Diplomarbeit

Labeling von Graphen

Bearbeiter:
Wolfgang Scholz
am Lehrstuhl für Informatik
mit Schwerpunkt Theoretische Informatik
Universität Passau
scholz@fim.uni-passau.de

Erstgutachter: Prof. Dr. Franz J. Brandenburg
Zweitgutachter: Prof. Christian Lengauer, Ph. D.
Betreuer: Marco Matzeder

11. August 2008

Zusammenfassung

In der wissenschaftlichen Disziplin des Zeichnens von Graphen nimmt die automatische Platzierung von Labels eine Schlüsselrolle ein. Gerade bei sich verändernden Datensätzen ist eine schnelle Visualisierung notwendig, um den Menschen bei der Arbeit zu unterstützen.

Mit dieser Arbeit ist es gelungen, einige unterschiedliche Verfahren für das *Graph Visualization Toolkit* zu implementieren und zueinander in Beziehung zu stellen. Zum Einen baut diese Arbeit auf das von Marco Matzeder erstellte Plugin SPRINGEMBEDDERFR auf und erweitert dieses zur Platzierung von Labels. Zum Anderen wurde ein restriktives Verfahren entwickelt und implementiert, das heuristisch aus einer begrenzten Menge von möglichen Positionen für jedes Label eine Platzierung auswählt.

Durch eine experimentelle Analyse stellt sich heraus, dass eine Kombination beider Verfahren hinsichtlich der Ästhetik die besten Ergebnisse liefert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

1 Einführung

Am Beispiel der automatisierten Platzierung von Beschriftungen lässt sich die Entwicklung des Fachgebiets des Zeichnens von Graphen von den 1950er Jahren bis heute gut nachvollziehen.

Vor der Verfügbarkeit von Computern war die breite Verwendung von Graphen hauptsächlich auf die Kartographie beschränkt. Bei den ersten Überlegungen zum Labeling geht es daher um die automatisierte Anordnung von Namen durch eine Kartiermaschine.

Mit der Erschließung des automatischen Zeichnens von Graphen für andere Fachbereiche gewann auch die theoretische Durchleuchtung des Themas an Bedeutung. Damit einher geht eine Vermehrung der Anwendungsgebiete, sei es durch technische Zeichnungen, in der Biologie oder in der Informatik selbst. Auch die Kartographie gewinnt mit neuen Technologien wie geographischen Informationssystemen weiteres Interesse am Labeling.

Mittlerweile haben sich Heuristiken bei der Lösung der Vielzahl von teils sehr unterschiedlichen Labeling-Problemen durchgesetzt, bei einigen Anwendungen jedoch gewinnen Algorithmen mit Optimalitätsgarantien an Bedeutung.

Diese Arbeit konzentriert sich darauf, zwei entgegengesetzte Herangehensweisen und ihre Kombination zu entwickeln und zu vergleichen. Allerdings will sie auch einen breiten Überblick über vorhandene Entwicklungen geben, um einer zu spezialisierten Darstellung entgegenzuwirken.

1.1 Gliederung

Kapitel 2 geht auf Grundlagen des Labelings von Graphen ein und entwickelt einen Vergleichsmaßstab für die folgenden Verfahren.

Kapitel 3 stellt das wissenschaftliche Umfeld vor und versucht, die verschiedenen Herangehensweisen zu klassifizieren.

Kapitel 4 beschreibt zwei im Umfang der Arbeit entwickelten Verfahren und geht auf ihre theoretischen Eigenschaften ein.

Kapitel 5 evaluiert die implementierten Verfahren experimentell und vergleicht sie anhand unterschiedlicher Problemstellungen.

2 Grundlagen

In diesem Abschnitt werden Grundlagen der Platzierung von Labels dargestellt. Begriffe werden eingeführt, die dem besseren Verständnis der späteren Ausführungen dienen sollen. Weiter wird auf die Bedeutung des Graphlayouts eingegangen und verschiedene Problemstellungen des Labelings angeführt. Einige Herangehensweisen werden diskutiert, die zur Lösung der verschiedenen Probleme beitragen. Schließlich folgt eine kurze Beschreibung der geschichtlichen Entwicklung der automatisierten Platzierung von Beschriftungen.

2.1 Definitionen

Definition. Sei $n \in \mathbb{N}_0$, dann ist $V = \{v_1, \ldots, v_n\}$ eine Menge von *Knoten*. Seien weiter $m \in \mathbb{N}_0$, $j \in \{1..m\}$, $v_{j_1}, v_{j_2} \in V$ und $e_j = \{v_{j_1}, v_{j_2}\}$, dann ist $E = \{e_1, \ldots, e_m\}$ eine Menge von *ungerichteten Kanten* und $G = (V, E)$ ein *Graph*.

Da die Richtung einer Kante für die meisten Labeling Probleme unwesentlich ist, werden der Einfachheit halber alle Kanten als ungerichtet angesehen. Die vorgestellten Algorithmen und Beschreibungen lassen sich auf gerichtete Graphen übertragen.

Definition. Sei $G = (V, E)$ ein Graph und $l \in \mathbb{N}_0$. Seien weiter Σ ein beliebiges Alphabet und $k \in \{1..l\}$. Dann ist $L = \{w_1, \ldots, w_l\}$ mit $w_k = (\alpha^* | \alpha \in \Sigma)$ eine Menge von Labels. Jedem Label wird ein $e \in (G \cup E)$ zugeordnet. Die Abbildung $e : L \to (G \cup E)$ ordnet jedem Label ein *Elterobjekt* zu

Labels (oder auch *Beschriftungen*) an Knoten und Kanten stellen zusätzliche textuelle Informationen dar, die über die durch den Graph abgebildeten Zusammenhänge hinausgehen. Der Knoten, dem ein spezielles Label zugeordnet ist, wird *Elterknoten* genannt, eine entsprechende Kante heißt *Elterkante*. Unter *Graphobjekten* werden Knoten und Kanten zusammengefasst.

Definition. Eine *Darstellung* Γ von G weist jedem Knoten $v_i \in V$ und jedem Label $w_k \in L$ eine *Position* $p_i \in \mathbb{R}^2$ zu. Jede Kante $e_j \in E$ verbindet als Linie oder Jordankurve die beiden Knoten v_{j_1} und v_{j_2}.
$\Gamma(G)$ und $\Gamma(L)$ sind wie folgt definiert:

$$\Gamma \begin{cases} V \to \mathbb{R}^2 \\ E \to (J[0,1] \to \mathbb{R}^2 | J \text{ ist Jordankurve}) \\ L \to \mathbb{R}^2 \end{cases}$$

$J = \Gamma(\{u, v\})$ erfüllt dabei folgende Eigenschaften:

- J ist stetig und injektiv (Definition der Jordankurve)

- $J(0) = \Gamma(u)$ und $J(1) = \Gamma(v)$

Eine Darstellung dient dazu, Informationszusammenhänge flächig darzustellen. Die zu betrachtenden Aspekte werden durch Knoten abgebildet. Kanten stellen Zusammenhänge zwischen bestimmten Knoten her.

Für den Menschen bieten Graphen den Vorteil, sowohl den globalen Informationszusammenhang überschauen zu können, als auch beliebige lokale Informationen einzusehen. Wird im Graphlayout darauf geachtet, möglichst kurze Kanten zu produzieren, so enthalten Gruppierungen von Knoten Informationen über deren Zusammengehörigkeit. Für die Identifizierbarkeit von Knoten sind die diesen zugeordneten Labels verantwortlich.

Definition. Die *Distanzfunktion* $d : (L \times (G \cup E)) \to \mathbb{R}$ eines Labels zum Elterobjekt wird folgendermaßen definiert:

$$d(w, e(w)) := \begin{cases} \sqrt{(\Gamma(e(w))_x - \Gamma(w)_x)^2 + (\Gamma(e(w))_y - \Gamma(w)_y)^2} & e(w) \in G \\ \min \sqrt{(\Gamma(e(w))_x - \Gamma(w)_x)^2 + (\Gamma(e(w))_y - \Gamma(w)_y)^2} & e(w) \in E \end{cases}$$

$$\text{wobei } (a_x, a_y) = a \in \mathbb{R}^2$$

Als Abstandsmetrik wird die euklidische verwendet. Die Definition geht von punktförmigen Labels und Knoten aus. In der Praxis werden Labels und Knoten zumindest als rechteckförmig angenommen. Die Distanz zum Elterobjekt ist dabei durch den kürzesten Abstand zwischen Label und Elterobjekt gegeben. Eine *Überlappung* findet statt, wenn es je einen Punkt bei Label und Elterobjekt gibt, die eine Distanz von 0 haben oder ein Objekt das andere verdeckt.

Überlappungen zwischen Label und beliebigem Graphobjekt sowie zwischen Labels sind ähnlich definiert.

Definition. Eine *Positionierung* weist jedem Objekt $a \in G \cup L$ seine Darstellung $\Gamma(a)$ zu. Es wird versucht, dass $\Gamma(a)$ gewissen Kriterien genügt.

Bei der Positionierung von Beschriftungen stehen oft mehrere Platzierungsmöglichkeiten zur Verfügung. Diese werden *Positionskandidaten* benannt.

2.2 Kriterien zur Platzierung von Labels

Damit Informationszusammenhänge in einem Graph schnell ersichtlich werden, ist es beim Graphlayout -wie oben erwähnt- nötig, miteinander verbundene Knoten in räumlicher Nähe zu platzieren. Diese Arbeit befasst sich mit dem allgemeinen Layouting von Graphen nur peripher, daher wird der geneigte Leser auf entspechende Literatur verwiesen [BETT98, Mat06].

Die Beschriftung von Landkarten ist ein klassisches Problem in der Kartographie. Zusätzlich zur geographisch exakten Position der einzelnen Knoten wird

textuelle Information dargestellt, welche deren Eigenschaften beschreibt. Laut den Kartographen Imhof [Imh75] und Yoeli [Yoe72] sind dies die wichtigsten Kriterien für die Platzierung von Beschriftungen:

- Eine eindeutige Zuordnung einer Information zum entsprechenden Knoten ist möglich,
- Texte überlappen keine anderen Objekte und
- die Schriftgröße ist lesbar.

Ähnliche Kriterien gelten bei der Positionierung von Labels in allgemeinen Graphen. Eine Platzierung, die diesen Kriteren genügt, wird auch als *gültig* bezeichnet.

2.2.1 Eindeutigkeit bei der Zuordnung

Da Labels Informationen zu einem Graphobjekt enthalten, ist bei ihrer Platzierung besonders wichtig, dass eine eindeutige Zuordung zum Elterobjekt hergestellt werden kann (siehe dazu Abbildungen 1 und 2). Räumliche Nähe zum Elterobjekt wirkt dabei förderlich, Nähe zu anderen Objekten des Graphen oder anderen Labels beeinflussen die Zuordnung negativ.

Abbildung 1: Irreführende Platzierung von Labels

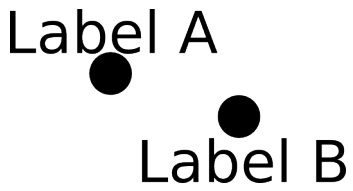

Abbildung 2: Eindeutige Zuordnung von Labels möglich: Der Platzverbrauch ist größer

2.2.2 Überlappungsfreiheit

Wird eine Schrift durch andere graphische Objekte überdeckt, verringert sich ihr Informationsgehalt. Daher ist die Überlappungsfreiheit der eindeutigen Zuordnung in den meisten Fällen vorzuziehen, wenn nicht beides eingehalten werden

kann. Wenn ein Label ein anderes graphisches Objekt -wie etwa einen Knotenverdeckt, so kann zusätzlich zur schlechteren Lesbarkeit die Information des Objektes für den Betrachter verloren gehen. Da Kanten im Regelfall durch eine Überlappung wenig an sichtbarer Information verlieren, wird die Überlappung mit Knoten als gravierender als die mit Kanten angesehen. Abbildung 3 veranschaulicht entsprechende Überlappungen.

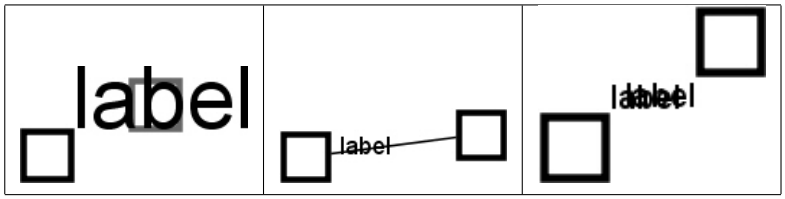

Abbildung 3: Verschiedene Überlappungen

2.2.3 Lesbare Schriftgrößen

Eine Möglichkeit, sowohl eindeutige Zuordnung zu garantieren, als auch Überlappungen auszuschließen, ist es, überlappende Beschriftungen kleiner zu skalieren. Dabei wird allerdings sowohl die Lesbarkeit von Labels eingeschränkt, als auch die Ästhetik der Darstellung beeinträchtigt. Ein Beispiel hierfür zeigt Abbildung 4.

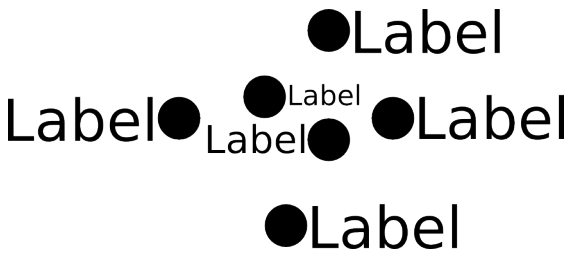

Abbildung 4: Überlappungsfreiheit durch Verkleinerung der Schriftgrößen

Im Unterschied zur Kartographie ist es bei einigen Anwendungen möglich, Überlappungen durch eine Veränderung des Graphlayouts aufzulösen. Dabei werden Knoten nach außen gerückt, so dass mehr Platz für entsprechende Beschriftungen entsteht. Wie Abbildung 5 zeigt, wird dabei im Allgemeinen der Flächenbedarf der Darstellung vergrößert.

Ein Verschieben ist einer Verkleinerung von Schriftgrößen vorzuziehen, da Problembereiche der Darstellung entschärft werden. Dies ist auch dann noch der Fall, wenn für das Layout nur eine begrenzte Fläche verfügbar ist. Da nach der Überlappungsauflösung die Darstellung in die Fläche eingepasst werden muss,

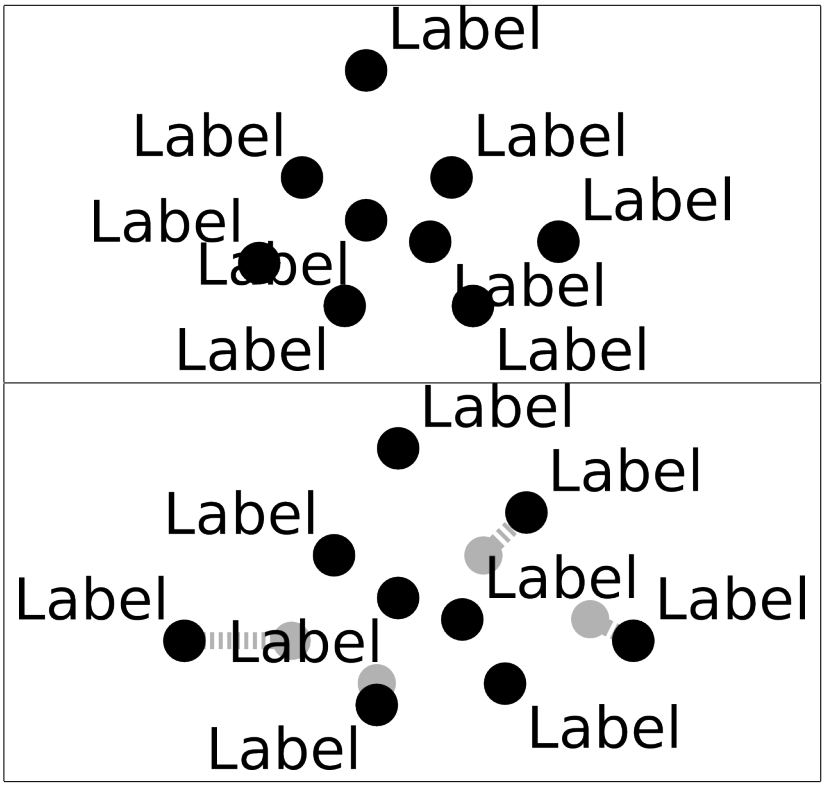

Abbildung 5: Durch Verschieben lassen sich Überlappungen auflösen.

werden Größen und Abstände aller Graphobjekte gleichermaßen verändert. Werden dagegen nur einzelne Labels kleiner skaliert, so wird die Lesbarkeit lokal stärker beeinträchtigt.

Manche Anwendungen erlauben es, dass einige nicht platzierbare Labels verworfen werden. Wie Abbildung 6 zeigt, geht dabei zwar die Information der ausgeblendeten Beschriftung verloren, allerdings wird dafür andere Information wieder für den Betrachter erkennbar.

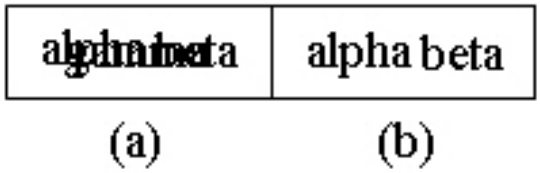

Abbildung 6: (a) Bei der Überlappung ist die Lesbarkeit aller beteiligten Schriftzüge eingeschränkt. (b) Durch Löschen eines Labels können andere wieder lesbar gemacht werden.

2.3 Problemstellungen beim Labeling

Ebenso wie die Unterschiedlichkeit der darzustellenden Information ist die Anzahl der verschiedenen Problemstellungen bei der Platzierung von Beschriftungen mannigfaltig. Es gestaltet sich als schwierig, generell gültige Kriterien für eine automatische Platzierung von Labels zu finden. Verschiedenen Anforderungen muss Rechnung getragen werden. Daher existieren verschiedene Lösungsansätze zu dem Thema.

2.3.1 Point Feature Label Placement Problem

Eines der grundlegendsten und im wissenschaftlichen Umfeld am öftesten erwähnten Probleme bei der automatischen Platzierung von Labels ist das diskrete *Point Feature Label Placement Problem* (PFLP). Dabei werden punktförmige Graphobjekte mit jeweils einem Label versehen, für das eine endliche Anzahl von möglichen Positionen zur Verfügung steht (siehe Abbildung 7). Häufig wird zur Vereinfachung angenommen, dass alle Labels rechteckig und von gleicher Größe sind. Weiter wird in der Literatur meist angenommen, dass sich das zu beschriftende punktförmige Objekt in einer der Ecken des das Label umgebenden Rechtecks befindet. Dieser Spezialfall wird für die weitere Verwendung *4-Point Feature Label Placement Problem* benannt. Ziel ist, bei einer gegebenen Menge von Punkten und einer gegebenen Labelgröße unter den vorhandenen Wahlmöglichkeiten die Beschriftungen so zu platzieren, dass die Anzahl der Überlappungen minimiert wird.

Unter dem binären *Point Feature Label Placement Problem* (BPFLP) wird die Fragestellung verstanden, ob eine überlappungsfreie Platzierung möglich ist - also die Anzahl der Überlappungen 0 ist. Die NP-Vollständigkeit von BPFLP wurde von mehreren unabhängigen Forschungsgruppen bewiesen ([MS91], [FW91]).

Falls nun PFLP in polynomieller Zeit lösbar wäre, würde es ausreichen, die Anzahl der Überlappungen zu zählen und gegen 0 zu vergleichen, um BPFLP zu simulieren. Da allerdings im Allgemeinen angenommen wird, dass $P \neq NP$, ist dies als Widerspruch zu werten. Demnach ist also das *Point Feature Label Placement Problem* ebenfalls NP-vollständig.

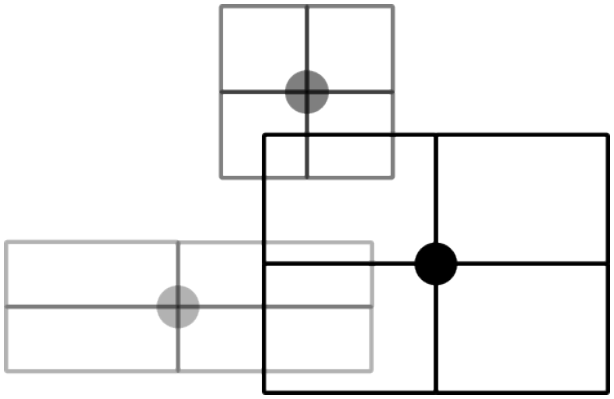

Abbildung 7: Das diskrete *Point Feature Label Placement Problem* sieht für jedes punktförmige Graphobjekt eine Anzahl (hier 4) unpriorisierte Positionskandidaten für die Beschriftung vor.

Stehen nur 2 verschiedene Positionen zur Platzierung jeder Beschriftung zur Wahl, ist eine Reduktion des Problems BPFLP auf 2-SAT möglich. Wagner und Formann geben hierfür einen $O(n^2)$ Algorithmus an [FW91]. Imai und Asano zeigen einen einen Algorithmus mit $O(n \log^2 n)$ Laufzeit, der die geometrischen Eigenschaften des Problems ausnutzt [IA86].

2.3.2 Elastic Labeling Problem

Beim *Elastic Labeling Problem* handelt es sich um eine Alternative zum Problem des PFLP, bei der Überlappungen nicht durch selektive Positionierung der Beschriftungen vermieden werden, sondern durch Veränderungen ihrer Form. Dabei werden Labels zwar weiterhin als Rechtecke individueller Größe angesehen, allerdings können Höhe und Breite mit Hilfe von Zeilenumbrüchen variiert werden. Es wird angenommen, dass Flächeninhalte der Beschriftungen jeweils konstant bleiben. Außerdem wird eine Ecke des Rechtecks jedes elastischen Labels als *Aufhängungspunkt* ausgewiesen, der trotz Formänderung an konstanter Position verbleibt.

Definition. Ein *elastisches Rechteck* ε ist eine Familie aus Rechtecken, gegeben durch ein Quintupel (p, α, H, W, Q), wobei p ein allen Rechtecken aus ε gemeinsamer Punkt im zweidimensionalen Raum ist, α ist die Fläche jedes Rechtecks, $H = [H^{\min}, H^{\max}]$ die Spanne der ihrer Höhen, $W = [W^{\min}, W^{\max}]$ die Spanne der Breiten, $Q \subseteq \{1, 2, 3, 4\}$ ist eine Menge von möglichen Ecken, die p darstellen kann. Ein Wert von 1 bedeutet, dass sich p auf der linken unteren Ecke befindet, bei 2 auf der linken oberen Ecke, 3 auf der rechten oberen und 4 auf der rechten unteren Ecke [IV99].

Elastische Labels verwenden die Form von elastischen Rechtecken. Wie Abbildung 8 zeigt, gibt es eine Anzahl verschiedener möglicher Formen eines elastischen Labels. In der Praxis kann allerdings nur eine ganzzahlige Anzahl von Zeilen gewählt werden, was wiederum die Anzahl dieser Formen auf einen endlichen Wert beschränkt.

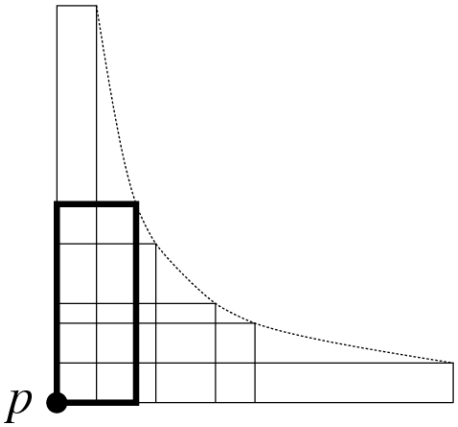

Abbildung 8: Ein elastisches Labels (aus [IV99])

Das *Elastic Labeling Problem* lässt sich mit PFLP kombinieren, indem sowohl mehrere Positionen als auch elastische Labels verwendet werden. Eine solche Kombination ist als Verallgemeinerung von PFLP-Problems wiederum NP-vollständig. Wie Iturriaga-Velazquez zeigt, bleibt das Problem sogar dann NP-vollständig, wenn keine Wahlmöglichkeit hinsichtlich der Position des Labels besteht (sogenanntes *One-Corner Elastic Labeling Problem*) [IV99].

Falls allerdings als zusätzliche Einschränkung der Aufhängungspunkt aller elastischen Labels in der gleichen Ecke liegt (Q hat nur einen und bei allen Labels den selben Wert), etwa links unten, so existiert ein Polynomialzeitalgorithmus zur Lösung des Problems [IV99].

2.3.3 Multiple Label Placement Problem

In der Kartographie entsteht vor allem bei längeren linienförmigen Objekten die Notwendigkeit, mehrere Beschriftungen pro Objekt zuzulassen, die in mehr oder weniger regelmäßigen Abständen zu platzieren sind. Ebenso besteht dieses Problem, falls mehr als eine Eigenschaft eines Graphobjekts textuell dargestellt werden soll.

Außer den in Abschnitt 2.2 genannten allgemeinen Kriterien müssen bei der Platzierung mehrerer Labels zusätzliche Kriterien zur relativen Positionierung der Labels untereinander gefunden werden. In der Kartographie liegt eine langjährige Praxis im Umgang mit mehreren Labels vor, auch ist durch den begrenzten Anwendungsbereich der intuitive Sinn einer entsprechenden Verwendung bekannt. Daher ist es vermutlich möglich, allgemeine zusätzliche Kriterien zu finden.

Bei technischen Zeichnungen oder Karten allerdings sind geeignete Zusatzkriterien je nach Verwendungshintergrund sehr unterschiedlich. Daher ist es wichtig, bei der Formulierung eines entsprechenden Algorithmus zusätzliche Parametrisierungsmöglichkeiten zu berücksichtigen.

Kakoulis und Tollis unterscheiden drei wesentliche Typen von Nebenbedingungen zur Platzierung mehrerer Labels [KT06]:

- **Nähe**: Sofern ein Label eine besondere Beziehung zwischen dem Elterobjekt und einem mit ihm verbundenen anderen Objekt des Graphen beschreibt, ist eine Platzierung in räumlicher Nähe zur Verbindungsstelle vorteilhaft. Abbildung 9 verdeutlicht eine solche Situation.

- **Teilweise Ordnung**: Sofern ein Label eine besondere Beziehung zwischen dem Elterobjekt und einem mit ihm verbundenen Objekt des Graphen beschreibt, ist eine Platzierung in räumlicher Nähe zu einer Verbindungsstelle zu einem anderen Graphobjekt irreführend. Abbildung 10 verdeutlicht dies.

- **Priorität**: Falls eine große Anzahl von Labels für ein Elterobjekt auftreten, ist es eventuell sinnvoll, gewisse Labels bei der Platzierung zu benachteiligen, um eine Lesbarkeit zu gewährleisten. Dafür ist eine Priorisierung der Beschriftungen erforderlich.

2.3.4 Sliding Labels Problem

Das *Sliding Labels Problem* ist eine Verallgemeinerung des *Point Feature Label Placement Problem*s. Als mögliche Positionen werden alle diejenigen gewertet, bei denen irgendein Punkt des das Label umgebenden Rechtecks den zu beschriftenden Punkt berührt [Hir82]. Dies sind im Regelfall unendlich viele. Abbildung 11 zeigt ein gültiges Labeling nach einer Instanz des *Sliding Labels Problem*.

Als eine Verallgemeinerung des *Point Feature Label Placement* ist auch das *Sliding Labels Problem* NP-vollständig. Wie Iturriaga-Velazquez zeigt, bleibt das

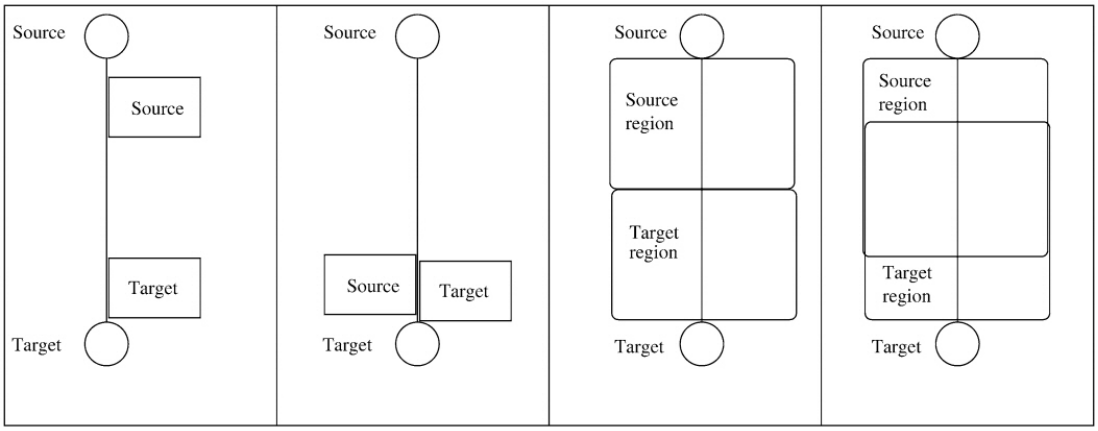

Abbildung 9: Mehrere Labels an einem Graphobjekt mit individuellen Positionierungspräferenzen (entnommen aus [KT06]): (a) vorteilige Platzierung. (b) irreführende Platzierung. (c) strikte Positionierungskriterien. (d) überlappende Positionierungskriterien.

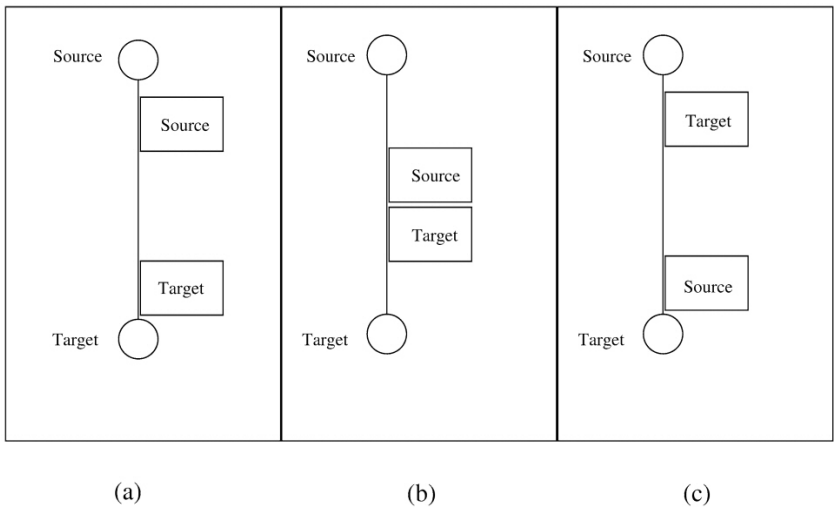

Abbildung 10: Bedeutung der Einhaltung einer teilweisen Ordnung bei mehreren Labels (entnommen aus [KT06]): (a) vorteilige Platzierung. (b) akzeptable Platzierung. (c) irreführende Platzierung.

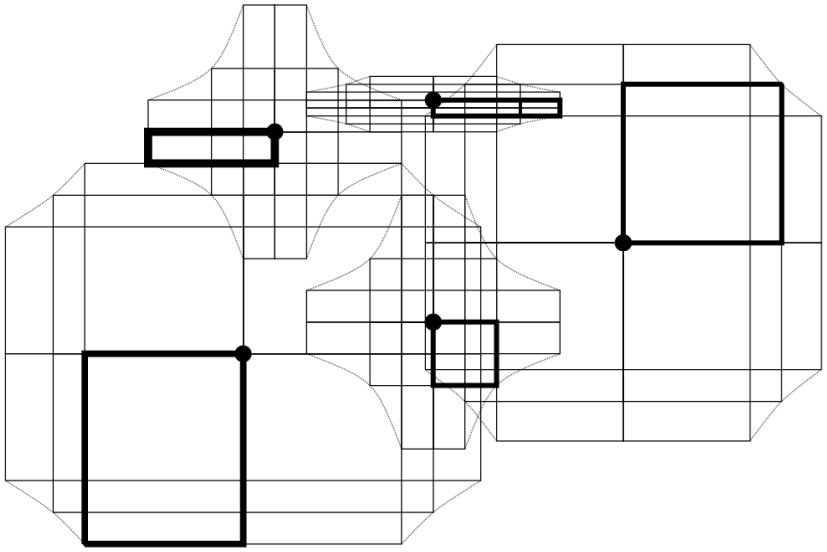

Abbildung 11: Eine gültiges Labeling nach einer Instanz des *Sliding Labels Problem* (aus [IV99])

Problem sogar dann NP-vollständig, wenn nur die Punkte links oder rechts des umgebenden Rechtecks am zu beschriftenden Punkt platziert werden dürfen (sogenanntes *Left-Right Sliding Labels Problem*). Erst, wenn nur noch eine Seite des Rechtecks am zu beschriftenden Punkt platziert werden darf, kann das Problem mit Hilfe eines *sweep-line-Algorithmus* in polynomieller Zeit gelöst werden [IV99].

2.3.5 Edge Label Placement Problem

Die Platzierung von Kantenlabels ist -zumindest in der Kartographie- weniger restriktiv als die Platzierung von Knotenlabels. Normalerweise sind Kanten lang genug, so dass aus vielen akzeptablen Positionen gewählt werden kann. Allerdings kann vor allem in technischen Zeichungen die Kantendichte stark variieren. Gerade bei nicht-planaren Graphen werden hohe Ansprüche an den Layoutalgorithmus gestellt, da eine Platzierung von Beschriftungen in der Nähe eines Schnittpunktes zweier Kanten leicht zu Zuordnungsschwierigkeiten führt.

Für Kantenlabels gelten im Allgemeinen die selben ästhetischen Richtlinien wie in Abschnitt 2.2 erläutert. Um bei horizontaler Schriftführung eine eindeutige Zuordnung zu erhalten, empfiehlt es sich, dass zumindest eine Ecke des die Beschriftung umgebenden Rechtecks auf der zugeordneten Kante liegt. Eine mit der Kante überlappende Positionierung kann zu einer Beeinträchtigung der Lesbarkeit führen und sollte daher eher vermieden werden. Abbildung 12 zeigt mögliche Platzierungen. Yoeli schlägt vor, die Beschriftung parallel zur Kante auszurichten [Yoe72]. Dies bringt eine zusätzliche Sicherheit bei der Zuordnung mit sich.

Allerdings ist die Lesbarkeit von schräger oder senkrechter Schrift eingeschränkt.

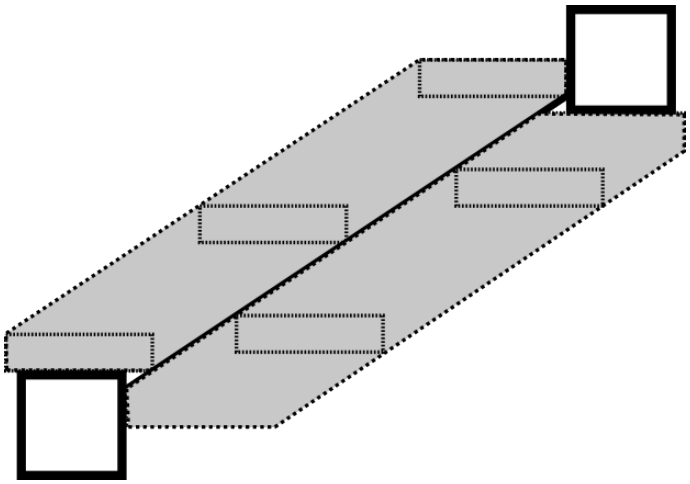

Abbildung 12: Platzierung von Kantenlabels

Kakoulis und Tollis zeigen, dass eine direkte Übertragung des *Point Feature Label Placement Problem* auf das *Edge Label Placement Problem* nicht sinnvoll ist [KT01]. Dies liegt im Wesentlichen daran, dass zur Beschriftung von Punkten keine Kanten notwendig sind, Kanten jedoch über die Existenz von Knoten definiert sind. Zur Vermeidung von Überlappungen müssen bei der Beschriftung von Kanten zumindest auch die Knoten berücksichtigt werden. Daher kann das *Edge Label Placement Problem* nicht mit ähnlicher Abstraktion theoretisch erfasst werden.

Wegen den freieren Platzierungsmöglichkeiten bei Kantenlabels führen Kakoulis und Tollis eine Kostenfunktion zur Bewertung von Positionskandidaten ein [KT01]. Bei dem *Admissible Edge Label Placement Problem* geht es um die Frage, ob ein Labeling existiert, bei dem die Kostenfunktion zu 0 auswertet.

Das *Discrete Admissible Edge Label Placement Problem* fordert als zusätzliche Einschränkung, dass sich keine der möglichen Positionen überlappen. Dadurch ist die Menge der Positionskandidaten auf jeden Fall endlich, da auch die Länge der Kante endlich ist. Kakoulis und Tollis zeigen die NP-Vollständigkeit des *Discrete Admissible Edge Label Placement Problem* [KT01].

2.3.6 Rectangle Perimeter Labeling Problem

Bei der Verwendung von geographischen Informationssystemen wird oft nur ein Ausschnitt des verfügbaren Datensatzes visuell präsentiert. Durch die zunehmende Verfügbarkeit und Integration dieser Systeme - etwa in Navigationssystemen - werden zusätzlich Realzeiteigenschaften gefordert. Der angezeigte Datenausschnitt bewegt sich mit einem Objekt mit und zeigt die unmittelbare geographi-

sche Umgebung. Weiter entfernt liegende, hervorgehobene Objekte sollen währenddessen mitverfolgt werden können. Informationen werden hierzu am Rand der anzeigbaren Fläche dargestellt.

Das *Rectangle Perimeter Labeling Problem* liegt vor, wenn mehrere Labels am Rand einer rechteckigen Fläche überlappungsfrei dargestellt werden sollen. Jede solche Beschriftung wird in der hier behandelten Version des Problems als Rechteck beschrieben und besitzt ein zugehöriges Graphobjekt, das sich jenseits der darstellbaren Fläche befindet.

Jeder Beschriftung wird eine Seite des darstellbaren Rechtecks als *Basis* zugeordnet: An dieser Seite muss das Label anliegen, um zu einer akzeptablen Lösung beizutragen. Dazu wird diejenige Seite gewählt, deren Schwerpunkt dem zugehörigen Objekt am nächsten ist.

Iturriaga-Velazquez zeigt, dass es zur Lösung des Problems ausreicht, das darstellbare Rechteck in eine polynomiell große Anzahl an Unterteilungen zu zerlegen, so dass in jeder dieser Unterteilungen zur Platzierung nur Labels von insgesamt zwei Seiten des Rechtecks berücksichtigt werden müssen [IV99]. Abbildung 13 zeigt eine solche Zerteilung.

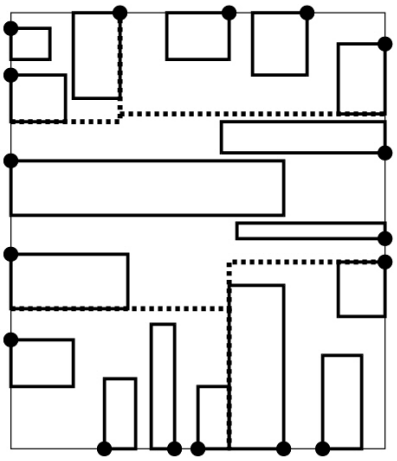

Abbildung 13: Eine Zerteilung des darstellbaren Rechtecks (aus [IV99])

2.3.7 Weitere Problemstellungen

Eine Vielzahl von weiteren Abwandlungen des Problems der Platzierung von Beschriftungen treten in der Praxis auf.

Kombination verschiedener Beschriftungen Oft tragen verschiedene Graphobjekte Beschriftungen. Knoten-, Kanten- und Randlabels haben unterschiedliche Positionierungspräferenzen. Ein Platzierungsalgorithmus hat

hierbei die verschiedentlichen Platzierungskriterien untereinander abzuwägen.

Knotenform Allen Knoten gemeinsam ist, dass sie in relativer Nähe des Elterknotens zu platzieren sind. Für die Berechnung des Abstands ist die Form des Knotens von Bedeutung. Außer Punkten, Kreisen und Rechtecken können je nach Anwendung auch beliebige Polygone die Form von Knoten bestimmen.

Kantenform Kanten sind nicht durchweg geradlinig. In manchen Anwendungen sind Multilinen oder gekrümmte Linien für ein überschneidungsfreies Graphlayout oder zur Darstellung geographischer Eigenschaften unverzichtbar. Bei der Platzierung von Kantenlabels muss die Kantenform berücksichtigt werden, sowohl um die unmittelbare Nähe zur Kante zu gewährleisten, aber auch, um Überschneidungen zwischen Labels und Kanten zu vermeiden.

Verschiebbarkeit des Graphlayouts Anders als bei geographischen Anwendungen spielt bei manchen technischen Graphen das Graphlayout gegenüber der Lesbarkeit eine untergeordnete Rolle. Falls in einem Bereich des Graphen eine Häufung von Beschriftungen vorliegt, kann es sinnvoll sein, die Graphknoten etwas zu verschieben, um Platz für eine lesbare Labelplatzierung zu schaffen. Eventuell werden Beschriftungen gegenüber den übrigen Graphobjekten auch gleichbehandelt und gemeinsam positioniert.

Unterschiedliche Objektgrößen Abgesehen von Labels können auch zu beschriftende Knoten unterschiedliche Größe besitzen oder Kanten unterschiedliche Dicke haben. Im Allgemeinen kann davon ausgegangen werden, dass größere Objekte mehr Möglichkeiten zur Platzierung von Beschriftungen bieten.

Variierbarkeit der Form von Labels Einige Anwendungen erlauben eine Variation der Form von Beschriftungen. Zeilenumbrüche beeinträchtigen bei Beschriftungen mit mehreren Wörtern in den meisten Fällen die Lesbarkeit nicht. Ein Abweichen von einer horizontalen Schriftführung kann zu einer Verschlechterung des Layouts führen, in einigen Anwendungen allerdings sogar die Zuordnung erleichtern [Yoe72]. Ein elliptisches Modell von Beschriftungen anstelle eines rechteckigen kann für die Zuordnung zum Elterobjekt vorteilhafter sein [DMM+97]. In geringem Maße kann eine Verkleinerung der Schriftgröße zur Erleichterung der Platzierung manchmal toleriert werden. Allerdings hat vor allem bei geographischen Karten die Schriftgröße oft eine genau bemessene Bedeutung, zum Beispiel bildet sie die Einwohnerzahl der beschrifteten Stadt ab. Daher sollten andere Methoden zur Auflösung von Überlappungen vorgezogen werden.

Flächige Knoten Die Zuordnung zum Knoten ist leicht möglich, wenn ein Label in dessen Innerem platziert werden kann. Allerdings sollte darauf geachtet werden, dass die Schrift nicht mit dem Knotenrand überlappt. Wenn nun in einem Graph nicht alle Beschriftungen innerhalb ihrer Elterknoten platziert werden können, erzeugt die unterschiedliche Positionierungsstrategie eine scheinbare Zweiteilung der Knotenmenge, die auf eine eventuell nicht beabsichtigte Semantik schließen lässt. Die Abwägung ob im Knoteninneren platziert wird oder nicht, hängt also von der gegebenen Anwendung ab.

Dreidimensionales Layout In einigen technischen Anwendungen kommt Dreidimensionalität zum Einsatz, wenn die Unterschiede mehrerer Graphen herausgearbeitet werden sollen. Dabei werden mehrere nahezu gleiche Graphen überlagert dargestellt. Die dritte Dimension wird hierbei nur bedingt ausgenutzt, da es sich um eine endliche Anzahl von Schichten handelt.

Daneben gibt es auch Anwendungen, bei denen Graphen voll dreidimensional gelayoutet werden. Beschriftungen sollten hierbei dynamisch je nach Kameraposition platziert werden, damit Elterknoten bzw. Elterkanten nicht mit der Schrift überlappen.

3 Labeling-Verfahren

Verschiedene Entwicklungen finden im Umfeld der automatisierten Platzierung von Labels statt. Es werden Algorithmen aus den unterschiedlichen Anwendungsgebieten vorgestellt.

3.1 Historische Entwicklung

Der Themenbereich des automatischen Labeling von Graphen knüpft an Forschungsergebnisse an, die in der Kartographie seit den 1960-er Jahren erbracht wurden. Die automatisierte Darstellung von Graphen zur Visualisierung von Daten gewinnt mit zunehmenden technischen Möglichkeiten an Bedeutung. Aus einer anwendungsorientierten Sicht in der Kartographie entwickelt, gewinnt mit einer Verallgemeinerung des Themenbereichs die Theorie an Bedeutung.

Die historische Entwicklung des Labeling von Graphen lässt sich anhand von drei Stationen beschreiben:

- Der Stand in der Kartographie in den 1970-er Jahren
- Die aktuelle theoretische Erfassung des Themenbereichs
- Die aktuelle Praxis in der Kartographie und in der allgemeinen Verwendung von Graphen

3.1.1 Kartographie

Die Platzierung von Beschriftungen spielt in der Kartographie eine bedeutende Rolle. Bereits in den 1960-er Jahren und davor gab es wissenschaftliche Überlegungen, diesen Prozess zu automatisieren (siehe hierzu [Wit58, Imh62]).

Nach Yoeli [Yoe72] wird 1972 noch ein Großteil des Zeitbedarfs zur Erstellung neuer Landkarten auf die Platzierung der Namen verwendet. Dabei beschreibt Yoeli zwei wesentliche Aufgabenfelder, die von Hand durchgeführt werden müssen:

- Die Bestimmung der Inhalte der Beschriftungen und
- die eigentliche Platzierung derselben.

Für die Inhalte schlägt Yoeli eine Datenbank vor, die Koordinaten Beschriftung und Typ von geographischen Merkmalen zuordnet. Ebenfalls gibt ein Wert die Bedeutung eines Merkmals wieder, woraus die Größe der Beschriftung abgeleitet wird.

Im wissenschaftlichen Fokus der Beschriftung befinden sich dabei mehrere verschiedene Arten von geographischen Merkmalen (entnommen aus [Yoe72] und [Imh75]):

- Punktförmige Graphobjekte

- Linien und Linienzüge
- Flächige Objekte

Es werden jeweils unterschiedliche Kriterien für die Platzierung aufgeführt, die nachfolgend erläutert werden. Allen gemeinsam ist, dass sich die Beschriftungen nicht mit geographischen Merkmalen überschneiden dürfen und durch Größe und Schriftform eine Zuordnung erleichtern sollen.

Für die Ausrichtung von Beschriftungen punktförmiger Objekte legt Yoeli Prioritäten fest (siehe Abbildung 14). Eine Platzierung am rechten oberen Rand eines Objekts kann demnach intuitiv am besten zugeordnet werden. Damit die Objektmarkierung nicht als ein Teil der Beschriftung fehlinterpretiert wird, wird eine Positionierung links oder rechts des Objekts ausgeschlossen. Außerdem ist darauf zu achten, dass zwischen Beschriftungen von gleicher Farbe ein Abstand eingehalten wird, um sie visuell voneinander abzugrenzen.

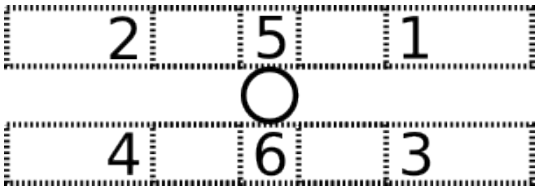

Abbildung 14: Prioritäten verschiedener Labelpositionen bei Knoten nach Yoeli ([Yoe72])

Als Linien oder Linienzüge werden graphisch vor allem Straßen und Flüsse dargestellt. Für die Platzierung von Beschriftungen eignen sich dabei geradlinige Abschnitte des Streckenverlaufs. Die Schrift ist gedreht zu positionieren, so dass sie dem entsprechenden Kurvenverlauf folgt.

Bei der Unterscheidung zwischen punktförmigen und flächigen Objekten sind der Maßstab der Karte und die Größe des Objekts heranzuziehen. Die Beschriftung flächiger Objekte wird im Objekt platziert. Wie in Abbildung 15 sichtbar, stellt die Betrachtung des Objekts als Rechteck eine zu grobe Vereinfachung dar, so dass zur Vermeidung von Überschneidungen ein feineres Modell des Umrisses gewählt werden muss.

Bei großen Karten kann es durch die notwendige Abbildung von sphärischen Koordinaten auf die Kartenfläche zu einer erkennbaren Krümmung des geographischen Koordinatengitters kommen. Yoeli schlägt vor, Beschriftungen ebenfalls zu krümmen, und gibt zusätzlich einen Algorithmus für Ausgabegeräte an, die keine Schriftkrümmung unterstützen.

Abbildung 15: Überschneidung bei der Beschriftung flächiger Objekte (entnommen aus [Yoe72])

3.1.2 Theoretische Erfassung

Während der Themenbereich um das Labeling von Graphen für die Kartographie bereits als erschlossen gilt, so erfährt er durch die zunehmende Verwendung von technischen Visualisierungen neue Aufmerksamkeit. In der Technik sind die Problemstellungen zum Teil sehr unterschiedlich. Neu entwickelte Verfahren werden daher aus speziellen Anwendungen heraus realisiert [IL99, WW97].

Wie im Abschnitt 2.3.1 angegeben, wurde der Beweis zur NP-Vollständigkeit des allgemeinen *Point Feature Label Placement Problem* erst um 1990 erbracht. Heuristiken mit bewiesener Optimalität werden nach diesem Zeitpunkt datiert, wie etwa Formann und Wagner [FW91] oder Wagner und Wolff [WW97].

Einerseits durch die fortlaufende Entwicklung in der Rechnerarchitektur, andererseits durch Verfahrensverbesserungen bei der vollständigen Lösung des *Point Feature Label Placement Problem* nimmt die Problemgröße, die in akzeptabler Zeit bewältigt werden kann, zu.

Strijk, Verweij und Aardal stellen einen Algorithmus vor, der bewiesenermaßen maximale Labelgrößen für überschneidungsfreie Platzierungen von Beschriftungen für das *4-Point Feature Label Placement Problem* findet [SVA00]. Als Einschränkungen werden global gleich große, rechteckige Labels angenommen. Mittels empirischer Tests wird dargelegt, dass ein optimales Labeling für 950 Städte in akzeptabler Zeit berechnet werden kann.

3.1.3 Aktuelle Praxis

Verfahrensweise In der Praxis haben sich verschiedene heuristische Suchverfahren bei der Platzierung von Labels durchgesetzt. Als die Wichtigsten bezeichnen Edmondson, Christensen, Marks und Shieber [ECMS96]:

- Physikalische Relaxation

- Dynamische Priorisierung einer diskreten Menge von möglichen Positionen
- Gradientenabstieg
- Simulated Annealing

In einem Vergleich stellen Christensen, Friedman, Marks und Shieber *Simulated Annealing* als die am besten geeignete Heuristik fest. Heuristiken, die mit Tiefensuche arbeiten, sind den oben genannten Verfahrensweisen unterlegen [CFMS97].

Funktionalität Edmondson, Christensen, Marks und Shieber stellen einen Algorithmus vor, der sowohl den Stand der Technik an Funktionalität bietet, als auch aktuelle Heuristiken verwendet [ECMS96]. Das Verfahren wird in drei Phasen eingeteilt:

Generierung möglicher Positionen Je nach Art des zu beschriftenden Graphobjekts werden eine endliche Anzahl an möglichen Positionen generiert.

Evaluierung der möglichen Positionen Die generierten Positionen werden anhand ihrer Lage und der Überschneidung mit Graphobjekten oder anderen Beschriftungen bewertet.

Positionsselektion Es werden die Positionen zur Platzierung der Beschriftungen gewählt, welche die Gesamtwertung aller Platzierungen optimieren.

Eine obere Schranke für die Anzahl generierter möglicher Positionen pro Label wird eingehalten, um den Berechnungsaufwand bei der Positionsselektion gering zu halten. Außerdem ist darauf zu achten, dass die Menge der generierten Positionen ausreichend unterschiedliche Lösungen anbietet, um echte Alternativen bei der Selektion zu erhalten. Dem Tradeoff zwischen der Güte der Positionskandidaten und der Schnelligkeit ihrer Auffindung muss ebenfalls Rechnung getragen werden.

Sowohl bei flächigen als auch bei linienförmigen Graphobjekten wird zur Generierung von Positionskandidaten folgendes Schema eingehalten [ECMS96]:

1. Generierung einer größeren Anzahl Positionskandidaten
2. Verwerfen einiger Kandidaten anhand von einigen schnell zu berechnenden Metriken
3. Evaluierung der restlichen Kandidaten mit allen verfügbaren Metriken
4. Selektion einer vorher festgelegten Anzahl der besten Kandidaten zur weiteren Verwendung

Linienförmige Graphobjekte werden als Polygonzug modelliert. Krümmung der Linie und durchschnittliche Distanz zu den Buchstaben der Beschriftung werden einbezogen, um mögliche Positionen zu generieren. Falls ein linienförmiges Graphobjekt nur eine Beschriftung besitzt, ist es von zusätzlicher Bedeutung, diese am Objekt zentriert zu positionieren.

Die finale Platzierung aller Beschriftungen wird mit einem Algorithmus des *Simulated Annealing* durchgeführt. Anfangs werden alle Labels zufällig in eine ihrer individuell möglichen Positionen gesetzt. Iterativ werden nun zufällige Labels in eine andere Position gesetzt. Die nach jedem Schritt durchgeführte Evaluation des Gesamtlayouts hat Einfluss auf die Wahrscheinlichkeit, mit der der Schritt rückgängig gemacht wird. Zusätzlichen Einfluss hat eine mit der Laufzeit des Algorithmus sinkende Temperatur, die den Zweck hat, Verschlechterungen im Layout gegen Ende der Iterationszahl weniger zu tolerieren.

Für die für *Simulated Annealing* benötigte Evaluation ist eine Anzahl von individuell gewichteten Metriken verantwortlich.

Überlappungsmetriken ermöglichen eine Differenzierung in der Schwere einer Überschneidung. Eine anfängliche Priorisierung zwischen den generierten möglichen Positionen begünstigt Platzierungen, die intuitiv besser platziert werden können. Jeder Buchstabe einer Beschriftung wird als Rechteck aufgelöst, was dazu führt, dass Überschneidungen präziser bestimmt werden können, als dies der Fall ist, wenn ein einziges umfassendes Rechteck für den ganzen Schriftzug eingesetzt wird.

3.2 Herangehensweisen

Bedingt durch die verschiedenen Problemstellungen sind im Einzelfall eine Anzahl von Herangehensweisen gegeneinander abzuwägen, um eine geeignete Platzierung von Labels zu erreichen. Es wird versucht, eine Klassifizierung durchzuführen, die bei der Wahl des Layouting-Algorithmus hilfreich ist.

Ein Layouting-Algorithmus gilt als *vollständig*, wenn dabei aus dem gesamten Lösungsraum hinsichtlich einer Metrik optimale Lösungen gefunden werden. Hierbei ist es möglich, einen von vornherein reduzierten Lösungsraum zu betrachten, in dem Optimalität leichter zu erfüllen ist. Im Gegenzug ist ein Algorithmus *heuristisch*, falls Teile des Lösungsraumes nicht betrachtet werden, obwohl ihre Optimalität vom Algorithmus nicht ausgeschlossen wurde.

Eine vollständige Suche ist nicht vollständigen dann vorzuziehen, wenn bei der gegebenen Problemstellung eine besondere Gefahr von lokalen Optima besteht. Ein solches lokales Optimum wird anhand von Abbildung 16 deutlich.

Restriktive Algorithmen schließen die möglichen Positionen von Labels auf eine endliche Menge ein. Aus einer globaleren Perspektive betrachtet handelt es sich bei allen restriktiven Herangehensweisen um Heuristiken, da möglicherweise optimale Lösungen von vorneherein ausgeschlossen werden. Um dies zu zeigen, genügt es, ein Beispiel zu finden, bei dem alle im eingeschränkten Lösungsraum

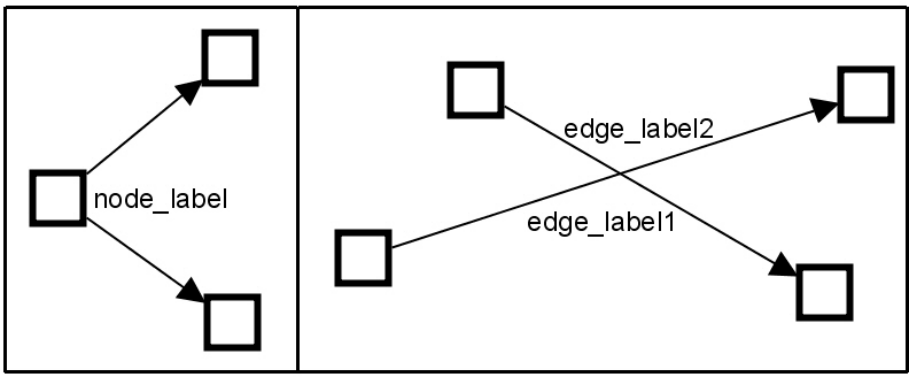

Abbildung 16: Lokale Optima bei der Platzierung von Beschriftungen

möglichen Labelpositionen inoptimal sind. Im Allgemeinen lassen sich entsprechende Situationen leicht konstruieren, wie Abbildung 17 zeigt.

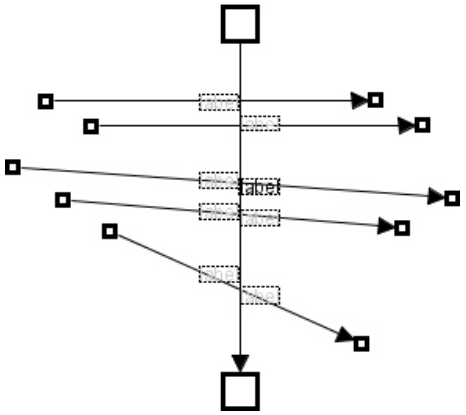

Abbildung 17: Inakzeptables Labeling wegen ungüstiger Wahl der Positionskandidaten

Als *iterative* Verfahren werden solche bezeichnet, die mehrmals auf den Graphen angewendet werden und das Ziel haben, das Layout bei jeder Anwendung zu verbessern. Meist handelt es sich dabei um Heuristiken, weil nicht ausgeschlossen werden kann, dass das Verfahren in einem schlechten lokalen Optimum konvergiert. Iterative Verfahren bieten den Vorteil, dass ihre Zwischenergebnisse bereits als Darstellungen visualisiert werden können. Daher ist es möglich, sie nach einer gegebenen Zeit mit Resultat abzubrechen.

3.2.1 Diskretes Labeling mit logischem Ausschluss

Iterative Verfahren neigen dazu, in lokalen Optima zu konvergieren. Um diesem Problem beizukommen, muss eine vollständige Suche im gegebenen Suchraum

durchgeführt werden. Meist ist dies nur möglich, wenn der Suchraum auf eine endliche Größe eingeschränkt wird.

Bei den im Folgenden vorgestellten Verfahren existiert eine feste Anzahl möglicher Positionen für jedes Label. Durch die Vorgehensweise ihrer Generierung wird bei jeder dieser Positionen garantiert, dass sie in optimaler Entfernung zum zu beschreibenden Graphelement liegt. Daher muss dem Kriterium der Identifizierbarkeit keine Rechnung getragen werden, was das Positionierungsverfahren vereinfacht.

Allgemeine Verfahrensweise beim diskreten Labeling Die Verfahren für diskretes Labeling lassen sich in 3 Stufen zerlegen:

- **Generierung von Positionskandidaten** In einem ersten Schritt werden eine endliche Anzahl möglicher Positionen für jedes Label generiert. Überlappungen zwischen Labels und den anderen Graphobjekten sind bereits bei der Generierung der Positionskandidaten bekannt, da beim diskreten Labeling die Position und Form von Knoten und Kanten nicht verändert wird.

- **Positionsbewertung** Anschließend erfolgt eine Bewertung aller Positionskandidaten nach verschiedenen Kriterien. Teilweise fließt eine solche Bewertung bereits bei der Generierung mit ein.

- **Festlegung der Labelpositionen** Schließlich werden die Positionen der Labels in geeigneter Reihenfolge festgelegt. Wie in Abschnitt 2.3.1 erwähnt, besteht die komplexe Fragestellung beim Labeling darin, bei einer Überlappung zweier Labels zu entscheiden, welches der beiden an eine andere Position zu verlegen ist. Hierbei gibt es verschiedene Herangehensweisen:

  - Eine Heuristik bestimmt die Reihenfolge. Ist die Position eines Labels einmal festgelegt, ist sie nicht mehr veränderbar. Die Heuristik wird darauf optimiert, die Platzierung so vorzunehmen, dass übrige Labels möglichst wenig in ihren Platzierungsmöglichkeiten eingeschränkt werden. Ein Beispiel für diesen Typ ist der Algorithmus *Finite Positions*, der in Kapitel 4 vorgestellt wird.
  - Wiederum bestimmt eine Heuristik die Reihenfolge der Platzierung, allerdings können schlechte Platzierungen im Einzelfall rückgängig gemacht werden. Dies geschieht zum Beispiel im *Simulated Annealing* beim Algorithmus von Edmondson, Christensen, Marks und Shieber [ECMS96].
  - Bei der vollständigen Tiefensuche werden nicht nur einzelne Platzierungen rückgängig gemacht, sondern alle Kombinationen von Platzierungen betrachtet. Mit Hilfe des *Prunings* werden einige irrelevante

Platzierungsoptionen von vorneherein ausgeschlossen, was die Anzahl der real zu überprüfender Kombinationen verringert, ohne, dass Optimalitätsgarantieen aufgegeben werden müssen [WW97].

Verfahren mit garantierter Optimalität Da ein solches Verfahren nur eine endliche Anzahl verschiedener Lösungen produzieren kann, können Optimalitätskriterien definiert werden. Eine optimale Lösung des im Abschnitt 2.3.1 vorgestellten *Point Feature Label Placement Problem* bespielsweise ist eine Platzierung, bei der die Anzahl der Überlappungen zweier Labels in der Menge aller möglichen Platzierungen minimal ist. Diese Eigenschaft lässt zu, dass für bestimmte Algorithmen eine minimale Qualität der von ihnen produzierten Lösungen bewiesen werden kann.

Wie in Abschnitt 2.3.1 erwähnt, ist es im Allgemeinen schwierig, zu entscheiden, welche Position verworfen werden soll, falls sich zwei Labelpositionen überlappen. Um die hohe Laufzeit zu vermeiden, schlagen Wagner und Wolff einen Algorithmus zur Lösung des *point feature label placement problem* vor, der als ersten Schritt die Anzahl der zu betrachtenden Quadrate pro Label auf maximal 2 reduziert [WW97]. Als zusätzliche Einschränkung sind alle Labels von quadratischer Form mit einheitlicher Kantenlänge $\sigma \in \mathbb{R}$. Das Ziel ist es, ein maximales σ_{opt} zu finden, für das eine überlappungsfreie Platzierung (siehe Abschnitt 2.2) möglich ist. Zwar ist so die Anzahl der Positionskandidaten pro Label endlich, jedoch gibt es die kontinuierliche Größe σ, die optimiert werden soll.

Durch die frei wählbare Kantenlänge wird der Suchraum erheblich vergrößert. Um trotzdem eine Suche in allen nötigen Längen zu ermöglichen, wird zu Beginn des Algorithmus aus den möglichen eine Anzahl wichtiger σ extrahiert. Wagner und Wolff zeigen, dass nur eine Anzahl linear zur Anzahl der Knoten betrachtet werden muss [WW97]. Diese Erkenntnis macht den Suchraum wieder endlich und damit vollständig durchsuchbar.

Die Suche nach σ_{opt} startet mit gleich großen Quadraten an jedem Knoten in allen vier Richtungen. Alle Quadrate werden uniform vergrößert. Um zwischen diesen Überlappungen aufzulösen, werden all jene, die einen Knoten des Graphen umfassen würden, falls sie doppelte Größe hätten, verworfen. So verbleiben an jedem Knoten nicht mehr als zwei Quadrate, die sich mit anderen überlappen [WW97].

Anschließend werden alle Konflikte boolesch codiert. Jedes Quadrat erhält hierzu eine boolesche Variable q_i, die anzeigt, ob das Quadrat als Position verwendet wird. Überlappen sich zwei Quadrate q_a und q_b, so ist es nicht möglich, beide zu verwenden. Dies wird boolesch ausgedrückt als $\overline{(q_a \wedge q_b)}$. Alle solchen Konflikte werden nun konjunktiv verknüpft. Die Erfüllbarkeit dieses Terms sagt aus, ob eine überlappungsfreie Lösung des Problems mit gegebenem σ existiert. Sie ist vom Typ 2-SAT, weil an jedem Punkt nur noch zwei Quadrate bestehen.

Wagner und Wolff zeigen, dass der Algorithmus ein $\sigma \geq \sigma\text{opt}/2$ findet [WW97].

Ein Problem ist allerdings, dass bei einer Überlappung beide Positionskandidaten verworfen werden, obwohl einer allein den gemeinsamen Platz überlappungsfrei belegen könnte.

Daher wird der Algorithmus dahingehend abgewandelt, dass die Quadrate erst dann eliminiert werden, sobald sich herausstellt, dass diese in keiner Lösung für irgendein σ gewählt werden können. Dabei kann es sein, dass nach der Eliminationsphase Punkte existieren, die noch 3 oder 4 Quadrate besitzen. In einem Zwischenschritt werden daher mit einer Heuristik die Anzahl der Positionskandidaten jedes Labels auf 2 reduziert.

Wagner und Wolff zeigen, dass der erweiterte Algorithmus mit einer Laufzeit von $O(n \cdot \log n)$, wobei n die Anzahl der Punkte ist, ein $\sigma \geq \sigma_{\text{opt}}/2$ findet [WW97].

3.2.2 Kontinuerliches Labeling

Ein Nachteil von diskreten Verfahren besteht darin, dass eventuell mit keiner der möglichen Labelpositionen ein ausreichend gutes Layout erreicht werden kann. In diesem Fall müssen Nachbearbeitungsschritte eingefügt werden, die, um wirkungsvoll zu sein, von einigen Vorteilen wie garantierte Optimalität oder Laufzeit Abstand nehmen müssen.

Im Gegensatz zu diskreten Verfahren sehen *kontinuierliche* für jedes Label einen ganzen Bereich an möglichen Positionen vor und decken daher einen größeren Suchraum ab.

Freie Platzierung Die wenigsten Einschränkungen an mögliche Labelpositionen schreiben *freie* Platzierungsverfahren vor. Prinzipiell ist hierbei jede Position ein mögliches Ergebnis der Platzierung. Dies stellt einen Vorteil in speziellen Situationen dar, da der Lösungsraum nicht eingeschränkt wird und daher keine möglicherweise optimalen Positionen von vorneherein ausgeschlossen werden. Allerdings vergrößert sich der Suchraum, so dass eine vollständige Suche nur durch Vereinfachungen durchgeführt werden kann.

Zusätzlich zur Überlappungsfreiheit muss ein Kriterium eingeführt werden und für jede Position evaluiert werden können, das Aufschluss darüber gibt, wie leicht ein Label dem Elterobjekt eindeutig zugeordnet werden kann.

Durch die Komplexität des Suchraums eignen sich vor allem iterative Heuristiken für dieses Platzierungsmodell.

Physikalisch orientierte Verfahren bilden ästhetische Platzierungskriterien durch Federspannungen ab. Anziehende Kräfte wirken bei eine entsprechende Nähe begünstigende Faktoren. Wirkt die Nähe zu einem gewissen Objekt einer Platzierung entgegen, so stellt sich dies als eine abstoßende Federkraft dar. Mit Hilfe einer physikalischen Simulation werden Positionen iterativ angepasst.

Platzierung mit einem Freiheitsgrad Eine freie Platzierung bedingt einen großen Suchraum für gute Labelpositionen. Die daraus resultierende Komplexität

des Layouting-Algorithmus macht Fehlplatzierungen wahrscheinlicher. Daher gibt es Bestrebungen, Verfahren einzusetzen, die die Patzierungsfreiheit beschneiden und dadurch weniger Freiheitsgrade besitzen.

Je mehr sich der Abstand eines Labels zu seinem Elterknoten vergrößert, desto unwahrscheinlicher ist es, dass das Kriterium der eindeutigen Zuordnung erfüllt ist. Daher bietet es sich an, den Suchraum zumindest so zu beschneiden, dass er keine Positionen enthält, an denen ein Label dem Elterobjekt nicht zugeordnet werden kann. Mögliche Positionen erstrecken sich also im Raum mehr oder weniger unmittelbar um das Elterobjekt.

Lohnend ist ist diese Sichtweise vor allem bei Kantenlabels, da eine Verschiebung der Beschriftung auf der Kantenlinie die visuelle Zuordnung nicht beeinträchtigt. Andererseits jedoch können Kantenkreuzungen, in deren Nähe wegen der Ambiguität der Zuordnung eher nicht platziert werden sollte, an beliebiger Stelle der Kante vorkommen. Abbildung 17 zeigt hierzu ein Beispiel. Daher ist es nötig, dass verfügbare freie Stellen der Kante auch für die Platzierung nutzbar sind.

Das von Kakoulis und Tollis vorgestellte Verfahren beschränkt die Platzierungsfreiheit eines Labels auf eine Dimension [KT01]. Mit Hilfe einer Kostenfunktion gehen Positionierungspräferenzen und Überlappungen in die Bewertung einer möglichen Position ein.

Verschiebemethoden Bei der *Verschiebemethode* kann durch ein einfacheres Modell der Layoutveränderungen auf eine iterative Simulation verzichtet werden. Ähnlich den physikalischen Herangehensweisen wird die Verbesserung eines vorhandenen Layouts angestrebt. Hierbei werden günstige Layoutveränderungen jedoch nicht iterativ, sondern in einem Schritt vollzogen. Allerdings ist die Methode, bedingt durch das einfache Modell von Verschiebungen, nicht darauf ausgelegt, als primärer Layoutalgorithmus eingesetzt zu werden.

In einem vorhergehenden Arbeitsschritt wird ein Layout ohne Rücksicht auf Überlappungen durchgeführt. Diese naive Positionierung hat den Zweck, eine möglichst gute Startsituation für eine Überlappungseliminierung per Verschiebemethode zur Verfügung zu stellen. Dabei sollten sowohl minimale Kantenlängen als auch eine ausreichende Verteilung des Graphen auf die Darstellungsfläche gewährleistet werden.

Dwyer, Marriott und Stuckey beschreiben einen Algorithmus zur Überlappungsauflösung durch Verschieben von rechteckigen Knoten mit einer worst-case Laufzeit von $O(n \cdot \log n)$ [DMS05]. Um das Modell zu vereinfachen, werden nur Überlappungen in einer Dimension aufgelöst. Daher wird der Algorithmus zwei Mal durchlaufen: Zuerst auf der x-Achse und danach auf der y-Achse. Dabei wird jeweils in einem ersten Schritt nach Bedingungen für Überlappungsfreiheit gesucht. Überlappende Knoten werden sodann unter Betrachtung dieser Bedingungen auseinandergeschoben.

Separations-Constraints Um Abhängigkeiten zwischen überlappenden Rechtecken herauszufinden, werden in einem ersten Schritt mit Hilfe eines line-sweep-Algorithmus sogenannte *Separations-Constraints* gesucht. Diese haben die Form $u + a \leq v$, wobei u und v Variablen und $a \geq 0$ eine Konstante sind.

Als Kandidaten, zwischen denen Constraints generiert werden können, fallen nur solche Rechtecke in Betracht, deren Projektionen auf die andere Achse überlappen. Wie Abbildung 18 zeigt, kann durch eine Verschiebung auf der Verschiebungsachse nur bei solchen Rechtecken eine neue Überlappung entstehen. Sie werden *Nachbarn* genannt.

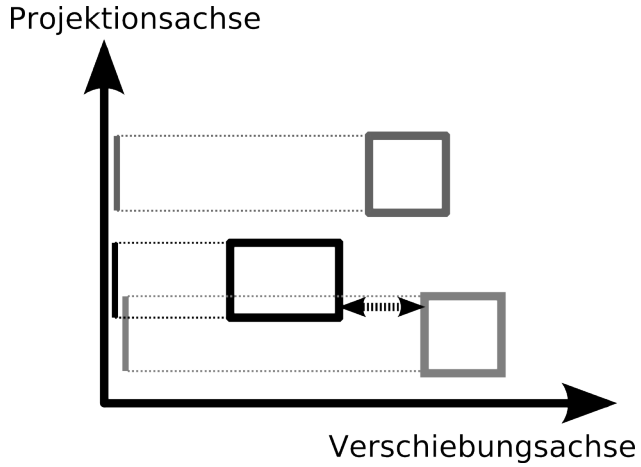

Abbildung 18: Separations-Constraints können nur zwischen zwei Rechtecken entstehen, die auf gleicher Höhe sind.

Eine Heuristik entscheidet darüber, zwischen welchen Nachbarn Constraints erzeugt werden. Die Anzahl der vom angegebenen Algorithmus generierten Constraints wird mit $O(k \cdot |V|)$ abgeschätzt [DMS05]. Die Annahme hierbei ist, dass ein Rechteck mit nicht mehr als k anderen Rechtecken überlappt.

Äuflösung von Separations-Constraints Durch Einsetzen der Constraints in einen *Constraint Graph* kann das Problem analytisch gelöst werden. Um eine Überlappung aufzulösen, werden die auseinandergeschobenen Rechtecke an ihrem gemeinsamen gewichteten Mittelpunkt positioniert.

Resultate Mittels empirischer Tests zeigen Dwyer, Marriott und Stuckey, dass der Algorithmus sowohl im Hinblick auf möglichst geringe Verschiebung, als auch hinsichtlich der Laufzeit gegenüber vergleichbaren Ansätzen überlegen ist [HIMF98, Lyo92]. Die Laufzeit des Algorithmus wird mit $O(n \log n)$ abgeschätzt, wobei angenommen wird, dass die Anzahl der Rechtecke, mit denen ein Rechteck überlappt, mit einer Konstante abgeschätzt werden kann [DMS05].

3.2.3 Fusion mit freien Verfahren

Falls ein Verfahren mit eingeschränktem Suchraum kein günstiges Layout finden kann, ist es möglich, das Ergebnis mit einem freien Verfahren zu verbessern. Hierbei wird für die Problembereiche des Graphen, für die keine akzeptable Platzierung gefunden werden konnte, lokal eine Neuanpassung durchgeführt.

3.2.4 Map Perimeter Labeling

Die bisher betrachteten Herangehensweisen nehmen an, dass die zu beschriftenden Graphobjekte auf einer zweidimensionalen Fläche verteilt sind.

Im Folgenden wird das von Iturriaga und Lubiw formulierte Verfahren für das *Elastic Rectangle Perimeter Labeling Problem* beschrieben [IL99]. Hierbei werden Beschriftungen von Punkten außerhalb des darstellbaren rechteckigen Bereichs an dessen Rand platziert (siehe *Rectangle Perimeter Labeling Problem* in Abschnitt 2.3.6). Zusätzlich kann, wie beim *Elastic Labeling Problem* aus Abschnitt 2.3.2, die Form einer Beschriftung durch Zeilenumbrüche variiert werden.

Im Algorithmus werden Spezialfälle des *One Corner Elastic Labeling Problem* behandelt, das im Allgemeinen NP-vollständig ist (siehe Abschnitt 2.3.2). Wie in Abschnitt 2.3.6 erwähnt, genügt es, das darstellbare Rechteck in eine polynomiell große Anzahl an Unterteilungen zu zerlegen, so dass in jeder dieser Unterteilungen zur Platzierung nur Labels von insgesamt zwei Seiten des Rechtecks berücksichtigt werden müssen. Nachfolgend werden die möglichen Unterteilungen aufgeführt.

Two-Axis Labeling Problem Die Aufpunkte p aller Labels befinden sich auf der positiven x-Achse oder der positiven y-Achse. Abbildung 19 zeigt eine gültige Lösung einer Instanz des *Two-Axis Labeling* Problems. Iturriaga und Lubiw zeigen, dass das Problem in einer Laufzeit von $O(m \cdot n)$ lösbar ist, wobei n und m die Anzahl der Rechtecke bezeichnet, deren Aufpunkt auf der x-Achse bzw. y-Achse liegt.

Two Parallel Lines Labeling Problem Die Aufpunkte aller Labels befinden sich auf zwei parallelen Linien. Die Richtung Q der Labels zeigt dabei nach innen. Abbildung 19 zeigt eine gültige Lösung einer Instanz des *Two Parallel Lines Labeling* Problems. Iturriaga und Lubiw geben einen Polynomialzeitalgorithmus zur Lösung an, der eine Laufzeit von $O(n^2)$ besitzt, wobei n die Anzahl der Labels ist.

Das darstellbare Rechteck wird nun in vier Instanzen des *Two-Axis Labeling* Problems zerlegt. Sogenannte *Jagged Lines* grenzen die Teilprobleme voneinander ab. Befindet sich zwischen den vier Teilproblemen in den Ecken ein *Korridor*, bildet dieser eine Instanz des *Two Parallel Lines Labeling* Problems. Abbildung 20 zeigt eine Instanz des Problems, bei der ein Korridor existiert.

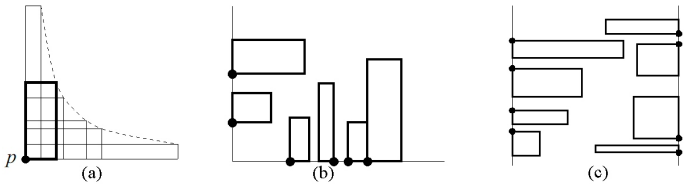

Abbildung 19: (a) Elastisches Label. (b) Two Axis Labeling Problem. (c) Two Parallel Lines Labeling Problem. (entnommen aus [IL99])

Abbildung 20: (a) Ein Korridor. (b) Dazugehörige Jagged Horizontal Lines (entnommen aus [IL99])

Insgesamt besitzt der Algorithmus von Iturriaga und Lubiw zur Lösung des *Elastic Rectangle Perimeter Labeling* Problems eine Laufzeit von $O(n^7)$, wobei die Autoren Verbesserungsmöglichkeiten angeben, um $O(n^4)$ zu erreichen.

4 Entwickelte Verfahren

Im Zuge dieser Arbeit wurden für den Graphen editor *Gravisto* [BBF+05] am Lehrstuhl Informatik mit Schwerpunkt Theoretische Informatik der Universtät Passau einige Verfahren entwickelt und implementiert. Im Folgenden werden deren Funktionsweise und theoretische Eigenschaften diskutiert.

4.1 Spring Embedder

Das existierende Plugin SPRINGEMBEDDERFR [Mat06] wird genutzt, um Knoten zu platzieren. Es handelt sich dabei um eine Implementierung des von Fruchterman und Reingold in [FR91] vorgestellten Verfahrens mit einigen Erweiterungen. Das Verfahren ist eine physikalisch orientierte, iterative Heuristik zur freien Platzierung von Graphknoten.

4.1.1 Das Verfahren nach Fruchterman und Reingold

Die Idee bei einer physikalisch orientierten Heuristik ist, sich das Streben physikalischer Prozesse zu Nutze zu machen, einen energieminimalen Zustand zu erreichen. Wie Eades in seinen Ausführungen [Ead84] beschreibt, kann dies erreicht werden, indem positive und negative Layoutkriterien als anziehende und abstoßende Kräfte in einem mechanischen System simuliert werden. Diese Kräfte müssen nicht physikalischen Ursprungs sein, sondern können nach Präferenz gewählt werden. Der Vorteil der Anwendung kinetischer Formeln besteht darin, dass der Gradient der wirkenden Kräfte -so ihre Wahl günstig ist- in Richtung eines energieminimalen Zustands zeigt.

Wie bei allen Heuristiken besteht freilich auch bei diesem Ansatz das Problem, das es sich bei dem Zustand in Kraftrichtung eventuell nur um ein lokales Optimum handelt.

Im Unterschied zu echten physikalischen Kräften werden aus Potenzialen direkt Geschwindigkeiten abgeleitet und nicht Beschleunigungen. Dies geschieht, um dynamische Äquilibrien -wie etwa Orbits- zu vermeiden.

Das Verfahren unterscheidet zwei Phasen, welche fließend ineinander übergehen [FR91]. In der einleitenden *quenching*-Phase wirken starke Kräfte auf die Knoten, da davon ausgegangen werden muss, dass diese lange Wege bis zum gewünschten entspannten Zustand zurücklegen müssen. Auch ist es so möglich, einzelne lokale Optima zu überspringen. Schließlich werden in der sogenannten *simmering*-Phase die Kräfte niedriger skaliert, um eine Feinjustierung vorzunehmen. Dies ist notwendig, da wegen Oszillationen mit starken Kräften eine gewisse Layout-Qualität nicht mehr verbessert werden kann.

4.1.2 Kräfte für Knotenplatzierung

In jeder Iteration des SPRING EMBEDDERS werden eine Anzahl Kräfte auf die Knoten angewendet. Ihre gewichtete Summe entscheidet über die Bewegung, die der jeweilige Knoten am Ende der Kraftberechnung durchführt.

Die einzelnen für die Knotenplatzierung verwendeten Kräfte haben nur teilweise Bedeutung für das Labeling. Sie werden der Vollständigkeit halber dennoch hier aufgeführt. Matzeder zählt folgende Kräfte auf, die im SPRING EMBEDDERS-Plugin für die Knotenpositionierung eingesetzt werden:

- **Abstoßende Kraft zwischen Knoten**: Damit die Knoten des Graphen auf der Zeichenfläche gleichmäßig verteilt werden, wird eine abstoßende Kraft zwischen allen Knoten eingeführt. Prinzipiell reicht es dabei aus, nur die nächsten räumlich umgebenden Knoten abzustoßen, um einen erwünschten Mindestabstand einzuhalten.

- **Anziehende Kraft zwischen benachbarten Knoten**: Um die durchschnittliche Kantenlänge im Graph zu verkleinern, werden mit einer Kante verbundene Knoten voneinander angezogen, bis sie zueinander den angestrebten Mindestabstand einhalten.

- **Abstoßende Kraft zwischen Knoten und Kanten**: Kanten, die nicht an einem Knoten befestigt sind, sollten sich nicht in dessen unmittelbarer Nähe befinden. Daher kann eine Kraft zugeschaltet werden, die Knoten und Kanten voneinander abstößt.

- **Gravitationskraft**: Durch die Entspannung von Federn neigt das Verfahren dazu, Knoten tendenziell von der Mitte der Darstellung abzustoßen. Damit die Darstellung eine gewisse Fläche nicht überschreitet, wird auf alle Knoten eine Kraft in Richtung ihres gemeinsamen Schwerpunktes ausgeübt.

- **Magnetische Kraft**: Falls eine bestimmte Richtung gewünscht wird, in die die Kanten tendenziell zeigen sollen, kann eine magnetische Kraft gewählt werden. Wird ein Knoten festgehalten, so entsteht eine baumähnliche Darstellung.

4.1.3 Erweiterungen zur Platzierung von Labels

Um das Verfahren zur Platzierung von Labels zu verwenden, wird die generelle Struktur beibehalten. Allerdings sind dazu einige Erweiterungen nötig, auf die im Weiteren eingegangen wird.

Algorithmus 1 zeigt den schematischen Ablauf. Die Struktur des Graphen wird um Labels erweitert. Damit diese im SPRING EMBEDDER verarbeitet werden können, werden sie als spezielle, rechteckförmige Knoten modelliert.

Im Prinzip lassen sich die Kräfte zwischen Graphknoten auf Labels übertragen. Ein Unterschied besteht jedoch darin, dass jedes Label genau einem Elterobjekt zugeordnet wird. Dies soll sich auch durch das Layout ausdrücken. Daher werden spezielle Kräfte eingeführt, die auf Labels wirken.

Algorithmus 1 : Der Algorithmus SPRING EMBEDDER nach Fruchterman und Reingold [FR91], angepasst an die Positionierung von Labels

Eingabe : $G := (V, E)$ (Graph mit Knoten und Kanten)
$\Gamma(G)$ (Darstellung des Graphen)
L (Labels)
$\Gamma(L)$ (Darstellung der Labels)
iterations (Anzahl der Durchläufe des Algorithmus)
F (Menge der ausgewählten Kräfte samt Parameter)

Ausgabe : $\Gamma(G)$ (Veränderte Darstellung der Labels)

```
1  Daten( temp ∈ ℝ⁺ (momentane Temperatur)
2  z_w ∀w ∈ L (kumulative Kraftvektoren der jeweiligen Label))
3  for i := 1 to iterations do
4      Temperatur anpassen:
5      temp := temperatur_einstellen(i);
6      Wirkende Kräfte rücksetzen:
7      forall w ∈ L do
8          z_w := (0, 0);
9      Berechnen aller Kräfte:
10     forall f ∈ F do
11         forall w ∈ L do
12             z_w+ = f.berechne_kraft(w, G, L, Γ(G), Γ(L));
13     Knoten verschieben:
14     forall w ∈ L do
15         Γ(w) := knoten_verschieben(Γ(w), z_w);
```

Abstoßende Kraft zwischen Labels Um Überlappungen von Labels zu vermeiden, wird eine abstoßende Kraft definiert, die analog zur abstoßenden Kraft zwischen Knoten im Plugin SPRINGEMBEDDERFR funktioniert [Mat06], jedoch nur zwischen Labels wirkt.

Abstoßende Kraft zwischen Labels und Knoten Diese Kraft ist wiederum analog zur abstoßenden Kraft zwischen Knoten im Plugin SPRINGEMBEDDERFR

[Mat06], wirkt jedoch nur zwischen Knoten und Labels. Sie wird an dieser Stelle separat aufgeführt, da sie individuell parametrisiert werden kann.

Abstoßende Kraft zwischen Labels und Kanten Diese Kraft ist ähnlich zur abstoßenden Kraft zwischen Knoten und Kanten im Plugin SPRINGEMBEDDERFR [Mat06]. Sie wirkt nur zwischen Labels und Kanten und hat eigene Parameter.

Da Überlappungsfreiheit von Labels und Kanten für ein gültiges Labeling notwendig ist, ist diese Kraft standardmäßig angeschaltet. Sie kann aber bei Bedarf abgeschaltet werden, falls beim jeweiligen Anwendungsszenario Kantenüberlappungen erlaubt sind.

Kraft zwischen Knotenlabel und Elterknoten Damit ein Label seinem Elterobjekt visuell zugeordnet werden kann, ist räumliche Nähe förderlich. Bei Knotenlabels wird sowohl eine anziehende als auch eine abstoßende Kraft zwischen Label und Elterknoten definiert (siehe Abbildung 21). Sie ist der Kraft zwischen benachbarten Knoten im existierenden Plugin [Mat06] ähnlich, hat jedoch eigene Parameter. Besonders die gewünschte Distanz zwischen Label und Elterknoten ist wesentlich kürzer als zwischen zwei benachbarten Graphknoten.

Die Richtung dieser Kraft ist parallel zur Linie durch die Mittelpunkte der beiden Objekte:

$$berechne\_kraft_{NLPN}(w) \parallel (\Gamma(w) - \Gamma(e(w)))$$

Die abstoßende Komponente der Kraft berechnet sich mit dem Skalierungsparameter P^{rep}_{NLPN} und der optimalen Distanz $OptDist_{NLPN}$ folgendermaßen:

$$berechne\_kraft^{\text{rep}}_{NLPN}(w) := \frac{P^{\text{rep}}_{NLPN} \cdot OptDist^3_{NLPN}}{d(w, e(w))}$$

Zur Berechnung der anziehenden Komponente werden der Skalierungsparameter P^{rep}_{NLPN} und die optimale Distanz $OptDist_{NLPN}$ benötigt:

$$berechne\_kraft^{\text{att}}_{NLPN}(w) := \frac{-P^{\text{rep}}_{NLPN} \cdot d(w, e(w))^3}{OptDist_{NLPN}}$$

Die Herleitung der Berechnungen findet sich in [Mat06].

Kräfte zwischen Kantenlabel und Elterkante Bezüglich des Elterobjekts wirken auf Kantenlabels zwei annähernd orthogonale Kräfte. Eine Kraft beeinflusst die Distanz zwischen Elterkante und Label, eine andere bewirkt eine Zentrierung der Beschriftung auf der Länge der Kante.

Um eine Überlappung zwischen Schrift und Kante zu vermeiden, wird eine der Kante orthogonale abstoßende Kraft definiert. Abbildung 22 zeigt ihre Funktionsweise.

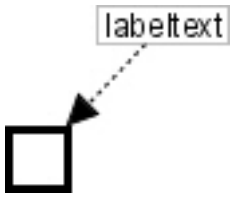

Abbildung 21: Kraft zwischen Knotenlabel und Elterknoten

Der Betrag wird nicht mit einer negativen Potenz der Distanz berechnet, sondern besitzt ein Maximum und hat eine endliche Reichweite. Damit soll es dem Label möglich sein, bei schlechter anfänglicher Platzierung die Kantenlinie zu überschreiten.

$$berechne\_kraft^{\text{rep}}_{ELPE}(w) \perp e(w)$$

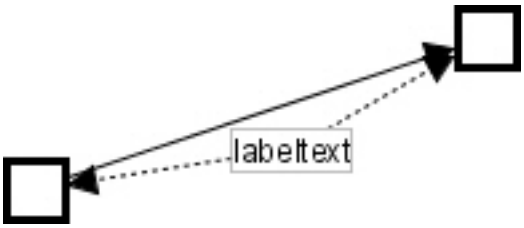

Abbildung 22: Eine abstoßende Kraft zwischen Kantenlabel und Elterkante vermeidet ihre Überlappung

Die vorliegende Implementierung modelliert je eine zusätzliche Kante zwischen den durch die Elterkante verbundenen Knoten und dem Label, wie es Abbildung 22 zeigt. Die entstehende Kraft zentriert einerseits das Label auf der Kante, andererseits wird verhindert, dass sich die Beschriftung von der Kante entfernt.

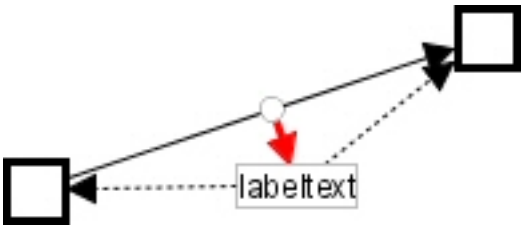

Abbildung 23: Zentrierende Kraft zwischen Kantenlabel und Elterkante

Die Berechnung erfolgt ähnlich zur oben erwähnten Kraft zwischen Knotenlabel und Elterknoten, allerdings wird das Label an zwei Knoten befestigt. Die

festgelegte optimale Länge jeder der beiden generierten Kanten ist kleiner oder gleich der halben Elterkantenlänge, damit immer eine Zugbelastung herrscht, die das Label in Richtung der Kante bewegt.

Mit dieser Methode geht einher, dass die Beträge der Kraft, mit der das Label die Mitte der Kante findet, und der Kraft, die den Text in die Nähe der Kante schiebt, nicht unabhängig voneinander parametrisiert werden können.

Kraft zwischen Knotenlabel und ausgehenden Kanten Vom Elterknoten ausgehende Kanten schränken die Platzierungsmöglichkeiten eines Knotenlabels ein. Ist der Winkel zwischen zwei im Einheitskreis benachbarten Kanten zu klein, ist es in ihrem Zwischenraum nicht möglich, die Beschriftung überlappungsfrei zu platzieren. Erfährt diese Situation keine besondere Berücksichtigung, so verbleibt das Label in einem ungünstigen lokalen Minimum zwischen beiden Kanten.

Um diese Situation zu vermeiden, wird eine zirkuläre Kraft eingeführt, die die reguläre Kantenabstoßung bei den ausgehenden Kanten des Elterknotens ersetzt. Wie Abbildung 24 zeigt, wirkt diese immer orthogonal zum Vektor Elterknoten-Label.

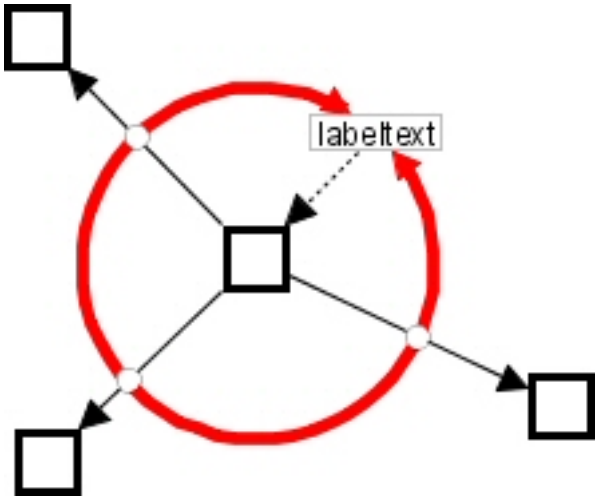

Abbildung 24: Kraft zwischen Knotenlabel und Elterknoten

$$berechne\_kraft_{NLEE}(w) \perp (w - e(w))$$

Zur Berechnung des Betrags werden vom Elterknoten aus die Winkel aller ausgehenden Kanten sowie der Winkel des Vektors Elterknoten-Label herangezogen. Einfluss haben nur jene beiden ausgehenden Kanten, die im bzw. gegen den Uhrzeigersinn dem Labelvektor am nächsten sind.

Überlappt der Text beide Kantenlinien, so ist eine Platzierung an dieser Stelle nicht geeignet. Der Betrag der Kraft wird in dem Fall so gewählt, dass die Position

des Labels innerhalb einer Iteration so weit in eine definierte Richtung versetzt wird, dass ihr Mittelpunkt eine der beiden Kanten überquert. Auf diese Weise werden ungünstige lokale Optima vermieden.

4.1.4 Laufzeitabschätzung

Die Laufzeitabschätzung des SPRING EMBEDDER Algorithmus zur Platzierung von Labels orientiert sich im Wesentlichen an der Abschätzung des SPRINGEMBEDDERFR-Plugins durch Marco Matzeder [Mat06].

Dies ist möglich, da Labels intern als zusätzliche Knoten behandelt werden. Für jedes Knotenlabel wird ein Knoten und eine Kante in den Graph eingeführt. Die Kante verbindet Elterknoten mit Knotenlabel. Bei Kantenlabels werden ein Knoten und zwei Kanten eingefügt, die das Label an die beiden durch die Elterkante verbundenen Knoten anknüpft.

Für die abstoßende Kraft zwischen Labels untereinander entsteht eine Komplexität von $\Theta(l^2)$, da jede Zweierkombination von Labels herangezogen werden muss. Für die Laufzeitabschätzung der abstoßenden Kraft zwischen Labels und Knoten wird analog $\Theta(l \cdot n)$ angesetzt. Die abstoßende Kraft zwischen Labels und Kanten produziert eine Laufzeit von $\Theta(l \cdot m)$, um alle Paare von Labels und Kanten zu durchlaufen. Da jedes Knotenlabel genau einen Elterknoten besitzt, ist die Laufzeit der Kraft zwischen Knotenlabel und Elterknoten mit $\Theta(l)$ zweitrangig. Ebensolches gilt für die Kräfte zwischen Kantenlabel und der jeweiligen Elterkante ($\Theta(l)$). Bei der Kraft zwischen Knotenlabel und der ausgehenden Kanten des Elterobjekts wird im Mittel eine Laufzeit von $\Theta(l \cdot \frac{m}{n})$ angesetzt. Diese Abschätzung gilt, wenn Knotenlabels im Graph auf die Knoten annähernd gleich verteilt sind.

Der Labeling-Algorithmus SPRING EMBEDDER besitzt also eine Laufzeitkomplexität von $\Theta(l \cdot (l + n + m))$.

Grid-Variante Reingold, Nievergelt und Deo schlagen in [RND77] eine raumaufteilende Datenstruktur vor, die auch im Algorithmus SPRING EMBEDDER Verwendung findet. Diese nutzt die Fähigkeit von Computern aus, die Modulo-Operation in konstanter Zeit durchzuführen.

Dabei werden mögliche Labelpositionen und Knoten anhand eines rechteckigen Gitters mit ebenmäßiger Zellenlänge und -höhe aufgeteilt. Die Anzahl der Kacheln wird so gewählt, dass in jeder Kachel eine konstante Anzahl an Objekten angenommen werden kann.

Für die Laufzeitanalyse wird davon ausgegangen, dass Labels und Knoten in der Darstellung annähernd gleichmäßig auf der quadratischen Zeichenfläche verteilt sind und dass die Anzahl der Überlappungen mit einer Konstante abgeschätzt werden kann.

Zum Aufbau der Datenstruktur wird eine Laufzeit von $\Theta(o)$ angesetzt, wobei o die Anzahl der Objekte in der Datenstruktur ist. Für jedes Objekt muss dazu

die konstante Menge an Kacheln ermittelt werden, in denen es sich befindet. Dies geschieht in konstanter Laufzeit.

Da sich in jeder Kachel mit einer Konstante abschätzbar viele Objekte befinden, entstehen zur Ermittlung aller Paare von Objekten, die eine durch eine Konstante angegebene maximale Entfernung nicht überschreiten, in der Theorie nur lineare Kosten.

Die Laufzeit für die Überlappungserkennung zwischen möglichen Positionen der Labels liegt somit in $\Theta(l)$. Für die Überlappungserkennung zwischen Positionskandidaten und Knoten gilt die selbe Abschätzung.

Bei der abstoßenden Kraft zwischen Labels und Kanten kommt erschwerend hinzu, dass sich Kanten über mehr als eine konstante Anzahl Kacheln erstrecken, sofern ihre verknüpften Knoten nicht in angrenzenden Kacheln liegen.

Falls die durchschnittliche Kantenlänge mit der Größe der Graphdarstellung abgeschätzt werden kann, befindet sich eine Kante in $O(n)$ Gridsektionen. Insgesamt existieren nach [FR91] n^2 Sektionen. Daher befinden sich in jeder Sektion $O(\frac{m \cdot n}{n^2}) = O(\frac{m}{n})$ Kanten. Die Laufzeit für die abstoßenden Kraft zwischen Labels und Kanten ist damit mit $O(\frac{l \cdot m}{n})$ abzuschätzen. Nimmt man zusätzlich an, dass die maximale Kantenlänge nicht von der Problemgröße abhängt, kann gefolgert werden, dass sich jede Kante nur in einer konstanten Anzahl von Kacheln des Grids befindet. Dies ist der Fall, wenn nur eine konstante Menge an Kanten Knoten miteinander verbindet, die nicht in benachbarten Kacheln sind. Die Laufzeit ist dann sogar in $O(l)$.

Die Gesamtlaufzeit des Algorithmus SPRING EMBEDDER unter Verwendung eines Grids als raumaufteilende Datenstruktur ist somit unter den gegebenen Bedingungen in $\Theta(l \cdot (\frac{m}{n}))$. Ob sich diese theoretischen Folgerungen in der Praxis umsetzen lassen, muss allerdings noch gezeigt werden (siehe hierzu Abschnitt 5.2).

4.1.5 Verwendungsmöglichkeiten

Wie bei den meisten Verfahren kann eine Platzierung von Labels bei festem Knotenlayout erfolgen. Es können Knoten- als auch Kantenlabels behandelt werden. Auch können sich beliebig viele Labels ein Elterobjekt teilen. Die Platzierung von Beschriftungen innerhalb von flächigen Elterobjekten wird in der aktuellen Implementierung nicht unterstützt.

Gemeinsame Platzierung Knoten und Labels Zusätzlich bietet der SPRING EMBEDDER als iteratives Verfahren die Möglichkeit, gleichzeitig Knoten und Labels zu platzieren. Hierbei wird das Graphlayout an etwaige Beschriftungen angepasst. Labels und Knoten müssen unterschieden werden, da ihr Layout unterschiedlich bewertet wird.

Fine-tuning Als *freies* Verfahren ist der SPRING EMBEDDER dazu geeignet, ein vorhandenes Layout zu verbessern. Vorteilig platzierte Labels und Knoten werden nur geringfügig verschoben, während Graphobjekte, für die durch das vorhergehende Verfahren keine günstige Position ermittelt werden konnte, ohne Einschränkung der Position neu platziert werden.

Bereiche des Graphen, für die *restriktive* Verfahren kein ausreichend gutes Layout erreichen können, können durch eine Vergrößerung des Lösungsraumes eventuell besser platziert werden, da zusätzliche Wahlmöglichkeiten entstehen.

So kann der SPRING EMBEDDER in der Lage sein, Nachteile eines anderen Platzierungsverfahrens auszugleichen.

4.1.6 Parameter des Algorithmus

Wesentlich bei *freien* Verfahren ist die Wahl der geeigneten Parameter. Zusammengefasst werden die Parameter des Algorithmus SPRING EMBEDDER aufgeführt:

Interaktiver Modus Falls der interaktive Modus angewählt ist, wird nach jedem Iterationsschritt eine Eingabe vom Benutzer erwartet. Außerdem werden die aktuellen Positionen der Knoten in den angezeigten Graph übernommen.

Iterationszahl Anzahl der Iterationen ergibt sich aus der Summe der Iterationen in der sogenannten *Quenching*-Phase und in der *Simmering*-Phase. Da in der *Quenching*-Phase Knoten- und Labelgrößen nicht beachtet werden, kann sie auch ausgeschaltet werden. Der Algorithmus beginnt dann mit der *Simmering*-Phase. Eine genaue Erklärung findet sich in [Mat06].

Temperatur Für jede der beiden Phasen kann eine Start- bzw. End-Temperatur festgelegt werden. Die Temperatur der einzelnen Iterationen wird durch lineare Interpolation bestimmt.

Lokale Temperatur Lokale Temperaturen werden verwendet, um Schwingungen einzelner Knoten zu dämpfen. Weitere Informationen sind in [Mat06] enthalten.

Kraft zwischen Elterknoten und Knotenlabel Dei Kraft zwischen Elterknoten und Knotenlabel wird durch mehrere Parameter bestimmt:

- **Optimale Distanz** Bei dieser Distanz ist die resultierende Kraft null.
- **Skalierung** Sowohl für die anziehende Komponente, als auch für die abstoßende Komponente der Kraft kann ein Skalierungsfaktor angegeben werden.

Abstoßende Kraft zwischen Labels Labels stoßen sich ähnlich wie Knoten voneinander ab. Der Skalierungsfaktor der Kraft kann eingestellt werden.

Abstoßende Kraft zwischen Elterkante und Kantenlabel Wie in Abschnitt 4.1.3 dargestellt, hat die abstoßende Kraft zwischen Elterkante und Kantenlabel nur begrenzte Reichweite und Stärke. Folgende Parameter können eingestellt werden:

- **Maximalkraft** Der maximale Betrag der Kraft.
- **Weicher Rand** Es kann ein weicher Rand innen und außen in Pixeln angegeben werden. Ein innerer Rand toleriert leichte Überlappungen zwischen Kante und Schrift eher, ein äußerer Rand bewirkt bereits eine kleine Kraft, wenn das Label in die Nähe der Kante kommt.

Abstoßende Kraft zwischen Kanten und Label Anders als die Kraft zwischen Elterkante und Kantenlabel wirkt diese Kraft zwischen allen Labels und Kanten. Ihre Reichweite ist unbegrenzt, allerdings fällt sie mit der Distanz stark ab.

Abstoßende Kraft zwischen Knotenlabels und ausgehenden Kanten Wie in Abschnitt 4.1.3 erwähnt, teilen sich Knotenlabels den Platz um den Elterknoten mit den von diesem ausgehenden Kanten.

- **Maximalkraft** Mit dem maximalen Betrag der Kraft wird der Bereich, in dem der Betrag des Kraftvektors liegen kann, skaliert.
- **Weicher Rand** Es kann ein innerer weicher Rand definiert werden, mit dem geringfügige Überlappungen mit einer Kante zu einer schwächeren Abstoßung führen.

Erweiterte Parameter betreffen die Anwendung, bei der Knoten und Labels gleichzeitig positioniert werden. Sie werden von Matzeder in [Mat06] beschrieben.

Eine geeignete Einstellung der Parameter ist von der jeweiligen Situation abhängig. Dabei skalieren gute Einstellungen unterschiedlich mit der Anzahl von Knoten, Kanten und Labels. Falls nur eine Platzierung von Labels durchgeführt wird, spielt das vorhandene Layout des Graphen für die Wahl der Parameter eine wichtige Rolle.

Dies kann als Nachteil des Algorithmus gewertet werden, da bereits Expertenwissen vorhanden sein muss, um gute Ergebnisse zu erzielen. Auch erschweren Wechselwirkungen der Parameter untereinander eine automatische Einstellung.

4.1.7 Mögliche Erweiterungen

Kräfte für Kantenlabels In Abschnitt 4.1.3 wurden die implementierten Kräfte, die zwischen Kantenlabel und Elterkante wirken, diskutiert und die fehlende

Orthogonalität der Parametrisierung hinsichtlich der beiden Dimensionen "Zentrierung auf der Kante" und "Abstand zur Kante" aufgezeigt.

Eine orthogonale Implementierung hätte den Vorteil, nicht zentral auf der Elterkante befindliche Labels weniger stark zu sanktionieren. Ein auf der Kante verschobenes Label gilt noch als gültige Platzierung, während ein von der Kante entfernt platziertes als nicht eindeutig zuzuordnendes gesehen werden kann.

Grid Um den Umfang der Arbeit nicht zu sehr zu erweitern, war es nicht möglich, die reale Performanz der vorhandenen raumaufteilenden Datenstruktur *Grid* unter verschiedenen Anwendungsszenarien zu analysieren. In Abschnitt 5.2 werden Tests an einem speziellen Graphen durchgeführt. Um genaue Aussagen machen zu können, wären weitere Tests an unterschiedlichen Graphen nötig.

4.2 Finite Positions Label Placement

Im Gegensatz zu kontinuierlichen bieten diskrete Verfahren den Vorteil, dass der Lösungsraum kleiner ist. Bei einer endlichen Anzahl von Positionskandidaten kann jeder einzeln betrachtet werden. Zum Auffinden der bestmöglichen Position innerhalb des Lösungsraumes ist daher einerseits weniger Laufzeit notwendig, andererseits sind entsprechende Algorithmen weniger kompliziert.

Im Zuge dieser Arbeit wurde Finite Positions Label Placement Algorithm als diskretes Platzierungsverfahren für unterschiedliche, rechteckige Labelgrößen entwickelt. Klassifiziert werden kann es als *greedy Heuristik*, das heißt, es bezieht nur lokale Information ein, um die Platzierung der einzelnen Labels vorzunehmen. Das Verfahren unterstützt Beschriftungen von geradlinigen Kanten und rechteckförmigen Knoten. Auch werden mehrere Label pro Graphobjekt unterstützt.

Der Algorithmus hält das in 3.2.1 vorgestellte Schema für diskrete Verfahren ein. Die einzelnen Schritte des Finite Positions Label Placement Algorithm lassen sich also wie folgt unterteilen:

- Bestimmung von Positionskandidaten
- Überlappungserkennung
- Schrittweise Festlegung von Labelpositionen

Als Eingabe wird zwar die Darstellung des Graphen verwendet, nicht aber die bisherige Position der Labels. Bei diskreten Verfahren sind im Allgemeinen die bisherigen Labelpositionen nicht im Lösungsraum enthalten, daher bildet die in Algorithmus 2 bezeichnete Bewertung der Labelposition ζ_p die tatsächliche Güte der Position eventuell nur unzureichend ab.

Die einzelnen Phasen des Algorithmus werden nun näher betrachtet.

Algorithmus 2 : Algorithmus FINITE POSITIONS

Eingabe : $G := (V, E)$ (Graph mit Knoten und Kanten)
$\Gamma(G)$ (Darstellung des Graphen)
L (Labels)
$e : L \rightarrow (V \cup E)$ (Zuordnung zu Elterobjekten)
k (Anzahl Positionskandidaten pro Label)
$\eta_p, \eta_n, \eta_e, \eta_k, \eta_l \in \mathbb{R}_0^+$ (Gewichtungen: Positionspräferenz, Knotenüberlappung, Kantenüberlappung, Kandidatenüberlappung, Labelüberlappung)

Ausgabe : $\Gamma(L)$ (Positionen der Labels)

1 K_w*: Menge der Positionskandidaten von* $w \in L$

2 *Generierung von Positionskandidaten:*
3 **forall** *(*$w \in L$*)* **do**
4 $K_w := \{\}$;
5 **for** $i := 1$ **to** k **do**
6 $K_w := K_w \cup generiere\_Positionskandidat(e(w), w, i)$;

7 *Bewertung von Positionskandidaten:*
8 $berechne\_Ueberlappungen(L)$;
9 **forall** *(*$w \in L$*)* **do**
10 **forall** *(*$p \in K_w$*)* **do**
$\zeta_p := \eta_p \cdot bewerte\_Positionspraeferenz(p)$
$-\eta_n \cdot Anzahl\_Knotenueberlappungen(p)$
$-\eta_e \cdot Anzahl\_Kantenueberlappungen(p)$
$-\eta_k \cdot Anzahl\_Kandidatenueberlappungen(p)$
11 $-\eta_l \cdot Anzahl\_Labelueberlappungen(p)$;
12 $\zeta_w := \max\{\zeta_p \mid p \in K_w\}$;

13 *Festlegung der Labelpositionen:*
14 **while** *(*$\exists w \in L : w$ *ist nicht markiert*$)$ **do**
$w = w \in L : w$ ist nicht markiert $\wedge$
15 $\zeta_w = \max\{\zeta_x \mid x \in L \wedge x \text{ ist nicht markiert}\}$;
16 $platziere(w, p \in K_w : \zeta_p = \max\{\zeta_z \mid z \in K_w\})$;
17 $markiere(w)$;

4.2.1 Bestimmung von Positionskandidaten

In einem ersten Schritt werden für jedes Label eine Anzahl $k \in \mathbb{N}$ *Positionskandidaten* bestimmt. Ihre Auswahl stellt sicher, dass sie den im Folgenden genannten Bedingungen genügen:

- **Nähe zum Elterobjekt:** Die Erfüllung der ersten Bedingung soll sicherstellen, dass das Label dem Elterobjekt zugeordnet werden kann. Hierbei kann der Text prinzipiell an beliebiger Position um das Elterobjekt herum angeordnet werden. Yoeli allerdings empfiehlt, manche als besonders günstig angesehene Lagen zu bevorteilen [Yoe72]. Daher wird mit jedem Positionskandidat ein Prioritätswert mitgeführt, der dazu dient, bei ansonsten gleicher Disposition bestimmte Platzierungen zu bevorzugen. In Algorithmus 2 wird dieser Prioritätswert mit $bewerte\_Positionspraeferenz(p)$ ausgedrückt, der zwischen 0 und 1 variieren kann.

  Abbildung 25 zeigt die verschiedenen Werte, die ζ_p für Knoten- und Kantenlabels in der vorliegenden Implementierung einnehmen kann. Die Bewertung von Knotenlabels hält sich im Wesentlichen an die von Yoeli empfohlene Priorisierung. Hierbei werden die Positionen an den Ecken den anderen vorgezogen. Kantenlabels werden entlang beiden Seiten der Kante generiert, wie es durch Kakoulis und Tollis vorgeschlagen wird [KT01]. Da Kantenlabels in Gravisto keine zusätzliche Semantik[1] enthalten, die sie mit einem der beiden mit der Elterkante verbundenen Knoten assoziiert, werden mittige Platzierungen relativ zur Kante besser bewertet.

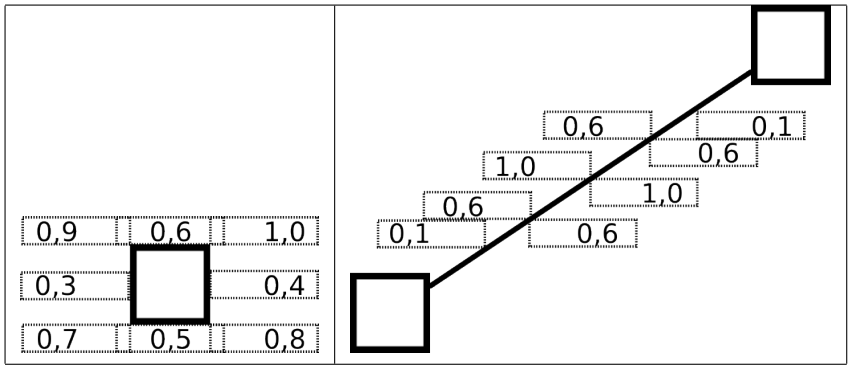

Abbildung 25: Positionierungspräferenzen des Algorithmus FINITE POSITIONS

- **Keine Überlappung mit dem Elterobjekt** Diese Bedingung kann bereits bei Erstellung der möglichen Platzierungen erfüllt werden, da nur je-

[1] Zusätzliche Semantik wird etwa bei UML-Diagrammen verwendet, um die Wertigkeit von Assoziationen in beide Richtungen darzustellen. Abbildung 9 gibt Anhaltspunkte, wie in entsprechenden Situationen die Priorisierung vorzunehmen ist

weils ein Objekt betrachtet werden muss. Eine Berücksichtigung der genauen Größe und Form des Elterobjekts ist hier besonders wichtig, um trotz Überlappungsvermeidung Nähe zu gewährleisten. In der vorliegenden Implementierung werden Knoten als Rechtecke und Kanten als Linien angenommen.

- **Knotenlabels: keine Überlappung mit ausgehenden Kanten des Elterknotens** Knotenlabels konkurrieren mit ausgehenden Kanten um den verfügbaren Platz um den Elterknoten. Daher besteht ein besonderes Interesse daran, dass Überlappungen nach Möglichkeit vermieden werden. Um dies zu erreichen, wird die Güte überlappender Positionskandidaten mit einem Strafterm versehen. Dieser schließt den Kandidaten zwar nicht von einer Wahl aus, benachteiligt ihn aber gegenüber anderen, nicht überlappenden Kandidaten.

- **Keine Überlappungen mit Knoten des Graphen** Positionskandidaten können sich mit anderen Graphknoten überlappen. Dies ist der Fall, wenn die Graphdarstellung zu wenig Raum für Labels vorsieht, kann aber auch bei einzelnen großen Labels oder bei lokal dichteren Bereichen vorkommen. Solche Positionen werden ebenfalls mit einem entsprechenden Strafterm versehen.

  Die Häufigkeit dieser Art von Überlappung wird deutlich erhöht, falls das vorhandene Layout des Graphen ungünstig ist. Ungünstig in diesem Zusammenhang sind relativ zur durchschnittlichen Labelgröße dicht platzierte Knoten.

- **Keine Überlappungen mit sonstigen Kanten des Graphen** Es gibt noch die Möglichkeit, dass ein Knotenlabel mit einer Kante überlappt, die nicht direkt mit dem Elterknoten verbunden ist. Auch besteht bei Kantenkreuzungen das Risiko, dass ein Kantenlabel in der Nähe des Kreuzungspunktes platziert wird. Entsprechende Positionskandidaten erhalten wiederum einen Strafterm auf ihren Qualitätswert. Bei der Auswertung wird nicht zwischen ausgehenden Kanten des Elterknotens und beliebigen anderen Kanten unterschieden.

Wie im Algorithmus 2 gezeigt wird, zählen i Überlappungen vom gleichen Typ i-fach. Eine Unterscheidung hinsichtlich der Schwere der Überlappung und dem damit verbundenen Verlust an Lesbarkeit wird in der aktuellen Implementierung nicht gemacht.

4.2.2 Überlappungserkennung

Nach Abschluss der ersten Phase stehen alle Positionskandidaten, nicht jedoch die Positionen der einzelnen Label fest. In der Phase der Überlappungserkennung

werden sich gegenseitig überlappende Positionskandidaten verschiedener Labels markiert und erhalten zunächst einen entsprechenden Strafterm bei der Berechnung ihrer Güte.

Aufgefundene Überlappungen werden bei beiden Kollisionspartnern registriert. Eine solche Überlappung bedeutet zwar noch keine Verdeckung zweier Labels, da ein Label im Regelfall mehrere Positionskandidaten zur Auswahl hat, sie ist jedoch ein Maß dafür, wie viele andere Labels in ihren Positionierungsmöglichkeiten eingeschränkt würden, wenn der Kandidat als Position des dazugehörigen Labels gewählt werden würde.

Beim Zählen der Überlappungen werden nur Positionskandidaten von unterschiedlichen Labels berücksichtigt. Die Überlappung mit zwei Positionskandidaten vom selben Label gilt also nur als eine Überlappung. Der Grund hierfür ist, dass jedes Label letztendlich nur an genau einer Position montiert wird und daher nicht mehrmals mit einer anderen Beschriftung überlappen kann.

Andernfalls würden zwei Knotenlabels am selben Elterobjekt wie im Beispiel von Abbildung 26 platziert werden. `label_no.1` wird dabei als erstes platziert. Da alle anderen Stellen mehrere Überlappungen mit Positionskandidaten von `label_no.2` anzeigen, bleibt lediglich die Platzierung auf einer der beiden Kanten übrig. Da im Beispiel die Zahl der Überlappungen mit Positionskandidaten ähnlich wie Kantenüberlappungen gewichtet werden, wird eine ungünstige Platzierung gewählt.

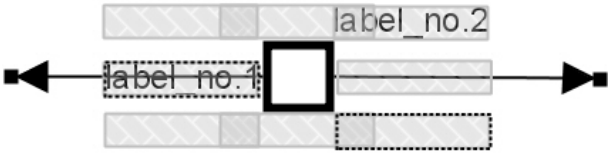

Abbildung 26: Probleme bei mehrfachen Überlappungen mit Positionskandidaten des selben Labels

Bei der Auswahl der Labelposition wird derjenige Positionskandidat gewählt, der bei der Bewertungsfunktion am besten abschneidet. Daher leitet sich die momentane *Platzierbarkeitgüte* ζ_w eines Labels w nach der Formel in Algorithmus 2 auf Zeile 12 ab.

Die Platzierbarkeitsgüte wird benötigt, um die Labels für die nächste Phase abfallend zu sortieren.

4.2.3 Schrittweise Festlegung von Labelpositionen

Die generelle Vorgehensweise des Algorithmus ist es, zuerst solche Labels zu platzieren, bei denen am wenigsten Überlappungskonflikte existieren. Eine solche unkritische Stelle wird nun für das Label festgelegt. Überzählige Positionskandidaten des Labels werden verworfen. Entsprechende Überlappungen entfallen. Auf diese

Weise verringert sich die Zahl der Kollisionen und die Platzierbarkeit weiterer Labels steigt an.

Den nach der Platzierbarkeitsgüte abfallend sortierten Labels wird das erste entnommen. Wie im Algorithmus 2 dargestellt sei dieses w genannt. w wird nun an die bestmögliche Stelle platziert.

Durch die Platzierung von w entfallen die Überlappungen der überzähligen, nicht verwendeten Positionskandidaten. Dagegen werden Überlappungen mit dem Positionskandidaten, an dem w platziert wurde, fortan als Label-Überlappung bewertet, was im Allgemeinen einen wesentlich größeren Malus für die Qualität der Position bedeutet. Die Befestigung eines Labels an einem Kandidaten mit Label-Überlappung bedeutet direkt, dass sich zwei Beschriftungen in der Darstellung überlappen werden.

Die durch die Platzierung von w entstandenen Bewertungsveränderungen einiger Positionskandidaten wirken sich unter Umständen auf die Platzierbarkeitsgüte ihrer Labels aus. Dadurch muss auch die Sortierung der Labels aktualisiert werden, damit bei der Platzierung des nächsten Labels wiederum dasjenige gewählt wird, das mit am wenigsten Konflikten montiert werden kann.

Die erwähnte *Markierung* in Algorithmus 2 hat den Zweck, dass jedes Label nur genau ein Mal platziert wird. Nach der Festlegung der Position wird w markiert und kann vom Algorithmus nicht mehr verschoben werden.

Sind alle Labels markiert, terminiert der Algorithmus und die Darstellung der Labels kann als Ergebnis zurückgegeben werden.

4.2.4 Beispiel

Die Funktionsweise des Algorithmus FINITE POSITIONS wird nun an einem Beispiel demonstriert.

Abbildung 27 zeigt die Situation vor Anwendung des Algorithmus. Es gilt, 2 Labels zu positionieren. Dabei handelt es sich um ein Knotenlabel und ein Kantenlabel.

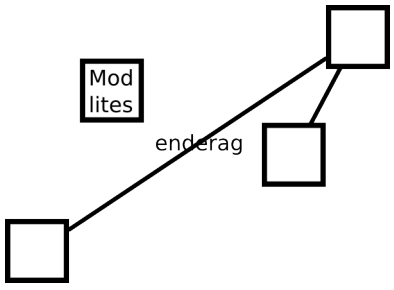

Abbildung 27: Startsituation des Beispiels

FINITE POSITIONS startet mit den Standard-Parametern. Es werden also 8 Positionskandidaten pro Label generiert. Ihre Position und initiale Bewertung

nach der ersten Phase ist durch Abbildung 28 gegeben. Alle Werte befinden sich zwischen 0 und 1.

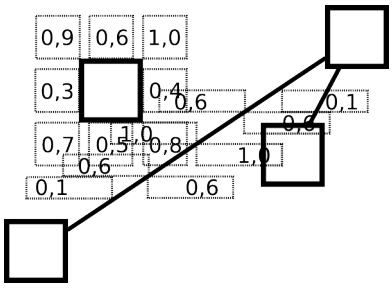

Abbildung 28: Generierte Positionskandidaten mit jeweiligen Positionspräferenzen

Nach der Phase der Überlappungserkennung werden die möglichen Positionen des Knotenlabels wie in Abbildung 29 bewertet. Links werden die Positionskandidaten des Knotenlabels, rechts die des Kantenlabels dargestellt. Überlappende Objekte sind der Übersichtlichkeit halber grau gehalten.

Eine Überlappung mit der Kante ergibt einen Malus von 4 Punkten. Überlappungen mit den Positionskandidaten des jeweils anderen Labels zählen 2 Maluspunkte. Obwohl einige Positionen mit mehr als einer Position des anderen Labels überlappen, wird der Malus nur ein Mal gewertet. Schneidet die Fläche eines Positionskandidaten einen Knoten, so werden in der Bewertung 10 Punkte abgezogen.

Die bestbewertete Position des Knotenlabels ist rechts oben mit einem Wert von $1,0$. Dies ist auch die momentane Platzierbarkeitsgüte für das Knotenlabel. Das Kantenlabel wird wegen der Position rechts unten mit $0,6$ bewertet.

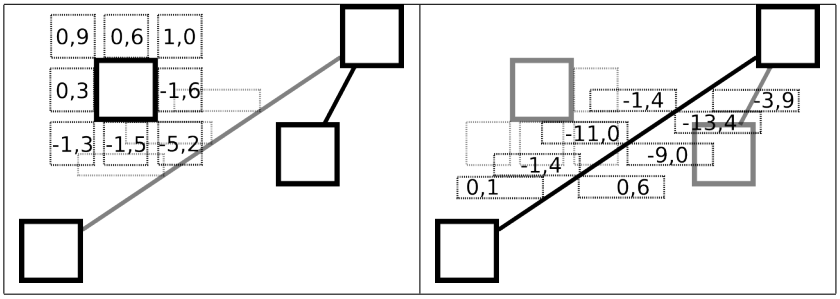

Abbildung 29: Bewertung der Positionskandidaten nach der Überlappungserkennung

In der Platzierungsphase kommt das Knotenlabel vor dem Kantenlabel, da es die höhere Platzierbarkeitsgüte besitzt.

Da es nun an die Position rechts oben montiert wird, entfallen alle weiteren Positionskandidaten des Knotenlabels. Wie Abbildung 30 zeigt, verändert dies die Bewertung einiger Kandidaten des Kantenlabels.

Die beiden in der Mitte der Kante generierten Positionen überlappen jeweils einen Knoten und sind daher nicht zur Platzierung geeignet. Allerdings gibt es nun bedingt durch den Wegfall von Kandidatenüberlappungen drei Positionskandidaten, deren Bewertung $0,6$ erreicht.

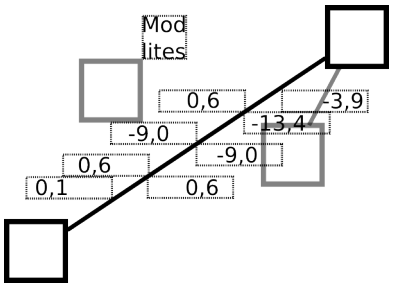

Abbildung 30: Positionsneubewertung nach Platzierung des Knotenlabels

Welche der drei gleich bewerteten möglichen Positionen bei der Platzierung verwendet werden, ist nicht durch den Algorithmus angegeben. Abbildung 31 stellt ein mögliches Ergebnis dar. Die Kriterien für ein gültiges Labeling werden von den anderen ebenfalls erfüllt.

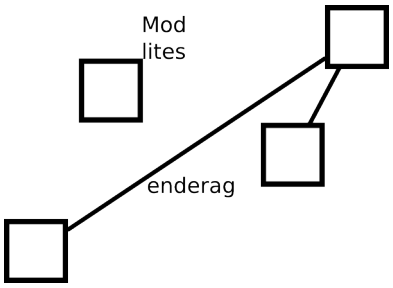

Abbildung 31: Situation nach Anwendung des Algorithmus FinitePositions

4.2.5 Laufzeitabschätzung

Eine größere Anzahl von Positionskandidaten verbessert tendenziell die Chance für eine überlappungsfreie Platzierung, allerdings ähneln sich die generierten Positionen immer mehr, da alle den im Abschnitt 4.2.1 angegebenen Bedingungen genügen müssen. Daher reichen in den meisten Fällen 8 Positionskandidaten für ein zufriedenstellendes Ergebnis aus. Aus diesem Grund kann die Anzahl der möglichen Positionen pro Label mit einer Konstante abgeschätzt werden.

Wenn eine geeignete raumaufteilende Datenstruktur für die Erkennung von Kollisionen verwendet wird, muss keine erschöpfende Suche für alle kombinatorisch möglichen Paarungen zweier Positionskandidaten unterschiedlicher Labels durchgeführt werden.

Es erfolgt nun eine Laufzeitabschätzung des Algorithmus FINITE POSITIONS anhand verschiedener Datenstrukturen. Da die Positionsgenerierung nur konstanten Aufwand pro Label verursacht, ist sie mit einer Laufzeit von $O(l)$ nicht für den Algorithmus ausschlaggebend.

- **naive/keine Strukturierung** Es werden alle potenziellen Kollisionspartner aufgezählt und getestet. Die Laufzeit ist $O(l \cdot (l + n + m))$, was aus Algorithmus 3 hervorgeht.

Algorithmus 3 : Algorithmus *berechne_Ueberlappungen*(p) in FINITE POSITIONS

Eingabe : $G := (V, E)$ (Graph mit Knoten und Kanten)
$\Gamma(G)$ (Darstellung des Graphen)
L (Labels)
$\bigcup_{w \in L} K_w$ (Menge aller Positionskandidaten)

Ausgabe : $\bigcup_{w \in L} K_w$ (Menge aller Positionskandidaten, angereichert mit Kollisionsinformation)

```
1  forall (w ∈ L) do
2      forall (k_w ∈ K_w) do
3          Überlappungen mit Knoten:
4          forall (v ∈ V) do
5              if (is_Ueberlappung(k_w, v)) then
6                  k_w.markiere_Knotenueberlappung(v);
7          Überlappungen mit Kanten:
8          forall (e ∈ E) do
9              if (is_Ueberlappung(k_w, e)) then
10                 k_w.markiere_Kantenueberlappung(e);
11         Überlappungen mit Positionskandidaten:
12         forall (u ∈ L) do
13             forall (k_u ∈ K_u) do
14                 if (is_Ueberlappung(k_w, k_u)) then
15                     k_w.markiere_Kandidatenueberlappung(k_u);
```

- **Quadtree** Finkel und Bentley zeigen, dass für den Aufbau dieser raumaufteilenden Datenstruktur eine Laufzeit von $O(n \cdot \log(n))$ zu veranschlagen

ist, wobei n die Anzahl der einzufügenden Objekte ist [FB74]. Um das nächste Objekt zu einem gegebenen Punkt zu suchen, ist logarithmische Zeit in Abhängigkeit der Größe der Datenstruktur notwendig.

Eine Abwandlung von Quadtrees, genannt *R-Tree*, ist für die räumliche Trennung achsenorientierter Rechtecke speziell zugeschnitten [Gut84]. Unter der Annahme, dass die maximale Labelgröße sowie die Gesamtzahl von Überlappungen mit jeweils einer Konstante abgeschätzt werden können, lassen sich die Laufzeiten von Quadtrees auf rechteckige Labels und Knoten übertragen [Gut84].

Die erwartete Laufzeit für eine Kollisionserkennung zwischen allen Labels unter Verwendung eines Quad- oder R-Trees ist $O(l \cdot \log(l))$. Diese zergliedert sich einmal in $O(l \cdot \log(l))$ für den Aufbau der Struktur und noch einmal $O(l \cdot \log(l))$ für die Erkennung von Überlappungen für jedes Label.

Überlappungen mit Knoten lassen sich auf ähnliche Weise berechnen, sofern Knoten ebenfalls als achsenorientierte Rechtecke dargestellt werden. Die Laufzeit hierfür wird mit $O(l \cdot \log(n))$ abgeschätzt.

Für Überlappungen mit Kanten fällt es schwerer, Laufzeitgarantien zu geben, da diese sich quer über den gesamten Graph erstrecken können, um gegenüberliegende Knoten der Darstellung zu verbinden. Dies ist in den meisten Anwendungen jedoch nicht der Fall. Daher wird davon ausgegangen, dass die maximale Länge aller Kanten mit einer Konstante abgeschätzt werden kann, während die Fläche der Darstellung des Graphen mit zunehmender Problemgröße wächst. Die Laufzeit für die Überlappungserkennung zwischen Labels und Kanten wird mit $O(l \cdot \log(m))$ abgeschätzt.

- **Grid** Da sich in jeder Kachel mit einer Konstante abschätzbar viele Objekte befinden, entstehen zur Ermittlung aller Kollisionspartner eines Positionskandidaten konstante Kosten. Die Laufzeit für die Überlappungserkennung zwischen möglichen Positionen der Labels liegt somit in $O(l)$. Für die Überlappungserkennung zwischen Positionskandidaten und Knoten gilt die selbe Abschätzung.

  Die Berechnung von Überlappungen mit Kanten des Graphen kann ähnlich erfolgen, jedoch ist es bei Kanten schwieriger, geeignete Raumaufteilungen zu finden. Falls die durchschnittliche Kantenlänge mit der Größe der Graphdarstellung abgeschätzt werden kann, befindet sich eine Kante in $O(n)$ Gridsektionen. Insgesamt existieren nach [FR91] n^2 Sektionen. Daher befinden sich in jeder Sektion $O(\frac{m \cdot n}{n^2}) = O(\frac{m}{n})$ Kanten. Die Laufzeit für die Überlappungserkennung zwischen Positionskandidaten und Kanten ist damit mit $O(\frac{l \cdot m}{n})$ abzuschätzen. Nimmt man zusätzlich an, dass die maximale Kantenlänge nicht von der Problemgröße abhängt, kann gefolgert werden, dass sich jede Kante nur in einer konstanten Anzahl von Kacheln des Grids

befindet. Die Laufzeit ist dann sogar in $O(l)$.

Für das Finden von Überlappungen unter Verwendung eines Grids als raumaufteilende Datenstruktur ergibt sich somit eine Laufzeit in $O(\frac{l \cdot m}{n})$, bei Annahme von kurzen Kanten in $O(l)$.

Bei der Festlegung der Positionskandidaten wird zuerst eine Sortierung der Labels hinsichtlich ihrer Platzierbarkeitsgüte vorgenommen, die für einen balancierten Baum eine Laufzeit von $O(l \cdot \log(l))$ beansprucht. Wie aus Algorithmus 2 hervorgeht, wird jedes Label bei der Positionierung genau ein Mal behandelt. Jeweils wird die Beschriftung an der individuell besten Stelle befestigt und alle überlappenden Positionskandidaten neu bewertet. Dies kann zu einer Umsortierung der noch nicht platzierten Labels führen, wozu für jedes kollidierende Label logarithmischer Aufwand veranschlagt werden kann, sofern als Datenstruktur ein balancierter Baum angenommen wird. Nach den bisherigen Annahmen existiert jedoch nur eine konstante Anzahl von Überlappungen, daher liegt die Laufzeit für die Festlegung der Positionskandidaten in $O(l)$.

Für den Algorithmus FINITE POSITIONS ergibt sich also unter Verwendung eines Quad- oder R-Trees [FB74, Gut84] im Mittel eine Laufzeitabschätzung von $O(l \cdot \log(l \cdot n \cdot m))$. Bei Verwendung eines Grids [RND77] liegt die asymptotische Laufzeit in $O(\max(\frac{l \cdot m}{n}, l \cdot \log(l)))$, wobei die Festlegung der Positionskandidaten die obere Schranke bestimmt. Ohne die Verwendung einer raumaufteilende Datenstruktur kann die asymptotische Laufzeit mit $O(l \cdot (l + n + m))$ abgeschätzt werden.

4.2.6 Parameter des Algorithmus

Durch die starke Einschränkung des Lösungsraums auf eine endliche Anzahl von möglichen Positionen ist die auch Anzahl der Parameter des Algorithmus begrenzt, was eine Bedienung vereinfacht.

Anzahl generierter Positionskandidaten Für jedes Label wird eine gegebene Anzahl von Positionskandidaten erstellt. Je mehr Positionierungsmöglichkeiten vorhanden sind, desto wahrscheinlicher kann eine überlappungsfreie Platzierung erfolgen. Allerdings erhöht sich auch die Laufzeit des Algorithmus.

Vermeidung der Überlappung mit Knoten Die Positionen und Größen von Graphobjekten stehen bereits zu Beginn des Algorithmus fest. Bereits bei der Generierung von Positionskandidaten können Überlappungen mit Knoten des Graphen getestet werden.

Vermeidung der Überlappung mit beliebigen Kanten Wie bei der Überlappungsvermeidung gegenüber Knoten sind für den Algorithmus die graphischen Parameter von Kanten ebenfalls unveränderlich.

Vermeidung der Überlappung mit ausgehenden Kanten Bei Knotenlabels ist eine Überlappung mit ausgehenden Kanten des Elterknotens häufig, da sich sowohl die Beschriftungen als auch die Kanten den Platz um einen Knoten herum teilen müssen. Da eine entsprechende Prüfung weniger rechenaufwändig ist als die Überlappungsvermeidung mit beliebigen Kanten, wird diese zusätzlich angeboten.

Freiraum zwischen Labels Yoeli führt freizuhaltende Bereiche um Labels ein, da eine Platzierung von Labels unmittelbar nebeneinander oder übereinander fälschlicherweise suggeriert, es handele sich um eine zusammengehörende Beschriftung [Yoe72].

Abbildung 32 zeigt ein Beispiel, bei dem die Beschriftungen zweier Knoten überlappungsfrei platziert wurden. Da beide Labels auf gleicher Höhe sind und der Abstand zwischen ihnen nur gering ist, entsteht der Eindruck, es handele sich um ein einziges Label.

Um dem entgegenzuwirken, werden die umgebenden Rechtecke von Labels größer behandelt. Damit werden zu nahe Platzierungen zusätzlich bestraft. Dies erhöht die Anzahl der Kandidaten- als auch Labelüberlappungen und kann bei mehreren Labels pro Knoten auch zu einer Verschlechterung des Layouts führen.

Gewichtung verschiedener Bewertungsmerkmale Die Bewertungsaspekte der Positionskandidaten können individuell gewichtet werden. Diese sind:

- Malus pro Labelüberlappung: Dieser Wert entscheidet darüber, wie stark Überlappungen zweier Labels bestraft werden. Ist der Wert hoch, werden eher Überlappungen mit anderen Graphobjekten in Kauf genommen oder Positionierungspräferenzen missachtet.
- Malus pro Knotenüberlappung: Überlappen Beschriftungen mit Knoten des Graphen, so kann hier durch Einstellen eines höheren Wertes Abhilfe geschafft werden.
- Malus pro Kantenüberlappung: Bei jeder Berührung einer Kante erhält der Positionskandidat einen entsprechenden Malus.
- Malus pro Kandidatenüberlappung: Überlappungen zweier Positionskandidaten bedeuten noch keine Verschlechterung des Labelings. Allerdings werden, falls solche Überlappungen nicht ausreichend beachtet werden, Beschriftungen in problematischen Bereichen des Graphen eventuell zuerst festgelegt.
  Es sollte darauf geachtet werden, dass der Wert nicht größer gewählt wird als der Malus für Labelüberlappungen, da anderenfalls die Überlappung mit bereits platzierten Labels als das "geringere Übel" bevorzugt als Position gewählt wird.

- Bereich für Bonus auf Positionspräferenz: Die Verwendung von bestimmten, in der Kartographie wegen ihrer Lage relativ zum Elterobjekt als besser lesbar angenommenen Positionen kann hier bevorzugt werden.

Abbildung 32: Notwendigkeit von freien Bereichen um Labels

4.2.7 Mögliche Erweiterungen

Freie Generierung von Positionskandidaten In der aktuellen Implementierung des FINITE POSITIONS LABEL PLACEMENT ALGORITHM werden maximal 8 Positionskandidaten an festen relativen Positionen um das Graphobjekt generiert. Wie Abbildung 17 zeigt, kann es vorkommen, dass eine Quantelung des Bereichs um das Graphobjekt ein erfolgreiches Labeling verhindert, obwohl akzeptable freie Positionen existieren. Besonders bei langen Kanten und bei großen Knoten ist dies der Fall. Diesem Problem kann entweder mit einer erhöhten Anzahl Positionskandidaten oder mit einer intelligenteren Generierungsstrategie begegnet werden. Beispielsweise ist es möglich, eine zur Größe des Graphobjekts proportionale Anzahl von möglichen Positionen bereitzustellen.

Da sowohl Knoten als auch Kanten während der Laufzeit des Algorithmus feststehen, ist es möglich, bereits bei der Generierung der Positionskandidaten auf Überlappungsfreiheit zu achten. Hierzu muss ein Modell des Elterobjekts mitsamt aller sich in der Nähe befindenden Graphobjekte algorithmisch erfasst werden. Günstige Positionen werden anhand von Freiräumen ermittelt. Hierbei ist der Implementierungsaufwand gegenüber der zusätzlichen Laufzeit bei einer erhöhten Zahl von Positionskandidaten abzuwägen.

Löschen von Positionskandidaten bei Überlappung mit Graphknoten Die Überlappung eines Labels mit einem Knoten des Graphen kann als eine grobe Verletzung der Ästhetik bei der Platzierung von Beschriftungen angesehen werden. Diese Verletzung mag sogar als schwerwiegender als eine Überlappung zweier Labels untereinander gewertet werden. Um dieses Prinzip im Algorithmus umzusetzen, ist es möglich, solche Positionskandidaten bereits in der Generierungsphase wieder zu verwerfen. Allerdings wird dadurch die Wahl auf die verbleibenden Kandidaten eingeschränkt. Es muss auf jeden Fall mindestens ein gültiger Positionskandidat verbleiben, da sonst keine Platzierung des Labels möglich ist.

Falls Positionskandidaten also gelöscht werden können, muss eine geeignete Lösung für den Fall gefunden werden, dass keine Positionsierungsmöglichkeiten mehr vorhanden sind.

Unterdrückung von schlecht platzierbaren Labels Falls die Qualität aller möglichen Positionen eines Labels durch Überlappungen mit Knoten, Kanten oder anderen Labels unzureichend ist, ist es eventuell sinnvoll, das Label nicht darzustellen. Wie in Abbildung 6 gezeigt, kann immerhin so die Lesbarkeit anderer Labels erhalten werden, da bei einer starken Überlappung keiner der beteiligten Schriftzüge lesbar ist.

Außerdem stellt dieser Ansatz eine Lösung dar, wenn wie bei der oben genannten Erweiterung alle Platzierungsmöglichkeiten wegen geringer Qualität ausgeschlossen wurden.

Unterscheidung von teilweisen Überlappungen Edmondson, Christensen, Marks und Shieber unterscheiden in ihrem Verfahren die Schwere von Überlappungen [ECMS96]. Falls die Wahl besteht, werden bei ansonsten gleichen Voraussetzungen Positionen bevorzugt, an denen sich ein Label weniger stark mit anderen Graphobjekten überlagert als bei anderen. Beispielsweise wird in Abbildung 33 die mit (a) markierte Position der mit (b) markierten vorgezogen.

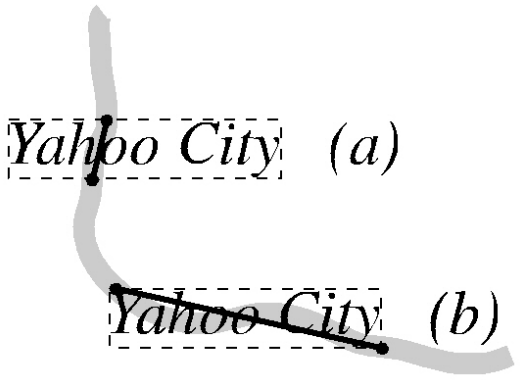

Abbildung 33: Teilweise Überlappungen zwischen Label und Linienzug in [ECMS96]

Definition. Seien p_1 und p_2 die Schnittpunkte des das Label umgebenden Rechtecks mit dem überlappten linienförmigen Graphobjekts und b der normalisierte Richtungsvektor entlang der Schriftrichtung des Labels, so gilt für die *Schwere der Überlappung* w_{linie} [ECMS96]:

$$w_{\text{linie}} := |((p_2 - p_1)/|p_2 - p_1|) \circ b|$$

Die Schwere der Überlappung variiert in der hier dargestellten Form zwischen 0 (Graphobjekt trennt die Schrift senkrecht) und 1 (Graphobjekt durchläuft die Schrift auf voller Länge). Selbstverständlich ist eine senkrechte Überlappung immer noch nicht optimal, daher sollte in der Definition von w_{linie} ein konstanter Term hinzuaddiert werden.

Ähnliche Metriken lassen sich auch für andere Überlappungstypen finden. Der Vorteil ist, dass leichte Überlappungen, die die Lesbarkeit nicht wesentlich beeinträchtigen, die entsprechenden Positionskandidaten nicht völlig unbrauchbar werden lassen. Gerade bei dichten Graphen und vielen Labels kann eine solche Unterscheidung förderlich sein.

Raumaufteilende Datenstruktur Für eine interaktivere Verwendung des Algorithmus kann es lohnend sein, die Laufzeit durch eine optimierende raumaufteilende Datenstruktur zu beschleunigen.

In der Benutzerschnittstelle des *Graph Visualisation Toolkit* könnte eine interaktive Verschiebung von Labels durch eine automatische, dynamische Justierung von Labels während der Verschiebung eines Knotens ersetzt werden. Dafür ist es notwendig, die Labels zumindest von Teilen des Graphen in Echtzeit positionieren zu können.

Die Möglichkeit einer solchen Erweiterung wurde bereits bei der Implementierung berücksichtigt und ist an entsprechender Stelle im Quelltext als Dokumentation nachzulesen. An die Schnittstelle der vorhandenen naiven Überlappungsfindung lässt sich mit wenig Aufwand eine optimierende raumaufteilende Datenstruktur anschließen.

4.3 Fusion von Finite Positions und Spring Embedder

Wie in Abbildung 17 gezeigt, bietet Finite Positions nur eine begrenzte Anzahl von Labelpositionen an und ist somit bei manchen Spezialfällen nicht in der Lage, ein günstiges Labeling durchzuführen. Dies erfordert eine Nachbearbeitung einzelner Labels.

Hierzu eignet sich der Spring Embedder aus Abschnitt 4.1, da er als iteratives und freies Verfahren beliebige Ausgangsplatzierungen übernehmen kann. Außerdem stehen bei einem freien Verfahren mehr Positionierungsmöglichkeiten zur Verfügung, so dass Nachteile wie etwa durch die in Abschnitt 4.2.7 erwähnte Quantelung des Bereichs um Graphobjekte aufgewogen werden können.

4.3.1 Vorgehensweise

Algorithmus 4 stellt Aufrufreihenfolge und Parameter dar. Zuerst wird der Algorithmus Finite Positions mit entsprechenden Parametern aufgerufen. Schließlich wird der Spring Embedder angewendet, wobei ausgenutzt wird, dass dieser die Positionierung der Labels $\Gamma(L)$ übernehmen kann. Damit von der Qualität

des vorangegangenen Verfahrens profitiert werden kann, werden nur wenige Iterationen und vergleichweise schwache Kräfte verwendet.

Algorithmus 4 : Kombination von FINITE POSITIONS mit SPRING EMBEDDER

Eingabe : $G := (V, E)$ (Graph mit Knoten und Kanten)
$\Gamma(G)$ (Darstellung des Graphen)
L (Labels)
P_f (Parameters des Algorithmus FINITE POSITIONS)
P_s (Parameters des Algorithmus SPRING EMBEDDER)

Ausgabe : $\Gamma(L)$ (Positionen der Labels)

1 *Ausführung von* FINITE POSITIONS*:*
2 $\Gamma(L) := finitePositions(G, \Gamma(G), L, P_f)$;
3 *Ausführung von* SPRING EMBEDDER*:*
4 $\Gamma(L) := springEmbedderLabeling(G, \Gamma(G), L, \Gamma(L), P_s)$;

4.3.2 Mögliche Erweiterungen

Falls der SPRING EMBEDDER nur dazu verwendet wird, die Positionen einzelner nicht zufriedenstellend platzierter Labels zu verbessern, ist eventuell nur eine Verschiebung weniger Knoten notwendig. Kräfte für die übrigen, günstig platzierten Labels zu berechnen kostet daher unnötige Laufzeit. Auch kann es notwendig sein, starke Kräfte wirken zu lassen, damit lokale Minima übersprungen werden können. Da die Kraftstärke in der vorliegenden Implementierung global gesetzt wird, kann sich dies nachteilig auf günstig platzierte Labels auswirken, die dann ebenfalls verschoben werden oder stark oszillieren können.

Die Implementierung des SPRING EMBEDDERS bietet die Möglichkeit, die Positionierung mit sogenannten *lokalen Temperaturen* durchzuführen [Mat06]. Im Gegensatz zu einer globalen Temperatur wird dabei die maximale Distanz, um die ein Label in einem Iterationsschritt bewegt werden kann, für jedes Label individuell bestimmt. Dies hat den Sinn, dass oszillierende Knoten früh gedämpft werden.

Da die lokale Temperatur jedes Labels in ihrem bisherigen Verwendungszweck monoton fällt, wäre es möglich, sie für günstig platzierte Labels von vorneherein niedriger anzusetzen, damit gute Positionen auch bei starken Kräften nicht verfälscht werden.

Die Ausgangstemperatur jedes Labels ist an die Positionsbewertung von FINITE POSITIONS zu koppeln. Labels mit Überlappungen erhalten dabei eine höhere Temperatur und werden daher durch den SPRING EMBEDDER stärker verschoben. Dies führt auch dazu, dass eventuelle Vorteile, die FINITE POSITIONS gegenüber

dem SPRING EMBEDDER bei der Platzierung besitzt, nicht verlorengehen, sofern die Bewertungsfunktion von FINITE POSITIONS die Vorteile als solche erkennt.

5 Experimentelle Untersuchung

Von den im Kapitel 4 entwickelten Verfahren stehen Implementierungen zur Verfügung, die im Zuge der Arbeit angefertigt wurden. Sie bieten Grundlage und Beispielmaterial für Vergleiche.

Für Laufzeittests wurde ein Pentium IV (2,53 GHz) mit 1536 Mibit RAM unter Windows 2000 verwendet.

5.1 Vergleich anhand unterschiedlicher Graphen

Die 3 vorgestellten Verfahren werden auf verschiedene Graphen angewendet und miteinander verglichen.

5.1.1 Graph $13 \times 18 \times 13$

Als allgemeines Verfahren zur Positionierung von Objekten ist der SPRING EMBEDDER in der Lage, sowohl Knoten als auch Labels innerhalb des gleichen Programmlaufs zu positionieren.

Im nachfolgenden Beispiel wird ein Graph verwendet, dessen Darstellung nicht mit einem Verfahren optimiert wurde und so aus einem Datensatz importiert worden sein könnte. Abbildung 34 (a) zeigt den nicht gelayouteten Graphen. Er besitzt 13 Knoten, 18 Kanten und 13 Knotenlabels, die überlappungsfrei auf wenig Fläche dargestellt werden sollen.

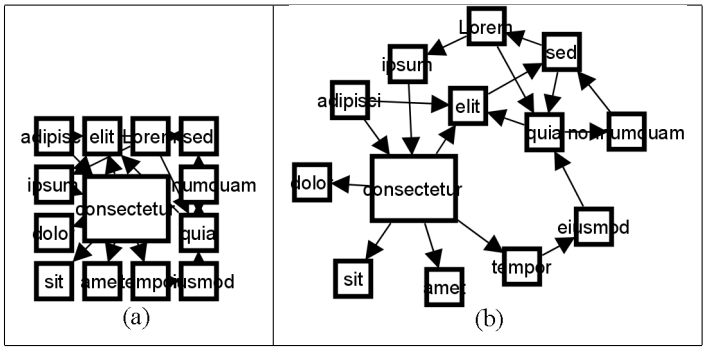

Abbildung 34: Beispielgraph *Lorem ipsum*. (a) nicht gelabelt mit naivem Layout. (b) nicht gelabelt, Layout mit Plugin SPRINGEMBEDDERFR.

Damit ein Vergleich mit FINITE POSITIONS stattfinden kann, wird eine Knotenplatzierung mit dem Plugin SPRINGEMBEDDERFR von Marco Matzeder [Mat06] durchgeführt. Um den Platzverbrauch zu beschränken, wird der Parameter *ideal node distance* auf den Wert 3 gesetzt. Das Resultat der Knotenplatzierung zeigt Abbildung 34 (b).

Der Algorithmus SPRING EMBEDDER wird einmal auf den nicht gelayouteten Graphen angewendet. Um Labels und Knoten gleichzeitig zu platzieren, wird der gesamte Graph vor der Ausführung markiert. Da sich hier Knoten auch von den Labels abstoßen, wird für den Parameter *ideal node distance* ein Wert von $1,0$ festgelegt. Die Anzahl der Iterationen der *Simmering*-Phase wird auf von 40 auf 80 gesetzt, um der erwarteten langsameren Konvergenz entgegenzuwirken.

Zum Vergleich wird SPRING EMBEDDER auch zum ausschließlichen Platzieren von Labels beim gelayouteten Graph genutzt. Hierzu werden die voreingestellten Parameter verwendet. Ebenso wird der Algorithmus FINITE POSITIONS mit Standardparametern auf den gelayouteten Graph angewendet.

Der kombinierte Algorithmus aus FINITE POSITIONS und SPRING EMBEDDER benötigt zum Lauf ebenfalls einen gelayouteten Graphen. Der Parameter *adjacent edges -> node label repulsion: max force* wird auf den Wert $4,0$ gesetzt, um die Wirkung des SPRING EMBEDDERS an Knotenlabels deutlicher zu machen.

Die resultierenden Graphen werden in Abbildung 35 zusammengefasst.

Ein gültiges Labeling wird durch die Algorithmen FINITE POSITIONS, dem Kombinationsalgorithmus von FINITE POSITIONS und SPRING EMBEDDER, sowie in manchen Läufen auch SPRING EMBEDDER erreicht.

FINITE POSITIONS liefert die über mehrere Läufe hinweg konstantesten Ergebnisse. Wird jedes Mal der Graph aus Abbildung 34 (b) verwendet, sind die Resultate sogar identisch. Am wenigsten konstant sind die Ergebnisse, wenn der SPRING EMBEDDER zur gleichzeitigen Platzierung von Knoten und Labels verwendet wird. Dies lässt sich zum Teil dadurch erklären, dass sich die Labels zu Beginn an der Position ihrer Elterobjekte befinden und daher zufällig verschoben werden. Aber auch bei gleichen Ausgangssituationen ist dies der Fall, da die Initialisierung des Zufallszahlengenerators der Implementierung unter anderem von der Systemzeit abhängt.

Bei der gemeinsamen Positionierung von Knoten und Kanten durch den SPRING EMBEDDER lief die Platzierung des *ipsum*-Elterknotens in der Mehrzahl der durchgeführten Läufe ähnlich ungünstig wie in Abbildung 35 (a). Hauptsächlich liegt dies daran, dass in der vorliegenden Implementierung bei einer Verschiebung des Elterobjekts ein Label an seiner Stelle verbleibt, was dessen Position im Allgemeinen eher verschlechtert. Diese Abhängigkeit führt zu einer vermehrten Konvergenz in schlechten lokalen Äquilibrien.

Ein Vorteil der gemeinsamen Platzierung ist jedoch, dass bei Vorhandensein von Labels von stark unterschiedlicher Größe das Knotenlayout entsprechend Platz für ein günstiges Labeling schaffen kann.

Die Position des Labels *consectetur* in Abbildung 35 (a) und (b) zeigt, dass es dem SPRING EMBEDDER als iterativem Algorithmus schwer fällt, inital ungünstig gewählte Positionen zu verbessern. Mit zusätzlichen Iterationen lässt sich die Kantenüberlappung im Graph (b) zwar verbessern, allerdings ist die Konvergenz langsam. Um diese zusätzlichen Iterationen zu vermeiden, stellt die Kombination mit FINITE POSITIONS zum Graph in Abbildung 35 (d) eine vorteilhafte

Initialisierung dar.

Die Positionierung des Labels *Lorem* im Graph 35 (c) im Vergleich mit dem Graph (d) zeigt, dass die Verwendung des SPRING EMBEDDERS als zweiten Schritt ein durch FINITE POSITIONS durchgeführtes Labeling verbessert. Deutlich erkennbar ist die Tendenz der Labels am Rand des Graphen, sich nach außen zu orientieren. Wegen der begrenzten Anzahl von möglichen Positionen ist dies mit FINITE POSITIONS nur eingeschränkt möglich.

Die Laufzeiten der Algorithmen variieren wesentlich zwischen den einzelnen Programmläufen. In Tabelle 1 handelt es sich um Mittelwerte über jeweils 4 Läufe. Abschnitt 5.2 befasst sich mit einer ausführlichen Laufzeituntersuchung, daher sind entsprechende Informationen an dieser Stelle knapp gehalten.

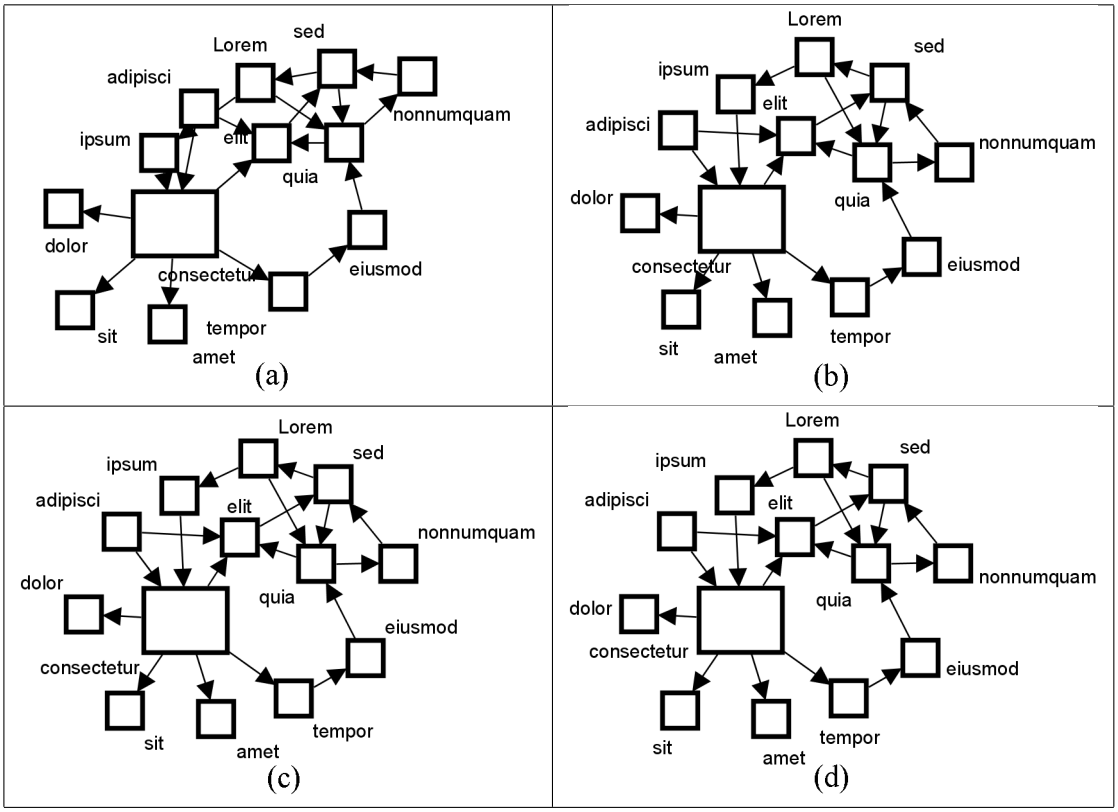

Abbildung 35: (a) SPRING EMBEDDER aus naivem Graphlayout. (b) SPRING EMBEDDER aus gelayoutetem Graph. (c) FINITE POSITIONS aus gelayoutetem Graph. (d) FINITE POSITIONS und SPRING EMBEDDER aus gelayoutetem Graph.

5.1.2 Graph $131 \times 154 \times 211$

An einem größeren Graphen werden die Algorithmen FINITE POSITIONS, SPRING EMBEDDER und der Kombinationsalgorithmus aus beiden getestet.

Spring Embedder mit Knotenlayout	0,25s
Spring Embedder	0,092s
Finite Positions	0,016s
Kombination FP&SE	0,088s

Tabelle 1: Laufzeiten der Algorithmen am Beispielgraph *Lorem ipsum* (je über 4 Läufe gerundet)

Es handelt sich dabei um einen zufällig generierten Graphen von ehemals 200 Knoten und 240 Kanten, bei dem die kleineren Zusammenhangskomponenten entfernt wurden. Zusätzlich wurden 5 der höchstgradigen Knoten entfernt, um die Übersichtlichkeit zu erhalten. Anschließend wurde das Plugin springembedderFR [Mat06] mit Kantenabstoßung, einem optimalen Knotenabstand von 5,3 und einer hohen Gravitationskraft von 140 angewendet, um den Anforderungen der Algorithmen an das Layout zu genügen.

Labels wurden von dem im Umfang dieser Arbeit implementierten *Label Generators* erstellt. Dieser generierte zunächst pro Knoten ein Label. Außerdem wurden zusätzlich 40 Knotenlabels und 40 Kantenlabels zufällig verteilt.

Der Ausgangsgraph für den Test umfasst also 131 Knoten, 154 Kanten und 211 Labels, wobei nicht alle Kanten Labels besitzen, einige Knoten und Kanten allerdings mehrere. Wie Abbildung 36 zeigt, ist die Darstellung so kompakt, dass ein gültiges Labeling möglicherweise nicht existiert. In Ermangelung eines optimalen Labeling-Algorithmus kann dies jedoch nicht bestätigt werden.

Die Algorithmen werden ausschließlich mit Standardparametern gestartet. Die resultierenden Graphen werden in den Abbildungen 37, 38 und 39 dargestellt. Dafür benötigte Laufzeiten finden sich in Tabelle 2.

Spring Embedder	14,22s
Finite Positions	1,56s
Kombination FP&SE	14,84s

Tabelle 2: Laufzeiten der Algorithmen am Beispielgraph $131 \times 154 \times 211$ (je über 4 Läufe gerundet)

Für alle Algorithmen kann ausgesagt werden, dass die Labels am Rand des Graphen besser platziert werden als in der Mitte, da dort ihre Dichte größer ist.

Da im Algorithmus Finite Positions Überlappungen für jedes platzierte Label gezählt werden, steht diese Zählung als Statistik am Ende des Laufs zur Verfügung. Im Ergebnis von Abbildung 38 werden 3 Label-Label-Überlappungen gezählt, 7 Label-Knoten-Überlappungen und 51 Überlappungen mit Kanten.

Überlappungen von Labels sind beim Algorithmus Finite Positions seltener

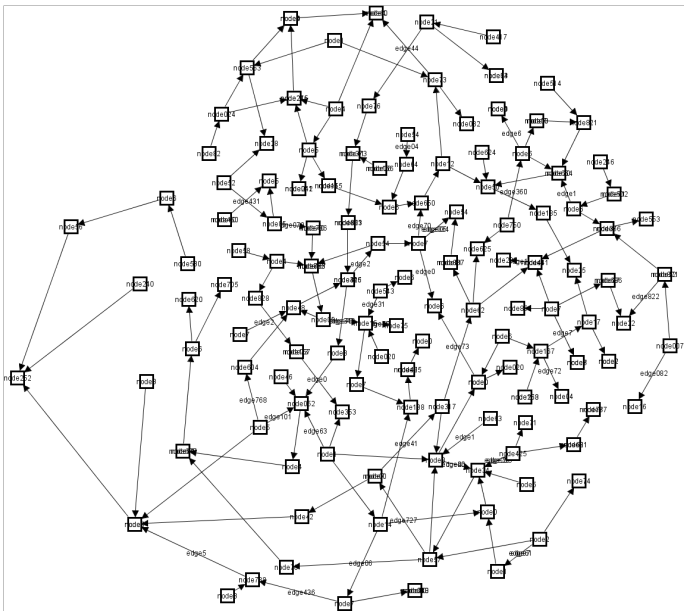

Abbildung 36: Der Ausgangsgraph des Tests 131×154×211

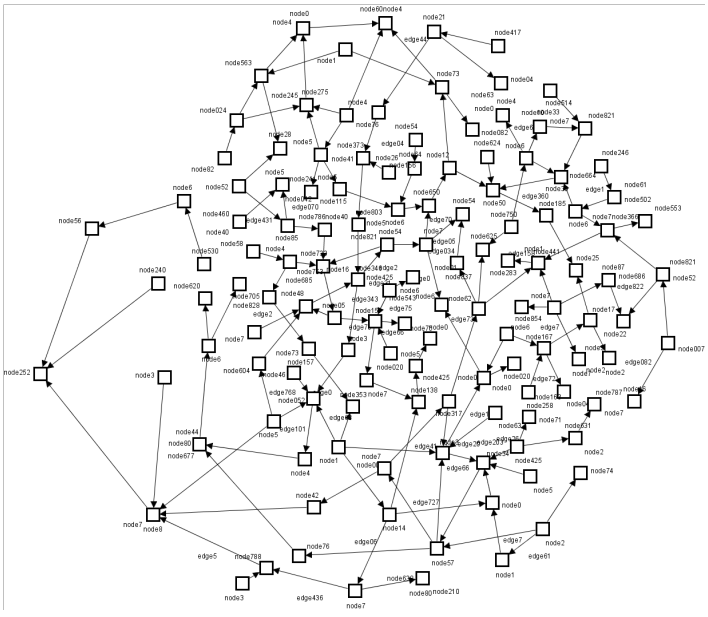

Abbildung 37: Der Graph 131×154×211 nach Anwendung des Spring Embedders

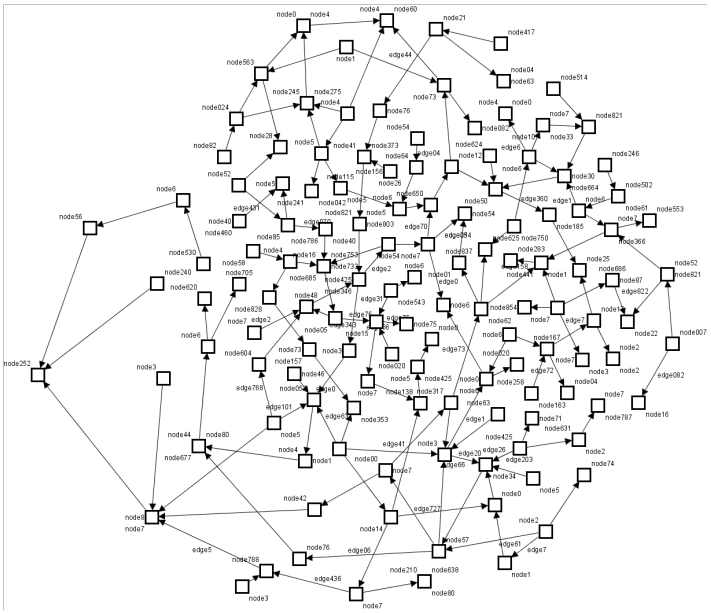

Abbildung 38: Der Graph 131×154×211 nach Anwendung von Finite Positions

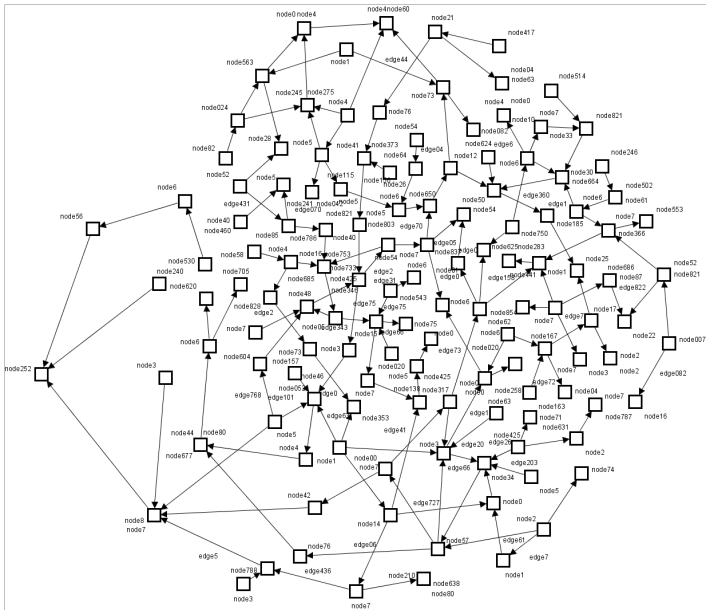

Abbildung 39: Der Graph 131×154×211 nach Anwendung des Kombinationsalgorithmus aus Finite Positions und Spring Embedder

als beim SPRING EMBEDDER, da diese in der Standardparametrisierung stark bestraft werden. Ebenso verhält es sich bei Label-Knoten-Überlappungen.

Der SPRING EMBEDDER unterscheidet die Stärke von Überlappungen. Daher sind leichte Überlappungen häufig, komplette Überlappungen kommen jedoch nicht vor.

Bei kurzen waagerechten Kanten misslingt die Positionierung durch FINITE POSITIONS, da auch die Richtung des Schriftverlaufs waagerecht ist. Dies ist der Grund für die meisten Knotenüberlappungen. Beim SPRING EMBEDDER wird dieses Problem durch einen größeren Abstand zwischen Label und Elterkante gelöst, allerdings erschwert dies die Zuordnung.

Beim Ergebnis des SPRING EMBEDDERS ist zu erkennen, dass initial schlecht gewählte Platzierungen von Labels kaum verbessert werden können. Durch die Kombination mit FINITE POSITIONS wird dieses Problem erheblich verbessert.

Insgesamt lässt sich aussagen, dass FINITE POSITIONS als Vorverarbeitung für den SPRING EMBEDDER bessere Ergebnissse für das Labeling liefert als eine vorgeschaltete *Quenching*-Phase.
Die Verwendung des SPRING EMBEDDERS zur Verbesserung der Ergebnisse von FINITE POSITIONS ist in Bereichen niedriger Dichte angemessen, führt aber in Bereichen hoher Dichte mitunter zu Überlappungen.

5.2 Experimentelle Laufzeituntersuchung

Um eine Aussage über die Laufzeit der verschiedenen Verfahren treffen zu können, wird ein entsprechendes Experiment durchgeführt.

Problematisch ist hierbei, eine Serie von Graphen zu finden, die für möglichst große Anzahl von Anwendungen repräsentativ ist. Graphen mit zufälliger Knotenplatzierung werden nicht als solche angesehen, da die Algorithmen zum Labeling von Graphen davon ausgehen, dass ein gewisser Knotenabstand eingehalten und die durchschnittliche Kantenlänge in der Darstellung bereits im Vorfeld minimiert werden.

Zu diesem Zweck wird ein regulärer Graph gewählt, der leicht an verschiedene Problemgrößen angepasst werden kann. Abbildung 40 zeigt den Aufbau des Graphen. Es sind $s \cdot s$, $s \in \mathbb{N}$ Knoten in einem festen Gitter angeordnet. Kanten gehen von jedem Knoten zum rechten als auch unteren Nachbarknoten, soweit vorhanden. Da dann mit jedem neuen Knoten nur eine konstante Anzahl an Kanten hinzugefügt wird, gilt $n \propto m$. Sowohl Knoten als auch Kanten werden mit jeweils einem Label versehen; Knoten mit der Knotennummer und Kanten mit einer zufälligen dreistelligen Zahl. Es gilt also der Zusammenhang $l = n + m$.

Optimierungen bei den Kollisionstests zwischen Labels wirken sich am gegebenen Graph gut aus, da eine ebenmäßige Verteilung zwischen den Knoten vorliegt. Raumaufteilende Strukturen wie Grid oder Quadtree [FB74] haben daher eine gute Effizienz. Die vorliegende Implementierung des FINITE POSITIONS LABEL PLACEMENT ALGORITHM verwendet jedoch keine dieser Strukturen und

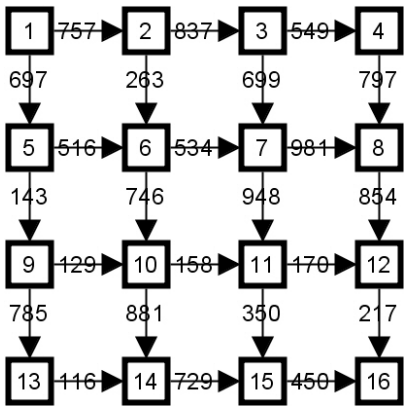

Abbildung 40: Aufbau des Testgraphen für die Laufzeituntersuchung ($s = 4$)

benötigt für die Kollisionserkennungsphase daher quadratische Laufzeit (siehe Abschnitt 4.2.5).

Es werden 4 verschiedene Konfigurationen getestet:

Finite Positions Der Algortihmus FINITE POSITIONS wird mit Standardparametern gestartet. Die theoretische Laufzeit wurde in Abschnitt 4.1.4 mit $\Theta(l \cdot (l+n+m))$ angegeben. Da im Beispiel alle drei Konstanten zusammenfallen, ist eine quadratische Laufzeitentwicklung in der Anzahl der Labels zu erwarten.

Spring Embedder Der Algortihmus SPRING EMBEDDER wird mit Standardparametern gestartet. Dies umfasst 10 Iterationen in der *Quenching*-Phase und 40 Iterationen in der *Simmering*-Phase.

SE mit Grid Der Algortihmus SPRING EMBEDDER wird mit Grid als raumaufteilende Datenstruktur gestartet. Es werden wiederum 10 Iterationen in der *Quenching*-Phase und 40 Iterationen in der *Simmering*-Phase durchgeführt.

Kombination FP&SE Der Algortihmus FINITE POSITIONS wird zuerst mit Standardparametern gestartet. Nach dessen Termination beginnt SPRING EMBEDDER ohne *Quenching*-Phase.

Abbildung 41 und 42 zeigt die Laufzeiten der vier Algorithmen am Beispielgraphen. Ergebnisse, die stochastisch nur unzureichend abgesichert sind, werden gestrichelt dargestellt. Bei den übrigen Messwerten werden 95%-Konfidenzintervalle eingezeichnet. Zur besseren Orientierung wurden der Abbildung zwei quadratische Funktionen hinzugefügt:

$$f_1(x) := 0,0004 \cdot x^2$$

$$f_2(x) := 0,00002 \cdot x^2$$

Es ist zu erkennen, dass die Laufzeit der Algorithmen im getesteten Bereich grob durch einen quadratischen Term dominiert wird. Eine Ausnahme bildet die Konfiguration *SE mit Grid*, bei der möglicherweise wegen Verwendung einer raumaufteilenden Datenstruktur eine geringere Potenz anzusetzen ist. Um hierzu eine genauere Aussage machen zu können, sind jedoch weitere Tests nötig.

Die Laufzeit von FINITE POSITIONS ist im Bereich etwa um den Faktor 20 kleiner als SPRING EMBEDDER mit Standardparametern. Dies hängt damit zusammen, dass der SPRING EMBEDDER als iterative Heuristik mehrere Platzierungen hintereinander berechnet.

Die Konfiguration *SE mit Grid* ist signifikant schneller als die Konfiguration *Spring Embedder* mit Standardparametern. Dies bestätigt die Effektivität des Grids als raumaufteilende Datenstruktur. Es werden dabei nicht alle kombinatorisch möglichen Paare von Objekten zur Kraftberechnung durchlaufen, sondern nur solche, deren Abstand einen gewissen Wert nicht überschreitet.

Die Konfiguration *Kombination FP&SE* hat gegenüber der Konfiguration *Spring Embedder* mit Standardparametern keine signifikant unterschiedliche Laufzeit. Einerseits wird auf 10 Iterationen der *Quenching*-Phase verzichtet, andererseits wird ein Programmlauf von FINITE POSITIONS vorgeschaltet, was sich in der Laufzeit in etwa ausgleicht.

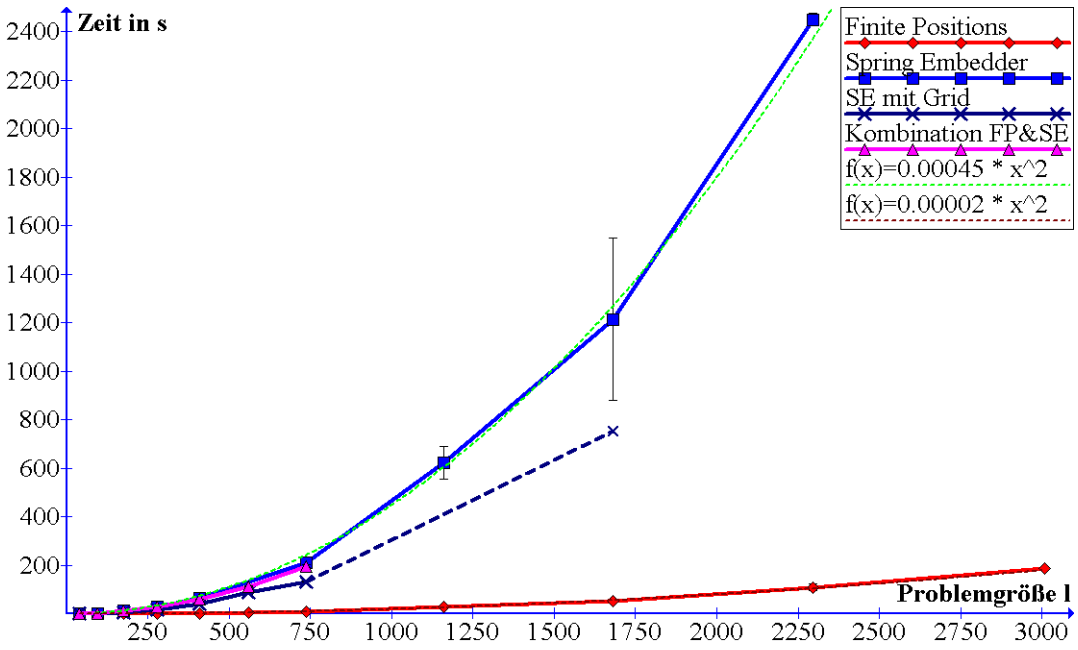

Abbildung 41: Laufzeiten der verschiedenen Algorithmen bei unterschiedlichen Gittergrößen mit 95% Konfidenzintervallen (lineare Zeitskala)

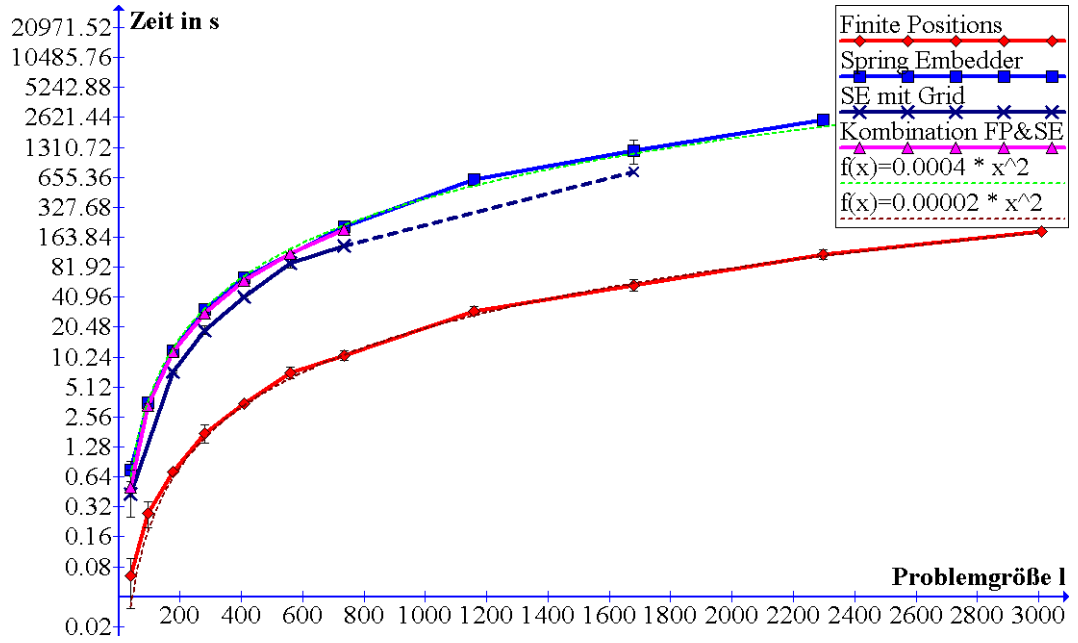

Abbildung 42: Laufzeiten der verschiedenen Algorithmen bei unterschiedlichen Gittergrößen mit 95% Konfidenzintervallen (logarithmische Zeitskala)

6 Zusammenfassung

Im Zuge dieser Arbeit ist es gelungen, sowohl durch die Entwicklung des FINITE POSITIONS LABELING ALGORITHM als auch durch die Erweiterung des Plugins SPRINGEMBEDDERFR von Marco Matzeder, zwei unterschiedliche Herangehensweisen zur Positionierung von Labels zur Verfügung zu stellen.

Der restriktive Algorithmus FINITE POSITIONS platziert heuristisch aus einer begrenzten Anzahl von möglichen Positionen. Das Verfahren ist deterministisch und schnell genug, um alle mit dem *Graph Visualization Toolkit* verwendbaren Größen von Graphen zu handhaben. Es ist für viele Anwendungsbereiche einsetzbar und das Ergebnis ist durch die nachvollziehbare Vorgehensweise und die einfache Parametrisierung vorhersehbar.

Der freie Algorithmus SPRING EMBEDDER ist eine physikalisch angelehnte, iterative Heuristik. Durch die Flexibilität des Verfahrens lässt er sich bei beliebigen Anwendungsszenarien einsetzen. Weiter kann er zur Verbesserung eines vorhandenen Labelings verwendet werden, da er einerseits eine vorhandene Positionierung nicht zerstört und andererseits der Umfang der Manipulation dosiert werden kann. Die große Anzahl an Parametern erlaubt eine Feinjustierung, jedoch ist diese mitunter zeitintensiver als bei FINITE POSITIONS. Bei einer initialen Positionierung von Labels an einem Graphen empfiehlt es sich, eine Vorverarbeitung zuzuschalten, um ein optimales Ergebnis zu erreichen.

Die Kombination von FINITE POSITIONS und SPRING EMBEDDER verbessert das Ergebnis wesentlich. Bei den meisten Anwendungen reichen die vorgegebenen Standardparameter für ein gültiges und graphisch ansprechendes Labeling aus. Um jedoch die Qualität bei dichten Anhäufungen von Graphobjekten beizubehalten, ist es sinnvoll, die Parameter für den SPRING EMBEDDER anzupassen. Dies ist mit weniger Aufwand möglich als bei alleiniger Verwendung des SPRING EMBEDDER, da die Einstellungen nur Teile der Positionierung betreffen und daher präziser optimiert werden können.

Literatur

[BBF+05] C. Bachmaier, F. Brandenburg, M. Forster, M. Raitner, and P. Holleis. Gravisto: Graph visualization toolkit, 2005.

[BETT98] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[CFMS97] Jon Christensen, Stacy Friedman, Joe Marks, and Stuart M. Shieber. Empirical testing of algorithms for variable-sized label placement. In *Symposium on Computational Geometry*, pages 415–417, 1997.

[DMM+97] Srinivas Doddi, Madhav V. Marathe, Andy Mirzaian, Bernard M. E. Moret, and Binhai Zhu. Map labeling and its generalizations. In *SODA*, pages 148–157, 1997.

[DMS05] Tim Dwyer, Kim Marriott, and Peter J. Stuckey. Fast node overlap removal. In *Graph Drawing*, pages 153–164, 2005.

[Ead84] Peter Eades. A heuristic for graph drawing. 42:149–160, 1984.

[ECMS96] Shawn Edmondson, Jon Christensen, Joe Marks, and Stuard M. Shieber. A general cartographic labelling algorithm. In *Cartographica*, volume 33, pages 13–26. University of Toronto Press, 1996.

[FB74] Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.

[FR91] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[FW91] Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *SCG '91: Proceedings of the seventh annual symposium on Computational geometry*, pages 281–288, New York, NY, USA, 1991. ACM.

[Gut84] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, New York, NY, USA, 1984. ACM.

[HIMF98] Kunihiko Hayashi, Michiko Inoue, Toshimitsu Masuzawa, and Hideo Fujiwara. A layout adjustment problem for disjoint rectangles

preserving orthogonal order. In *GD '98: Proceedings of the 6th International Symposium on Graph Drawing*, pages 183–197, London, UK, 1998. Springer-Verlag.

[Hir82] Stephen A. Hirsch. An algorithm for automatic name placement around point data. *Cartography and Geographic Information Science*, 9:5–17(13), April 1982.

[IA86] Hiroshi Imai and Takao Asano. Efficient algorithms for geometric graph search problems. *SIAM Journal on Computing*, 15(2):478–494, 1986.

[IL99] Claudia Iturriaga and Anna Lubiw. Elastic labels around the perimeter of a map. In *WADS*, pages 306–317, 1999.

[Imh62] Eduard Imhof. Die anordnung von namen in der karte. In *Internationales Jahrbuch fnr Kartographie*, 1962.

[Imh75] *Positioning Names on Maps*, volume 2 of *The American Cartographer*, 1975.

[IV99] Claudia Carmen Iturriaga-Velazquez. *Map labeling problems.* PhD thesis, University of Waterloo, Waterloo, Ont., Canada, Canada, 1999. Adviser-Anna Lubiw.

[KT01] Konstantinos G. Kakoulis and Ioannis G. Tollis. On the complexity of the edge label placement problem. *Comput. Geom.*, 18(1):1–17, 2001.

[KT06] Konstantinos G. Kakoulis and G. Tollis. Algorithms for the multiple label placement problem. In *Computational Geometry*, pages 143–161, 2006.

[Lyo92] Kelly A. Lyons. Cluster busting in anchored graph drawing. In *CASCON '92: Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research*, pages 7–17. IBM Press, 1992.

[Mat06] Marco Matzeder. Erweiterung eines spring embedder verfahrens. Master's thesis, University of Passau, 2006.

[MS91] Joe Marks and Stuart Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard University, 1991.

[RND77] Edward M. Reingold, Jurg Nievergelt, and Narsingh Deo. *Combinatorial Algorithms: Theory and Practice*. Prentice Hall College Div, 1977.

[SVA00] Tycho Strijk, Bram Verweij, and Karen Aardal. Algorithms for maximum independent set applied to map labelling. Technical Report UU-CS-2000-22, Utrecht University, Netherlands, 2000.

[Wit58] H. Wittke. Elektronische schreib- und kartiermaschinen - wünsche, pläne, möglichkeiten. In *Vermessungstechnische Rundschau*, volume 6, pages 185–197, 235–242, 1958.

[WW97] Frank Wagner and Alexander Wolff. A practical map labeling algorithm. *Comput. Geom.*, 7:387–404, 1997.

[Yoe72] *The logic of automated map Lettering*, volume 9(2) of *The Cartographic Journal*, 1972.

Eidesstattliche Erklärung

Ich versichere, diese Arbeit selbständig verfasst und nur die angegebenen Quellen verwendet zu haben.

Passau, den 11. August 2008

(Wolfgang Scholz)