Entwicklung und Evaluation eines Edge-Routingverfahrens für Softwarelandschaften

Bachelorarbeit

Marcus Uhlig

Matrikel: 2727182

Studiengang: eBusiness

Betreuer:

Prof. Dr. Claus Lewerentz

Lehrstuhl Software-Systemtechnik

Brandenburgische Technische Universität Cottbus

03.01.2012

Inhaltsverzeichnis

1. Einleitung

1.1 Motivation

Zur Bedeutung von Software

Dass Software in der heutigen Welt eine wesentliche Rolle spielt, ist kein Geheimnis. Software ist allgegenwärtig und übernimmt immer mehr und auch immer komplexere Aufgaben. Damit einhergehend steigert sich auch die Komplexität von Software ständig. Alle 2 bis 4 Jahre verdoppelt sich heutzutage der Umfang von Software, verbunden auch mit zunehmender Multifunktionalität der Softwaresysteme und Heterogenität assoziierter technischer Komponenten [2].

Je mehr aber eine Sache alle Bereiche des Lebens durchdringt, desto größer werden auch die Anforderungen an deren Qualität und vor allem Zuverlässigkeit. Man denke nur an die Abhängigkeit unserer Zivilisation vom Stromnetz. Ebenso wie Stromausfälle kann auch der Ausfall von Software, je nach Zeitraum und Bedeutung, erhebliche Schäden verursachen, zumindest wirtschaftliche.

Dieser Entwicklung gegenüber steht eine Entwicklung der Absolventenzahlen an Hochschulen im IT-Bereich, die mit der Nachfrage nach Fachkräften nicht Schritt halten kann. Basierend auf Zahlen des Branchenverbandes BITKOM und des Statistischen Bundesamtes schätzt Balzert die jährliche Lücke in Deutschland auf etwa 5000 Absolventen [2].

Es muss also immer mehr Software, mit steigender Komplexität und Qualität, mit relativ betrachtet immer weniger Personal – und damit Zeit – entwickelt werden. Dabei wäre es nicht praktikabel, Software immer wieder vollkommen neu zu entwickeln. Software bleibt häufig über Jahrzehnte im Einsatz und muss dabei ständig an sich verändernde Bedingungen angepasst werden. Nach Balzert fallen etwa 2/3 der Kosten, die Software verursacht, auf diese Anpassungen [2].

Zur Bedeutung von Softwarelandschaften als Visualisierungsmöglichkeit

Gerade bei wechselndem Personal und zunehmend knapperem Zeitrahmen ist es deshalb besonders wichtig, auch komplexe Software möglichst gut und schnell überschaubar zu machen. Je intuitiver Zusammenhänge und Strukturen dabei zu erkennen sind, desto besser.

Einen modernen Ansatz, Software zu visualisieren, stellt dabei die Stadt-Metapher dar (Abbildung 1). Die Nützlichkeit von Städten als mentale Bilder für Softwaresysteme liegt vor allem darin begründet, dass der Mensch zum einen den Umgang mit Städten und deren wesentlichen Elementen gewohnt ist, das Bild bei gelungener Nutzung also schnelle intuitive Schlüsse über die Struktur des dargestellten Softwaresystems zulässt und fördert. Zum anderen besteht eine gewisse strukturelle Verwandtschaft zwischen Softwaresystemen und Städten. Beide werden von Menschen geschaffen, wobei sie mehr oder weniger stark geplant entstehen oder aber historisch wachsen. Außerdem, vielleicht am wichtigsten, sind sie

sich ständig ändernden Anforderungen unterworfen und müssen deshalb immer wieder an neue Gegebenheiten angepasst werden [14].

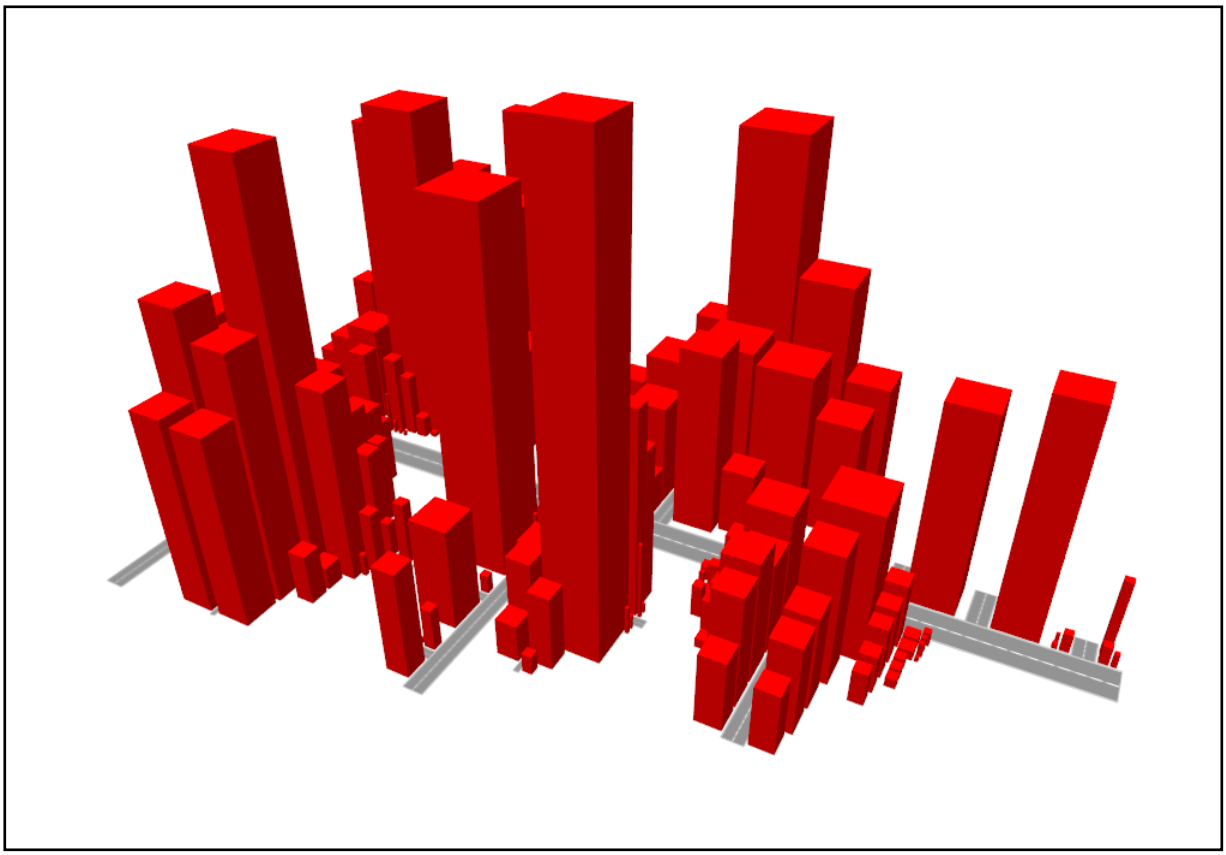

Abbildung 1: Eine einfache Softwarestadt bestehend aus Klassen (rote Gebäude), geordnet in einer hierarchischen Paketstruktur (Straßennetz).
Quelle: Ausdruck der Visualisierung im Evoviewer (3D-Softwarevisualisierungstool des Lehrstuhls Software-Systemtechnik der Brandenburgischen Technischen Universität Cottbus).

Je größer allerdings das darzustellende Softwaresystem ist, desto schwieriger kann eine klare Darstellung aller relevanten Informationen in einer einzigen Stadt sein, ebenso wie die Orientierung in einer realen Stadt mit zunehmender Größe schwieriger wird. Um also mehr Freiheiten in der Darstellung von Informationen zu gewinnen, lässt sich das mentale Bild der Stadt erweitern, indem man über die Grenzen einer einzigen Stadt hinausgeht und ein Softwaresystem als Landschaft darstellt, in der mehrere Städte Teilsysteme des Gesamtsystems repräsentieren und deren Lagebeziehungen zu- und Verflechtungen untereinander die Zusammenhänge zwischen den Teilsystemen visualisieren [1]. Abbildung 2 stellt das Softwaresystem aus Abbildung 1 beispielhaft als Landschaft dar.

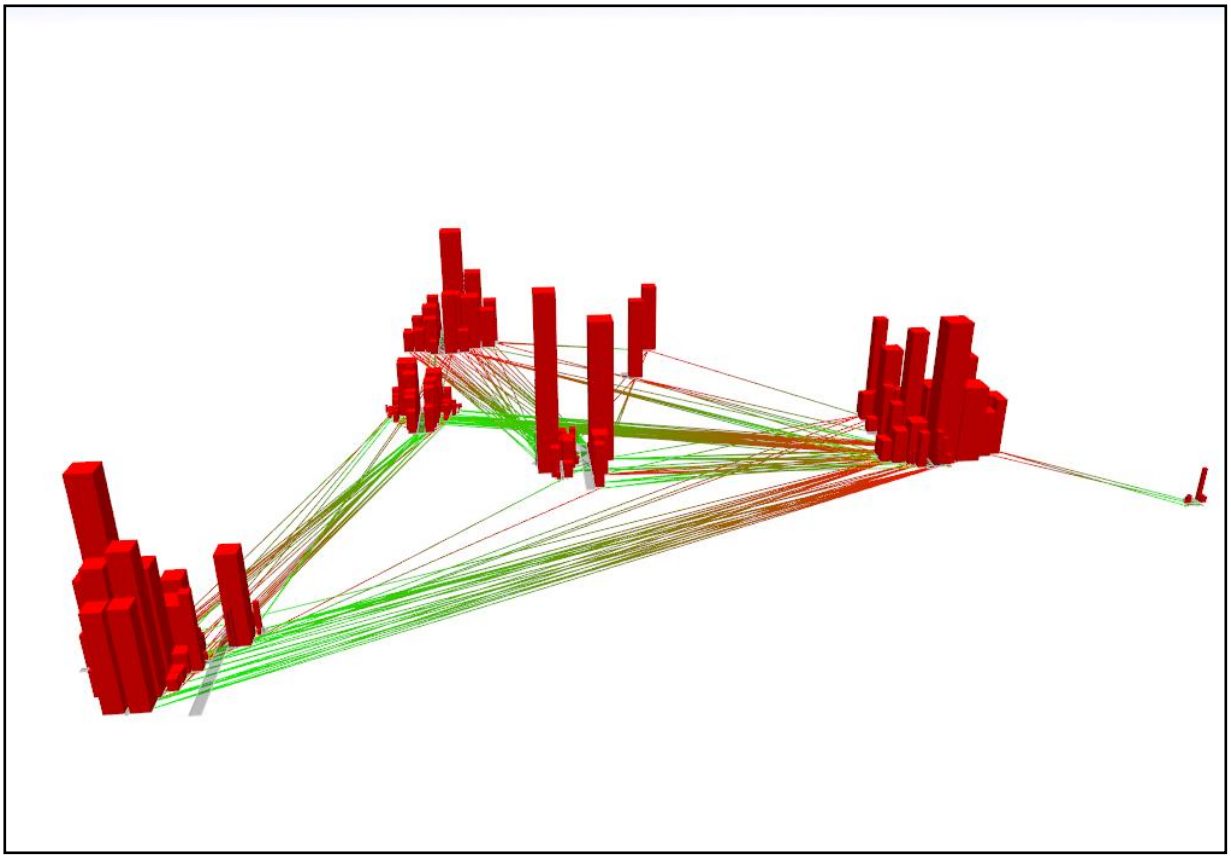

Abbildung 2: Die Softwarestadt aus Abbildung 1 dargestellt als Softwarelandschaft. Die grün-roten Linien stellen Abhängigkeiten zwischen Klassen dar, wobei der Farbverlauf die Richtung der Assoziation anzeigt.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Zur Visualisierung von Elementabhängigkeiten in Softwarelandschaften

Die statische Struktur eines Softwaresystems wird in Softwarestädten und -landschaften durch Strukturelemente realer Städte und Landschaften visualisiert. In Bezug zu objektorientierten Softwaresystemen können beispielsweise einzelne Klassen als Gebäude dargestellt werden, mit unterschiedlicher Grundfläche und Höhe in Abhängigkeit der Größe der dargestellten Klassen. Übergeordnete Organisationsstrukturen, wie zum Beispiel Pakete, in denen zusammengehörige Softwareelemente zusammengefasst werden, können als Straßen dargestellt werden, an denen die zugehörigen Gebäude liegen. Diese Systematik kann über mehrere Hierarchieebenen fortgeführt werden, indem kleinere Straßen (Unterpakete) in größeren zusammenlaufen (Oberpakete). Schließlich können in Softwarelandschaften Pakete bis zu einer bestimmten Ebene zu einer Stadt zusammengefasst werden, sodass diese Stadt in der Landschaft mehrerer Städte als Teilsystem des Gesamtsystems betrachtet werden kann.

Diese Darstellung der strukturellen Ordnung von Elementen (Klassen) stellt aber nur einen Teil der relevanten Informationen dar. Besonders wichtig sind darüber hinaus Abhängigkeiten zwischen einzelnen Elementen (Gebäuden) und Teilsystemen (Städten), beispielsweise um den Kopplungsgrad zwischen Teilsystemen erkennen zu können. Damit ließe sich zum

Beispiel abschätzen, wie weitreichend Auswirkungen von Änderungen an einem Teilsystem sein könnten. Für die Darstellung dieser Abhängigkeiten lassen sich Kanten verwenden, wenn man die einzelnen Elemente des Systems als Knoten eines Graphen betrachtet (Abbildung 2).

Zum Routing von Kanten

Das Routing dieser Kanten stellt nun allerdings kein triviales Problem dar. Zeichnet man die Kanten auf direktem Wege zwischen Start- und Zielknoten, entstehen schnell Überschneidungen mit anderen Objekten, wie Gebäuden und Städten, die die Lesbarkeit stark beeinträchtigen können (Abbildung 3).

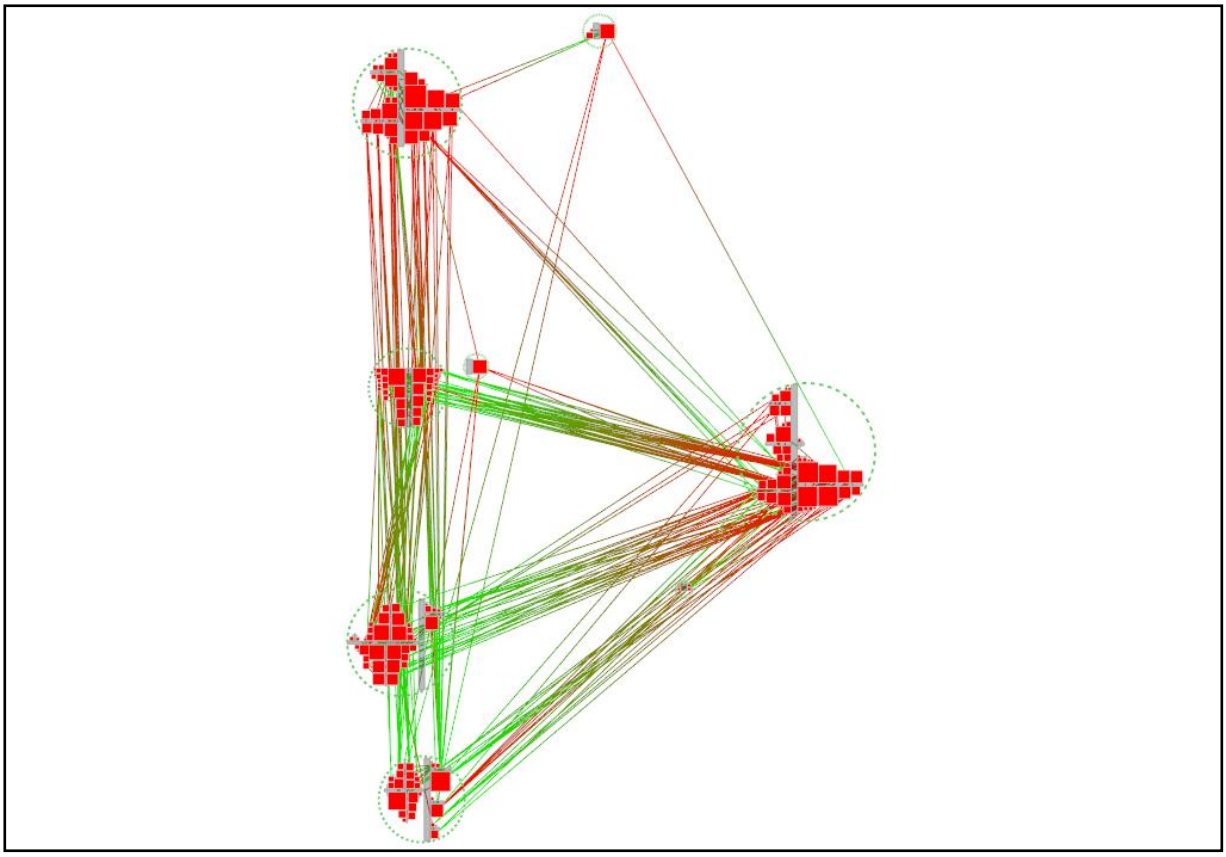

Abbildung 3: Softwarelandschaft mit Kanten ohne explizites Kantenrouting (Draufsicht).
Quelle: Ausdruck der Visualisierung im Evoviewer.

Zur Vermeidung solcher Überschneidungen wurde am Lehrstuhl Software-Systemtechnik der Brandenburgischen Technischen Universität Cottbus ein kräftebasiertes Routingverfahren entwickelt (Abbildung 4).

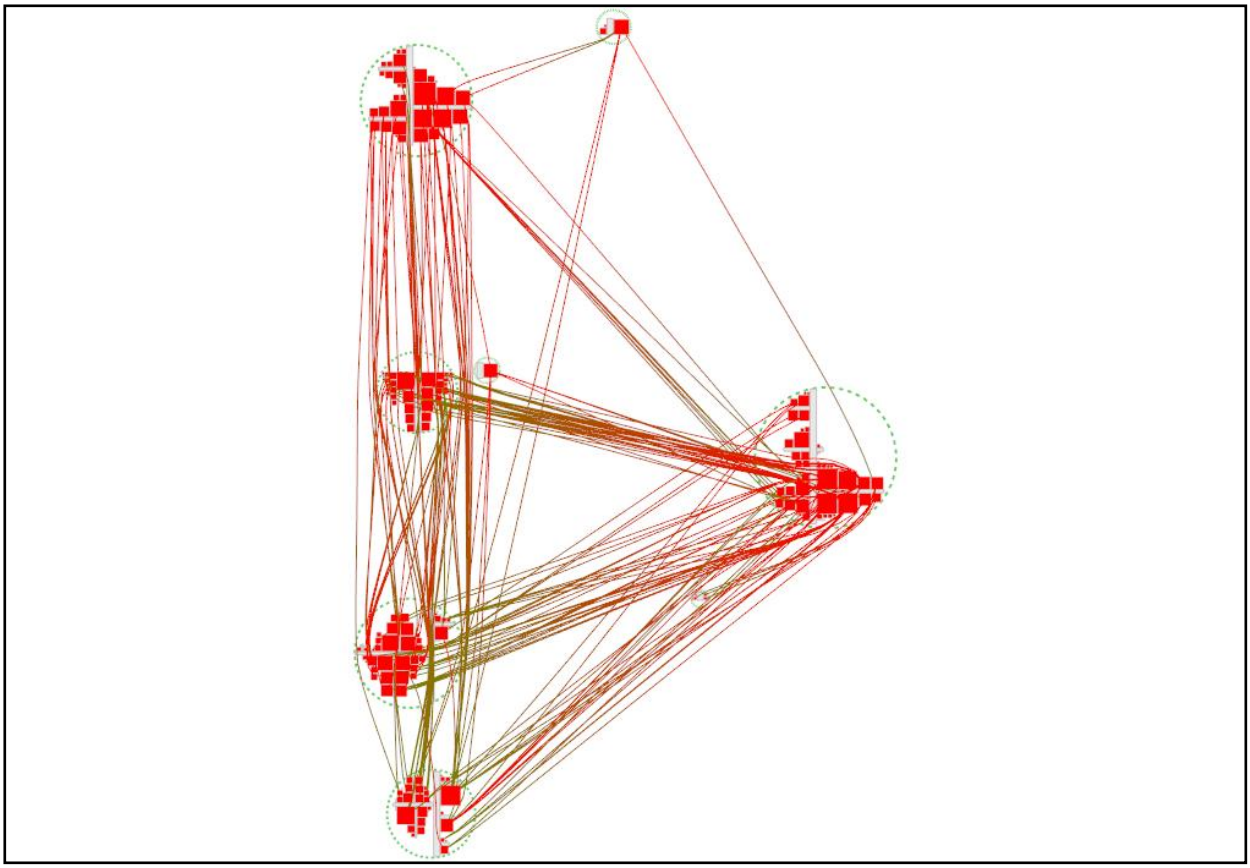

Abbildung 4: Die Softwarelandschaft aus Abbildung 3 mit kräftebasiertem Kantenrouting.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Grundidee des Verfahrens ist, dass Objekte (Gebäude) in Abhängigkeit ihrer Lage relativ zu einer Kante, Abstoßungskräfte auf diese ausüben (Abbildung 5a). Es leuchtet ein, dass Durchschneidungen dieser Objekte damit vermieden werden. Umgesetzt wird die Idee durch Annäherung der Kante mittels einer Reihe von Punkten, die als Angriffspunkte für die wirkenden Kräfte dienen (Abbildung 5b). Durch wiederholte kleine Bewegungen, die sich aus der Kombination aller beteiligten Kraftvektoren ergeben, werden die Punkte, und damit die Kante, so nach und nach an eine energetisch minimale Position angenähert. Um dabei die Stabilität der Kante an sich zu gewährleisten, d.h. um zu verhindern, dass die einzelnen Punkte einer Kante zu weit auseinanderdriften, wird außerdem eine gegenläufige Anziehungskraft zwischen den benachbarten Punkten einer Kante verwendet (Abbildung 5c) [10].

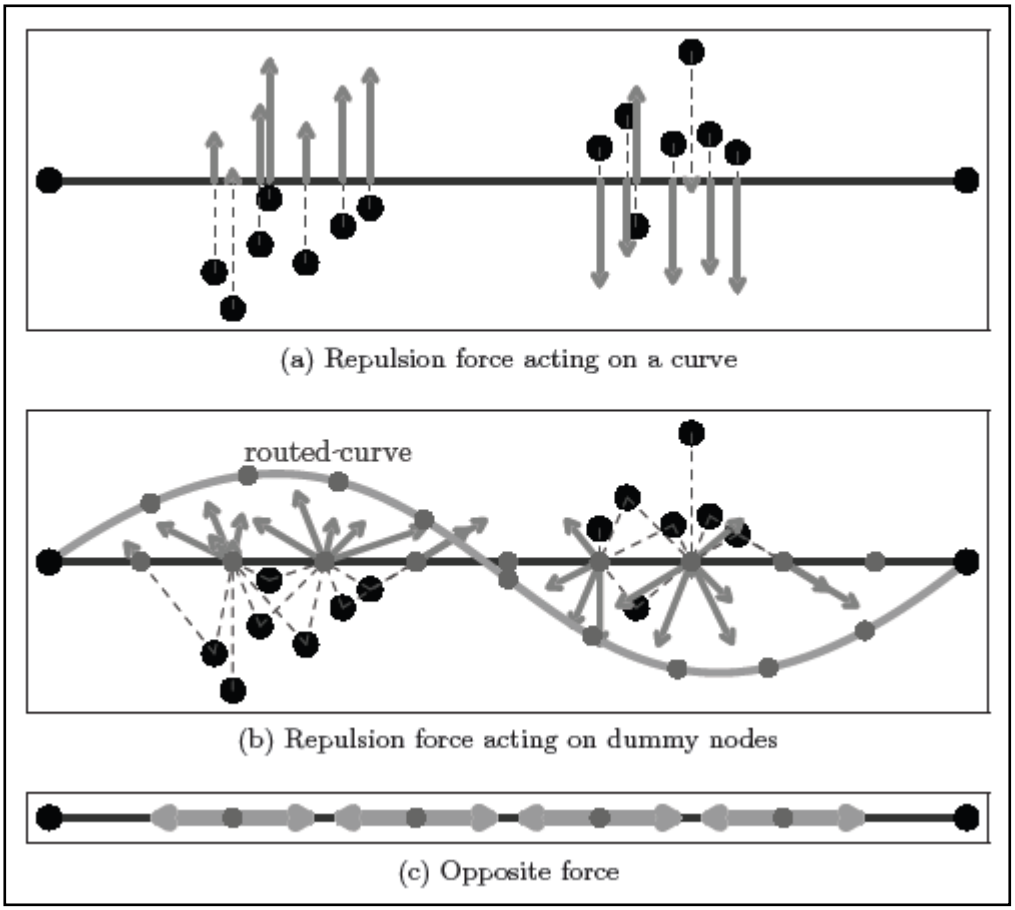

Abbildung 5: Idee des kräftebasierten Verfahrens von [10].
Quelle: [10].

Dieses Verfahren löst das Problem allerdings nur teilweise. Zwar werden durchaus Überschneidungen mit Städten vermieden, vollständige Überschneidungsfreiheit kann allerdings nicht garantiert werden. Hauptproblem des Verfahrens ist außerdem, dass optisch gute Ergebnisse nur mit hoher Rechenzeit erzielt werden können.

1.2 Ziel

Ziel dieser Arbeit ist es, ein geometrisches Kanten-Routingverfahren als Alternative zum im vorherigen Abschnitt skizzierten kräftebasierten Verfahren zu entwickeln. Dieses Verfahren soll nicht nur optisch ansprechende Ergebnisse liefern, indem es unter anderem Stadtdurchschneidungen vermeidet. Es soll darüber hinaus deutlich weniger Rechenzeit benötigen, als das kräftebasierte Verfahren. Das Verfahren soll zudem ohne Nutzung der dritten Dimension auskommen, um auch für Ausdrucke in der Draufsicht eine Lösung zu bieten.

1.3 Gliederung

Zur Entwicklung eines geeigneten Verfahrens wird zunächst in Kapitel 2 ein Überblick über bekannte Kanten-Routingverfahren gegeben. Es wird versucht, diese zu klassifizieren und hinsichtlich ihrer Eignung zur Erreichung des definierten Ziels grob einzuschätzen.

Auf den daraus gewonnenen Erkenntnissen aufbauend wird in Kapitel 3 ein eigenes Verfahren unter Beachtung zu definierender Zielkriterien entwickelt.

Anschließend wird das entwickelte Verfahren in Kapitel 4 hinsichtlich der definierten Zielkriterien bewertet und mit dem kräftebasierten Verfahren verglichen.

Im abschließenden Kapitel 5 werden die wesentlichen Resultate schließlich zusammengefasst und ein Ausblick auf mögliche Weiterentwicklungen gegeben.

2. Verwandte Arbeiten

In diesem Kapitel werde ich einen Überblick über bereits entwickelte Kanten-Routingverfahren geben. Ziel ist es dabei einerseits, deren Eignung für den Anwendungsfall in dieser Arbeit einzuschätzen, andererseits hilfreiche Erkenntnisse für die Entwicklung eigener Verfahren zu erlangen.

Zum Routing von Kanten wurden schon verschiedenste Verfahren vorgeschlagen, nicht nur in Verbindung mit Softwarestädten und -landschaften. Viele dieser Ansätze scheiden für unseren Anwendungsfall von vornherein aus naheliegenden Gründen aus. Als Beispiele seien hier die Lösungsvorschläge von [5], [3] und [6] genannt, die sich jeweils die dritte Dimension zu Nutze machen, was der in Kapitel 1.2 definierten Zielstellung dieser Arbeit widerspricht. Auf diese Ansätze werde ich hier deshalb nicht weiter eingehen.

Kräftebasierte Ansätze

Neben dem Lehrstuhl Software-Systemtechnik der Brandenburgischen Technischen Universität Cottbus (siehe Kapitel 1.1 und Abbildung 4) schlagen auch [9] ein kräftebasiertes Kanten-Routingverfahren vor (Abbildung 6).

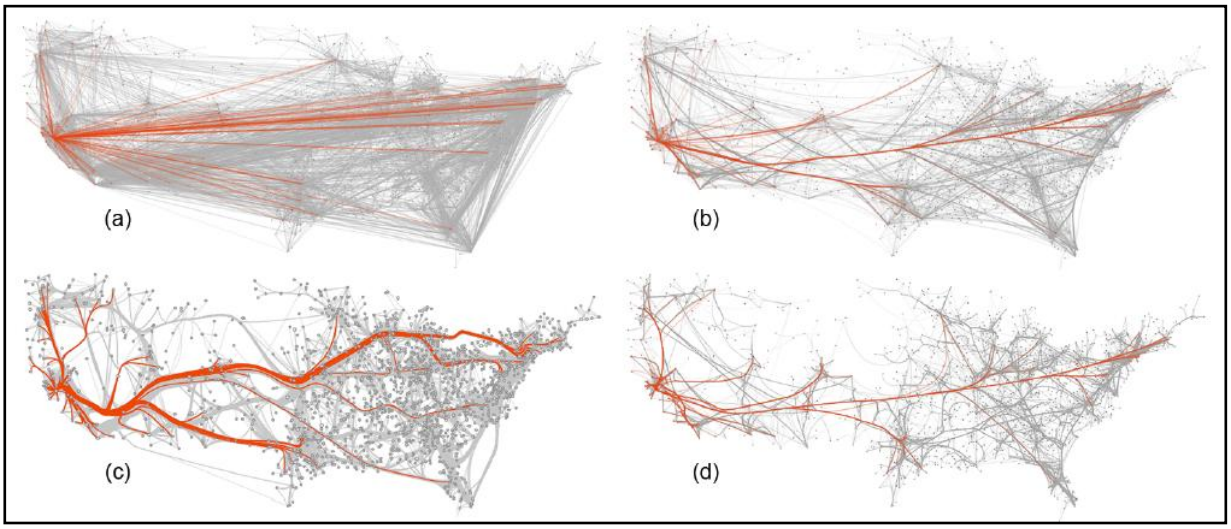

Abbildung 6: Graph mit 1715 Knoten und 9780 Kanten ohne Routing (a) und mit verschiedenen Ausprägungen des kräftebasierten Verfahrens von [9] (b, c, d). Die jeweils gleiche Auswahl an Kanten ist orange markiert.
Quelle: [9].

Im Gegensatz zu dem in Kapitel 1.1 vorgestellten Verfahren, steht beim Verfahren von [9] die Bündelung der Kanten im Mittelpunkt. Auch hier wird jede Kante durch eine Reihe von Angriffspunkten angenähert und es sichern Anziehungskräfte zwischen benachbarten Punkten die Stabilität der Kante. Statt der Abstoßungskräfte anderer Objekte, werden in diesem Verfahren jedoch Anziehungskräfte zwischen äquivalenten Punkten je zweier Kanten verwendet (Abbildung 7). Es ist klar, dass das Verfahren Objektdurchschneidungen damit nicht explizit vermeidet. Nur als Folge der Bündelung werden diese implizit reduziert (vgl. Abbildung 6 a vs. c).

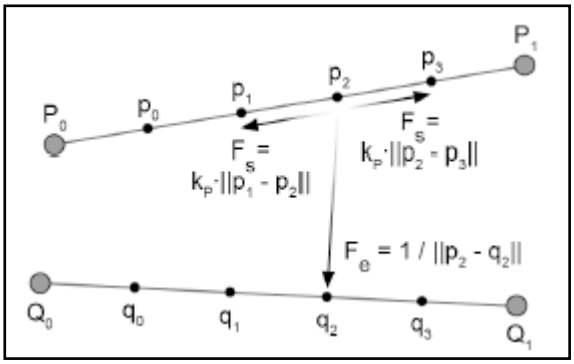

Abbildung 7: Idee des kräftebasierten Verfahrens von [9].
Quelle: [9].

Obwohl beide Verfahren optisch gute Ergebnisse liefern, nennen auch [9] die hohe Rechenzeit als größten Nachteil.

Flow Map Layouts

Flow Map Layouts stammen aus der Kartographie. Ursprünglich von Hand gezeichnet, werden sie beispielsweise zur Darstellung von Migrations- und Güterbewegungen genutzt. Eine entsprechende rechentechnische Lösung wurde von [11] vorgeschlagen (Abbildung 8).

Grundidee des Verfahrens ist ein hierarchisches Clustering der Knoten des Graphen und Routing der Kanten zwischen den entstehenden Clustern. Dabei werden benachbarte Knoten bzw. Knotencluster binär zu einem Knotencluster zusammengefasst. Das Verfahren wird hierarchisch fortgesetzt, sodass eine Baumstruktur bezüglich der Entfernungen der Knoten zueinander entsteht. Kanten werden daraufhin im Baum absteigend jeweils vom Vater zu dessen Kindern berechnet. Ist ein Kind ein Knoten des ursprünglichen Graphen, wird die Kante gerade zu diesem gezeichnet. Ist das Kind seinerseits ein Cluster, wird die Kante in Richtung des Mittelpunkts des Cluster gezeichnet. Zur optischen Optimierung werden die so berechneten Punkte als Stützpunkte für Splines verwendet [11].

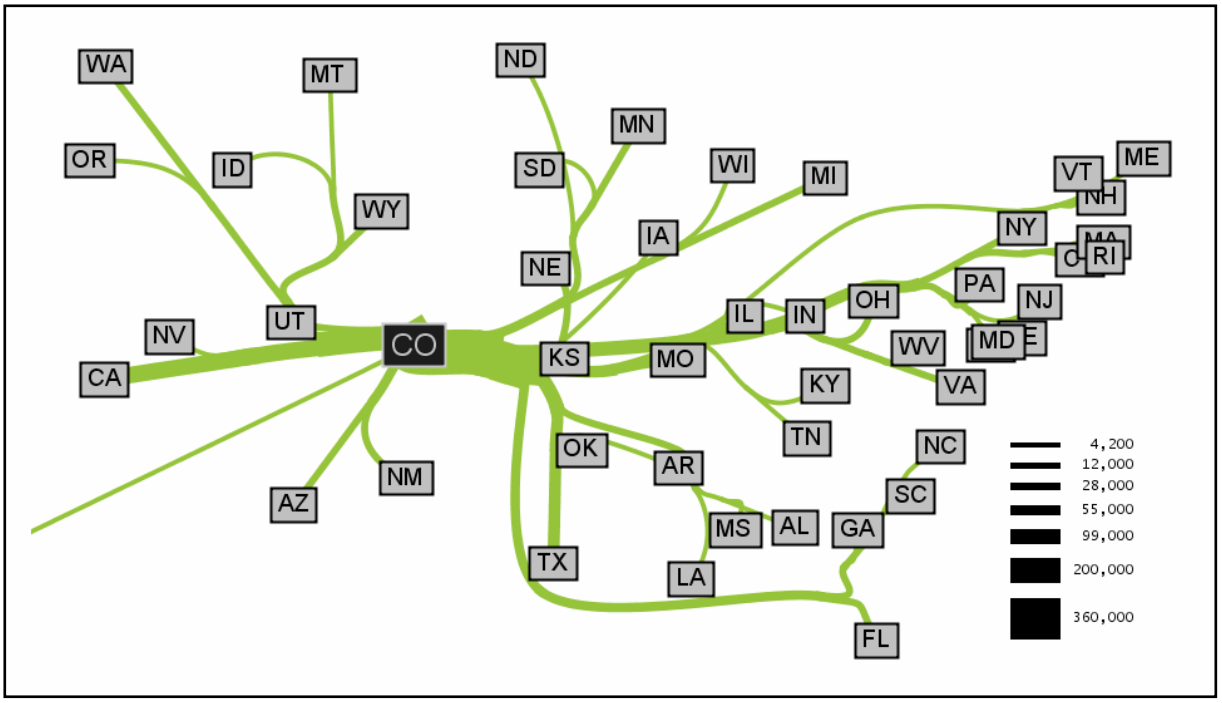

Abbildung 8: Flow Map zu Migrationsbewegungen in den USA von Colorado aus für die Jahre 1995-2000, erzeugt mit dem Layoutverfahren von [11].
Quelle: [11].

Ausgehend vom Startknoten sind die Größenordnungen globaler Verbindungen gut erkennbar. So sind in Abbildung 8 beispielsweise größere Wanderungsbewegungen nach Kalifornien und an die nördliche Ostküste zu beobachten. Übertragen auf Software-Landschaften wären High-Level-Verbindungen zwischen Teilsystemen mit diesem Verfahren gut darstellbar. Low-Level-Verbindungen zwischen einzelnen Elementen könnte man hingegen gar nicht mehr erkennen, da die Einzelkanten vollkommen zu globalen Strömen zusammengefasst sind. Hier wäre eine gewisse Flexibilität des Verfahrens wünschenswert, sodass man je nach Interessenlage die Bündelung variabel einstellen könnte. Dies sollte bei der Entwicklung eines eigenen Verfahrens berücksichtigt werden.

Das größte Problem des Flow Map Layouts stellt allerdings der Bezug auf einen Start- oder Endknoten dar. Während Flow Maps immer von einem Startknoten ausgehen oder auf einen Zielknoten zusteuern, von dem aus die Kantenzusammenfassung berechnet wird, stellt in einer Softwarelandschaft jeder Knoten gleichzeitig Start- und Endknoten für diverse aus- bzw. eingehende Kanten dar. Um das Verfahren auf unseren Anwendungsfall zu übertragen, könnte man das Verfahren zwar beispielsweise auf jede Softwarestadt separat anwenden und die so entstehenden Teilkarten geeignet kombinieren. Ob das Ergebnis allerdings qualitativ angemessen wäre, bleibt zu bezweifeln.

Kontrollnetzlösungen

Sowohl [12] als auch [4] schlagen Kontrollnetzlösungen vor. Dabei wird mit verschiedenen Strategien ein Netz aus Kontrollpunkten generiert und alle Kanten per Shortest-Path-Routing

durch diese Kontrollpunkte geleitet. Beide Verfahren unterscheiden sich dabei vor allem durch die Strategie zum Generieren des Kontrollnetzes.

[12] generieren das Kontrollnetz durch Finden und geeignetes Zusammenfassen der Schnittpunkte zwischen den Originalkanten und den Kanten der Delaunay Triangulation der Originalknoten des Graphen (Abbildung 9).

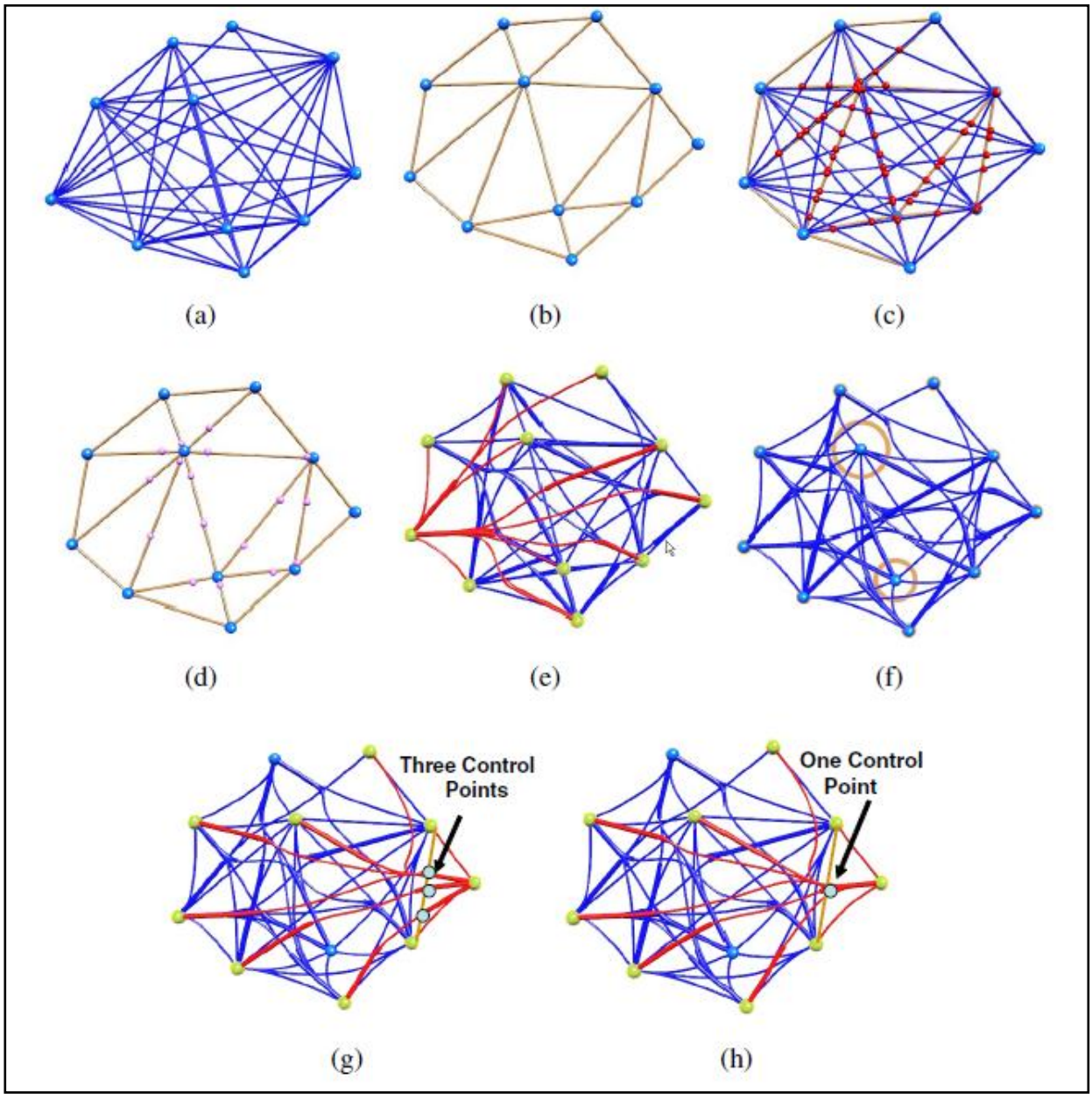

Abbildung 9: Kanten-Routingverfahren nach [12]. (a) Kanten ohne Routing, (b) Delaunay Triangulation der Knoten, (c) Schnittpunkte der Delaunay-Kanten mit den Originalkanten, (d) Zusammenfassung der Schnittpunkte zum Kontrollnetz, (e) Routing der Kanten durch das Kontrollnetz, (f) Aussparung um Knoten, (g, h) unterschiedliches Kontrollnetz durch unterschiedlich ausgeprägte Zusammenfassung der Schnittpunkte.
Quelle: [12].

[4] schlagen sowohl eine manuelle Positionierung der Kontrollnetzpunkte als auch eine Generierung des Kontrollnetzes aufgrund der primären Kantenrichtungen innerhalb des Graphen vor (Abbildung 10).

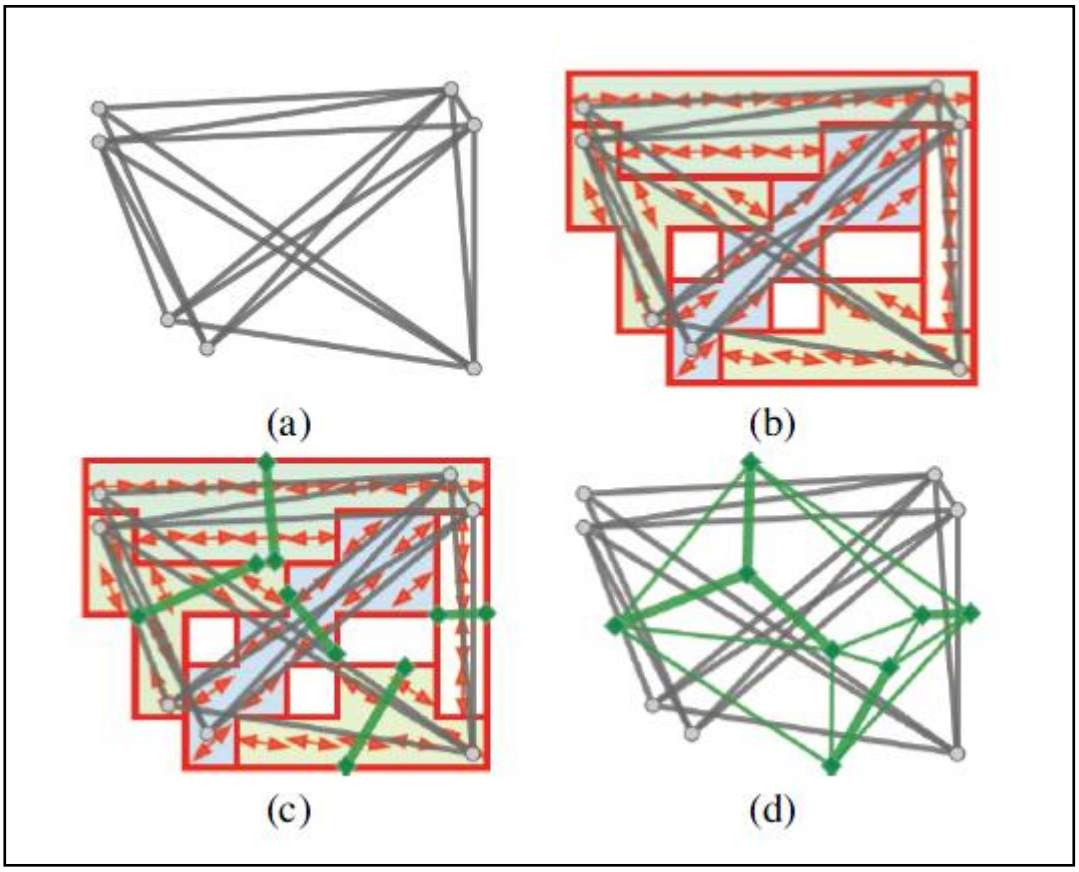

Abbildung 10: Kanten-Routingverfahren nach [4]. (a) Kanten ohne Routing, (b) Zusammenfassung von Zellen mit ähnlicher Richtung (in Abhängigkeit eines Maximalwinkels), (c) Konstruktion der Knoten für ein per Delaunay Triangulation zu erstellendes Netz (d). Das weitere Vorgehen entspricht Abbildung 9c-e.
Quelle: [4].

Im Gegensatz zu der im vorherigen Abschnitt dargestellten Flow Map Lösung bietet diese Herangehensweise den erheblichen Vorteil der Konfigurierbarkeit in Hinblick auf die Stärke der Kantenbündelung. Nach Generierung der Delaunay-Kanten und Finden von deren Schnittpunkten mit den Originalkanten können diese Schnittpunkte mehr oder weniger stark zusammengefasst werden, was zu unterschiedlichem Bündelungsgrad führt (Abbildung 9g-h). Damit können je nach Konfiguration High-Level- oder Low-Level-Verbindungen besonders gut erkannt werden.

Das Verfahren bietet allerdings auch einige Nachteile. Erstens können besonders längere Kanten, da sie potentiell durch viele Kontrollpunkte geleitet werden müssen, zu Verläufen mit sehr vielen Kurven führen. Dadurch können unnötig viele Überschneidungen mit anderen Kanten und unnötig lange Kanten entstehen. Außerdem kann die Erkennbarkeit der primären Richtung von Kanten damit verloren gehen. Zweitens werden die Kanten nur aufgrund ihrer Verlaufsrichtung gebündelt, nicht jedoch aufgrund ihrer semantischen Zusammengehörigkeit. Beide Punkte, insbesondere aber der zweite sollte im Weiteren berücksichtigt werden.

Hierarchical Edge Bundles

[8] leitet Kanten in seinem Verfahren als Splines entlang der Knotenhierarchie. Zur Bildung einer Kante wird dazu der Pfad bis zum nächsten gemeinsamen Vorfahren von Start- und Zielknoten in der Hierarchie gesucht und der so gefundene Pfad als Kontrollpolygon genutzt (Abbildung 11).

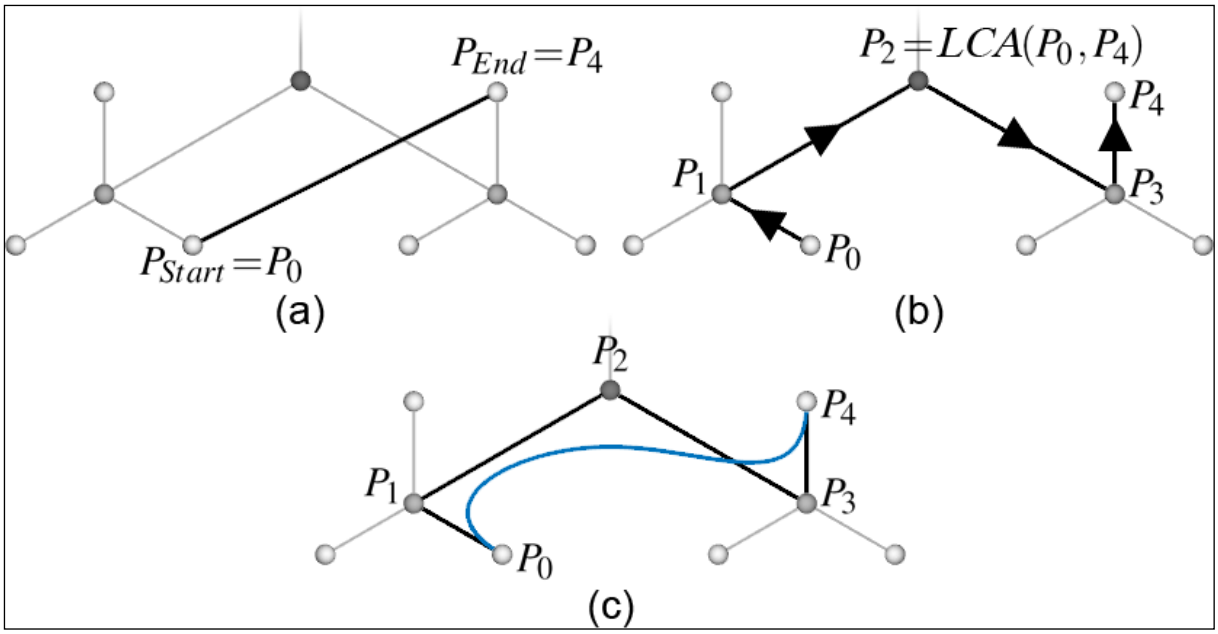

Abbildung 11: Kanten-Routingverfahren von [8]. (a) Kante ohne Routing, (b) Pfad entlang der Hierarchie in der Baumstruktur, (c) Kante als Spline unter Nutzung des Pfades aus (b) als Kontrollpolygon. Quelle: [8].

Das Verfahren bietet vor allem zwei große Vorteile. Erstens ist die Bündelung ähnlich wie bei der im vorherigen Abschnitt vorgestellten Lösung konfigurierbar. Je stärker der Einfluss des Kontrollpolynoms auf den Verlauf der Kante, desto stärker die resultierende Bündelung. Darüber hinaus werden mit diesem Verfahren Kanten gebündelt, die auch semantisch verwandt sind. Bündel entstehen hier um die Kanten der Kontrollpolygone, also um die Pfade zwischen den Hierarchieebenen innerhalb der Baumstruktur. Je näher die Start- und Endpunkte mehrerer Kanten beieinander liegen, desto weiter ist ihr gemeinsamer Weg im Baum und desto länger hält somit ihre Bündelung. Diese semantische Form der Bündelung kann als größter Vorteil angesehen werden, da sie Zusammenhänge besonders gut abbildet.

Gleichzeitig kann sie aber auch einen Nachteil darstellen, da sie die grafische Existenz aller Knoten innerhalb der Baumstruktur voraussetzt. Sobald ein nächster gemeinsamer Vorfahre nicht dargestellt wird, entfällt er als Stützpunkt des Kontrollpolygons, womit das Routing in diesem Bereich ungelöst wäre. Übertragen auf Softwarelandschaften stellt dies das größte Problem dar. Zwar kann innerhalb der Städte das Straßennetz genutzt werden, da dieses die Hierarchie abbildet. Damit würde sogar ein Bild sehr nahe an der Stadt-Metapher erreicht werden, wenn man direkt auf den Straßen entlang routet, indem man auf die Nutzung von Splines verzichtet. Außerhalb von Städten jedoch fehlen die Stützpunkte der Baumhierarchie, womit zumindest in diesem Bereich durch das Verfahren keine Lösung gefunden wer-

den kann. Außerdem würde beim Routing auf den Straßen entlang der Vorteil der variablen Bündelung entfallen, sodass keine Einzelkanten mehr erkennbar wären. Damit kann das an sich sehr vielversprechende Verfahren für den in dieser Arbeit behandelten Anwendungsfall keine Lösung liefern.

3. Eigene Umsetzung

Ziel dieses Kapitels ist es, unter Beachtung der aus den existierenden Lösungen im vorherigen Kapitel gewonnenen Erkenntnisse, ein eigenes Verfahren zu entwickeln. Dazu werde ich im ersten Abschnitt Zielkriterien definieren, die ein solches Verfahren erfüllen sollte, und hinsichtlich ihrer Wichtigkeit bewerten. Daraufhin werde ich im zweiten Abschnitt in Vorbereitung des eigentlichen Verfahrens und in Hinblick auf die Forderung geringer Rechenzeit eine naive Lösung als Referenz entwickeln und implementieren, um festzustellen, was eine solche einfache Lösung leisten kann und was nicht. Im dritten Abschnitt werde ich schließlich basierend auf den definierten Kriterien und gewonnenen Erfahrungen ein eigenes Verfahren konzipieren und umsetzen.

3.1 Zielkriterien

Ziel dieses Abschnittes ist es, ausgehend von den Überlegungen aus der Literatur, Kriterien zu definieren, die ein Kantenrouting-Verfahren im konkreten Anwendungsfall der Softwarelandschaft erfüllen sollte. Die Kriterien sollen dabei so konkret wie möglich formuliert werden, um später in der Evaluierungsphase die Messbarkeit zu begünstigen. Außerdem soll eine grobe Gewichtung der Kriterien vorgenommen werden.

Nimmt man die Zielstellung dieser Arbeit als Ausgangspunkt, so lassen sich zunächst zwei grobe Kriterien nennen: Geringe Rechenzeit und optische Qualität (vgl. Kapitel 1.2). Während die Rechenzeit jedoch als gutes Kriterium angesehen werden kann, da sie direkt messbar ist, lässt sich die optische Qualität nicht ohne weiteres definieren. In der Literatur wird in dem Zusammenhang gern von Lesbarkeit gesprochen (z.B. [4]), was zunächst auch einleuchtet. Grafische Darstellungen sollen ja gerade die schnelle und fehlerfreie Erfassbarkeit von Informationen vereinfachen, was man auch als Verbesserung der Lesbarkeit bezeichnen könnte.

Doch auch die Lesbarkeit stellt keine direkt messbare Größe dar. Bestenfalls könnte man versuchen, diese statistisch zu erfassen, indem man die Schnelligkeit und Fehlerfreiheit der Erfassung von Informationen durch Menschen an Beispielen misst – eine Herangehensweise, die beispielsweise [7] verfolgen. Abgesehen davon, dass diese Art der Messung sehr aufwändig wäre, bleibt die Lesbarkeit als Kriterium für die zielgerichtete Entwicklung eines Verfahrens in dieser Form jedoch ungeeignet, da sie noch nicht konkret genug vorstellbar ist, um eine Optimierung auf ihrer Grundlage zuzulassen.

Auch wenn die Lesbarkeit als Ziel von zentraler Bedeutung ist, sollte sie also durch klarer definierbare Kriterien beschrieben werden. [13] schlagen hierfür folgende Kriterien vor: Vermeidung von Kantenkreuzungen, räumliche Kompaktheit, Vermeidung von Kantenknicken, kurze Kanten, gleichmäßige Kantenlänge und gleichmäßige Knotendichte. Aus der Zielstellung dieser Arbeit lässt sich außerdem die Vermeidung von Stadtdurchschneidungen ableiten. Im Folgenden werde ich diese und weitere selbst definierte, vor allem aus den Stärken der im vorherigen Kapitel vorgestellten Verfahren abgeleitete, Kriterien betreffend ihrer Anwendbarkeit und Wichtigkeit für den Anwendungsfall dieser Arbeit bewerten.

Kriterien hoher Wichtigkeit (Klasse 1)

Diese Kriterien betrachte ich als besonders wichtig:

K1.1 Geringe Rechenzeit: Eine möglichst geringe Rechenzeit ist wohl das trivialste Kriterium, da es die größte Schwäche des bestehenden kräftebasierten Verfahrens darstellt und im Sinne einer guten interaktiven Nutzbarkeit unerlässlich ist. Gleichzeitig ist zu erwarten, dass es das Kriterium ist, welches die meisten Konflikte mit anderen Kriterien aufweist. Bei Designentscheidungen muss also immer abgewogen werden, ob ein Optimierungsansatz für ein Qualitätskriterium die dafür aufzuwendende zusätzliche Rechenzeit rechtfertigt. Im Zweifelsfall sollte der betreffende Optimierungsschritt per Parameter zur Laufzeit frei zuschaltbar sein.

K1.2 Vermeidung von Stadtdurchschneidungen: Das Vermeiden von Stadtdurchschneidungen wird bereits in der Zielstellung der Arbeit direkt gefordert. Wie in Abbildung 3 in Kapitel 1.1 gut ersichtlich, wird die Lesbarkeit durch Stadtdurchschneidungen stark beeinträchtigt. Selbst in der recht kleinen dargestellten Softwarelandschaft sind vor allem im linken Bereich, in dem vier Städte etwa auf einer gemeinsamen vertikalen Achse liegen, Start- und Endpunkte der Kanten kaum mehr erkennbar. Es kann kaum beurteilt werden, in welcher Stadt eine Kante, die in der obersten Stadt beginnt, endet. Auch globale Aussagen über den Kopplungsgrad der Städte sind schwierig, da sich die globalen Kantenbündel überlagern. Es ist leicht vorstellbar, dass dieser Effekt in größeren Landschaften stark zunehmen wird und soweit gehen kann, dass insbesondere kleinere Städte durch Kantencluster zwischen größeren Städten vollständig überlagert werden und somit nicht mehr zu erkennen sind.

K1.3 Erkennbarkeit von High-Level-Zusammenhängen: Wie bereits in Kapitel 2 im Rahmen der Diskussion bestehender Verfahren zu erkennen war, unterscheiden sich die verschiedenen Ansätze vor allem auch darin, wie gut High-Level- und Low-Level-Zusammenhänge zu erkennen sind. Dabei fällt auch auf, dass ein Verfahren, das eins der beiden Kriterien besonders gut erfüllt, das andere in der Regel besonders schlecht erfüllt. Beide Kriterien behindern sich gegenseitig, was an sich auch nicht sonderlich überraschend ist. Egal in welcher Darstellungsform: Je mehr Einzelinformationen dargestellt werden, desto weniger gut erkennbar sind globale Zusammenhänge. Daraus wird ersichtlich, dass im Wechselspiel beider Kriterien eine Gewichtung stattfinden muss. Es stellt sich also die Frage, ob es wichtiger ist, Abhängigkeiten zwischen einzelnen Gebäuden/Klassen zu erkennen oder globale Zusammenhänge zwischen Städten/Teilsystemen. Obwohl letztlich beide Informationen relevant sein können, komme ich zu der Überzeugung, dass im Falle der Darstellung in Softwarelandschaften die globalen Zusammenhänge wichtiger sind. Meine Meinung begründet sich dabei vor allem auf die Tatsache, dass Visualisierungen von Software vor allem einen schnellen Überblick über deren Grobstruktur geben sollen, da diese nicht direkt aus dem Quelltext abgeleitet werden kann. Detailinformationen lassen sich hingegen viel einfacher aus dem Quelltext entnehmen.

K1.4 Semantisches Routing: In Kapitel 2 wurde die Bündelung semantisch verwandter Kanten als größter Vorteil des Hierarchical Edge Bundling - Verfahrens von [8] gesehen. Mit diesem Einfluss semantischer Zusammenhänge auf das Kantenrouting nimmt das Verfahren eine Sonderstellung unter allen vorgestellten Lösungen ein. Da es in der Softwarevisualisierung eben genau um die Erkennbarkeit semantischer Zusammenhänge geht, stellt der expli-

zite Einfluss semantischer Informationen auf das Routing eine besonders wünschenswerte Eigenschaft dar. Auch [13] weisen neben den weiter oben erwähnten ästhetischen Kriterien der Lesbarkeit bereits auf diesen Zusammenhang hin.

Kriterien mittlerer Wichtigkeit (Klasse 2)

Diese Kriterien betrachte ich als ebenfalls wünschenswert:

K2.1 Erkennbarkeit von Low-Level-Zusammenhängen: Auch wenn ich die High-Level-Zusammenhänge wie oben dargestellt als wichtiger erachte, bleibt die Erkennbarkeit von Low-Level-Zusammenhängen dennoch wünschenswert. Ideal wäre ein Verfahren, bei dem der Fokus auf High- oder Low-Level-Zusammenhänge frei einstellbar ist, ähnlich wie es bei den in Kapitel 2 vorgestellten Kontrollnetzlösungen von [12] und [4], sowie beim Hierarchical Edge Bundling von [8] der Fall ist.

K2.2 Vermeidung von Kantenkrümmungen: Die Vermeidung von Kantenkrümmungen leitet sich aus dem von [13] vorgeschlagenen Kriterium der Vermeidung von Kantenknicken ab. Die einzelnen Nachteile von übermäßig vielen Kantenkrümmungen wurden bereits in Kapitel 2 als Nachteile der Kontrollnetzlösungen von [12] und [4] hinreichend diskutiert.

K2.3 Flache Kantenkrümmungen: Je stärker einzelne Krümmungen ausfallen, desto mehr wächst vor allem die Gefahr, den Überblick über die globale Richtung einer Kante zu verlieren, ähnlich wie bei übermäßig vielen Krümmungen. Deshalb sollten Krümmungen, wenn sie nötig sind, möglichst flach ausfallen.

Kriterien niedriger Wichtigkeit (Klasse 3)

Diese Kriterien stammen von [13] und sollten nicht unbeachtet bleiben:

K3.1 Räumliche Kompaktheit: Solange keine Kriterien der ersten beiden Wichtigkeitsklassen dadurch verletzt werden, sollte ein Routingverfahren eine räumlich möglichst kompakte Lösung liefern. Unnötige Platzverschwendung, etwa durch unnötig weite Kantenumlenkung sollte also vermieden werden.

K3.2 Vermeidung von Kantenkreuzungen: In der klassischen Graphentheorie wird dieses Kriterium als besonders wichtig angesehen. Graphen sollen im Idealfall planar sein, was natürlich sehr wünschenswert wäre. In Hinblick auf den Anwendungsfall dieser Arbeit halte ich das Kriterium allerdings für weniger wichtig, da Planarität aufgrund der großen Anzahl an Kanten im Regelfall nicht erreichbar sein dürfte. Obwohl das Kriterium in seiner Grundform nicht sonderlich relevant ist, kann als wünschenswert angesehen werden, dass zumindest zwischen globalen Kantenclustern Überschneidungen vermieden werden, soweit das möglich ist.

K3.3 Möglichst kurze Kanten: Im Sinne der Verfolgbarkeit des Kantenverlaufs ist es hilfreich, wenn Kanten möglichst kurz sind. Eine übermäßige Verlängerung gegenüber dem kürzesten möglichen Verlauf einer Kante sollte also vermieden werden. Stehen in Hinblick auf alle anderen Kriterien mehrere gleichwertige Verläufe zur Auswahl, sollte folglich immer der kürzeste gewählt werden.

Unwichtige Kriterien (Klasse 4)

Auch diese Kriterien stammen von [13], sind meines Erachtens für den hiesigen Anwendungsfall aber nicht relevant:

K4.1 Vermeidung von Kantenknicken: Optisch sichtbare Knicke in Kanten sollten natürlich vermieden werden. Diese Forderung wird allerdings durch die Kriterien K2.2 und K2.3 bereits mit abgedeckt, da Knicke nichts anderes als besonders starke Krümmungen sind. Hinzu kommt, dass die oben geforderten flachen Krümmungen gerade durch besonders viele sehr kleine Knicke erreicht werden können. In dem Fall stellen zusätzliche Knicke also sogar eine Qualitätssteigerung dar.

K4.2 Gleichmäßige Kantenlänge: Die Forderung nach gleichmäßiger Kantenlänge leuchtet zunächst ein. Besonders, da die Kantenlänge an sich im Fall der Softwarelandschaften keine semantische Information beinhaltet, ist die gleichmäßige Kantenlänge zur Vermeidung von Fehlinterpretationen vorteilhaft. Allerdings wird diese Kantenlänge zunächst einmal hauptsächlich durch die Lage der Knoten beeinflusst – siehe dazu die Argumentation zum nächsten Kriterium. Darüber hinaus hat nur noch der Verlauf der Kanten Einfluss auf die Erfüllbarkeit dieses Kriteriums. Um aber eine Angleichung der Länge der einzelnen Kanten aneinander auf diesem Wege zu erreichen, müssten eine ganze Reihe anderer Kriterien negativ beeinflusst werden, insbesondere K2.2, K2.3, K3.1 und K3.3. Deshalb wird auf die Optimierung in Hinblick auf dieses Kriterium verzichtet.

K4.3 Gleichmäßige Knotendichte: Die gleichmäßige Verteilung von Knoten ist an sich wünschenswert, da so keine besonders dichten Regionen entstehen würden, die besonders zur Unübersichtlichkeit neigen. Da im Falle der Softwarelandschaften die Nähe von Städten zueinander aber semantische Bedeutung haben kann, indem beispielsweise besonders stark voneinander abhängige Teilsysteme in relativer Nähe zueinander positioniert werden, scheidet die Umordnung von Knoten als Maßnahme aus.

3.2 Einfaches geometrisches Routing

Hauptziel der Entwicklung dieses Verfahrens ist es, zunächst die Kernforderungen der Zielstellung dieser Arbeit zu erfüllen – das heißt vor allem Stadtdurchschneidungen zu vermeiden und dabei so wenig Rechenzeit wie möglich in Anspruch zu nehmen – um durch eigene Erfahrung ein tieferes Verständnis für die Problematik zu erlangen und eine naive Referenzlösung zu erhalten, die zeigt, was eine solche einfache Lösung leisten kann und was nicht.

Geometrische Interpretation der beteiligten Objekte

Da es das wesentlichste Ziel dieses Ansatzes ist, Stadtdurchschneidungen zu vermeiden, muss zunächst festgelegt werden, wie die beteiligten Objekte geometrisch zu interpretieren sind. Dabei muss immer die zu erwartende rechentechnische Komplexität mit betrachtet werden.

Für Kanten ergeben sich zunächst zwei Möglichkeiten: Beschreibung als Punktmenge oder als Funktion. Dabei ist sofort ersichtlich, dass die Beschreibung als Funktion bezogen auf die rechentechnische Komplexität die günstigere Variante ist, da Schnitt- bzw. Lageberechnungen für Funktionen einfacher sind als für Punktmengen. Bleibt die Frage, ob die Beschreibung als Funktion anwendbar ist. Für den einfachen Fall der direkten Verbindung von Start- und Endknoten mittels Gerade ist die Frage leicht zu beantworten. Es genügt eine Geradengleichung, unter Beachtung von Start- und Endpunkt der Teilstrecke der Geraden, die die Kante einnimmt. Eine Kante wird also zunächst als Geradenabschnitt betrachtet. Damit ist es naheliegend, diesen Fall als Ausgangsposition zu nutzen. Wenn die Kante umgelenkt werden muss, weil eine Stadtdurchschneidung festgestellt wird, kann genauso verfahren werden, indem die Kante in mehrere Geradenabschnitte zerlegt wird.

Eine Kante wird also geometrisch als eine geordnete Menge von Geradenabschnitten interpretiert, wobei ein Geradenabschnitt gekennzeichnet ist durch die Gleichung der zugehörigen Gerade sowie dem Start- und Endpunkt des Abschnitts. Da zur Definition einer Geraden zwei Punkte genügen, ist es trivialerweise ausreichend, nur Start- und Endpunkt zu kennen. Dabei ist der Startpunkt eines Geradenabschnitts entweder der Startpunkt der gesamten Kante oder der Endpunkt genau eines anderen Geradenabschnitts, seines Vorgängers. Äquivalent gilt für den Endpunkt eines Geradenabschnitts, dass er entweder Endpunkt der gesamten Kante ist oder Startpunkt genau eines anderen Geradenabschnitts, seines Nachfolgers.

Für Städte, als die komplexeren Objekte, ergeben sich mehrere Möglichkeiten. Erstens könnte man Städte als Mengen der Einzelobjekte betrachten, die ihnen zugeordnet sind, also als Menge von Gebäuden und Straßen. Da ein Verfahren für die zweidimensionale Darstellung entwickelt werden soll, würde es sich folglich um eine Menge von Polygonen handeln. Für jedes Paar (städtisches Einzelobjekt, Geradenabschnitt einer Kante) müsste damit eine Schnittberechnung durchgeführt werden, was rechentechnisch aufwändig und somit nicht praktikabel wäre. Außerdem könnte bei einer Umlenkung einer Kante um ein Einzelobjekt eine Verschiebung in ein benachbartes Einzelobjekt erfolgen, in Anbetracht der dichten Verteilung der Objekte in einer Stadt sogar sehr häufig. Diese Herangehensweise ist also ungeeignet.

Als zweite Variante könnten die Polygone der Einzelobjekte, da sie direkt benachbart sind, zu einem Polygon zusammengefasst werden. Die Anzahl der Schnittberechnungen würde sich reduzieren und eine Verschiebung in benachbarte Objekte würde seltener auftreten. Je nach Struktur der Stadt und Verlauf der Kante sind dennoch Situationen denkbar, in denen unnötig viele Schnittberechnungen nötig wären. Beispielsweise wäre eine Stadt denkbar, die aus einer Hauptstraße besteht, mit vielen direkten Nebenstraßen. Das Polygon, das durch einfache Zusammenfassung der Polygone der Einzelobjekte entstünde, würde einem Kamm ähneln. Verliefe nun ein Geradenabschnitt als Teil einer Kante parallel zur Hauptstraße, würde es zu Durchschneidungen mit allen Nebenstraßen kommen. Es wäre eine Umlenkung um jede Nebenstraße notwendig.

Erheblich weiter reduzieren ließe sich der so zu erwartende Berechnungsaufwand durch Einschließen aller Einzelelemente einer Stadt in eine konvexe Hülle. Das im vorherigen Abschnitt dargestellte Problem des Durchschneidens eines Geradenabschnitts einer Kante mit

vielen Polygonabschnitten entfiele in diesem Fall. Ein Geradenabschnitt hätte nur noch bis zu zwei Schnittpunkte mit dieser geometrischen Interpretation der Stadt. Damit wäre die Zahl der maximal notwendigen Umlenkungen deutlich beherrschbarer. Allerdings sind immer noch Schnittberechnungen mit allen Polygonabschnitten notwendig.

Um dieses Problem zu lösen und damit den Berechnungsaufwand zu minimieren, komme ich zu dem Schluss, Städte geometrisch als deren Umkreise zu interpretieren. Es genügt damit, pro Stadt und Geradenabschnitt, eine einzige Schnittberechnung zwischen Gerade und Kreis.

Gestaltung der Kantenumlenkung

Nach der Festlegung der geometrischen Interpretation der Objekte ist nun die Gestaltung der Kantenumlenkung im Falle von Stadtdurchschneidungen zu diskutieren. Nach der im vorherigen Abschnitt gegebenen Definition der geometrischen Interpretation einer Kante als geordnete Menge von Geradenabschnitten, ist klar, dass die Umlenkung einer Kante, bzw. eines Geradenabschnitts, auf der Zerlegung dieses Abschnitts in mehrere Teilabschnitte beruht. Noch nicht geklärt ist allerdings, in wie viele Abschnitte zerlegt werden sollte. In Hinblick auf die rechnerische Komplexität sollten es nicht allzu viele Abschnitte sein, da für jeden neuen Abschnitt wieder Schnittberechnungen mit allen anderen Kreisen durchgeführt werden müssten. Besonders interessant ist daher die Zerlegung in zwei (Abbildung 12) oder drei (Abbildung 13) Abschnitte.

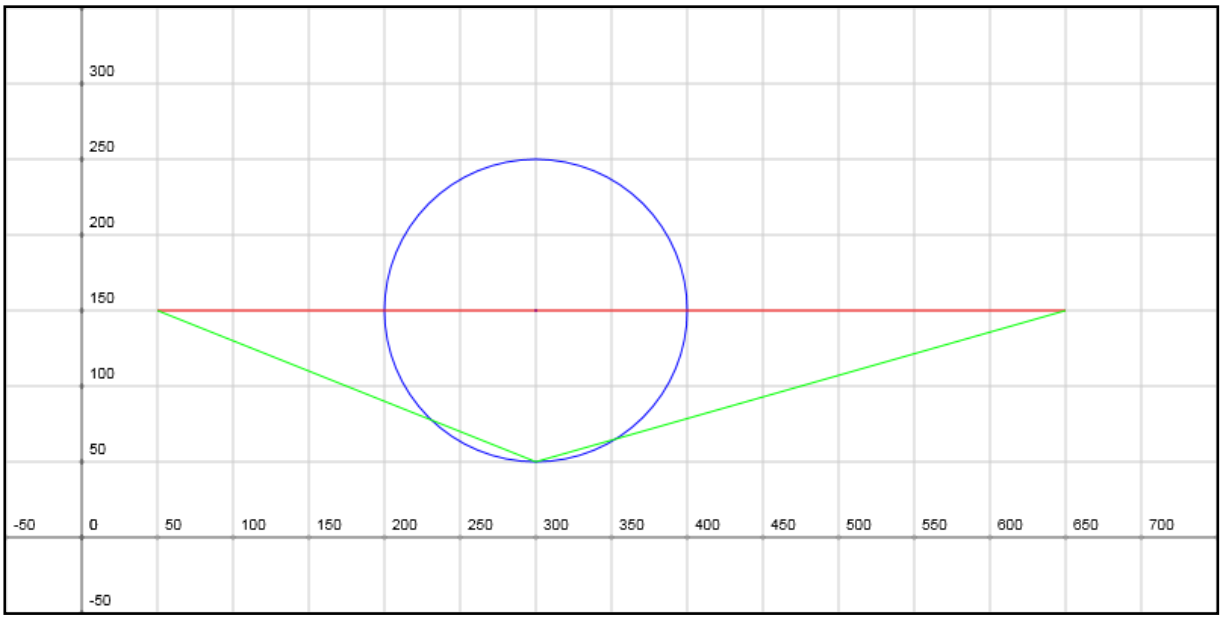

Abbildung 12: Kantenumlenkung beim einfachen geometrischen Routing mit Zerlegung in zwei Teilstrecken. Blau: Stadtkreis, Rot: Originalkante, Grün: umgelenkte Kante.
Quelle: Eigene Darstellung.

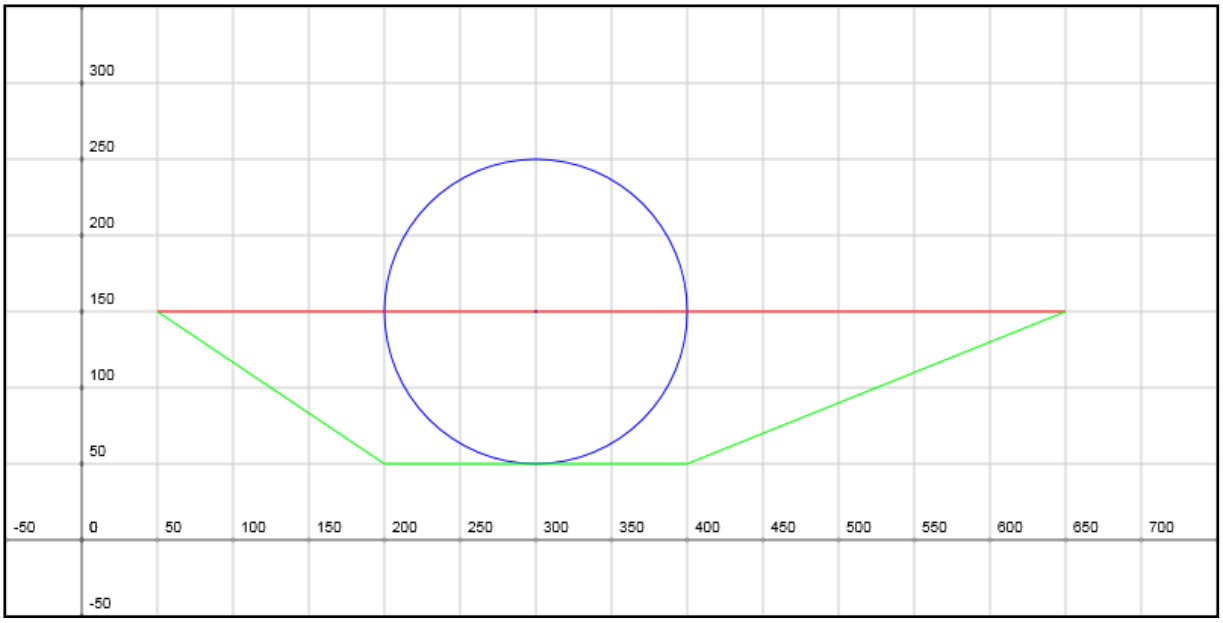

Abbildung 13: Kantenumlenkung beim einfachen geometrischen Routing mit Zerlegung in drei Teilstrecken. Blau: Stadtkreis, Rot: Originalkante, Grün: umgelenkte Kante.
Quelle: Eigene Darstellung.

In Abbildung 12 und Abbildung 13 ist jeweils die Durchschneidung einer Stadt (blauer Kreis) und einer Originalkante (rote Strecke), sowie die umgelenkte Kante (grüne Strecken) dargestellt. Im ersten Fall wurde die Kante in zwei Teilkanten zerlegt, im zweiten Fall in drei. Es ist ersichtlich, dass die Zerlegung in zwei Teilkanten auf diese Weise noch nicht zum Erfolg führt. Es wären weitere Umlenkungen notwendig, um Überschneidungsfreiheit herzustellen, was sich in zusätzlichem Rechenaufwand niederschlagen würde und somit nicht wünschenswert wäre. Alternativ wäre es möglich, den Berührungspunkt der beiden Teilkanten nicht auf den Kreis zu setzen, sondern soweit hinauszuschieben, dass keine Überschneidungen mehr auftreten. Dies könnte allerdings zu sehr weiten Verschiebungen führen, insbesondere dann, wenn der durchschnittene Kreis sehr groß wäre oder Start- bzw. Endpunkt der Kante sehr nahe am Kreis lägen.

Im zweiten Fall, dargestellt in Abbildung 13, wurde die Kante in drei Teilkanten zerlegt, indem die Schnittpunkte mit dem Kreis selbst nach außen verschoben wurden. Es wurde eine Parallelverschiebung des Kantenabschnitts zwischen den beiden Schnittpunkten durchgeführt. Es ist offensichtlich, dass diese Variante immer eine Lösung im ersten Schritt garantiert. Deshalb ziehe ich sie der ersten vor.

Neben der Schnittberechnung und vor Bestimmung der Verschiebungsstärke ist als nächstes die Richtung der Verschiebung festzulegen. Dabei gibt es aus meiner Sicht zwei Alternativen. Betrachtet man nur Einzelkanten, sollte immer in die Richtung aus dem Kreis heraus verschoben werden, in der die Verschiebung, also der maximale Abstand zur Kreislinie am geringsten ist, um möglichst dicht am Originalverlauf der Kante zu bleiben. Betrachtet man darüber hinaus jedoch alle Kanten, die einen Kreis schneiden, würden sich diese mit dieser Herangehensweise auf beide Seiten mehr oder weniger gleichmäßig verteilen. Der Kreis würde so von zwei Kantenbündeln eingeschlossen. Um dies zu vermeiden, wäre es stattdessen auch möglich, eine Verschiebungsrichtung für alle Kanten gemeinsam zu bestim-

men. Ansatzpunkte dafür wären etwa die häufiger auftretende individuelle Verschiebungsrichtung oder aber der gemeinsame Schwerpunkt. Da beide Varianten ihre Vor- und Nachteile haben, die erste aber rechentechnisch weniger aufwändig ist, bevorzuge ich diese.

Vorläufiges Ergebnis

Mit dem bisher beschriebenen Verfahren ergibt sich für das Beispiel aus Abbildung 3 das in Abbildung 14 dargestellte Ergebnis.

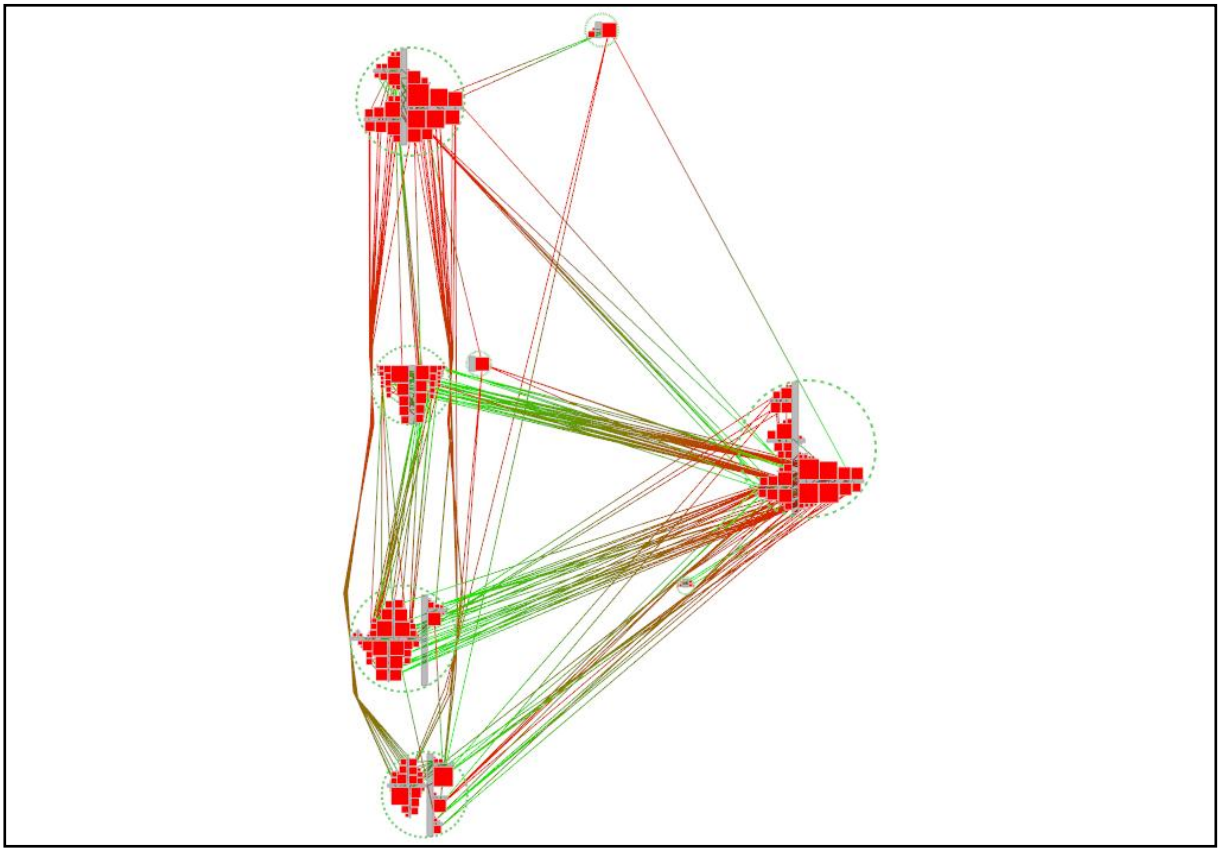

Abbildung 14: Die Softwarelandschaft aus Abbildung 3 mit einfachem geometrischen Routing ohne optische Optimierung.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Die beiden Kernforderungen der Zielstellung werden erfüllt: Es sind keine Stadtdurchschneidungen zu erkennen und das Verfahren ist, gemessen am in Kapitel 1.1 vorgestellten kräftebasierten Verfahren, deutlich schneller. Für das dargestellte Beispiel lag die Rechenzeit bei zwei hundertstel Sekunden, beim kräftebasierten Verfahren je nach Konfiguration im zweistelligen Sekunden- bis Minutenbereich.

Allerdings sind auch direkt einige Schwächen zu erkennen. Besonders auffällig sind die entstehenden Knicke innerhalb der Kanten. Obwohl sie relativ klein sind, bezogen auf den Knickwinkel, sind sie dennoch störend. Gerade in Hinblick auf größere Systeme mit stärkeren Überlagerungen der Kanten, kann an solchen Stellen die Erkennbarkeit der Verlaufsrichtung beeinträchtigt werden. Zumindest dieses offensichtliche Manko sollte behoben werden.

Optische Optimierung

Durch Verwendung von Bézier-Splines lässt sich das in Abbildung 14 dargestellte Beispiel in Bezug auf die Erkennbarkeit der Verlaufsrichtung der Kurven deutlich verbessern (Abbildung 15). Dabei wird für jeden Knick ein Spline erzeugt, wobei der Knickpunkt als Kontrollpunkt verwendet wird und die Mittelpunkte der anliegenden Teilkanten als Start- und Endpunkt.

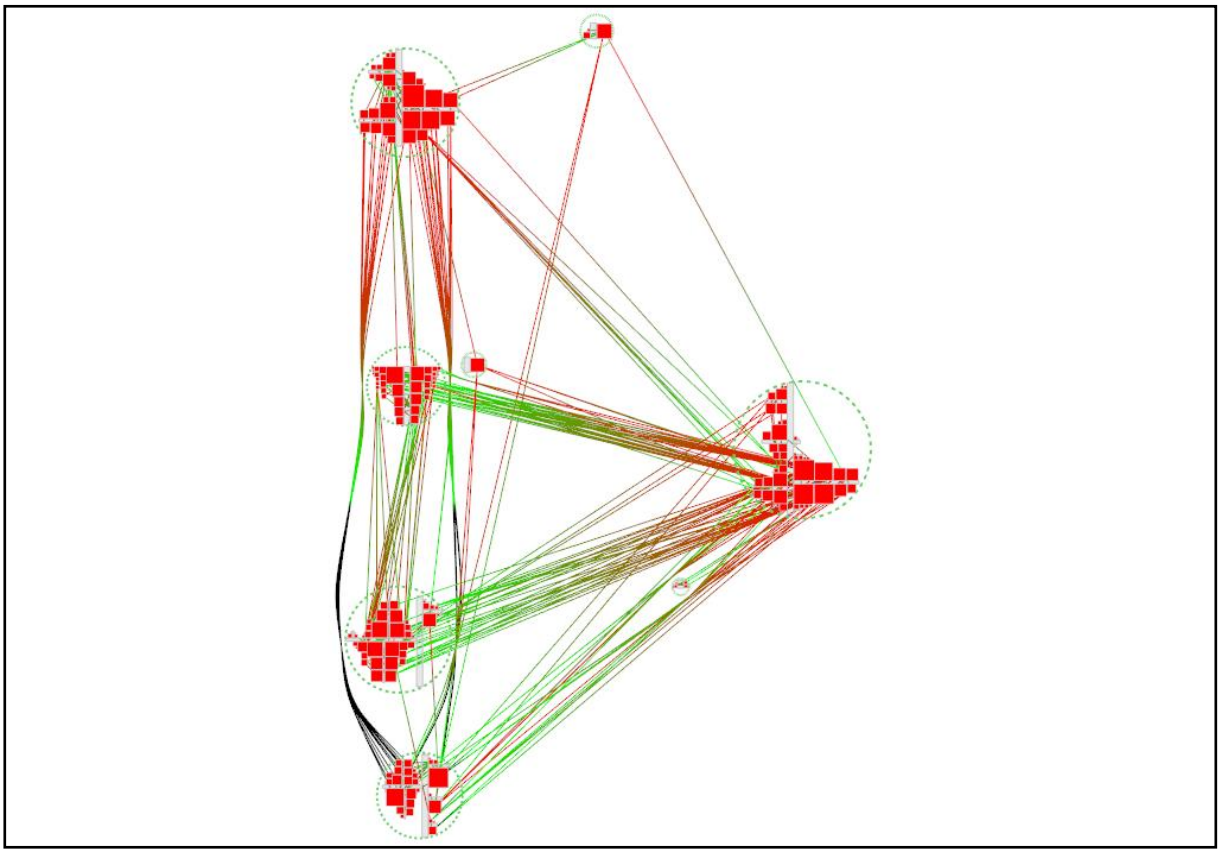

Abbildung 15: Die Softwarelandschaft aus Abbildung 14 mit optischer Optimierung mittels Bézier-Splines.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Dennoch kann das erzielte Ergebnis nicht vollständig überzeugen. Obwohl Stadtdurchschneidungen vermieden werden, treten immer noch ähnliche Probleme auf, wie in den in Kapitel 2 vorgestellten Ansätzen. Das Hauptproblem ist dabei, dass die Bündelung von Kanten nur als Nebeneffekt und somit unkontrolliert erfolgt. Damit ist auch die Erkennbarkeit von Einzelkanten bzw. von globalen Kantenrichtungen eher zufällig gegeben. Kanten, die keine Städte durchschneiden, behalten ihren Originalverlauf bei, wie im rechten Teil von Abbildung 15. In diesem Bereich sind die Einzelkanten gut erkennbar. Globale Kantenrichtungen sind jedoch nicht gut erkennbar. Durchschneiden mehrere Kanten mit ähnlichem Verlauf eine oder mehrere Städte, findet eine Bündelung statt, wie im linken Teil von Abbildung 15 zu sehen. Die Bündelung ist allerdings so stark, dass der Verlauf der einzelnen Kanten nicht mehr zu erkennen ist.

Die wesentlichste Erkenntnis aus den Ergebnissen dieses einfachen Verfahrens ist also für mich, dass die Kontrollierbarkeit der Kantenbündelung eine zentrale Rolle spielt. Deshalb soll

diese Regelbarkeit der Bündelung der Ausgangspunkt für die Entwicklung eines zweiten Verfahrens sein.

3.3 Gummiband-Routing

In diesem Abschnitt werde ich aufbauend auf allen bisher gesammelten Erkenntnissen ein eigenes Verfahren entwickeln. Dabei wird der Fokus zunächst auf der Bündelung der Kanten liegen, d.h. diese wird separat behandelt. Erst im zweiten Schritt wird das eigentliche Routing im Sinne der Wegsuche zwischen den Städten diskutiert werden.

Die ursprüngliche Idee

Die wesentlichste Erkenntnis aus dem ersten entwickelten Verfahren war, dass eine kontrollierbare Bündelung von zentraler Bedeutung für die optische Qualität des Kantenlayouts ist. Nur durch direkte Kontrolle über die Bündelung, können die Ziele der Erkennbarkeit globaler Kantenströme und der Erkennbarkeit von Einzelkanten erreicht werden, da beide Ziele sich gegenseitig behindern. Da ich beiden Zielen außerdem eine hohe Bedeutung zumesse, soll eben diese kontrollierbare Bündelung der Ausgangspunkt für die Entwicklung dieses Verfahrens sein. Genauer gesagt, halte ich die Bündelung für so bedeutsam, dass ich sie vom eigentlichen Routing im Sinne der Wegsuche zwischen den Städten trennen und als separaten Schritt betrachten möchte.

Der ursprünglichen Idee des Verfahrens liegt eine Metapher aus der realen Welt zugrunde: Der Kabelbinder. Wie würde man vorgehen, müsste man die einzelnen Elemente der Softwarestädte untereinander verkabeln. Zunächst wäre es sinnvoll, im Sinne eines geringen Materialeinsatzes und möglichst geringer Kabellänge, alle Kabel als Geraden zu verlegen. Damit entstände zunächst, von Gebäudedurchschneidungen mal abgesehen, das in Abbildung 3 dargestellte Ausgangsbild. Um dieses chaotische Wirrwarr an Kabeln besser kontrollieren zu können, würde man nun an geeigneten Stellen Kabelbinder einsetzen, um Einzelkabel zu Kabelsträngen zusammenzufassen.

Übertragen auf die Kanten zwischen Softwarewarestädten, könnte man analog an geeigneten Stellen Stützstege positionieren und Kanten durch diese Stege leiten. Es stellt sich allerdings die Frage, was geeignete Stellen sind. Hier wären verschiedene Herangehensweisen denkbar. Zum ersten könnte man zur Erzeugung von Stützstellen für Stege die Delaunay Triangulation nutzen, wie sie auch in den in Kapitel 2 vorgestellten Kontrollnetzlösungen verwendet wird. Damit würde man zunächst ein Kantennetz wie in Abbildung 9b erzeugen, indem jede Stadt als Knoten betrachtet würde. Auf diesen Kanten könnte man Stützstellen für die Stützstege mit einer definierten Auflösung gleichmäßig verteilen und an diesen Stellen schließlich Stützstege orthogonal zu den Delaunay-Kanten platzieren. Würde man diese Vorgehensweise für das Kantennetz aus Abbildung 9b anwenden, wäre klar zu erkennen, dass die so platzierten Stützstege für die Bündelung von Kanten zwischen benachbarten Knoten gut platziert wären. Für Knoten jedoch, die weiter auseinander liegen, stellt die Herangehensweise offensichtlich keine gute Lösung bereit. Kanten zwischen diesen Knoten müssten durch Stützstege verschiedener Delaunay-Kanten geführt werden. Insbesondere

bei langen Kanten bestände ähnlich wie bei den Kontrollnetzlösungen die Gefahr, dass Kurven mit unnötig vielen Krümmungen (Zick-Zack-Kurven) entständen, was der Qualität des Ergebnisses abträglich wäre.

Als Alternative zur Delaunay Triangulation liegt es nahe, das Kantennetz zur Platzierung der Stützstege als voll vermaschtes Netz bezogen auf die als Knoten betrachteten Städte zu konstruieren. Damit ständen auch für weit auseinander liegende Kanten genügend Stützstege zur Verfügung. Es würden allerdings mit einer gewissen Wahrscheinlichkeit in Abhängigkeit der Anzahl der zu platzierenden Stützstellen je Kante auch Stützstege erzeugt werden, die innerhalb von Städten lägen und somit ungeeignet wären. In diesem Fall müssten wieder Stützstege anderer Kanten genutzt werden, was nicht unbedingt zu guten Ergebnissen führen würde, da nicht garantiert werden kann, dass Stützstege an geeigneten Positionen existieren. Bei sehr groben Auflösungen der Verteilung der Stützstege im Kantennetz, wenn beispielsweise nur ein einziger Stützsteg in der Mitte jeder Kante platziert würde, wären potentiell sehr weite Umlenkungen notwendig. Bei sehr feinen Auflösungen hingegen, würden die Umlenkungen nur in unmittelbarer Nähe von Städten stattfinden, sodass Stadtdurchschneidungen entstehen könnten oder bei deren Umgehung sehr starke Krümmungen (direkt am Kreis entlang) entstehen würden (Abbildung 16).

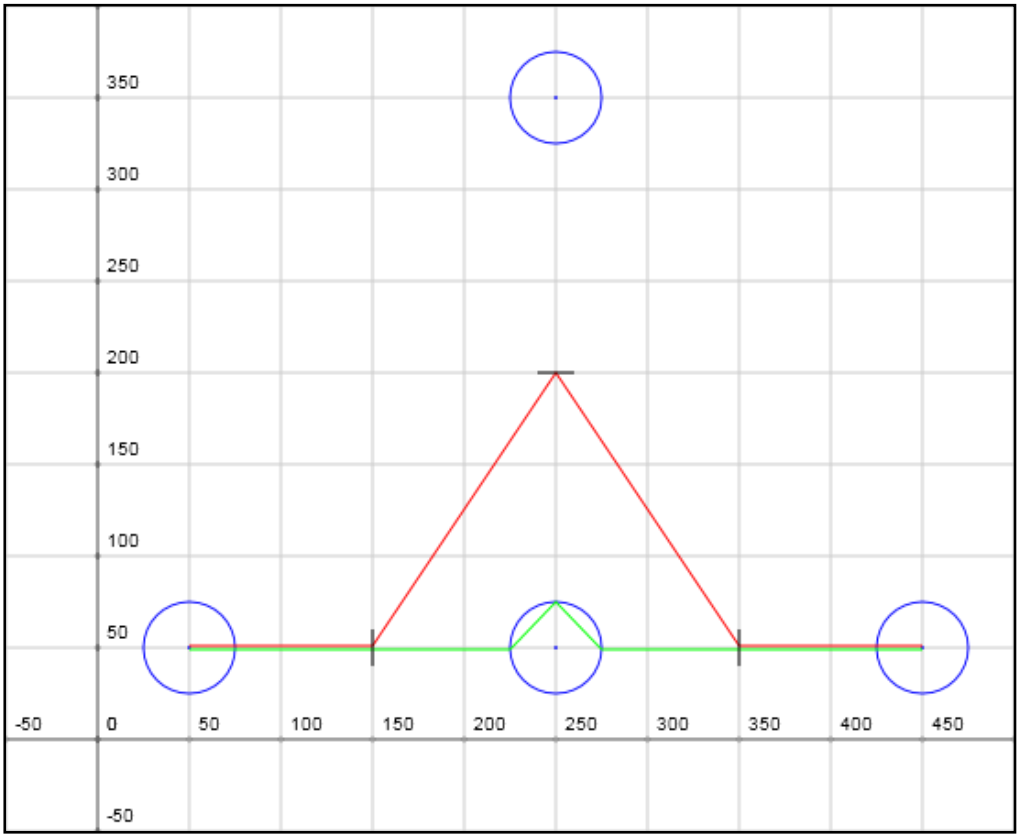

Abbildung 16: Ungünstige Positionierung der Stützstege. Eigentlich würde im mittleren Kreis ein Stützsteg für die Verbindung der beiden äußeren Kreise liegen. Da dieser ungültig wäre, muss über Stützstege des oberen Kreises umgeleitet werden. Einmal mit Verwendung weniger Stützstege (rot) und einmal mit Verwendung vieler Stützstege (grün). Die Stützstege selbst (schwarz) sind der Übersicht halber nur für den ersten Fall mit eingezeichnet, im zweiten Fall wäre alle 25 Einheiten ein Stützsteg platziert.
Quelle: Eigene Darstellung.

Als zweite Frage wäre zu beantworten, wie die Breite der Stützstege gewählt werden sollte. Es ist klar, dass diese steuerbar sein muss, um die Stärke der Bündelung variabel zu halten. Nicht klar ist allerdings, ob alle Stützstege gleich breit sein sollten. Um zum Bild der Kabelbinder zurückzukommen: Je nach Anzahl der zu bündelnden Kabel würden hier unterschiedlich starke Bündel entstehen. Dass das auch für den Anwendungsfall der Softwarestädte sinnvoll wäre, leuchtet ein, wenn man sich überlegt, dass auf diese Weise besonders starke globale Verflechtungen zwischen Städten stärker hervortreten würden. Die Breite der Bündel wäre ein Indikator für die Stärke globaler Kantenströme.

Angenommen, die Konstruktion der Stützstege wäre gelöst, bleibt die Frage, wie die einzelnen Kanten darauf platziert werden sollten. Im Sinne möglichst kurzer Kanten und einer somit zu erreichenden minimalen Ablenkung gegenüber dem geraden Verlauf, liegt es nahe, die Kanten beim Führen über die Stützstege so wenig wie möglich umzulenken. Damit würden allerdings im statistischen Mittel Häufungen der Kanten an den beiden Enden jedes Stützsteges entstehen. Das würde dazu führen, dass Einzelkanten überhaupt nicht mehr zu erkennen wären, selbst bei relativ breiten Stegen. Der Vorteil der regelbaren Stärke der Bündelung wäre zum Großteil zunichte gemacht. Günstig wäre eine gleichmäßigere Verteilung der Kanten auf dem Stützsteg, die die Originallageverhältnisse zwischen den Kanten bei geradem Verlauf widerspiegelt. Das wäre allerdings beim herkömmlichen Legen der Kanten durch geeignete fertig konstruierte Stützstege kaum zu erreichen.

Eine mögliche Lösung wäre, die Stützstege zunächst sehr breit zu wählen und festzustellen, welche Kanten welche Stützstege durchschneiden. Merkt man sich die relative Lage der Schnittpunkte auf jedem Stützsteg, kann man diese relative Positionierung auf dem Stützsteg auch nach Verkleinerung des Steges wiederherstellen. Die relative Lage der Kanten zueinander bliebe erhalten und künstliche Häufungen würden vermieden. Leider würde gleichzeitig die Gefahr von Zick-Zack-Kurven steigen, da es potentiell zu Schnitten mit vielen Stützstegen kommen könnte (Abbildung 17). Es würde ein ähnliches Bild, wie bei den in Kapitel 2 vorgestellten Kontrollnetzlösungen entstehen.

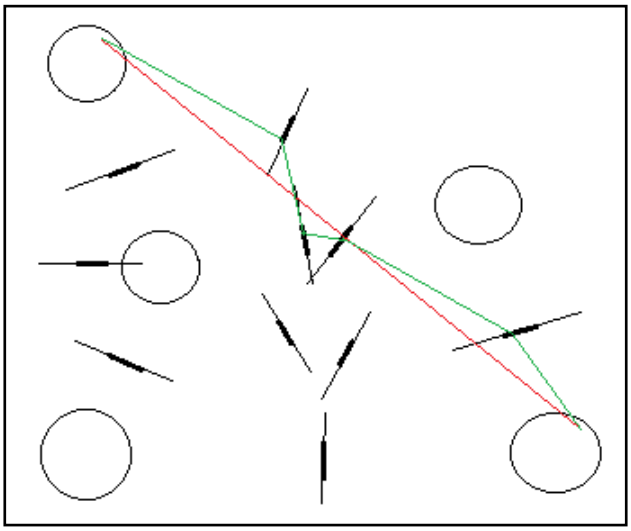

Abbildung 17: Unnötige Entstehung von Zick-Zack-Kurven. Zwischen je zwei Stadtumkreisen wurde zentral ein Stützsteg positioniert und zur Bündelung zusammengezogen. Die rote Kante würde damit offensichtlich unnötigerweise dem grünen Verlauf folgen.
Quelle: Eigene Darstellung.

Semantische Bündelung

Die Idee der bewussten Bündelung mit Stützstegen betrachte ich als guten Ansatz, insbesondere weil die Stärke der Bündelung regelbar ist und dabei die relative Lage beteiligter Kanten zueinander erhalten werden kann. Die bisher diskutierten Ansätze zur Konstruktion und Verwendung der Stege halte ich allerdings für ungeeignet, insbesondere weil sie zu Zick-Zack-Kurven führen können. Um das Problem zu lösen, führe ich eine einschränkende Bedingung für die Bündelung ein: Es werden nur Kanten gebündelt, die zum gleichen Städtepaar gehören.

Dieser Ansatz unterscheidet sich von den bisherigen grundlegend. Zunächst einmal handelt es sich um eine Einschränkung gegenüber anderen Verfahren, da bestimmte Bündelungen ausgeschlossen werden. Gäbe es in einer Softwarelandschaft beispielsweise zwei Ballungszentren mit mehrere Städten an entgegengesetzten Enden der Landschaft, zwischen denen stark ausgeprägte Beziehungen beständen, also viele Kanten verliefen, so würden zwar zwischen je zwei Städten Bündel erzeugt, diese aber nicht weiter zusammengefasst werden. Übertragen auf die Metapher des Straßennetzes, würden mehrere Schnellstraßen nebeneinander verlaufen und höchstwahrscheinlich auch einige Kreuzungen dieser globalen Ströme entstehen (Abbildung 18).

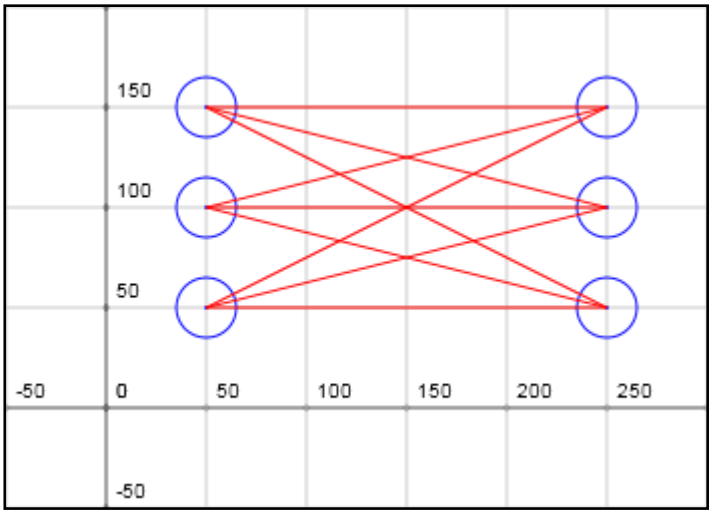

Abbildung 18: Semantische Bündelung: Die einzelnen Bündel zwischen je zwei Städten, hier dargestellt durch einfache Geraden, werden nicht gebündelt.
Quelle: Eigene Darstellung in Anlehnung an [4].

Diese Einschränkung hat aber auch einen entscheidenden Vorteil. Die Bündelung der Kanten hat nicht mehr nur den Zweck, die Lesbarkeit zu erhöhen, indem das Wirrwarr vieler Einzelkanten vermindert wird. Darüber hinaus stellen so konzipierte Kantenbündel in besonderem Maße semantische Zusammenhänge dar, da die Bündelung nicht auf Basis geometrischer oder energetischer Lagebeziehungen stattfindet, sondern aufgrund semantischer Nähe der Kanten zueinander. Nur Kanten die semantisch zur gleichen globalen Verbindung zwischen Teilsystemen (Städten) gehören, werden auch geometrisch zu einer globalen Verbindung zusammengefasst.

Darüber hinaus bietet der Ansatz weitere Vorteile. Erstens entfallen die Schnitt- oder Lageberechnungen zwischen Kanten und Stützstegen, da die Zuordnung von Kanten zu Stützstegen a priori gegeben ist, was sich positiv auf die Rechenzeit auswirkt. Zweitens entfällt an dieser Stelle die Gefahr der Entstehung von Zick-Zack-Kurven, zumindest bei geeigneter Gestaltung der Stützstege – allerdings könnten diese beim nachgelagerten Routing noch entstehen. Drittens gestaltet sich die Platzierung der Stützstege in diesem Fall recht einfach. Im Sinne einer gleichmäßigen Bündelung können diese einfach gleichmäßig auf der Verbindungslinie der Mittelpunkte der beiden beteiligten Städte verteilt werden.

Bevor ich weiter auf die Gestaltung dieses Ansatzes eingehe, sei darauf hingewiesen, dass auf diese Weise nur die Frage der Bündelung gelöst wird, nicht jedoch das Problem der Stadtdurchschneidungen. Selbst wenn durch die Bündelung auch einige Stadtdurchschneidungen in den Randbereichen von Kantenbündeln aufgehoben werden, muss hierfür eine separate Lösung gefunden werden. Auf diesen zweiten Schritt nach der Bündelung werde ich später eingehen.

Breite der Stützstege

Bei der Bestimmung der Breite der Stützstege muss zunächst unterschieden werden zwischen der initialen und der Zielbreite. Die initiale Breite ist dabei die Breite, die ein Stützsteg ursprünglich hat. In diesem Zustand soll die relative Lage der Einzelkanten zueinander bestimmt werden. Die Zielbreite ist die Breite, die der Stützsteg beim Zusammenziehen der zugehörigen Kanten zu einem Bündel bekommt.

An die initiale Breite ist zumindest eine Forderung zu stellen. Es muss garantiert werden, dass alle zugehörigen Kanten hindurch laufen. Um diese Forderung zu erfüllen, könnte man die Breite so wählen, dass die jeweils äußersten Kanten gerade noch innen liegend wären. Der Vorteil wäre, dass die Abweichung der Kanten vom Originalverlauf gering gehalten würde. Vom Berechnungsaufwand abgesehen, hätte diese Herangehensweise jedoch den Nachteil, dass einseitige oder schiefe Bündel entstehen könnten. Lägen die Kantenstart- und -zielpunkte in den beteiligten Städten jeweils nur auf einer Seite, könnten Bündel entstehen, deren primäre Verlaufsrichtung an den Städten vorbeizeigen. Damit würde die optische Verbindung der beiden Städte verloren gehen (Abbildung 19).

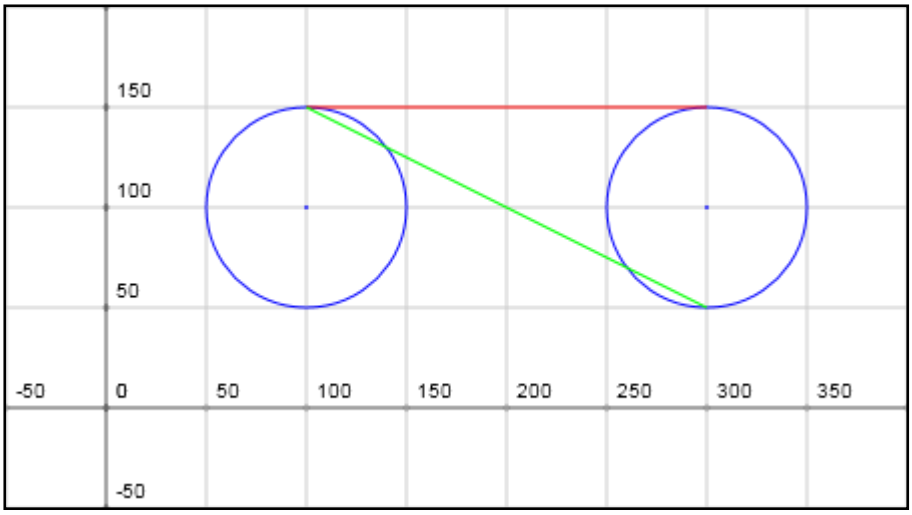

Abbildung 19: Bündelverläufe (rot und grün), die nicht auf Start- und Zielstadt zeigen. In größeren Systemen könnte so die optische Verbindung zweier Städte verloren gehen.
Quelle: Eigene Darstellung.

Die Alternative wäre, die Breite der Städte als Maßstab für die Breite der Stützstege zu verwenden. Damit wäre die Verlaufsrichtung der Bündel immer auf die Mittelpunkte der Städte ausgerichtet. Um die Sichtbarkeit der Städteverbindungen zu gewährleisten, bevorzuge ich diese Variante. Außerdem wähle ich zur Repräsentation der Städte wie im ersten Verfahren die Form des Umkreises. Neben den bereits im Abschnitt zum ersten Verfahren dargestellten Vorzügen, bietet die Darstellung als Kreis durch die inne liegende Symmetrie Vorteile bei der Berechnung, wie noch zu sehen sein wird. Die initiale Breite wird somit durch die gemeinsamen äußeren Tangenten der Umkreise der beiden beteiligten Städte determiniert (Abbildung 20).

Bei der Konstruktion der Zielbreite müssen mehrere Bedingungen beachtet werden. Erstens muss diese natürlich regelbar sein, um die Stärke der Bündelung variabel zu halten und somit den Fokus auf High- oder Low-Level-Zusammenhänge richten zu können. Zweitens sollen unnötig viele und unnötig starke Krümmungen der Kanten als Folgen der Bündelung vermieden werden. Besonders aus dem zweiten Punkt resultiert, dass eine konstante Breite des Bündels, sprich gleiche Breite aller Stützstege zwischen zwei Städten, nicht praktikabel ist. Es würden zwangsweise starke Krümmungen bzw. Knicke der Kanten entstehen, wenn sie in relativer Nähe zum Start- bzw. Endknoten zu einem Bündel konstanter Breite zusammengezogen werden würden.

Optimal wäre eine möglichst flache gleichmäßige Zu- bzw. Abnahme der Bündelung über eine längere Strecke. Deshalb strebe ich, um den Verlauf so flach wie möglich zu halten, eine Lösung an, die die maximale Bündelung nur in der Mitte zwischen Quell- und Zielstadt erreicht. Allerdings sollte die Stärke der Bündelung ausgehend von der Quellstadt, bzw. hinführend zur Zielstadt, in relativer Nähe zu den Städten bereits ein vergleichsweise hohes Niveau erreichen, um Überlagerungen mit anderen Städten frühzeitig zu reduzieren. Die einfachste und damit rechnerisch am wenigsten aufwändige Funktion, die zur Beschreibung eines Verlaufs mit diesen Eigenschaften dienen kann, ist meiner Meinung nach die Parabel. Legt man den Scheitelpunkt in die Mitte zwischen die Stadtkreise und führt die Parabeläste zu den Berührungspunkten der Tangenten an den Kreisen, erfüllt der Verlauf alle Forderun-

gen. Er ist einerseits insgesamt flach und gleichmäßig und erreicht andererseits bei Annäherung an beide Städte seine maximale Steigung (Abbildung 20).

Zusammenfassung der Idee

Gebündelt werden also nur alle Kanten die zu jeweils zwei Städten gehören (Start- und Zielstadt). Dabei ist festzustellen, dass alle Kanten innerhalb eines gemeinsamen Korridors verlaufen, der durch die beiden gemeinsamen äußeren Tangenten der Umkreise der Städte dargestellt werden kann. Der Verlauf wird durch zwischen den beiden Städten gleichmäßig verteilte Stützstege bestimmt, die zunächst diesen Korridor repräsentieren. Die Auflösung des Korridors, d.h. die Zahl der repräsentierenden Stützstege ist dabei variierbar. Die eigentliche Bündelung wird durch parabelförmiges Zusammenziehen des Korridors und damit der Stützstege erreicht, wobei der Grad der Bündelung frei definierbar ist. Die im Korridor verlaufenden Kanten werden relativ in Bezug auf ihre ursprüngliche Lage im Korridor mit zusammengezogen, d.h. ihre relative Position auf den Stützstegen bleibt konstant. Siehe hierzu auch Abbildung 20.

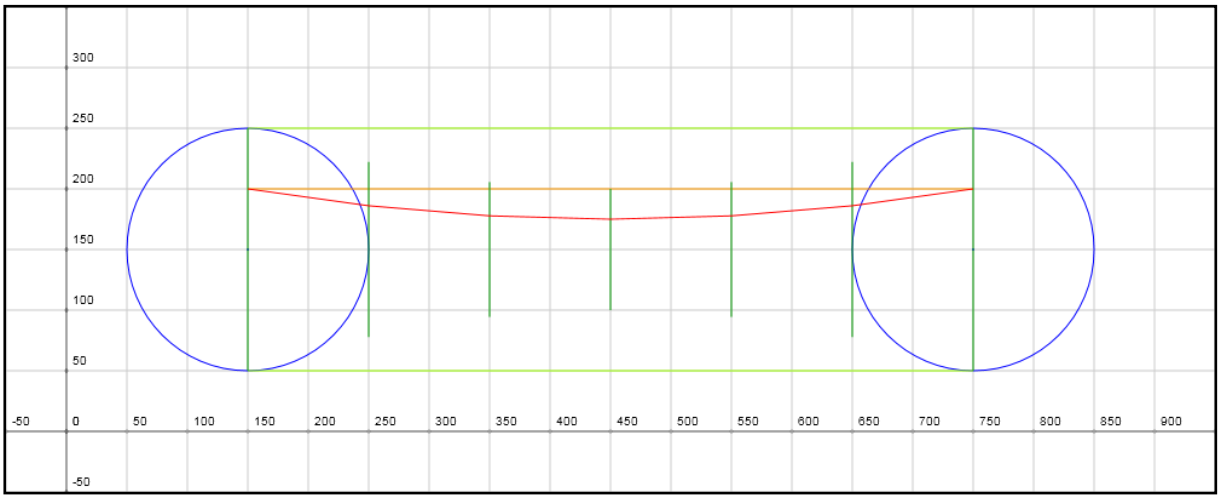

Abbildung 20: Idee der Bündelung beim Gummiband-Routing. Dargestellt sind die Umkreise zweier Städte (blau), der Korridor der beiden gemeinsamen äußeren Tangenten in dem immer alle ursprünglichen Kanten verlaufen und der somit die Ursprungsbreite des Bandes darstellt (hellgrün), das parabelförmig zusammengezogene/gebündelte Band (dunkelgrüne Stützstege), sowie eine beispielhafte Kante in Ursprungs- (orange) und gebündelter Lage (rot) bei einer Bündelung von 50%. Im allgemeinen Fall ist natürlich nicht davon auszugehen, dass die Umkreise gleich groß sind und Start- und Zielpunkt symmetrisch liegen – das wurde hier nur der Einfachheit der Darstellung halber angenommen. Quelle: Eigene Darstellung.

Implementierung

Ausgangspunkt für die Implementierung sind alle Paare aus Städten C_1 und C_2, wobei $(C_1, C_2) == (C_2, C_1)$ gilt, und alle zugehörigen Kanten, d.h. alle Kanten, deren Startknoten Nachfahre von C_1 und deren Endknoten Nachfahre von C_2 ist oder umgekehrt. Von den beiden Varianten, die jeweils das gleiche Städtepaar beschreiben, wähle ich (C_1, C_2) so, dass $C_1.Umkreisradius >= C_2.Umkreisradius$. Dafür werden vor der Paarbildung die Umkreise aller

Städte berechnet und die Städte nach dem Umkreisradius absteigend sortiert. Diese Festlegung hat im weiteren Verlauf entscheidende Vorteile. Es müssen etwa bei den folgenden geometrischen Konstruktionen weniger Spezialfälle unterschieden werden, wenn davon ausgegangen werden kann, dass der erste Umkreis des Paares der größere ist. Viel wichtiger ist diese Sortierung jedoch später für das Routing, wie noch zu sehen sein wird.

Nach diesen vorbereitenden Maßnahmen muss als nächstes für jedes Stadtkreispaar und seine zugehörigen Kanten ein Zustand hergestellt werden, der möglichst einfache geometrische Konstruktionen ermöglicht. Dazu soll eine geeignete Koordinatensystemtransformation gefunden werden, die folgenden Bedingungen genügt:

$$M_1 = (0, 0)$$

$$M_2 = \left(\left|\overrightarrow{M_1 M_2}\right|, 0\right)$$

Dabei ist M_1 der Mittelpunkt des größeren Umkreises und M_2 der Mittelpunkt des kleineren Umkreises. Diese Transformation der Lage beider Kreise auf eine Koordinatenachse erleichtert nicht nur die Konstruktion der Kreistangenten und der Polynome. Aufgrund der so erreichten Achsensymmetrie der Kreise genügt es jeweils, die Berechnungen nur für eine Seite des Bandes auszuführen, da der Verlauf der Bandgrenze auf der anderen Seite dem Betrag nach gleich ist.

Mit den beiden formulierten Bedingungen ließe sich eine einfache 2x2-Transformationsmatrix berechnen, die im Folgenden zur Transformation aller zugehörigen Kanten verwendet werden könnte. Leider ergibt sich aufgrund der Beschaffenheit der formulierten Bedingungen eine Matrix der Form:

$$\begin{pmatrix} a & b \\ 0 & 0 \end{pmatrix}$$

Da die beiden Spaltenvektoren offensichtlich linear abhängig sind, wäre diese Matrix nicht invertierbar und damit die Transformation nicht umkehrbar. Somit ist diese einfachste Form der Transformation unbrauchbar. Um die Transformationsberechnung dennoch rechnerisch so wenig aufwändig wie möglich zu halten, könnte man diese direkte Transformation per Matrix so anpassen, dass die 0 der zweiten Komponente der Zielkoordinaten der Kreismittelpunkte durch eine 1 ersetzt würde. Damit würde sich keine Nullzeile in der Matrix bilden, womit sie invertierbar und die Transformation somit umkehrbar wäre. Zusätzlich zur Transformation per Matrix wäre dann noch eine Verschiebung um den Vektor *(0, -1)* notwendig.

Leider erweist sich auch diese Lösung als nicht praktikabel, da die Transformation nicht kongruent wäre. Form und Größe der Kreise blieben nicht erhalten, sodass die folgenden geometrischen Konstruktionen nicht möglich wären. Es muss also eine kongruente Möglichkeit der Koordinatentransformation gefunden werden. Dazu wird zunächst eine Verschiebung angestrebt, die den Mittelpunkt des größeren Umkreises in den Koordinatenursprung projiziert. Daraufhin genügt eine Drehung um den Koordinatenursprung, so dass der Mittelpunkt des kleineren Umkreises auf eine Koordinatenachse abgebildet wird (Abbildung 21).

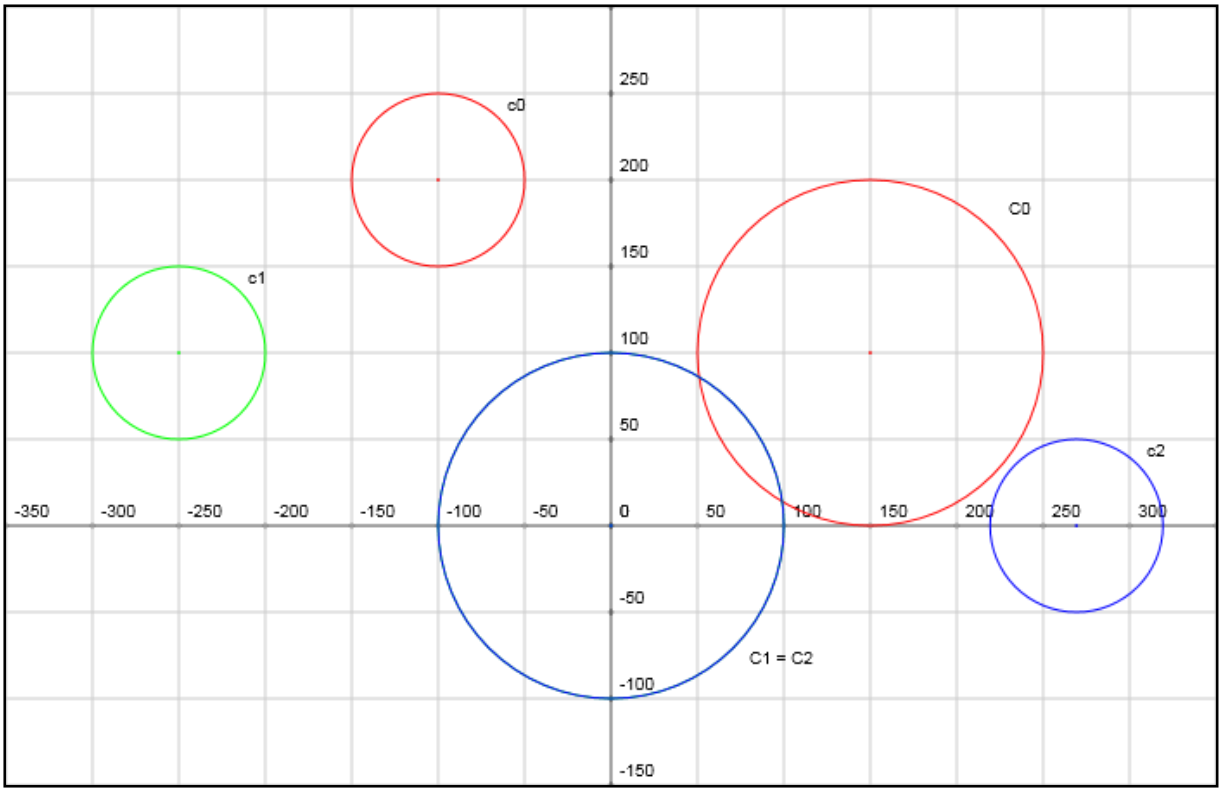

Abbildung 21: Koordinatentransformation. Die ursprünglichen Kreise C0 und c0 (rot) werden so verschoben, dass der Mittelpunkt des größeren Kreises im Koordinatenursprung liegt (C1 blau und c1 grün). Daraufhin werden beide Kreise so gedreht, dass der Mittelpunkt des kleineren Kreises auf der horizontalen Achse liegt (C2 und c2 blau).
Quelle: Eigene Darstellung.

Auf diese Weise kann die Koordinatentransformation auch für alle zugeordneten Kanten per Addition des Verschiebungsvektors und Multiplikation mit der Drehmatrix durchgeführt werden. Die Umkehrung ist ebenfalls ohne weiteres möglich. Die Umkehrung der Verschiebung ist trivial und selbst zur Umkehrung der Drehung ist keine Berechnung notwendig, da sich die Inverse der Drehmatrix aufgrund der Orthogonalität von Drehmatrizen einfach als die transponierte Matrix ergibt.

Als nächster Schritt müssen die beiden gemeinsamen äußeren Tangenten der beiden Kreise und deren Berührungspunkte bestimmt werden. Dabei genügt es aufgrund der Achsensymmetrie, eine der beiden Tangenten zu berechnen (Abbildung 22).

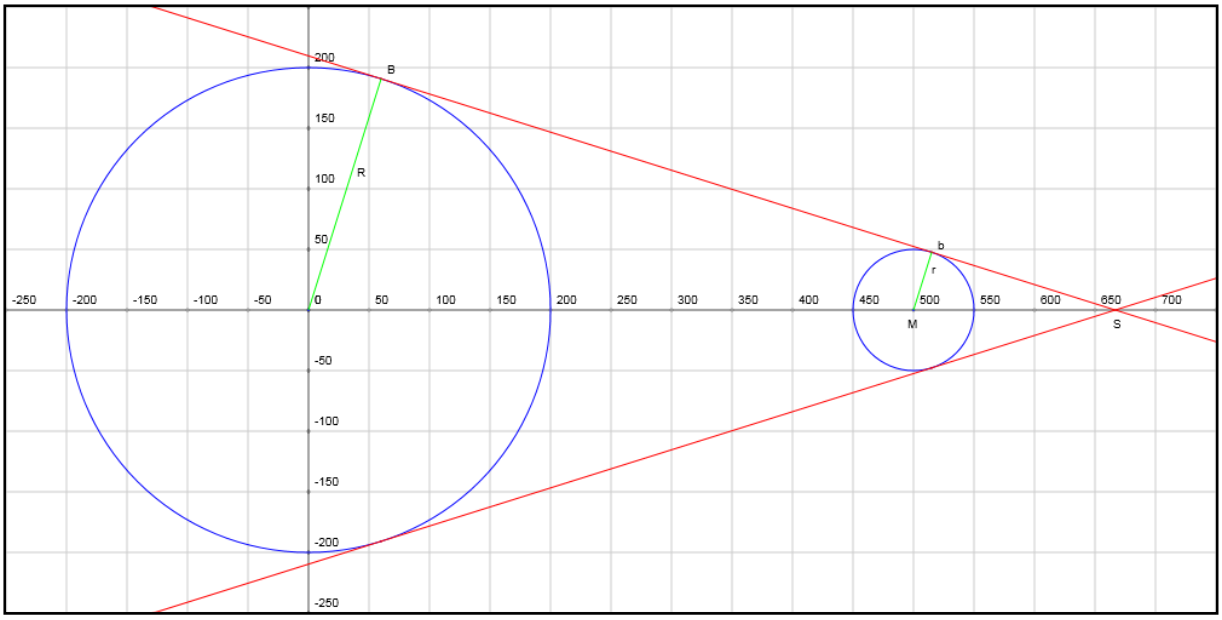

Abbildung 22: Bestimmung der gemeinsamen äußeren Tangenten der Kreise. Aufgrund der Symmetrie bezüglich der horizontalen Achse genügt es, eine Tangente zu berechnen. Die Tangente und die Berührungspunkte ergeben sich dabei aus den Gleichungen der Kreise unter Nutzung von Strahlensatz, Sinussatz und Tangensdefinition.
Quelle: Eigene Darstellung.

Die Berechnung der Parabel kann nun im allgemeinen Fall nicht direkt erfolgen, da der Scheitelpunkt horizontal mittig in Bezug auf die Berührungspunkte zwischen Tangente und Kreisen liegen soll und beide Parabeläste durch die Berührungspunkte laufen sollen. Eine Parabel, die diese Bedingungen erfüllt, kann im allgemeinen Fall (Abbildung 22) nicht gefunden werden. Als Lösung für dieses Problem wäre es möglich, zwei unabhängige Parabeläste zu konstruieren. Diese Herangehensweise könnte allerdings auch nicht immer angewendet werden, da die vertikale Position des Scheitelpunkts, die letztlich den Grad der Stauchung und somit der Bündelung bestimmt, zwischen 0 (auf der Achse, Bündelung von 100%) und dem Wert der Tangente an dieser Stelle (auf der Tangente, Bündelung von 0%) liegen kann. Damit kann es passieren, dass der Scheitelpunkt höher liegt als der Berührungspunkt mit dem kleinen Kreis. In diesem Fall wäre die Konstruktion eines positiv gekrümmten Parabelastes nicht möglich.

Um also eine einfache Konstruktion der Parabel gewährleisten zu können, muss der Spezialfall aus Abbildung 20 hergestellt werden. Das heißt, die Tangente muss derart schief gestaucht werden, dass sie achsenparallel ausgerichtet wird. Dazu müssen nun zunächst die Stellen bestimmt werden, an denen die Stützstege liegen sollen. Um Flexibilität bezüglich der Qualität des Ergebnisses im Sinne von Auflösung und Rechenleistung zu erreichen, wird dem Nutzer dabei die Möglichkeit gegeben, die Anzahl der Stützstege zu bestimmen. Im Sinne der Symmetrie wird dabei die Einschränkung getroffen, dass die Unterteilung des Gesamtbandes in Abschnitte durch die Stützstege der folgenden Forderung genügen muss:

$$k = 2^n$$

Dabei ist *k* die Anzahl der Abschnitte und *n* eine beliebige natürliche Zahl, wobei für *n* eine obere Schranke in Abhängigkeit der Länge der vorkommenden Bänder festgelegt werden

sollte, um die Rechenlast zu begrenzen. Für die verwendeten Testdaten wurde *n* <= *10* verwendet, was in allen betrachteten Fällen ausreichend war.

Für die gefundenen Stützstellen kann nun jeweils ein Stauchungsfaktor aus dem tatsächlichen Wert der Tangente an dieser Stelle und dem Wert der Tangente am Berührungspunkt mit dem kleineren Kreis bestimmt werden. Mit diesen Stauchungsfaktoren kann die Tangente achsenparallel ausgerichtet und die daraus konstruierten Parabelwerte schließlich wieder gestreckt werden (Abbildung 23). Die Berechnung der Parabel stellt in dieser Situation nun kein Problem mehr dar.

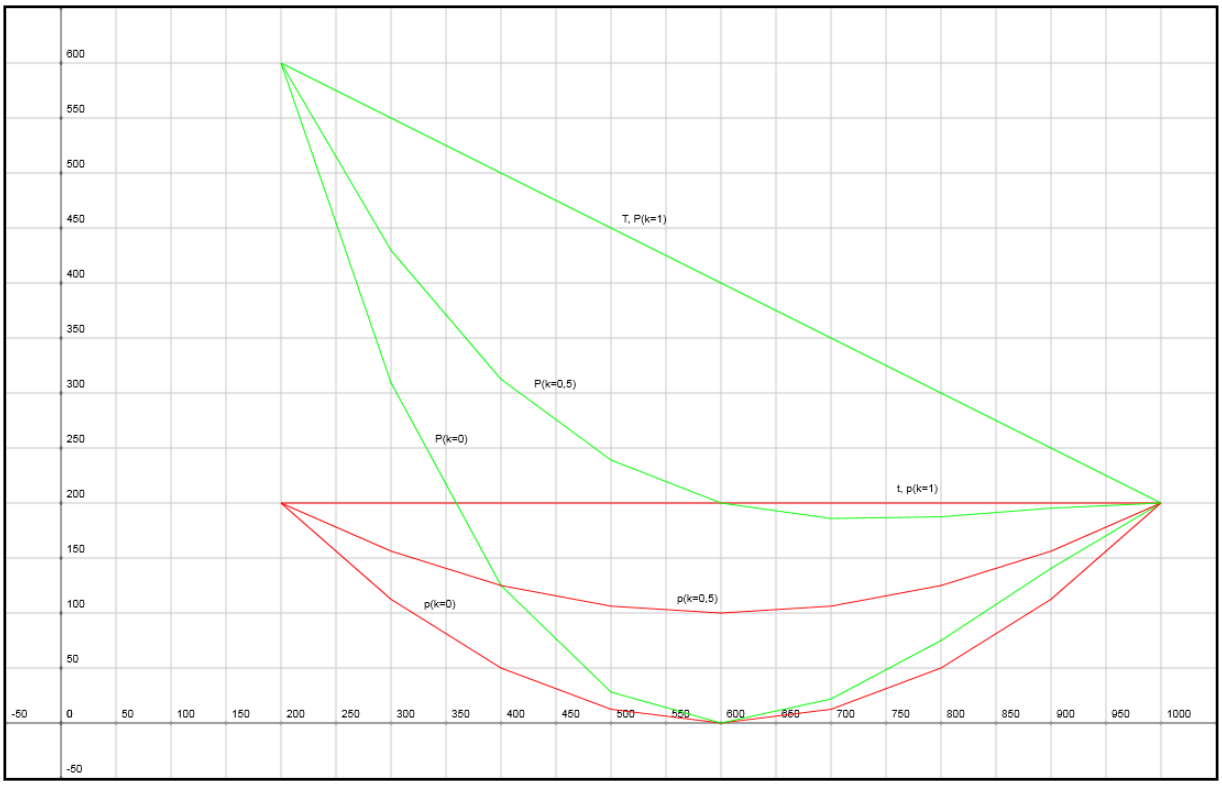

Abbildung 23: Tangente T ungestaucht (grün) und t gestaucht (rot), sowie die konstruierten Parabeln (gestaucht rot, gestreckt grün) unter Berücksichtigung des Stauchungskoeffizienten k, der sich als 1 - (s / 100) ergibt, wenn s die Bündelungsstärke in % ist. Dargestellt ist eine Bündelungsstärke von 0, 50 und 100%.
Quelle: Eigene Darstellung.

Für jede Stützstelle sind nun bekannt:

- der Funktionswert der Tangente aus der Tangentengleichung,
- der Funktionswert der Parabel aus der Parabelgleichung,
- der Funktionswert jeder Kante aus der Kantengleichung, die sich aus Start- und Endpunkt der Kante ergibt.

Damit lässt sich der neue Funktionswert jeder Kante für ihre Position in gebündelter Form als einfache Verhältnisgleichung darstellen:

$$f(x_s) = k(x_s) * \frac{p(x_s)}{t(x_s)}$$

Dabei ist x_s die Stützstelle, k die ursprüngliche Kante, p die Parabel und t die Tangente. Allerdings ist zu beachten, dass nicht jede Stützstelle für jede Kante geeignet ist. Die Stützstellen laufen vom Berührungspunkt der Tangente mit dem größeren Kreis bis zum Berührungspunkt der Tangente mit dem kleineren Kreis. Kanten können jedoch kürzer sein. An eine geeignete Stützstelle ist damit zumindest die Forderung zu stellen, dass sie zwischen dem Start- und Endpunkt der Kante liegt. Andere Stützstellen werden für die betreffende Kante ignoriert. Darüber hinaus hat es sich als zweckmäßig ergeben, nur Stützstellen außerhalb der Kreise zu verwenden, um Knickstellen an den Stellen zu vermeiden, an denen die Kantenbündelung beginnt (Abbildung 24).

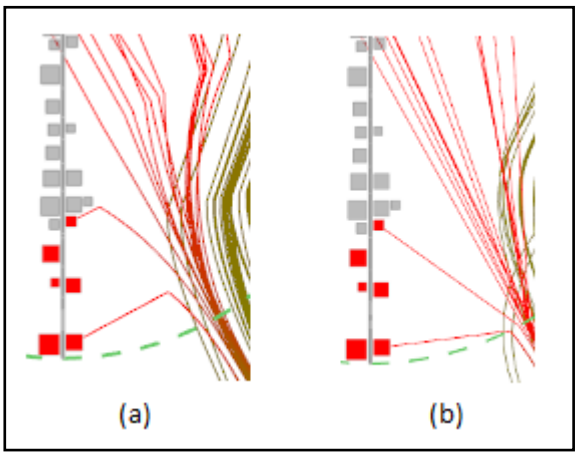

Abbildung 24: Knickstellen am Bündelungsbeginn. (a) Bündelungsbeginn an frühest möglicher Stelle, (b) reduzierte Knicke durch Bündelung erst beim Verlassen des Kreises.
Quelle: Eigene Darstellung.

Abschließend muss ins ursprüngliche Koordinatensystem zurückgekehrt werden, indem die Inversen aller bisher durchgeführten Koordinatentransformationen in umgekehrter Reihenfolge auf alle Kanten angewendet werden.

Ergebnis der Bündelung

Das Ergebnis des Bündelungsverfahrens ist beispielhaft in Abbildung 25 dargestellt. Es ist gut zu erkennen, dass abgesehen von der Vermeidung von Stadtdurchschneidungen und der Erkennbarkeit von Einzelkanten alle wichtigen und sehr wichtigen Kriterien erfüllt werden. Die Erkennbarkeit von Einzelkanten kann durch Reduzierung der Bündelungsstärke bis auf das Niveau der Ausgangssituation gerader Kanten (Abbildung 3) geregelt werden. Die Vermeidung von Stadtdurchschneidungen ist als nächster Schritt durch das eigentliche Routing zu lösen ohne dabei das gute aus der Bündelung resultierende Bild negativ zu beeinflussen.

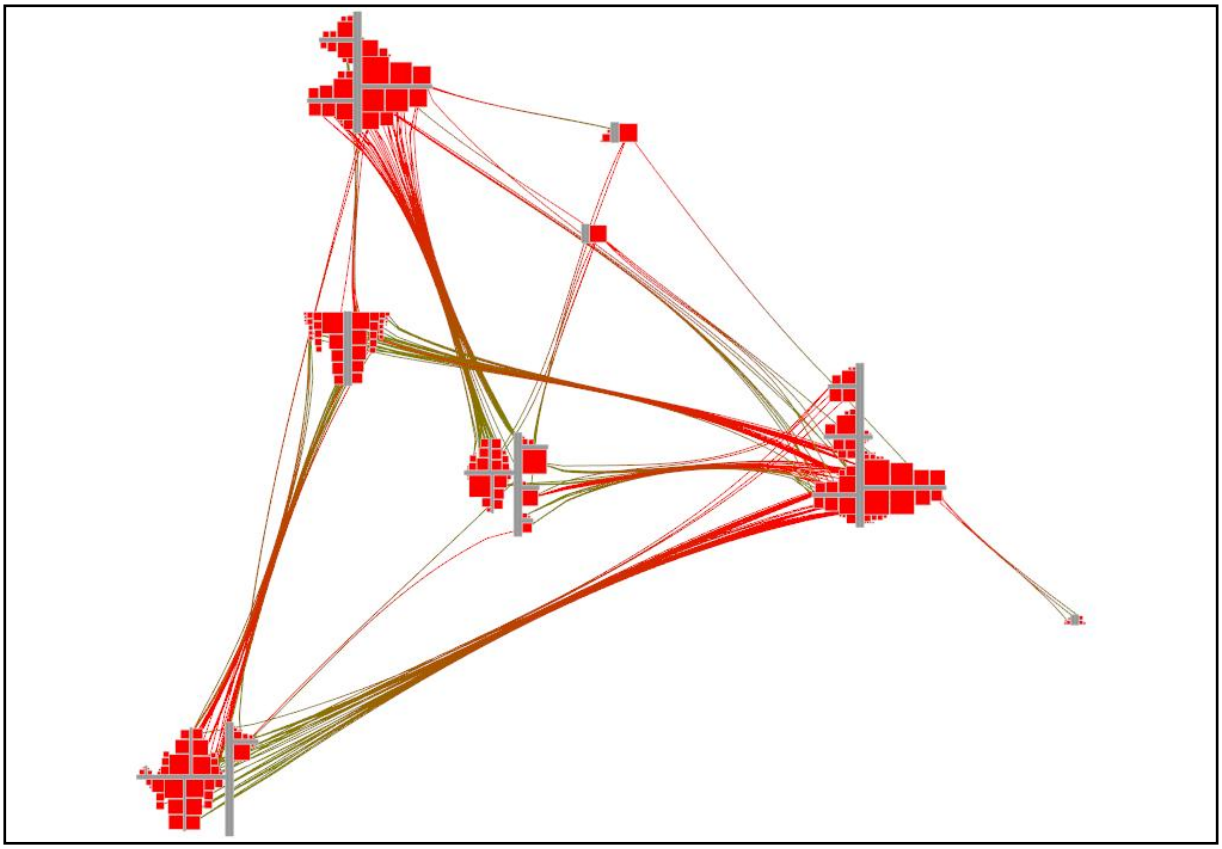

Abbildung 25: Softwarelandschaft mit Kantenbündelung aber ohne -routing. High-Level-Verbindungen zwischen Städten sind bei hoher Bündelung (hier 90%) gut zu erkennen, solange keine anderen Städte dazwischen liegen. Selbst bei hoher Kantenauflösung (hier 2^{10}) bleibt die Rechenzeit überschaubar (hier 60ms).
Quelle: Ausdruck der Visualisierung im Evoviewer.

Routing

Gegenüber anderen Verfahren besteht hier der Vorteil, dass alle Kanten zwischen je zwei Städten von einem einzigen geometrischen Objekt vollständig eingeschlossen werden. Alle Kanten zwischen je zwei Städten verlaufen innerhalb des zugehörigen Bandes, das durch seine Stützstege beschrieben wird. Damit kann die Vermeidung von Durchschneidungen der Städte durch Kanten gewährleistet werden, indem die Vermeidung von Überschneidungen der Städte mit den Bändern realisiert wird. Für jedes Paar von Städten reduziert sich damit der Aufwand der Lageberechnung aller zugehörigen Kanten zu jeder anderen Stadt auf die Lageberechnung des umschließenden Bandes zu jeder anderen Stadt. Damit sind, bei gleicher Auflösung der Kanten, im Regelfall deutlich weniger Lageberechnungen notwendig, was der Rechenzeit zu Gute kommt.

Da jedes Band durch seine Stützstege repräsentiert wird, wird die Vermeidung von Überschneidungen zwischen Band und Stadtkreisen durch Auflösung der Durchschneidungen der Stützstege mit den Kreisen realisiert. Dazu werden die betreffenden Stützstege aus dem jeweils betreffenden Kreis herausgeschoben. Wie im ersten Verfahren wird dabei die Rich-

tung gewählt, die die kürzeste Verschiebung hervorruft, um die Kanten nicht unnötig weit vom Originalverlauf abzulenken (Abbildung 26).

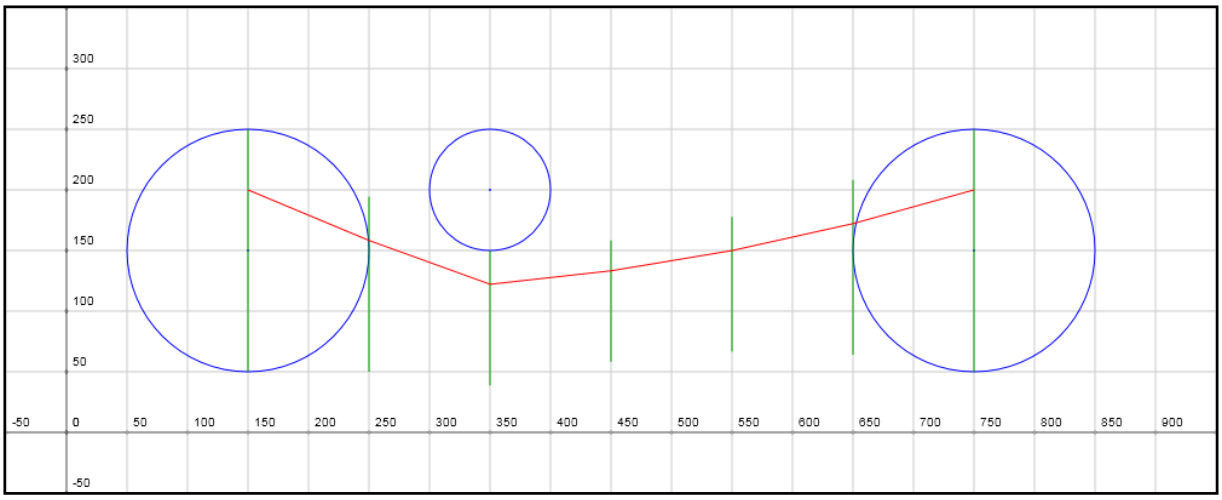

Abbildung 26: Idee des Routings beim Gummiband-Routing. Da alle Kanten innerhalb ihres gemeinsamen Bandes verlaufen, genügt es, die Stützstege des Bandes zu routen. Dargestellt ist das Beispiel aus Abbildung 20 mit einem Kreis, der im Bereich der ursprünglichen Lage des Bandes liegt und der resultierende um den Kreis herum geroutete Verlauf des Bandes.
Quelle: Eigene Darstellung.

Es liegt nahe, dass die Begrenzung der Verschiebung auf die direkt von Durchschneidungen betroffenen Stützstege zwar für die Auflösung der Durchschneidungen ausreicht, für ein qualitativ hochwertiges Bild jedoch nicht. Würde man nur die direkt betroffenen Stützstege verschieben, ergäbe sich ein Bild wie in Abbildung 27, bei dem das Band direkt am Kreisradius entlang gelegt würde. Es entstünde einmal mehr eine Art Zick-Zack-Layout mit unnötig starken Krümmungen, bei dem das Auge dem Kurvenverlauf teilweise nur noch schwer folgen kann.

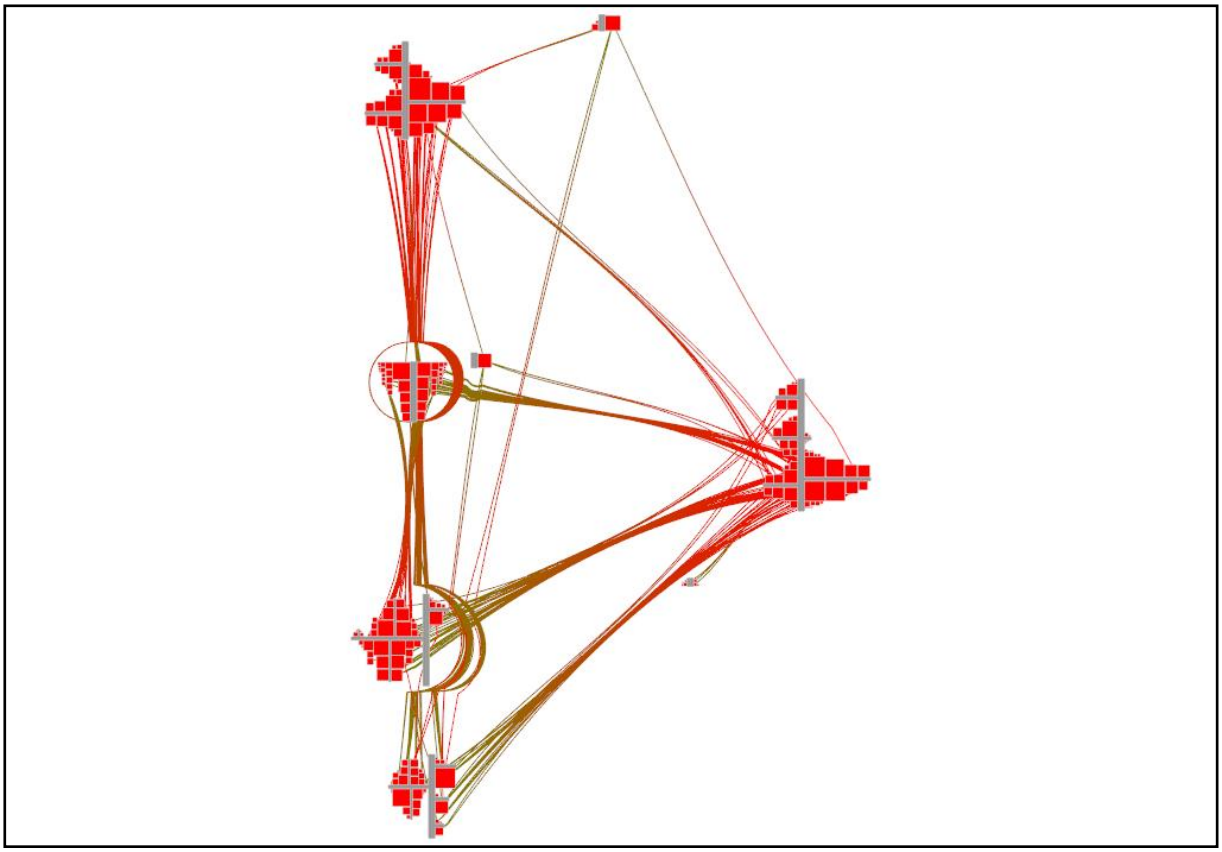

Abbildung 27: Alleinige Verschiebung der direkt von Durchschneidungen betroffenen Stützstege. Offensichtlich stellt sich kein qualitativ hochwertiges Bild ein.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Deshalb ist es zunächst sinnvoll, wie schon in der Konzeptzeichnung aus Abbildung 26 zu sehen, auch alle anderen Stützstege anteilig mit zu verschieben. Vorläufig wird dafür der Einfachheit halber eine lineare Abnahme bis zum Start- und Endpunkt des Bandes verwendet.

Die an dieser Stelle realisierte erste Implementierung des Routings der Bänder zeigt nun allerdings noch einige Schwächen. Das offensichtlichste Problem resultiert dabei aus der Tatsache, dass im Allgemeinen pro Band und Stadt für mehrere Stützstege Durchschneidungen gefunden werden. Wird das Band bei der Suche nach Durchschneidungen in einer Richtung durchlaufen und für jede Durchschneidung die resultierende Verschiebung auf das gesamte Band (anteilig) ausgeführt, bevor der nächste Stützsteg untersucht wird, entsteht ein unsymmetrisches Bild. Die ersten Stützstege, deren Durchschneidungen mit der aktuellen Stadt bereits aufgelöst wurden, werden durch jeden folgenden Stützsteg weiter von der Stadt weg verschoben. Das entstehende Bild ist somit nicht unabhängig von der Richtung, in der das Band durchlaufen wird (Abbildung 28).

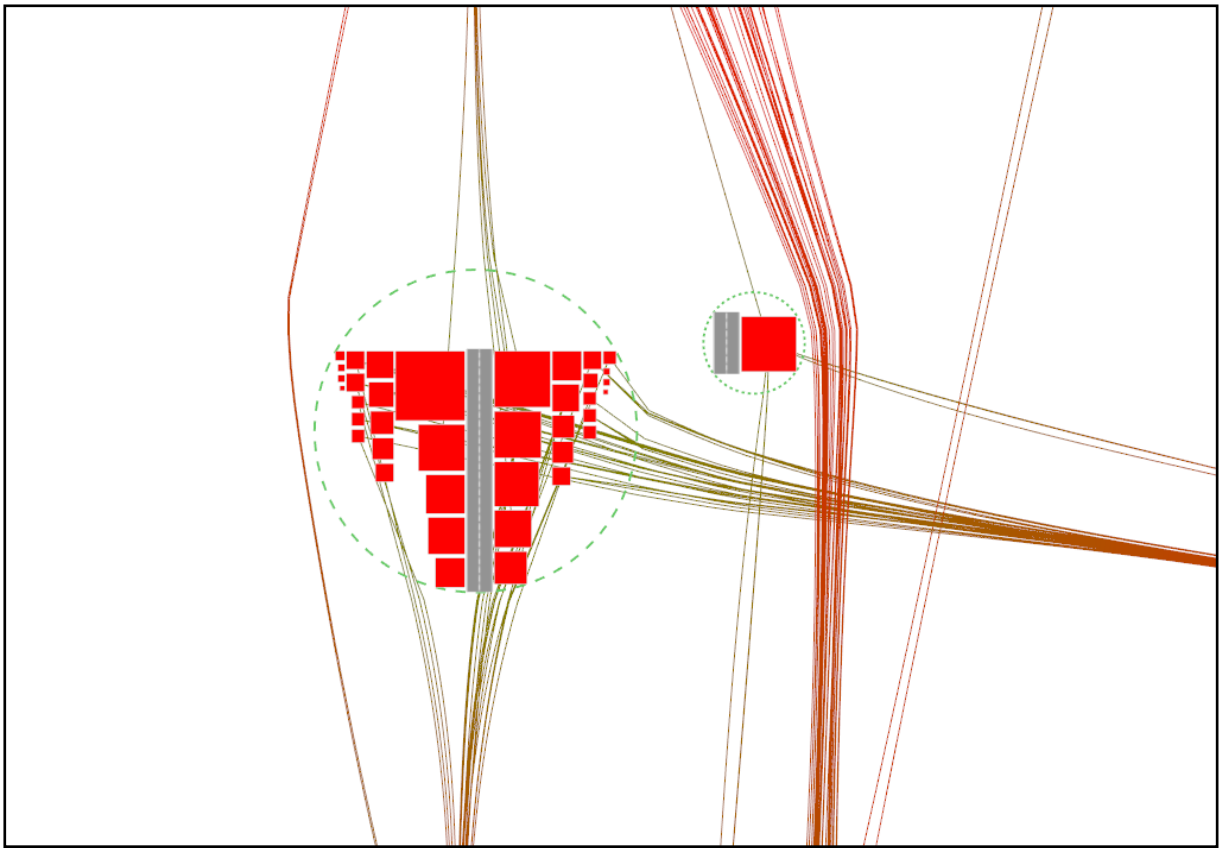

Abbildung 28: Unsymmetrisches Bild durch Abhängigkeit von der Richtung, in der das Band beim Routing durchlaufen wird. Im dargestellten Beispiel wurde das Band von oben nach unten durchlaufen.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Um das Problem zu lösen, kann die Reihenfolge der Abarbeitung so verändert werden, dass in Bezug auf jede Stadt symmetrisch vorgegangen wird. Dazu werden zunächst alle Durchschneidungen von Stützstegen mit einer Stadt und alle resultierenden Verschiebungen der Stützstege ermittelt und gesammelt. Die Ausführung der Verschiebungen kann dann entweder in der geometrischen Mitte aller gesammelten Verschiebungen begonnen werden oder aber bei dem Stützsteg, für den die längste Verschiebung ermittelt wurde. Nachdem für diesen einen Steg die Verschiebung ausgeführt wurde, inklusive der anteiligen Verschiebungen für alle anderen Stützstege, können die anderen Verschiebungen abgearbeitet werden, jeweils reduziert um die bereits ausgeführten anteiligen Verschiebungen. Dabei kann entweder linear immer der nächste Nachbar genutzt oder rekursiv vorgegangen werden, indem für jede Seite wiederum die Mitte bzw. die stärkste Verschiebung bestimmt wird. Die Asymmetrie wird dadurch weitestgehend aufgelöst, wenn man davon absieht, dass immer noch die Entscheidung zu treffen ist, ob ausgehend von einer mittleren oder stärksten Verschiebung erst nach links oder erst nach rechts fortgefahren wird. Dieses Restproblem ließe sich beheben, indem Verschiebungen links der Mitte bzw. der stärksten Verschiebung nur nach links anteilige Verschiebungen bewirken und Verschiebungen rechts entsprechend nur nach rechts.

Allerdings bliebe bei den skizzierten Vorgehensweisen ein fundamentaleres Problem. Es kann passieren, dass für verschiedene Stützstege im Kontext einer Stadt gemäß dem An-

satz, Verschiebungen klein zu halten, Verschiebungen in entgegengesetzte Richtungen ermittelt werden. Dieses Zerreißen des Bandes, wenn einige Stützstege links neben die Stadt geroutet werden, andere rechts, muss zwingend mit bedacht werden. Entweder müsste von vornherein als zusätzlicher Schritt eine Richtung bestimmt werden, die für alle Stützstege im Bereich einer Stadt bindend ist, oder es müsste nach Ausführung der ersten Verschiebung die Verschiebung für alle anderen Stege durch erneute Lageberechnung gegenüber der Stadt nochmal errechnet werden. In beiden Fällen würde erheblicher Mehraufwand entstehen.

Bei der probehalber erfolgten Umsetzung des Ansatzes mit wiederholten Lageberechnungen bei linearer Weiterführung nach links und rechts von der Maximalverschiebung aus, konnte jedoch neben der wie erwartet erheblich ansteigenden Rechenzeit beobachtet werden, dass bei den wiederholten Lageberechnungen nur noch sehr selten Schnitte festgestellt werden konnten und wenn, dann meistens in extremen Ausnahmesituationen, in denen die wiederholten Verschiebungen zu qualitativ minderwertigen Ergebnissen führten. Im Sinne eines angemessenen Verhältnisses zwischen Kosten und Nutzen habe ich deshalb die Entscheidung getroffen, nach Bestimmung aller potentiellen Verschiebungen von Stützstegen gegenüber einer Stadt nur noch die maximale Verschiebung auszuführen, wobei die anteiligen Verschiebungen anderer Stützstege natürlich unangetastet bleiben. Alle anderen ermittelten Verschiebungen im Kontext der Stadt werden ignoriert, da in den allermeisten Fällen davon auszugehen ist, dass keine Schnitte mehr existieren oder nur noch sehr kleine.

Damit ist allerdings klar, dass absolute Überschneidungsfreiheit nicht mehr garantiert werden kann. Trotzdem halte ich den Ansatz für angemessen, da neben dem Argument der Rechenzeit auch beachtet werden muss, dass hierbei von Überschneidungen zwischen Bändern und Stadtumkreisen gesprochen wird. Existieren nun noch kleine Überschneidungen zwischen Bändern und Umkreisen, was schon an sich eher selten ist, ist es umso unwahrscheinlicher, dass auch noch Überschneidungen zwischen den eigentlichen Kanten und den Elementen einer Stadt existieren. Dazu sollte man sich vor Augen führen, dass nur wenige Elemente einer Stadt in relativer Nähe zur Umkreislinie liegen und noch weniger Kanten an den Rändern eines Bandes, da der Ausgangspunkt für die Ermittlung der Bandränder die gemeinsamen äußeren Tangenten der Umkreise der Städte sind und bei Stauchung der Stützstege beim Bündeln die relativen Positionen der Einzelkanten im Band konstant bleiben. Wie viele Schnitte nach Durchlaufen des Verfahrens noch existieren, wird im folgenden Kapitel in der Evaluierung noch zu thematisieren sein.

Aus dem anteiligen Verschieben der vor- und nachgelagerten Stützstellen resultiert allerdings noch ein weiteres Problem. Angenommen, bei der Abarbeitung der Stützstege wird zunächst eine Überschneidung zwischen dem Band und einer Stadt entdeckt und behoben und im weiteren Verlauf eine Überschneidung mit einer weiteren Stadt gefunden. Beim bisherigen Design des Verfahrens, hätte die Ablenkung des Bandes im Kontext der zweiten Stadt anteilige Verschiebungen im Bereich des gesamten Bandes zur Folge, also auch im Bereich der ersten Stadt. Je nach Richtung kann das Band dabei im Bereich der ersten Stadt noch weiter von dieser weg oder aber zurück in diese hinein verschoben werden. Das Phänomen ist zwar durch die Wahl der Verarbeitungsreihenfolge der Städte von groß nach klein insofern reduziert, dass zumindest im statistischen Mittel die größten Umlenkungen zuerst stattfinden. Damit sind die Korrekturen durch spätere Umlenkungen vergleichsweise klein.

Außerdem finden so kleinere Umlenkungen teilweise gar nicht erst statt, da die Überschneidungen mit kleineren Städten im ursprünglichen Verlauf zum Teil schon durch die vorherigen großen Umlenkungen aufgelöst wurden. Das ist der für die optische Qualität wahrscheinlich größte Vorteil der Sortierung der Städte nach dem Umkreisradius, da auf diese Weise unnötig viele Kurven im Verlauf eines Bandes vermieden werden.

Vollständig gelöst ist das Problem damit aber noch nicht. Als Lösung hierfür führe ich neben den beiden Fixpunkten an Anfang und Ende des Bandes, die als Bezugspunkte für das anteilige Verschieben dienen, die bereits stattgefundenen Verschiebungen als weitere ein. Die anteilige Verschiebung nimmt nun nicht mehr bis zum Start- und Endpunkt des Bandes ab, sondern bis zum jeweils nächst gelegenen bereits ermittelten (maximalen) Verschiebungspunkt. Start- und Endpunkt werden nur noch als Default-Bezugspunkte genutzt, wenn es in der entsprechenden Richtung noch keinen gespeicherten Verschiebungspunkt gibt (Abbildung 29).

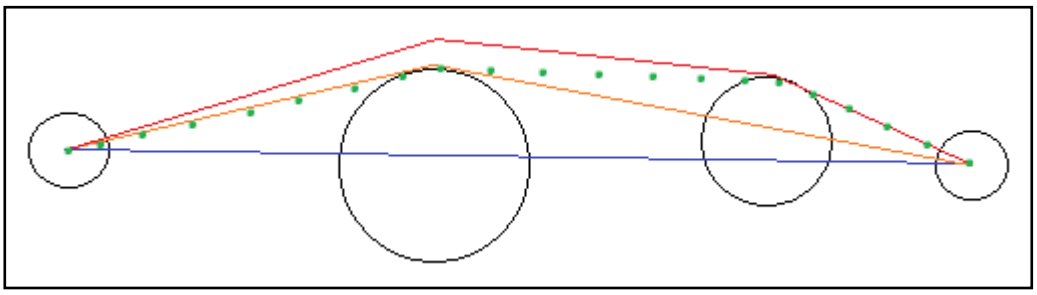

Abbildung 29: Wiederholtes Verschieben bei Schnitten mit mehreren Kreisen, vereinfachte Darstellung. Der ursprüngliche Verlauf (blau) wird zuerst durch den größeren Kreis verändert (orange). Werden nur Start- und Endpunkt des Bandes als Fixpunkte verwendet, entsteht nach Abarbeitung aller Kreise der rote Verlauf. Werden auch die bereits gefundenen (maximalen) Verschiebungspunkte als Fixpunkte verwendet, entsteht der grüne Verlauf.
Quelle: Eigene Darstellung.

Diese Lösung generiert nun allerdings ein neues Problem. Vorher sind auch bei mehreren Umlenkungen eines Bandes durch die lineare Abnahme der anteiligen Verschiebung bis zum Start- und Endpunkt des Bandes über den gesamten Verlauf relativ flache Kurven entstanden. Da nun die Abnahme der anteiligen Verschiebung nur noch bis zu benachbarten maximalen Verschiebungspunkten läuft, wird die Steigung des Bandes durch eine Umlenkung auch nur noch bis zu diesen Punkten beeinflusst. Hinter diesen Punkten wird die Steigung nicht mehr verändert und somit geht der fließende Übergang, der durch die anteilige Verschiebung erreicht wurde, verloren und es entstehen Knicke an allen maximalen Verschiebungspunkten.

Da das Problem offensichtlich durch die Überlagerung des Bandverlaufs mit linearen Funktionen mit unterschiedlichen Anstiegen entsteht, liegt es nahe, diese Funktionen, die die Abnahme der anteiligen Verschiebung beschreiben, durch geeignetere zu ersetzen. An diese geeigneteren Funktionen ist zumindest die Forderung zu stellen, dass sie am Anfang und Ende ihres Verlaufs von 100 nach 0% den gleichen Anstieg besitzen, um einen fließenden Übergang zu erreichen. Am einfachsten lässt sich diese Forderung durch Punktsymmetrie gegenüber dem Funktionspunkt bei 50% erreichen. Darüber hinaus halte ich es für sinnvoll, dass die Abnahme am Anfang relativ langsam vonstattengeht. Auf diese Weise kann ge-

währleistet werden, dass in relativer Nähe der maximalen Verschiebung anteilige Verschiebungen stattfinden, die nur geringfügig kleiner sind. Damit steigt die Wahrscheinlichkeit, dass der Verlauf des Kreisbogens in dem Bereich schneller abfällt als der Verlauf des Bandes, womit kleine übrig gebliebene Überschneidungen zwischen Band und Kreis eliminiert werden. Klar ist außerdem, dass nicht unnötig viele Kurven entstehen sollten, es sollte also nur einen Wendepunkt der Krümmung geben. Die so skizzierte Funktion sollte damit etwa dem Verlauf aus Abbildung 30 folgen.

Abbildung 30: Skizze der Funktion, die die Abnahme der anteiligen Verschiebung beschreiben soll. Auf der horizontalen Achse ist die Nähe zum maximalen Verschiebungspunkt in Prozent dargestellt (100%: der Verschiebungspunkt selbst, 0%: der nächste Bezugspunkt), auf der vertikalen Achse der Anteil der maximalen Verschiebung, der genutzt werden soll, ebenfalls in Prozent.
Quelle: Eigene Darstellung.

Da der skizzierte Verlauf stark an eine Sinuskurve erinnert, verwende ich für die Beschreibung der Abnahme der anteiligen Verschiebung eine modifizierte Sinusfunktion:

$$f(x) = \frac{1}{2} * \sin(\pi * (x - 0{,}5)) + \frac{1}{2}$$

Auf diese Weise werden nun fließende Übergänge der einzelnen Bandabschnitte gewährleistet und die meisten übrigen Überschneidungen zwischen Städten und Bändern aufgelöst. Allerdings können Überschneidungen beim Gummiband-Routing noch an ganz anderer Stelle entstehen. So vorteilhaft die Beschreibung aller Kanten zwischen je zwei Städten durch ein umgebendes Band ist, erzeugt sie unumgänglich ein neues Problem, das bei allen anderen Verfahren nicht besteht. Während beim Routing der einzelnen Kanten davon ausgegangen werden kann, dass jede Kante durch jede Lücke zwischen Städten hindurch geroutet werden kann, ist das hier nicht der Fall. Einzelkanten sind in der Breite dimensionslos und passen somit durch jede noch so enge Lücke. Für die hier beschriebenen Bänder gilt dies naturgemäß nicht. Da jedes Band eine Breite hat, kann es nicht durch jede Passage zwischen Städten geführt werden (Abbildung 31).

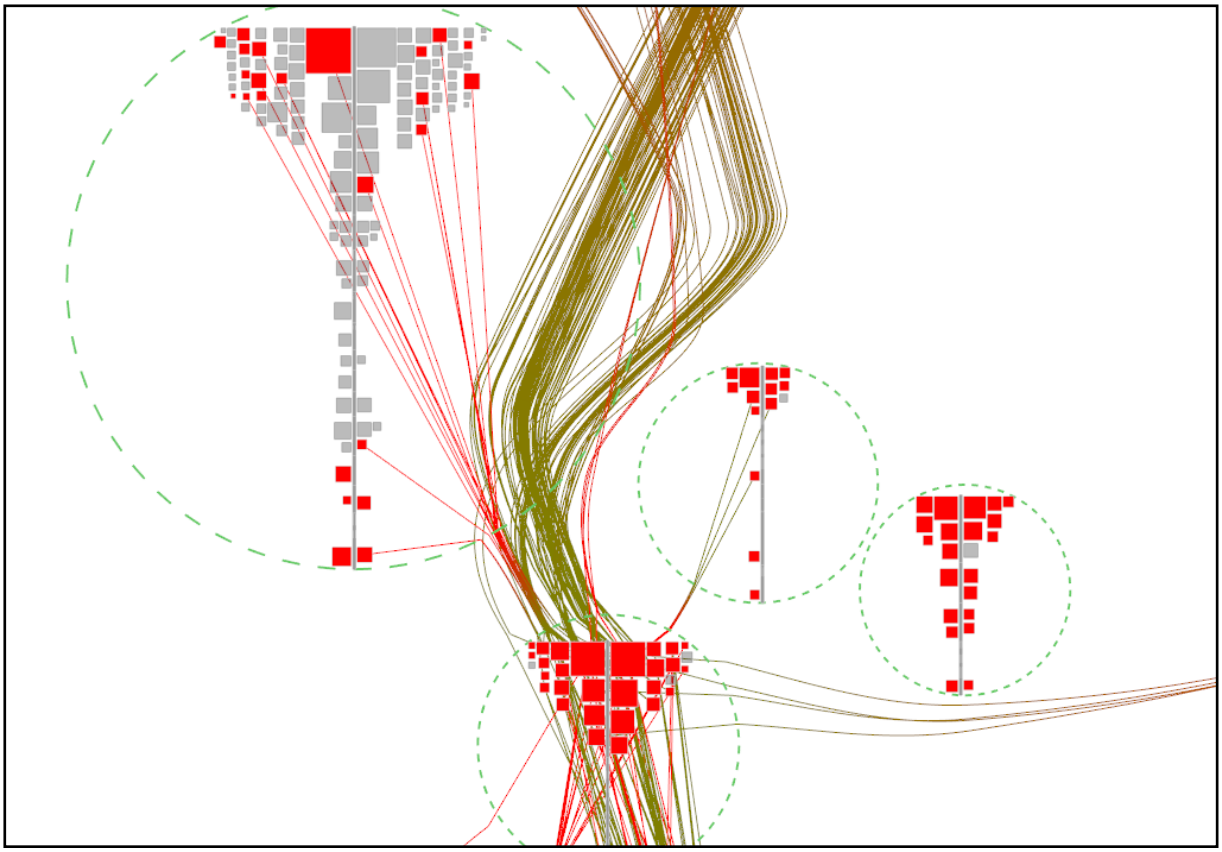

Abbildung 31: Das Problem der Engstellen. Da Bänder im Gegensatz zu Einzelkanten eine Breite haben, können sie nicht durch jede Lücke zwischen Städten geführt werden.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Diese Tatsache hat weitreichende Konsequenzen. Da nicht jede Lücke geeignet ist, wäre ein naheliegender Ansatz, geeignete Lücken zu suchen. Damit müsste das Verfahren aber über lokale Untersuchungen bezüglich der Lage zwischen einzelnen geometrischen Objekten zueinander hinausgehen. Statt lokaler Umleitungen müsste global nach einem geeigneten Weg gesucht werden, da im Labyrinth zwischen den Städten jeder Weg eine potentielle Sackgasse wäre. Erschwert würde das Ganze dadurch, dass die hier konzipierten Bänder keine feste Breite haben, die über die Gesamtlänge konstant bleibt. Je nach Weg würde sich die Länge eines Bandes verändern und damit auch die Breite an allen vorher bestimmten Positionen. Hinzu kommt, dass die Aufhängung der Bänder an ihren Start- und Zielstädten fixiert und (bisher) nicht drehbar ist, was die Start- und Endverlaufsrichtung einschränkt. Insgesamt zwar keine unlösbare Aufgabe, aber in Hinblick darauf, dass die geringe Rechenzeit eines der wichtigsten Kriterien für die Entwicklung einer Lösung darstellt, kein allzu guter Ansatz. Zumal nicht einmal garantiert werden kann, dass auf diese Weise überhaupt ein Weg gefunden werden kann, beispielsweise wenn eine Stadt vollständig von anderen Städten umringt ist und es nur schmale Durchgänge gibt.

Aus diesen Überlegungen heraus verfolge ich einen anderen Ansatz. An sich ist es nur die konsequente Anwendung der Eigenschaften der zu Grunde liegenden Metapher. Wie würde sich ein Gummiband an Engstellen verhalten? Es könnte sich einfach weiter zusammenziehen. Und genau das ist auch die Lösung in diesem Fall. Wurde für ein Band eine maximale Verschiebungsstelle gefunden und die Verschiebung und zugehörige anteilige Verschiebun-

gen ausgeführt, wird – bevor mit der Suche fortgefahren wird – geprüft, ob die Verschiebung in eine andere Stadt hinein geführt hat. Ist das der Fall, wird nicht zurückgeschoben, was bisher der Fall war, sondern zurückgestaucht. Die Stauchung funktioniert dabei analog zur Verschiebung, d.h. es wird ebenfalls anteilig bis zu benachbarten Maximalpunkten sinusförmig abnehmend weiter gestaucht. Die Tatsache, dass dabei nur an der maximalen Verschiebungsstelle auf Verschiebung in eine andere Stadt hinein geprüft wird, hat neben geringem Rechenaufwand den Vorteil, dass die Stauchung so gering wie möglich ausfällt. Ist die Verschiebung an einer anderen Stelle in der näheren Umgebung weiter in die andere Stadt hinein erfolgt, wird diese Stelle beim weiteren Abarbeiten der Städte ohnehin gefunden und durch eine kleine Verschiebung in die Gegenrichtung gelöst.

Ein letztes selten auftretendes Problem besteht nun noch darin, dass bei Verschiebungen des Bandes bei spät untersuchten Kreisen Verschiebungen in früher untersuchte Kreise entstehen können. Das Problem tritt vor allem deshalb selten auf, weil durch die Reihenfolge der Städte von groß nach klein die Verschiebungen nach hinten hin statistisch immer kleiner werden. Es tritt also nur dann auf, wenn Bänder anfangs so dicht an Städten vorbeilaufen, dass sie durch kleine Verschiebungen hinein gelangen können, ohne dass sie diese Städte jedoch schon von vornherein durchschneiden. Hier sehe ich keine andere praktikable Lösung, als die Städte erneut zu durchlaufen, um solche Fälle auszuschließen. Da das Problem an sich jedoch nur selten auftritt und durch erneute Untersuchung aller Städte die Rechenzeit damit häufig unnötig erhöht würde, biete ich dem Nutzer die Möglichkeit, die Anzahl der Ausführungen zu begrenzen.

Mit dem nun erreichten Zustand des Algorithmus betrachte ich die Optimierung vorläufig als abgeschlossen, da sowohl die optischen Ergebnisse als auch die benötigte Rechenzeit zumindest auf den ersten Blick zufriedenstellend sind (Abbildung 32). Eine ausführliche Evaluierung im folgenden Kapitel wird zeigen, ob diese Aussage auch einer genaueren Betrachtung standhält.

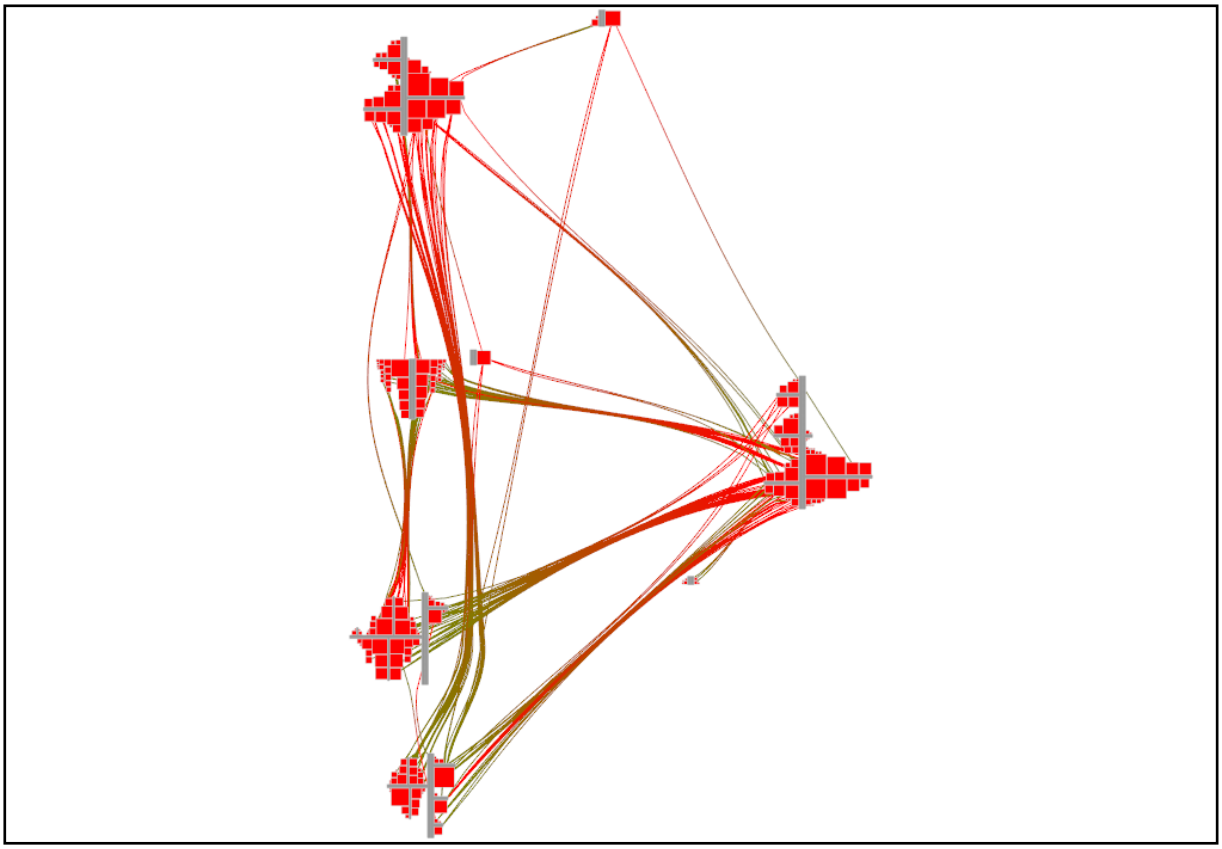

Abbildung 32: Softwarelandschaft mit vollständigem Gummiband-Routing. Zumindest auf den ersten Blick kann die optische Qualität überzeugen. Auch die Rechenzeit bleibt im Rahmen (hier 157ms bei einer Kantenauflösung von 2^{10}).
Quelle: Ausdruck der Visualisierung im Evoviewer.

4. Evaluierung

In diesem Kapitel sollen die bisher erzielten Ergebnisse evaluiert werden. Dazu werde ich im ersten Teil festlegen, wie die Erreichung der im vorherigen Kapitel definierten Zielkriterien gemessen werden soll. Im zweiten Teil werde ich daraufhin geeignete Beispielsysteme zur Durchführung der Messungen ermitteln. Im dritten Teil werde ich die einzelnen Verfahren dann anhand der Messungen der Zielkriterien an den Beispielsystemen vergleichend bewerten.

4.1 Messung der Zielkriterien

In diesem Abschnitt soll die Messung der Zielkriterien festgelegt werden. Die geplante Messung ist in nachstehender Tabelle 1 zusammengefasst.

Kriterium	**Messung**
K1.1 Geringe Rechenzeit	Zeitmessung
K1.2 Vermeidung von Stadtdurchschneidungen	Zählung übrigbleibender Durchschneidungen
K1.3 Erkennbarkeit von High-Level-Zusammenhängen	Vergleichende optische Einschätzung
K1.4 Semantisches Routing	Logische Beurteilung des Algorithmus
K2.1 Erkennbarkeit von Low-Level-Zusammenhängen	Vergleichende optische Einschätzung
K2.2 Vermeidung von Kantenkrümmungen	Vergleichende optische Einschätzung; Beurteilung, ob Verlauf mit weniger Krümmungen sinnvoll möglich wäre
K2.3 Flache Kantenkrümmungen	Vergleichende optische Einschätzung; Beurteilung, ob Verlauf mit flacheren Krümmungen sinnvoll möglich wäre
K3.1 Räumliche Kompaktheit	Vergleichende optische Einschätzung; Beurteilung, ob kompakterer Verlauf sinnvoll möglich wäre
K3.2 Vermeidung von Kantenkreuzungen	Logische Beurteilung des Algorithmus und vergleichende optische Einschätzung
K3.3 Möglichst kurze Kanten	Messung der durchschnittlichen Kantenlängen

Tabelle 1: Messung der Zielkriterien.
Quelle: Eigene Darstellung.

Leider sind nicht alle Kriterien direkt automatisiert messbar. Am einfachsten direkt zu messen sind die Rechenzeit, die Anzahl der Stadtdurchschneidungen und die durchschnittliche Länge der Kanten. Da es bei der Messung nicht auf eine gute Performance sondern mehr auf Genauigkeit ankommt, werde ich zur Berechnung von Durchschneidungen im Gegensatz zu den beiden entwickelten Verfahren Städte nicht als deren Umkreise abstrahieren, sondern als deren konvexe Hüllen.

Inwieweit semantische Informationen Einfluss auf das jeweilige Routingverfahren haben und inwieweit Kantenkreuzungen vermieden werden, lässt sich anhand der logischen Struktur der Verfahren beurteilen. Alle weiteren Kriterien lassen sich meiner Ansicht nach nur durch optische Einschätzung beurteilen. Obwohl mir klar ist, dass die damit einhergehende Subjektivität der Beurteilung diese angreifbar macht, muss doch beachtet werden, dass das wesentlichste Ziel der Softwarevisualisierung eben die auf optische Erfassbarkeit optimierte Darstellung von Informationen ist. Damit ist klar, dass ohne optische Einschätzung nur schwer ein Urteil getroffen werden kann. Neben der vergleichenden optischen Einschätzung der Ergebnisse aller Verfahren strebe ich bei einigen der Kriterien außerdem eine Beurteilung dahingehend an, ob eine mögliche stärkere Erfüllung dieser im konkreten Fall im Hinblick auf optische Qualität auch sinnvoll wäre.

4.2 Auswahl geeigneter Beispielsysteme

Ziel dieses Abschnitts ist es, aus dem Pool der Softwaresysteme, für die die relevanten Daten zur Verfügung stehen, geeignete Kandidaten auszuwählen. Dabei sollen vor allem Softwaresysteme verschiedener Größenordnungen verwendet werden, um jeweils unterschiedliche Ansprüche an die zu testenden Verfahren zu stellen. Als Maßstab für die Größenordnung soll dazu in erster Linie die Struktur der Systeme dienen, insbesondere die Anzahl der Teilsysteme (Städte) und die Anzahl der Verbindungen zwischen den einzelnen Elementen (Kanten). Insgesamt wähle ich zur Erfüllung dieser Bedingungen drei Systeme, die im Folgenden kurz vorgestellt werden sollen.

Das erste und kleinste System ist CrocoCosmo, wobei Pakete der obersten Ebene (unterhalb des Wurzelpakets) als Städte dargestellt werden sollen (Abbildung 33). Das System umfasst 8 Städte mit insgesamt 174 Kanten zwischen den Städten. Daraus resultieren für das Gummiband-Routing 28 Bänder. Besonders interessant ist dabei vor allem die Anordnung der Städte im linken Bereich in einer Linie.

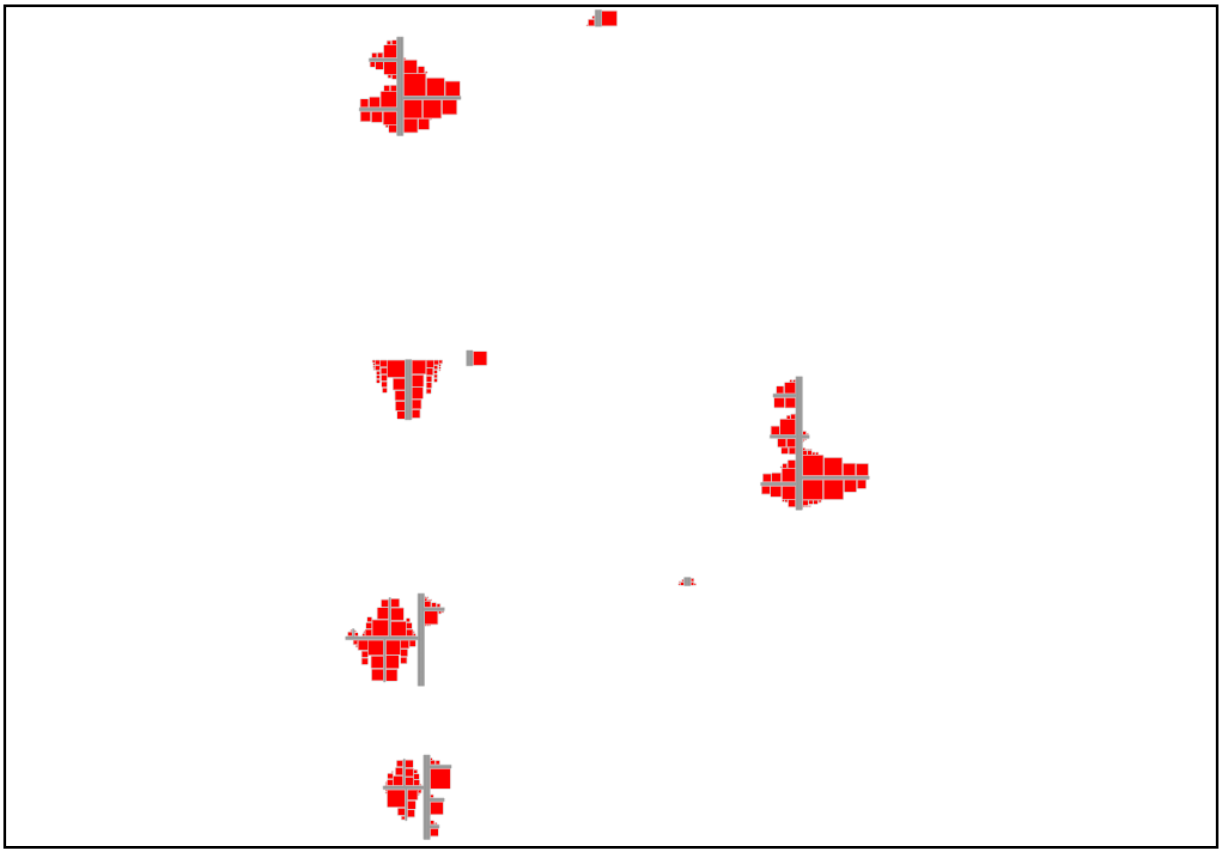

Abbildung 33: CrocoCosmo mit Städten basierend auf der obersten Paketebene.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Als zweites, mittelgroßes System wähle ich ebenfalls CrocoCosmo, wobei diesmal Pakete unterster Ebene als Städte dargestellt werden sollen (Abbildung 34). Das System umfasst damit nun 21 Städte und 217 Kanten zwischen diesen. Daraus resultieren für das Gummiband-Routing 210 Bänder. Interessant ist hier vor allem die relativ dichte Verteilung der Städte, besonders im Bereich rechts unten.

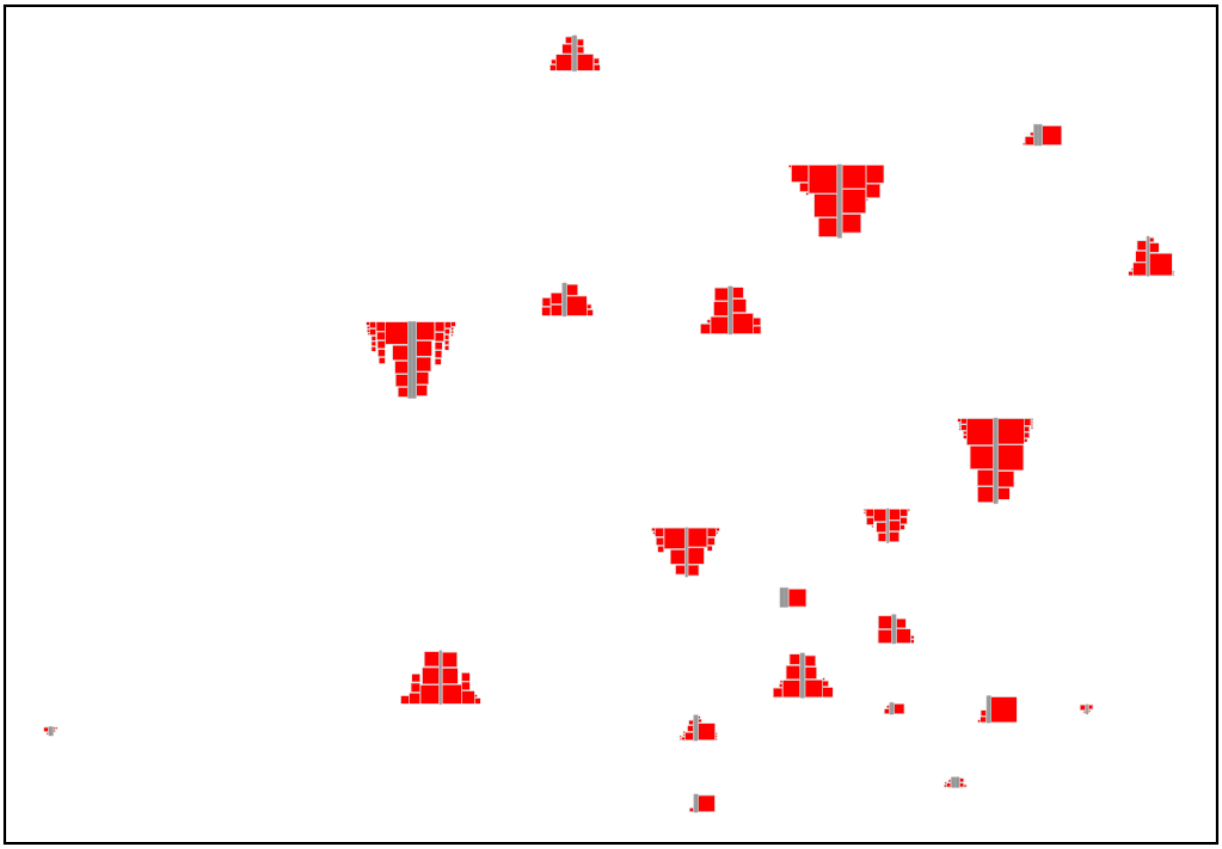

Abbildung 34: CrocoCosmo mit Städten basierend auf der niedrigsten Paketebene.
Quelle: Ausdruck der Visualisierung im Evoviewer.

Als drittes und größtes System wähle ich JFreeChart mit Darstellung der Pakete vierter Ebene als Städte (Abbildung 35). Das System umfasst 35 Städte mit 1901 Kanten zwischen diesen. Daraus resultieren für das Gummiband-Routing insgesamt 595 Bänder. Abgesehen von der Gesamtgröße des Systems ist hier vor allem der mittlere Bereich interessant, da die Städte dort sehr dicht beieinander liegen. Auf die Darstellung noch wesentlich größerer Systeme verzichte ich hier, da diese bei üblichen Bildschirmauflösungen ohnehin nicht mehr vollständig und auf einer Zoomstufe so gezeichnet werden können, dass sie noch erfassbar sind.

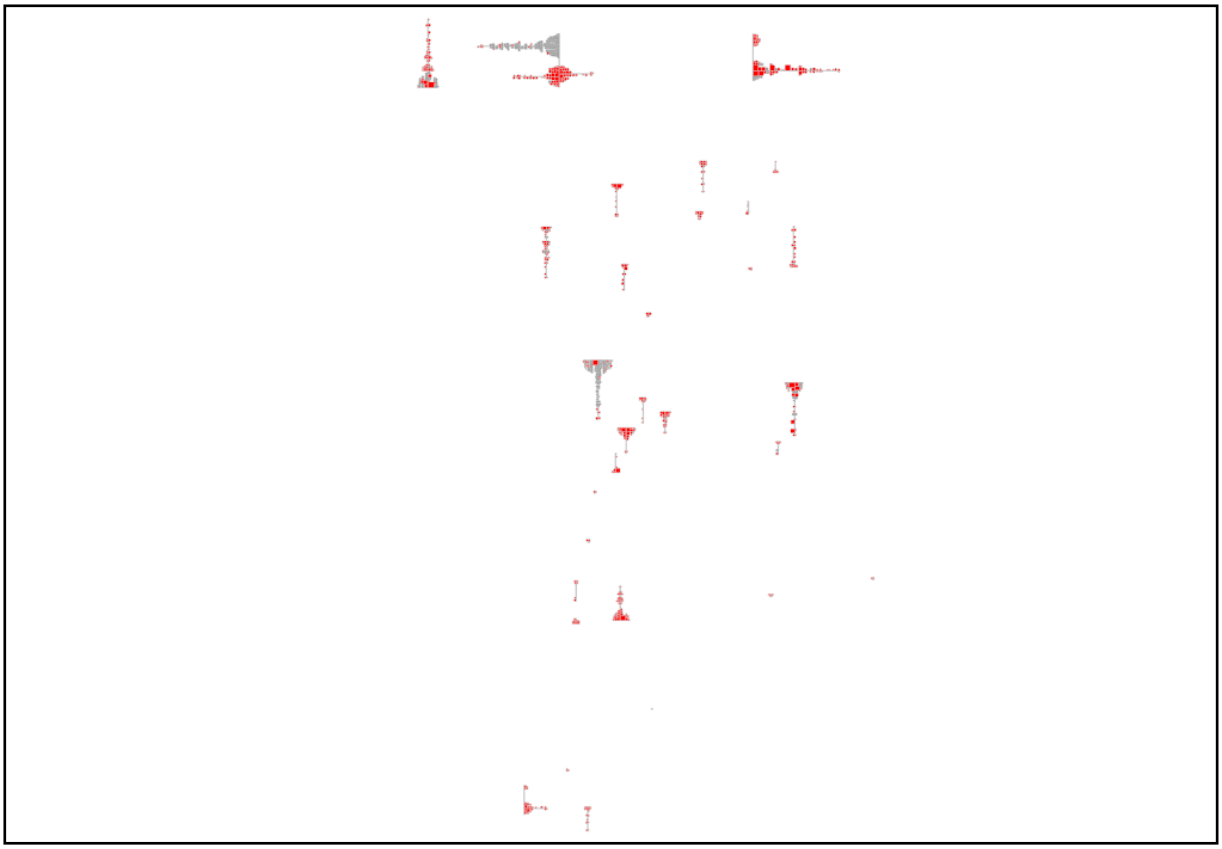

Abbildung 35: JFreeChart mit Städten basierend auf der vierten Paketebene.
Quelle: Ausdruck der Visualisierung im Evoviewer.

4.3 Vergleich der Ergebnisse der verschiedenen Verfahren

Hier werden nun die verschiedenen Verfahren hinsichtlich der gelieferten Ergebnisse bezogen auf die definierten Kriterien anhand der drei Beispielsysteme beurteilt. Als Referenz wird dazu auch das jeweils erzielte Ergebnis ganz ohne Routing mit einfachen geraden Kanten dargestellt. Um den Umfang der Arbeit in Grenzen zu halten, werden die der optischen Einschätzung zu Grunde liegenden Ausdrucke gesammelt am Anfang dargestellt und an den jeweiligen Stellen darauf verwiesen.

Zuvor möchte ich noch ein paar Eckdaten für die Tests festlegen:

- Testumgebung ist ein Intel(R) Core(TM)2 Duo P8600 mit 2,4GHz und 4GB RAM.
- Für das kräftebasierte Verfahren wird eine Kantenauflösung von 2^6 und 20 Iterationen verwendet.
- Für das Gummiband-Routing wird im Sinne der Vergleichbarkeit ebenfalls eine Kantenauflösung von 2^6 verwendet, sowie zunächst eine Bündelung von 90%.

Die Auswahl der Konfiguration der Routingverfahren zielt dabei darauf ab, dass möglichst gute Ergebnisse bei annehmbarer Rechenzeit erzielt werden.

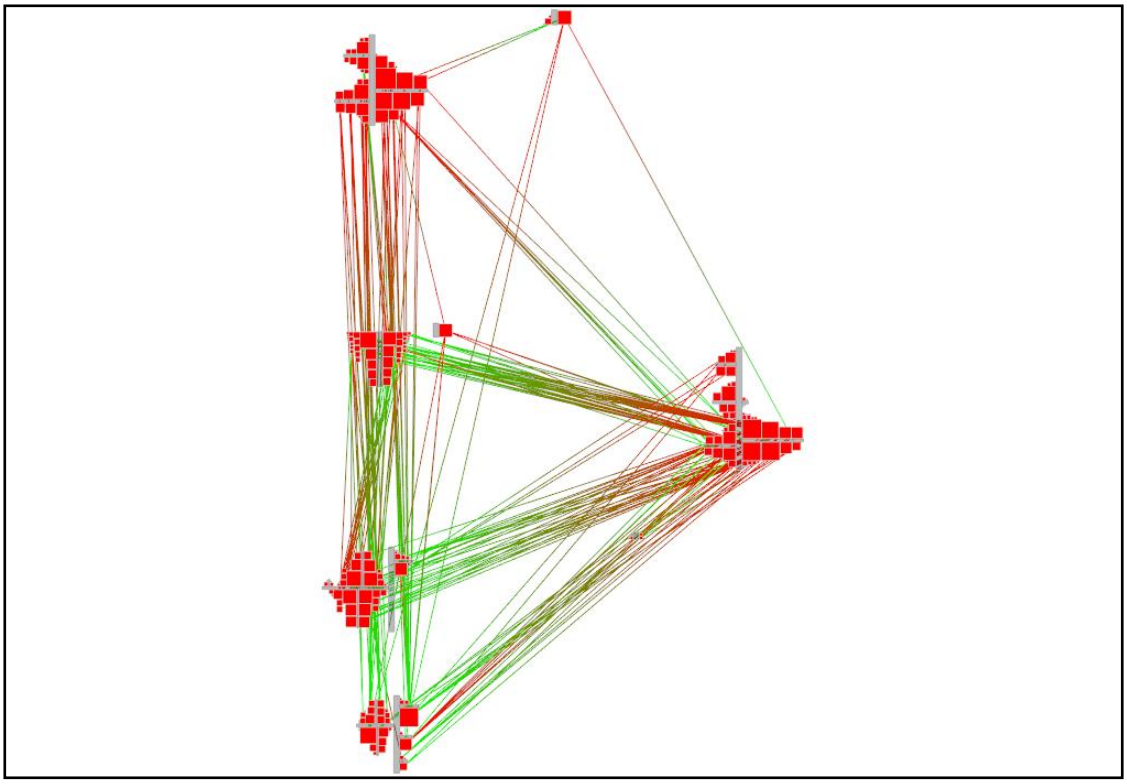

Abbildung 36: Beispielsystem 1 ohne explizites Kantenrouting.
Quelle: Ausdruck der Visualisierung im Evoviewer.

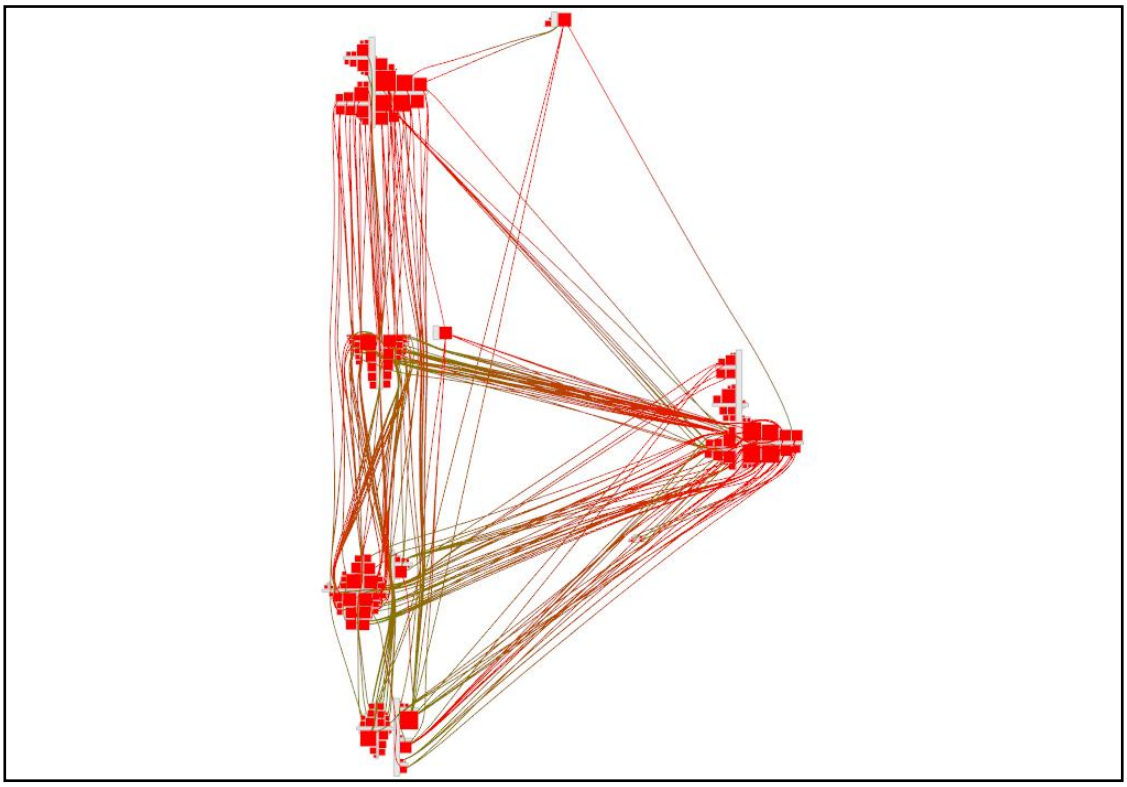

Abbildung 37: Beispielsystem 1 mit kräftebasiertem Kantenrouting.
Quelle: Ausdruck der Visualisierung im Evoviewer.

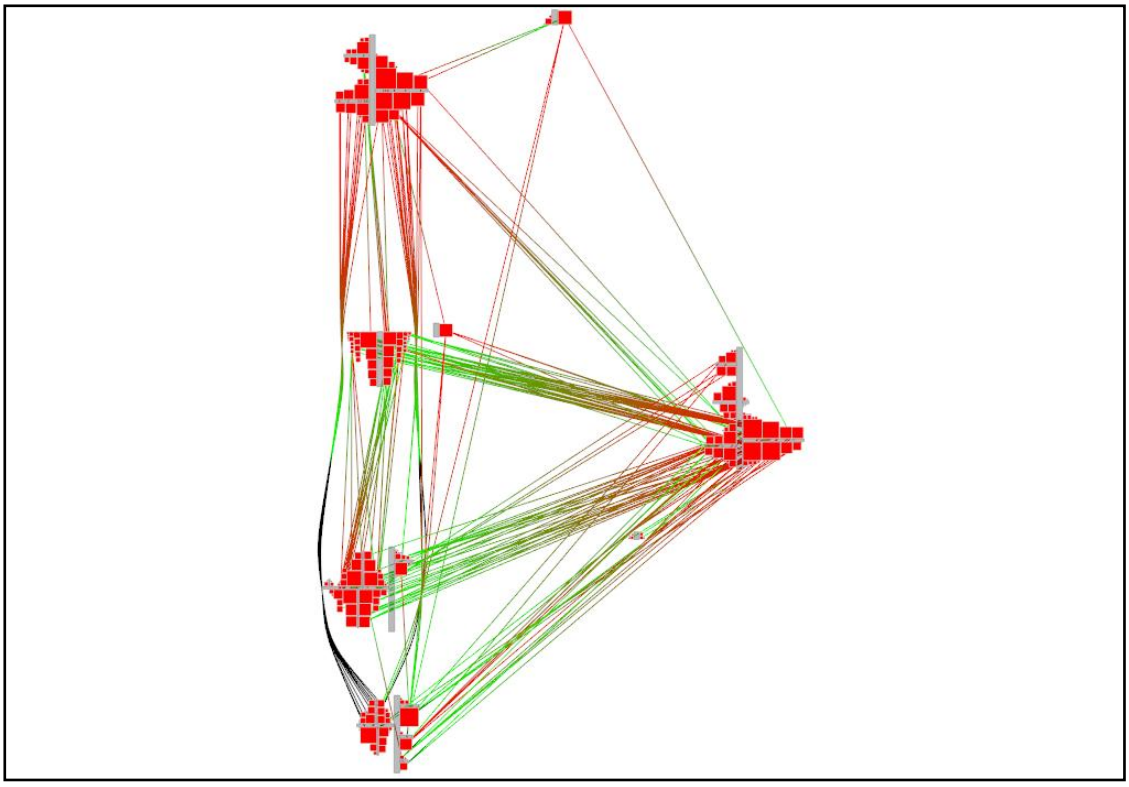

Abbildung 38: Beispielsystem 1 mit einfachem geometrischem Routing.
Quelle: Ausdruck der Visualisierung im Evoviewer.

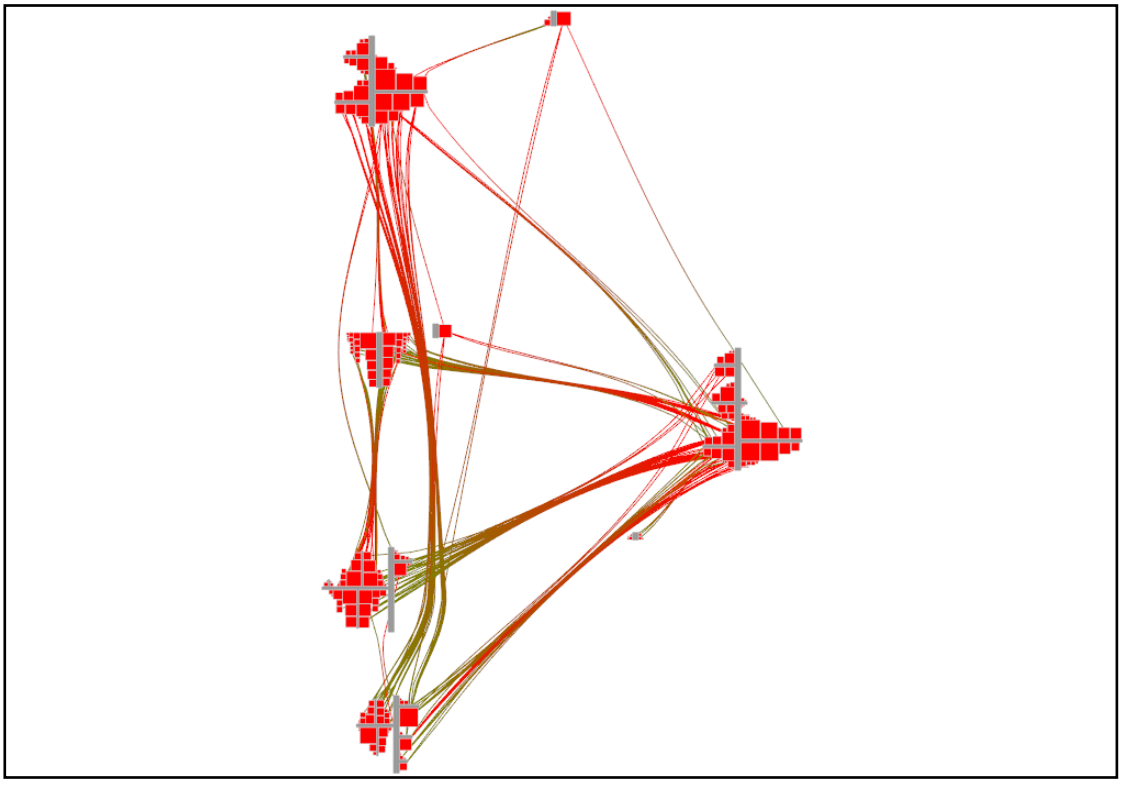

Abbildung 39: Beispielsystem 1 mit Gummiband-Routing.
Quelle: Ausdruck der Visualisierung im Evoviewer.

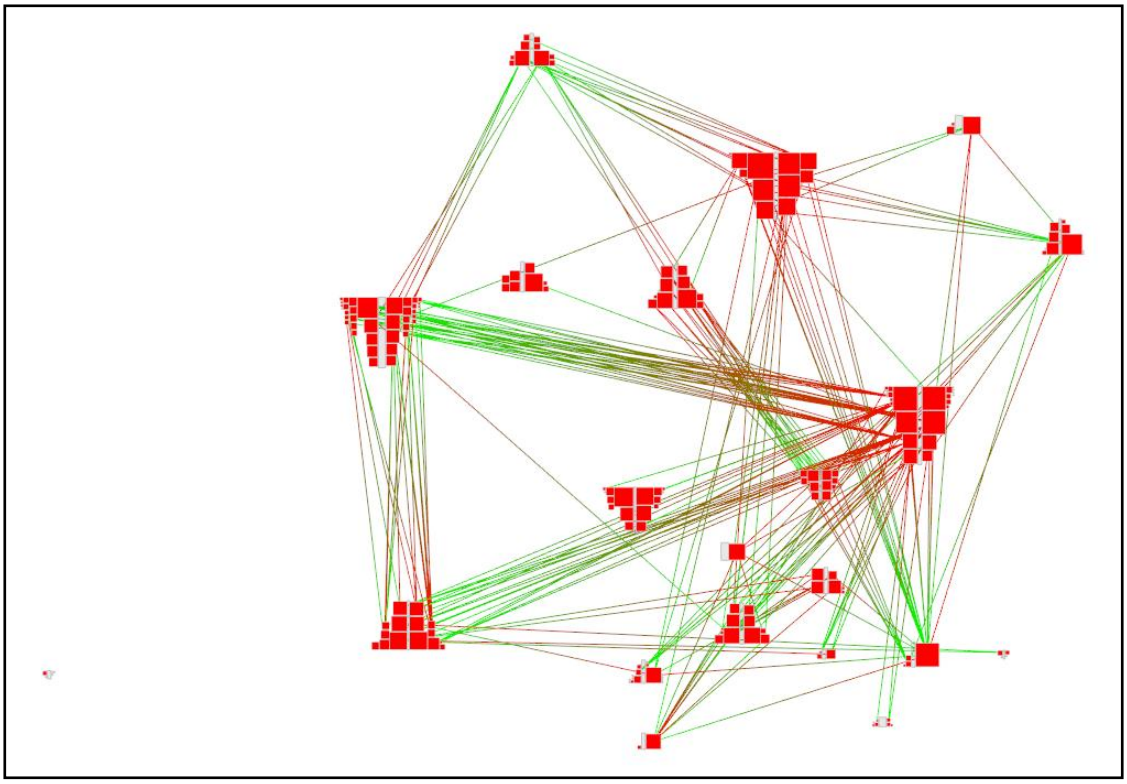

Abbildung 40: Beispielsystem 2 ohne explizites Kantenrouting.
Quelle: Ausdruck der Visualisierung im Evoviewer.

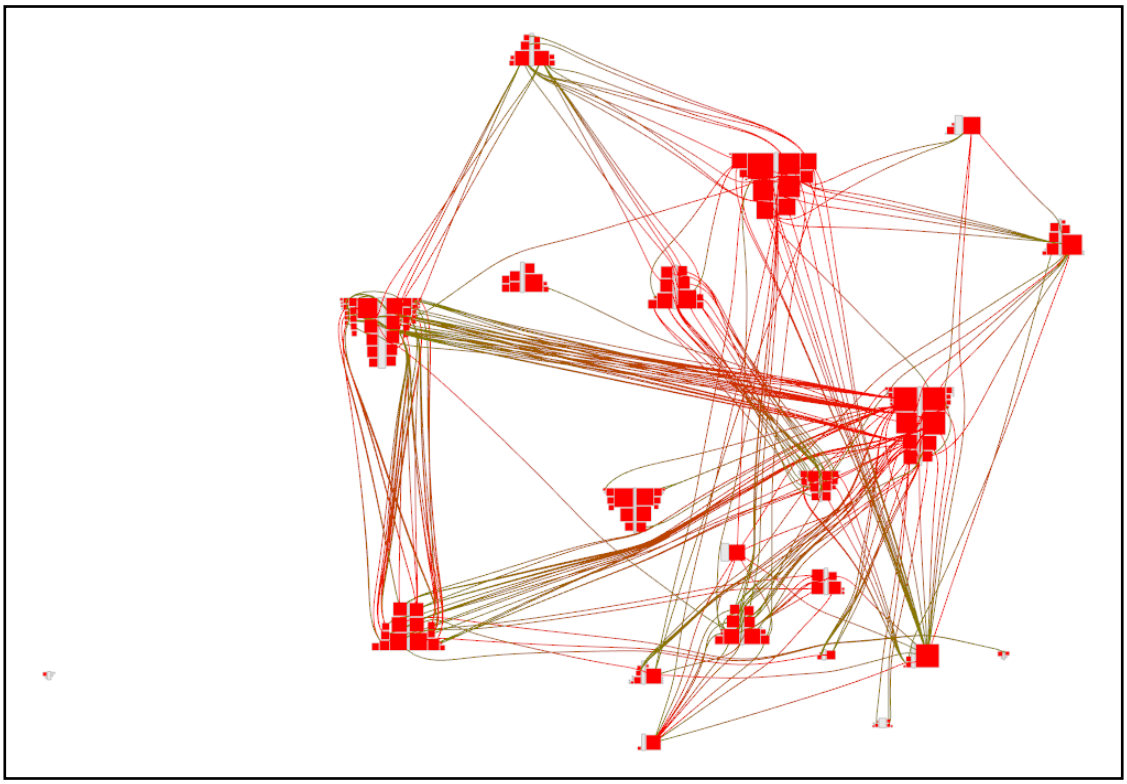

Abbildung 41: Beispielsystem 2 mit kräftebasiertem Routing.
Quelle: Ausdruck der Visualisierung im Evoviewer.

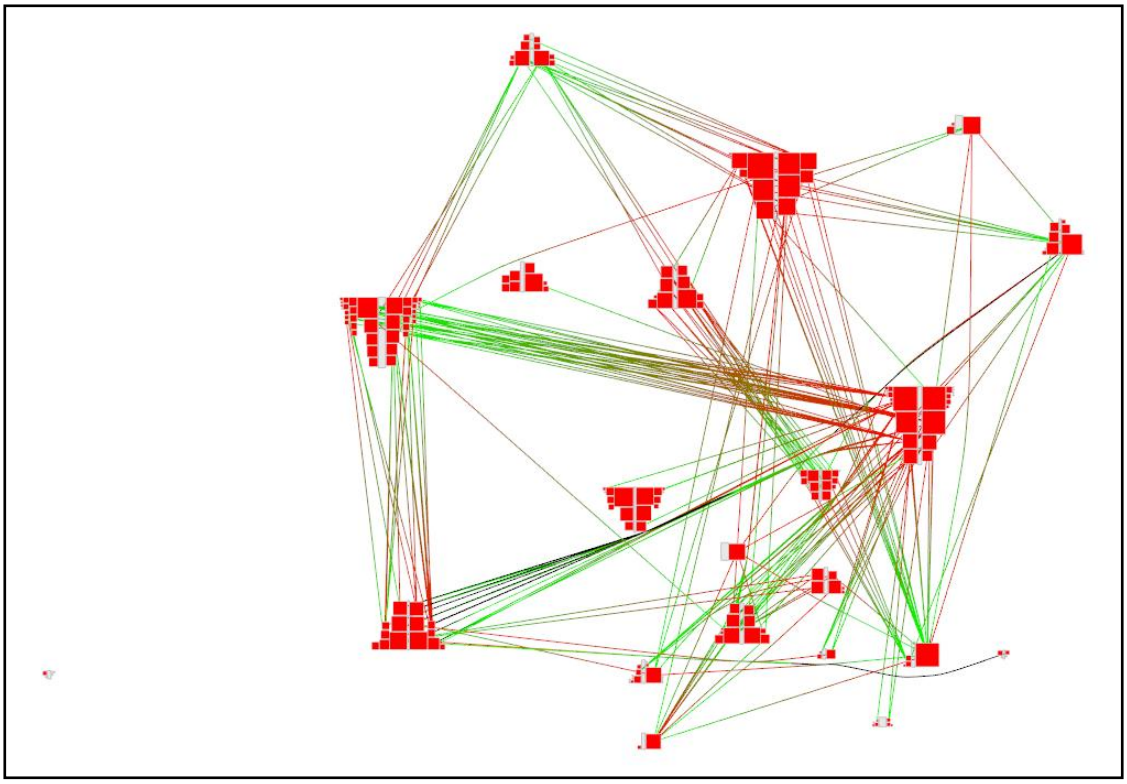

Abbildung 42: Beispielsystem 2 mit einfachem geometrischem Routing.
Quelle: Ausdruck der Visualisierung im Evoviewer.

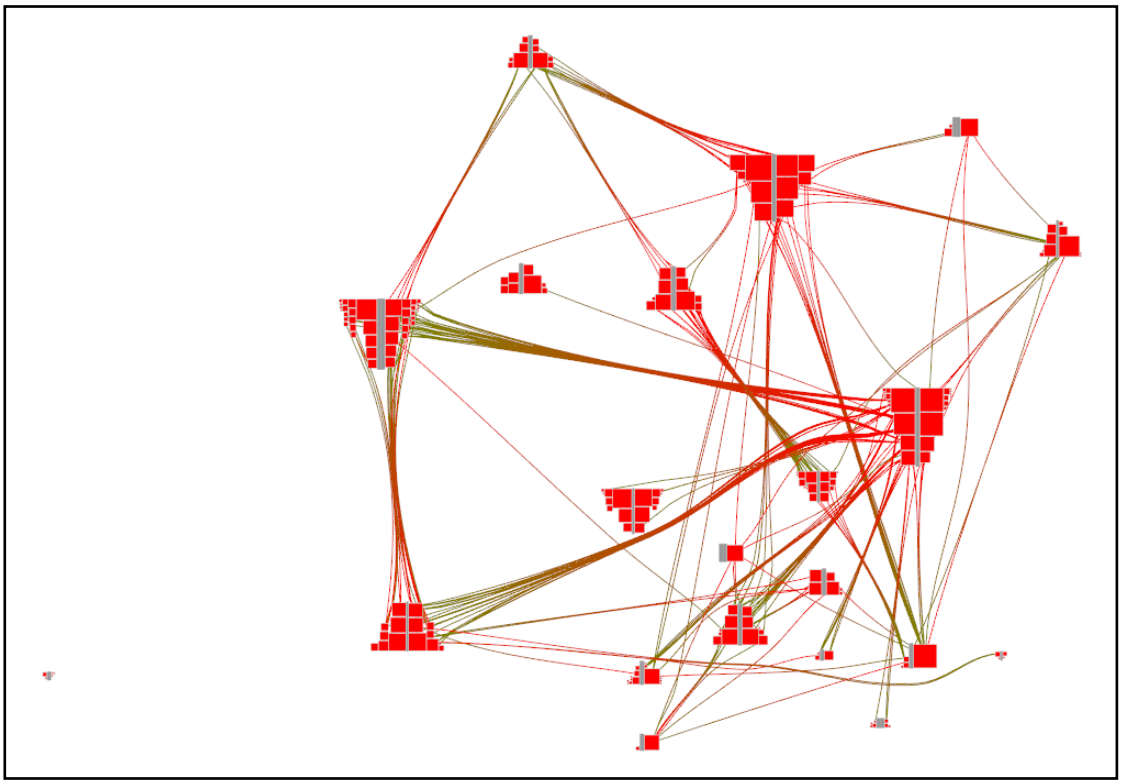

Abbildung 43: Beispielsystem 2 mit Gummiband-Routing.
Quelle: Ausdruck der Visualisierung im Evoviewer.

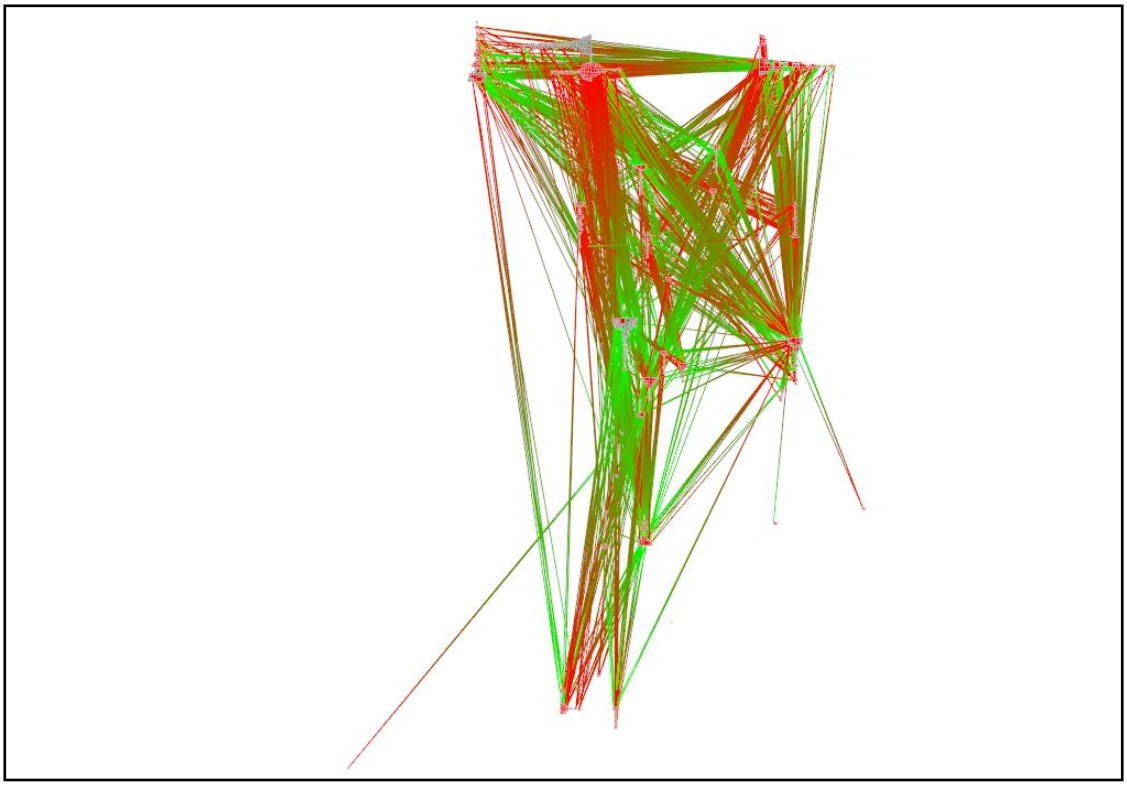

Abbildung 44: Beispielsystem 3 ohne explizites Kantenrouting.
Quelle: Ausdruck der Visualisierung im Evoviewer.

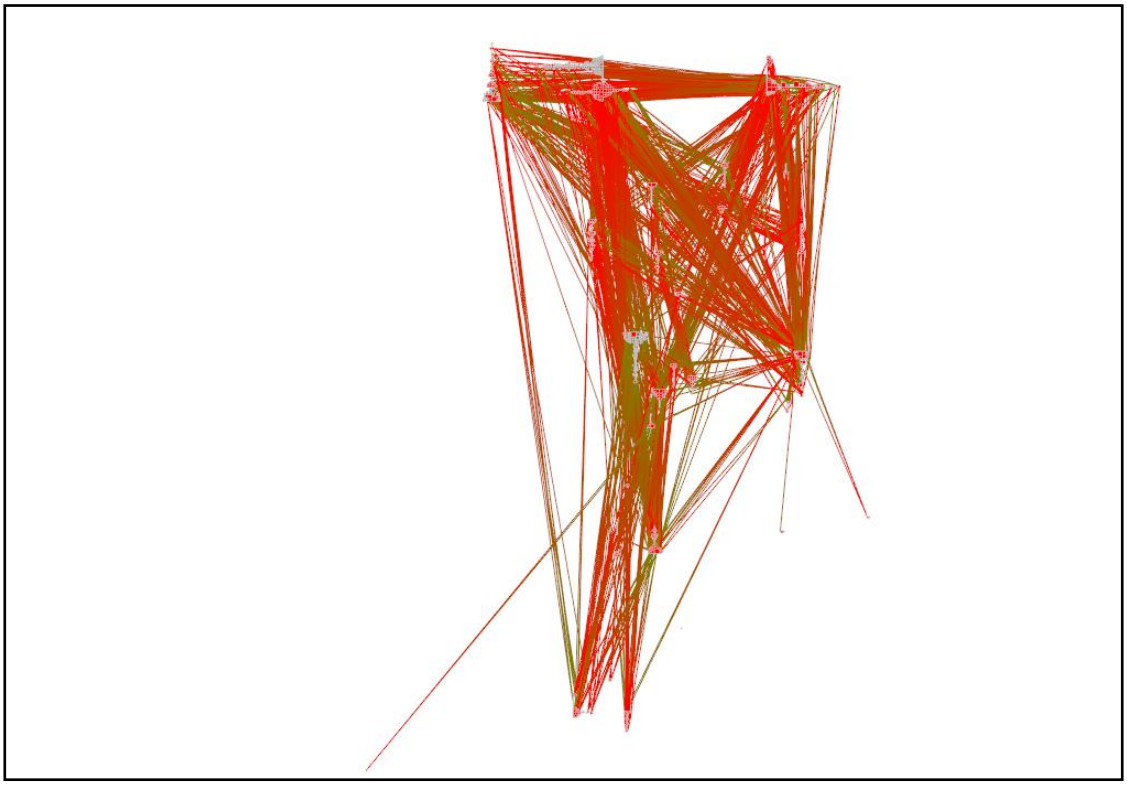

Abbildung 45: Beispielsystem 3 mit kräftebasiertem Routing.
Quelle: Ausdruck der Visualisierung im Evoviewer.

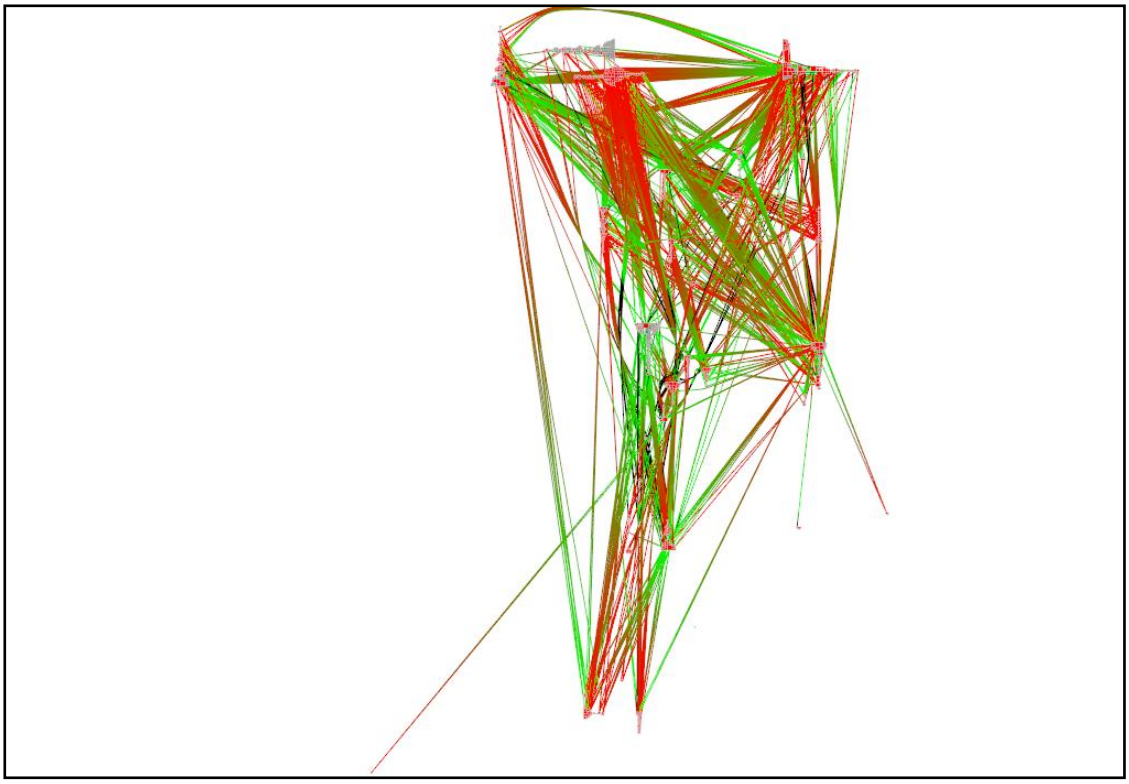

Abbildung 46: Beispielsystem 3 mit einfachem geometrischem Routing.
Quelle: Ausdruck der Visualisierung im Evoviewer.

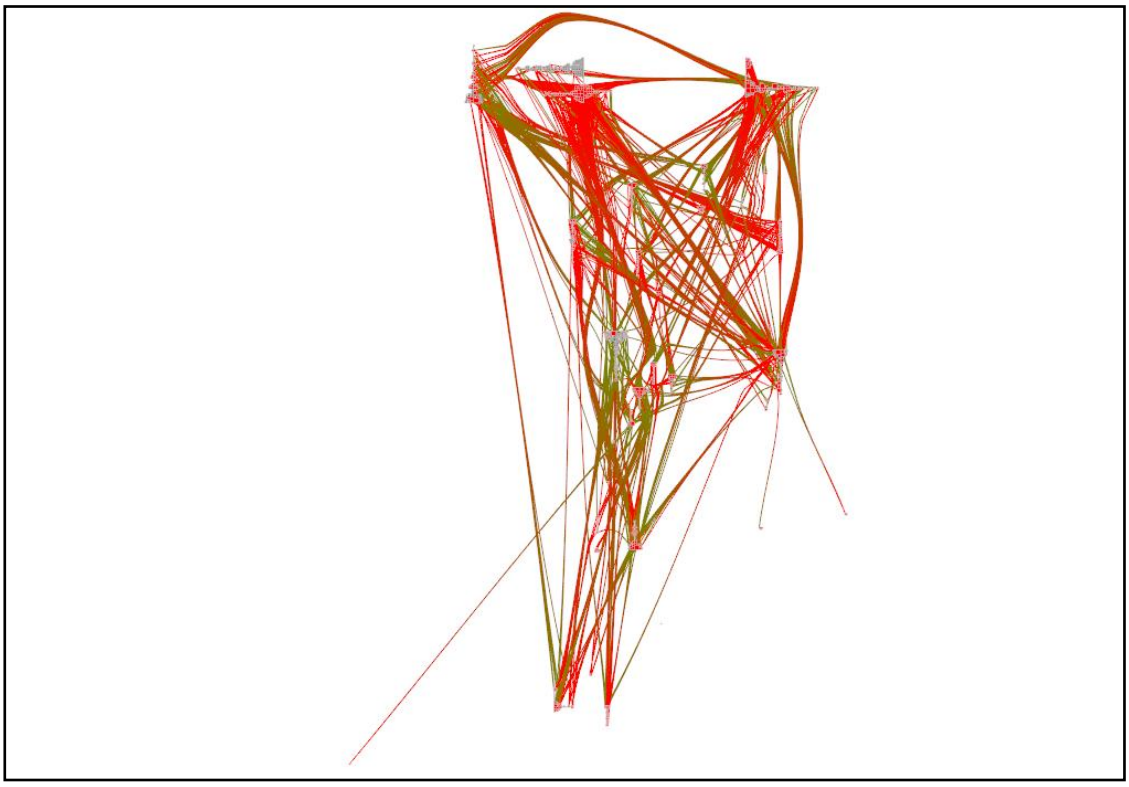

Abbildung 47: Beispielsystem 3 mit Gummiband-Routing.
Quelle: Ausdruck der Visualisierung im Evoviewer.

K1.1 Geringe Rechenzeit

Zur Bewertung der Rechenzeiten wurden die in folgender Tabelle 2 dargestellten Werte erfasst.

	Ohne explizites Kantenrouting	Kräftebasiertes Kantenrouting	Einfaches geometrisches Kantenrouting	Gummiband-Kantenrouting
Beispielsystem 1	25	29.374	23	28
Beispielsystem 2	24	36.472	28	29
Beispielsystem 3	58	880.857	66	188

Tabelle 2: Rechenzeiten zur Erzeugung der Kantenlayouts der Beispielsysteme. Es sind jeweils Mittelwerte aus 10 Messungen dargestellt, außer beim kräftebasierten Routing. Angaben in ms.
Quelle: Eigene Darstellung, Zahlen basierend auf eigenen Messungen mit dem Evoviewer.

Es ist direkt ersichtlich, dass alle Verfahren außer dem kräftebasierten nur geringe Rechenzeiten benötigen. Das kräftebasierte Verfahren hebt sich hier mit einer Rechenzeit, die etwa um den Faktor 1000 über den anderen liegt, extrem negativ ab. Das war allerdings auch nicht anders zu erwarten, da eben dieser große Nachteil des Verfahrens wesentlicher Teil der Motivation dieser Arbeit war. Darüber hinaus fällt auf, dass das Gummiband-Routing bezüglich der Rechenzeit nicht wesentlich hinter dem naiven Verfahren zurückliegt. Die gemessenen Zeiten würde ich im Sinne der Interaktivität als angemessen bezeichnen.

K1.2 Vermeidung von Stadtdurchschneidungen

Zur Bewertung der Vermeidung von Stadtdurchschneidungen wurden die in folgender Tabelle 3 dargestellten Werte erfasst.

	Ohne explizites Kantenrouting	Kräftebasiertes Kantenrouting	Einfaches geometrisches Kantenrouting	Gummiband-Kantenrouting
Beispielsystem 1	146	100	0	0
Beispielsystem 2	84	35	0	0
Beispielsystem 3	1134	792	8	30

Tabelle 3: Anzahl verbleibender Schnitte zwischen Städten und Kanten. Städte wurden hierbei als deren konvexe Hüllen abstrahiert. Die Zahlen stellen dementsprechend präziser formuliert Schnitte zwischen Kanten und Abschnitten der konvexen Hüllen dar.
Quelle: Eigene Darstellung, Zahlen basierend auf Messungen mit dem Evoviewer.

Offensichtlich können sowohl das naive Verfahren als auch das Gummiband-Routing im Gegensatz zum kräftebasierten Verfahren fast alle Stadtdurchschneidungen vermeiden. Beim Gummiband-Routing hängt der Grad der Vermeidung von Stadtdurchschneidungen dabei hauptsächlich von der Auflösung der Kanten ab. Ist die Auflösung zu klein, werden einige Stadtdurchschneidungen gar nicht erkannt, da sie zwischen zwei Stützstegen liegen. Im drit-

ten Beispielsystem ließe sich die Anzahl der Durchschneidungen beispielsweise mit einer Kantenauflösung von 2^8 auf 10 reduzieren. Absolute Überschneidungsfreiheit kann allerdings aus Gründen der Performance nicht immer garantiert werden.

K1.3 Erkennbarkeit von High-Level-Zusammenhängen

Betrachtet man zunächst die beiden kleineren Beispiele, wird deutlich, dass die Erkennbarkeit von High-Level-Zusammenhängen eine der größten Stärken des Gummiband-Routings ist. Es ist das einzige Verfahren, bei dem alle Verbindungen zwischen je zwei Städten erkennbar sind. Bei den anderen Verfahren können diese High-Level-Verbindungen nur teilweise erkannt werden, da besonders kleinere Verbindungen mit wenigen Kanten durch die Überlagerung mit anderen verloren gehen. Besonders deutlich wird diese Tatsache auch beim Betrachten des dritten Systems. Hier können fast keine Verbindungen mehr erkannt werden, am ehesten noch beim naiven Ansatz. Nur beim Gummiband-Routing sind die High-Level-Verbindungen prinzipiell noch erkennbar. Allerdings ist auch ersichtlich, dass das Verfahren bei einem System dieser Größe langsam an seine Grenzen stößt. Insbesondere durch die vielen Überschneidungen der einzelnen Bänder aufgrund deren großer Anzahl wird das Bild zunehmend unübersichtlich. Das intuitive Erkennen der Zusammenhänge auf einen Blick ist nur noch eingeschränkt möglich.

K1.4 Semantisches Routing

Expliziten Einfluss semantischer Informationen auf das Routing bietet nur das Gummiband-Routing. Nur hier werden bewusst semantisch nahe Kanten, nämlich Kanten zwischen je zwei Städten, gebündelt und nehmen beim Routing einen gemeinsamen Weg. Bei den anderen Verfahren entstehen semantische Bündel dieser Art nur dann, wenn die eigentlichen Bündelungskriterien, die geometrischer bzw. kräftebasierter Natur sind, mit den semantischen korrelieren. Beispielsweise können durch die geometrische Bedingung der ursprünglichen Verlaufsrichtung grundsätzlich ähnliche Bündel entstehen wie bei der semantischen Bündelung des Gummiband-Routings, da Kanten zwischen je zwei Städten ursprünglich eine ähnliche Verlaufsrichtung haben. Allerdings können dabei auch weitere Kanten, die zufällig die gleiche Ursprungsrichtung haben, mit in das Bündel einbezogen werden, womit der semantische Zusammenhang der Verbindung zweier Städte verloren ginge. Eine strenge Kausalbeziehung, die eben nur das Gummiband-Routing bietet, ist in Bezug auf die Darstellung dieser semantischen Zusammenhänge deutlich stabiler.

K2.1 Erkennbarkeit von Low-Level-Zusammenhängen

Es ist klar, dass Einzelkanten mit den hier zur Verfügung stehenden Mitteln nur in verhältnismäßig kleinen Systemen flächendeckend erkennbar sein können. In großen Systemen gibt es einfach zu viele Kanten, als dass man sie ohne gesonderte Hervorhebung noch im Detail erkennen könnte. Selbst in kleinen Systemen kann die Erkennbarkeit von Einzelkanten eingeschränkt sein. Besonders für sehr ausgeprägte Verbindungen zwischen zwei Städten mit vielen Einzelkanten bietet keins der vorgestellten Verfahren eine wirklich gute Lö-

sung. Am ehesten können Einzelkanten daher in Systemen mit kleinen Städten erkannt werden, wie Beispielsystem 2. Obwohl der in Abbildung 43 dargestellte Ausdruck des Gummiband-Routings verglichen mit den anderen Verfahren am schlechtesten abschneidet, zeigt sich hier auch die große Stärke der Konfigurierbarkeit der Bündelung. Regelt man die Bündelung von den in Abbildung 43 dargestellten 90% (für High-Level-Verbindungen) auf 0% herunter, bietet sich ein deutlich besseres Bild (Abbildung 48), das auch den Darstellungen der anderen Verfahren überlegen ist.

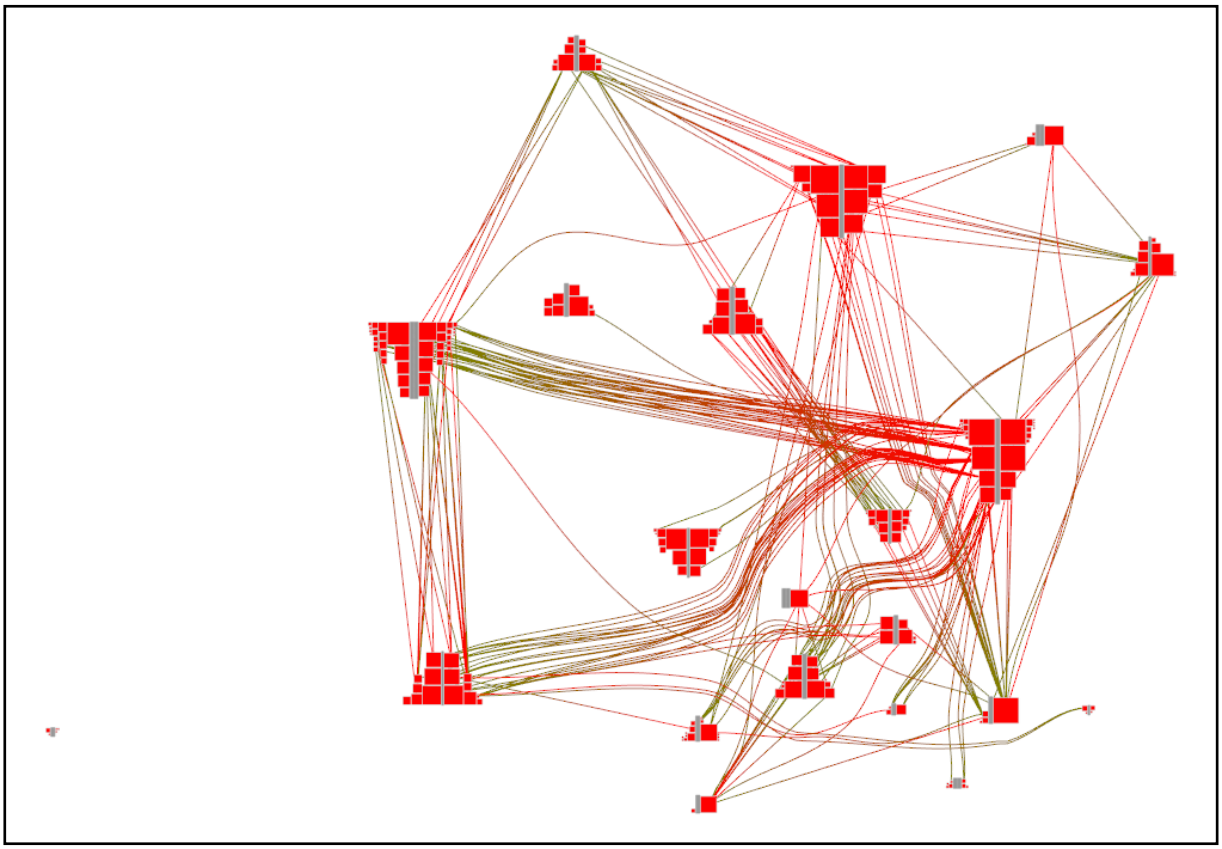

Abbildung 48: Beispielsystem 2 mit Gummiband-Routing bei 0% Bündelung für bessere Sichtbarkeit der Einzelkanten.
Quelle: Eigene Darstellung.

Es fällt außerdem auf, dass auch in diesem Zustand High-Level-Verbindungen immer noch zu erkennen sind, da Kanten die zu einer gemeinsamen High-Level-Verbindung gehören durch die Zugehörigkeit zum gemeinsamen Band ähnliche Verläufe haben.

K2.2 Vermeidung von Kantenkrümmungen

Inwieweit unnötig viele Kantenkrümmungen entstehen, hängt sowohl beim kräftebasierten als auch beim Gummiband-Routing stark von den Rahmenbedingungen ab. Beim kräftebasierten Verfahren kann der Einfluss vieler Einfluss nehmender Massepunkte in der Umgebung zu kurvigen Verläufen führen. Bei geeigneter Konfiguration treten diese Kurven allerdings nur bei sehr nahen Städten in sichtbarer Stärke auf – und bei diesen sind sie zur Umgehung der Städte ohnehin sinnvoll (Abbildung 41). Beim Gummiband-Routing tritt zunächst

pro Band eine Krümmung durch die Bündelung auf, falls diese nicht nahe 0% liegt. Diese lässt sich offensichtlich nicht wegoptimieren. Die Anzahl der Krümmungen beim Routing hängt stark von der Bündelungsstärke ab. Bei starker Bündelung entwickeln sich weniger Krümmungen als beim kräftebasierten Verfahren, da einige Städte schon durch die Bündelung umgangen werden (Abbildung 43). Bei schwacher Bündelung können dagegen besonders viele Kurven entstehen, da aufgrund der unveränderlichen Breite der Bänder (Ausnahme: Engstellen) viele Städte durch den kurvigen Verlauf umgangen werden müssen (Abbildung 48). Verstärkt wird dieser Effekt noch durch die Annäherung der Städte als deren Umkreise, da deren Größe damit zunimmt. Bei sehr unförmigen Städten können so auch Leerräume, die im Kreis liegen, durch kurvigen Verlauf umgangen werden. Außerdem könnten natürlich durch globale Betrachtungen teilweise Verläufe mit weniger Kurven gefunden werden, häufig jedoch zu Lasten der Kantenlänge. Besonders im Bereich schwacher Bündelung erfüllt das Gummiband-Routing also offensichtlich das Kriterium der Vermeidung von Kantenkrümmungen nicht optimal. Dabei ist allerdings zu bedenken, dass die hier erwähnten Effekte letztlich auf die Optimierung hinsichtlich wichtigerer Kriterien zurückgehen, insbesondere auf die geringe Rechenzeit und die Erkennbarkeit von High- und Low-Level-Verbindungen durch die Konfigurierbarkeit der Bündelung. Insofern sehe ich die nur mäßige Erfüllung dieses Kriteriums nicht als Problem an. Das naive Verfahren erzeugt durch die nicht vorhandene explizite Bündelung zwar zunächst keine Krümmungen, dafür aber beim Umgehen der Städte mehr als das Gummiband-Routing, kann hier also auch nur mittelmäßig überzeugen.

K2.3 Flache Kantenkrümmungen

Hier gilt an sich die gleiche Argumentation wie beim vorherigen Kriterium, der Vermeidung von Kantenkrümmungen. Weder das kräftebasierte noch das Gummiband-Routing erzeugen viele unnötig stark gekrümmte Kurven, wobei die Stärke der Krümmungen beim Gummiband-Routing mit abnehmender Bündelung zunimmt, was aber letztlich auf die höherwertigen Kriterien zurückgeht. Beim naiven Ansatz können die Krümmungen teilweise unnötig stark ausfallen (Abbildung 38), was allerdings auch an der fehlenden Konfigurierbarkeit des verwendeten Spline-Generators liegt.

K3.1 Räumliche Kompaktheit

Offensichtlich zeigt keins der diskutierten Verfahren übermäßig weite Umlenkungen. Den einzigen möglichen Kritikpunkt sehe ich hier wieder in der Abstrahierung der Städte als deren Umkreise, was aber – wie bereits hinreichend dargestellt – auf die Optimierung auf geringe Rechenzeit zurückzuführen ist.

K3.2 Vermeidung von Kantenkreuzungen

Hier liegt vielleicht noch der größte Schwachpunkt des Gummiband-Routings. Betrachtet man die dargestellten Ausdrucke der Beispielsysteme, ist deutlich zu erkennen, dass keins der Verfahren geeignet ist, Kreuzungen zwischen Kantenclustern zu vermeiden. Das ist auch

nicht weiter verwunderlich, denn in keinem der Algorithmen spielt die Vermeidung dieser Kreuzungen eine Rolle. Da beim Gummiband-Routing die Cluster in Form der Bänder als logische Objekte existieren, ist hier am ehesten eine Verbesserung möglich – zu Ungunsten der Rechenzeit natürlich. Ein größerer Spielraum zur Optimierung hinsichtlich dieses Kriteriums entstände, wenn die Positionen der Städte nicht als fix angenommen würden. Da deren Positionierung aber auch wieder semantische Gründe hat, ist die Optimierung in diesem Bereich kaum erreichbar, ohne semantische Zusammenhänge zu gefährden. Insgesamt kann anhand der dargestellten Ausdrucke festgestellt werden, dass die Vermeidung von Kreuzungen der Kantencluster nur selten möglich wäre, ohne Städte zu bewegen oder sehr weite Umlenkungen zu erzeugen, was nicht sinnvoll wäre. Dennoch wäre hier eine weitere Optimierung – zu Lasten der Rechenzeit – denkbar, beispielsweise durch Verlagerung des Bandes zwischen oberster und unterster Stadt in Abbildung 39 nach links. Mehr dazu im Ausblick im abschließenden Kapitel 5 dieser Arbeit.

K3.3 Möglichst kurze Kanten

Für die durchschnittliche Länge der Kanten bezogen auf die Ausgangssituation ohne explizites Kantenrouting wurden die in nachfolgender Tabelle 4 stehenden Daten erfasst.

	Ohne explizites Kantenrouting	Kräftebasiertes Kantenrouting	Einfaches geometrisches Kantenrouting	Gummiband-Kantenrouting
Beispielsystem 1	100,00	100,53	100,55	101,28
Beispielsystem 2	100,00	101,16	100,18	100,68
Beispielsystem 3	100,00	100,16	100,99	102,29

Tabelle 4: Durchschnittliche Längen der Kanten in Bezug auf die Ausgangssituation ohne explizites Kantenrouting in %.
Quelle: Eigene Darstellung, Zahlen basierend auf eigenen Messungen mit dem Evoviewer.

Zunächst fällt auf, dass die Verlängerung der Kanten im Verhältnis zur Ausgangssituation bei allen Verfahren mit maximal 2,29% sehr gering ausfällt. Dass das Gummiband-Routing im Schnitt etwas ausgeprägtere Verlängerungen aufweist als das kräftebasierte Verfahren, hängt neben der Optimierung in Hinblick auf wichtigere Kriterien vor allem damit zusammen, dass das kräftebasierte Verfahren bei weitem nicht alle Stadtdurchschneidungen vermeidet. Wenn Städte nicht umgangen werden, ist es logisch, dass kürzere Kanten entstehen. Insofern kann dieser Vergleich nicht unbedingt als negativ für das Gummiband-Routing angesehen werden.

Zusammenfassung

Hier sollen die Ergebnisse der Evaluation noch einmal zusammenfassend dargestellt werden. Dazu soll die nachstehende Tabelle 5 einen Überblick geben. Dabei wird an dieser Stelle der Übersichtlichkeit halber nur noch auf das ursprüngliche kräftebasierte Verfahren und das Gummiband-Routing eingegangen, da bereits mehrfach festgestellt werden konnte, dass

das naive Verfahren in Sachen Lesbarkeit noch deutliche Schwächen zeigt und ohnehin mehr als Entwicklungsschritt in Richtung des zweiten Verfahrens zu verstehen war.

Kriterium	**Kräftebasiertes Kantenrouting**	**Gummiband-Kantenrouting**
K1.1 Geringe Rechenzeit	Sehr schlechte Rechenzeiten im Minutenbereich.	Sehr gute Rechenzeiten im Bereich von Millisekunden bis Sekunden.
K1.2 Vermeidung von Stadtdurchschneidungen	Viele Stadtdurchschneidungen bleiben erhalten (hier: zwischen 42 und 70%).	Nur wenige Stadtdurchschneidungen bleiben erhalten (hier: zwischen 0 und 3%).
K1.3 Erkennbarkeit von High-Level-Zusammenhängen	High-Level-Zusammenhänge sind nur teilweise zu erkennen.	High-Level-Zusammenhänge sind fast immer gut zu erkennen.
K1.4 Semantisches Routing	Kein Einfluss semantischer Informationen.	Expliziter Einfluss semantischer Informationen.
K2.1 Erkennbarkeit von Low-Level-Zusammenhängen	Low-Level-Zusammenhänge sind nur teilweise zu erkennen.	Low-Level-Zusammenhänge sind nur teilweise zu erkennen, bei geringer Bündelung besser als beim kräftebasierten Verfahren.
K2.2 Vermeidung von Kantenkrümmungen	Mäßig gut, abhängig von Rahmenbedingungen und Konfiguration.	Mäßig gut, abhängig von Rahmenbedingungen und Konfiguration.
K2.3 Flache Kantenkrümmungen	Mäßig gut, abhängig von Rahmenbedingungen und Konfiguration.	Mäßig gut, abhängig von Rahmenbedingungen und Konfiguration.
K3.1 Räumliche Kompaktheit	Keine unnötige Platzverschwendung.	Keine unnötige Platzverschwendung.
K3.2 Vermeidung von Kantenkreuzungen	Kantenkreuzungen werden nur zufällig vermieden.	Kantenkreuzungen werden nur zufällig vermieden.
K3.3 Möglichst kurze Kanten	Nur geringfügige Kantenverlängerung.	Nur geringfügige Kantenverlängerung.

Tabelle 5: Zusammenfassung der Ergebnisse der Evaluation.
Quelle: Eigene Darstellung.

Insgesamt ist festzustellen, dass das Gummiband-Routing dem kräftebasierten Verfahren in der Erfüllung der aufgestellten Zielkriterien besonders bei den Kriterien der obersten Wichtigkeitsklasse deutlich überlegen ist. Mit dem Gummiband-Kantenrouting erzeugte Softwarelandschaften sind trotz erheblich geringerer Rechenzeit dennoch besser lesbar als mit dem kräftebasierten Verfahren erzeugte.

5. Zusammenfassung und Ausblick

In diesem Kapitel soll diese Arbeit im ersten Abschnitt noch einmal zusammenfassend kurz dargestellt werden. Im zweiten Teil soll daraufhin ein Ausblick auf Weiterentwicklungsmöglichkeiten gegeben werden, die im Rahmen dieser Arbeit nicht mehr umgesetzt werden konnten.

5.1 Zusammenfassung

Ziel dieser Arbeit war es, ein geometrisches Kanten-Routingverfahren als Alternative zum vorhandenen kräftebasierten Verfahren zu entwickeln, das sich durch gute optische Qualität bei geringer Rechenzeit auszeichnet. Zur Erreichung dieses Ziels wurden zunächst verschiedene bereits entwickelte Ansätze auf ihre Eignung hin analysiert. Dabei wurde festgestellt, dass alle Ansätze Schwächen zeigten, aufgrund derer sie als ungeeignet angesehen werden können. Allerdings konnten aus den Problemen dieser Verfahren sowie auch aus ihren Stärken wichtige Erkenntnisse für die Entwicklung eines eigenen Verfahrens gewonnen werden. Basierend vor allem auf diesen Erkenntnissen und den direkten Forderungen aus der Zielstellung dieser Arbeit wurden daraufhin Zielkriterien definiert, denen das zu entwickelnde Verfahren genügen sollte. Die so definierten und gewichteten Kriterien dienten daraufhin als Leitlinien bei der Entwicklung und Optimierung des Verfahrens. Als besonders wichtig wurden dabei herausgestellt:

- Geringe Rechenzeit,
- Vermeidung von Stadtdurchschneidungen,
- Erkennbarkeit von High-Level-Zusammenhängen,
- Einfluss semantischer Informationen auf das Routing.

Da die geringe Rechenzeit als von so zentraler Bedeutung eingeordnet worden war, wurde daraufhin zunächst ein naives Verfahren als Referenz entwickelt, das vor allem die Kernforderungen der Zielstellung erfüllen sollte: Geringe Rechenzeit und Vermeidung von Stadtdurchschneidungen. Dabei wurde vor allem festgestellt, dass die Bündelung von Kanten von hoher Bedeutung ist, um High- oder Low-Level-Zusammenhänge erkennen zu können, und daher nicht nur als Nebeneffekt auftreten sollte. Da sich beide Kriterien gegenseitig behindern, wurde die Konfigurierbarkeit dieser Bündelung als zentraler Ansatzpunkt für die Entwicklung eines zweiten Verfahrens erkannt.

Bei der Entwicklung des zweiten Verfahrens wurde deshalb die Bündelung als separater Schritt vor dem eigentlichen Routing im Sinne der Wegsuche zwischen den Städten behandelt. Außerdem wurde die entscheidende Einschränkung getroffen, dass nur Kanten gebündelt werden, die zu dem gleichen Städtepaar gehören, um somit semantisch aussagekräftige Bündel entstehen zu lassen. Die relative Breite der Bündel in Bezug zur Breite der zugehörigen Stadtumkreise wurde dabei variabel gehalten. Diese kann vom Nutzer jederzeit geändert werden, um den Fokus der Darstellung bewusst auf High- oder Low-Level-Zusammenhänge zu setzen. Diese Zusammenfassung von Kanten zwischen je zwei Städten zu einem Band führt beim nachfolgenden Routing schließlich dazu, dass semantisch verwandte Kanten ähn-

liche Wege nehmen und dass erheblich weniger Routingberechnungen durchgeführt werden müssen, was der rechentechnischen Performance deutlich zu Gute kommt. Die Interpretation dieser gemeinsamen Hülle der Kanten als Gummiband hat sich als sehr vorteilhaft erwiesen, weil eben die damit assoziierte Flexibilität dieser Hülle sie beim Routing auch alle Engstellen überwinden lässt.

In der abschließenden Evaluation konnte schließlich festgestellt werden, dass die entwickelte Lösung bei der Erfüllung der Zielkriterien oberster Priorität dem kräftebasierten Verfahren deutlich überlegen ist und bei niedriger wertigen Kriterien zumindest gleichwertig bewertet werden kann. Insofern konnte das Ziel der Arbeit also erreicht werden, da das entwickelte Verfahren optisch bessere, das heißt vor allem klarer lesbare, Ergebnisse liefert und dabei mit erheblich geringerer Rechenzeit auskommt, die bei den dargestellten Beispielsystemen etwa um den Faktor 1000 niedriger ausfiel.

5.2 Ausblick

Der wichtigste Schritt bei der Entwicklung des Gummiband-Routings war die strikte Trennung der Bündelungsproblematik vom eigentlichen Routing im Sinne der Wegsuche zwischen den Städten. Genau an diesem Punkt bieten sich auch die größten Weiterentwicklungsmöglichkeiten an. Beispielsweise könnte die Wegsuche erneut geteilt werden in eine globale Komponente, die für jedes Band einen guten Weg sucht und eine nachgelagerte lokale Komponente, die lokale Probleme (z.B. Überschneidungen) löst. Ziel der globalen Komponente könnte dann vor allem auch sein, Kreuzungen der Bänder zu vermeiden, um die Übersichtlichkeit weiter zu erhöhen. Zur Vermeidung der Kreuzungen der Bänder würde es dabei genügen, jedes Band als Einzelkante zu interpretieren, womit sich die zusätzliche Rechenzeit in Grenzen halten würde. Für diesen Routingschritt könnte ein beliebiges Routingverfahren verwendet werden. Mit dem Teile-und-Herrsche-Prinzip könnte auch noch ein geeignetes unabhängiges Verfahren für das Kantenrouting innerhalb von Städten gefunden werden, das in der vorliegenden Arbeit nicht thematisiert wurde.

Auch die Bündelungstechnik an sich bietet noch Raum für Variationen und Erweiterungen. So könnten beispielsweise Alternativen zur Parabelform ausprobiert werden, etwa ein schnelleres Zusammenziehen des Bandes in Stadtnähe, sodass das Band mittig einen geraden Verlauf annimmt. Ebenfalls denkbar ist, die Aufhängung der Bänder an den Städten drehbar zu gestalten. Damit würden starke Umlenkungen vermieden, die entstehen können, wenn eine Stadt direkt vor der Ausgangs- oder Zielstadt im Bandverlauf liegt. Außerdem könnte man die Flexibilität der Bündelung noch erweitern, indem man auch Bündelungen von weniger als 0% zuließe. Eine negative Bündelung, d.h. eine Dehnung über die Ursprungsbreite des Bandes hinaus kann Sinn machen, um bei sehr stark ausgeprägten Verbindungen zwischen zwei Städten Einzelkanten besser erkennen zu können.

Generell sind verschiedene Maßnahmen denkbar, um vor allem die Grenzen der hier vorgestellten Lösung zu erweitern. Insbesondere bei sehr großen Systemen kann es sinnvoll sein, einzelne Bündel zu größeren Bündeln zusammenzufassen, auch wenn dann der Vorteil der semantischen Bündelung verloren ginge. Außerdem wäre auch die Anwendung verschiede-

ner Hervorhebungstechniken möglich, wie etwa farbliche Hervorhebungen oder Zoomtechniken (insbesondere Fisheye).

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Balzer, Michael; Noack, Andreas; Deussen, Oliver; Lewerentz, Claus (2004): Software Landscapes. Visualizing the Structure of Large Software Systems, In: VisSym 2004, Symposium on Visualization, S. 261-266, Eurographics Association, o.O.

[2] Balzert, Helmut (2009): Lehrbuch der Softwaretechnik. Basiskonzepte und Requirements Engineering, 3. Auflage, Heidelberg.

[3] Caserta, Pierre; Zendra, Olivier; Bodénès, Damien (2011): 3D Hierarchical Edge Bundles to Visualize Relations in a Software City Metaphor, In: 6th IEEE International Workshop on Visualizing Software for Understanding and Analysis, o.S., o.O.

[4] Cui, Weiwei; Zhou, Hong; Qu, Huamin; Wong, Pak Chung; Li, Xiaoming (2008): Geometry-Based Edge clustering for Graph Visualization, In: IEEE Transactions on Visualization and Computer Graphics, Vol. 14, No. 6, S. 1277-1284, o.O.

[5] Dugerdil, Philippe; Sazzadul, Alam (2008): Execution Trace Visualization in a 3D Space, In: Fifth International Conference on Information Technology: New Genereations, S. 38-43, o.O.

[6] Fekete, Jean-Daniel; Wang, David; Dang, Niem; Aris, Aleks; Plaisant, Catherine (o.J.): Overlaying Graph Links on Treemaps, o.O.

[7] Ghoniem, Mohammad; Fekete, Jean-Daniel; Castagliola, Philippe (2004): A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations, In: IEEE Symposium on Information Visualization 2004, S. 17-24, Austin.

[8] Holten, Danny (2006): Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, In: IEEE Transactions on Visualization and Computer Graphics, Vol. 12, No. 5, o.S., o.O.

[9] Holten, Danny; van Wijk, Jarke J. (2009): Force-Directed Edge Bundling for Graph Visualization, In: Eurographics/IEEE-VGTC Symposium on Visualization, o.S., Oxford, Malden.

[10] Junghans, Martin (2008): Visualization of Hyperedges in Fixed Graph Layouts, Cottbus.

[11] Phan, Doantam; Xiao, Ling; Yeh, Ron; Hanrahen, Pat; Winograd, Terry (o.J.): Flow Map Layout, o.O.

[12] Qu, Huamin; Zhou, Hong; Wu, Yingcai (2007): Controllable and Progressive Edge Clustering for Large Networks, In: Kaufmann, M.; Wagner, D. (Eds.): GD 2006, LNCS 4372, S. 399-404, Berlin, Heidelberg.

[13] Tamassia, Roberto; Di Battista, Giuseppe; Batini, Carlo (1988): Automatic Graph Drawing and Readability of Diagrams, in: IEEE Transactions on Systems, MAN, and Cybernetics, Vol. 18, No. 1: S. 61-79, o.O.

[14] Wettel, Richard (2010): Software Systems as Cities, Lugano.

Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Literatur und Hilfsmittel angefertigt habe. Alle verwendeten Hilfsmittel und Quellen sind im Literaturverzeichnis vollständig aufgeführt und die aus den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht. Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Marcus Uhlig

Cottbus, 03.01.2012