
2. Welche allgemeinen Optimierungsverfahren wurden schon zum Beschriften eingesetzt und wie funktionieren sie?
3. Welche Spezialfälle der Beschriftung von punktförmigen Objekten (etwa Städte auf der Deutschlandkarte oder Proteine auf einem Elektrophorese-Gel) kann man trotz der Komplexität des allgemeinen Beschriftungsproblems effizient oder wenigstens approximativ lösen?

Wegen der Erfahrungen, die ich in Berlin gesammelt hatte, habe ich den Seminarteilnehmern einen Terminplan vorgelegt, der verhindern sollte, dass die Vorträge erst in der Nacht vor dem Seminarwochenende auf Folie gebannt werden. Teil dieses Plans war es, die Teilnehmer dazu zu bringen, vier Wochen vor dem Seminar Kurzzusammenfassungen ihrer Vorträge abzugeben und schon zwei Wochen vor dem Seminar die Folien oder wenigstens eine Vortragskonzeption vorzulegen. Die Zusammenfassungen wurden dann den anderen Teilnehmern über eine Seminar-Webseite[1] zugänglich gemacht, so dass jeder sich zumindest theoretisch auch mit den Themen der anderen vertraut machen konnte. In der Praxis wurden die Termine allerdings nur von einer kleinen Minderheit der Teilnehmer eingehalten. In solchen Situationen habe ich mir doppelt so viele Teilnehmer gewünscht, so dass ich mich mit der Drohung, jemanden vom Seminar auszuschliessen, nicht lächerlich gemacht hätte... Andererseits gab es auch einen Teilnehmer, der „seinen“ Algorithmus kurzerhand implementiert hat, um ihn beim Seminar vorzuführen. Andere wiederum haben sich so gut in ihr Material eingearbeitet, dass sie kurz vor dem Seminar noch einen technischen Fehler in der Veröffentlichung bemerkten, die sie vorstellen sollten.

Das Gros der studentischen Vorträge war dann auf gutem bis sehr gutem Niveau – trotz des wenig befolgten Terminplans (aber wie hätte es ohne ihn ausgesehen?). Die Qualität der Vorträge war unabhängig von den eingesetzten Techniken, die von handgeschriebenen Folien bis zur Powerpoint-Präsentation reichten. In den Diskussionen, die auf jeden Vortrag folgten, ging es darum, sich noch einmal die Knackpunkte des eben gehörten zu vergegenwärtigen.

Unser „Tagungsort“ war das Feriencamp Trassenheide auf Usedom, in dem wir sehr einfach, aber auch äusserst preisgünstig von Samstag auf Sonntag übernachten konnten. Tageslichtprojektor und Beamer mussten wir selbst mitbringen. Dafür konnten wir am Samstagabend einen langen Strandspaziergang mit Sonnenuntergang geniessen. Den Rest des Abends hat ein Teilnehmer sehr schön auf der Gitarre begleitet, bis einer nach dem anderen den Anstrengungen des Tages erlag.

Die meisten Teilnehmer liessen sich anstecken von der Idee, aus den Vorträgen dieses ersten Blockseminars des Instituts für Mathematik und Informatik ein Preprint zu machen. Ich glaube, dass sich viele Studenten freuen würden, wenn diese Veranstaltungsform auch bei anderen

[1] `http://www.math-inf.uni-greifswald.de/~awolff/lehre/label_ss01/`

Dozenten auf Interesse stieße. Die Ausarbeitungen haben den Charakter kleiner Diplomarbeiten bekommen und waren sehr gute Übungen im Umgang mit mathematischer Textverarbeitung (sprich Latex) und dazu passenden Zeichenprogrammen (xfig und ipe). Manche Teilnehmer haben ihre Ausarbeitung zum Anlass genommen, sich mit dem Betriebssystem Linux und seinen Möglichkeiten vertraut zu machen. Es war für alle Beteiligten viel Arbeit, die Kapitel dieses Preprints immer wieder durchzulesen und zu verbessern, aber ich glaube, es hat sich gelohnt.

In Kapitel 1 gibt Peter Schreiber einen Einblick in das Beschriftungsproblem jenseits der Kartografie. In den Kapiteln 2 und 3 werden allgemeine Optimierungsverfahren am Beispiel der Landkartenbeschriftung vorgestellt: Michael Thon gibt eine Einführung in ganzzahliges lineares Programmieren, Tim Hoffmann in Simulated Annealing [CMS95] und genetische Algorithmen [vD01]. Besonders die beiden heuristischen Verfahren Simulated Annealing und genetische Algorithmen sind in der Praxis sehr beliebt, da sie sich leicht implementieren lassen und sich aufgrund ihrer Allgemeinheit an verschiedene problemspezifische Zielfunktionen anpassen lassen. Allerdings sind sie meist deutlich langsamer als spezialisierte Verfahren, die die Geometrie des Problems ausnutzen [vKSW99] oder direkt den Konfliktgraphen der Beschriftungskandidaten vereinfachen [WWKS01]. In Kapitel 4 zeigt Wolfgang Kresse mit einem Auszug aus seiner Dissertation [Kre94], wie man sehr allgemein kartografische Beschriftung modellieren kann, so dass die verschiedenen Bedeutungen von Städten, Flüssen und Ländern zum Tragen kommen.

In den darauf folgenden vier Kapiteln werden theoretisch interessante Spezialfälle des allgemeinen Beschriftungsproblems behandelt. In Kapitel 5 untersucht Katharina Bach, wie man mithilfe von exakten Algorithmen für eindimensionale Beschriftungsprobleme [HTC92] approximative Lösungen für zweidimensionale Probleme erhält – falls alle Beschriftungen gleiche Höhe haben. In Kapitel 6 beschreibt Julia Löcherbach einen verallgemeinerten Sweepline-Algorithmus [vKSW99], der das gleiche zweidimensionale Problem auf ganz andere Weise, aber mit demselben Approximationsfaktor (nämlich $1/2$) löst. Sie hat diesen Algorithmus schon für eine Greifswalder Bioinformatik-Firma implementiert. In Kapitel 7 erklärt Kristina Hanig, was ein recht neues Approximationsschema [EJS01], das annähernd maximale unabhängige Mengen in Kreisschnittgraphen findet, mit der Beschriftungsproblematik zu tun hat. Schließlich untersuchen Steffen Rudnick und Paul Rosenthal in Kapitel 8 eines der wenigen effizient exakt lösbaren Spezialfälle des allgemeinen Beschriftungsproblems: die Beschriftung von Punkten mit möglichst großen Quadrattripeln [DQZ01]. Als Anregung zum Weitermachen, zum eigenständigen Forschen gibt's in Kapitel 9 eine Liste neuer und offener Fragen zu Beschriftungsproblemen, die ich letzten Sommer zu einem anderen Anlaß (und daher auf Englisch) zusammengetragen habe. Wer sich einen Überblick über das Gebiet verschaffen möchte, dem sei ein „Besuch“ der Map-Labeling-Bibliography [WS96] empfohlen.

Viel Spaß beim Lesen!

Greifwald, im April 2002.

Alexander Wolff

Kapitel 1

Das Beschriftungsproblem jenseits der Kartografie

Peter Schreiber

Text ohne Bilder vermittelt Information in vielen Zusammenhängen auf eine sehr schwerfällige, schwer verständliche, schwer zu behaltende Weise. Bild ohne Zusatzwissen über das Dargestellte (oft in Form von Text) ist fast immer vieldeutig. Die richtige Kombination von Text und Bild, seit den Anfängen der menschlichen Kultur mit mehr oder weniger Geschick praktiziert, ist im Zeitalter der elektronischen Kommunikationsmedien vor größere Herausforderungen gestellt als je zuvor. Dabei sind praktisch zwei Fälle zu unterscheiden:

A) Wo und wie plaziert man am günstigsten Bilder in einen Text, damit der Leser ein Bild und den darauf bezüglichen Text zugleich im Blick haben kann? In alten mathematischen Werken wurde das Problem manchmal tatsächlich dadurch gelöst, dass ein bestimmtes Bild, auf das sich ein mehrere Seiten langer Text oder verschiedene, voneinander entfernte Textstellen bezogen, auf mehreren Seiten wiederholt wurde (so z.B. in der vielmals nachgedruckten kommentierten Euklidausgabe (1574ff) des Christoph Clavius). Umgekehrt kennt jeder Leser alter Bücher die durch die damaligen Drucktechniken begründete Praxis, Abbildungen in ausklappbaren Tafeln am Ende des Buches unterzubringen. Der zu A) entgegegengesetzte Fall

B) ist der im Bild verteilte Text, wie ihn jedermann von Landkarten und Stadtplänen, aber auch von technischen Zeichnungen und naturwissenschaftlichen oder medizinischen Abbildungen kennt, siehe Abbildung 1.1. Im wesentlichen erst im 20. Jahrhundert setzte sich für technische Zeichnungen die Praxis durch, nur Zahlen oder Buchstaben im Bild zu plazieren und diese durch eine Legende zu erklären, siehe Abbildung 1.2.

Ist Text an vielen jeweils eindeutig zugehörigen Teilen einer bereits vorliegenden Karte oder sonstigen bildlichen Darstellung zu plazieren, so entstehen die typischen Probleme, denen dieses Seminar gewidmet ist. Dabei spielt es aber gar keine Rolle, dass *Text* in Bildern zu plazieren ist. Es kommt auch vor, dass zu bestimmten Punkten eines Bildes zusätzliche Bilder gehören (siehe Abbildung 1.3), dass zum Beispiel allen Städten einer Landkarte eine Ansicht eines typischen Bauwerkes der betreffenden Stadt angeheftet werden soll oder dass zu vielen Textstellen Anmerkungen oder Kommentare gehören, die als solche in den Text einzubauen sind. Im Zeitalter des Computers spielt nun die Größe dieser einzusetzenden Bilder oder Texte

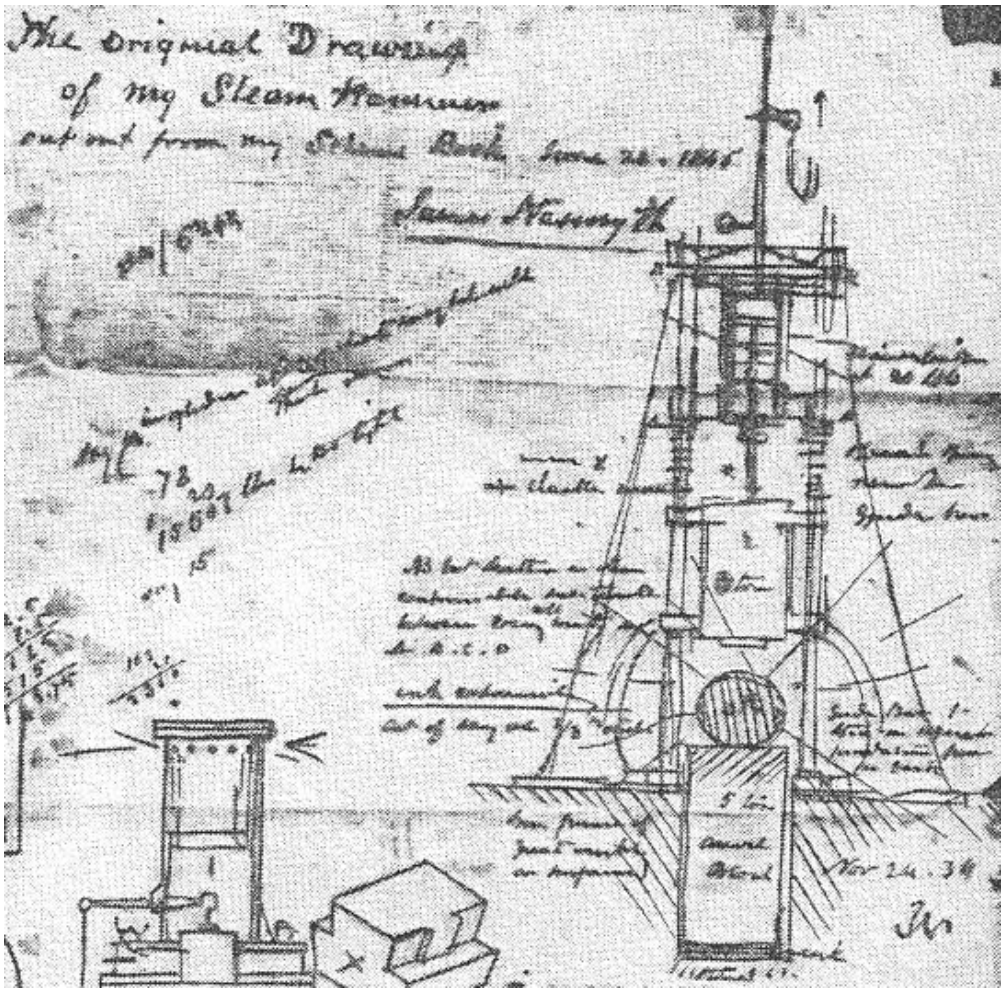


Abbildung 1.1: Dampfhammer, Zeichnung des Erfinders James Nasmyth 1839 zum Zweck der Patentanmeldung mit viel erklärendem Text. Quelle: [SRW+81].

oft keine Rolle mehr: Es genügt, ein Feld zum Anklicken zu plazieren, wodurch das Einzusetzende dann aufgerufen wird, und man kann diesen Prozess des Aufrufens sogar iterieren.

Beschriftungsprobleme ganz anderer Art entstehen, wenn Bilderzeugung und Bildbeschriftung so miteinander verzahnt sind, dass sie sich gegenseitig beeinflussen. Wir betrachten dazu im folgenden den Fall der Entstehung einer geometrischen Konstruktion nach einem Programm (Konstruktionsalgorithmus) und den Fall des Zeichnens von Graphen, deren Knoten nicht mehr als punktförmig angenommen werden können bzw. deren Kanten zusätzliche Bedingungen zu erfüllen haben, da sie mehr oder weniger umfangreiche Beschriftungen tragen müssen.

Zu den Elementarschritten von numerischen Algorithmen gehören bekanntlich Ergibtanweisungen, d.h. Befehle der allgemeinen Form

$$z := F(x,y,\ldots)$$

mit der Bedeutung: Führe an den Objekten (Zahlen), die zur Zeit unter den Adressen $x,y,\ldots$ gespeichert sind, die Operation F aus und speichere des Resultat unter der Adresse z (d.h. in der Speicherzelle mit diesem Namen), wobei der vorhergehende Inhalt dieser Speicherzelle automatisch gelöscht bzw. überschrieben wird. Ein Elementarschritt einer geometrischen Konstruktionsanweisung hat syntaktisch im wesentlichen die gleiche Gestalt. Nur die zugehörige

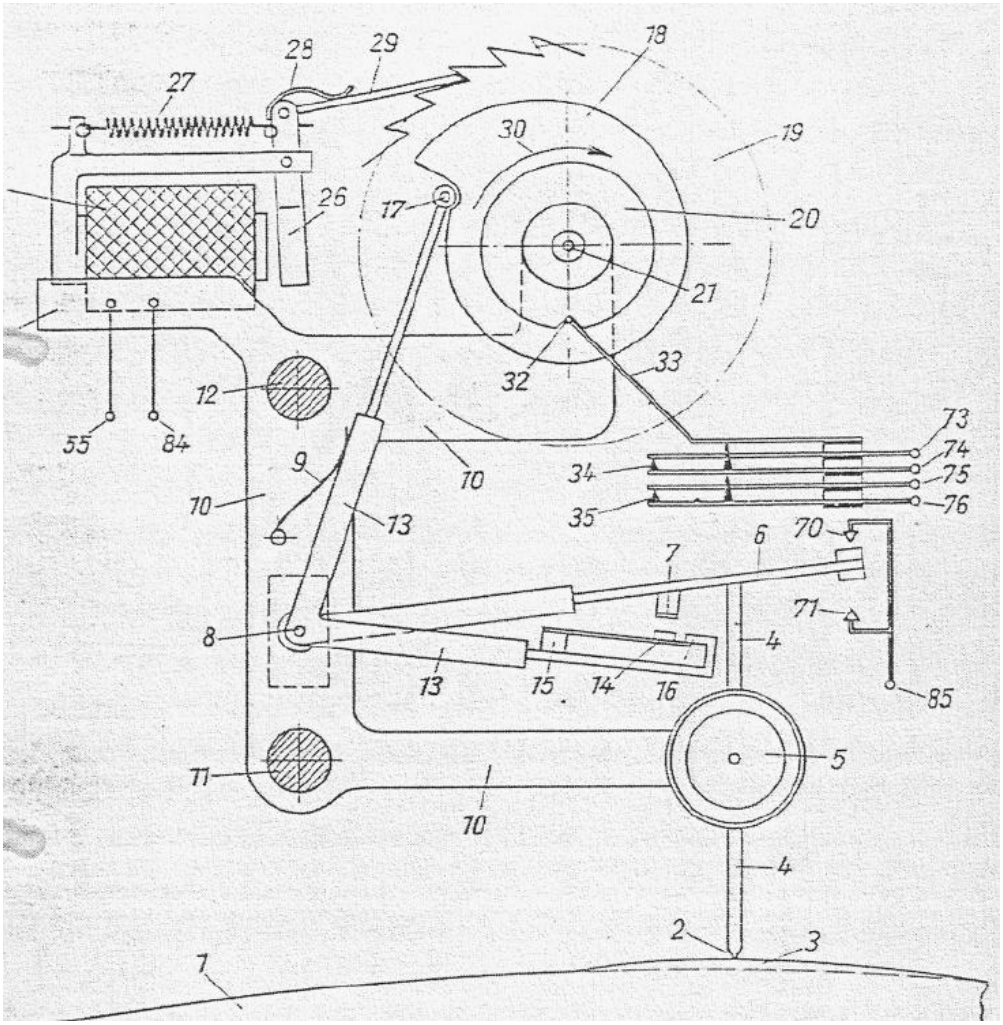


Abbildung 1.2: Zeichnung eines Analog-Digital-Wandlers um 1944 von Konrad Zuse, ebenfalls zur Patentanmeldung. Quelle: [Zus01].

Semantik ist etwas anders: Die Zeichenfläche hat hier die Funktion des Speichers, in dem gewisse Objekte durch Einzeichnen gespeichert und durch Heranschreiben von Namen adressiert sind, und zugleich die Funktion des *Rechenwerkes*, in dem an den gespeicherten Objekten bestimmte Operationen ausgeführt werden. Zum Beispiel bedeutet $g := L(A,D)$ folgendes: Suche im Speicher (in der Zeichenfläche) diejenigen Punkte auf, die zur Zeit die Adressen A bzw. D tragen, führe an ihnen die Operation L des Verbindens aus und adressiere die entstandene Gerade mit g. Der wesentlichste Unterschied zur numerischen Ergibtanweisung ist, dass durch die Neuvergabe der Adresse g diese nicht automatisch von einer anderen Geraden verschwindet, welche womöglich vor der Ausführung des besprochenen Schrittes diese Adresse schon trug. Im Falle geometrischer Konstruktionen ist die Semantik der Ergibtanweisungen also durch den Befehl zu ergänzen, die neuvergebene Adresse an den Stellen, an denen sie vorher stand, zu löschen, damit jede Adresse zu jedem Zeitpunkt eine eindeutig bestimmte Bedeutung hat. (Es genügt aus dieser Sicht, die Adresse statt des adressierten Objektes zu tilgen, da der Zugriff auf Objekte nur durch ihre Adressen erfolgen kann, ein Punkt oder eine Gerade ohne Adresse also ähnlich wie eine Person ohne Ausweis in keinem „offiziellen“ Vorgang eine Rolle spielen kann.) Diese zusätzliche Anweisung ist nicht immer leicht realisierbar. Sie soll ja – wie überhaupt ein Algorithmus – im Prinzip von einem geeigneten technischen Apparat, ohne In-

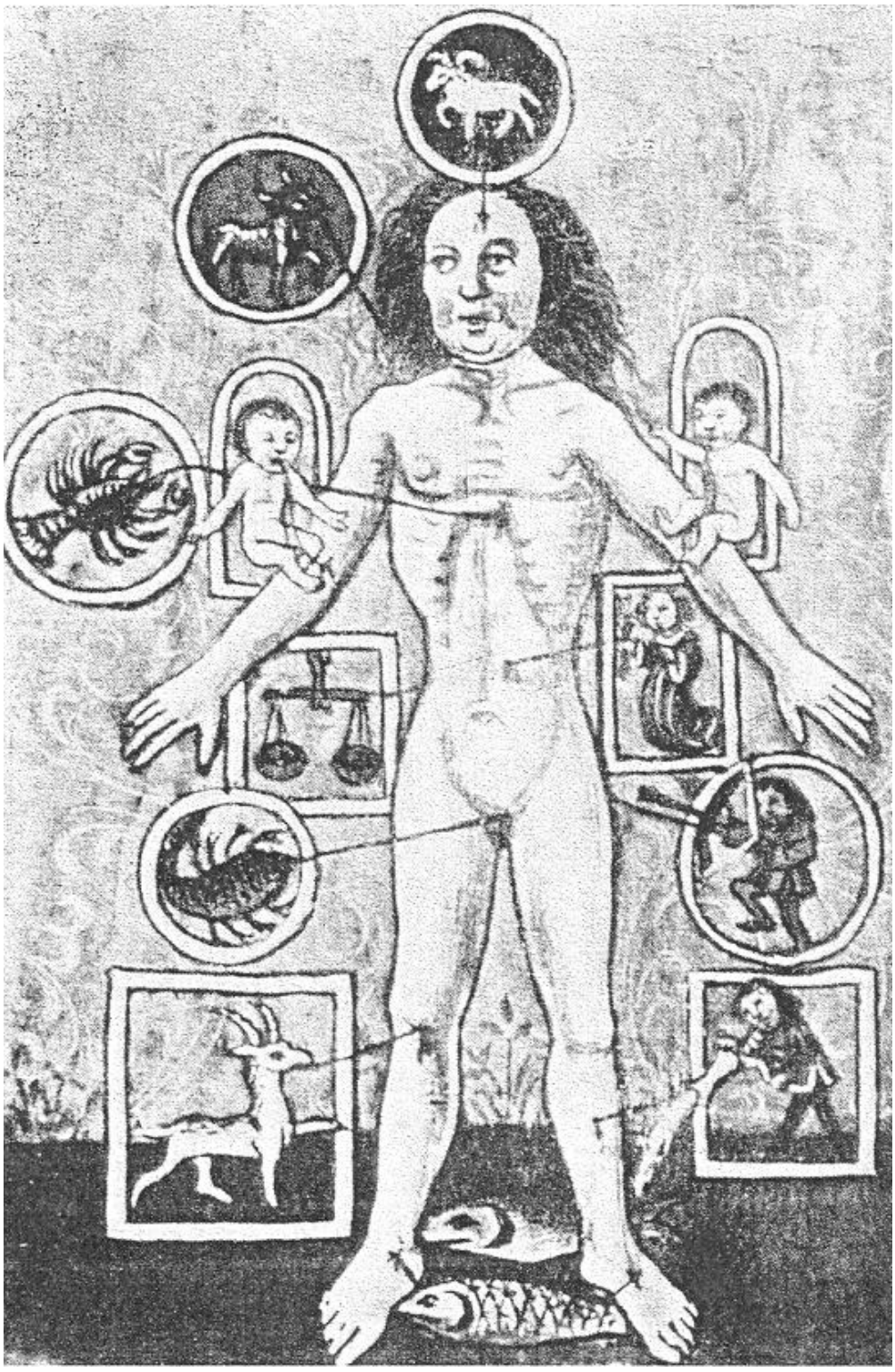


Abbildung 1.3: Im 15. Jahrhundert angenommene Beziehungen der Tierkreiszeichen zu verschiedenen Organen und ihren Erkrankungen. Quelle: [FM37].

telligenz, ausführbar sein. Hinter der Löschanweisung verbirgt sich also ein Suchprozess, die bereits vergebene Adresse in der Zeichenfläche wiederzufinden. Ist ein Algorithmus zyklenfrei, so kann man ohne Neuvergabe bereits vergebener Adressen auskommen. Zum Wesen zyklischer Algorithmen gehört es aber unvermeidlich, dass bestimmte Adressen eine unvorhersehbare Anzahl von Neuzuweisungen von Bedeutungen bekommen. Man wird insbesondere die in numerischen Algorithmen unproblematischen selbstbezüglichen Anweisungen der Form $z := F(z,x,\ldots)$ mittels einer zusätzlichen Adresse w in $w := F(z,x,\ldots)$, $z := w$ verwandeln, da einerseits z erst neu vergeben werden darf, wenn die alte Bedeutung von z getilgt ist, andererseits die alte Bedeutung von z für die Konstruktion der neuen Bedeutung von z noch gebraucht wird. Beim numerischen Arbeiten tritt dies nicht auf, weil die Neuberechnung von z im Rechenwerk erfolgt, der alte Wert von z aber im Speicher steht. (Beim Konstruieren von Hand werden diese und viele andere Schwierigkeiten durch unbewusstes intelligentes Handeln verschleiert: Ich „merke mir“ z.B. für eine kurze Zeitspanne, welcher Punkt z schon da war und welchen ich neu konstruiert habe.) Wegen der Mehrdeutigkeit mancher typischer geometrischer Operationen (z.B. haben zwei sich schneidende Kreise im allgemeinen zwei gleichberechtigte Schnittpunkte), der Notwendigkeit, mitunter nichtdeterministische Schritte (in Gestalt der Wahl willkürlicher „Hilfs“-Punkte, der Auswahl eines von mehreren ununterscheidbaren Objekten oder der Notwendigkeit, eine Reihenfolge in einer Menge von gleichartigen und vertauschbaren Teilprozessen zu wählen) und anderer Spezifika ist der Entwurf problemorientierter Programmiersprachen für verschiedene Arten geometrischer Konstruktionen insgesamt keineswegs trivial. Für Details sei auf [Sch75, Sch84] verwiesen.

Eine weitere – freilich im Zeitalter des Computerzeichnens meist nur noch theoretisch interessante – Schwierigkeit besteht darin, dass bei gewissen zyklischen Konstruktionsprogrammmen nicht nur die Anzahl der konstruierten Punkte (oder anderen Objekte) in Abhängigkeit von der speziellen Eingabefigur unbeschränkt wachsen kann, sondern die zu adressierenden Punkte sich beliebig häufen können, so dass die Zuordnung zwischen den Punkten und ihren anzuschreibenden Adressen praktisch undurchführbar wird. Hierbei sind die zwei folgenden Fälle zu unterscheiden:

a) Die Syntax der Programmiersprache lässt nur einfache, d.h. nicht parameterabhängige Adressen zu. Dann kommt in jedem Programm nur eine feste Zahl von Adressen vor, die lediglich im Falle zyklischer Programme unbeschränkt oft neu vergeben (neu belegt) werden.

   Ein einfacher, aber charakteristischer Fall ist das Messen einer durch zwei Punkte A, B gegebenen Streckenlänge bezüglich einer durch zwei Punkte O, E gegebenen Maßeinheit. Nachdem durch fortlaufendes Abtragen der Einheitsstrecke auf dem Strahl von A nach B der Punkt B in ein Intervall zweier benachbarter Punkte C, D mit ganzzahligen Längen $n = |AC|$, $n+1 = |AD|$ eingeschlossen ist, sind diese unteren und oberen Näherungswerte bis zum Erreichen einer vorgegebenen Genauigkeit durch Halbierung (oder Zehnteilung) der jeweils vorliegenden Strecke CD zu verbessern. Das eigentliche Resultat der Konstruktion, die angenäherte Maßzahl der Strecke AB, wird bei einer solchen Konstruktion gewonnen, indem unter zwei Adressen x, y der Sorte „rationale Zahl“ der Fortgang des Messprozesses, d.h. die jeweils erreichte untere (Länge AC) und obere (Länge AD) Schranke der Maßzahl gespeichert wird. Im Falle der Streckenhalbierung kommt dann das Programm mit den Adressen A, B, C, D und M (Mittelpunkt der Strecke CD) sowie x und y aus, wobei aber die Adressen C, D und M an immer neue Punkte zu vergeben sind, die sich um B häufen.

   Abhilfe kann hier durch eine modifizierte Semantik des Konstruierens geschaffen werden, die auch in anderen Fällen (z.B. beim Ausradieren von Hilfslinien) eine Rolle spielt [Sch91]: Man lasse das Tilgen von Objekten (in diesem Fall das Tilgen der nicht mehr benötigten Punkte selbst und nicht nur ihrer Adressen) als Konstruktionsschritt zu.

b) Aus inhaltlichen Gründen des zu programmierenden Prozesses ist eine „Adressenrechnung“ nötig, d.h. es gibt z.B. potentiell beliebig viele Punktadressen $P(n,m)$, wobei die

natürlichen Indices *n*, *m* ihrerseits durch das Programm immer neue Werte erhalten und eine wachsende Zahl der konstruierten Punkte nicht wieder „vergessen“ werden darf. Dies ist u.a. in Konstruktionsprogrammen für potentiell unendlich ausgedehnte Parkettierungen erforderlich. Spielt sich eine derartige Konstruktion dann noch in einem euklidisch beschränkten Modell der hyperbolischen Geometrie ab, das in die euklidische Zeichenebene eingebettet ist, so werden sich die zu konstruierenden Punkte gegen den Rand dieses Modells beliebig häufen.

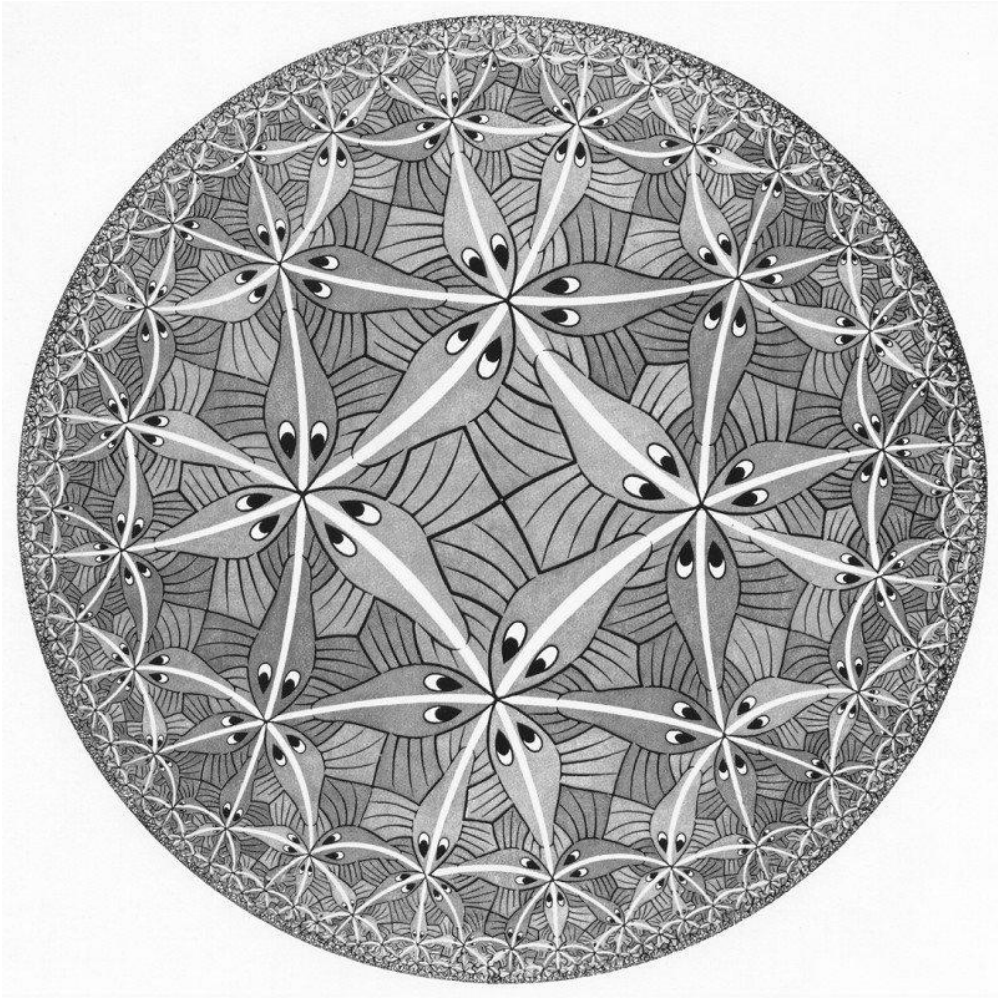


Abbildung 1.4: „Kreislimit III“ von M. C. Escher, Holzschnitt (1959).

Als Illustration verweisen wir auf eines der unter dem Sammelnamen „Kreislimit“ von M. C. Escher entworfenen Muster (Abbildung 1.4), bei denen es sich um nichts anderes als die künstlerisch ausgestaltete Pflasterung der nichteuklidischen Ebene mit einem archimedisch-halbregulären Parkett handelt. Escher hat dabei das Poincaré-Modell der hyperbolischen Geometrie im Innern eines Kreises benutzt, weil sich bei diesem Modell (im Unterschied zu dem ebenfalls innerhalb einer Kreisscheibe existierenden Kleinschen Modell) die Konstruktionen mit Zirkel und Lineal im hyperbolischen Sinn durch Konstruktionen mit euklidischem Zirkel und Lineal realisieren lassen [Sch84, Kap. 2.6], [Sch96]. In einem solchen Fall würde die Plazierung von immer mehr immer dichter beieinander liegenden Adressen ein unlösbares Problem darstellen.

Im konkreten Fall löst sich das Problem praktisch dadurch, dass die Adressierung der Punk-

te sozusagen indirekt durch ihre Lage im bereits konstruierten Teil des Parketts gegeben wird. (Bei im weiteren Sinne regelmäßigen Parketten gibt es natürlich in diesem Sinne „ununterscheidbare“ Punkte, mit denen aber auch in gleicher Weise weiter zu verfahren ist, so dass hier wieder das oben skizzierte Problem eine Rolle spielt, eine willkürliche Reihenfolge von vertauschbaren Teilkonstruktionen syntaktisch korrekt zu beschreiben.) Zugleich wirft dies aber neue theoretische Fragen für die benutzte Programmiersprache auf: Wie formuliert man in bezug auf ein gegebenes „Muster“ (als Adressenersatz) unzweideutig, d.h. im Prinzip für eine Maschine verständlich, mit welchen Punkten der nächste Konstruktionsschritt auszuführen ist?

Ein interessanter Spezialfall des Adressierens im Zusammenhang mit geometrischen Konstruktionen ist die Nutzung der Zeichenebene als konstruktives Modell einer anderen geometrischen Struktur in solchen Fällen, in denen die Zuordnung zwischen den Punkten der zu modellierenden Struktur und ihren Bildobjekten in der Ebene nur dadurch eineindeutig wird, dass man sich zwei oder mehr Exemplare der Ebene übereinandergelegt denkt. Realisiert wird dies dadurch, dass die effektiv verwendeten Punkte bzw. sonstigen Objekte durch einen zusätzlichen Index als der einen oder anderen Ebene zugehörig ausgewiesen werden. Der einfachste und bekannteste Fall ist das Zweitafelverfahren von Monge, wenn man es wirklich als ein ebenes Modell des gesamten Raumes auffassen und nicht (wie in der technischen Praxis üblich) nur beschränkte Objekte darstellen will und daher die „untere“ Halbebene als Bereich des Grundrisses und die „obere“ Halbebene als Bereich des Aufrisses genügt. Im allgemeineren Fall ist – wie in [Sch84, Kap. 2.2] ausgeführt – jeder gegebene oder konstruierte Punkt durch einen Index als Grundrissbild oder als Aufrissbild eines Raumpunktes zu charakterisieren, wobei einundderselbe Punkt mit zweifacher Beschriftung in dieser Doppelrolle auftreten kann. Ein analoger, weniger geläufiger Fall tritt auf, wenn man die gnomonische Projektion benutzt, um ein ebenes konstruktives Modell der Kugeloberfläche zu erhalten [Sch92]. Dabei wird vom Zentrum der Kugel auf dieTangentialebenen zweier diametral gegenüberliegender Punkte projiziert, und die so erhaltenen ebenen Bilder zweier komplementärer Halbsphären werden übereinandergelegt. In dieser Karte lässt sich, wie in [Sch92] ausgeführt wurde, u.a. das Problem der kürzesten Verbindung zweier Punkte mit dem euklidischen Lineal konstruktiv lösen, aber auch hier tritt wieder jeder Punkt der Karte in der Doppelrolle als ein Punkt der „Nord-“ und zugleich als ein Punkt der „Süd-“halbkugel auf und ist durch einen entsprechenden Index an seiner Adresse zu charakterisieren.

Bei den bisher skizzierten Problemen schreibt der fortschreitende Konstruktionsprozess die Lage der zu adressierenden Objekte zwingend vor, während er selbst andererseits ohne diese Adressierung nicht weitergeführt werden kann. Von ganz anderer Art ist das Beschriftungsproblem beim Zeichnen von Graphen mit zu beschriftenden Knoten und/oder Kanten. Hier ist die metrische Realisierung des zu zeichnenden unbeschrifteten Graphen im allgemeinen beliebig oder nur schwach durch bestimmte Nebenbedingungen (etwa, dass die Kanten geradlinig oder stückweise gerade dargestellt werden oder sich nicht überkreuzen oder mit nicht zu spitzen Winkeln nebeneinander in einen Knoten münden sollen, oder dass bei gerichteten Bäumen der Eingangsknoten „oben“ liegen und die hierarchische Struktur schichtweise abgebildet werden soll) eingeschränkt, so dass der Konstruktionsprozess des Graphen im Prinzip vom nachträglichen Beschriftungsprozess getrennt bzw. ein schon gezeichneter Graph nach den Notwendigkeiten der Beschriftung nachträglich deformiert werden kann. Derartige Probleme sind schon seit 1963 in der Literatur behandelt worden [Tut63, Knu63, DETT98, BW00, KW01] und wie sich zeigt, auch ohne Berücksichtigung der Beschriftung schwer. Während die Beschriftung der Knoten nur davon abhängt, wieviel Platz für den jeweiligen „Text“ (es kann sich auch um andere Inhalte, z.B. Symbole oder kleine Bilder handeln) benötigt wird, erfordert die Beschriftung von Kanten unter Umständen, dass die Kanten lang genug für ihren Text sind oder dass die Text tragenden Teile der Kanten zwecks besserer Lesbarkeit waagerecht verlaufen sollen. Einige, zum Teil historische, Beispiele hierzu sollen den Artikel beenden, siehe Abbildungen 1.5 bis 1.9.

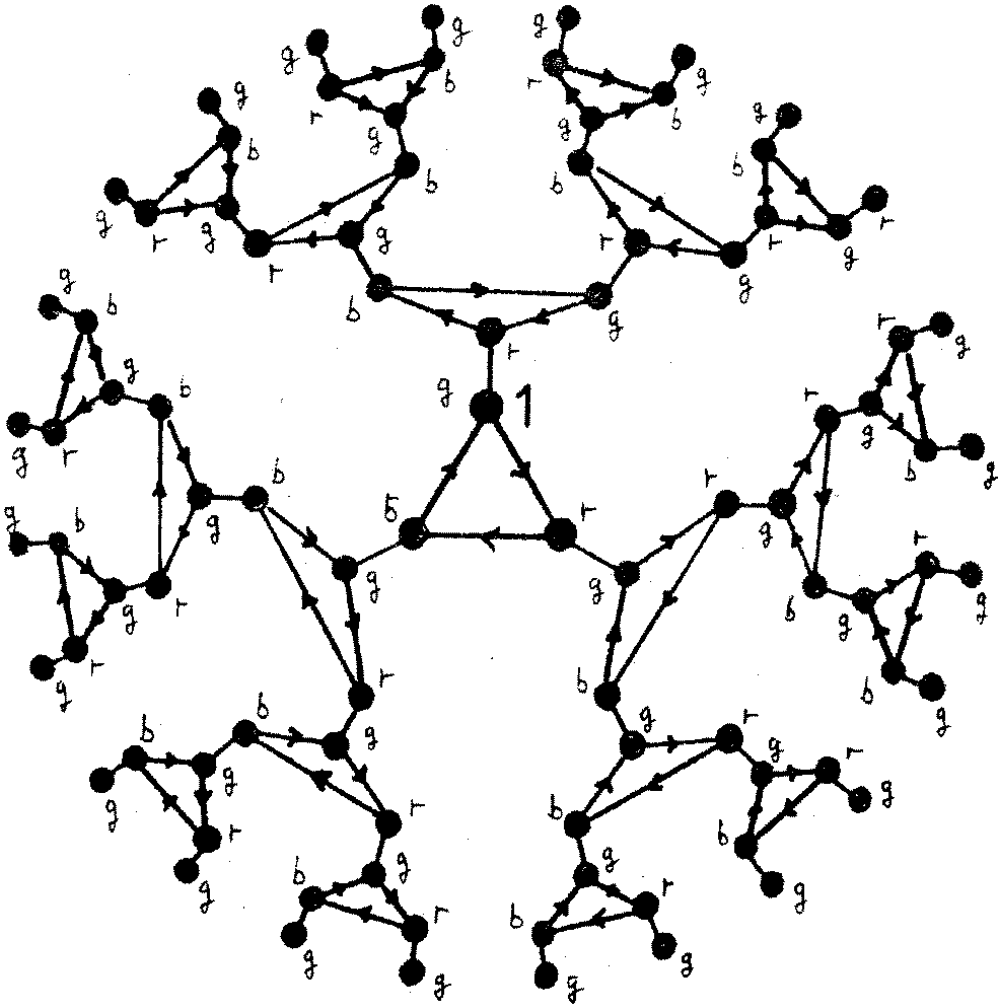


Abbildung 1.5: Dehnsches Gruppenbild des freien Produktes von zwei zyklischen Gruppen der Ordnungen 2 bzw. 3. Die Kanten ohne Pfeil stellen Anwendung eines Gruppenelements der Ordnung 2, die mit Pfeil Anwendung eines Elements der Ordnung 3 dar. Die Knoten sind mit je einem der Buchstaben b(lau), g(rün), r(ot) markiert. Man hat sich den in Wahrheit unendlichen Graphen nach der erkennbaren Regel fortgesetzt zu denken. Er dient zur Visualisierung der Beweisidee des Hausdorffschen Kugelparadoxons. Quelle: [Bri92].

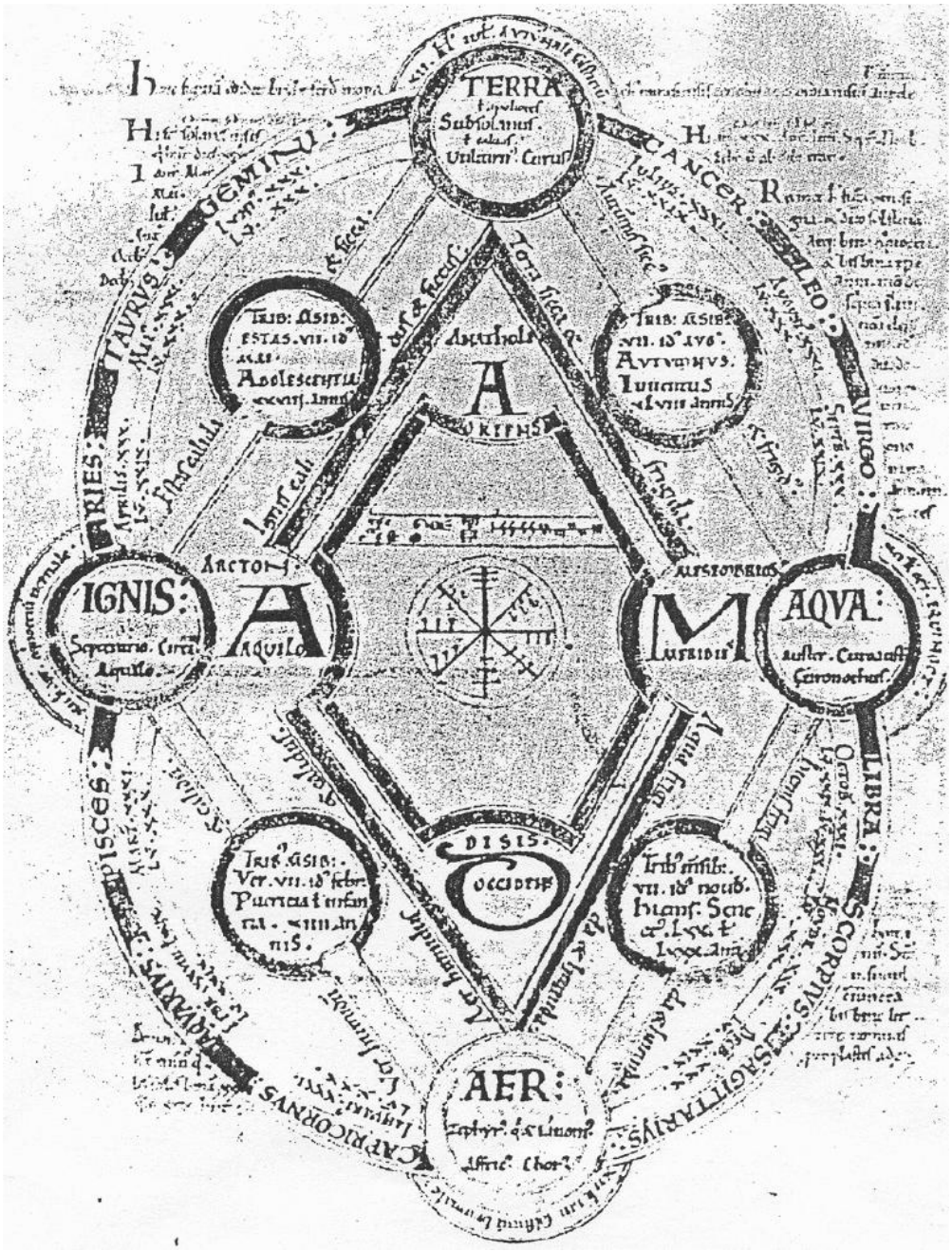


Abbildung 1.6: Die Technik, abstrakte Begriffe und Beziehungen zwischen ihnen durch Graphen mit beschrifteten Knoten und beschrifteten Kanten darzustellen, wurde schon im Mittelalter zu hoher Blüte entwickelt. Der gezeigte Graph stellt Beziehungen zwischen den vier Elementen, Erdteilen, Tierkreiszeichen und Temperamenten dar. (Quelle: Bibliothek des St. John's College, Oxford, Ms. 17, Fol. 7v.)

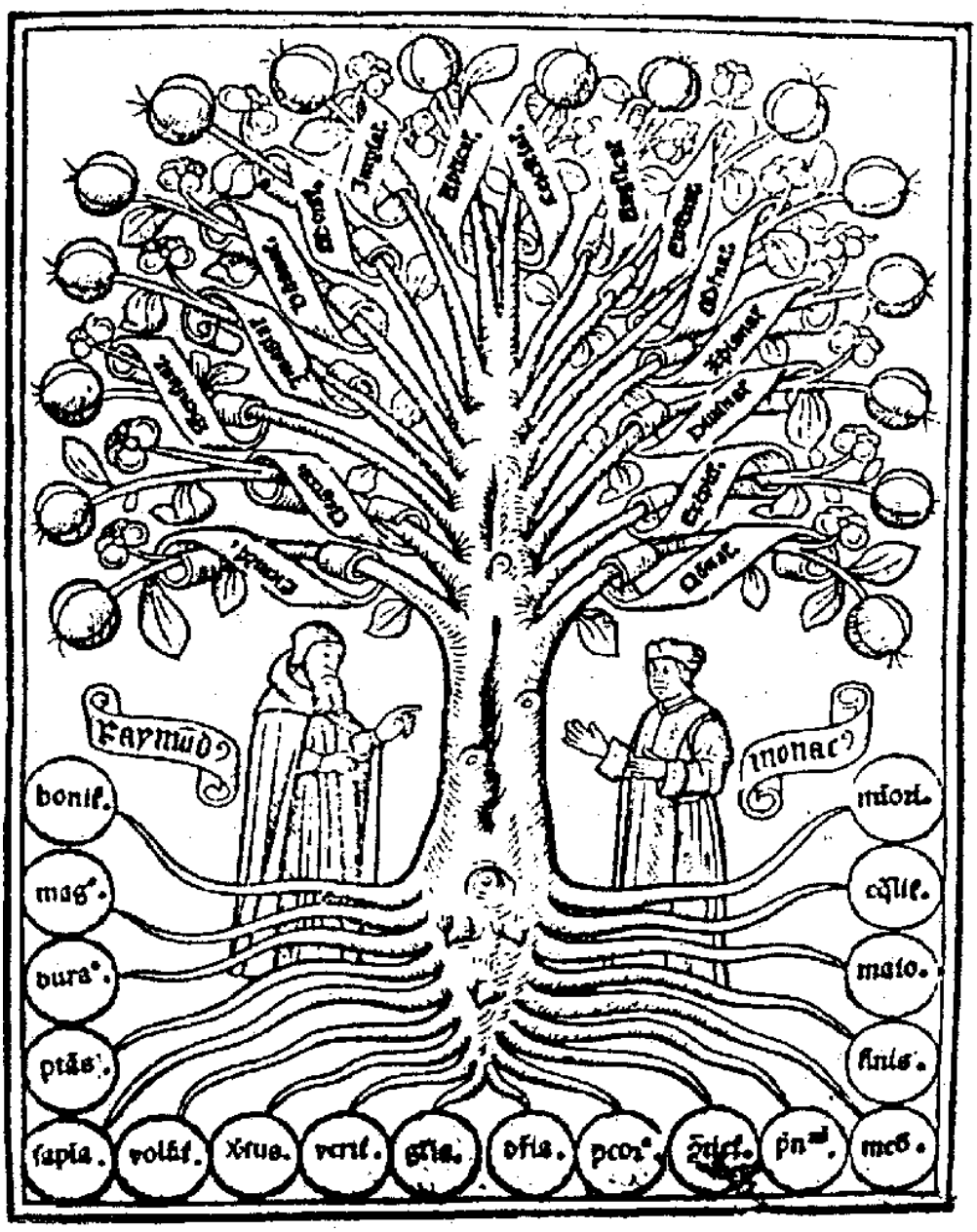
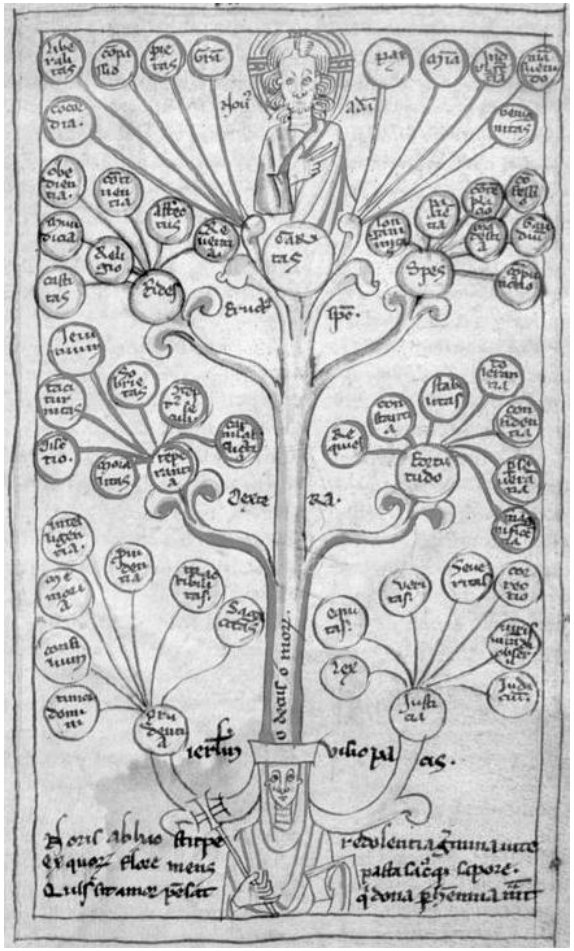


Abbildung 1.7: „Baum der Wissenschaften“ von Raimundus Lullus (1295/96). Graphentheoretisch betrachtet, handelt es sich um einen Wurzelbaum, von dessen Wurzel zwei Teilbäume ausgehen, die Wurzel im botanischen Sinn und die Krone. Während die Endknoten der Wurzel direkt beschriftet sind, sind die Beschriftungen der Endknoten der Krone den zuführenden Kanten zugeordnet. Um die Platzprobleme zu lösen, wurde, wie auch in Abbildung 1.8, reichlich Gebrauch von Abkürzungen gemacht.



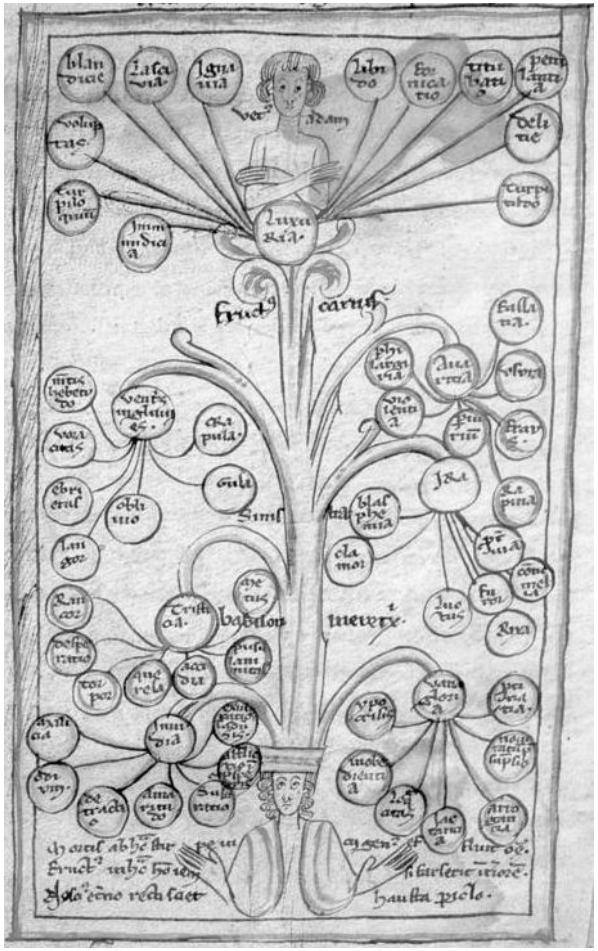


Abbildung 1.8: Baum der Tugenden und Baum der Laster (14. Jh.). Quelle: Hessische Landes- und Hochschulbibliothek Darmstadt.

Abbildung 1.9: Ausschnitt eines Stammbaums der spanischen Herrscher, 11.–16. Jahrhundert. Quelle: [Sok93].

Kapitel 2

Lineares Programmieren

Michael Thon

Die (lineare) Optimierung findet ihren Anfang 1939 bei Kantorovich [Kan39]. Von Dantzig wurde 1947 das Simplexverfahren entwickelt [Dan51], welches bis heute ein Standardverfahren zur Lösung linearer Optimierungsaufgaben darstellt. Die allgemeine Problemstellung der Optimierung lautet: Finde max $f(x)$, wobei $x \in X$. Dabei wird die zu maximierende Funktion $f(x)$ *Zielfunktion* genannt, während x gewissen *Nebenbedingungen* unterliegt.

Bei der *linearen* Optimierung – auch *lineare Programmierung* (LP) genannt – wird zusätzlich die Linearität der Zielfunktion und der Nebenbedingungen gefordert. Jedes lineare Optimierungsproblem lässt sich durch Umformungen, ggf. durch das Ausdrücken einer Gleichung $a_1 x = b$ durch die zwei Ungleichungen $a_1 x \leq b$ und $-a_1 x \leq -b$ und ggf. durch das Aufteilen einer reellen Variablen x_i in zwei nicht-negative Variablen $x_i^+, x_i^- \in \mathbb{R}_0^+$, wobei $x_i = x_i^+ - x_i^-$, in die folgende Form bringen:

$$\begin{array}{l} \max\ c_1 x_1 + \cdots + c_n x_n, \text{ wobei} \\ a_{11} x_1 + \cdots + a_{1n} x_n \leq b_1 \\ \quad \vdots \qquad\qquad \vdots \quad \vdots \\ a_{m1} x_1 + \cdots + a_{mn} x_n \leq b_m \\ x_1, \ldots, x_n \in \mathbb{R}_0^+ \end{array}$$

bzw. in Matrixschreibweise:

$$\begin{array}{l} \max\ c^T x, \text{ wobei} \\ Ax \leq b \\ x \in (\mathbb{R}_0^+)^n \end{array}$$

Anwendung findet das lineare Programmieren häufig bei Produktions- oder Transportproblemen, wo es darum geht, begrenzte Ressourcen optimal einzusetzen. Auch Netzwerk-Fluss-Probleme sind ein typisches Anwendungsgebiet. Dabei geht es darum, den Fluss (z.B. Datenverkehr) durch ein Netzwerk zu maximieren, wobei die einzelnen Verbindungen im Netzwerk eine unterschiedliche maximale Last (Bandbreite) tragen können, wie in Abbildung 2.1 dargestellt.

Dieses Problem soll nun exemplarisch als lineares Optimierungsproblem modelliert werden. Dazu numeriert man die Kanten durch und teilt jede Kante in zwei gerichtete Kanten auf. Den Fluss durch diese Kanten bezeichnet man dann mit x_i^+ und x_i^-, den Gesamtfluss durch das Netzwerk mit x_0, wie in den Abbildungen 2.1 und 2.2 dargestellt.

Dann beschreibt folgendes lineares Optimierungsproblem das Flussproblem, dessen Optimum bei einem Gesamtfluss von 6 liegt:

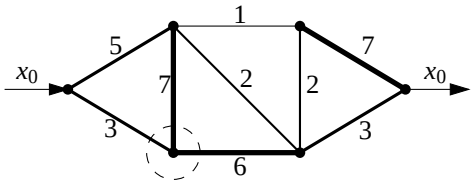


Abbildung 2.1: Netzwerk-Fluss-Problem

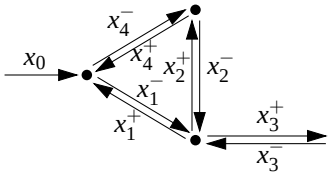


Abbildung 2.2: Modellierung als LP

max x_0 unter den Nebenbedingungen:

- $x_0, x_1^+, x_1^-, \ldots, x_n^+, x_n^- \in \mathbb{R}_0^+$
- $\left\{\begin{array}{c} -3 \le x_1^+ - x_1^- \le 3 \\ -7 \le x_2^+ - x_2^- \le 7 \\ -6 \le x_3^+ - x_3^- \le 6 \\ \cdots \end{array}\right\}$ Der Fluss durch eine Kante wird durch ihre Maximallast begrenzt (hier neun Paare von Ungleichungen).
- $\left\{\begin{array}{c} x_1^+ + x_2^+ + x_3^+ = x_1^- + x_2^- + x_3^- \\ x_0 + x_1^+ + x_4^+ = x_1^- + x_4^- \\ \cdots \end{array}\right\}$ Gleichgewicht zwischen Ein- und Ausfluss bei den Knoten (hier sechs Gleichungen)

Um nun zu einem Lösungsansatz für lineare Optimierungsprobleme zu gelangen, benötigt man noch einige Begriffe und Feststellungen (siehe hierzu auch Abbildung 2.3).

Die Menge $M := \{x \in (\mathbb{R}_0^+)^n \mid Ax \le b\}$ der Punkte im $\mathbb{R}^n$, die den Nebenbedingungen genügen, nennt man die *zulässige Menge*. Ein Punkt x^* der zulässigen Menge heißt *optimal*, wenn $c^T x^* \ge c^T x$ für alle $x \in M$ gilt. Man kann leicht zeigen, dass

1. die zulässige Menge ein konvexes Polyeder bildet,
2. falls ein optimales $x^* \in M$ existiert, so existiert auch ein optimales $x^* \in M$, das sogar *Ecke* von M ist.

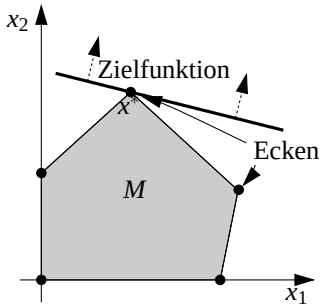


Abbildung 2.3: Geometrische Interpretation eines LPs

Dies motiviert als Lösungsansatz den *Simplex-Algorithmus*, der 1947 von Dantzig [Dan51] vorgestellt wurde, siehe Abbildung 2.4. Als Eingabe benötigt der Simplex-Algorithmus die zu maximierende Zielfunktion und die Angabe der zulässigen Menge in Form der Nebenbedingungen. Als Ausgabe erhält man eine optimale Lösung, falls diese existiert.

Falls der Simplex-Algorithmus einen Halbstrahl als Kante der zulässigen Menge auswählt, entlang dem die Zielfunktion also zunimmt, bedeutet dies aufgrund der Linearität der Zielfunktion gerade, dass es kein Optimum gibt, sondern der Wert der Zielfunktion unter Beachtung der Nebenbedingungen beliebig verbessert werden kann. Der Algorithmus bricht dann ab.

Meist existiert mehr als eine Kante, entlang der die Zielfunktion verbessert werden kann. Es kann dann von großem Vorteil sein, eine geschickte Kantenauswahl zu treffen. Setzt man das recht naheliegende Auswahlprinzip voraus, bei dem die Kante mit dem größten Zuwachs der Zielfunktion gewählt wird, so hat der Simplex-Algorithmus im schlechtesten Fall exponenti-

SIMPLEX-ALGORITHMUS

Finde eine Ecke v_0 der zulässigen Menge M
$v \leftarrow v_0$
while ($\exists$ Kante $k \in M$, die von v ausgeht,
entlang der $c^T x$ zunimmt) **do**
if (k Halbstrahl) **then halt**
else
$v \leftarrow$ zu v nächste Ecke von M auf k
end

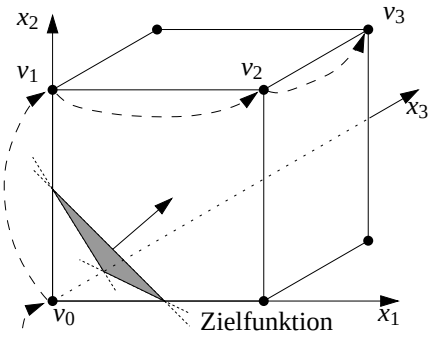


Abbildung 2.4: Pseudocode und geometrische Interpretation des Simplex-Algorithmus

elle Laufzeit [KM72].

Es gibt zwar auch Algorithmen zur linearen Optimierung, die im schlechtesten Fall nur polynomielle Laufzeit haben, wie etwa den von Karmarkar [Kar84], allerdings hat sich der Simplex-Algorithmus in der Praxis bewährt und ist paradoxerweise meist sogar schneller als der von Karmarkar.

Als Ausgangspunkt für einen tieferen Einstieg in das umfangreiche Thema der linearen Optimierung kann die Linear-Programming-FAQ des amerikanischen Optimization Technology Center im Internet empfohlen werden [Fou00].

2.1 Ganzzahliges lineares Programmieren

Beim *ganzzahligen* linearen Programmieren (englisch *integer programming*; IP) fordert man zusätzlich $x \in \mathbb{Z}^n$. Solche Probleme findet man oft, da viele Ressourcen nur in ganzzahligen Stückzahlen vorliegen. Man überlegt sich leicht, dass das Optimum des zugehörigen linearen Optimierungsproblems eine Schranke für das Optimum des IP-Problems ist, da die diskrete zulässige Menge des IP-Problems eine Teilmenge der zulässigen Menge des zugehörigen linearen Optimierungsproblems ist. Man kommt schnell auf die Idee, einfach das lineare Optimierungsproblem zu lösen und dann zu runden, doch dies liefert im Allgemeinen keine optimale Lösung, wie Abbildung 2.1 zeigt.

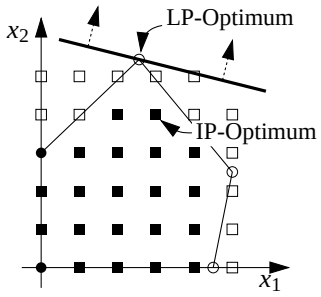


Abbildung 2.5: Ganzzahliges lineares Programmieren

Ein Spezialfall des IP ist das *0-1-IP*. Dabei werden für die einzelnen Variablen nur die Werte 0 und 1 zugelassen. Solche 0-1-Variablen können dazu benutzt werden, logische Nebenbedingungen als lineare Gleichungen oder Ungleichungen zu formulieren, wie in Tabelle 2.1 zu sehen ist. Dabei entspricht die Zahl 0 dem Wahrheitswert *falsch* und 1 *wahr*.

Für IP-Probleme sind zwei Lösungsverfahren sehr verbreitet, das 1958 von Gomory vorgestellte *Schnittebenen*-Verfahren [Gom58] und das 1960 von Land und Doig beschriebene *Branch-and-Bound*-Verfahren [LD60]. Oft wird auch eine Kombination von beiden angewandt, was man dann meist als *Branch-and-Cut*-Verfahren bezeichnet.

Beim Schnittebenenverfahren wird schrittweise das zugehörige LP gelöst und die optimale Ecke v, wenn sie nicht ganzzahlig ist, „abgeschnitten“, indem weitere Nebenbedingungen (Schnittebenen) hinzugefügt werden, die die zulässige Menge des IP nicht verändern, jedoch

x_1 **or** x_2	x_1 **and** x_2	x_1 **nand** x_2	$x_1 \Rightarrow x_2$	$(a_1x \leq b)$ **or** $(a_2x \leq b)$
$x_1 + x_2 \geq 1$	$x_1 + x_2 = 2$	$x_1 + x_2 \leq 1$	$x_1 - x_2 \leq 0$	$\left\{\begin{array}{l} a_1x - My \leq b_1 \\ a_2x - M(1-y) \leq b_2 \\ y \in \{0,1\},\ M \text{ groß} \end{array}\right\}$

Tabelle 2.1: Logische Verknüpfungen von Nebenbedingungen können durch Ungleichungen mit den entsprechenden 0-1-Variablen codiert werden.

die zulässige Menge des LPs um mindestens v echt verkleinern. Es ist immer möglich, Schnittebenen so anzugeben, dass das Schnittebenenverfahren nach endlich vielen Schritten bei einem Optimum des IPs abbricht, jedoch ist die Effizienz des Verfahrens in starkem Maße von einer geschickten Wahl der Schnittebenen abhängig.

Die Idee des Branch-and-Bound-Verfahrens besteht darin, die zulässige Menge des IP-Problems immer weiter zu zerteilen (*branching*), wodurch man einen Baum aus Teilproblemen erhält. Diese Verzweigung führt man solange fort, bis man bei einer Menge angelangt, von der man weiß, dass sie das Optimum nicht enthält (*bounding*). Am Ende dieses Prozesses stellen die Blätter des Baumes mögliche Lösungen dar, von denen eine optimal ist, falls eine optimale Lösung überhaupt existiert. Für das Bounding benötigt man gute obere und untere Schranken für das Optimum der Zielfunktion bei gegebener zulässigen Menge. Beim Branchen gibt es meist mehrere Möglichkeiten, von denen einige besser geeignet sind als andere. Ferner hat man bei diesem Verfahren die Freiheit, den Baum der Teilprobleme mit einer Tiefen- oder einer Breitensuchstrategie aufzubauen.

Für das ganzzahlige lineare Programmieren gibt es nicht generische Lösungsverfahren, wie den Simplex-Algorithmus beim LP, sondern nur Verfahren, deren Effizienz von weiterem Wissen über das konkrete Problem abhängt.

2.2 Anwendung bei der Landkartenbeschriftung

Um einen Eindruck davon zu bekommen, wie ganzzahliges lineares Programmieren bei der Landkartenbeschriftung angewandt werden kann, soll das einfache Problem der Punktbeschriftung mit achsenparallelen Rechtecken gegebener Größe im 4-Positionen-Modell als 0-1-IP modelliert werden. Dabei bedeutet 4-Positionen-Modell, dass für jeden zu beschriftenden Punkt p gerade die vier achsenparallelen Rechtecke gegebener Größe, die p mit einer Ecke berühren, als Beschriftungskandidaten benutzt werden, siehe Abbildung 2.6.

Definition 2.1 *Unter dem Problem der Beschriftungs-Anzahl-Maximierung (BAM) versteht man folgendes: Zur Punktmenge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ und gegebenen Menge B der Beschriftungskandidaten finde man eine Teilmenge $L = \{l_1, \ldots, l_m\} \subset B$, so dass*

- $l_i \cap l_j = \emptyset \quad \forall\, i \neq j$,
- *jeder Punkt höchstens eine Beschriftung erhält und*
- $|L|$ *maximal ist.*

Um Relationen wie Überschneidung zwischen den Beschriftungskandidaten zu erfassen, bietet es sich an, die Menge der Beschriftungskandidaten als Graphen aufzufassen. Man berechnet also zunächst den *Konfliktgraphen* $G = (B, E)$ zur Menge der Beschriftungskandidaten. Da es im 4-Positionen-Modell gerade $4n$ Beschriftungskandidaten gibt, kann man den Konfliktgraphen naiv in $O(n^2)$ Zeit berechnen. Allerdings ist die Zahl k der Paare von sich schneidenden

Beschriftungskandidaten bei „vernünftigen“ Landkarten nicht quadratisch, sondern eher linear, so dass es sich lohnt, einen *output-sensitiven* Algorithmus zu verwenden, das heißt einen Algorithmus, dessen Laufzeit nicht nur von der Größe $4n$ der Eingabe, sondern auch von der Größe k der Ausgabe abhängt. Ein Beispiel dafür ist der *Sweepline*-Algorithmus von Edelsbrunner und Maurer [EM81], dessen Laufzeit $O(k+n\log n)$ beträgt.

Um auszuschließen, dass ein Punkt mit mehr als einer Beschriftung versehen wird, fügt man noch Kanten zwischen allen Beschriftungskandidaten hinzu, die zu einem Punkt gehören, und erhält so den *erweiterten* Konfliktgraphen, siehe Abbildung 2.7.

Definition 2.2 *Der Graph $G=(B,E)$ wird als* Konfliktgraph *(KG) der Menge B der Beschriftungskandidaten bezeichnet, wenn*

- *dessen Knoten gerade den Beschriftungskandidaten entsprechen*
- *zwischen zwei Knoten genau dann eine Kante existiert, wenn sich die entsprechenden Beschriftungskandidaten überschneiden.*

Ein Konfliktgraph $G=(B,E)$ wird als erweiterter *Konfliktgraph (EKG) bezeichnet, wenn zusätzlich Kanten zwischen allen Beschriftungskandidaten, die zu einem Punkt gehören, existieren.*

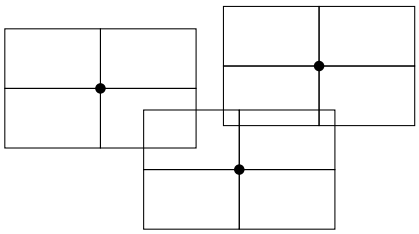


Abbildung 2.6: Das Beschriftungsproblem BAM im 4-Positionen-Modell.

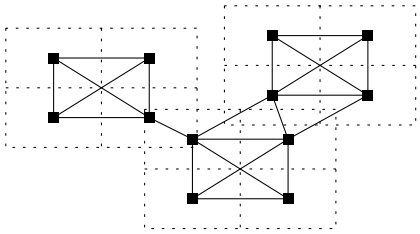


Abbildung 2.7: Darstellung als EKG.

Eine zulässige Beschriftung der Punktmenge P entspricht dann gerade einer *unabhängigen Menge U* von Knoten im erweiterten Konfliktgraphen, also einer Menge $U \subset B$ mit der Eigenschaft: $\forall b_i, b_j \in U,\ i \neq j$ gilt $(b_i, b_j) \notin E$. Demnach entspricht eine *optimale* Beschriftung der Punktmenge P gerade einer unabhängigen Menge maximaler Kardinalität im erweiterten Konfliktgraphen.

Da im 4-Positionen-Modell zu jedem Punkt vier Beschriftungskandidaten gehören, kann man jede Teilmenge $B_x \subset B$ durch einen $4n$-Tupel $x=(x_1,\ldots,x_{4n})$ aus Nullen und Einsen darstellen, indem man für alle i festlegt $x_i := 1$ genau dann, wenn $b_i \in B_x$, ansonsten $x_i := 0$. Damit kann man nun die Menge X_U aller unabhängigen Mengen von Knoten aus dem erweiterten Konfliktgraphen G – und diese entspricht gerade der Menge der legalen Beschriftungen der Punktmenge P – durch lineare Nebenbedingungen folgendermaßen charakterisieren:

$$X_U = \{x \in \{0,1\}^{4n} \mid \forall (b_i,b_j) \in E, i \neq j \text{ gilt } x_i + x_j \leq 1\}.$$

Dabei besagt die Bedingung $x_i + x_j \leq 1$ gerade, dass höchstens eine der sich ausschließenden Beschriftungen (da $(b_i,b_j) \in E$) in einer unabhängigen Menge vorkommen kann (siehe auch Tabelle 2.1).

Ferner gilt offenbar: $|B_x| = x_1 + \cdots + x_{4n}$. Somit ist das ursprüngliche Beschriftungsproblem BAM im 4-Positionen-Modell äquivalent zum 0-1-IP:

$\max (x_1 + \cdots + x_{4n})$, wobei $x = (x_1, \ldots, x_{4n}) \in X_U$.

Um dieses 0-1-IP zu lösen, haben Verweij und Aardal [VA99] einen Branch-and-Cut-Algorithmus entwickelt, wobei sie die besondere geometrische Natur des Beschriftungsproblems ausnutzen, um geeignete Schnittebenen sowie gute obere und untere Schranken für das jeweilige Optimum eines Teilproblems zu finden.

Zum Abschluss soll noch ein interessantes Ergebnis zum allgemeineren BAM-Problem mit Beschriftungskandidaten im 4-Schiebe-Modell genannt werden. Vom 4-Schiebe-Modell spricht man, wenn als Beschriftungskandidaten eines Punktes alle achsenparallelen Rechtecke gegebener Größe, die den Punkt berühren, betrachtet werden. Durch eine geschickte 0-1-IP-Formulierung ist es Klau und Mutzel [KM00] erstmals gelungen, einen Algorithmus anzugeben, der das Problem noch mit einer Eingabegröße von 600 zu beschriftenden Punkten in vertretbarer Laufzeit lösen kann. Es ist erstaunlich, dass sie zur Lösung des kontinuierlichen Schiebe-Modells ein diskretes 0-1-IP verwenden. Allerdings stellen sie fest, das die Laufzeit beim optimalen Lösen sehr unterschiedlich auf Probleme gleicher Eingabegröße verteilt ist. So schreiben sie über ein verwandtes Problem [KM02], dass ihr Algorithmus bei den meisten Problemen weniger als eine Sekunde benötigt, bei einem jedoch ganze 68 Minuten.

Kapitel 3

Simulated Annealing und genetische Algorithmen

Tim Hoffmann

Bei der Anordnung von Punktbeschriftungen ist eine Lage der Label zu finden, so dass sich möglichst wenige überschneiden oder entfernt werden müssen. Man kann diesen Sachverhalt auch als allgemeines Optimierungsproblem $\{(g, c(g)) : g \in G\}$ betrachten. Dabei ist die Grundmenge G die Menge aller möglichen Beschriftungen, oder mathematischer: die Menge aller n-Tupel über M, wobei n die Anzahl der zu beschriftenden Punkte und M die Menge der Lagemöglichkeiten ist. Die Zielfunktion $c(g)$ soll dabei minimiert werden.

Im speziellen Fall der Beschriftung dient als Zielfunktion zum Beispiel die Anzahl der sich überschneidenden Label addiert zur zweifachen Anzahl der unbeschrifteten Punkte. Simulated Anneling und genetische Algorithmen sind Instrumente zur Lösung des beschriebenen allgemeinen also auch des speziellen Optimierungsproblems. Ihre Gemeinsamkeit: Sie versuchen mit Hilfe der Simulation natürlicher Vorgänge und Gesetzmäßigkeiten ein optimales Arrangement der Label zu erreichen. Vorbild ist dabei das langsame Abkühlen einer Molekülmenge für Simulated Annealing und die darwinistische Evolutionslehre für genetische Algorithmen. Es folgt eine kurze Beschreibung.

3.1 Simulated Annealing

Voraussetzungen für die Anwendung von Simulated Annealing ist, dass jeder Label um einen seinen zu beschriftenden Punkt herum eine gewisse (meist kleine konstante) Anzahl von Lagemöglichkeiten hat (d.h. $|M| < \infty$), siehe etwa das 8-Positionen-Modell in Abbildung 3.1. Hier gilt $M = \{1, \ldots, 8\}$; ist auch das Entfernen von Labeln erlaubt, so nimmt man $M = \{0, \ldots, 8\}$.

Abbildung 3.1: Beispiel für mögliche Labelanordnungen

Christensen at al. [CMS95] beschreiben ihren Algorithmus wie folgt. Als Startposition nimmt jeder Label eine zufällige Position ein, also einen Wert aus M. Nun wird zufällig mit Gleichverteilung den Labeln eine neue Position zugewiesen. Verbessert sie sich, also sinkt die Zielfunktion, wird diese neue Lage übernommen, steigt sie oder bleibt sie konstant, wird die neue Position mit einer gewissen Wahrscheinlichkeit P übernommen. Diese Wahrscheinlichkeit hängt von der Kontrollgröße T ab, die hier auch als Temperatur bezeichnet wird. Je geringer die Temperatur, um so geringer die Wahrscheinlichkeit, dass sich der Label aus seiner Position entfernt. Die Intensität der Temperaturabkühlung wird durch den Erfolg des Prozesses gesteuert. Je größer der Erfolg, um so schneller die Abkühlung.

Diese Intensität ist auch in Abhängigkeit von gewünschter Endqualität und verfügbarer Zeit beeinflussbar. Je stärker die Abkühlung, um so kürzer ist die Simulationszeit, aber um so größer ist auch die Wahrscheinlichkeit einer schlechten Qualität des Endprodukts. Zur Bewertung dieser Qualität dient die Zielfunktion. Ist T so gering, dass sich die Lage keines Labels mehr verändert, bricht der Algorithmus ab.

Weitere Möglichkeiten, Simulated Annealing zur Lösung von Beschriftungsproblemen insbesondere bei sehr dichten Punktmengen einzusetzen, wurden von Zoraster [Zor97] untersucht.

3.2 Genetische Algorithmen

Van Dijk et al. [vDTdB99, vDTdB00, vD01] arbeiten mit einer bestimmten Menge von Labelanordnungen, d.h. mit einer Teilmenge von G, die *Population* genannt wird. Diese Anordnungen werden als Individuen einem Wettbewerb ausgesetzt. Sie unterliegen einer Evolution, nur die fittesten werden überleben. Als Maß für die Fitness dient die Zielfunktion. Je kleiner der Funktionswert, um so größer ist die Fitness. Die durch ein festes Kriterium bestimmten Individuen müssen sich „paaren“. Dabei werden zufällig mit einer gewissen Wahrscheinlichkeit P_c ausgewählte Labelpositionen ausgetauscht. Diesen Operator nennt man *Crossover*. Außerdem kommt es zu einer *Mutation*, also zufälligen Labelpositionsveränderungen mit einer Wahrscheinlichkeit P_m. Diese neuen Individuen ersetzen nun die bisher „unfittesten“. Die Qualität der Lösungen sollte im Mittel zunehmen. Der Algorithmus bricht ab, wenn die Fitness der Individuen sich der des besten genug angenähert hat, der Evolutionsdruck also keine Verbesserung mehr bewirkt.

Genetische Algorithmen für Landkartenbeschriftungsprobleme wurden auch von Rumplmaier [Rum98] und Raidl [Rai98, Rai99] untersucht.

Kapitel 4

Ein Konzept für die automatische Schriftplatzierung

Wolfgang Kresse

Die Grafik des Kartenbildes und die Schrift müssen sich die begrenzte Fläche eines Kartenblattes teilen. Dabei entstehen an vielen Stellen Konflikte um den verfügbaren Platz, und zwar zwischen der Schrift und der Grafik der Karte einerseits und zwischen Schriften untereinander andererseits. Das Ziel der Schriftplatzierung ist eine gut lesbare Anordnung der Schriftzüge und ein optimaler Ausgleich zwischen den konkurrierenden Ansprüchen auf die begrenzte Fläche.

Es wird eine Methode zur automatischen Platzierung von Schrift in topographischen Karten und Stadtplänen vorgestellt. Das Konzept sieht vor, aus dem digitalen Datenbestand einer Karte ohne Schrift, dem zugehörigen Schriftgut und weiteren Parametern die endgültigen Positionen und Schreibrichtungen der einzelnen Namen zu berechnen. Dabei wurde von den folgenden Grundsätzen ausgegangen:

- Die Schriftplatzierung wird als ein flächenhaftes Optimierungsproblem betrachtet, bei dem unter gegebenen Randbedingungen, den Platzierungsregeln, und innerhalb einer strukturierten Fläche, der Kartengraphik, eine optimale Position für die Schrift gefunden werden muss.
- Die Schriftplatzierung wird in zwei Phasen ausgeführt. In der ersten Phase wird jeder Name so platziert, als wäre er der einzige auf der Karte. Dabei werden seine möglichen Positionen gesucht, entsprechend der überdeckten Kartengraphik gewichtet und abgespeichert. In der zweiten Phase wird für alle Namen gemeinsam eine optimale Verteilung gesucht, also die gegenseitige Überlappung von Beschriftungen beseitigt. Dieses Vorgehen ermöglicht eine individuell abgestimmte Positionsauswahl für jeden Namen einerseits und den Einsatz mathematischer Standardverfahren zur Lösung des Optimierungsproblems andererseits.
- Bei der Konzipierung der Algorithmen werden Raster- und Vektorverfahren in Betracht gezogen und dasjenige ausgewählt, das im Einzelfall die bessere Modellierung verspricht.
- Die Kartengraphik, ob als digitaler Vektor- oder Rasterdatenbestand vorliegend, wird zu einer Rastermatrix umgebildet. Die Pixel dieser Matrix repräsentieren kleine Teilflächen innerhalb der Karte. Die Pixelwerte geben an, wie wichtig diese Flächen für das

Verständnis der Karte sind. Die Werte der Rastermatrix stellen das Bedeutungsgebirge dar.

- Für die Platzierung werden die Kartenobjekte in die Geometrietypen punktförmig, linienförmig und flächenförmig eingeteilt und nach unterschiedlichen Gesichtspunkten beschriftet. Punkthafte Objekte werden meist als Sonderfall von flächenhaften Objekten behandelt.
- Namen werden geometrisch während der Platzierung durch ein umschreibendes Rechteck repräsentiert. In Ausnahmefällen wird jeder einzelne Buchstabe als Rechteck dargestellt.
- Das Konzept beschreibt Maßnahmen, die bei einem durch die Optimierung nicht lösbaren Geometriekonflikt zu treffen sind. Dazu gehören Verkleinern, Abkürzen oder Weglassen von Schrift.
- Die Eingabedaten umfassen die folgenden Typen:
  - digitale Karte
  - Schriftgut
  - Parameter, die den Einfluss der Kartenobjekte auf die Schriftposition beschreiben
  - sonstige Eingabewerte
- Die Ausgabedaten enthalten vor allem die endgültigen Positionen der Schriftzüge. Hinzu treten ergänzende Angaben, z.B. besondere Standlinien, gesperrte Darstellungen oder graphische Variationen der Schrift.

4.1 Modellierung der Kartengraphik mithilfe des Bedeutungsgebirges

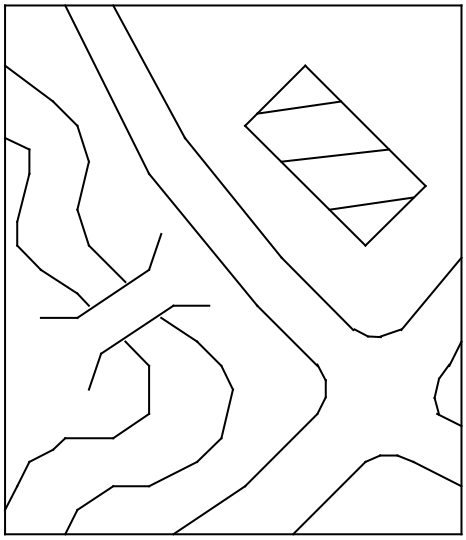
Für die Berücksichtigung der Kartengraphik bei der Platzierung der Schrift wird die gesamte digitale Karte vor Beginn der Schriftplatzierung in eine Zahlenmatrix überführt, die Bedeutungsgebirge genannt wird. Die rasterförmige Darstellung der Karte eignet sich für eine zahlenmäßige Darstellung der Karteninformation pro Flächeneinheit. Jedes Pixel repräsentiert ein bestimmtes Maß an Bedeutung. Die Bedeutung drückt nicht nur den Schwärzungsgrad des Kartenuntergrundes aus. Vielmehr lässt sich über den Pixelwert der Informationgehalt der Karte an der betreffenden Stelle vermitteln. Für die Festlegung eines Pixelwertes sind die Kartengraphik und die Bedeutung des Kartenobjekts zugrunde zu legen. Bei der Platzierung des Namens ist dann, unabhängig von einzelnen Signaturen diejenige Fläche zu finden, auf der der Name am wenigsten Karteninformation verdeckt, auf der also die Summe der Werte der überdeckten Pixel minimal ist.

Die Pixelwerte dieser Zahlenmatrix sind mit 8 Bit kodiert und können daher ganzzahlige Werte zwischen 0 und 255 annehmen. Jeder Pixelwert ist ein Maß dafür, wie stark die betreffende Position von Schrift freigehalten werden soll. Hohe Pixelwerte können als Berge des Bedeutungsgebirges interpretiert werden. Sie repräsentieren eine hohe Dichte der Karteninformation und bedingen ein niedriges Gewicht für die Eignung als Schriftposition. Niedrige Pixelwerte dagegen können als Täler des Bedeutungsgebirges interpretiert werden. Sie stehen für eine geringe Dichte der Karteninformation und kennzeichnen bevorzugte Positionen für die Schriftplatzierung.

Das Bedeutungsgebirge ermöglicht es, flächenhafte Bedingungen einzuführen, die für alle zu platzierenden Namen in gleicher Weise gelten. Dabei handelt es sich vor allem um Platzierungsregeln, die den Kartenhintergrund betreffen.

- Das Bedeutungsgebirge stellt dem Rechner die Information zur Verfügung, die der Kartograph während der manuellen Schriftplatzierung ständig vor Augen hat, nämlich den Kartenhintergrund. Der Rechner ist mit Hilfe des Bedeutungsgebirges in der Lage, zeichnungsarme Teile der Karte zu finden.
- Die Forderung, bestimmte Kartenelemente von Schrift freizuhalten, z.B. Trigonometrische Punkte, kann über hohe Pixelwerte des Bedeutungsgebirges erfüllt werden.
- Bei dichter Anordnung der Kartengraphik müssen Teile davon durch Schrift verdeckt werden. Mittels des Bedeutungsgebirges kann gesteuert werden, in welcher Prioritätenfolge welche Kartenobjektarten überdeckt werden, z.B. zunächst Wiesen und erst bei stärkerer Verdichtung auch Straßen.

Das Bedeutungsgebirge liegt deckungsgleich über der zu beschriftenden Karte. Die als Pixelwert dargestellte Karteninformation muss so gut wie möglich dem Kartenbild entsprechen, auf das die Schrift platziert wird, denn das Bedeutungsgebirge dient als geometrischer Bezug für die Schriftplatzierung. Abweichungen zwischen der für die Platzierung benutzten Rastermatrix und dem eigentlichen Kartenbild können sich dahingehend auswirken, dass Schrift in Bereiche gesetzt wird, die in der Karte mit anderen Informationen gefüllt sind. Abweichungen zwischen dem Bedeutungsgebirge und der Karte sind vor allem durch die begrenzte Auflösung bedingt. Eine 5 km $\times$ 5 km große Karte, die von einem Bedeutungsgebirge überzogen wird, das 1024 mal 1024 Pixel besitzt, wird z.B. nur in etwa 5 m $\times$ 5 m große Flächenelemente aufgelöst. Folglich werden mögliche Namenpositionen nur in 5-m-Sprüngen gesucht. 5 m entsprechen 0.2 mm in einer TK 25, 1:25000. Feinere dazwischenliegende Strukturen bleiben unberücksichtigt.



0 0 7 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 7 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0
3 0 0 7 0 7 0 0 0 0 0 0 4 0 0 0 0 0 0
0 3 0 0 7 0 7 0 0 0 0 4 0 4 0 0 0 0 0
0 0 3 0 7 0 7 0 0 0 4 4 4 0 4 0 0 0 0
3 0 0 3 0 7 0 7 0 0 4 0 0 0 0 4 0 0 0
0 3 0 3 0 7 0 0 7 0 0 4 4 4 4 0 4 0 0
3 0 0 3 0 0 7 0 0 7 0 0 4 0 0 0 0 4 0
3 0 0 3 0 0 0 7 0 7 0 0 0 4 4 4 4 0 0
3 0 0 3 0 5 0 0 7 0 7 0 0 0 4 4 0 0 0
0 3 0 0 3 5 0 0 7 0 0 7 0 0 0 0 0 0 0
0 0 3 0 5 0 0 0 0 7 0 0 7 0 0 0 0 0 7
0 5 5 5 0 0 5 5 0 0 7 0 0 7 0 0 0 7 0
0 0 0 0 0 5 0 3 0 0 0 7 0 0 7 7 7 0 0
0 0 0 0 5 3 0 0 3 0 0 0 7 0 0 0 0 0 7
0 0 0 5 0 3 0 0 0 3 0 0 0 7 0 0 0 7 0
0 0 0 0 0 3 0 0 0 3 0 0 0 7 0 0 0 7 0
0 0 0 0 3 0 0 0 0 3 0 0 7 0 0 0 0 0 7
0 3 3 3 0 0 0 0 3 0 0 7 0 0 0 7 7 0 0
3 0 0 0 0 3 3 3 0 0 7 0 0 0 7 0 0 7 7
0 0 0 3 3 0 0 0 0 7 0 0 0 7 0 0 0 0 0
0 0 3 0 0 0 0 0 7 0 0 0 7 0 0 0 0 0 0

Abbildung 4.1: Kartengraphik (links) und zugehöriges Bedeutungsgebirge (rechts) Zuordnung: Straße≙7, Brücke≙5, Haus≙4 und Bach≙3

In heutigen Geoinformationssystemen liegen digitale Karten meistens als Definitionsgeometrie der Kartenobjekte vor. Die graphische Ausgestaltung, die Signaturierung, findet erst unmittelbar vor der Ausgabe statt. Genau diese graphische Ausgestaltung muss aber vor der Ableitung des Bedeutungsgebirges fertiggestellt sein, weil die Schriftplatzierung vor dem Hintergrund des fertigen Kartenbildes erfolgen muss. Die Daten der Kartengraphik können sowohl in Form von Vektoren als auch als Rasterbild vorliegen. In beiden Fällen müssen die Daten zur Herstellung des Bedeutungsgebirges in das Raster des Bedeutungsgebirges umgeformt

werden. Die Vektordaten werden einer Vektor-Raster-Wandlung unterzogen. Die Rasterdaten müssen in den meisten Fällen ebenfalls umgebildet werden, weil das gerasterte Kartenbild nur selten mit der angestrebten Auflösung des Bedeutungsgebirges übereinstimmen dürfte. Die notwendige Umbildung ist ein „Resampling-Prozess“ , für den in der digitalen Bildverarbeitung Standardverfahren zur Verfügung stehen, z.B. die Suche nach dem „nächsten Nachbarn“ , die „bilineare Interpolation“ oder die „bikubische Interpolation“ .

Die Bestimmung des Bedeutungsgebirges umfasst neben der geometrischen Datenwandlung auch die Festlegung der Höhen, die als Werte jedes einzelnen Pixels dargestellt sind. Die folgenden Betrachtungen gehen von einer Vektorkarte aus. Im einfachsten Fall könnten Pixel an Stellen mit Kartenzeichnung den Wert „1“ und an zeichnungsfreien Stellen den Wert „0“ erhalten. Das Bedeutungsgebirge ist dann nicht weiter nach Objekten differenziert. Die Schrift wird vorzugsweise an zeichnungsfreien Stellen platziert. Will man die Schriftplatzierung gezielter beeinflussen und bestimmte Objekte von Schrift freihalten, dann müssen diese Objekte einen höheren Pixelwert, z.B. „2“ bekommen, während die anderen weniger wichtigen Objekte den Wert „1“ behalten. Auf diese Weise kann das Bedeutungsgebirge im Rahmen der technischen Grenzen – Objektstruktur der Karte, maximal 256 Stufen der Pixelwerte und geometrische Auflösung des Bedeutungsgebirges – beliebig verfeinert werden. Für einen Praxiseinsatz ist ein sorgfältig auf die Objekte abgestimmter Aufbau des Bedeutungsgebirges erforderlich.

Zwei wichtige, in Deutschland angewendete Geoinformationssysteme (GIS) sind die Automatisierte Liegenschaftskarte (ALK) und das Amtliche Topographisch-Kartographische Informationssystem (ATKIS). Bei diesen Systemen sind die folgenden Gliederungseinheiten für den Aufbau des Bedeutungsgebirges zu empfehlen:

- ALK: Folie und Objektschlüssel
- ATKIS: Signaturteilnummer

Wenn die digitalen Kartendaten nichtgeometrische Strukturen besitzen und sich zum Beispiel nach Ebenen oder Layern gliedern lassen, dann könnten auch diese Gliederungen verwendet werden, um unterschiedliche Bedeutungen für die Schriftplatzierung einzuführen.

Bei der Umwandlung der Vektor- in Rasterdaten wird es regelmäßig vorkommen, dass Zeichnungsteile übereinanderliegen und daher Pixel des Bedeutungsgebirges mehrfach bestimmt sind. In diesen Fällen erhält das Pixel den höchsten an dieser Stelle vorkommenden Wert. Es ist nicht zu empfehlen, die Pixelwerte bis zum maximalen Wert aufzusummieren. Bei einer solchen Lösung würden Schnittpunkte immer einen sehr hohen Wert erhalten und unangemessen stark schriftabweisend sein, so dass Schrift bevorzugt an Nicht-Kreuzungsstellen liegt.

Wenn die Strichkarte Rasterelemente enthält oder wenn die zu beschriftende Karte überhaupt nur ein Rasterbild ist, können die Pixelwerte des Bedeutungsgebirges grundsätzlich nach der gleichen Methode festgelegt werden wie bei der Vektorkarte. Die Höhe des Bedeutungsgebirges richtet sich nach den Objektarten. Alle Pixel, die zu einem Rasterelement der gleichen Objektart gehören, erhalten im Bedeutungsgebirge den gleichen Pixelwert.

Dieses Verfahren ist nicht anwendbar, wenn das digitale Bild keine Objektstruktur besitzt, z.B. bei einem digitalen Orthophoto. In diesem Fall kann nur das Bedeutungsgebirge anhand der Grauwerte des digitalen Bildes aufgebaut werden. Es ist zu unterscheiden, ob die Beschriftung mit schwarzen oder mit weißen Buchstaben erfolgen soll. Im ersten Fall, schwarze Buchstaben, bekommen die hellen Bildteile (hohe Grauwerte) niedrige Pixelwerte des Bedeutungsgebirges. Das Bedeutungsgebirge ist gewissermaßen das „Negativ“ zum digitalen Bild. Im zweiten Fall, weiße Buchstaben, bekommen die dunklen Bildteile (niedrige Grauwerte) die niedrigen Pixelwerte. Das Bedeutungsgebirge ist das „Positiv“ zum digitalen Bild.

Wenn als Kartengrundlage Vektor- und Rasterdaten gemeinsam vorliegen, z.B. bei einer Orthophotokarte, müssen die Verfahren zur Ausformung des Bedeutungsgebirges kombiniert angewendet werden.

4.2 Positionsbewertung mittels Bedeutungskessel, Bedeutungstal und Bedeutungsbecken

In der ersten Phase der Schriftplatzierung wird jeder einzelne Schriftzug unter Berücksichtigung bestimmter geometrischer Restriktionen behandelt, als wäre er allein in die Karte zu platzieren. Dafür wird für jeden Schriftzug einzeln eine lokale Geometrie des Bedeutungsgebirges im Bereich der möglichen Positionen ausgeformt. An allen möglichen Positionen des Schriftzuges wird dann die Summe aller an der betreffenden Stelle überdeckten Pixel festgestellt. Die Schrift wird dabei durch eine rechteckige Fläche repräsentiert. Eine niedrige Summe bedeutet eine gute Platzierungseignung. Eine hohe Summe bedeutet eine schlechte Platzierbarkeit wegen auftretender Überdeckungen.

Wäre der Schriftzug allein auf der Karte, dann wäre die Position mit der geringsten Pixelsumme die optimale Stelle für die Schriftplatzierung. Tatsächlich müssen aber auch alle anderen Positionen zusammen mit ihrer Pixelsumme auf einem File zwischengespeichert werden, um als Alternativpositionen bei Überdeckungskonflikten Schrift-Schrift bereitzustehen.

Punkthafte Elemente erhalten ihre Beschriftung in unmittelbarer Nähe, wenn keine Hintergrundzeichnung vorhanden ist, meist rechts oberhalb, sonst dort, wo das Kartenbild Platz lässt. Linienhafte Elemente werden in der Regel entlang und oberhalb der Linie in etwa gleichen Abständen beschriftet. Für Stadtpläne gelten oft Sonderregeln, beispielsweise die Trennung des Straßennamens in zwei Teile. Diese beiden Teile werden dann so an die Straßenenden gesetzt, dass dem Kartenleser die Ausdehnung der Straße durch die Bezeichnung deutlich wird. Flächen erhalten ihre Beschriftung abhängig vom Verhältnis der Schriftgröße zur Flächengröße. Der Namenzug wird entweder innerhalb oder außerhalb der Fläche platziert. Oft wird zur Kennzeichnung der Zuordnung der Namenzug auch so gesetzt, dass die Schrift teilweise innerhalb und teilweise außerhalb steht. Die lokale Geometrie des Bedeutungsgebirges wird für punkthafte Objekte Bedeutungskessel, für linienhafte Objekte Bedeutungstal und für flächenhafte Objekte Bedeutungsbecken genannt.

Die Beschriftung von punkthaften Kartenelementen kann nach unterschiedlichen Gesichtspunkten erfolgen:

- In topographischen Karten wird die Schrift wegen der Konkurrenz zwischen Schrift und Grafik meist dort platziert, wo im Kartenbild am wenigsten verdeckt wird.
- In Karten mit geringer Dichte des Kartenbildes wird die Schrift meist an eine in Bezug auf das punkthafte Objekt bevorzugte Position gesetzt, beispielsweise rechts oberhalb.

Der Bedeutungskessel dient dazu, beide Aspekte zu modellieren. Er ist eine lokale von der Hintergrundkarte unabhängige Pixelmatrix. Diese Pixelmatrix kann wahlweise trichterähnlich oder weitgehend beliebig geformt werden. Für die trichterähnliche Form können in 10-Grad-Schritten 36 „Höhenwerte“ für den Trichterrand angegeben werden. Damit kann der Trichter in einigen Richtungen flacher und in anderen steiler ausgeformt werden. Die flachen Richtungen repräsentieren Vorzugspositionen für die Schriftplatzierung.

Der Bedeutungskessel wird dem Bedeutungsgebirge überlagert, siehe Abbildung 4.2. Die Pixelwerte, deren Summe für die Bewertung der verschiedenen Schriftpositionen maßgeblich sind, gehen aus dieser Überlagerung hervor. Sie sind an jeder Position der größere Pixelwert aus Bedeutungskessel und Bedeutungsgebirge. Bei einer trichterförmigen Geometrie des Bedeutungskessels hängt es von der „Steilheit“ des Trichters ab, ob eher der Kartenhintergrund oder eher die Form des Bedeutungskessels die Schriftposition beeinflussen. Wenn der Bedeutungskessel zu einer flachen Scheibe wird, dann richten sich die Schriftpositionen nur noch nach der zu beschriftenden Karte. Wenn der Bedeutungskessel ein steiler Trichter mit z.B. 45° Neigung ist, dann bleibt der Kartenhintergrund praktisch unbeachtet. Letztere Form könnte man wählen, wenn die Karte nur aus Punktsignaturen besteht.

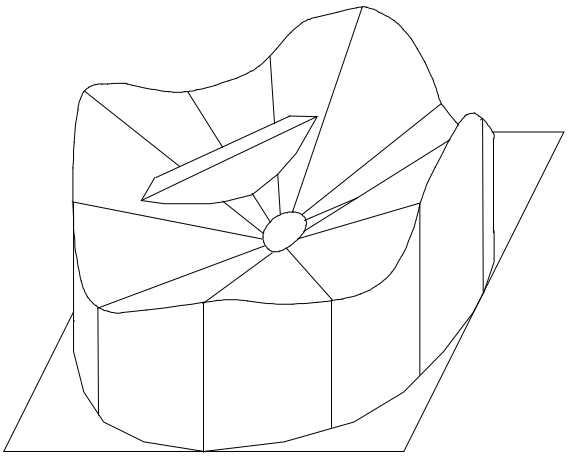


Abbildung 4.2: Überlagerung eines trichterförmigen Bedeutungskessels mit dem Bedeutungsgebirge. Das Bedeutungsgebirge ist hier nur durch eine Linie in der Nordwestecke sichtbar.

Innerhalb des Bedeutungskessels werden Schriftpositionen nur in der Nähe der Punktsignatur gesucht. Diese Nähe wird durch den Rand der Punktsignatur und einen Saum in vorgebbarer Breite definiert. Mindestens ein Pixel des Namenrechtecks muss Anteil an dem Saum oder an dem Rand haben. Kein Pixel des Namen-rechtecks darf aber innerhalb der Fläche der Punktsignatur liegen. Auch punkthafte Signaturen, z.B. Höhenkoten, besitzen Größen von mindestens einem Pixel. Dieses Verfahren gewähr-leistet eine Schriftposition in der Nähe der Punktsignatur, ohne dass die eigentliche Signatur überdeckt wird.

Mit Hilfe der Saumbreite kann je nach Kartentyp die Schrift mehr oder weniger eng an die Signatur gebunden werden. Der um die punktförmige Signatur gelegte Saum gewährleistet auch die eindeutige Zuordnung von Schrift und Signatur. Die Fläche des Saumes wird für alle Schriften außer für die zur Signatur gehörigen gesperrt. Dadurch steht immer die zugehörige Schrift näher an der Signatur als alle anderen Schriften. Die Fläche der Punktsignatur selbst ist sowohl für die zugehörige als auch für alle anderen Schriften gesperrt.

Die Beschriftung von Linien zeichnet sich durch einige spezifische Merkmale aus.

- Die Beschriftung ist vorzugsweise an gestreckten Linienteilen zu platzieren.
- Die Beschriftung folgt der Linienrichtung und liegt daher meist nicht parallel zum unteren Kartenrand oder zum Kartennetz. Die Beschriftung sollte aber immer vom unteren Kartenrand aus lesbar sein und nicht auf dem Kopf stehen [Imh62].
- Die Beschriftung kann oberhalb, unterhalb oder abhängig vom Kartenbild beiderseits der Linien stehen; die bevorzugte Stellung ist oberhalb.
- Die Beschriftung steht nicht unmittelbar an der Linie, sondern hält einen Abstand zu ihr, der eine Minimaldimension nicht unterschreiten darf.
- Längere Linien werden mehrfach in etwa gleichen Abständen beschriftet.

Die Grundlage für die Suche der optimalen Schriftpositionen ist wie bei den punkthaften Kartenelementen eine Kombination aus dem globalen Bedeutungsgebirge und einer linienspezifischen Geometrie. Diese Geometrie wird wegen ihrer Form Bedeutungstal genannt. Dieses Tal folgt dem als bekannt vorausgesetzten Linienverlauf. An gestreckten Linienteilen besitzen die Pixel geringere Werte, das heißt, die Talsohle liegt tiefer. Je stärker die Krümmung der Linie ist, desto größer werden die Pixelwerte und erschweren dort die Schriftplatzierung. Alle Bereiche

außerhalb des Tals werden mit maximalen Pixelwerten belegt und damit für Schrift gesperrt. Die Form des Talquerschnitts bestimmt die Stärke, mit der die Schrift an den vorgegebenen Linienverlauf gebunden wird. Ein V-förmiger Querschnitt bindet die Schrift sehr stark an die vorgegebene Linie. Ein U-förmiger oder ein kastenförmiger Querschnitt belässt eine größere Freiheit, aufgrund des Kartenhintergrundes die beste Position beiderseits der Linie zu suchen, siehe Abbildung 4.3. Ein durch einen Rücken in der Mitte zweigeteiltes Tal (W-förmig) verhindert Beschriftungen auf der Linie selbst und lässt die Festlegung eines Mindestabstandes von der Linie zu.

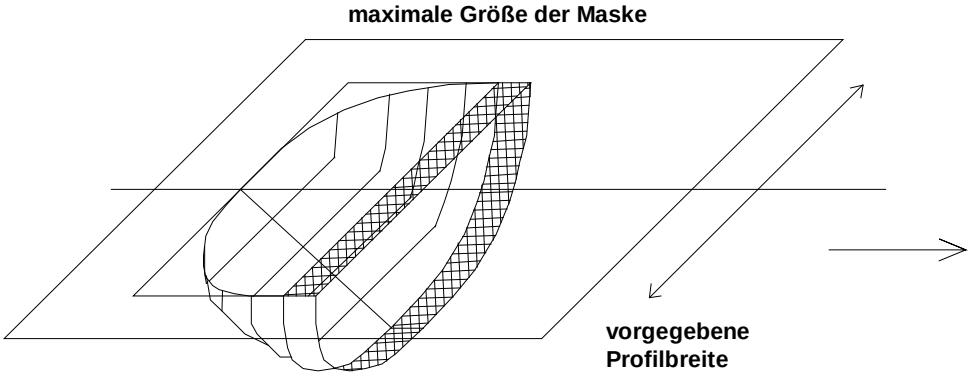


Abbildung 4.3: Maske für die Definition des Bedeutungstales. Das schraffierte Profil wird auf gerader Strecke verwendet. Die weißen Querprofile werden zusätzlich an Linienkrümmungen verwendet.

Die Breite des Bedeutungstales sollte der Signaturbreite angepasst werden. Bei einfachen Linien lässt sich dadurch eine Maximaldistanz zwischen Linie und Schrift vorgeben. Bei Doppellinien wird eine eventuell unerwünschte Schriftposition außerhalb verhindert. Zusätzlich kann bei Doppellinien die Tiefe des Bedeutungstales so abgestimmt werden, dass die in der Karte vorhandene Signatur die Schriftposition maßgeblich beeinflusst. Aus dem vorgegebenen Querprofil wird eine „Schablone“ geformt, die vom Anfang zum Ende der Linie geführt wird und gewissermaßen das Tal ausschürft. Die „Schablone“ ist ein digitaler Filter, der an jeder Pixelposition entlang der Linie dem globalen Bedeutungsgebirge überlagert wird. Bei geradem Linienverlauf wird so ein Querprofil neben das andere gelegt und ein gleichmäßig geformtes Tal gebildet. Bei Linienkrümmungen liegen die Querprofile nicht mehr parallel zueinander. Es entstehen daher Klaffungen, die bei der Bildung der Talform ohne weitere Modellierung undefiniert blieben. Diese Tatsache wird ausgenutzt, um krümmungsabhängig die Talsohle anzuheben. Der digitale Filter enthält nicht nur das vorgegebene Querprofil, sondern auch eine Reihe von daneben liegenden, flacher werdenden Querprofilen, die wie ein Talschluss die Eintiefung des Tals beenden, siehe Abbildung 4.4. Die Form des Filters ist nur „nach hinten“ in Richtung auf das bereits geformte Tal ausgebildet. Bei geradem Tal bleiben diese zusätzlichen Querprofile unberücksichtigt. Erst wenn Klaffungen auftreten, also wenn in einer Linienkrümmung undefinierte Pixel stehenbleiben, werden sie von den zusätzlichen Querprofilen belegt. Je stärker die Krümmung ist, desto breiter werden auch die Klaffungen und desto flachere Querprofile mit hohen Pixelwerten sind am Auffüllen beteiligt. Als Ergebnis wird die Platzierung umso stärker verhindert, je mehr die Linie gekrümmt ist.

Das fertige Bedeutungstal ist die Basis für die Suche der besten Schriftpositionen. Die in drei Schritten ausgeführte Suche hat das Ziel,

- die Stellen entlang der Linie zu finden, an denen der Name wiederholt geschrieben wird,
- in der Umgebung der Wiederholungspunkte diejenigen Stellen zu finden, die die besten Schriftpositionen bieten

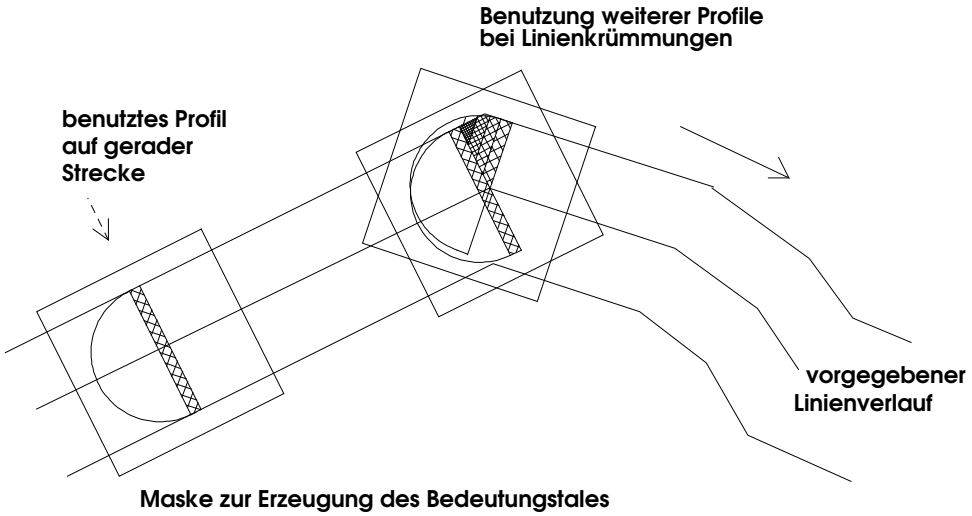


Abbildung 4.4: Verschieben der Maske entlang einer Linie. Auf dem geraden Linienstück wird nur ein Querprofil pro Position für die Talbildung verwendet. An Linienknicken entstehen Klaffungen zwischen den Hauptquerprofilen. Zum Auffüllen der Klaffungen werden weitere Querprofile, die das Tal weniger tief ausbilden, hinzugenommen (dichtere Schraffur).

- den Abstand des ersten Wiederholungspunktes vom Linienanfang zu bestimmen.

Im ersten Schritt wird an jeder Pixelposition entlang der Linie die Wertigkeit für die Platzierung festgestellt. An der betreffenden Linienposition wird die Schrift entlang des Querprofils über die ganze oder halbe Talbreite verschoben und die beste Position gesucht. Diese meist neben der Linie liegende Position und die dort ermittelte Wertigkeit werden für den untersuchten Linienpunkt gespeichert. Dann wird das Verfahren beim nächsten Linienpunkt wiederholt. Am Ende gibt es für jedes Linienpixel eine zugehörige Schriftposition und Wertigkeit, siehe Abbildung 4.5.

Im Standardfall kann die von einem Schriftzug überdeckte Fläche als Rechteck angenommen werden. Wenn sich die Schrift besonders gut der Linie anschmiegen soll, muss an jeder zu prüfenden Position eine kreisbogenförmige Standlinie für die Schrift festgelegt und bei der Bestimmung der Wertigkeit berücksichtigt werden. Im einfachsten Fall kann dieser Kreisbogen durch drei Punkte auf der Linie, nämlich am Beginn, in der Mitte und am Ende des Schriftzuges, festgelegt werden. Die Parameter zur Definition der Standlinie müssen dann für jede Position einzeln zwischengespeichert werden. Eine weitere Verfeinerung des Modells, z.B. hinsichtlich S-förmiger Standlinien, ist denkbar.

Im zweiten Schritt werden die möglichen Verteilungen entlang der Linie ermittelt. Die geometrischen Vorgaben sind der Abstand der Beschriftung und die erlaubte Abweichung von den Sollpositionen. Das Ziel der Suche in diesem Verfahrensschritt ist die Ermittlung der Position der ersten Beschriftung. Die Positionen der anderen Beschriftungen liegen, abgesehen von den erlaubten Verschiebungen um die Sollpositionen, durch den vorgegebenen Abstand fest, siehe Abbildung 4.6.

Zunächst wird das erste Linienpixel als erste Schriftposition angenommen. Die anderen Schriftpositionen ergeben sich aufgrund der vorgegebenen Abstände. Nun wird unter Berücksichtigung der erlaubten Verschiebungsbeträge die minimale Summe aller von Schrift überdeckten Bedeutungspixel ermittelt. Danach wird die Schriftposition um ein Pixel verschoben und das zweite Linienpixel als erste Schriftposition angenommen. Erneut wird die minimale Summe bestimmt. Diese Summierung wird solange wiederholt, bis die erste Schriftposition einen vollen Beschriftungsabstand zurückgelegt hat. Danach würden Konstellationen geprüft, die be-

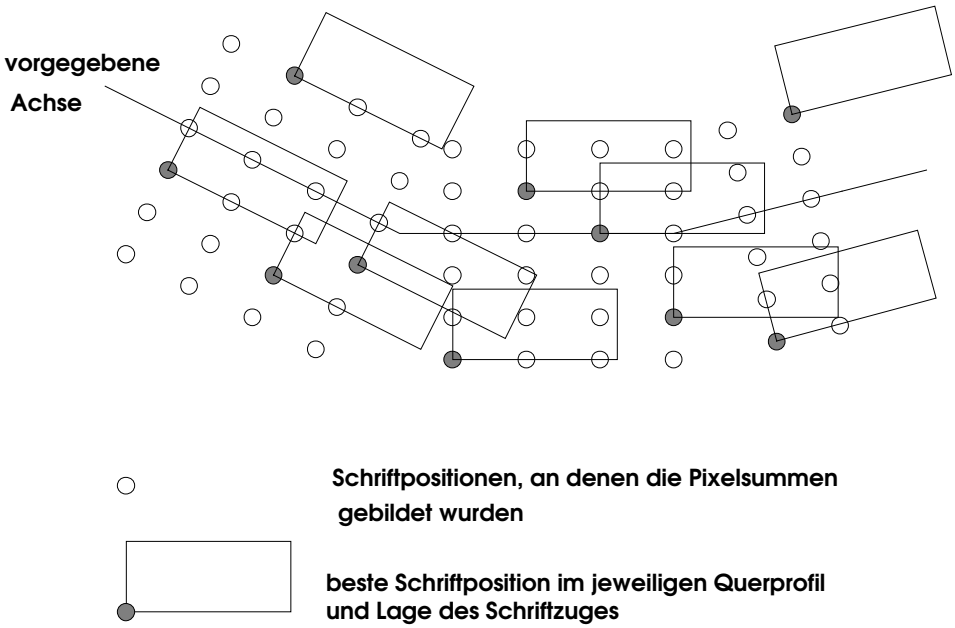


Abbildung 4.5: Positionssuche entlang von Linien, Auswahl der besten Positionen

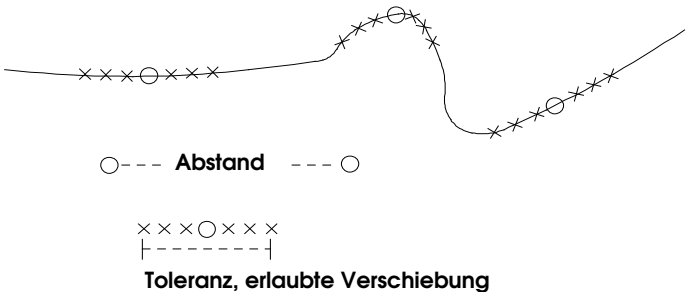


Abbildung 4.6: Schriftabstand und Toleranz entlang einer Linie

reits untersucht wurden, siehe Abbildung 4.7. Aus den gefundenen Summen wird die minimale ermittelt. Diese ergibt die Position der ersten Schrift, also den Abstand vom Linienanfang, und legt damit die Verteilung der Schriften über die ganze Linie fest.

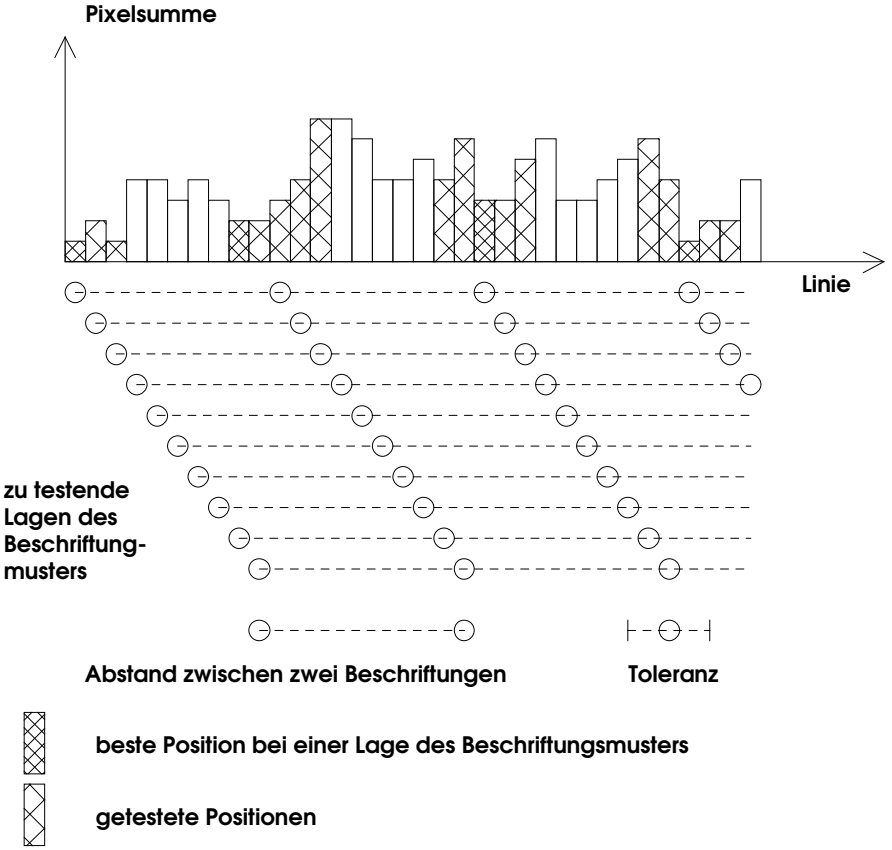


Abbildung 4.7: Prinzip der Positionssuche an Linien, zu testende Startpunkte für die Lage des Beschriftungsmusters.

Im dritten Schritt wird die gefundene optimale Anfangsposition zugrunde gelegt. Für alle nun möglichen Positionen der Schrift werden die Wertigkeit und die Richtung ermittelt.

Die Beschriftungen von Flächen lassen sich grob in folgende Kategorien einteilen:

- Wenn die Fläche in der Karte größer ist als die von der Schrift beanspruchte Fläche, wird die Schrift meist innerhalb der Fläche platziert.
- Wenn die Fläche etwa gleich groß ist wie der Schriftzug; wird die Schrift meist so platziert, dass sie die Fläche teilweise überdeckt.
- Ist die Fläche deutlich kleiner als die Schrift, dann steht die Schrift ähnlich wie bei Punkten neben dem Kartenelement.

Die Bezeichnung „Bedeutungsbecken“ ist von der muldenförmigen Geometrie des lokalen Bedeutungsgebirges abgeleitet. Das lokale Bedeutungsgebirge entsteht wie bei Punkt und Linie durch Überlagerung des globalen Bedeutungsgebirges mit einer vom Benutzer definierten Geometrie. Diese Geometrie kann wie beim Punkt die Form eines Trichters, der bei Flächen eher flach ausfällt, oder die Form eines kleinen „digitalen Höhenmodells“ besitzen. Mit der Form der Oberfläche des lokalen Bedeutungsgebirges lassen sich Bereiche vorgeben, in denen

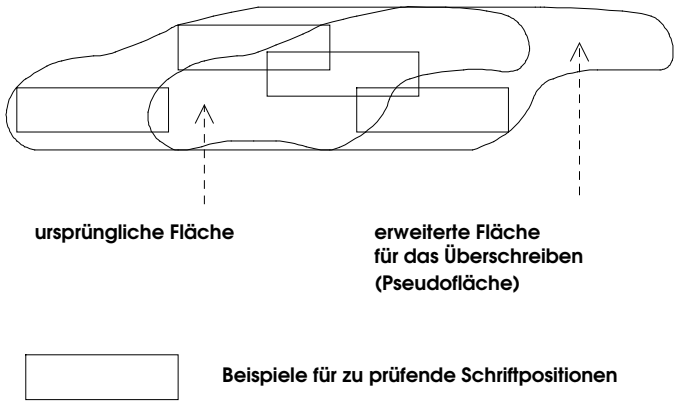


Abbildung 4.8: Flächenvergrößerung zur Steuerung des Überschreibens.

die Schrift vorzugsweise zu platzieren ist, zum Beispiel in der Mitte. Mit der Lage des Randes des lokalen Bedeutungsbeckens wird gesteuert, ob Schrift nur innerhalb der Fläche oder teilweise auch außerhalb stehen darf. Wenn Schrift nur innerhalb stehen soll, dann sind der Rand des lokalen Bedeutungsgebirges und der Flächenrand identisch. Auf dem Flächenrand und außerhalb erhalten die Bedeutungspixel Maximalwerte und verhindern dort eine Platzierung. Wenn die Schrift teilweise innerhalb und teilweise außerhalb liegen soll, wird das Bedeutungsgebirge über den Flächenrand hinaus gerade um soviel erweitert, dass der Schriftzug nie ganz außerhalb der Fläche stehen kann, sondern immer noch ein wenig in die Fläche hineinragen muss. Die entstehende größere Fläche wird Pseudofläche genannt, siehe Abbildung 4.8.

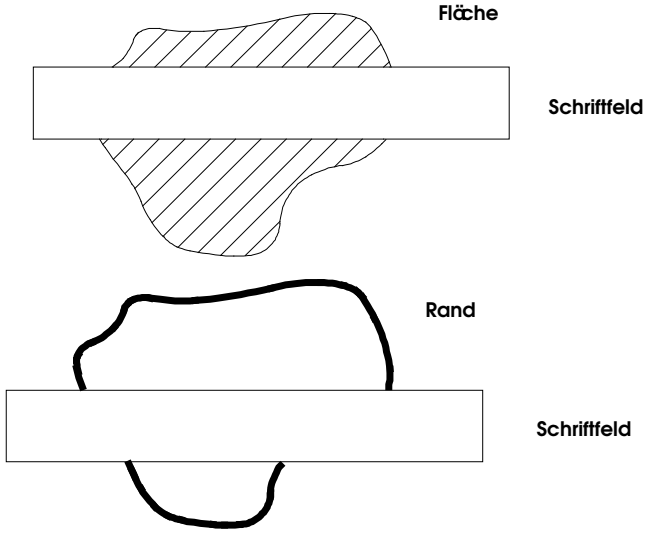


Abbildung 4.9: Maximale Flächen- und Randüberdeckung.

Wenn die Schrift gegenüber der Fläche zu groß wird, kann es passieren, dass soviel vom Flächenrand überdeckt wird, dass die Form der Fläche unkenntlich wird oder dass die ganze Fläche unter der Schrift verschwindet. Um dies zu verhindern, müssen maximale Rand- und Flächenanteile, die überdeckt werden dürfen, vorgegeben werden, siehe Abbildung 4.9.

4.3 Suche nach der optimalen Verteilung aller Namen

In der zweiten Phase der Schriftplatzierung wird die optimale Verteilung des Schriftgutes gesucht. Die Suche ist dann am Ziel, wenn die Summe aller von allen Schriften überdeckten Pixel des Bedeutungsgebirges das Minimum erreicht hat. Diese Suche ist eine flächenhafte Optimierung, bei der theoretisch alle Kombinationen von Positionen nacheinander geprüft werden müssen. Diejenigen Kombinationen, bei denen Überdeckungen auftreten, werden verworfen. Von den verbleibenden wird diejenige gesucht, deren oben angegebene Summe das Minimum erreicht.

Die Anzahl der zu untersuchenden Kombinationen ist in den meisten Fällen sehr hoch. So gibt es beispielsweise bei 100 Namen und nur 10 möglichen Positionen pro Namen schon $10^{1}00$ Kombinationen. Man spricht in einem solchen Fall auch von einer „kombinatorischen Explosion“, die auch auf den schnellsten Rechnern nicht handhabbar ist. Es sind daher Strategien zu entwickeln, die den Suchprozess nachhaltig abkürzen. Insbesondere muss versucht werden, die Rechenschritte bei einer Erhöhung der Anzahl der zu verteilenden Namen nicht exponentiell steigen zu lassen.

Zur Lösung der Optimierungsaufgabe wird das Verfahren der Linearen Programmierung oder Linearen Optimierung eingesetzt, das im Kapitel 2 erläutert wurde. Trotz des Einsatzes der Linearen Programmierung bleibt die Suche nach der besten Verteilung der Schrift ein zeitkritischer Prozess. Daher wird eine zusätzliche Suchstrategie eingesetzt, um die Zahl der in Frage kommenden Kombinationen von Schriftpositionen so klein wie möglich zu halten. Die möglichen Positionen einer einzelnen Schrift lassen sich in zwei Gruppen einteilen: Gruppe 1 beinhaltet die Positionen, an denen Konflikte mit anderen Schriften auftreten können, und Gruppe 2 beinhaltet die Positionen, die nie von einer anderen Schrift gestört werden können.

Daraus folgt, dass auch im ungünstigsten Fall, nämlich bei einer Blockierung aller Positionen der Gruppe 1 durch andere Schriften, der Schriftzug an allen Positionen der Gruppe 2 platziert werden könnte, so auch an der besten Position der Gruppe 2. Daher werden die Schriftpositionen der Gruppe 2 außer der genannten besten im weiteren Optimierungsprozess nicht mehr berücksichtigt.

Kapitel 5

Anzahlmaximierung von Intervallen auf einer Geraden

Katharina Bach

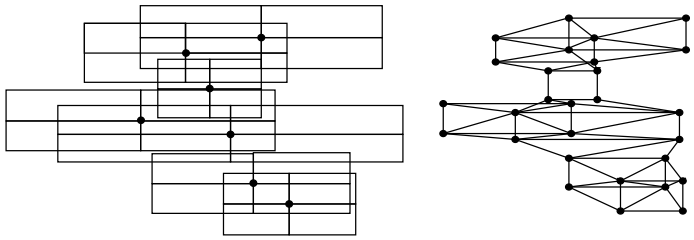
Beim Beschriften einer Landkarte (oder anderer graphischer Darstellungen) steht man vor der Aufgabe, Punkte, die zum Beispiel Städte oder Berggipfel repräsentieren, mit Namen, Höhenangaben oder anderer Information zu beschriften. Dabei sollte die Beschriftung direkt am jeweiligen Punkt liegen, horizontal sein, eine einheitliche Größe haben und sich vor allem nicht mit anderen Beschriftungen schneiden. Die hier gestellte Aufgabe ist nun, eine maximale Anzahl von Punkten zu finden, die beschriftet werden können, ohne dass sich ihre Beschriftungen überschneiden, und diese Punkte dann auch zu beschriften. Dieses Problem ist aber schon für die Beschriftung mit achsenparallelen Einheitsquadraten NP-schwer [FPT81], das heißt, dass man nicht erwarten kann, einen effizienten Algorithmus zu finden, der dieses Problem exakt löst. Es ist jedoch, wie im folgenden gezeigt wird, möglich, einen Faktor-$\frac{1}{2}$-Approximationsalgorithmus hierfür anzugeben, der auch das allgemeinere Problem der Beschriftung mit achsenparallelen Rechtecken gleicher Höhe löst.

Die Lösung eines Beschriftungsproblems kann betrachtet werden als maximale unabhängige Menge (MUM) auf einem Rechtecksschnittgraphen (RSG) $G(V,E)$, in dem jedes Rechteck durch einen Knoten (V) repräsentiert wird und je zwei Knoten durch eine Kante (E) verbunden werden, falls sich die jeweiligen Rechtecke schneiden. Zusätzlich sollen die Rechtecke achsenparallel sein, die gleiche Höhe haben, und eine Ecke jedes Rechtecks soll auf dem dazugehörigen Punkt liegen. Es stehen also für jeden Punkt vier Rechtecke zur Auswahl. Dies ist das sogenannte *4-Positionen-Modell* der Punktbeschriftung, siehe Abbildung 5.1.

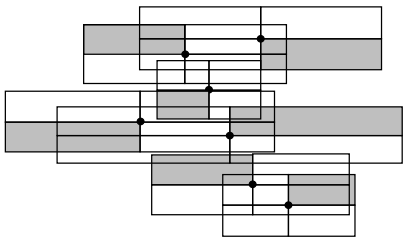
Ein Faktor-$\frac{1}{2}$-Approximationsalgorithmus für eine MUM auf einem RSG lässt sich angeben, wenn man das Problem in der Ebene folgendermaßen in ein eindimensionales Problem umwandelt:

1. Man unterlege die Ebene mit horizontalen Spießen im Abstand der Höhe der Rechtecke. So wird jedes Rechteck von genau einem Spieß durchstochen und kann als Intervall auf dem Spieß gekennzeichnet werden. Gesucht wird nun eine MUM auf diesen Intervallschnittgraphen.

2. Mithilfe eines Algorithmus für MUM auf Intervallschnittgraphen berechne man die maximale Anzahl sich nicht überschneidender Intervalle auf jedem Spieß.



4-Positionen-Modell Rechtecksschnittgraph



Maximale unabhängige Menge

Abbildung 5.1: Eine Beschriftung von Punkten mit Rechtecken im 4-Positionen-Modell, der zugehörige Rechtecksschnittgraph und eine maximale unabhängige Menge.

3. Nun verwandelt man die erhaltenen Intervalle wieder in Rechtecke, indem man jeden zweiten Spieß wählt. So erhält man zwei Mengen von sich nicht überschneidenden Rechtecken, von denen man die größere auswählt.

Man hat auf diese Weise mindestens die Hälfte der maximalen Lösung erwischt, da ja auch die Rechtecke der maximalen Lösung aufgespießt wurden. Falls sie gleichmäßig über gerade und ungerade Spieße verteilt sind, wird mit der beschriebenen Methode die Hälfte von ihnen erfasst. Sind sie ungleichmäßig verteilt, so wird durch die Wahl der Spieße, auf denen die Mehrzahl liegt, eine Ausbeute von über 50% gemacht. Natürlich können nun in den freien Raum zusätzliche Rechtecke gelegt werden, um die Ausbeute zu vergrößern.

Eine andere Möglichkeit, das Ergebnis sicher zu verbessern, besteht darin, MUM von Rechtecken auf zwei benachbarten Spießen zu bestimmen. In [AvKS98] wird dafür ein Algorithmus vorgestellt, der auf dynamischem Programmieren (mehr dazu in Abschnitt 5.2) beruht. Bei der Rückwandlung hat man drei Möglichkeiten der Plazierung und somit einen Faktor von $\frac{2}{3}$. Die Anzahl der benachbarten Spieße läßt sich beliebig erhöhen, so dass z.B. bei vier Spießen bereits ein Faktor von $\frac{4}{5}$ erreicht wird. Allerdings steigt die Rechenzeit exponentiell mit der Anzahl der benachbarten Spieße, für die gemeinsam eine MUM berechnet werden soll [AvKS98].

5.1 Greedy-Algorithmus

Da nun das Verfahren bekannt ist, wie durch Reduzierung und Rückverwandlung des RSG um eine Dimension eine unabhängige Menge gefunden werden kann, die mindestens halb so groß wie eine MUM ist, bleibt die Beschäftigung mit dem Algorithmus, der die entsprechenden Intervalle liefern soll. Der Algorithmus für MUM auf Intervallschnittgraphen (siehe Abbildung 5.2) ist ein einfacher sogenannter Greedy-Algorithmus (greedy, engl.: gefräßig, d.h. was er einmal gespeichert hat, bleibt in der Lösung). Er sortiert die Intervalle nach rechtem Endpunkt in nicht-absteigender Reihenfolge, speichert das Intervall mit dem kleinsten Index in der

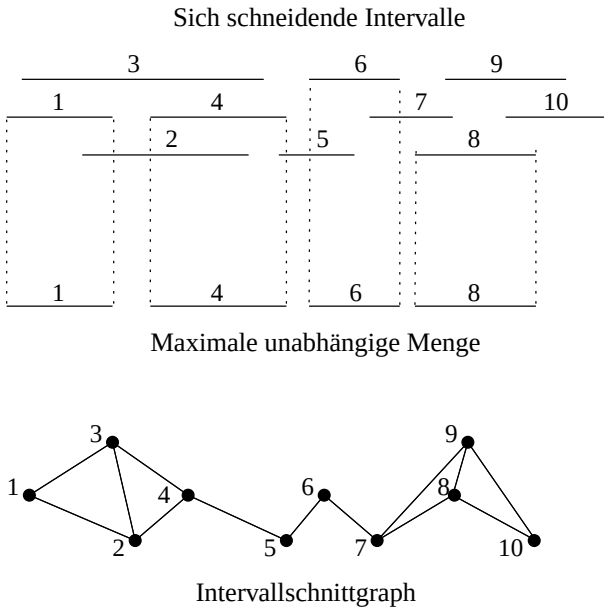


Abbildung 5.2: Eie Menge sich schneidender Intervalle, eine maximale unabhängige Teilmenge und der entsprechende Intervallschnittgraph.

Lösungsmenge I^* und löscht dann alle Intervalle, die das gespeicherte schneiden. Nun speichert er von den übrigen Intervallen wieder das mit dem kleinsten Index (also dessen rechter Eckpunkt am weitesten links liegt) und löscht alle Intervalle, die es schneiden. Dieser Vorgang wird wiederholt, bis alle Intervalle entweder in I^* oder gestrichen sind.

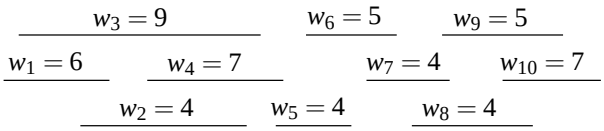
Satz 5.1 *I^* ist unabhängig und maximal.*

Beweis. Da sich die Unabhängigkeit aus der Vorgehensweise direkt ergibt, bleibt die Maximalität zu zeigen. Sei I eine maximale unabhängige Menge und sei $n = |I^*|$. Wir untersuchen beide Mengen von links nach rechts. Aufgrund der Konstruktion von I^* folgt induktiv, dass für $i = 1,\ldots,n$ der rechte Endpunkt des i. Intervalls in I höchstens so weit links liegen kann wie der des i. Intervalls in I. Gälte nun $|I| > n$, so läge das $(n+1)$. Intervall von I aufgrund der Unabhängigkeit von I rechts vom n. Intervall in I, also aufgrund der letzten Feststellung rechts vom letzten Intervall in I^*. Aber dann wäre es nach der oben beschriebenen Konstruktion in I^* aufgenommen worden. ∎

5.2 Gewichtete Intervalle

Es ergibt sich aber nun ein weiteres Problem: In einer MUM werden sich vor allem die relativ kurzen Intervalle befinden, da sie eine größere Kardinalität der Menge ermöglichen, so dass Punkte mit langer Beschriftung, obwohl sie vielleicht wichtig sind, benachteiligt werden. Den Punkten, und somit den entsprechenden Intervallen, soll also nun eine Gewichtung zugeordnet werden, um dann eine unabhängige Teilmenge der Intervalle zu finden, deren Gesamtgewicht maximal ist. So eine Teilmenge nennt man *maximal gewichtete unabhängige Menge* (MGUM).

Der von Hsiao, Tang und Chang hierfür vorgeschlagene Algorithmus [HTC92] basiert auf dem Prinzip des dynamischen Programmierens. Das Problem der maximalen Gewichtung wird also in kleinere Gewichtungsprobleme unterteilt, indem alle Mengen von sich nicht schneidenden



$k = 6$

$S_1 = \{i_1, i_6\}$	$w(S_1) = 6+5 = 11$	$X(6) = \{S_1, S_2, S_3, S_4, S_5\}$
$S_2 = \{i_2, i_6\}$	$w(S_2) = 4+5 = 9$	$\text{MGUM}(6) = S_5$
$S_3 = \{i_3, i_6\}$	$w(S_3) = 9+5 = 14$	$\chi(6) = w(S_5) = 18$
$S_4 = \{i_4, i_6\}$	$w(S_4) = 7+5 = 12$	
$S_5 = \{i_1, i_4, i_6\}$	$w(S_5) = 6+7+5 = 18$	

Abbildung 5.3: Bezeichnungen für den Algorithmus von Hsiao et al.

Intervallen der Ausgangsmenge in Teilprobleme zusammengefasst werden. Durch Erweiterung der Lösungen der Teilprobleme wird dann das ursprüngliche Problem gelöst. Es soll nun also die Art des Zusammenfassens, dann der Zusammenhang zwischen Teillösung und Gesamtlösung und schließlich die Art der Erweiterung festgelegt werden:

Definition 5.1 *Sei $I = \{i_1, i_2, \ldots, i_n\}$ eine Menge gewichteter Intervalle mit $i_k =]a_k, b_k[$ für $1 \leq k \leq n$. Dabei sei a_k der linke, b_k der rechte Endpunkt von i_k und w_k seine Gewichtung. Die Intervalle seien in nicht-absteigender Reihenfolge nach rechtem Endpunkt geordnet. Für eine unabhängige Teilmenge S von I sei $w(S)$ die Summe der Gewichte der Intervalle in S. Die Menge aller unabhängigen Teilmengen S von I mit höchstem Index k sei $X(k)$. Sei MGUM(k) ein maximal gewichtetes Element von $X(k)$ und $\chi(k)$ seine Gewichtung, siehe Abbildung 5.3. Sei $\chi(I)$ die Gewichtung einer MGUM von I.*

Lemma 5.1 $\chi(I) = \max\{\chi(k) \mid 1 \leq k \leq n\}$

Beweis. Geht unmittelbar aus der Definition hervor. □

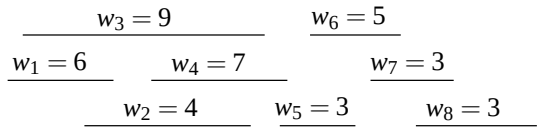
Der Algorithmus soll also die Mengen $X(k)$ bilden und ihre MGUM(k) und $\chi(k)$ speichern, um die am höchsten gewichtete der MGUM(k) zu erhalten. Es gilt dabei, dass sich die maximale Gewichtung von $X(k)$ zusammensetzt aus w_k und der maximalen Gewichtung aller unabhängigen Intervalle links von i_k.

Lemma 5.2 $\chi(k) = w_k + \max\{\chi(l) \mid b_l \leq a_k\}$ *für* $1 \leq k \leq n$

Beweis. 1. Wir zeigen, dass $\chi(k) \geq w_k + \max\{\chi(l) \mid b_l \leq a_k\}$.
Sei $\chi(j) = \max\{\chi(l) \mid b_l \leq a_k\}$ und $S = \text{MGUM}(j) \cup \{i_k\}$. Dann ist S unabhängig und in $X(k)$. Also gilt $\chi(k) \geq w(S) = w_k + \max\{\chi(l) \mid b_l \leq a_k\}$.

2. Wir zeigen, dass $\chi(k) \leq w_k + \max\{\chi(k) \mid b_l \leq a_k\}$.
Sei $N = \text{MGUM}(k) \setminus \{i_k\}$. Da MGUM($k$) unabhängig ist, ist auch N unabhängig. Sei i_j das am weitesten rechts gelegene Intervall von N. Nach Definition von N gilt $b_j \leq a_k$ und daher $\chi(j) \leq \max\{\chi(l) \mid b_l \leq a_k\}$. Andererseits ist die Gewichtung von N höchstens so groß wie die Gewichtung einer MGUM(j), also $w(N) \leq \chi(j)$. Die Gewichtung der MGUM(k) setzt sich zusammen aus w_k und $w(N)$. Also gilt $\chi(k) = w_k + w(N) \leq w_k + \chi(j) \leq w_k + \max\{\chi(l) \mid b_l \leq a_k\}$.

Aus 1. und 2. folgt nun die Behauptung. □



	$\chi(\cdot)$	μ	ν
a_1	$\chi(1) = 6$	0	0
a_3	$\chi(3) = 9$	0	0
a_2	$\chi(2) = 4$	0	0
b_1	$\chi(1) = 6 > \mu$	6	1
a_4	$\chi(4) = 6+7 = 13$	6	1
b_2	$\chi(2) = 4 < \mu$	6	1
b_3	$\chi(3) = 9 > \mu$	9	3
a_5	$\chi(5) = 9+3 = 12$	9	3
b_4	$\chi(4) = 13 > \mu$	13	4
a_6	$\chi(6) = 13+5 = 18$	13	4
b_5	$\chi(5) = 12 < \mu$	13	4
a_7	$\chi(7) = 13+4 = 17$	13	4
b_6	$\chi(6) = 18 > \mu$	18	6
a_8	$\chi(8) = 18+3 = 21$	18	6
b_7	$\chi(7) = 17 < \mu$	18	6
b_8	$\chi(8) = 21 > \mu$	21	8

$\Rightarrow$ MGUM= $\{i_8, i_6, i_4, i_1\}$

Abbildung 5.4: Beispieldurchlauf des Algorithmus von Hsiao et al.

Der hierauf aufbauende Algorithmus von Hsiao et al. benötigt zwei temporäre Variablen und ein Feld. Die Variable μ speichert die Gewichtung der MGUM der vollständig gescannten Intervalle, ν den höchsten Index dieser MGUM und das Feld $\chi()$ speichert die Daten wie oben definiert. Der Algorithmus ordnet die Intervalle in nicht-absteigender Reihenfolge nach rechtem Endpunkt und scannt dann ihre Endpunkte von links nach rechts. Einen Beispieldurchlauf des Algorithmus zeigt Abbildung 5.4. Zu Beginn ist $\mu = 0$ und $\nu = 0$. Wird ein linker Endpunkt erkannt, so speichert der Algorithmus die Gewichtung des zugehörigen Intervalls in der temporären Variablen $\chi(k)$, solange bis ein erster rechter Endpunkt gescannt wird. Die Gewichtung des ersten abgeschlossenen Intervalls wird in μ gespeichert, der Index des Intervalls in ν. Ab hier gibt es zwei Möglichkeiten:

1. Ein linker Endpunkt wird gescannt: Die Gewichtung des zugehörigen Intervalls plus μ wird in $\chi(k)$ gespeichert, da diese Summe ja der zu diesem Zeitpunkt maximalen Lösung entspricht.

2. Ein rechter Endpunkt wird gescannt: Es wird geprüft, ob $\chi(k) > \mu$ oder nicht. Falls ja, wird μ auf $\chi(k)$ und ν auf k gesetzt.

Ist der Vorgang abgeschlossen, kann anhand von μ und ν sowie der Gewichtungen der Intervalle die maximale Lösung folgendermaßen ermittelt werden: Der Index des ersten Elements der Lösungsmenge ist der, der zum höchsten Eintrag in μ gehört. Man zieht seine Gewichtung von dem Wert in μ ab und sucht in der Liste nach der neuen Gewichtung. Der zugehörige Index in ν nennt das zweite Intervall der Lösung, dessen Gewichtung wiederum abgezogen wird. So verfährt man weiter, bis μ den Wert 0 erreicht.

Satz 5.2 *Der Algorithmus von Hsiao et al. liefert in $O(n)$ Zeit eine MGUM von n Intervallen, falls diese nach rechtem Endpunkt sortiert sind.*

Beweis. Nach Lemma 5.1 ist die gesuchte Menge $\max\{\chi(k)|1 \leq k \leq n\}$. Der Algorithmus erzeugt jeweils ein temporäres Maximum, indem er nach Lemma 5.2 zur jeweils zur Zeit maximal gewichteten unabhängigen Menge das Intervall mit dem nächsten linken Endpunkt hinzufügt. Die Unabhängigkeit wird dabei dadurch gewährleistet, dass nach der Methode des Greedy-Algorithmus nur jeweils sich nicht schneidende Intervalle zusammengefasst werden.

Der Algorithmus benötigt lineare Laufzeit, da an jedem Intervallendpunkt nur konstant viele Operationen ausgeführt werden. □

5.3 Ausblick

Die Aufgabe, Punkte auf der Ebene so mit Schrift zu versehen, dass möglichst viele Punkte beschriftet werden, wobei sich die Beschriftung zweier Punkte nicht überlappt, kann auch von diesen Algorithmen nur annähernd gelöst werden. Um auf eine bessere Lösung zu kommen, ist es sinnvoll, die Anforderungen, die an die Plazierung der Beschriftung gestellt werden, zu modifizieren. Das Vier-Positionen-Modell, von dem in diesem Artikel ausgegangen wurde, betrachtet ausschließlich den Fall, dass genau ein Eckpunkt des Beschriftungsrechtecks auf dem zu beschriftenden Punkt liegt. So werden von Anfang an nur vier mögliche Positionen der Beschriftung eines Punktes zugelassen. Eine andere Möglichkeit der Positionierung ist es, jede Plazierung des Rechtecks zuzulassen, bei der sich der Punkt irgendwo auf dem Rand des Rechtecks befindet. Das Rechteck kann also um den Punkt herum beliebig verschoben werden. Solche Beschriftungen nennt man *schiebbar* (sliding labels). Bei ihnen ist die Anzahl der Positionen nicht mehr beschränkt. Zwar hat sich auch dieses Problem als NP-schwer herausgestellt [vKSW99], es wurde aber ein einfacher und effizienter Faktor-$\frac{1}{2}$ Approximationsalgorithmus gefunden [vKSW99]. Im Vergleich mit festen Positionsmodellen produziert er 10-15% mehr Rechtecke, besonders in dichten Gebieten. Wenn also auf die Beschriftung in den klassischen vier Positionen verzichtet werden kann, ist es sinnvoll, diesen Algorithmus anzuwenden. Allerdings ist er schwerer zu implementieren.

Kapitel 6

Punktbeschriftung mit Rechtecken gleicher Höhe

Julia Löcherbach

Die Informationsdarstellung in einer Landkarte, einem Graphen oder einem Diagramm geschieht durch Beschriftung von Punkten, Flächen (Länder, Gebiete) oder Linien (Flüsse, Straßen). Die Beschriftung von Punktmengen tritt in vielen Bereichen auf, in der Geografie zur Bennenung von Städten oder zur Angabe der Höhe von bestimmten Orten, in der Analysis und Statistik zur Durchnummerierung der Punkte in einem Graphen, in der Psychologie und Soziologie zur Beschriftung eines Diagramms. Die Form der Beschriftung ist gegeben durch den (imaginären) Rahmen um den Text als ein achsenparalleles Rechteck (der *Label*).

Es gibt drei grundlegende Anforderungen an eine gute Beschriftung.

1. Um eine eindeutige Zuordnung zu gewährleisten, ist es notwendig, den Label möglichst nah an den entsprechenden Punkt zu platzieren.
2. Zum Verständnis des Dargestellten sollten sich keine zwei Label überlappen und
3. eine lesbare Größe haben.

Darüberhinaus gibt es weitere Anforderungen an eine kartographische Beschriftung von hoher Qualität [Imh75, Yoe72]. Die grundlegenden Anforderungen können in einen Algorithmus umgesetzt werden, der entweder versucht, möglichst viele Punkte zu beschriften oder die größte Schriftgröße zu finden, mit der man alle Punkte beschriften kann. Im Allgemeinen sind diese beiden Probleme NP-vollständig [FW91, MS91]. Im Folgenden wird ein Algorithmus vorgestellt, der möglichst viele Punkte mit Text einer gegebenen Schriftgröße zu beschriften versucht.

Die Anzahl der möglichen Rechteckpositionen für einen Label ist bei den bisher vorgeschlagenen Beschriftungsalgorithmen fast immer eine endliche, feste Zahl. In der gängigsten Variante sind vier Positionen zugelassen, und zwar die, bei der eine der Labelecken den zu beschriftenden Punkt berührt [FW91, WW95, WW97]. Die mögliche Anzahl an Positionen kann eins, zwei, acht oder jede konstante Zahl sein [AvKS97]. Abbildung 6.1 zeigt die zugelassenen Labelpositionen im ein-, zwei- und vier-Positionen-Modell. Bei der Beschriftung mit acht möglichen Positionen berührt der Punkt entweder eine Labelecke oder die Mitte einer Labelseite.

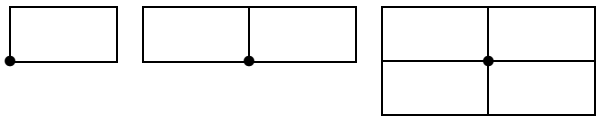


Abbildung 6.1: Das ein-, zwei- und vier-Positionen-Modell.

6.1 Greedy-Algorithmus

In diesem Abschnitt wird ein Algorithmus von van Kreveld, Strijk und Wolff [vKSW99] beschrieben, der eine Punktmenge mit rechteckigen Labeln gleicher Höhe und beliebiger Breite versieht, was der Beschriftung mit Text oder Zahlen gegebener Schriftgröße entspricht. Der Algorithmus verfolgt die Strategie, immer das Rechteck unter den verbleibenden möglichen zu wählen, dessen rechte Kante am weitesten links liegt. Es wird gezeigt, dass diese Strategie mindestens halb so viele Punkte beschriftet wie in einer in diesem Sinne optimalen Lösung. Dafür wird $O(n \log n)$ Zeit und $O(n)$ Speicherplatz benötigt. Der Algorithmus ist *greedy* (englisch: gierig), weil ein einmal zur Lösung hinzugefügtes Rechteck nicht mehr betrachtet wird. Er ist eine zweidimensionale Verallgemeinerung des Greedy-Algorithmus für maximal unabhängige Mengen (MUM) auf Intervallschnittgraphen, siehe Abschnitt 5.1 (Seite 39).

Die Strategie des Algorithmus besteht darin, aus der Menge der Rechtecke der noch nicht beschrifteten Punkte jeweils das *linkeste Reckteck* herauszusuchen, also dasjenige unter den möglichen Rechtecken, dessen rechte Kante am weitesten links liegt (gestricheltes Rechteck in Abbildung 6.2) und es zur Lösung hinzuzufügen. Die Label sind topologisch abgeschlossen, damit automatisch jeder Punkt höchstens ein Label bekommt.

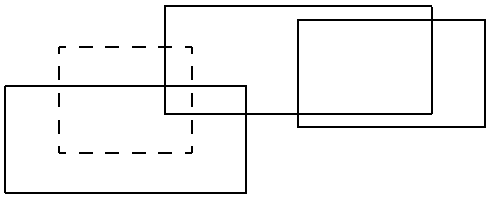


Abbildung 6.2: Beispiel zur Illustration des linkesten Recktecks.

6.2 Approximationsfaktor

Da unsere Zielfunktion die Anzahl der beschrifteten Punkte ist, bezeichnen wir im folgenden eine Beschriftung der gegebenen Punktemenge P als *optimal*, wenn sie unter allen Beschriftungen von P die meisten Punkte beschriftet.

Lemma 6.1 *Die Greedy-Strategie eine Punktmenge mit Labeln fester Höhe und beliebiger Breite durch wiederholte Auswahl des linkesten Rechtecks zu versehen, findet unabhängig vom verwendeten Modell eine Beschriftung, die mindestens halb so groß ist wie die optimale Beschriftung.*

Beweis. Sei L_{opt} die Menge der Label in einer festen optimalen Beschriftung. In L_{left} seien die Label, die vom Greedy-Algorithmus ausgewählt wurden. Wir führen einen Zuschlagsbeweis, um zu zeigen, dass $| L_{\text{left}} | \geq \frac{1}{2} | L_{\text{opt}} |$.

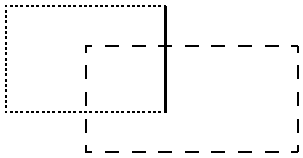


Abbildung 6.3: Label aus L_{left} (gepunktet) und aus L_{opt} (gestrichelt).

Die Menge L_{left} ist maximal, d.h. es kann kein weiterer Label hinzugefügt werden, ohne ein bereits vorhandenes zu schneiden. Deshalb ist jeder Label in L_{opt} entweder in L_{left} oder schneidet eines aus L_{left}, dessen rechte Kante mindestens genauso weit links liegt, siehe Abbildung 6.3.

Jeder Label aus der Differenz $L_{\text{opt}} - L_{\text{left}}$ zahlt einen Zuschlag von einem Dollar an einen Label aus L_{left}, das mindestens so weit links liegt und es schneidet. Und jeder Label aus der Schnittmenge $L_{\text{opt}} \cap L_{\text{left}}$ zahlt einen Dollar an sich selbst.

Insgesamt erhalten alle Label aus L_{left} höchstens zwei Dollar: Die Label aus L_{left} in der Schnittmenge bekommen genau einen Dollar. Jeder andere Label in L_{left} bekommt höchstens zwei Dollar, d.h. wird höchstens von zwei Labeln aus L_{opt} geschnitten. Denn erstens sind je zwei Label aus L_{opt} disjunkt und zweitens muss jeder solche Label, der die rechte Kante eines Labels $l \in L_{\text{left}}$ schneidet, einen der beiden rechten Eckpunkte von l enthalten. In Abbildung 6.4 wird ein Label aus L_{left} von zwei Labeln aus L_{opt} geschnitten und erhält somit zwei Dollar.

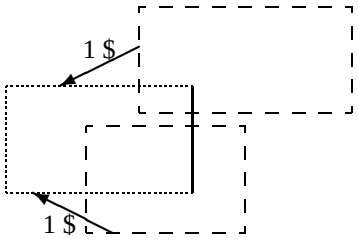


Abbildung 6.4: Label aus L_{left} (gepunktet), das je einen Dollar von zwei Labeln aus L_{opt} (gestrichelt) erhält.

Wenn aber jeder Label aus L_{opt} einen Dollar bezahlt und jeder Label in L_{Left} höchstens zwei Dollar bekommt, dann gibt es offensichtlich höchstens doppelt so viele Label in L_{opt} wie in L_{left}

An keiner Stelle wird ein bestimmtes Modell vorausgesetzt, deshalb gilt der Approximierungsfaktor von $\frac{1}{2}$ für alle Modelle. ◻

6.3 Implementierung und Laufzeitanalyse

Die folgende Beschreibung des Algorithmus bezieht sich auf das vier-Positionen-Modell, mit dem eine Menge von n Punkten beschriftet werden soll. Es sei $L = \{l_1, \ldots, l_m\}$ die Menge der bereits gesetzten Label mit $0 \leq m \leq n$. Am Anfang ist $L = \emptyset$. Die Breite b_i des Labels l_i ist gegeben, die Höhe ist 1 (man kann gegebenenfalls die Skala anpassen). Der *Referenzpunkt* eines Labels ist dessen linke untere Ecke, d.h. zu jedem Punkt gehören vier Referenzpunkte. Nur die vier Referenzpunkte eines Punktes werden im Algorithmus verwendet, der zu beschriftende Punkt ist im Weiteren unwichtig – ein Punkt wird niemals doppelt beschriftet, da die

Label sich nicht berühren dürfen. Somit besteht der Input aus allen Referenzpunkten und den zugehörigen Labelbreiten.

Ist ein Label l_i zur Lösungsmenge L hinzugefügt, kann innerhalb von l_i und in einem dazu kongruenten Rechteck eine Einheit darunter kein Referenzpunkt mehr liegen. Beide Rechtecke zusammen bilden das *erweiterte Rechteck* $\tilde{l}_i$, siehe Abbildung 6.5. Wegen der Greedy-Strategie kann auch links von $\tilde{l}_i$ kein Referenzpunkt mehr liegen.

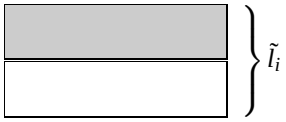


Abbildung 6.5: Das *erweiterte Rechteck* $\tilde{l}_i$ eines Labels besteht aus dem Label selbst (grau) und dessen unterer Kopie (weiß).

Eine naive Implementierung des Greedy-Algorithmus mit Listen und Arrays würde $O(n^3)$ Zeit benötigen. Eine höhere Effizienz erzielt man durch die Verwendung eines *Heaps* und eines *Prioritäts-Suchbaums* als Datenstrukturen, siehe Abschnitt 6.6. Der Heap wird benötigt zur Bestimmung des nächsten Labels, d.h. des *linkesten Recktecks*, denn darin wird die Summe von x-Koordinate und Labelbreite jedes Referenzpunktes gespeichert. Die Wurzel hält das Minimum und damit die x-Koordinate der rechten Kante des *linkesten Recktecks*. Der Prioritäts-Suchbaum wird mit den Referenzpunkten aller möglichen Labelpositionen initialisiert. Er wird zur Aktualisierung des Heap benötigt, d.h. um Referenzpunkte zu finden, die ungültig geworden sind, weil sie innerhalb oder links des erweiterten Rechtecks $\tilde{l}_i$ liegen und entfernt werden müssen.

Solange noch Einträge im Heap sind, werden folgende Schritte wiederholt:

a) Füge den nächsten Label, der durch die Wurzel des Heaps bestimmt wird, zur Lösungsmenge L hinzu.

b) Lösche das Minimum im Heap.

c) Anfrage an den Prioritäts-Suchbaum, welche Referenzpunkte ungültig werden und Löschen der entsprechenden Einträge im Heap und im Prioritäts-Suchbaum.

Am Schluss wird die Lösungsmenge L mit den vom Greedy-Algorithmus ausgewählten Labeln zurückgegeben.

Nun wird untersucht, welche asymptotische Laufzeit der Greedy-Algorithmus mit Hilfe der Datenstrukturen aus Abschnitt 6.6 erreicht.

Satz 6.1 *Gegeben sei eine Menge von n Punkten. Der Greedy-Algorithmus beschriftet mindestens halb so viele Punkte wie eine optimale Beschriftung. Er benötigt dafür* $O(n\log n)$ *Zeit und* $O(n)$ *Speicherplatz.*

Beweis. In Lemma 6.1 wurde gezeigt, dass es einen Approximationsfaktor von $\frac{1}{2}$ für den Greedy-Algorithmus gibt. Folgende Überlegungen führen zur Bestimmung der Laufzeit: Der Heap wird mit n Punkten initialisiert, was $O(n)$ Zeit benötigt. Die Anzahl der Operationen ist deshalb begrenzt durch $O(n)$, da außerdem ungültig gewordene Punkte gelöscht werden. Jede dieser Operationen braucht höchstens $O(\log n)$ Zeit. Die Struktur des Heap kann also in $O(n\log n)$ Zeit aufrecht erhalten werden. Das Gleiche gilt für den Prioritäts-Suchbaum, er wird mit n Punkten initialisiert und es werden später keine hinzugefügt. Anfragen brauchen $O(k+\log n)$ Zeit und die gefundenen k Punkte werden gelöscht und tauchen später nicht mehr auf. Daraus berechnet sich insgesamt eine Aktualisierungszeit von $O(n\log n)$. □

6.4 Andere Modelle

Natürlich ist die Festlegung von vier erlaubten Rechteckpositionen nur eine von vielen Möglichkeiten. Man kann auch nur eine Position zulassen oder acht oder noch mehr. Weiterhin ist es möglich, die Anzahl der Rechteckpositionen nicht zu beschränken, d.h. die Rechtecke müssen nur mit einer Kante den zu beschriftenden Punkt berühren. Solche Modelle mit einem Kontinuum von erlaubten Label-Positionen heißen *Schieber-Modelle*, siehe Abbildung 6.6. Aber auch hier gilt der Approximationsfaktor von $\frac{1}{2}$ für den Greedy-Algorithmus, denn der Beweis von Lemma 6.1 ist unabhängig vom Modell.

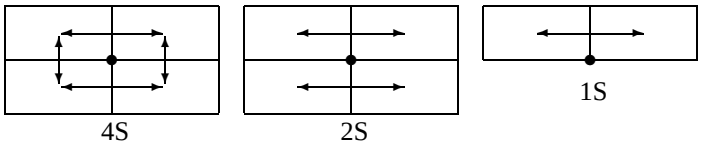


Abbildung 6.6: Vier-Schieber-, zwei-Schieber- und ein-Schieber-Modell (von links nach rechts) mit Pfeilen, die die möglichen Schieberichtungen angeben.

Um auch für Schiebemodelle eine Laufzeit von $O(n \log n)$ zu erreichen, muss der Algorithmus angepasst werden. Einige neue Bezeichnungen werden eingeführt. Die möglichen Positionen für den Referenzpunkt eines Punktes p_i werden beim allgemeinsten vier-Schieber-Modell dargestellt durch vier *Referenzsegmente*, zwei vertikale und zwei horizontale. Referenzpunktpositionen, die nicht mehr möglich sind, werden durch die *rechte Umhüllende* aller erweiterten Rechtecke $\tilde{l}_i$ begrenzt, siehe Abbildung 6.7. Wegen der nun unbeschränkten Anzahl möglicher Referenzpunktpositionen sind mehr Datenstrukturen nötig, um das *linkeste Rechteck* zu finden und diese Strukturen zu aktualisieren. Die Schritte des Algorithmus bleiben im Prinzip unverändert, sie sind nur komplizierter, da mehr Datenstrukturen beteiligt sind. Wieder ist die Laufzeit bei n Punkten durch $O(n \log n)$ begrenzt. Das liegt daran, dass einerseits die Operationen auf die Datenstrukturen $O(\log n)$ bzw. $O(k + \log n)$ Zeit brauchen und andererseits die Anzahl der Operationen durch $O(n)$ begrenzt ist.

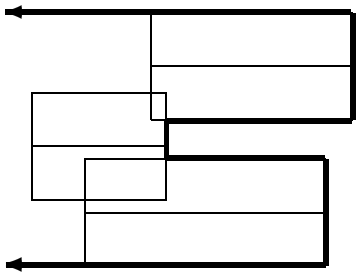


Abbildung 6.7: Die *rechte Umhüllende* dreier erweiterter Rechtecke (verstärkte Linie).

6.5 Vergleich von Modellen

Zwei Modelle werden in ihrer optimalen Lösung unterschiedlich viele Punkte einer Menge von Punkten beschriften, denn durch eine höhere Anzahl von zugelassenen Rechteckpositionen ist die Wahrscheinlichkeit größer, dass ein Punkt beschriftet werden kann.

Das *Verhältnis* zweier Modelle beschreibt, wie viel mehr Punkte mit dem einen als mit dem anderen Modell beschriftet werden können (mit quadratischen Labeln der Größe 1 x 1). Zur Bestimmung des Verhältnisses zweier Modelle, nimmt man eine optimale Beschriftung an für

das Modell M_1 mit der größeren Anzahl an zugelassenen Rechteckpositionen. Dann überführt man M_1 in das eingeschränktere Modell M_2 durch sukzessive Neubeschriftung der in M_1 beschrifteten Punkte, indem man nach und nach jeden Label in eine Position bewegt, die in M_2 zulässig ist. Dadurch entstehende Überlappungen werden eliminiert durch Entfernen der entsprechenden Label aus der Lösungsmenge. Dies macht man solange, bis alle Label aus M_1 behandelt wurden.

Beispiel 6.1 *Die Überführung des zwei-Schieber-Modells in das vier-Positionen-Modell geschieht durch Schieben des Labels nach links oder rechts. Man kann zeigen, dass das Verhältnis* zwei *ist, d.h. theoretisch können doppelt so viele Punkte mit dem zwei-Schieber-Modell beschriftet werden wie mit dem vier-Positionen-Modell.*

Da das Verhältnis für quadratische Label gilt, ist es nur ein Hinweis auf die relativen Unterschiede der Modelle untereinander. Abbildung 6.9 zeigt die gemessenen Werte einer Implementation des Algorithmus in C++ zur Beschriftung von 1000 Städten in den USA, siehe Abbildung 6.8. Die Anzahl der beschrifteten Städte durch das vier-Postionen-Modell (4P) dient als Referenzwert.

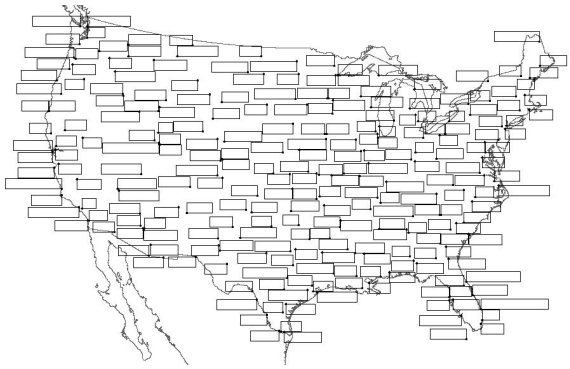


Abbildung 6.8: Beschriftung von 1000 Städten in den USA mit dem vier-Positionen-Modell bei einer Schriftgröße von zehn Punkt (entnommen aus [vKSW99]).

Schrift-	Modell					
größe in pt	1P	2P	4P	1S	2S	4S
5	87	97	100 ≙ 971	101	102	102
6	81	95	100 ≙ 952	101	103	103
7	78	94	100 ≙ 901	103	106	107
8	76	94	100 ≙ 896	102	106	106
9	74	92	100 ≙ 817	102	108	110
10	72	91	100 ≙ 769	102	110	113
11	72	91	100 ≙ 721	101	111	115
12	70	89	100 ≙ 709	101	112	114
13	70	89	100 ≙ 638	101	112	115
14	69	89	100 ≙ 624	102	111	114
15	69	89	100 ≙ 550	101	114	117

Abbildung 6.9: Beschriftung von 1000 Städten in den USA in %, absolute Anzahl angegeben beim 4P-Modell (entnommen aus [vKSW99]).

6.6 Verwendete Datenstrukturen

In diesem Kapitel werden die beiden Datenstrukturen Prioritätssuchbaum und Heap vorgestellt, die bei der Implementation des Greedy-Algorithmus für das vier-Positionen-Modell verwendet werden. Diese werden benötigt, um eine Laufzeit von $O(n \log n)$ zu erreichen.

In einem Prioritäts-Suchbaum [McC85] werden die Koordinaten von Punkten gespeichert. Die Struktur des Prioritäts-Suchbaums erlaubt eine Anfrage mit den Koordinaten einer vertikalen Strecke $\{x_{\max}\} \times [y_{\min}, y_{\max}]$, die alle Punkte liefert, die links davon liegen, siehe Abbildung 6.10. Der Baum benötigt bei n Punkten $O(n)$ Platz, eine Anfrage $O(k + \log n)$ Zeit, wobei k die Anzahl der gefundenen Punkte ist.

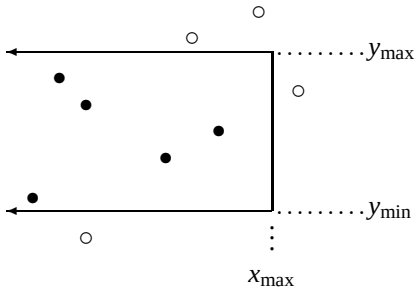


Abbildung 6.10: Eine Anfrage mit x-Koordinate $x_{\max}$ und y-Intervall $[y_{\min}, y_{\max}]$ an den Prioritäts-Suchbaum liefert alle ausgefüllten Punkte, also Punkte (x,y) mit $x \leq x_{\max}$ und $y_{\min} \leq y \leq y_{\max}$.

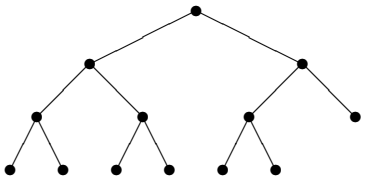


Abbildung 6.11: Ein nicht ganz gefüllter binärer Suchbaum mit 13 Knoten und vier Ebenen ($\log_2 13 < 4$).

Ein Heap ist ein Array, der als vollständiger binärer Suchbaum aufgefasst werden kann [CLR96]. Der Suchbaum heißt binär, da jeder Knoten höchstens zwei Kinder hat, sofern er Kinder hat, siehe Abbildung 6.11. Er wird von links nach rechts und von oben nach unten aufgefüllt. Da alle Ebenen – bis auf die letzte möglicherweise – vollständig gefüllt sind, heißt der Baum vollständig. Die Arraydarstellung entspricht der des Baumes insofern als die Reihenfolge beibehalten wird, d.h. der i-te Knoten im Baum hat den Index i im Array, die Wurzel z.B. hat den Index 1. Mit dem Index i eines Knoten kann man auch den Vorgänger und die Nachfolger bestimmen. Der vorhergehende Knoten hat den Index $\frac{i}{2}$, der linke nachfolgende Knoten $2i$ und der rechte nachfolgende Knoten $2i+1$. Das reduziert den Speicherplatz, da man für die Suche keine zusätzlichen Zeiger auf den Elter und/oder die Kinder braucht. Der Speicherplatz ist begrenzt durch $O(n)$ bei n Knoten. Der Zeitaufwand für die Basis-Operationen zum Erhalt der Struktur wie Einfügen und Löschen von Knoten, Entfernen des Minimums an der Wurzel (extract-min), sind durch die Höhe des Baumes begrenzt, also durch $O(\log n)$. Somit braucht der Sortieralgorithmus *Heapsort*, der aus einem beliebigen Array einen Heap erstellt, insgesamt $O(n \log n)$ Zeit.

Kapitel 7

Die Beschriftung von Punkten mit verschieden großen Kreisen

Kristina Hanig

Stellen wir uns vor, wir sollen auf einer Landkarte (z.B. der Deutschlandkarte) die Städte, die durch Punkte gekennzeichnet sind, mit ihren Namen beschriften. In einigen Gebieten Deutschlands liegen sehr viele Städte nebeneinander. Wie beschriftet man diese, so dass sich keine Namen überschneiden?

Ein ähnliches Problem ist die Beschriftung von Wetterkarten. Einzelne Regionen des Landes erhalten Symbole für Sonne, Regen, Wolken, usw.. Diese Symbole können durch Kreise dargestellt werden. Und auch diese müssen wieder so angeordnet sein, dass sie sich nicht überschneiden.

Wir formulieren nun das Problem *Wetterkartenbeschriftung* wie folgt. Sei eine Menge M von n Punkten $p_1, \ldots, p_n$ mit Labelgrößen $d_1, \ldots, d_n$ gegeben. Gesucht ist eine Beschriftung von möglichst vielen Punkten mit disjunkten Kreisen der gegebenen Größe in jeweils einer der folgenden Positionen:

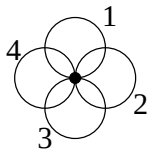


Abbildung 7.1: 4-Positionen-Modell für die Beschriftung von Punkten mit Kreisen.

Als Lösung möchte man ein n-Tupel $(b_1, \ldots, b_n)$ über der Menge $\{0,1,2,3,4\}$ erhalten, bei dem möglichst viele der b_i verschieden von 0 sind, wobei $b_i = 0$ bedeutet, dass der Punkt p_i nicht beschriftet wird. Wir betrachten die Kreise als topologisch abgeschlossen, damit sich je zwei der vier Beschriftungskandidaten eines Punktes schneiden.

Das Wetterkartenbeschriftungsproblem führt uns auf das allgemeinere Problem der maximal unabhängigen Menge auf Kreisgraphen, bei dem wir nun die zu beschriftenden Punkte nicht mehr betrachten müssen.

7.1 Mathematische Problemdefinition

Gegeben sei eine Menge von Kreisen $\mathcal{D} = \{D_1, \ldots, D_n\}$ in der Ebene mit Durchmessern d_i und Mittelpunkten $c_i = (x_i, y_i)$. Gesucht ist eine maximal unabhängige Menge (MUM) $\mathcal{D}' = \text{MUM}(\mathcal{D})$ von $\mathcal{D}$, d.h. $\mathcal{D}' \subseteq \mathcal{D}$ mit $D_i \cap D_j = \emptyset$ für alle $D_i \neq D_j \in \mathcal{D}'$ und $|\mathcal{D}'|$ ist maximal. Zwei Kreise D_i und D_j schneiden sich, wenn für den euklidischen Abstand ihrer Mittelpunkte gilt $\text{dist}(c_i, c_j) \leq \frac{d_i + d_j}{2}$. Diese Schnittinformation wird im *Kreis-Graphen* gespeichert, den wir nun definieren. Zur Illustration siehe Abbildung 7.2.

Definition 7.1 (Kreis-Graph) *Ein Kreis-Graph ist der Schnittgraph einer Menge von verschieden großen Kreisen in der Ebene, d.h. jeder Knoten entspricht einem Kreis, und es existiert eine Kante zwischen zwei Knoten, wenn sich die zwei zugehörigen Kreise schneiden.*

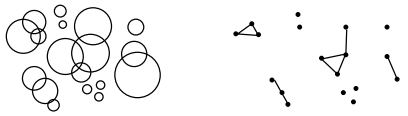


Abbildung 7.2: Eine Menge von Kreisen und der zugehörige Kreis-Graph.

MUM ist ein NP-schweres Optimierungsproblem, sogar falls alle Kreise gleich groß sind. Man kann also nicht erwarten, daß ein effizienter Algorithmus existiert. Der bisher beste Approximationsalgorithmus für MUM auf Schnittgraphen von Kreisen beliebigen Durchmessers erreichte einen Approximationsfaktor von fünf [MBH+95]. Dies ist schon ein erstaunlich gutes Resultat, denn für MUM auf allgemeinen Graphen (gegeben $G(V, E)$, finde $V' \subseteq V$ mit $|V'|$ maximal und $(v, w) \notin E$ für alle $v, w \in V'$) hat der beste bekannte Algorithmus nur einen Approximationsfaktor von $O(n^{1-\varepsilon})$ (siehe [EJS01]). Wir stellen hier ein Approximationsschema vor, mit dem man theoretisch beliebig nah an die optimale Lösung herankommt.

Definition 7.2 (PTAS für MUM auf Kreis-Graphen) *Eine Schar von Algorithmen A_ε heißt Polynomialzeit-Approximationsschema (PTAS) für* MUM, *falls für jedes $\varepsilon > 0$ der Algorithmus A_ε in jeder Menge $\mathcal{D}$ von n Kreisen in Polynomialzeit eine unabhängige Menge mit mindestens $\frac{1}{1+\varepsilon} |\text{MUM}(\mathcal{D})|$ Kreisen findet.*

Ein solches Schema mit polynomialen Aufwand haben Erlebach, Jansen und Seidel in [EJS01] präsentiert.

7.2 Ein PTAS für MUM auf Kreis-Graphen

Der Algorithmus arbeitet mit dynamischer Programmierung, d.h. das Problem wird in Teilprobleme desselben Typs zerlegt, die aber kleiner und deshalb leichter zu lösen sind. Aus den Ergebnissen der Teilprobleme wird dann die Lösung des Gesamtproblems zusammengesetzt. Hierfür werden die Kreise nach unterschiedlichen Größenordnungen sortiert und die Ebene nach den gleichen Größenordnungen in Quadrate zerlegt, wobei jedes größere Quadrat kleinere enthält. Es werden nur noch die Kreise betrachtet, die die Quadrate der gleichen Größenordnung nicht schneiden. Für jede unabhängige Menge großer Kreise wird eine maximale unabhängige Menge kleinerer Kreise bestimmt, die die großen Kreise nicht schneiden.

Konkret geht das folgendermaßen: Sei $k > 1$ fest. Die Kreise werden so skaliert, dass $d = 1$ der größte Kreis und $d_{\min}$ der Durchmesser des kleinsten Kreises ist. Die Menge der Kreise wird in $(l+1)$ verschiedene Levels unterteilt, wobei $l = \left\lfloor \log_{k+1} \frac{1}{d_{\min}} \right\rfloor$.

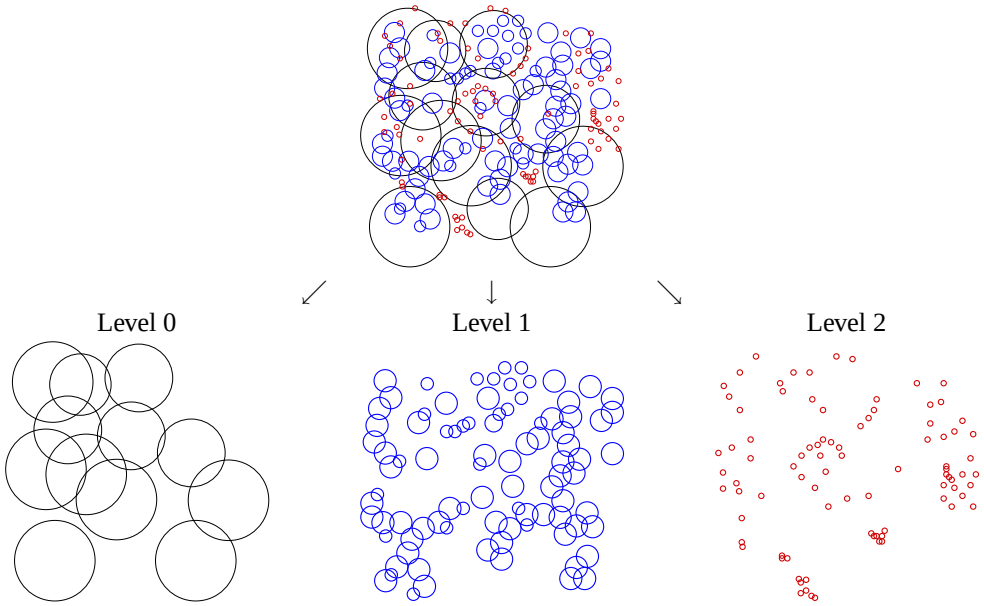


Abbildung 7.3: Einteilung der Kreise in Levels (entnommen aus [EJS01]).

Der Level j enthält alle D_i mit Durchmesser $(k+1)^{-j} \geq d_i > (k+1)^{-(j+1)}$, wobei $D_i \in \mathcal{D}$ und $0 \leq j \leq l$, siehe Abbildung 7.3. Für jeden Level j mit $0 \leq j \leq l$ wird ein Netz in der Ebene erstellt, welches die Geraden enthält, die $(k+1)^{-j}$ voneinander entfernt sind (vergleiche Abbildung 7.4). Die v-te vertikale Gerade ist bei $x = v(k+1)^{-j}$ und vom Index v; die h-te horizontale Gerade ist bei $y = h(k+1)^{-j}$ und vom Index h.

Definition 7.3 *Seien $r \geq 0$ und $s < k$. Die r-ten vertikalen Geraden mit $r \equiv v \mod k$ und die s-ten horizontalen Geraden mit $s \equiv h \mod k$ heißen* aktiv für (r,s).

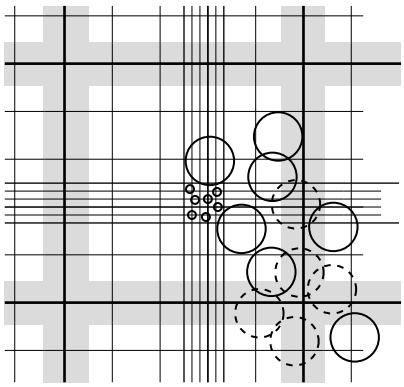


Abbildung 7.4: Beispiel für die Unterteilung der Ebene mit $k = 5$. Die großen Kreise sind vom Level j, die kleinen vom Level $j+1$. Aktive Geraden sind fett gezeichnet, Kreise, die diese Geraden schneiden, sind gestrichelt gezeichnet. Ein Kreis vom Level j schneidet nur eine aktive Gerade, wenn sein Mittelpunkt im grau markierten Bereich liegt. (Entnommen aus [EJS01].)

Sei $\mathcal{D}(r,s)$ die Menge aller Kreise, die durch das Löschen der Kreise entsteht, die eine für (r,s) aktive Gerade vom gleichen Level schneiden.

Der Algorithmus betrachtet alle k^2 möglichen Werte für r und s, so dass $0 \leq r,s < k$. Für jede Möglichkeit wird eine maximale unabhängige Menge in $\mathcal{D}(r,s)$ berechnet.

Wir betrachten jetzt das Problem auf dem Level j und fixieren (r,s). Ein Quadrat, das von jeweils zwei aufeinanderfolgenden aktiven Geraden berandet wird, nennt man *j-Quadrat*. Aufgrund unserer Definition von aktiven Geraden gilt:

Lemma 7.1 *Für $0 \leq j < l$ ist jedes $(j+1)$-Quadrat völlig in einem j-Quadrat enthalten.*

Lemma 7.2 *Ein einfaches Packungsargument liefert, dass die Anzahl der disjunkten Kreise vom Level höchstens j, die ein j-Quadrat schneiden, durch eine Konstante C beschränkt ist.*

Wir betrachten nur *relevante j-Quadrate*, d.h. jene Quadrate, für die $\mathcal{D}(r,s)$ mindestens einen Kreis vom Level j aus einem j-Quadrat S enthält. Ein j'-Quadrat S' wird *Kind* von S genannt, wenn $S' \subseteq S$, $j' > j$ und es kein relevantes j''-Quadrat S'' gibt, so dass $S' \subseteq S'' \subseteq S$ und $j' > j'' > j$. Während der Algorithmus ein relevantes j-Quadrat S verarbeitet, wird eine Tabelle T_S berechnet. In der Tabelle stehen Paare von Mengen. Die erste Menge M_1 enthält disjunkte Kreise vom Level höchstens j, während die zweite Menge $M_2 = T_S(M_1)$ die dazugehörige maximale unabhängige Menge disjunkter Kreise in S vom Level mindestens j ist. Für jede Menge M disjunkter Kreise vom Level höchstens j wird also die größte Menge M' zur Erweiterung von M durch Hinzunahme von Kreisen größeren Levels in S berechnet. Dies erfolgt durch Nachschauen in den Tabellen $T_{S'}$ nach den zugehörigen gespeicherten Mengen, wobei alle Kinder S' von S durchlaufen werden. Wurde die Tabelle für alle relevanten Quadrate berechnet, bildet man die Vereinigung aller Mengen $T_S(\emptyset)$, das ist gerade die zweite Menge, die zur leeren Menge gehört, für alle relevanten Quadrate S, die keinen Vater haben. Das Ergebnis ist die maximale unabhängige Menge in $\mathcal{D}(r,s)$, die in polynomialer Zeit berechnet wird.

Lemma 7.3 *Für ein festes $k > 0$ ist die Laufzeit dieses Algorithmus polynomial.*

Beweis. Bei gegebenen n Kreisen existieren höchstens n relevante Quadrate, die in polynomialer Zeit berechenbar sind. Für jedes dieser relevanten Quadrate werden $O(n^C)$ Mengen berechnet (siehe Lemma 7.2) und für jede dieser Mengen $O(n)$-mal auf die Kinder von S zugegriffen. Da der Algorithmus für k^2 mögliche Werte von r und s läuft, erhält man einen Aufwand von $O(k^2 n^{C+1})$. □

Zusammenfassend gilt:

Satz 7.1 *Es existiert ein PTAS für das* MUM*-Problem in Kreis-Graphen bei gegebener Kreisdarstellung.*

7.3 Ausblick

Es wurde ein PTAS für das MUM-Problem in Schnittgraphen von verschieden großen Kreisen dargestellt, das für jede natürliche Zahl k einen Approximationsfaktor von $(1+\frac{1}{k^2})$ hat und damit die bisher bestehenden Resultate deutlich verbessert. Dabei wurden keine speziellen Eigenschaften der Kreise ausgenutzt. Demzufolge ist das PTAS auch auf andere geometrische Objekte, wie z.B. Quadrate oder reguläre Polygone anwendbar. Für Rechtecke ist es auch nutzbar, wenn sich das Verhältnis zwischen Breite und Länge nicht mehr als um einen konstanten Faktor ändert. Das Problem MUM kann verallgemeinert werden, indem man eine Gewichtung der Kreise einführt. Das in [EJS01] beschriebene PTAS löst die gewichtete Version dieses Problems. Um die Darstellung zu vereinfachen, haben wir hier nur die ungewichtete Version

vorgestellt. Der einzige Unterschied im Algorithmus besteht darin, dass bei der Berechnung der Tabelleneinträge beim dynamischen Programmieren das Gewicht der Kreise statt derer Anzahl als Zielfunktion verwendet wird. Die grundlegenden Ideen sind ansonsten die gleichen wie die hier vorgestellten.

Unter bestimmten Bedingungen kann man auch PTAS für MUM auf Schnittgraphen von geometrischen Objekten im d-dimensionalen Raum erhalten.

Es ist noch nicht sicher, ob eine Erweiterung auf Schnittgraphen von Rechtecken mit beliebigem Verhältnis von Länge und Breite möglich ist. Es existiert bisher nur ein Approximationsalgorithmus mit einem Approximationsfaktor von $O(\log n)$ [AvKS98]. Ein Algorithmus mit konstantem Approximationsfaktor (also unabhängig von n) oder gar ein PTAS für diese Fragestellung ist ein schwieriges offenes Problem.

Kapitel 8

Punktbeschriftung mit gleichförmigen Quadrattripeln

Paul Rosenthal und Steffen Rudnick

Die Beschriftung von Punkten auf Karten, geometrischen Zeichnungen oder grafischen Auswertungen verschiedener Sachverhalte ist sehr zeitaufwendig. Vor allem, wenn es sich dabei um die Beschriftung von vielen Punkten handelt, ist dies ohne Algorithmen, die dann von Computern abgearbeitet werden können, kaum realisierbar.

Dabei verfolgt man zwei verschiedene Ziele, zum einen die Maximierung der Anzahl der beschrifteten Punkte bei vorgegebener Beschriftungsgröße und zum anderen die Maximierung der Beschriftungsgröße unter der Vorgabe, dass alle Punkte beschriftet werden sollen. In diesem Kapitel beschäftigen wir uns nur mit der Größenmaximierung. Zu dieser Zielfunktion gibt es schon Algorithmen, die Punkte mit einem [FW91] oder zwei [QWXZ00] Quadraten sowie mit einem [DMM02] oder zwei Kreisen [WTX02] beschriften.

Die Beschriftung von Wetterkarten mit Temperatur, Wettersymbol und Stadtname oder ähnlicher Karten kann dadurch modelliert werden, dass jeder Punkt mit drei achsenparallelen Quadraten beschriftet wird. Dieses Problem wird durch einen Algorithmus von Duncan, Quian und Zhu [DQZ01] gelöst, den wir im folgenden vorstellen möchten. Er berechnet zu n gegebenen Punkten in der Ebene die größtmögliche Beschriftungsgröße (Kantenlänge der Quadrate) $s_{\max}$, derart, dass sich keine zwei Quadrate schneiden und jedes Quadrat den Punkt, den es beschriften soll, mit einer seiner vier Ecken berührt. Weiterhin findet er auch eine solche Beschriftung der n Punkte mit $3n$ Quadraten der Kantenlänge $s_{\max}$. Die Zeitkomplexität dieses Algorithmus' beträgt $O(n \log n)$.

Wir betrachten die folgenden beiden Teilprobleme:

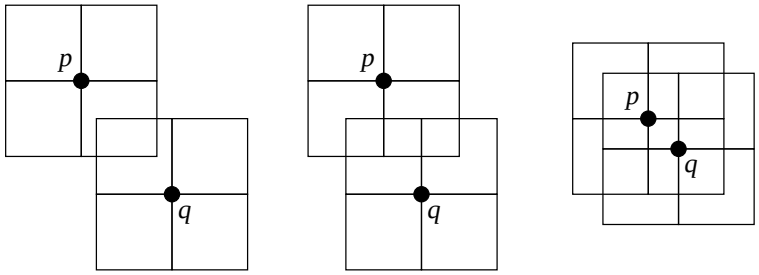
Entscheidungsproblem: Man muß von einer gegebenen Beschriftungsgröße s entscheiden können, ob eine Beschriftung mit Quadraten der Kantenlänge s möglich ist.

Suche nach der größtmöglichen Beschriftungsgröße: Man benötigt eine möglichst kurze und effizient berechenbare Liste von möglichen Beschriftungsgrößen, die auch die maximale Beschriftungsgröße $s_{\max}$ enthält. Nach dem Sortieren dieser Liste kann man dann mittels binärer Suche und jeweiligem Lösen des Entscheidungsproblems für die einzelnen Elemente $s_{\max}$ finden.

8.1 Entscheidungsproblem

Es gilt nun festzustellen, ob das Entscheidungsproblem für eine gegebene Beschriftungsgröße s lösbar ist. Das heißt alle Eingabepunkte müssen mit je drei Quadraten der Kantenlänge s überschneidungsfrei beschriftet werden können.

Definition 8.1 (Multigraph $G_s(P,E_s)$) *Seien an jeden Punkt $p \in P$ alle vier möglichen Label mit Kantenlänge s gesetzt. Dann entspricht jedem Punkt p ein Knoten des Graphen. Eine Kante (p,q) mit $p,q \in P$ existiert gdw. sich genau ein Label von p mit einem von q überschneidet, siehe Typ 1 in Abbildung 8.1. Zwei Kanten (p,q) existieren gdw. zwei Label von p Überschneidungen mit Labeln von q haben, siehe Typ 2 in Abbildung 8.1.*



Typ 1 Typ 2 Typ 3

Abbildung 8.1: Überschneidungstypen

Um das Entscheidungsproblem für eine Beschriftungsgröße s zu lösen, wird der Multigraph $G_s(P,E_s)$ berechnet. Sobald sich alle vier Label von mindestens zwei Punkten überschneiden, ist keine Beschriftung mit s möglich, siehe Typ 3 in Abbildung 8.1, und der Aufbau des Graphen kann abgebrochen werden.

Wir wollen nun die Lösbarkeit des Entscheidungsproblems bei der Punktbeschriftung graphentheoretisch charakterisieren, um es dann mit gängigen Graphenalgorithmen zu lösen. Dazu benötigen wir den folgenden Grundbegriff aus der Graphentheorie.

Definition 8.2 (Kreis) *Ein* Kreis *K in einem Graphen ist eine Folge von Knoten $p_1, p_2, \ldots, p_k$ mit $p_1 = p_k$, die für $i = 1, \ldots, k-1$ durch Kanten (p_i, p_{i+1}) verbunden sind.*

Seien die Punkte $p_1, \ldots, p_k$ eines Kreises im Graphen $G_s(P,E_s)$ mit je drei Quadraten beschriftet und somit je Punkt drei Label gesetzt. Wir charakterisieren nun eine spezielle Form solch einer Beschriftung, um im anschließenden Lemma allgemeine Aussagen über die Beschriftung eines Kreises zu treffen.

Definition 8.3 *Bilden die Punkte $p_1, \ldots, p_k$ einen Kreis im Graphen $G_s(P,E_s)$, so heißen die Label dieser Punkte* zyklisch angeordnet *genau dann, wenn für alle Punkte p_i gilt, p_i hat genau die drei Label gesetzt, die keine Überschneidung mit den gesetzten Labeln von p_t haben, wobei $t = i+1 \mod k$.*

Lemma 8.1 *Um die Punkte p_i des Kreises K beschriften zu können, müssen die Label aller p_i zyklisch angeordnet werden. Solch eine Anordnung existiert immer.*

Beweis. Angenommen die Label werden nicht zyklisch angeordnet. Dann existiert ein $p_i \in K$ mit der Eigenschaft

- p_{i-1} setzt einen Label, der einen von p_i schneidet, und
- p_{i+1} setzt einen Label, der einen von p_i schneidet.

Daraus folgt, einer der drei gesetzten Label von p_i schneidet einen gesetzten Label von p_{i-1} oder p_{i+1}. Somit ist die Beschriftung nicht gültig. □

Damit wir auch allgemeine Aussagen über die Beschriftung von Punkten außerhalb von Kreisen machen können, charakterisieren wir diese nun.

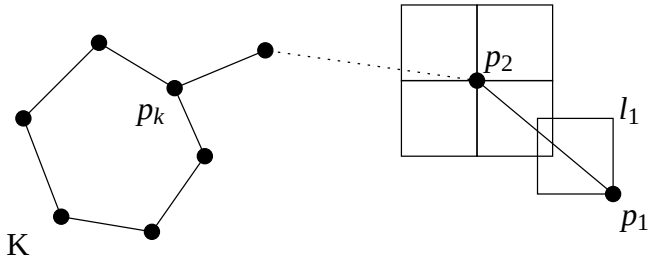


Abbildung 8.2: Der Label l_1 ist K zugewandt

Definition 8.4 *Ein Label l_1 von p_1 heißt* einem Kreis K zugewandt, *gdw.*

- *$p_1 \notin K'$ für alle Kreise K'*
- *es gibt einen Pfad $(p_1, \ldots, p_k)$ mit $p_k \in K$ und $(p_i, p_{i+1}) \in E_s$ für $i = 1, \ldots, k-1$*
- *l_1 schneidet einen möglichen Label von p_2.*

In Abbildung 8.3 sieht man eine Beschriftung einer Punktmenge mit eingezeichnetem Graphen. Man erkennt auch, dass sich höchstens ein Kreis in jeder Zusammenhangskomponente befindet.

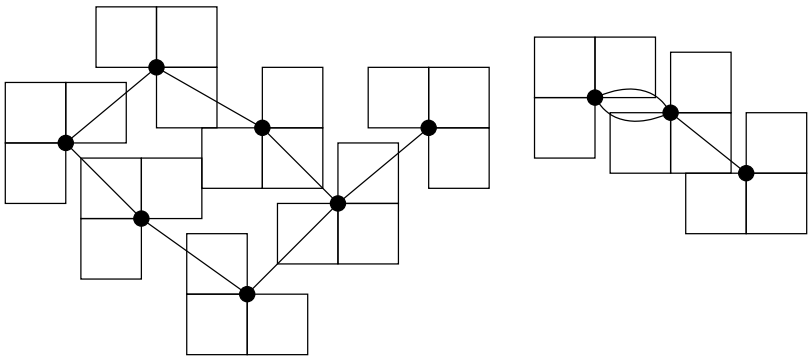


Abbildung 8.3: Beispiel einer möglichen Beschriftung

Lemma 8.2 *Alle Punkte einer Zusammenhangskomponente, welche nicht in einem Kreis liegen, müssen die drei Label setzen, die nicht dem Kreis zugewandt sind, um eine gültige Beschriftung zu erhalten.*

Beweis. Angenommen an einem Punkt $p_1 \notin K$ wird der dem Kreis zugewandte Label gesetzt. Die weiteren Punkte p_i müssen dann auch immer den dem Kreis zugewandte Label setzen. Somit setzt auch p_{k-1} einen Label l_{k-1}, der einen Label l_k von $p_k \in K$ schneidet. l_k kann kein Label sein, das wegen der zyklischen Anordnung in K sowieso schon weggelassen wurde, da sonst p_{k-1} ein Punkt des Kreises sein müsste. Somit kann p_k den Label zu p_{k-1} nicht setzen und läßt ein zweites zyklisch weg. Also erhält man keine gültige Beschriftung. ◻

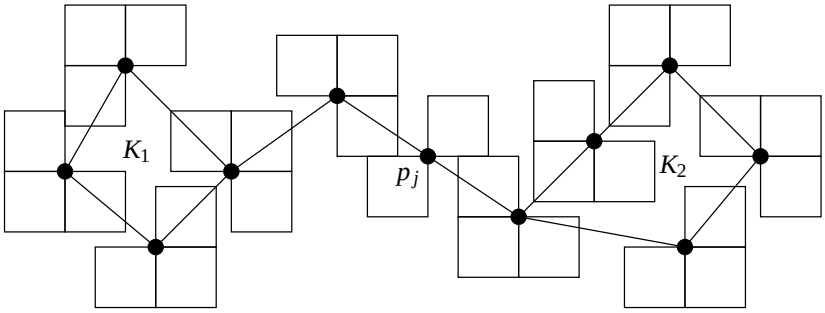


Abbildung 8.4: Beispiel, in dem eine Beschriftung mit der gegebenen Kantenlänge nicht möglich ist.

Beim Betrachten der Abbildung 8.4 erkennt man sofort, dass keine Beschriftung mit der aktuellen Beschriftungsgröße möglich ist. Im Graphen sind innerhalb einer Zusammenhangskomponente zwei Zyklen. Dies führt zu Lemma 8.3.

Lemma 8.3 *Eine Beschriftung von n Punkten mit einer Beschriftungsgröße s ist genau dann möglich, wenn jede Zusammenhangskomponente des Graphen G_s höchstens einen Kreis enthält.*

Beweis. Wir betrachten zwei Fälle.

(i) G_s hat eine Zusammenhangskomponente mit mindestens zwei Kreisen K_1, K_2.

   a. Falls $K_1 \cap K_2 = \emptyset$, so existiert ein Pfad $(p_1, \ldots, p_k)$ mit $p_1 \in K_1$, $p_k \in K_2$ und $(p_i, p_{i+1}) \in E_s$. Nach Lemma 8.2 muß ein p_j mit $j \in \{2, \ldots, k-1\}$ die Label setzen, welche nicht K_1 und auch nicht K_2 zugewandt sind. Somit kann p_j nur zwei Label setzen und eine gültige Beschriftung ist nicht möglich, siehe Abbildung8.4.

   b. Falls $K_1 \cap K_2 \neq \emptyset$ so sei $p \in K_1 \setminus K_2$. Dann hat p zwei, dem Kreis K_2 zugewandte Label und müßte nach Lemma 8.2 diese beiden weglassen. Damit ist keine gültige Beschriftung möglich.

(ii) In jeder Zusammenhangskomponente von G_s befindet sich höchstens ein Kreis.

   a. Wenn in einer Zusammenhangskomponente genau ein Kreis K existiert, so werden die Label der Punkte, wie in Lemma 8.1 und Lemma 8.2 beschrieben, gesetzt.

   b. In einer Zusammenhangskomponente ohne Kreis wähle man sich einen beliebigen Punkt p und setze an allen anderen Punkten der Zusammenhangskomponente diejenigen Label, welche nicht p zugewandt sind. An p läßt man ein beliebiges (z.B. grundsätzlich das rechts unten) weg.

◻

Zum Finden dieser Kreise durchsucht man jede Zusammenhangskomponente des Graphen von einem beliebigen Startpunkt per Tiefen- oder Breitensuche. Wenn man an einen Punkt gelangt, den man schon besucht hat, so hat man einen Kreis gefunden. Man streicht die Kante, auf der man zu dem Punkt gekommen ist, und durchläuft die gesamte Komponente noch einmal. Findet man einen weiteren Kreis, so ist die Beschriftung mit der Größe s nicht möglich.

Die Anzahl der Kanten des Graphen G_s ist linear in n und das Feststellen der Anzahl der Kreise linear in der Anzahl der Kanten. Es folgt also

Korollar 8.1 *Bei gegebenem G_s kann das Entscheidungsproblem für eine gegebene Beschriftungsgröße s in $O(n)$ Zeit gelöst werden.*

8.2 Mögliche Beschriftungsgrößen

Da gezeigt wurde, dass das Entscheidungsproblem lösbar ist, benötigt man nun eine Liste L von möglichen Beschriftungsgrößen s.

Offensichtlich gilt: Bei maximaler Beschriftungsgröße $s_{\max}$ müssen sich die Label von mindestens zwei Punkten berühren. Denn wenn dies nicht der Fall wäre, könnte man $s_{\max}$ noch größer wählen.

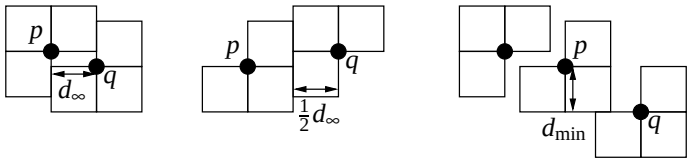


Abbildung 8.5: Kandidaten für die maximale Kantenlänge bei der Beschriftung von p und q

Sei $d_\infty(p,q) = \max\{|x_p - x_q|, |y_p - y_q|\}$ der Abstand der Punkte p, q in der Maximumsmetrik und sei $d_{\min}(p,q) = \min\{|x_p - x_q|, |y_p - y_q|\}$. Abbildung 8.5 zeigt alle möglichen Fälle maximal großer Beschriftung zweier Punkte $p,q \in P$. Dies führt uns zu folgendem Lemma.

Lemma 8.4 *Es gibt $p,q \in P$, so dass $s_{\max} \in \{d_\infty(p,q), \frac{1}{2}d_\infty(p,q), d_{\min}(p,q)\}$.*

Für die n gegebenen Punkte wären dies $O(n^2)$ mögliche Beschriftungsgrößen. Wie man in Abbildung 8.6(a) sieht, gibt es für jeden Punkt p eine Umgebung B, in der alle Punkte liegen müssen, welche mit ihren Labeln noch die von p berühren. Da die Label der Punkte in B in einem Quadrat C der Kantenlänge $6s$ liegen müssen und sich im Fall $s \leq s_{\max}$ nicht schneiden dürfen, ist klar, dass C nur 12 Punkte enthalten kann. Aber wir können diese Konstante mit einer anderen Argumentation noch weiter senken.

Lemma 8.5 *Als mögliche Beschriftungsgrößen kommen je Punkt nur die Abstände (im Sinne von Lemma 8.4) zu seinen vier nächsten Nachbarn, gemessen in der d_∞-Metrik, in Frage.*

Beweis. Für p gibt es vier kritische Punkte, nämlich die Ecken des Quadrates A mit Kantenlänge $2s$ und Mittelpunkt p. Einerseits muss jeder Punkt q, dessen Label die von p schneiden, einen Label haben, das einen dieser kritischen Punkte enthält. Andererseits darf jeder dieser Punkte bei einer gültigen Beschriftung höchstens in einem möglichen Label eines anderen Punktes p_i liegen, siehe Abbildung 8.6(b), da der Graph G_s sonst an dieser Stelle mindestens zwei Kreise enthalten würde. Damit können sich die Label von p nur mit den Labeln dieser vier Punkte p_i berühren, weshalb nur die Abstände zu diesen nächsten Punkten als Beschriftungsgrößen in Frage kommen. □

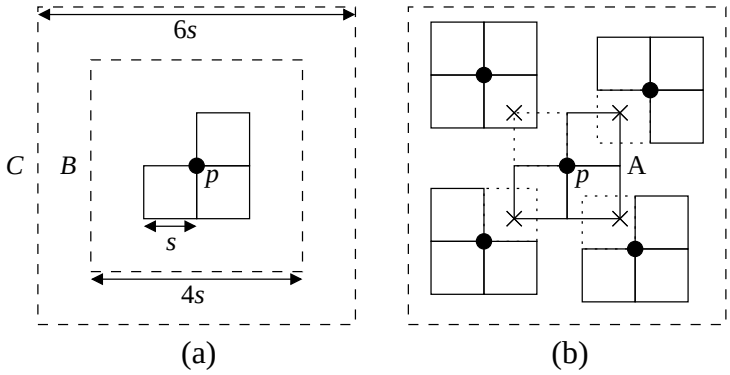


Abbildung 8.6: Umgebungen von p mit $s \leq s_{\max}$

Lemma 8.6 *Die Länge der Liste der möglichen Beschriftungsgrößen L ist linear. Wenn für jeden $p \in P$ seine vier nächsten Nachbarn bekannt sind, so ist L auch in linearer Zeit erstellbar.*

8.3 Algorithmus

Als erstes wird für jeden Punkt mit einem Sweepline-Algorithmus eine Liste seiner vier nächsten Nachbarn erstellt. Dies benötigt $O(n \log n)$ Zeit. Daraus wird die Liste L der möglichen Beschriftungsgrößen in $O(n)$ Zeit erstellt und dann in $O(n \log n)$ Zeit sortiert.

Nun beginnt die binäre Suche. Solange in L noch mehr als ein Element steht, wird mit dem mittleren Element das Entscheidungsproblem gelöst. Wenn mit der Größe eine Beschriftung möglich ist, werden alle kleineren Listenelemente gestrichen. Ansonsten wird das aktuelle und alle größeren Elemente aus der Liste entfernt. Das Listenelement, welches zum Schluss noch in L steht, ist die gesuchte größtmögliche Beschriftungsgröße $s_{\max}$.

Mit $s_{\max}$ wird $G_{s_{\max}}$ erstellt. Dann werden die Label entsprechend der Lemmata 8.1 und 8.2 gesetzt.

Satz 8.1 *Die Berechnung einer Beschriftung von n Punkten mit je drei Quadraten maximaler Kantenlänge benötigt $O(n \log n)$ Zeit und $O(n)$ Speicherplatz.*

Beweis. Die binäre Suche in L benötigt $O(\log |L|)$ Schritte und damit nach Lemma 8.6 $O(\log n)$ Schritte. Nach Korollar 8.1 benötigt die Lösung des Entscheidungsproblems für eine Beschriftungsgröße s $O(n)$ Zeit. Da das Entscheidungsproblem $O(\log n)$-mal gelöst werden muß, benötigt die gesamte Berechnung von $s_{\max}$ und die Beschriftung der Punkte $O(n \log n)$ Zeit. Da L lineare Länge und G_s eine lineare Anzahl von Punkten und Kanten hat, benötigt der Algorithmus $O(n)$ Speicherplatz. □

8.4 Ausblick

Zur weiteren Maximierung der Beschriftungsgröße kann man auch annehmen, dass einer der drei gesetzten Label den Punkt, den es beschriftet, nur mit einer Kante berühren muß. Es ist also möglich, je einen Label pro Punkt hin und her zu schieben.

Wenn man dieses Problem mit dem gleichen Ansatz lösen würde, benötigte man eine bedeutend längere Liste von möglichen Beschriftungsgrößen, da nicht nur die Abstände zu den nächsten vier Punkten betrachtet werden müssten. Aufgrund der Verschiebung der einzelnen Label können sich auch Punkte, die weit entfernt liegen, auf die Beschriftungsgröße auswirken. Die möglichen Beschriftungsgrößen sind

$$\frac{1}{i}d_\infty(p,q), \quad \frac{1}{2i}d_\infty(p,q), \quad \frac{1}{i}d_{\min}(p,q), \quad \forall p,q \in P, \quad i = \{1,\ldots,n\}\,,$$

das heißt $|L| \in O(n^3)$. Die Beschreibung eines Algorithmus für dieses Problem befindet sich ebenfalls in [DQZ01].

Kapitel 9

New and Open Problems in Automated Label Placement

Alexander Wolff

Annotating maps, graphs, and diagrams with pieces of text is an important step in information visualization that is usually referred to as label placement. The demand for the automation of label placement has increased with the amount of information to be visualized. The ACM Computational Geometry Impact Task Force report [Cc99] denotes label placement as an important research area. Manually labeling a map, for example, is a tedious task that is estimated to take 50% of total map production time [Mor80].

Like in computational geometry in general, many label-placement problems start with the sentence "given *n* points in the plane...". The first label-placement problem that was considered by theoreticians is the following. Given these *n* points in the plane, is it possible to attach to each of them an axis-parallel unit square, its *label*, such that each point coincides with one of the four corners of its label? Just ten years ago Formann and Wagner showed that it is NP-hard to decide this question [FW91]. So they considered the following maximization problem instead: what is the largest number ℓ such that all of the given points can be labeled with squares of edge length ℓ? They gave an algorithm that always finds a labeling with squares of at least half the maximum label size. They also showed that one cannot expect to find an efficient algorithm with an approximation factor greater than $\frac{1}{2}$.

Since then there has been a steady flow of publications in automated label-placement, see Figure 9.1. In this write-up I am going to list some of the problems that have not or only partially been solved yet. I would be glad if any item from this list catches your attention! If you need further information, do not hesitate to ask me, or simply check the Map-Labeling Bibliography [WS96]. From there you can download many papers.

9.1 Approximation factors to improve

So far approximation algorithms have only been given for labeling points. This may be due to the fact that for polylines and polygons the target function is not as obvious as for points, where you usually want to maximize the size of labels, the number or the weight sum of points that can be labeled without intersection. Size maximization, though not as relevant as the other two target functions, usually gives rise to very nice geometric algorithms and proofs, see e.g. [SK02, WTX02]. Number maximization is related to the maximum independent set

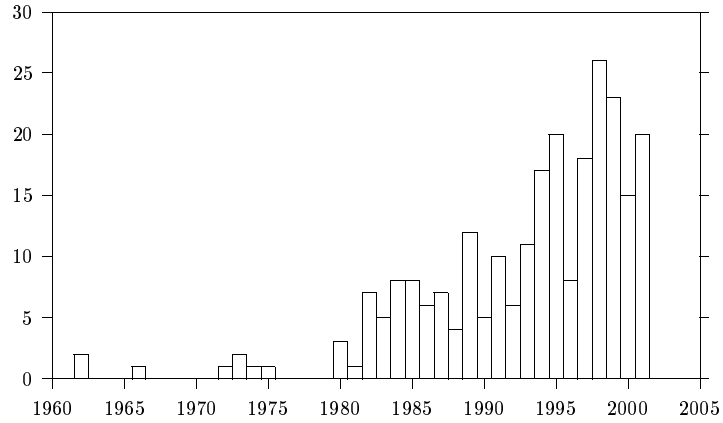


Abbildung 9.1: Number of publications about automated label placement over time.

problem (see Chapters 5 and 7 as well as [AvKS98]). Similarly, weight maximization is related to the maximum weight independent set problem (see Chapter 5) and to scheduling problems [PSSW01]. While geometry still plays a role in number maximization, weight maximization is rather combinatorial in nature.

Size maximization of sliding square labels. Consider the problem of Formann and Wagner, where the restiction on the label position is relaxed in that now a label can touch its point anywhere on the label boundary. These are so-called *sliding labels* as opposed to *fixed-position labels*. The decision problem, i.e. can a set of points be labeled with sliding unit squares, is NP-hard [vKSW99]. While there are papers on maximizing the number [vKSW99] or weight sum [PSSW01] of the points that receive a (sliding) label, so far the only approximation algorithm for maximizing the size of axis-parallel square labels is actually the algorithm of Formann and Wagner. It is based on the simple observation [ZQ02] that each optimal solution for sliding squares can be mapped to a labeling where labels are half the size and touch their point with a corner. Thus Formann and Wagner's algorithm has an approximation factor of $\frac{1}{4}$ for this problem. By generalizing the algorithm of Formann and Wagner to eight label positions, one gets a better mapping and thus a factor-$\frac{1}{3}$ approximation algorithm with the same time bound, i.e. $O(n \log n)$. Very recently a factor-$\frac{1}{2}$ approximation algorithm for the problem has in fact been suggested [QZ02], but its running time of $O(n^2 \log n)$ seems to be suboptimal.

Variants of the problem, where labels can additionally have arbitrary orientation, have also been studied [DMM$^+$97, ZQ02].

Size maximization of circular labels. For maximizing the size of circular point labels the currently best algorithm has an approximation factor of $\frac{1}{3.6}$ [DMM02]. However, the algorithm is pseudo-polynomial and the proof of its correctness is rather complicated, while the NP-hardness proof in [SW01] suggests that there is room for improvement. I conjecture that a factor anywhere between $\frac{1}{3}$ and $\frac{1}{2}$ should be possible.

Number maximization given points on a line. Recently the problem of labeling points on a given line (horizontal or sloping) has been considered [GIM$^+$01]. In that paper the size of square or rectangular labels is maximized. It should be possible to improve some of the given time bounds, but maybe it is more interesting to turn to other target functions: Can you give exact or approximate solutions for maximizing the *number* of points on a line that can be labeled given fixed-position or sliding labels?

Weight maximization. What about maximizing the weight sum, if points have weights? There is an article on weight maximization given sliding unit-height labels and points on a horizontal line [PSSW01], but in that article labels are intervals, i.e. they can only be placed on the line, not both above and below. In spite of that the authors achieve only a factor $(1+\varepsilon)$-approximation. Can you give an exact algorithm or show that the problem is NP-hard? Can you improve upon their factor $(2+\varepsilon)$-approximation for points in the plane? There is a polynomial-time approximation scheme maximizing the number of labels that receive unit-height labels, both in the case of fixed-position [AvKS98] and sliding labels [vKSW99].

Maximum (weight) independent set. This is related to the previous item. Erlebach et al. recently settled a long-standing open problem by giving a polynomial-time approximation scheme for maximum-weight independent set on intersection graphs of axis-parallel squares [EJS01]. However, their result does not carry over to rectangles. So far only a factor-$O(\log n)$ approximation algorithm for this problem is known [AvKS98, Itu99], but it seems *very* hard to come up with anything better.

9.2 New problems

I am continuing the list with problems that I always would have liked to solve myself but either failed or didn't find the time to... The combination of graph drawing and label placement that is addressed under the title "Subway labeling" is the most important open problem from a practical point of view.

Subway labeling. Suppose you are given the subway network of a city as a (geometric) graph, and a set of directions that you can use to draw the lines (e.g. 0°, 45°, 90°), how would you draw the subway map? There is a very nice recent paper on this problem [CdBvD$^+$01]. However, it does not consider two additional problems:

- How can subway stations be moved from their true geometric positions to positions that help straightening the subway lines?
- How can the above problem, which is a graph-drawing problem, be combined with the problem of labeling the subway stations?

Any approach to combining graph drawing and label placement would be *extremely* valuable for many practical applications, see Figure 9.2 for a beautiful example. However, I am only aware of one, very recent publication [KM02] that tries to tackle this important problem.

The subway-labeling problem could of course also be considered in isolation: given a polygonal area that is delimited by a few subway lines, how can you label all stations within this area? Related literature: there is a heuristic for labeling graph drawings [KT97] and a paper on *elastic* labels (i.e. labels of fixed area, but flexible height and length) that must be placed on the perimeter of a map [IL98].

Balloon Labeling. You are given a (possibly very dense) set of points in the plane and for each a label (e.g. a unit disk). All points must be labeled, but in general there won't be enough space to attach the labels directly to their points. Thus you must use so-called *leader symbols*, straight line segments or arrows that connect a point with its label. In other words: strings that connect points with their balloons. Now you might want to minimize

- the average string length,

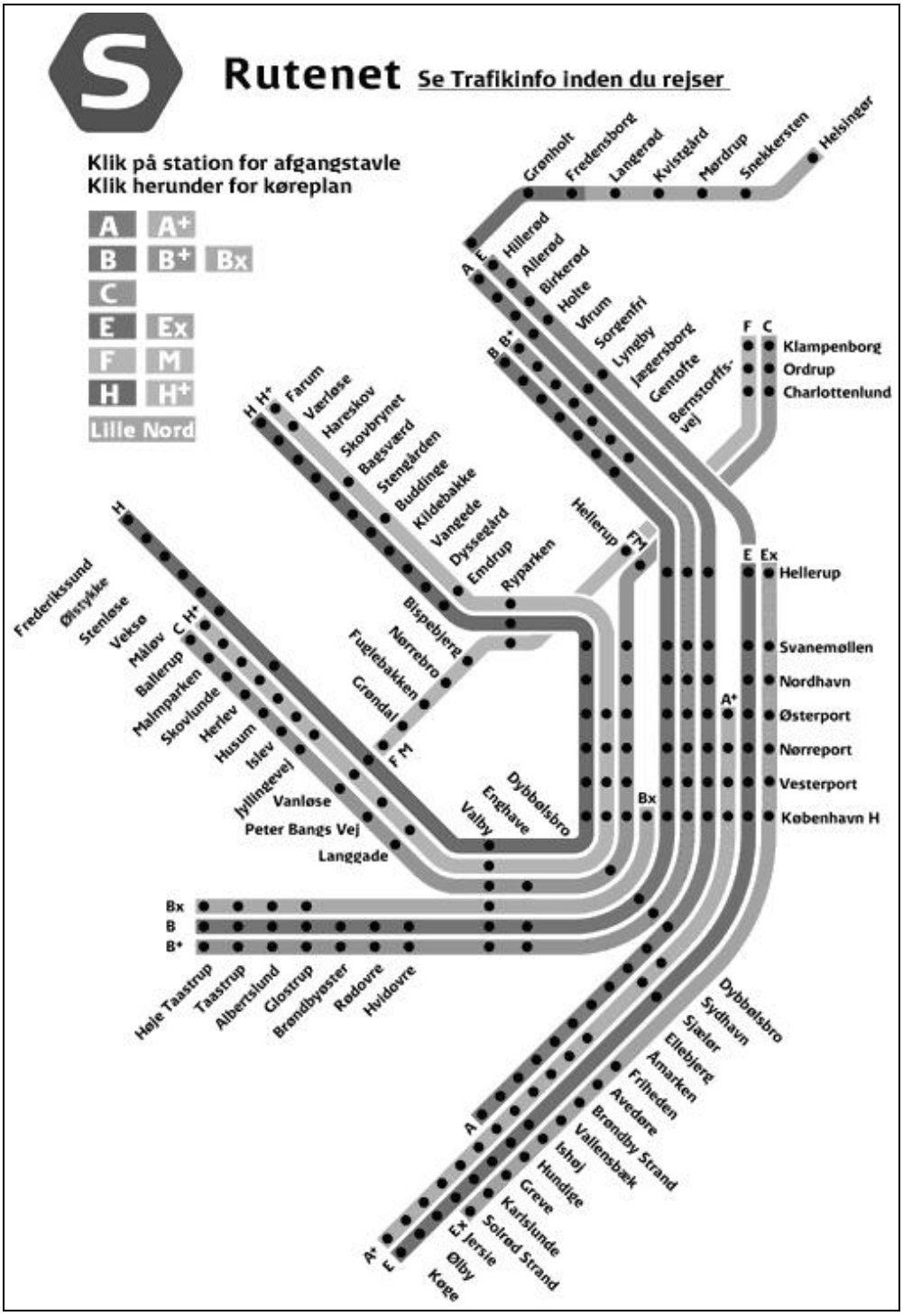


Abbildung 9.2: The commuter-train network of Copenhagen, see `www.dsb.dk/stog`.

- the maximum string length, or
- the sum of all string lengths.

Deciding whether a point set can be labeled using unit disks without leaders is NP-hard [SW01], so you cannot expect to find optimal algorithms for these problems. Can you find approximations or good heuristics? An incremental heuristic that is being used by a small GIS company near Greifswald is based on a physical model: you have repelling forces between the labels (as between electrons) and attracting forces between a point and its label (as between an electron and a proton). Any solution to this problem should be implemented and compared to this heuristic on real-world data.

Besides the direct application, e.g. for labeling dense scatter plots or oil drill-holes (see [Zor97] for examples), solutions to this problem might also be used for labeling small areas of land, e.g. in land-use or geological maps. In that case a good representative point for each area must be determined before applying the balloon-labeling program.

Interactive 3D-labeling. You have a 3D-object that the user can rotate into various directions. Some points on this object are labeled so that the user always knows which part (s)he sees. You must anticipate possible collisions between labels and then shift them smoothly in such a way that they never overlap (too much). Labels that are distant from the user should become smaller and might also have a lower priority than closer labels.

Area labeling. Although this problem is not new of course, there is no "nice" geometric algorithm for it so far. Previous attempts either simply compute the maximum-size inscribed rectangle of given aspect ratio [vR89, AIK89] or compute an estimate of a good baseline for the label text. The baseline is either derived from the medial axis [AF84] or it is the result of an incremental process that generates and then evaluates arcs between pairs of points on a grid that gets finer in each iteration [PF96].

Layer labeling. Suppose you want to label points with given fixed-size labels, but you cannot place all of them, then you could partition the points into several layers (as in a geographic information systems or in VLSI layout) and then label each layer seperately, such that labels of one layer don't intersect while labels of different layers may. What is the minimum amount of layers (or screens between which the user can swap)? Since the one-layer problem is NP-hard [FW91], you will have to look for approximation algorithms.

Autorenverzeichnis

Kathrin Bach
Studienrichtung: Lehramt Gymnasium Mathematik & Latein
10. Semester
Universität Greifswald
Email: kathrinbach@hotmail.com

Kristina Hanig
Studienrichtung: Mathematik Diplom
8. Semester
Universität Greifswald
Email: kh970325@uni-greifswald.de
WWW: http://www.math-inf.uni-greifswald.de/~hanig

Tim Hoffmann
Studienrichtung: Lehramt Gymnasium Geografie & Mathematik
10. Semester
Universität Greifswald
Email: Tim.G.Hoffmann@web.de
WWW: http://www.der-huehnerhabicht.de/tim

Wolfgang Kresse
Professor für Fotogrammmetrie, Geo-Informationssysteme (GIS), Kartografie und Fernerkundung
Fachbereich Bauingenieur- und Vermessungswesen
Fachhochschule Neubrandenburg
Email: kresse@fh-nb.de
WWW: http://www.fh-nb.de/forschung/vermess/3.asp

Julia Löcherbach
Studienrichtung: Biomathematik Diplom
4. Semester
Universität Greifswald
Email: Julia.Loecherbach@gmx.de
WWW: http://mitglied.lycos.de/julial49

Paul Rosenthal
Studienrichtung: Mathematik Diplom
2. Semester
Universität Greifswald
Email: paulrosee@web.de

Steffen Rudnick
Studienrichtung: Mathematik Diplom
2. Semester
Universität Greifswald
Email: Steffen.Rudnick@gmx.de

Peter Schreiber
Professor für Geometrie und Grundlagen der Mathematik
Institut für Mathematik und Informatik
Universität Greifswald
Email: schreibe@uni-greifswald.de
WWW: http://sun-10.math-inf.uni-greifswald.de/logik/schreiber

Michael Thon
Studienrichtung: Mathematik Diplom
4. Semester
Universität Göttingen
Email: mithon42@gmx.de

Alexander Wolff
Wissenschaftlicher Assistent
Institut für Mathematik und Informatik
Universität Greifswald
Email: awolff@uni-greifswald.de
WWW: http://www.math-inf.uni-greifswald.de/~awolff

Literaturverzeichnis

[AF84] AHN, JOHN und HERBERT FREEMAN: *A program for automatic name placement*. Cartographica, 21(2–3):101–109, 1984.

[AIK89] AONUMA, HIROMI, HIROSHI IMAI und YAHIKO KAMBAYASHI: *A Visual System of Placing Characters Appropriately in Multimedia Map Databases*. In: KUNII, T. L. (Herausgeber): *Proc. IFIP TC 2/WG 2.6 Working Conference on Visual Database Systems*, Seiten 525–546. North-Holland, 1989.

[AvKS97] AGARWAL, PANKAJ K., MARC VAN KREVELD und SUBHASH SURI: *Label Placement by Maximum Independent Set in Rectangles*. In: *Proceedings of the 9th Canadian Conference on Computational Geometry (CCCG'97)*, Seiten 233–238, 1997.

[AvKS98] AGARWAL, PANKAJ K., MARC VAN KREVELD und SUBHASH SURI: *Label placement by maximum independent set in rectangles*. Computational Geometry: Theory and Applications, 11:209–218, 1998.

[Bri92] BRIESKORN, EGBERT (Herausgeber): *Katalog der Ausstellung: Felix Hausdorff – Paul Mongré 1868–1942 (Bonn, 1992)*. Mathematisches Institut der Rheinischen Friedrich-Wilhelms-Universität Bonn, 1992.

[BW00] BRANDES, ULRIK und DOROTHEA WAGNER: *Visualisierung von Verkehrsdaten*. DMV-Mitteilungen, 1:11–15, 2000.

[Cc99] CHAZELLE, BERNARD und 36 CO-AUTHORS: *The Computational Geometry Impact Task Force Report*. In: CHAZELLE, B., J. E. GOODMAN und R. POLLACK (Herausgeber): *Advances in Discrete and Computational Geometry*, Band 223, Seiten 407–463. American Mathematical Society, Providence, 1999.

[CdBvD+01] CABELLO, SERGIO, MARK DE BERG, STEVEN VAN DIJK, MARC VAN KREVELD und TYCHO STRIJK: *Schematization of Road Networks*. In: *Proceedings of the 17th Annual ACM Symposium on Computational Geometry (SoCG'01)*, Seiten 33–39, Medford, MA, 2001.

[CLR96] CORMEN, THOMAS H., CHARLES E. LEISERSON und RONALD L. RIVEST: *Introduction to Algorithms*. MIT Press, Cambridge, MA, 16. Auflage, 1996.

[CMS95] CHRISTENSEN, JON, JOE MARKS und STUART SHIEBER: *An Empirical Study of Algorithms for Point-Feature Label Placement*. ACM Transactions on Graphics, 14(3):203–232, 1995.

[Dan51] DANTZIG, GEORGE B.: *Maximization of a Linear Function of Variables Subject to Linear Inequalities*. In: T. C. KOOPMANS (Herausgeber): *Activity Analysis of Production and Allocation*, Seiten 339–347. John Wiley & Sons, New York, 1951.

[DETT98] DI BATTISTA, GIUSEPPE, PETER EADES, ROBERTO TAMASSIA und IOANNIS G. TOLLIS: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, Upper Saddle River, New Jersey 07458, U.S.A, Juli 1998.

[DMM+97] DODDI, SRINIVAS, MADHAV V. MARATHE, ANDY MIRZAIAN, BERNARD M.E. MORET und BINHAI ZHU: *Map labeling and its generalizations*. In: *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, Seiten 148–157, New Orleans, LA, 4.–7. Januar 1997.

[DMM02] DODDI, SRINIVAS, MADHAV V. MARATHE und BERNARD M.E. MORET: *Point Set Labeling with Specified Positions*. International Journal of Computational Geometry and Applications, 12(1–2):29–66, 2002.

[DQZ01] DUNCAN, ROB, JIANBO QIAN und BINHAI ZHU: *Polynomial time algorithms for three-label point labeling*. In: *Proc. 7th Annual International Computing and Combinatorics Conf. (COCOON'01)*, Band 2108 der Reihe *Lecture Notes in Computer Science*, Seiten 191–200, Guilin, 20.–23. August 2001. Springer-Verlag.

[EJS01] ERLEBACH, THOMAS, KLAUS JANSEN und EIKE SEIDEL: *Polynomial-time approximation schemes for geometric graphs*. In: *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, Seiten 671–679, Washington, DC, 7.–9. Januar 2001.

[EM81] EDELSBRUNNER, HERBERT und HERMANN A. MAURER: *On the intersection of orthogonal objects*. Inform. Process. Lett., 13:177–181, 1981.

[FM37] FÜLÖP-MILLER, RENÉ: *Kulturgeschichte der Heilkunde*. Bruckmann, München, 1937.

[Fou00] FOURER, ROBERT: *Linear Programming Frequently Asked Questions*. `http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html`. Optimization Technology Center of Northwestern University and Argonne National Laboratory, 2000.

[FPT81] FOWLER, ROBERT J., MICHAEL S. PATERSON und STEVEN L. TANIMOTO: *Optimal Packing and Covering in the Plane are NP-Complete*. Information Processing Letters, 12(3):133–137, 1981.

[FW91] FORMANN, MICHAEL und FRANK WAGNER: *A Packing Problem with Applications to Lettering of Maps*. In: *Proc. 7th Annu. ACM Sympos. Comput. Geom. (SoCG'91)*, Seiten 281–288, 1991.

[GIM+01] GARRIDO, MARI ÁNGELES, CLAUDIA ITURRIAGA, ALBERTO MÁRQUEZ, JOSÉ RAMON PORTILLO, PEDRO REYES und ALEXANDER WOLFF: *Labeling Subway Lines*. In: EADES, PETER und TADAO TAKAOKA (Herausgeber): *Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01)*, Band 2223 der Reihe *Lecture Notes in Computer Science*, Seiten 649–659, Christchurch, 19.–21. Dezember 2001. Springer-Verlag.

[Gom58] GOMORY, RALPH E.: *Outline of an algorithm for integer solutions to linear programs*. Bulletin of the American Mathematical Society, 64:275–278, 1958.

[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*. Information Processing Letters, 43(5):229–235, Oktober 1992.

[IL98] ITURRIAGA, CLAUDIA und ANNA LUBIW: *Elastic Labels on the Perimeter of a Rectangle*. In: WHITESIDES, SUE H. (Herausgeber): *Proceedings of the Symposium on Graph Drawing (GD'98)*, Band 1547 der Reihe *Lecture Notes in Computer Science*, Seiten 452–453. Springer-Verlag, 13.–15. August 1998.

[Imh62] IMHOF, EDUARD: *Die Anordnung der Namen in der Karte*. In: *International Yearbook of Cartography*, Seiten 93–129, Bonn Bad Godesberg, 1962. Kirschbaum.

[Imh75] IMHOF, EDUARD: *Positioning Names on Maps*. The American Cartographer, 2(2):128–144, 1975.

[Itu99] ITURRIAGA, CLAUDIA: *Map Labeling Problems*. Doktorarbeit, University of Waterloo, 1999.

[Kan39] KANTOROVICH, LEONID V.: *Mathematical methods of organizing and planning production*. Universität Leningrad, 1939. Englische Übersetzung erschienen in Management Science, 6:366–422, 1960.

[Kar84] KARMARKAR, NARENDRA K.: *A new polynomial–time algorithm for linear programming*. Combinatorica, 4:373–395, 1984.

[KM72] KLEE, VICTOR und GEORGE J. MINTY: *How good is the simplex algorithm?* In: SHISHA, O. (Herausgeber): *Inequalities III*, Seiten 159–175. Academic Press, New York, 1972.

[KM00] KLAU, GUNNAR W. und PETRA MUTZEL: *Optimal Labelling of Point Features in the Slider Model*. In: DU, D.-Z., P. EADES, V. ESTIVILL-CASTRO, X. LIN und A. SHARMA (Herausgeber): *Proc. 6th Annual International Computing and Combinatorics Conf. (COCOON'00)*, Band 1858 der Reihe *Lecture Notes in Computer Science*, Seiten 340–350, Sydney, 26.–28. Juli 2000. Springer-Verlag.

[KM02] KLAU, GUNNAR W. und PETRA MUTZEL: *Automatic Layout and Labelling of State Diagrams*. In: *Mathematics—Key Technology for the Future*. Springer-Verlag, Berlin, 2002. Erscheint in Bälde.

[Knu63] KNUTH, DONALD: *Computer-Drawn Flowcharts*. Commun. Assoc. Comput. Mach., 6:555–563, 1963.

[Kre94] KRESSE, WOLFGANG: *Plazierung von Schrift in Karten*. Doktorarbeit, Hohe Landwirtschaftliche Fakultät der Rheinischen Friedrich-Wilhelms-Universität, Bonn, Mai 1994. Erhältlich als Heft 23 der Schriftenreihe des Instituts für Kartographie und Geoinformation.

[KT97] KAKOULIS, KONSTANTINOS G. und IOANNIS G. TOLLIS: *An Algorithm for Labeling Edges of Hierarchical Drawings*. In: *Proceedings of the Symposium on Graph Drawing (GD'97)*, Band 1353 der Reihe *Lecture Notes in Computer Science*, Seiten 169–180. Springer-Verlag, 1997.

[KW01] KAUFMANN, MICHAEL und DOROTHEA WAGNER: *Drawing graphs: methods and models*, Band 2025 der Reihe *Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 2001.

[LD60] LAND, A. H. und A. G. DOIG: *An automatic method of solving discrete programming problems*. Econometrica, 28:497–520, 1960.

[MBH+95] MARATHE, MADHAV V., HEINZ BREU, HARRY B. HUNT III, S.S. RAVI und DANIEL J. ROSENKRANTZ: *Simple Heuristics for Unit Disk Graphs*. Networks, 25:59–68, 1995.

[McC85] MCCREIGHT, EDWARD M.: *Priority search trees*. SIAM Journal on Computing, 14(2):257–276, 1985.

[Mor80] MORRISON, JOEL L.: *Computer Technology and Cartographic Change*. In: TAYLOR, D.R.F. (Herausgeber): *The Computer in Contemporary Cartography*. J. Hopkins Univ. Press, New York, 1980.

[MS91] MARKS, JOE und STUART SHIEBER: *The Computational Complexity of Cartographic Label Placement*. Technischer Bericht TR-05-91, Harvard CS, 1991.

[PF96] PINTO, ITZHAK und HERBERT FREEMAN: *The Feedback Approach to Cartographic Areal Text Placement*. In: PERNER, P., P. WANG und A. ROSENFELD (Herausgeber): *Advances in Structural and Syntactical Pattern Recognition*, Seiten 341–350. Springer-Verlag, New York, 1996.

[PSSW01] POON, SHEUNG-HUNG, CHAN-SU SHIN, TYCHO STRIJK und ALEXANDER WOLFF: *Labeling Points with Weights*. In: *Proc. 17th European Workshop on Computational Geometry (CG'01)*, Seiten 97–100, Berlin, 26.–28. März 2001.

[QWXZ00] QIN, ZHONGPING, ALEXANDER WOLFF, YINFENG XU und BINHAI ZHU: *New Algorithms for Two-Label Point Labeling*. In: PATERSON, MIKE (Herausgeber): *Proc. 8th Annu. Europ. Symp. on Algorithms (ESA'00)*, Band 1879 der Reihe *Lecture Notes in Computer Science*, Seiten 368–379, Saarbrücken, 5.–8. September 2000. Springer-Verlag.

[QZ02] QIN, ZHONGPING und BINHAI ZHU: *A factor-2 approximation for labeling points with maximum sliding labels*. In: *Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT'02)*, Lecture Notes in Computer Science, Turku, 3.–5. Juli 2002. Springer-Verlag. Erscheint in Bälde.

[Rai98] RAIDL, GÜNTHER: *A Genetic Algorithm for Labeling Point Features*. In: *Proc. of the Int. Conference on Imaging Science, Systems, and Technology*, Seiten 189–196, Las Vegas, NV, Juli 1998.

[Rai99] RAIDL, GÜNTHER: *An Evolutionary Approach to Point-Feature Label Placement*. In: BANZHAF, W., J. DAIDA, A.E. EIBEN, M.H. GARZON, V. HONAVAR, M. JAKIELA und R.E. SMITH (Herausgeber): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)*, Seite 807. Morgan Kaufmann, Juli 1999.

[Rum98] RUMPLMAIER, WOLFGANG: *Optimierung von Labelanordnungen mit Genetischen Algorithmen und Simulated Annealing*. Diplomarbeit, Institute of Computer Graphics, Vienna University of Technology, April 1998.

[Sch75] SCHREIBER, PETER: *Theorie der geometrischen Konstruktionen*. Deutscher Verlag der Wissenschaften, Berlin, 1975.

[Sch84] SCHREIBER, PETER: *Grundlagen der konstruktiven Geometrie*. Deutscher Verlag der Wissenschaften, Berlin, 1984.

[Sch91] SCHREIBER, PETER: *Generalized Construction Problems*. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 37:57–62, 1991.

[Sch92] SCHREIBER, PETER: *Kartographie als darstellende Geometrie*. Mathematik in der Schule, 7/8:427–432, 1992.

[Sch96] SCHREIBER, PETER: *On a Constructive Property of Poincaré's Model of the Hyperbolean Plane*. In: GREFFE, JEAN-LOUIS, GERHARD HEINZMANN und KUNO LORENZ (Herausgeber): *Akten des Internationalen Kongresses: Henri Poincaré: Wissenschaft und Philosophie (Nancy 1994)*, Publikationen des Henri-Poincaré-Archivs, Seiten 259–263, Paris/Berlin, 1996. Albert Blanchard/Akademie-Verlag.

[SK02] SPRIGGS, MICHAEL J. und J. MARK KEIL: *A New Bound for Map Labeling with Uniform Circle Pairs*. Information Processing Letters, 81(1):47–53, 2002.

[Sok93] SOKOP, BRIGITTE: *Stammtafeln europäischer Herrscherhäuser*. Böhlau-Verlag, Wien, 1993. 3. Auflage.

[SRW+81] SONNEMANN, R., S. RICHTER, H. WOLFFGRAMM, G. BUCHHEIM und H. ESCHWEGE (Herausgeber): *Allgemeine Geschichte der Technik von den Anfängen bis 1870*. Fachbuchverlag, Leipzig, 1981.

[SW01] STRIJK, TYCHO und ALEXANDER WOLFF: *Labeling Points with Circles*. International Journal of Computational Geometry and Applications, 11(2):181–195, April 2001.

[Tut63] TUTTE, WILLIAM T.: *How to Draw a Graph*. Proc. London Mathematical Society, 13:743–768, 1963.

[VA99] VERWEIJ, BRAM und KAREN AARDAL: *An Optimisation Algorithm for Maximum Independent Set with Applications in Map Labelling*. In: *Proc. 7th Annu. Europ. Symp. on Algorithms (ESA'99)*, Band 1643 der Reihe *Lecture Notes in Computer Science*, Seiten 426–437, Prague, 16.–18. Juli 1999. Springer-Verlag.

[vD01] DIJK, STEVEN VAN: *Genetic Algorithms for Map Labeling*. Doktorarbeit, Utrecht University, Department of Computer Science, November 2001.

[vDTdB99] DIJK, STEVEN VAN, DIRK THIERENS und MARK DE BERG: *On the Design of Genetic Algorithms for Geographical Applications*. In: BANZHAF, W., J. DAIDA, A.E. EIBEN, M.H. GARZON, V. HONAVAR, M. JAKIELA und R.E. SMITH (Herausgeber): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)*, Seiten 188–195. Morgan Kaufmann, Juli 1999.

[vDTdB00] DIJK, STEVEN VAN, DIRK THIERENS und MARK DE BERG: *Scalability and Efficiency of Genetic Algorithms for Geometrical Applications*. In: SCHOENAUER, MARC, KALYANMOY DEB, GUNTER RUDOLPH, XIN YAO, EVELYNE LUTTON, JUAN JULIAN MERCELO und HANS-PAUL SCHWEFEL (Herausgeber): *Proc. Parallel Problem Solving from Nature (PPSN VI)*, Band 1917 der Reihe *Lecture Notes in Computer Science*, Seiten 683–692, Paris, September 2000. Springer-Verlag.

[vKSW99] KREVELD, MARC VAN, TYCHO STRIJK und ALEXANDER WOLFF: *Point Labeling with Sliding Labels*. Computational Geometry: Theory and Applications, 13:21–47, 1999.

[vR89] ROESSEL, JAN W. VAN: *An Algorithm for Locating Candidate Labeling Boxes within a Polygon*. The American Cartographer, 16(3):201–209, 1989.

[WS96] WOLFF, ALEXANDER und TYCHO STRIJK: *The Map-Labeling Bibliography*. `www.math-inf.uni-greifswald.de/map-labeling/bibliography`, 1996.

[WTX00] WOLFF, ALEXANDER, MICHAEL THON und YINFENG XU: *A Better Lower Bound for Two-Circle Point Labeling*. In: LEE, D.T. (Herausgeber): *Proc. 11th Annual International Symposium on Algorithms and Computation (ISAAC'00)*, Band 1969 der Reihe *Lecture Notes in Computer Science*, Seiten 422–431, Taipei, 18.–20. Dezember 2000. Institute of Information Science, Academia Sinica, Springer-Verlag.

[WTX02] WOLFF, ALEXANDER, MICHAEL THON und YINFENG XU: *A Simple Factor-$\frac{2}{3}$ Approximation Algorithm for Two-Circle Point Labeling*. International Journal of Computational Geometry and Applications, 2002. Erscheint in Bälde. Siehe auch [WTX00].

[WW95] WAGNER, FRANK und ALEXANDER WOLFF: *Map Labeling Heuristics: Provably Good and Practically Useful*. In: *Proc. 11th Annu. ACM Sympos. Comput. Geom. (SoCG'95)*, Seiten 109–118, Vancouver, 5.–7. Juni 1995.

[WW97] WAGNER, FRANK und ALEXANDER WOLFF: *A Practical Map Labeling Algorithm*. Computational Geometry: Theory and Applications, 7:387–404, 1997.

[WWKS01] WAGNER, FRANK, ALEXANDER WOLFF, VIKAS KAPOOR und TYCHO STRIJK: *Three Rules Suffice for Good Label Placement*. Algorithmica, 30(2):334–349, 2001.

[Yoe72] YOELI, PINHAS: *The Logic of Automated Map Lettering*. The Cartographic Journal, 9:99–108, 1972.

[Zor97] ZORASTER, STEVEN: *Practical Results Using Simulated Annealing for Point Feature Label Placement*. Cartography and GIS, 24(4):228–238, 1997.

[ZQ02] ZHU, BINHAI und ZHONGPING QIN: *New Approximation Algorithms for Map Labeling with Sliding Labels*. Journal of Combinatorial Optimization, 6(1):99–110, 2002.

[Zus01] ZUSE, KONRAD: *Der Computer – Mein Lebenswerk*. Springer-Verlag, Heidelberg, 2001.