

■ fakultät für informatik

Bachelor-Arbeit

MetroMap-Layout-konforme
Visualisierung von
Höchstspannungsnetzen

Robin Möhring

9. September 2017

Gutachter:

Prof. Dr. Heinrich Müller
M.Sc. Dominic Siedhoff

Lehrstuhl Informatik VII
Graphische Systeme
TU Dortmund

Inhaltsverzeichnis

Mathematische Notation	1
1 Einleitung	3
1.1 Motivation und Hintergrund	3
1.2 Aufbau der Arbeit	3
2 Graphbasierte Modellierung des Höchstspannungsnetzes	7
2.1 Höchstspannungsnetz	7
2.2 Graph	13
2.3 Vom Höchstspannungsnetz zum Graphen	16
2.4 MetroMap Layout Problem	17
3 Spring-Embedder	25
3.1 Spring-Embedder - allgemein	25
3.2 Spring-Embedder - Vorgehen	28
4 Anwendung des Spring-Embedders auf den modellierten Graphen	39
4.1 Vergleich des vom Algorithmus erhaltenen Layout mit dem einer MetroMap	39
4.2 Nachteile des Spring-Embedders auf der Anwendung des Graphen	40
4.3 Anpassung des Spring-Embedders zur Anwendung auf das modellierte Höchstspannungsnetz	43
4.4 Bildung nahezu paralleler Leiterseile mittels dem Spring-Embedder	48
4.5 Einbettung der Knoten in Felder	52
4.6 Evaluierung und Fazit	53
A Weitere Informationen	59
Abbildungsverzeichnis	62
Algorithmenverzeichnis	63

Quellcodeverzeichnis **65**

Literaturverzeichnis **67**

Mathematische Notation

Notation	Bedeutung
\mathbb{N}	Menge der natürlichen Zahlen $1, 2, 3, \dots$
\mathbb{R}	Menge der reellen Zahlen
\mathbb{R}^d	d -dimensionaler Raum
$\mathcal{M} = \{m_1, \dots, m_N\}$	ungeordnete Menge \mathcal{M} von N Elementen m_i
$\mathcal{M} = \langle m_1, \dots, m_N \rangle$	geordnete Menge \mathcal{M} von N Elementen m_i
\mathbf{v}	Vektor $\mathbf{v} = (v_1, \dots, v_n)^T$ mit N Elementen v_i
$v_i^{(j)}$	i -tes Element des j -ten Vektors
\mathbf{A}	Matrix \mathbf{A} mit Einträgen $a_{i,j}$
$G = (V, E)$	Graph G mit Knotenmenge V und Kantenmenge E

1 Einleitung

1.1 Motivation und Hintergrund

Das Höchstspannungsnetz, welches sich über Deutschland erstreckt ist riesig und besteht aus tausenden von Strommasten und Verbindungen. Da ist sehr vom Nutzen eine Karte zu haben, die einige Verbindungen und Orte verschiebt um eine weitaus bessere und übersichtlichere Darstellung zu erhalten. Das ist genau die Methode einer MetroMap, den Verlust genauer Daten um sicherzustellen, dass die wichtigen Verbindungen und Standorte schnell und sicher erkannt werden.

Die Erstellung einer MetroMap von Netzen, findet bisher fast ausschließlich manuell statt. Das heißtt, die dafür Zuständigen drehen und verschieben die Knoten auf der eigentlichen Karte, bis es eine, für sie schöne, Darstellung des Netzes wird. Es gibt bereits einige sehr gute Lösungsansätze für eine automatische Anfertigung einer MetroMap. Diese beschränken sich jedoch auf die normalen Bus- und Bahnnetzen. Die Modellierung des Höchstspannungsnetzes und dessen Darstellung als MetroMap erfordert eine Anpassung der momentanen Ansätze. Demnach wird es in dieser Arbeit gezeigt, wie es möglich ist aus einem Teil eines Höchstspannungsnetzes eine MetroMap ähnliche Darstellung zu erhalten, die die Eigenheiten, des Höchstspannungsnetzes berücksichtigt.

1.2 Aufbau der Arbeit

In dieser Arbeit geht es darum, die Methoden einer MetroMap für Bus- und Bahnverbindungen zu nutzen, um eine angepasste MetroMap für das deutsche Höchstspannungsnetz zu erhalten. Dafür wird der Sachverhalt eines Höchstspannungsnetzes kurz erläutert. Anschließend werden die benötigten Sachen im Bezug des Graphen definiert, die gebraucht werden um aus dem Höchstspannungsnetz einen repräsentierenden Graphen zu erhalten.

Anschließend wird aus einem Teil des Höchstspannungsnetzes ein Graph gewonnen, der auch die Eigenheiten des Netzes modelliert. Dies sind besonders die Leiterseile. Das wird auch der größte und wichtigste Unterschied im Vergleich zu einer normalen MetroMap sein, welcher sich doch deutlich von einer MetroMap eines Bus- oder Bahnnetzes unterscheidet. Auf diesen Graphen ist es nun möglich einen in Kapitel 3 vorgestellten Algorithmus zur Bildung einer MetroMap konformen Darstellung anzuwenden. Dieser Algorithmus wird ein sogenannter Spring-Embedder sein. Dieser beruht auf ein physikalisches Verfahren, welches auf den Graphen modelliert wird, um ein deutlich schöneres Layout zu erhalten. Dieser Algorithmus wurde benutzt, da er sehr leicht erweiterbar ist und sich gut für Anpassungen eignet.

Dieser Algorithmus wird nun auf den gewonnenen Graphen aus Kapitel 2, welcher den Teil des Höchstspannungsnetzes modelliert, angewendet. Darauf wird besonders auf die Probleme, die die Eigenheiten des Höchstspannungsnetzes mit sich bringen, eingegangen und versucht diese zu lösen. Ebenso werden weitere Erweiterungsmöglichkeiten erläutert und erklärt. Letztendlich werden die erarbeiteten Lösungsansätze evaluiert und beurteilt.



Abbildung 1.1: Karte des deutschen Höchstspannungsnetzes



Abbildung 1.2: MetroMap der Bahnverbindungen in Singapur

2 Graphbasierte Modellierung des Höchstspannungsnetzes

2.1 Höchstspannungsnetz

Die vorliegenden Ausführungen orientieren sich am Lehrerfachheft "Übertragung und Verteilung der elektrischen Energie". Um elektrische Energie über große Distanzen zu transportieren werden Höchstspannungsnetze benutzt. Von einem Kraftwerk ausgehend wird versucht möglichst viele Haushalte, Industrie- und Gewerbebetriebe zu erreichen. Davon ausgehend wird auch der Standort der meisten Kraftwerke bestimmt. Einige Kraftwerke lassen sich nur an bestimmten Standorten errichten, zum Beispiel ein Wasserkraftwerk muss an einem Fluss oder Staudamm errichtet werden. So ist es oft nicht möglich genug Verbraucher zu erreichen, sodass es Mittel bedarf den elektrischen Strom auch über weite Strecken hinweg zu transportieren.

Die Generatoren der modernen Kraftwerke erzeugen eine Spannung von $10500V$, $21000V$ oder $27000V$ [1]. Die Höhe dieser Spannung ist bestimmt durch die Größe bzw. die Leistungsfähigkeit des Kraftwerks und somit von der Nennleistung des Generators.

Da der überwiegende Teil der elektrischen Energie in Wärmekraftwerken erzeugt wird und diese mit einer Generatorleistung von 600 bis $1300MW$, bedeutet dies, dass Ströme zwischen $15000A$ und $30000A$ abgegeben werden müssen. Das ist jedoch weder aus technischen noch wirtschaftlichen Gründen für einen Transport über lange Distanzen lohnenswert, da es entweder sehr großen Leiterquerschnitte oder sehr große Stromverluste zur Folge hätte.

Es kann die gleiche Leistung P auch mit weniger Strom I und einer erhöhten Spannung U erreicht werden, denn die Beziehung lautet:

$$P = U \cdot I \quad (2.1)$$

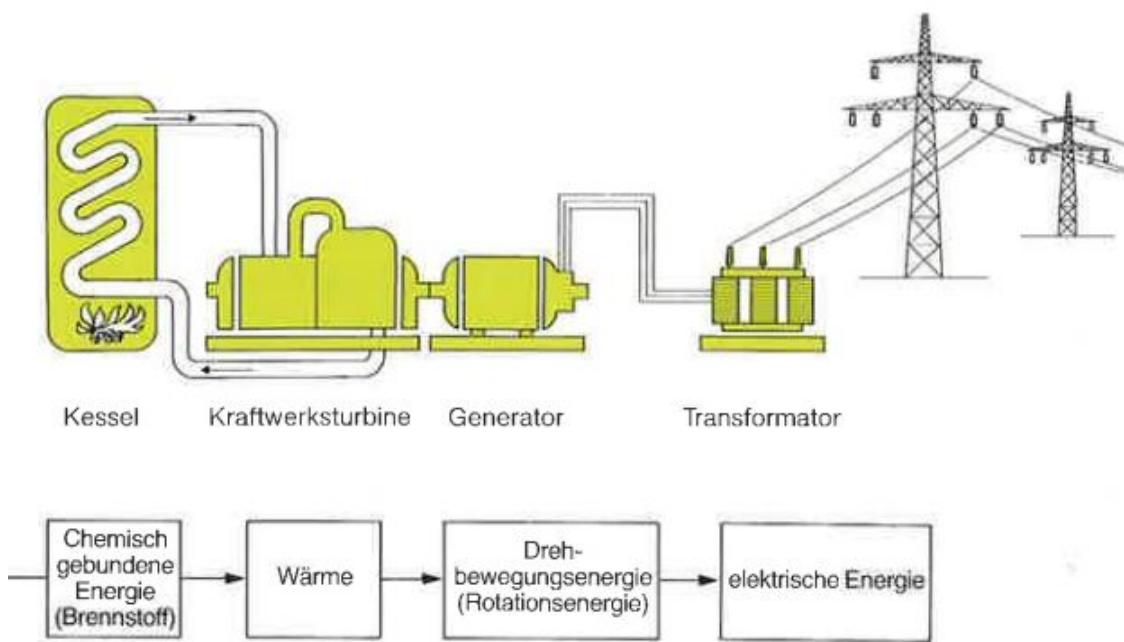


Abbildung 2.1: Prinzip der Stromerzeugung[1]

Diese Eigenschaft wird ausgenutzt und das führt dazu, dass die Generatorenspannung bereits direkt am Kraftwerk durch einen Transformator in eine höhere Spannung umgeformt wird. Dadurch wird die elektrische Leistung mit kleineren Stromstärken über die Netze geleitet. Nur die Übertragung in höheren Spannungen ermöglicht es die elektrische Energie effizient zu übertragen. Die Anpassungen führen zu einem sehr geringen Gesamtverlust von etwa 5% der erzeugten Energie.

Die erforderlichen Leitungen um ein effizientes Netz sicherzustellen werden in verschiedenen Ebenen anhand ihrer Betriebsspannung eingeteilt:

- Höchstspannungsleitungen mit Betriebsspannungen über 150000V
- Hochspannungsleitungen mit Betriebsspannungen über 60000V
- Mittelspannungsleitungen mit Betriebsspannungen über 1000V
- Niederspannungsleitungen mit Betriebsspannungen bis 1000V.

In Deutschland werden die Höchstspannungsleitungen mit 380000V oder mit 220000V betrieben. Sie sind zuständig für die überregionale Übertragung. Von den Kraftwerken werden mittels Höchstspannungsleitungen Umspannanlagen, die in der Nähe der Verbraucherschwerpunkte liegen verbunden. In den Umspannanlagen findet eine

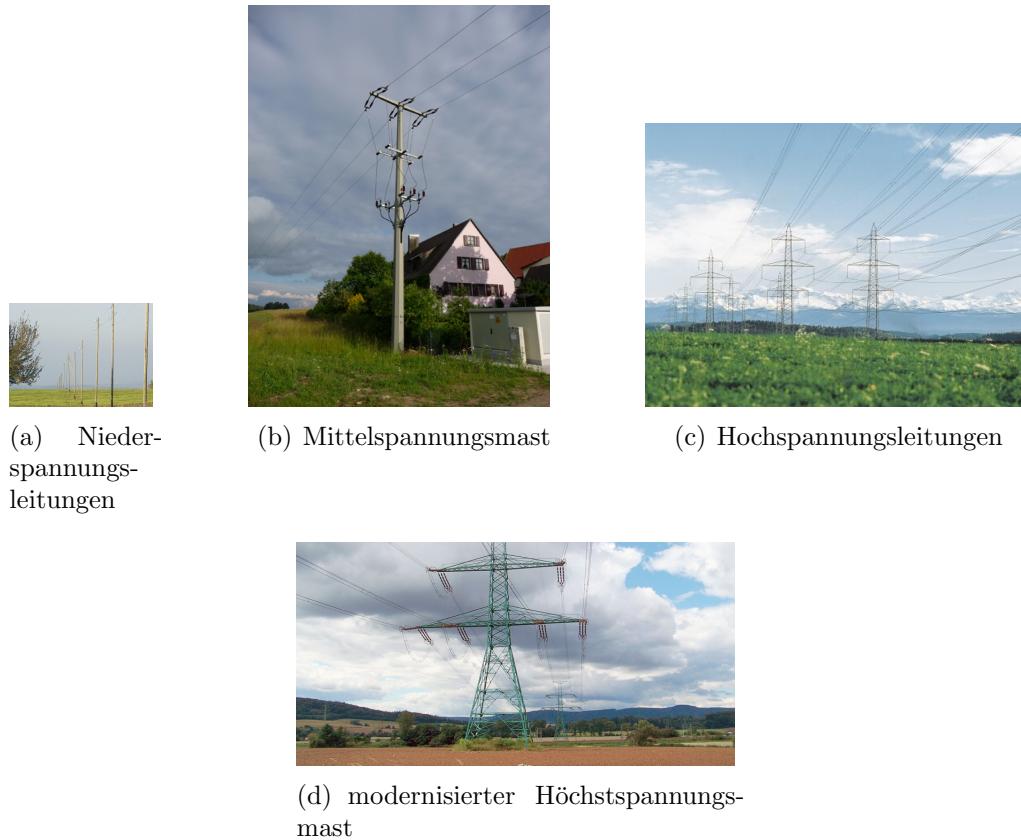


Abbildung 2.2: Die vier verschiedenen Leitungstypen

Herabtransformation auf $11000V$ statt. Dann sind Hochspannungsleitungen für die weitere regionale Übertragung zuständig. Nach einer weiteren Herabtransformation in einer Umspannanlage beträgt die Spannung noch $10000V$ oder $20000V$ und ist jetzt bereit für die Übertragung in der Stadt mittels Mittelspannungsleitungen.

Um die Transportaufgaben zu bewältigen müssen die Höchstspannungsleitungen oft über 100 km lang sein. Das komplette Netz, welches sich über Deutschland erstreckt wird auch das Verbundnetz genannt. Die Einzelnetze werden dabei über Kuppelstellen miteinander verbunden. Das Netz erstreckt sich nicht nur über Deutschland sondern über ganz Europa. Der Vorteil bei einem großen verbundenen Netz sind der Nutz- und Erzeugungsausgleich sowie der kostengünstigen Reservestellung für nicht verbundene Kraftwerke[1].

Da der Stromverbrauch stets variabel ist und sehr örtlich sowie zeitlich unterschiedlich, macht ein großes Verbundnetz viel Sinn. Ebenso aus Sicht der Kraftwerke, zum Beispiel wenn der Schneefall oder Regen ausbleibt, so gibt es an einem Kraftwerk welches an einem Staudamm oder Fluss errichtet worden ist weniger Energie und der Bedarf kann leicht durch andere Quellen über das Netz gedeckt werden.



Abbildung 2.3: Karte des deutschen Höchstspannungsnetzes[1]

Es gibt zwei verschiedene Arten für den Transport der elektrischen Energie:

- Freileitung
- Kabel.

Entschieden wird anhand der technischen Möglichkeiten, die unterschiedlichen physikalischen Eigenschaften der Freileitungen und Kabel, die Kosten sowie Vorstellungen hinsichtlich des Landschaftsschutzes und der Gestaltung des Stadtbildes[1].

Die meisten Leiter bestehen aus Kupfer und Aluminium, dank der hohen Elastizität und elektrischen Leitfähigkeit. Aluminium hat gerade für den Freileitungsbau eine besondere Eigenschaft des sehr geringen Eigengewichtes, was einen größeren Abstand der Masten ermöglicht.

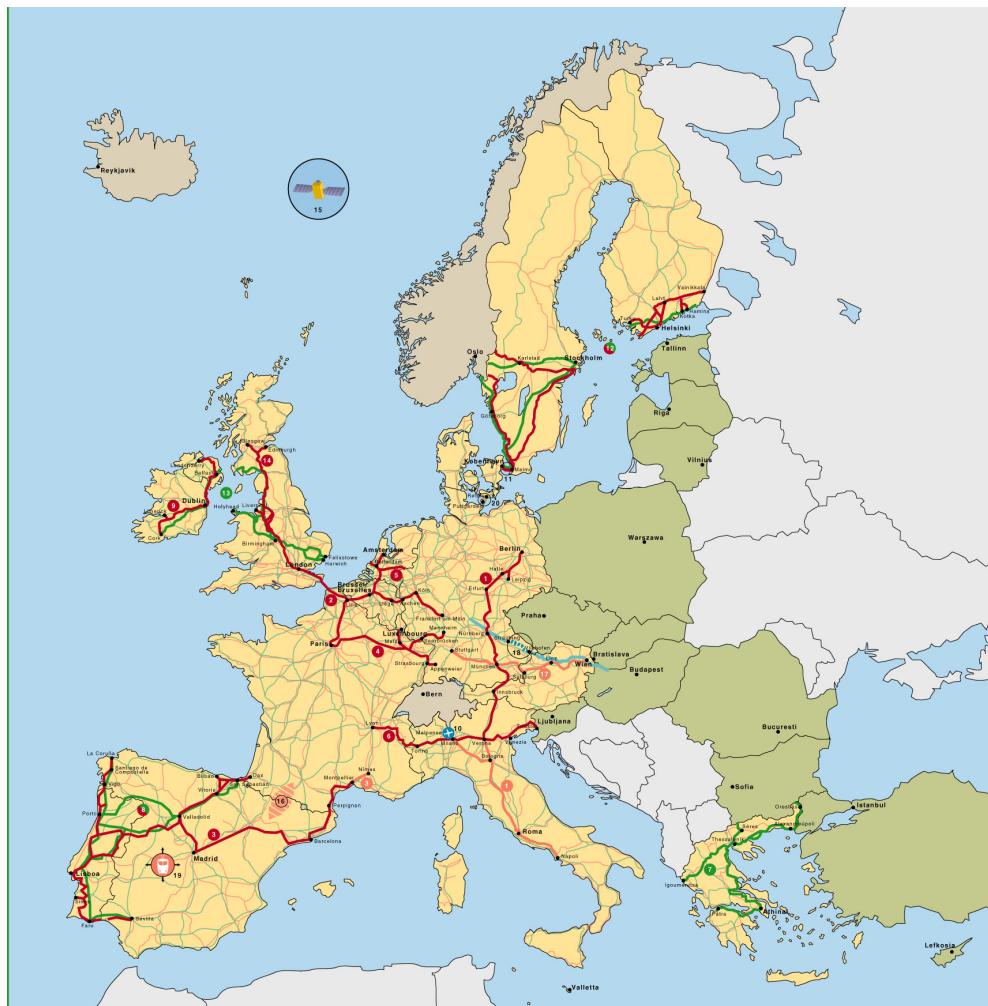


Abbildung 2.4: Karte des europäischen Verbundnetzes[1]

Die Kabel bestehen aus einem Kern der Leiter, welche voneinander durch Isolierungen geschützt werden. Diese werden Adern genannt. Mehrere Adern gebunden durch einen Mantel ergeben das letztendliche Kabel, diese können durch ein- oder mehradrig unterschieden werden. Die einzelnen Isolierungen der Adern bestehen aus ölgetränkten Papier, bei niedrigen Spannungen auch Kunststoff(PVC). Die dicke der Isolierung wird durch die jeweilige Nennspannung der Adern bestimmt.

Der Mantel soll die Kabel eng zusammenhalten und gegen äußere mechanische Beschädigungen schützen, ebenso muss er gegen eindringende Feuchtigkeit isolieren. Da Hochspannungsleitungen eine besonders große Isolierung benötigen, werden die Leiter in einem Stahlrohr mit Stickstoffgas unter sehr großen Druck verlegt.

Einzelne Stromkreise und ein Gestänge bilden eine Freileitung. Die Stromkreise bestehen jeweils aus drei bis vier Leitern. Die spannungsführenden Leiter werden durch Isolatoren an den Masten befestigt, während die anderen Leiter meist

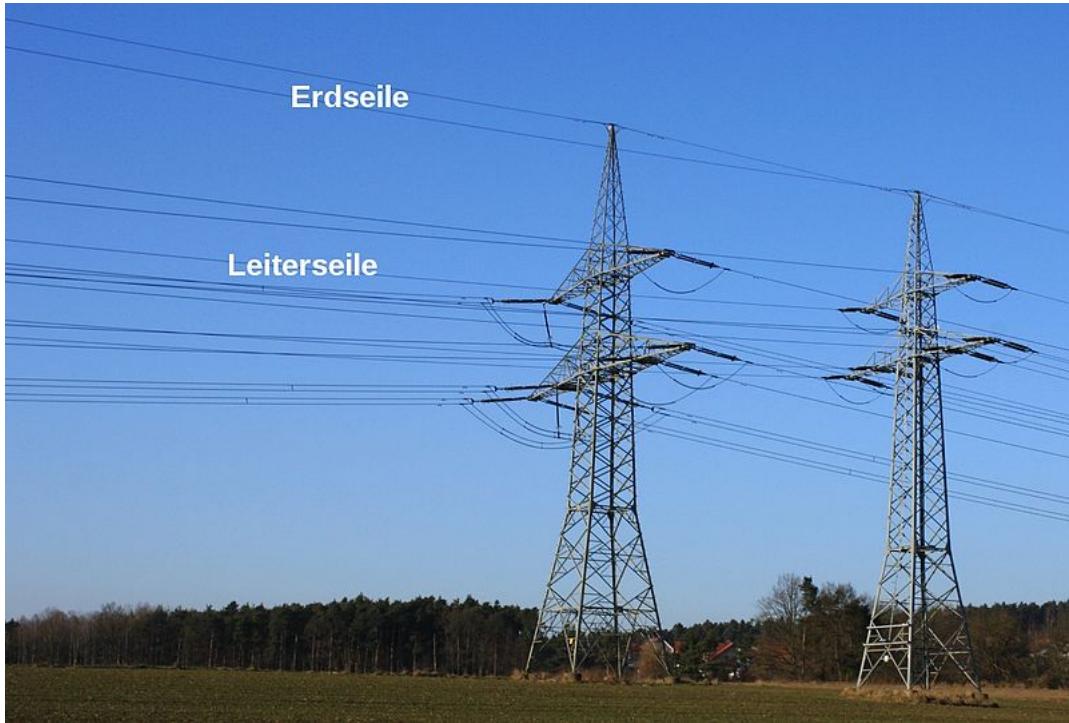


Abbildung 2.5: Freileitungsmast mit Leitern und Erdseil[1]

durch die Luft isoliert werden. Diese werden auch Seile genannt.

Es werden verschiedene Werkstoffe zum Bau der Masten verwendet, dabei entscheidet die Größe der Spannung welcher verwendet wird. Diese sind:

- Holzmasten für Niederspannungsleitungen
- Holz-/Betonmasten für Mittelspannungsleitungen
- Stahlgittermasten für Hoch- und Höchstspannungsleitungen.

Je nachdem wie viel Kraft auf die Masten ausgeübt wird, variiert deren Größe, Konstruktion und Abstand. Äußere Faktoren wie Windbelastungen sowie Schnee- und Eislasten werden dabei mitberücksichtigt.

Die einzelnen Leiter werden mit sogenannten Erdseilen über Isolatoren am jeweiligen Mast angebracht. Die Erdseile dienen dabei nicht dem Transport der elektrischen Energie sondern dem Schutz vor Blitzeinschlägen. Es verlaufen demnach mehrere Stromkreise mit mehreren Leitern und dem Erdseil über einen Mast. In Abbildung 2.5 wird die Konstruktion der Masten sehr gut verdeutlicht. Ebenso zu sehen sind die jeweiligen Isolatoren zwischen den Leitern und den Masten und das Erdseil welches an der Spitze angebracht ist.

2.2 Graph

Um aus dem Höchstspannungsnetz einen repräsentierenden Graphen zu bekommen wurden die jeweiligen Masten zu Knoten und jeweils eins ihrer Leiterseile zu einer verbindenden Kante. Die Position der Masten war aus einer Karte des Höchstspannungsnetz zu bekommen, ebenso deren Verbindungen. Im Folgenden werden nun die grundlegende Strukturen sowie Eigenschaften eines Graphen erläutert.

Ein Graph ist ein Tupel $G = (V, E)$ mit folgenden Eigenschaften:

- V ist eine nicht leere Menge, die sogenannten Knoten
- E ist eine Menge von Kanten zwischen jeweils zwei Knoten[2].

Eine Kante $e_{v_i, v_j} \in E$ mit $v_i, v_j \in V$ repräsentiert somit die Verbindung des Knoten v_i mit v_j .

Ein Graph ist demnach eine Struktur die Informationen anhand Knoten und verbindenden Kanten speichert. Ein einfaches Beispiel ist in der Abbildung 2.6 zu sehen. Es werden die Städte Köln, Kassel, Frankfurt, Nürnberg und München jeweils durch Knoten repräsentiert und deren Höchstleitungsverbindung durch eine Kante. Es ist nun sehr leicht möglich aus dem Graphen zu entnehmen welche Städte durch Freileitungen miteinander verbunden sind.

Ein Graph kann gerichtet oder ungerichtet sein. Bei einem ungerichteten Graphen gilt: $e_{v_i, v_j} = e_{v_j, v_i}$. Bei einem gerichteten Graphen gilt das nicht immer. Es wird der Kante somit eine Richtung mitgegeben. Ein Knoten v_i hat eine Verbindung e_{v_i, v_j} zu einem anderen Knoten v_j , dies muss jedoch nicht bedeuten dass es auch eine Verbindung ausgehend von v_j nach v_i gibt. Es werden im Folgenden nur noch ungerichtete Graphen verwendet, somit gilt stets $e_{v_i, v_j} = e_{v_j, v_i}$.

Gibt es keine Kante e_{v_i, v_i} mit $v_i \in V$ so ist der Graph schlingenfrei, es existiert also keine Kante von einem Knoten zu sich selbst. Eine Kante heißt demnach Schlinge sofern e_{v_i, v_i} gilt.

Kanten sind benachbart oder adjazent, sofern es zwei Knoten $v_i, v_j \in V$ gibt und eine Kante $e_{v_i, v_j} \in E$ sie verbindet. Da nur ungerichtete Graphen verwendet werden, ist es möglich den Start- sowie Endknoten einer Kante stets zu vertauschen und

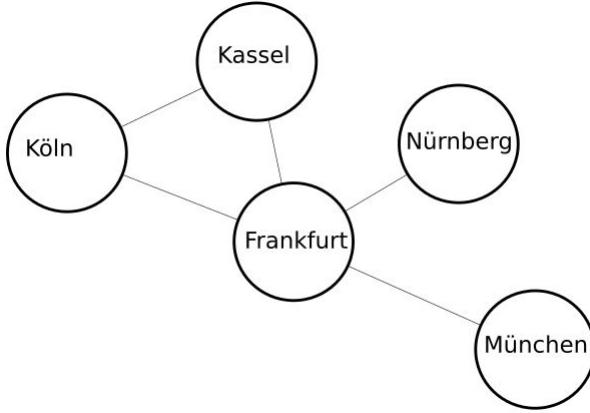


Abbildung 2.6: Einfacher Graph mit Städten als Knoten und deren Höchstspannungsleitungen als Kanten

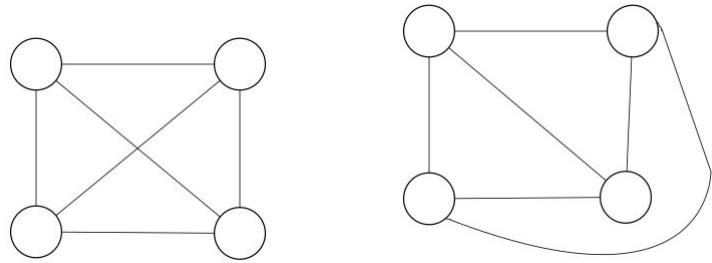
demnach ist auch $e_{v_j, v_i} \in E$ adjazent.

Ein Pfad w_{v_n, \dots, v_m} ist eine Folge von verbundenen Knoten ausgehend vom Knoten v_n nach v_m oder andersrum. Dabei sind die Knoten v_n und v_m Start- und Endknoten der Folge.

Eines der wichtigsten Probleme in der Graphentheorie befasst sich mit dem Finden des kürzesten Pfades zweier Knoten in einem Graphen. Dieser kürzeste Pfad, auch Weg genannt, zwischen zwei unterschiedlichen Knoten v_i , v_j ist der Pfad mit der minimalen Anzahl an Kanten vom Knoten v_i nach v_j [Wikipedia].

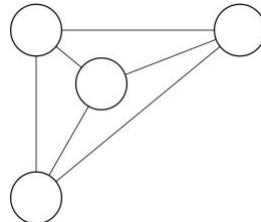
Bei einem Graphen ist nicht nur die Einhaltung der eigentlichen Struktur wichtig sondern auch wie dessen Knoten und Kanten platziert werden, um eine übersichtliche und anschauliche Zeichnung zu erhalten. Eine wichtige Eigenschaft eines Graphen um dies zu erreichen ist die Planarität. Im Zweidimensionalen Raum ist ein Graph planar sofern es möglich ist diesen so zu zeichnen, dass sich keine Kanten überschneiden. Existiert eine solche Darstellung so ist es auch stets möglich den Graphen planar mit geraden Kanten zu zeichnen [2]. In Abbildung 2.7 ist der Unterschied zwischen dem planaren Graphen und seiner unterschiedlichen Zeichnungen zu sehen. Der planare Graph kann somit auch mit überschneidbaren Kanten gezeichnet werden, mit eckigen oder auch immer mit geraden Kanten.

Der Graph wird auf einer simplen Oberfläche im R_2 gezeichnet. Die Knoten werden jeweils durch ihre Koordinaten auf der Karte als Punkte dargestellt. Kanten sind an



(a) Planarer Graph mit einer nicht planaren Zeichnung

(b) Planarer Graph mit einer planaren Zeichnung aber einer kurvigen Kante



(c) Planarer Graph mit planarer Zeichnung sowie geraden Kanten

Abbildung 2.7: Verschiedene Zeichnungen eines planaren Graphen

keinem expliziten Koordinatenpaar gebunden, werden jedoch indirekt durch die Koordinaten ihrer Knoten auf die Oberfläche gezeichnet. Existiert eine Kante $e_{v_i, v_j} \in E$ so wird diese gezeichnet von der Position des Knotens v_i zur Position des Knotens v_j . Durch das Koordinatenpaar der Knoten ist es demnach sehr einfach möglich den Graphen auf einer Oberfläche darzustellen.

Die Knoten werden als Punkte dargestellt, während die Kanten zwischen den Knoten als einfache gerade Striche gezeichnet werden. Es gibt jedoch viele weitere Methoden um dieses Struktur darzustellen. Die Knoten können ebenso mittels Boxen oder Vierecke dargestellt werden, Kanten als Kurven oder gestrichelte Linien[Algorithms for drawing graphs:an annotated bibliography *]. Die Größe und Farbe der einzelnen Knoten und Kanten können im Graph variieren und müssen nicht einheitlich gewählt werden. Dies ist sehr nützlich um bestimmte wichtige Knoten hervorzuheben oder um Beziehungen zwischen Knotengruppen darzustellen, indem ihnen alle die selbe Farbe zugewiesen wird oder ihnen die gleiche Form vergibt. Der Graph, der im Kapitel 2.3 erstellt wird hat zwei verschiedene Gruppen von Knoten. Die eine Knotengruppe lässt sich durch den Algorithmus im Kapitel drei verschieben, wäh-

rend die anderen Knoten zu den unbeweglichen Knoten gehören und ihre Position endgültig ist. Die unbeweglichen Knoten repräsentieren Große Städte und die beweglichen Knoten kleinere Städte und Abzweigungen. Um diese Knoten voneinander unterscheiden zu können sind die Knoten, die die größeren Städte repräsentieren, rot anstatt schwarz und wurden zur noch besseren Erkennbarkeit größer gezeichnet. Die Kanten wurden jedoch stets als schwarze Linien dargestellt, da es keine Unterschiede zwischen den Verbindungen in einem Höchstspannungsnetz gibt.

2.3 Vom Höchstspannungsnetz zum Graphen

In der Abbildung 2.x wird der Vorgang von einem Höchstspannungsnetz zu einem Graph dargestellt. In Abbildung 2.x(a) ist ein Ausschnitt des deutschen Höchstspannungsnetzes zu sehen. Gut zu erkennen sind die einzelnen Städte und ihre Verbindungen mittels den Höchstspannungsleitern. Diese Verbindungen repräsentieren einen Mast mit mehreren verschiedenen Leiterseilen. Die größeren Städte werden dabei als unbewegliche Knoten dargestellt. Ein Knoten ist unbeweglich, sofern er nicht mehr durch einen späteren Algorithmus verschoben werden kann, demnach ist seine Position endgültig. Die eigentlichen Positionen der Knoten werden mittels einfachen x und y Koordinaten zugewiesen.

Auf der Karte in der Abbildung 2.x(a) sind noch eckige Verbindungen zwischen den Städten zu sehen. Eine Verbindung auf dieser Karte repräsentiert den Verlauf der Leiterseile und die Positionen der Höchstspannungsmasten. Ist eine Ecke in der Verbindung zu sehen so wurden Hindernisse umgangen oder kleinere Orte vernetzt. Es ist nicht nötig jeden Mast auf der Karte auch als Knoten zu modellieren, da die späteren Kanten ohne Ecken verlaufen werden. Je mehr Masten als Knoten modelliert werden, desto besser werden die Verbindungen jedoch werden. Besonders wichtig sind die einzelnen Ecken der Verbindungen, diese müssen als Knoten modelliert werden, weil sonst der komplette Verlauf der Leiterseile im späteren Graphen verloren geht. Die Knoten v'_i und v'_n in der Abbildung 2.x(b) sind größere Städte und wurden dementsprechend zu unbewegliche Knoten. Die Knoten v'_j bis v'_m sind die Ecken im Verlauf der Verbindung der Karte in der Abbildung 2.x(a) und entsprechen ihren Masten v'_j bis v'_m . Diese Knoten können durch spätere Verfahren verschoben werden.

Um aus der Karte den abschließenden Graphen zu erhalten, der das Höchstspan-

nungsnetz modelliert, müssen noch die einzelnen Leiterseile der Masten hinzugefügt werden. Bis jetzt wurden nur die Knoten und die grundlegenden Verbindungen modelliert. Auf der Karte in Abbildung 2.x(a) fehlen die Informationen um wie viele Leiterseile es sich bei den jeweiligen Verbindungen handelt. Das ist einer der grundlegenden Unterschiede zwischen einer normalen MetroMap und der neuen Darstellung die aus dem Höchstspannungsnetz resultiert. Bei einer MetroMap oder generell bei Bus- und Bahnverbindungen existieren nur sehr selten mehrere Straßen oder Gleise die immer mit der selben Verbindung verlaufen. Da dies jedoch der Fall ist bei einem Höchstspannungsnetz muss diese Eigenschaft extra modelliert werden. Im folgenden werden die Verbindungen immer mit genau drei Leiterseilen modelliert, anstatt die echten Werte zu nehmen, welche sich zwischen 1-5 Leiterseile befinden, das macht den Graphen für die spätere Handhabung überschaubarer.

Um die jeweiligen Leiterseile modellieren zu können wurden weitere bewegliche Knoten erstellt. Ein Knoten für jede Ecke einer Verbindung steht für das jeweilige Leiterseile an einem Mast. Diese Knoten wurden um den ursprünglichen Knoten platziert, sodass sich parallele Verläufe gebildet haben, wie es auch bei einem echten Höchstspannungsnetz mit den Leiterseilen der Fall ist. Auf der Abbildung 2.x wird diese Verfahren dargestellt, der Knoten v'_j wird in Abbildung 2.x zu den drei Knoten v''_i , v''_j und v''_k , die jetzt den Mast mit den drei Leiterseilen repräsentieren. Da es drei Leiterseile pro Verbindung gibt, wurde die Anzahl der Kanten des resultierenden Graphen durch dieses Vorgehen fast verdreifacht.

2.4 MetroMap Layout Problem

Eine MetroMap ist die besondere Darstellungen eines Graphen, der aus einer Karte von verschiedenen Netzen gewonnen wird. Dieser Graph wird versucht so anschaulich wie möglich darzustellen, sodass es möglich ist die Struktur des Graphen schnellstmöglich zu verstehen und Verbindungen zwischen Knoten nachvollzogen werden können. Häufig werden allgemeine Netze wie das Bus- und Bahnnetz mittels einer MetroMap dargestellt, damit sind sie Bestandteil des täglichen Lebens. Viele benutzen sie zur schnellen Orientierung und Übersicht. Eine MetroMap macht aus, dass sie schnell zu lesen und zu verstehen ist. Es muss demnach möglich sein die benötigten Informationen in kurzer Zeit und sicher aus der Karte zu erhalten.

MetroMaps dienen hauptsächlich dazu die richtige Bahn oder Bus zum gewünschten

Ziel zu nehmen und sich eine gute Übersicht der Lage zu verschaffen. Die geographisch echten Karten mit ihren Verzweigungen und Überschneidungen erfüllen diesen Zweck nicht. In Abbildung 2.9 wird eine Dortmunder MetroMap der lokalen Bahnverbindungen gezeigt. Fast alle vorhandenen MetroMaps von Bahn- und Busnetzwerken wurden per Hand gezeichnet, jedoch ist es auch möglich dies mittels Algorithmen zu tun. Es gibt bereits viele verschiedene Algorithmen sowie Ansätze die MetroMaps automatisch aus einem Graphen generieren. Um diese zu behandeln muss zu erst geklärt werden was genau eine MetroMap ist und was diese ausmacht.

Eine essentielle Eigenschaft die der Graph der MetroMap haben sollte ist die Planarität. Wird der Graph wie ein Labyrinth gezeichnet mit überlappenden Kanten, so wird diese fast unmöglich für den alltäglichen Gebrauch zu benutzen sein.

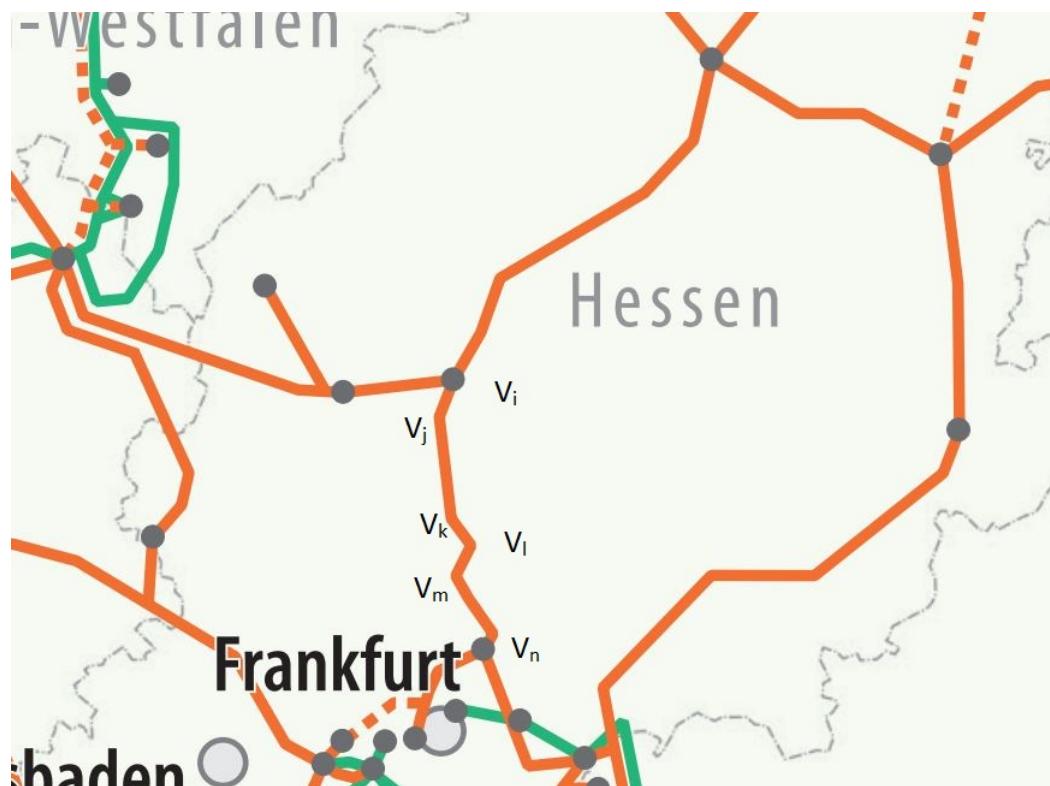
Eine weitere Sache die entscheidend dazu beiträgt die resultierende Karte möglichst übersichtlich zu gestalten ist die Vermeidung von eckigen und runden Kanten. Es ist viel einfacher den gewollten Weg auf einer senkrechten bzw. waagerechten Strecke zu verfolgen als wenn dieses über eckige und runde Kanten geschieht.

Bei einer MetroMap ist es sehr wichtig zu unterscheiden um welche Art der Verbindung es sich handelt. Denn ein Bus fährt nicht auf Gleisen und Züge nicht auf Straßen, somit müssen unterschiedliche Verbindungen gekennzeichnet werden. Dafür werden die jeweiligen Pfade $w_{v_n,..,v_m}$ im Graphen aufgestellt, die diese Verbindungen repräsentieren. Hierbei sind die Knoten v_n und v_m jeweils die letzten Stationen der Verbindung. Diese können nun auf unterschiedliche Weise gekennzeichnet werden, durch Färbung, Umrandung oder indem die Verbindung verschieden gestrichelt wird.

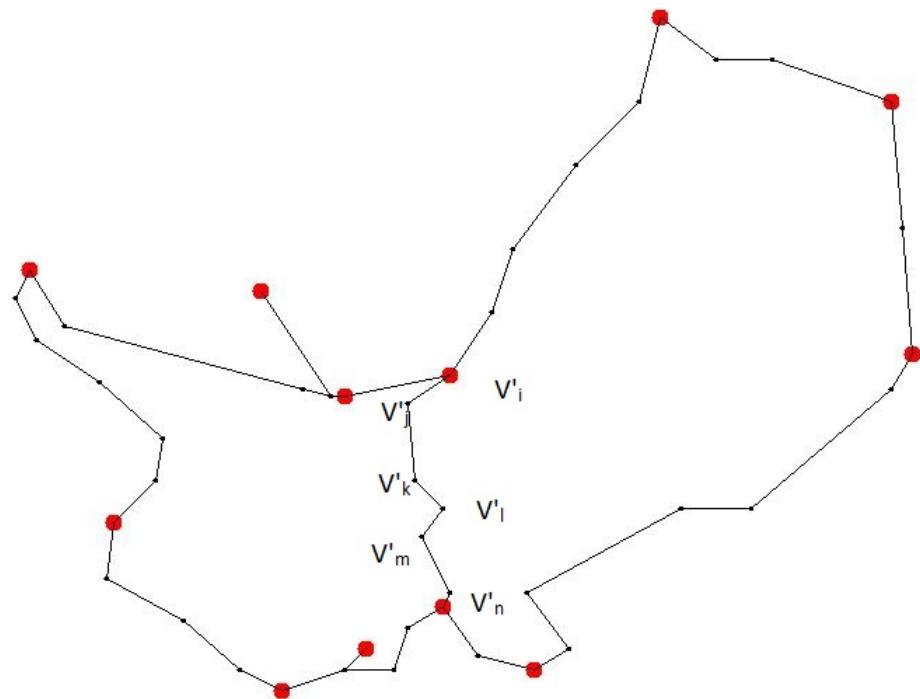
Eine MetroMap verliert ihre Eigenschaft der Topologie sowie der geographischen Richtigkeit im Vergleich zu einer normalen Karte, dies resultiert zwingend aus der Verschiebung der Knoten und Kanten um einen übersichtlicheren Graphen zu bekommen. Da die Verbindungen aus den realen Netzen modelliert wurden, wird stets eine gewisse Topologie beibehalten beim Erstellen der MetroMap. Dabei wird der geographische Abstand zwischen zwei Punkten vernachlässigt. Die Lage der ursprünglichen Stationen ist zur Bewahrung der Navigation da weitaus wichtiger. Es würde auch stark verwirren wenn eine Verbindung von A über B nach C geht, die eigentlichen Stationen auf der Karte jedoch in Reihenfolge A C B gezeichnet werden. Dies würde auch im Widerspruch dazustehen, gerade Kanten aufrechtzuerhalten und ganze Verbindungen möglichst senkrecht und waagerecht zu zeichnen.

Neben der Platzierung der Kanten und Knoten ist es sehr wichtig eine passende Beschriftung der Objekte in der MetroMap zu haben. Für diese Beschriftungen muss es den Platz geben, da sich die Beschriftungen nicht mit den Kanten oder Knoten überschneiden dürfen. Wenn der Abstand zwischen den jeweiligen Beschriftungen der Kanten und Knoten zu weit oder zu nah ist, wird es schwierig zu erkennen zu welchem Objekt diese gehört.

Ein möglicher Ansatz dieses Problem zu lösen wäre es jedem Objekt eine gewisse Anzahl von möglichen Beschriftungspositionen zu geben und anschließend die sich überschneidenden Positionen als mögliche Beschriftungen zu streichen. Was bleibt sind überschneidungsfreie Beschriftungen eines jeden Objektes. Es müssen demnach zuerst die möglichen Beschriftungspositionen eines Objektes gefunden werden. Dazu werden 8 Positionen um das jeweilige Objekt gewählt, wie in der Abbildung 2.x dargestellt. Es ist nun möglich diesen Positionen eine Wertigkeit zuzuweisen. Im 45 Winkel zum Objekt, wäre die Beschriftung generell am besten platziert um einen übersichtlichen Graphen zu bekommen. Nun ist es leicht die beste überschneidungsfreie Beschriftung mit den geringsten Kosten zu finden. Die Kosten stellen die Wertigkeit der Positionen aller Beschriftungen der Objekte.

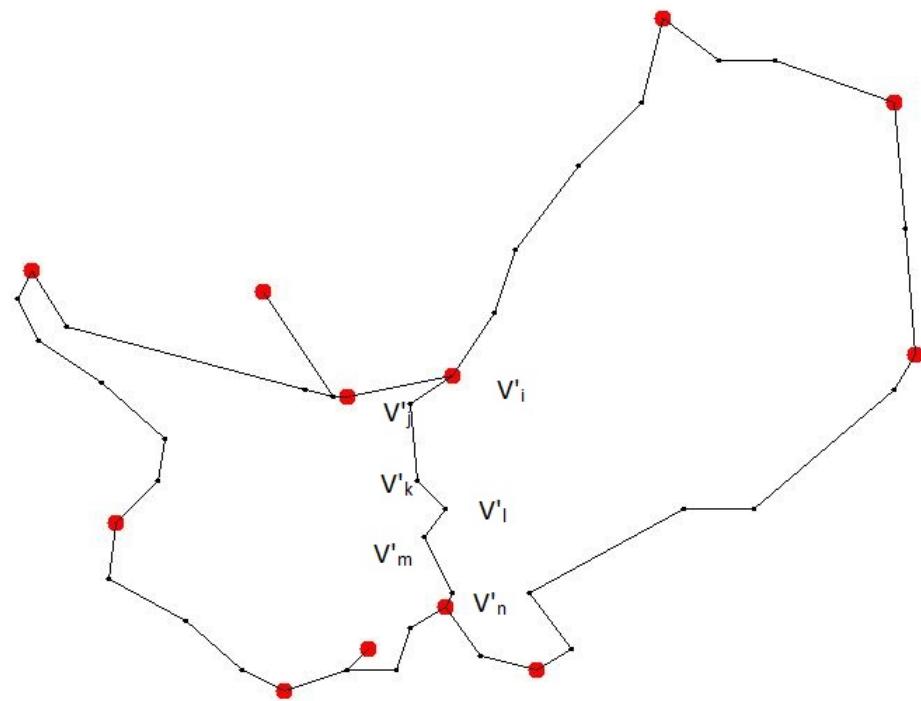


(a) Ausschnitt des deutschen Höchstspannungsnetzes

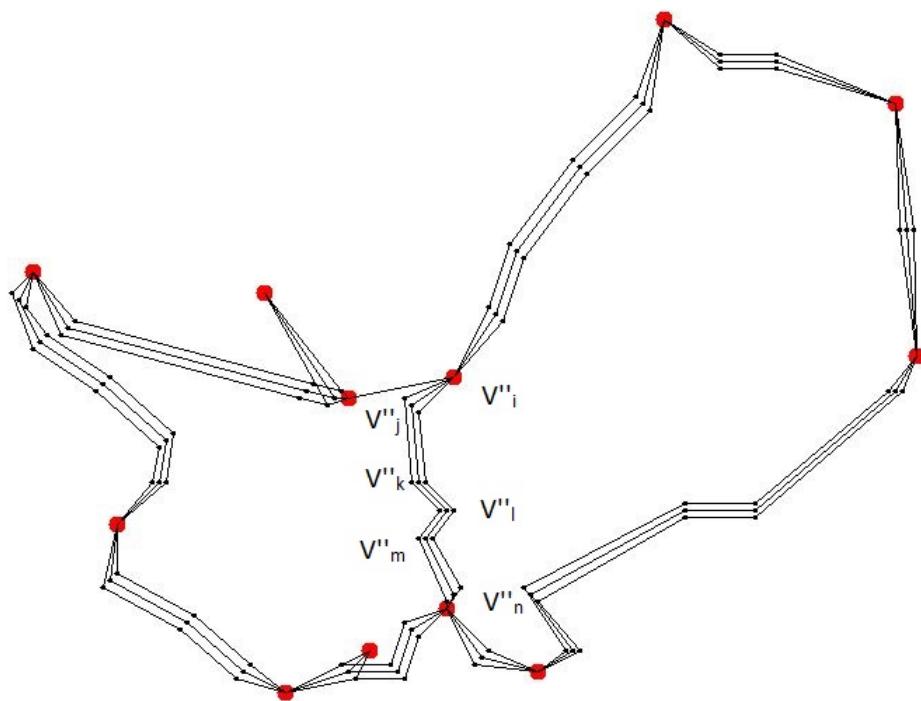


(b) Resultierender Graph aus der Karte

Abbildung 2.8: Übergang von der Karte des Höchstspannungsnetzes zu einem Graphen



(a) Resultierender Graph aus der Karte



(b) Der Graph mit den modellierten Leiterseilen

Abbildung 2.9: Modellierung der Leiterseile

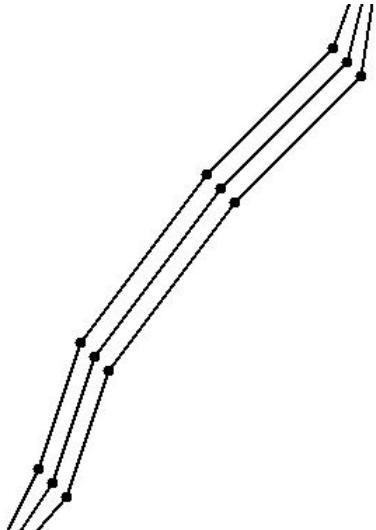


Abbildung 2.10: Erzeugung der neuen Knoten zur Modellierung der Leiterseile

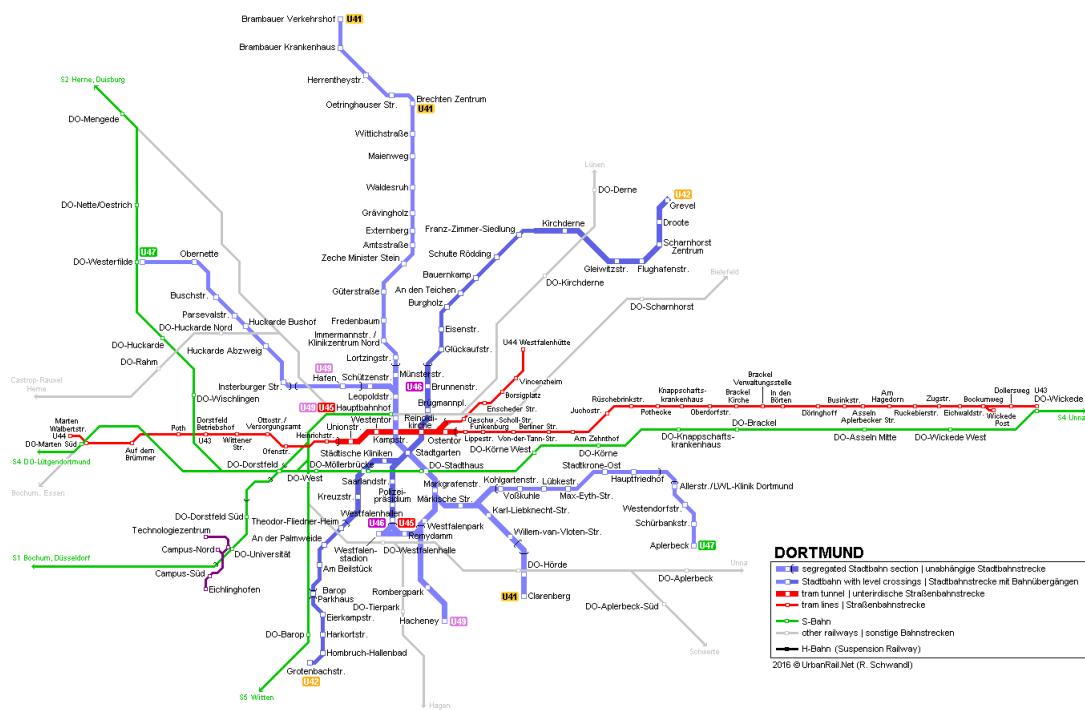


Abbildung 2.11: Dortmunder MetroMap [<http://www.urbanrail.net/eu/de/do/dortmund-map.png>]

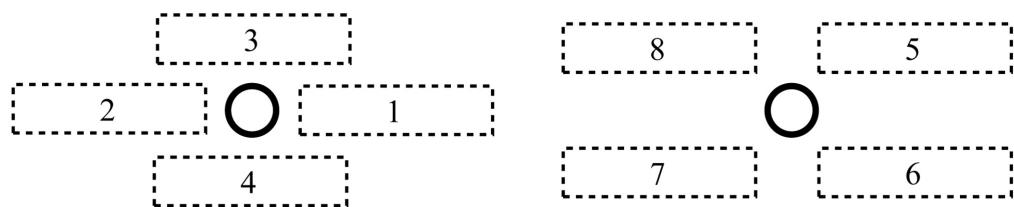


Abbildung 2.12: 8 Mögliche Platzierungen der Beschriftung
[<http://doi.ieeecomputersociety.org/cms/Computer.org/dl/trans/tg/2011/01/figures/ttg20110101018.jpg>]

3 Spring-Embedder

In diesem Kapitel werden die Grundlagen des Spring-Algorithmus erklärt, der im Kapitel 4 dazu verwendet wird um eine MetroMap ähnliche Darstellung des Höchstspannungsnetzes zu erhalten. Es geht um die Verwendung dieser Algorithmen und das grundsätzliche Vorgehen. Ebenso werden auf Unterschiede zwischen den meist verwendeten Spring-Embedder eingegangen.

3.1 Spring-Embedder - allgemein

Auf Kräfte basierte Algorithmen gehören zu den flexibelsten Methoden zur Berechnung eines simplen Layouts für ungerichtete Graphen. Diese Algorithmen, auch Spring-Embedder genannt, beruhen nur auf der zugrunde liegenden Struktur des Graphen. Zeichnungen des Graphen die aus diesen Algorithmen resultieren, neigen dazu sehr ästhetisch und symmetrisch zu sein. Zudem liefern sie meist einen überschneidungsfreie Darstellung des Graphen sofern dieser planar ist.[Spring Embedders and Force Directed GraphDrawing Algorithms]

Die folgenden Ausführungen stammen aus dem Buch “Software-Practice and Experience, VOL. 21(1 1), 1129-1164”(November 1991).

Das Problem der Darstellung basiert auf der Platzierung der Kanten sowie Knoten um eine möglichst ästhetische Zeichnung des Graphen zu erhalten, die gut lesbar und verständlich ist. Die grundlegenden Kriterien für eine ästhetischen Zeichnung eines Graphen sind:

1. die Knoten sollten gleichmäßig verteilt werden
2. minimale Überschneidung der Kanten
3. einheitliche Länge der Kanten
4. möglichst symmetrisch

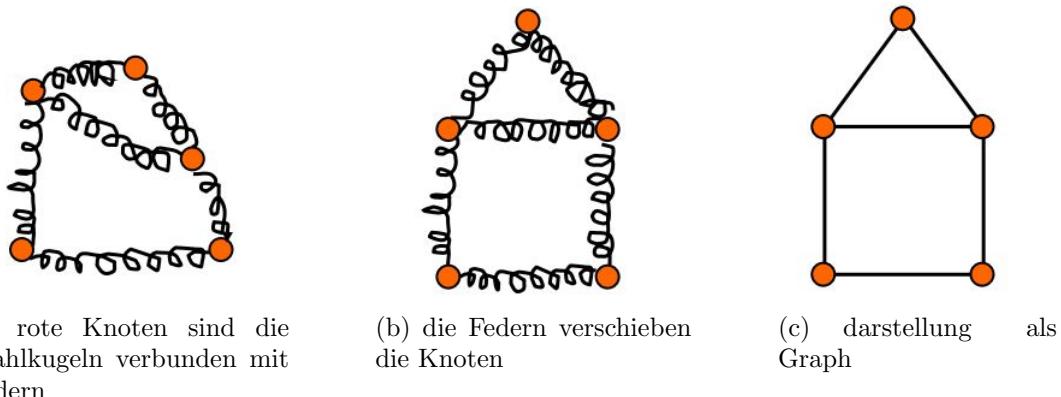


Abbildung 3.1: Darstellung als Stahlkugeln und Federn eines Graphen

5. alles soll sich innerhalb der Fläche befinden[Graph Drawing by Force-directed Placement].

Das heißt es ist leicht möglich die wichtigen Informationen des Graphen anhand einer visuellen Darstellung zu erhalten. Dazu gehören unter anderem die Erkennung von Verbindungen zwischen Knoten oder deren Lage. Dies ist vor allem wichtig bei einer MetroMap wo es darum geht schnell herauszufinden wie es möglich ist von einem Punkt zu einem anderen zu kommen. Um dies zu erreichen gibt es verschiedenste Ansätze. Diese müssen auf dem verwendeten Graphen anwendbar sein, sollten jedoch auch simpel und effizient sein.

Im Folgenden wird es um eine Erweiterung eines Verfahrens gehen, welches auf Grundlage des von Peter D. Eades entwickelten Algorithmus aus dem Jahre 1984 basiert. Er erklärt seine Metapher wie folgt:

The basic idea is as follows. To embed [lay out] a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system . . . The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state.

Es geht demnach um ein Modell, indem die Knoten durch Stahlringe und die Kanten durch Federn repräsentiert werden. Durch die Federn wirken anschließend Kräfte auf die Kugeln ein, wodurch sich diese verschieben. Die Kugeln sollten jedoch vorher zufällig verteilt werden damit es mit diesem Vorgang zu einer besseren Lösung kommt. Das Verschieben der Kugeln durch die Federn passiert solange bis es zu einem sogenannten minimal-energy-state kommt. Das bedeutet, dass die Federn und

Kugeln in einer Position befinden, indem die einwirkenden Kräfte der Federn keine Bewegung der Kugeln mehr zulassen. Dies ist meist erreicht sobald die Federn ihre optimale Länge erreicht haben, sie demnach keine weitere Kraft mehr auf die Kugeln wirken können. Es kommt jedoch nicht immer dazu. Es kann passieren, dass sich von zwei Federn, die noch nicht ihre optimale Länge haben, durch die Konstellation der Knoten ihre wirkende Kraft gegeneinander auflöst.

Die wirkenden Kräfte der Federn können nach belieben gewählt werden und müssen dabei nicht dem Hookeschem Gesetz entsprechen. Das hookesche Gesetz beschreibt die elastische Verformung von Festkörpern, wenn deren Verformung proportional zur einwirkenden Belastung ist[Wikipedia]. Es können demnach beliebige Definitionen, der wirkenden Kräfte der Federn, benutzt werden um ein gewünschtes Ergebnis zu erreichen. Ein weiterer Unterschied zur physikalischen Realität ist die Anwendung der Kraft auf die Knoten. Es können beliebige Knoten gewählt und auf diese, die gewünschte Kraft angewendet werden. Im Folgenden werden anziehende Kräfte zwischen jedem Knotenpaar, welches durch eine Kanten verbunden ist, angewendet. Abstoßende Kräfte wirken zwischen jedem Knoten, die nicht miteinander verbunden sind, demnach ebenso nicht durch eine Feder modelliert wurden. Das sind die Freiheiten des Verfahrens im Vergleich zur Realität.

Ein weiteres sehr ähnliches Verfahren, welches auf dem von Eades beruht ist der Algorithmus von Kamada und Kawai. Sie modellierten den Graphen ebenfalls als ein System von Federn, jedoch, im Vergleich zu Eades, hielten sich sie sich an das Hockesche Gesetz und versuchten dieses mit mehreren Differentialgleichungen zu lösen um ein verbessertes Layout zu erhalten. Eades hielt es für sehr wichtig, dass verbundene Knoten nah zueinander gezeichnet werden, demnach hat er ausschließlich anziehende Kräfte zwischen verbundenen Knoten berechnet. Kamada und Kawai haben hingegen ein weiteres Konzept eingefügt, welches auch anziehende Kräfte zwischen nicht verbundenen Knoten berechnet. Diese Kräfte sind proportional zur Länge des kürzesten Pfades der beiden Knoten. Das Problem der Zeichnung eines Graphen sahen Kamada und Kawai als das Vorgehen zur Reduzierung der noch zu wirkenden Kräfte der Federn in dem Graphen. Ist ein Zustand erreicht in dem die Summe der wirkenden Kräfte der Federn zum Tiefpunkt gekommen ist, so sind die Positionen und Abstände der Knoten nahe zu optimal gewählt. Sie definierten die komplette Energie des Graphen als:

$$\sum_{\leq i < j \leq |V|} k_{p_{v_i} p_{v_j}} (|p_{v_i} - p_{v_j}| - l_{v_i v_j})^2 \quad (3.1)$$

wobei p_{v_i} die Position des Ringes oder Knotens v_i ist, $k_{p_{v_i} p_{v_j}}$ ist die Feder-Konstante zwischen den Positionen p_{v_i} und p_{v_j} und $l_{v_i v_j}$ ist der optimale Abstand der beiden Knoten v_i und v_j . Diese Energie wird reduziert durch das Lösen von Differentialgleichungen eines Knotens zur Findung einer besseren Position mit einer geringeren Menge an Energie im System. Die Positionierung der Knoten findet so lange statt, bis eine gegebene Grenze erreicht wurde. Ein wesentlicher Unterschied im Vergleich zu Eades Algorithmus ist, dass pro Iteration immer nur ein Knoten bewegt wird.

3.2 Spring-Embedder - Vorgehen

Die optimale Länge der Kanten k , die als Federn modelliert werden, wird durch die Anzahl der Knoten sowie der zur Verfügung stehender Fläche A bestimmt:

$$k = C \sqrt{A/|V|} \quad (3.2)$$

$$A = W * L. \quad (3.3)$$

W ist die Weite und L die Länge der zugrunde liegenden Oberfläche, auf der, der Graph gezeichnet wird. Es ist wichtig zu wissen wie diese Maße sind um eine bessere Verteilung der Knoten zu ermöglichen. Da die bessere Verteilung der Knoten zu einem ästhetischen Graphen führt wird versucht die Oberfläche optimal zu nutzen. Wie weit Knoten voneinander entfernt sein sollten, wird demnach durch die Anzahl der Knoten und der Oberfläche bestimmt. Besonders wichtig ist es, dass die Knoten nicht zu nah aber auch nicht zu weit voneinander entfernt gezeichnet werden, denn beides führt zu einer Verschlechterung der Attraktivität des Graphen. $k = C \sqrt{A/|V|}$ führt dazu, dass sich die Knoten $|V|$ weit genug voneinander entfernt befinden um die Fläche gut zu nutzen und ebenso nah genug sind um die Ästhetik der Zeichnung zu bewahren. Die optimale Distanz k gibt somit auch die freie Fläche um einen Knoten an. Je weiter die momentane Distanz von der optimalen Distanz zwischen

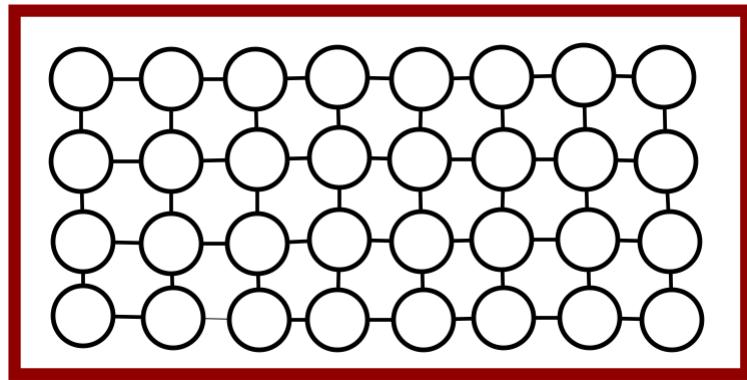


Abbildung 3.2: Eine Beispielhafte Abbildung der Knoten mit einer nahezu perfekten Verteilung

den Knoten abweicht, desto größer sollte ihr Einfluss auf das Layout werden. Die Neuausrichtung der Knoten passiert drastischer, wenn die Knoten noch zu weit von ihrer optimalen Distanz von einander entfernt sind und verringert sich sobald die Knoten näher an ihre optimale Distanz zu einander kommen. Dabei macht es keinen Unterschied ob die Knoten zu nah oder zu weit von einander entfernt sind. In der Abbildung 3.x wird die perfekte Verteilung der Knoten dargestellt. Die rote Umrandung gibt den Rand des Bildes an. Die Knoten sind dort alle symmetrisch verteilt mit einem optimalen Abstand zu einander. Die Knoten haben auf dieser Abbildung auch eine orthogonale Lage. Der Algorithmus führt jedoch nicht immer zu einer orthogonalen Positionierung.

Die Konstante C sollte experimentell gewählt werden, denn die Struktur eines Graphen kann stark variieren. Um dennoch in der Lage zu sein auf diese Schwankungen reagieren zu können kann die Konstante $C > 1$ beliebig vergrößert werden um einen größeren optimalen Abstand zu erhalten oder bei $C < 1$ die Größe des Abstandes verringern.

Bildet die bisherige Platzierung der Kanten und Knoten einen planaren Graphen, so wird die neue Platzierung in fast allen Fällen auch planar sein. Ein Graph ist planar wenn sich keine Kanten überschneiden, diese Eigenschaft ist besonders gewollt um den resultierenden Graphen noch übersichtlicher zu gestalten.

Die Ein- sowie Ausgabe des in 3.1 vorgestellten Algorithmus, ist ein Graph $G := (V, E)$, der bereits eine gewisse Positionierung der Knoten hat. Dieser kann sofern er noch keine hat, eine zufällige Positionierung der Knoten vor Beginn des Algorithmus

```

Eingabe:  $G := (V, E)$ 
Ausgabe:  $G := (V, E)$ 
1: for  $i := 1 \leq \text{iterations}$  do
2:   for  $v_i \in V$  do
3:      $d_{v_i} := 0;$ 
4:     for  $(v_j \in V)$  do
5:       if  $v_i \neq v_j$  then
6:          $\Delta := p_{v_i} - p_{v_j};$ 
7:          $d_{v_i} := d_{v_i} + (\Delta/|\Delta|) \cdot f_{v_i, v_j}^r(|\Delta|);$ 
8:       end if
9:     end for
10:    end for

11:   for  $e_{v_i, v_j} \in E$  do
12:      $\Delta := p_{v_i} - p_{v_j};$ 
13:      $d_{v_i} := d_{v_i} - (\Delta/|\Delta|) \cdot f_{v_i, v_j}^a(|\Delta|);$ 
14:      $d_{v_j} := d_{v_j} + (\Delta/|\Delta|) \cdot f_{v_i, v_j}^a(|\Delta|);$ 
15:   end for

16:   for  $v_i \in V$  do
17:      $p_{v_i} := p_{v_i} + (d_{v_i}/|d_{v_i}|) \cdot \min(d_{v_i}, t);$ 
18:      $p_{v_i}^x := \min(W/2, \max(-W/2, p_{v_i}^x));$ 
19:      $p_{v_i}^y := \min(L/2, \max(-L/2, p_{v_i}^y))$ 
20:   end for
21:    $t := \text{cool}(t)$ 
22: end for

```

Algorithmus 3.1: Spring-Algorithmus

erhalten, was den Algorithmus sogar noch effektiver macht, da sich dadurch einige Probleme vermeiden lassen, was mit reellen Graphen öfter der Fall ist. Auf die möglichen auftretenden Probleme werden in diesem Kapitel auch noch eingegangen. Die meisten Herausforderungen sind die lokalen Minima und die dadurch entstehenden Ausreißer.

Zwischen jedem Knotenpaar $v_i, v_j \in V$ wird eine abstoßende Kraft f_{v_i, v_j}^r berechnet. Alle benachbarten Knoten erhalten eine anziehende Kraft f_{v_i, v_j}^a . Zwei Knoten $v_i, v_j \in V$ sind benachbart wenn $e_{v_i, v_j} \in E$ ist. Das führt dazu, dass verbundene Knoten näher zusammen gezeichnet werden, während sie noch immer einen gewissen Abstand zueinander haben. Der Algorithmus geht dabei in drei Schritten vor:

1. zwischen jedem Knotenpaar die abstoßende Kraft berechnen

2. benachbarten Knoten eine anziehende Kraft zuweisen
3. jeden Knoten seiner neuen Kraft nach bewegen.

Diese drei Schritte werden so oft wiederholt bis keine weitere Bewegung mehr stattfindet. Die Anzahl der Iterationen können auch fest gewählt werden, um eine Obergrenze einzuhalten. Eine weitere Möglichkeit ist die $\text{cool}(t)$ Funktion zur Regulierung der Bewegungsmöglichkeiten der Knoten. Die genaue Wirkung und Vorgehensweise der Funktion wird am Ende des Kapitels noch erläutert.

Die Funktionen f_{v_i, v_j}^a und f_{v_i, v_j}^r sind wie folgt definiert:

$$f_{v_i, v_j}^a(x) = x^2/k \quad (3.4)$$

$$f_{v_i, v_j}^r(x) = -k^2/x. \quad (3.5)$$

In der Abbildung 3.x ist das Verhältnis zwischen den Kräften und der optimalen Distanz k abgebildet. Die x-Achse stellt die Distanz dar und die y-Achse die Wirkung der Kräfte f_{v_i, v_j}^a , f_{v_i, v_j}^r . Die Summe der beiden Kräfte wird mit der Funktion $f_{v_i, v_j}^a + f_{v_i, v_j}^r$ dargestellt. Wenn die Summe der beiden Kräfte null erreicht, ist dies die optimale Distanz k . Das passiert auf der x-Achse, sofern sich die abstoßende Kraft und die anziehende Kraft gegenseitig auflösen.

Die beiden Funktionen wurden dabei experimentell gewählt. Sie ergaben sich bei der Implementierung und ähnelten dem hookeschen Gesetz. Dabei ergaben sich auch andere Funktionen wie:

$$f_{v_i, v_j}^a(x) = x/k \quad (3.6)$$

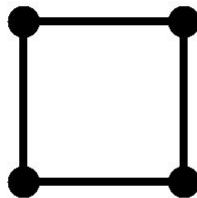
$$f_{v_i, v_j}^r(x) = -k/x. \quad (3.7)$$

Diese schienen auch das Problem zu lösen, lieferten aber bei komplexeren Graphen weitaus schlechtere Layouts. Das lag daran, dass sie nicht in der Lage waren ein sogenanntes lokales Minima zu überwinden. Ein lokales Minima tritt auf, wenn die



(a) anziehende Kraft f^a zwischen dem dritten und vierten Knoten

(b) abstoßende Kraft f^r zwischen dem ersten und vierten Knoten



(c) die wirkenden Kräfte lösen sich auf

Abbildung 3.3: Wirkende Kräfte im Graphen

initiale Positionierung eines Knotens sich in den Weg eines anderen Knotens stellt und sich dieser deswegen in eine eigentlich schlechtere Position bringt. Dieser Knoten müsste seine schlechte Position dadurch verbessern, dass er erst durch eine weitaus schlechtere Positionierung geht um anschließend eine bessere Position zu erhalten. Dieser Vorgang wird auch als *hillclimbing* bezeichnet. Da die Knoten durch Kräfte bewegt werden, die stets eine bessere Position anstreben, dabei aber nicht wissen, wie sich die anderen Knoten in Zukunft platzieren werden, ist ein direktes Vorgehen gegen dieses Problem mit diesem Ansatz nicht möglich. Die Funktionen f_{v_i, v_j}^a , f_{v_i, v_j}^r ergaben weitaus öfter den Fall, dass Knoten in einem lokalen Minima feststecken und sich dabei nicht mehr in eine bessere Position bringen konnten. Je größer die wirkenden Kräfte gewählt werden, desto wahrscheinlicher ist es, dass sich ein Knoten in einem Schritt an einen anderen vorbei bewegt und somit ein lokales Minima verhindert. In der Abbildung 3.x wird das lokale Minima Problem anhand eines kleinem

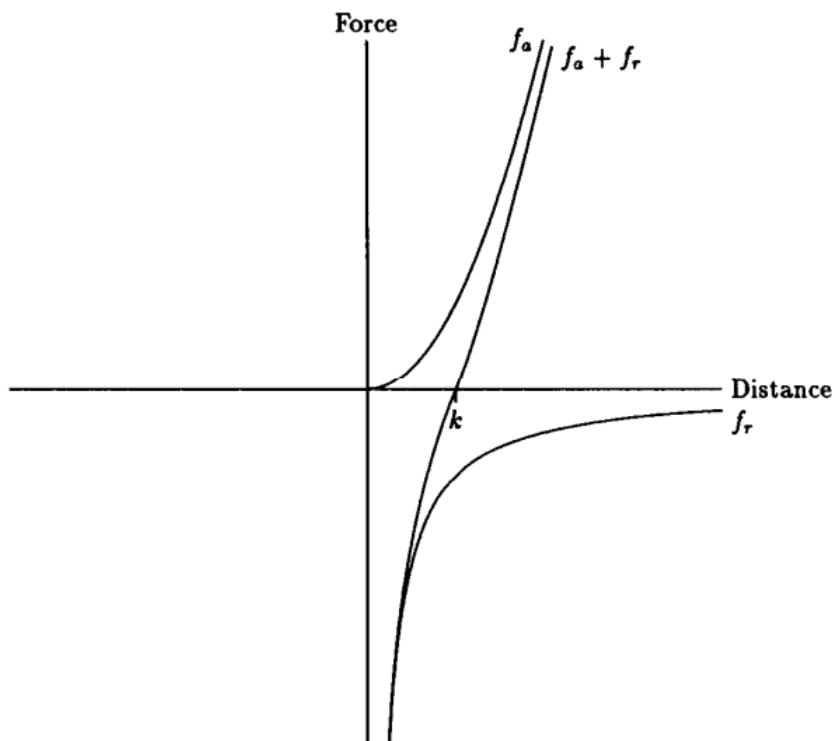


Abbildung 3.4: Verhältnis der Kräfte zueinander und deren Einfluss auf die optimale Distanz

Beispiel veranschaulicht. Vor der Anwendung des Algorithmus ist die Positionierung der Knoten wie in 3.x(a). Wird der Algorithmus nun angewendet verteilen sich die Knoten wie in der Abbildung 3.x(b). Sie können demnach nicht das lokale Minima überkommen. In 3.x(c) ist eine mögliche perfekte Darstellung der Knoten zu sehen. Die Knoten B und D oder aber die Knoten A und C müssten die Position komplett tauschen, was jedoch voraussetzt, dass erst einmal eine schlechtere Positionierung gewählt werden muss um zu dieser Darstellung zu kommen. Der Algorithmus wird demnach auch nach beliebig vielen Iterationen immer in diesem Minima stecken bleiben und niemals zur besseren Darstellung in 3.x(c) übergehen können.

Durch dieses Problem treten auch die sogenannten Ausreißer auf. Das sind Knoten die sich in eine besonders schlechte Position gebracht haben, dadurch, dass sie in einem lokalen Minima gefangen sind. Diese Position wird meist immer weiter verschlechtert und führt zu einer sehr unschönen Zeichnung, die geprägt wurde von einigen dieser Ausreißer. In der Abbildung 3.x(a) ist ein Graph zu sehen, welcher mit Anwendung des Algorithmus zur Bildung eines Ausreißers führt. Der Knoten D liegt bereits vor Ausführung des Algorithmus nicht optimal. Eine optimale und

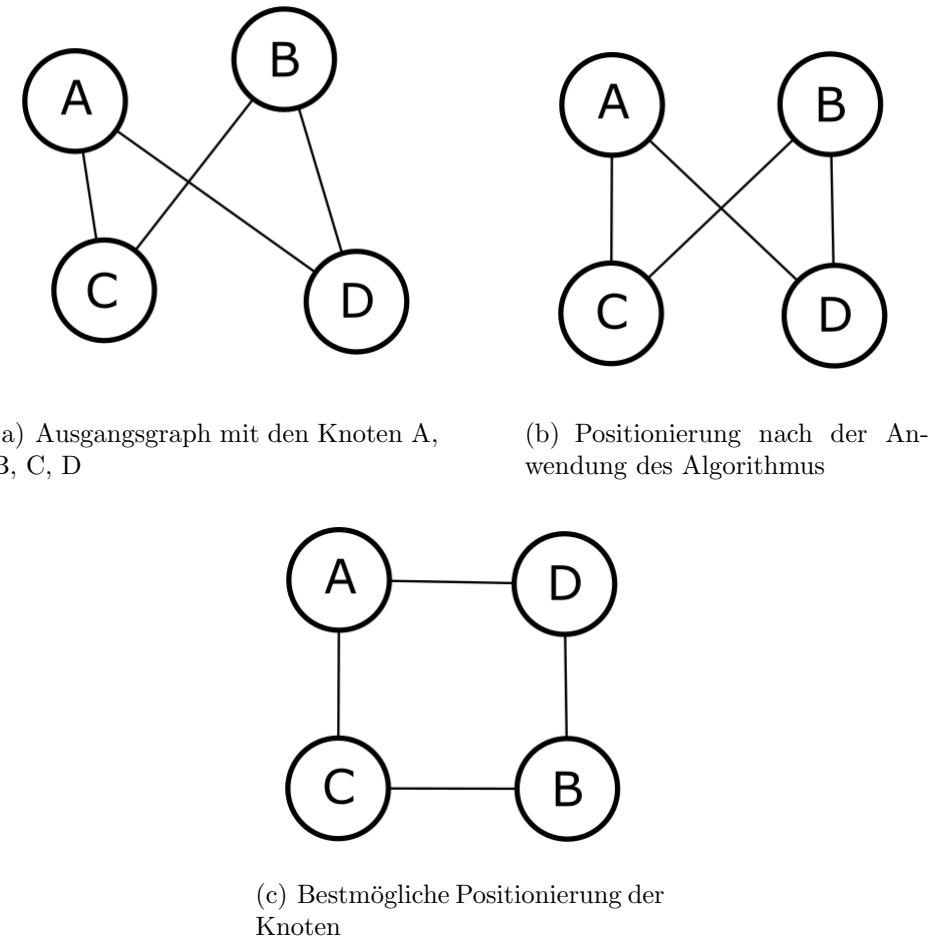
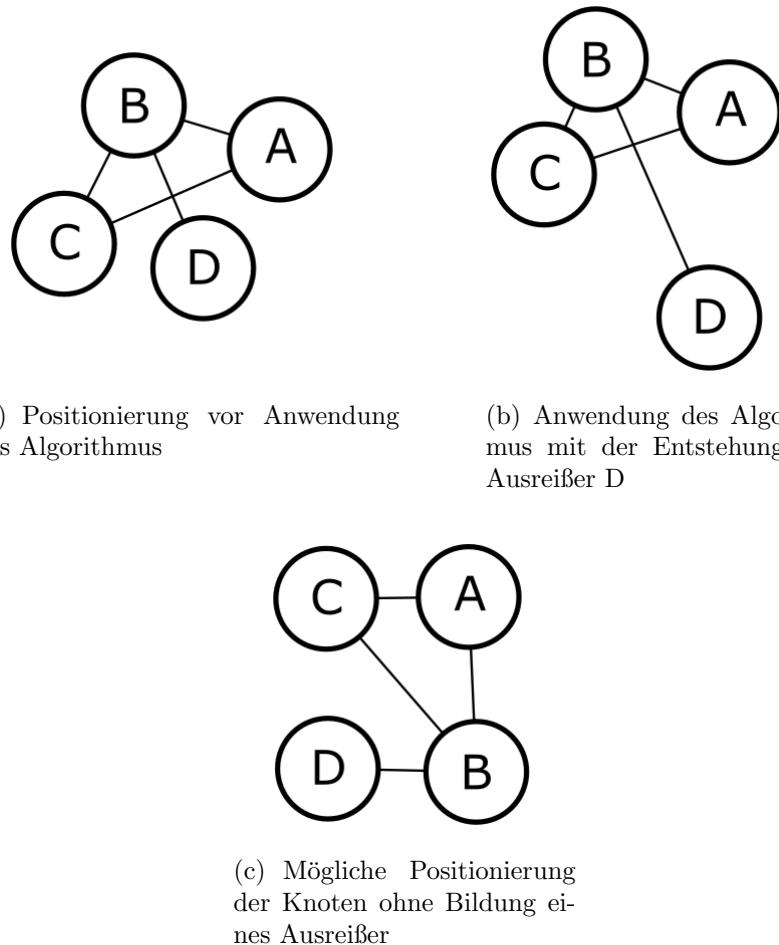


Abbildung 3.5: Lokales Minima-Problem

planare Darstellung des Graphen ist in der Abbildung 3.x(c) zu sehen. Es ist also möglich den Graphen planar und kompakt zu zeichnen, indem die Knoten überschneidungsfrei und alle die optimale Distanz zueinander haben. In der Abbildung 3.x(b) entsteht durch die Anwendung des Algorithmus der Ausreißer D. Dieser Knoten wird durch die abstoßenden Kräfte von den Knoten A, B und C weiter nach außen gedrückt und nur die anziehende Kraft zwischen B und D ist nicht stark genug um diesen näher zu halten. Dieser Knoten wird sich nicht mehr bewegen, sobald die Summe der abstoßenden Kräfte und der anziehenden Kraft gleich null ist und sie sich in einem Gleichgewicht befinden. Durch die ungünstige Verteilung der Knoten zu Beginn des Algorithmus und die besonderen Kanten zwischen ihnen, kommt es letztendlich zur Bildung eines Ausreißers.

Ein ähnliches Problem des Algorithmus, was auch zu einem lokalen Minima führt,

**Abbildung 3.6:** Bildung von Ausreißer

ist die Bildung von einzelnen vielen Knoten an verschiedenen Punkten, die meist nur durch wenige Kanten miteinander verbunden sind. Diese Ballen von Knoten entstehen wie die Ausreißer, nur dass es nicht einzelne Knoten sind sondern eine Sammlung von Mehreren. In der Abbildung 3.x(a) ist dieses Problem aufgetreten. Wenn die Knoten zu Beginn des Algorithmus bereits ähnlich, wie in dieser Abbildung zu sehen ist, liegen. So ist sehr wahrscheinlich, dass sich dort ein lokales Minima bildet. Dieses Minima wird dadurch definiert, dass sich die Knoten nicht mehr in eine optimale Lösung bewegen können, da sie sich vorher in eine schlechtere Position bringen müssten. Die Knoten A und C sowie E und F stoßen die Knoten B und D nach außen. Die Knoten B und D können durch diese abstoßende Kraft nicht zur ihrer optimalen Distanz gebracht werden, da die Kraft dafür nicht ausreichend ist. Die anziehenden Kräfte von A,C,E und F auf die Knoten B und D tragen dazu auch verstärkend bei. Diese Positionierung der Knoten würde auch nach beliebig vielen Iterationen nicht in ein optimales Layout resultieren, denn die Kräfte haben sich be-

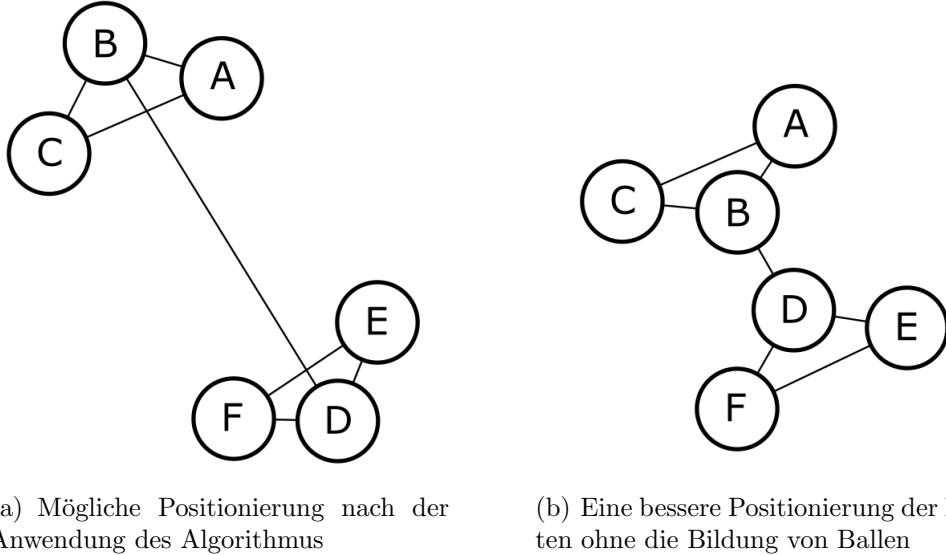


Abbildung 3.7: Resultat von Ballenbildung und dessen Auswirkung auf das Layout

reits ausbalanciert, was keine weitere Bewegung mehr zulässt. Eine optimale Lösung bei diesem Graphen, wäre ähnlich wie in der Abbildung 3.x(b). Die Knoten B und D haben ihre lokalen Minima beschränkend durch die Knoten A, C, E, F überwunden, was zu einem deutlich besseren Layout führt.

Zur Bewegung der Knoten wurde ein Vektor d_{v_i} eingeführt, dieser gibt an in welcher Richtung der Knoten v_i bewegt wird. Während der Vektor p_{v_i} auf die momentane Position des Knoten v_i zeigt. Der Vektor Δ ist die Differenz zwischen p_{v_i} und p_{v_j} , der Richtungsvektor von v_i nach v_j . Die Variable *iterations* gibt die Anzahl der Durchläufe an.

Bei jedem neuen Durchlauf wird der Vektor d_v eines jeden Knotens v auf null gesetzt. Dies geschieht in Zeile 3 des Algorithmus 3.1. Mit zwei For-Schleifen wird jede Knotenkombination durchgegangen und anschließend wird die abstoßende Kraft f_{v_i, v_j}^r mithilfe der Länge des Δ Vektors berechnet. Der Δ Vektor ist nichts anderes als der Abstand zwischen den zwei Knoten, dieser wird gebraucht um die Kraft anhand des bereits vorhandenen Abstandes zu regulieren. Sind zwei Knoten v_i, v_j bereits sehr weit entfernt, so wird die berechnete Kraft in der Funktion f_{v_i, v_j}^r durch die Eingabe von Δ schwächer ausfallen. Anschließend wird die berechnete Kraft mit dem Einheitsvektor multipliziert und auf den Bewegungsvektor schließlich addiert. Dadurch drücken sich die Knoten direkt voneinander weg.

In den Zeilen 11-15 geschieht der zweite Schritt des Algorithmus, die Berechnung der anziehenden Kraft zwischen den benachbarten Knoten. Dazu wird jede Kante einmal mit Hilfe einer For-Schleife durchgegangen und wie bei der Berechnung der abstoßenden Kraft muss auch der Abstand zwischen den beiden Knoten v_i, v_j für die Funktion f_{v_i, v_j}^a vorher berechnet werden, um festzustellen, ob die Knoten bereits die optimale Distanz zu einander haben. Ist dies der Fall, so wird berechnete anziehende Kraft deutlich schwächer ausfallen, als bei Knoten die noch zu weit von einander entfernt sind. Anschließend wird auch hier der Bewegungsvektor mit dem Ergebnis der Berechnung aktualisiert.

Dadurch dass jeder Knoten nun mehrmals eine Aktualisierung seines Bewegungsvektor erhalten hat, durch den Einfluss aller Knoten untereinander, ist dieser nun sehr genau bestimmt worden und die Knoten können sich in seine angegebene Richtung bewegen. Im letzten Schritt des Algorithmus, demnach Zeile 16-20, werden die Knoten nun in Richtung ihres Bewegungsvektors bewegt. Die Zeilen 18 und 19 dienen dabei, dass der Knoten während der Bewegung nicht die Grenzen des Bildes verlässt.

Die Variable t ermöglicht es, am Anfang viel Bewegung der Knoten zuzulassen und dies immer weiter einzuschränken, damit mit der Graph mit jeder Iteration feiner wird. Die Funktion $cool(t)$ regelt dabei wie weit sich jeder Knoten in der nächsten Iteration maximal bewegen darf. t könnte zum Beispiel bei der Länge des Bildes starten und sich nach jeder Iteration um ein Zehntel mit $cool(t)$ kürzen. Nach zehn Iterationen wären dadurch keine weiteren Bewegungen der Knoten mehr möglich. Die Implementierung dieser Funktion zeigte jedoch keinen sichtbaren Unterschied zur manuellen Regulierung der Iterationen. Dadurch wurde diese Funktion im Folgenden Kapitel ebenfalls nicht benutzt. Anstatt dessen, wird der Algorithmus durch eine verschiedene Anzahl an Iterationen getestet.

4 Anwendung des Spring-Embedders auf den modellierten Graphen

Der in Kapitel 3 vorgestellte Algorithmus wird nun so erweitert und angepasst, dass er für den gegebenen Graphen und unser Problem passend ist. Das Ziel wird es sein, den aus Kapitel 2 gewonnen Graphen mithilfe des Algorithmus zu einer MetroMap ähnlichen Darstellung zu führen. Die besonders leichte Erweiterbarkeit und Modifizierbarkeit des Algorithmus erlauben dies und ist auch der Grund, weshalb dieser Algorithmus als Basis zur Lösung des Problems verwendet wurde.

4.1 Vergleich des vom Algorithmus erhaltenen Layout mit dem einer MetroMap

Das Erstellen dieses Layouts im Vergleich zur herkömmlichen MetroMap unterscheidet sich vor allem durch:

- eine deutlich striktere Beibehaltung der topologischen und geographischen Eigenschaft des Graphen
- das Verteilen der Knoten auf die komplette Oberfläche soll nicht stattfinden
- es existieren Knoten, die sich nicht bewegen lassen
- durch das modellieren der Leiterseile existiert eine bestimmte Struktur innerhalb des Graphen, die noch immer sichtbar bleiben soll
- nichts außerhalb des Platzierens von Knoten und Kanten wird angewendet, das sind vor allem, bei einer MetroMap, das Erstellen von Labels und die Färbung von Kanten.

Der vorgestellte Algorithmus in Kapitel 3 wandelt einen Graphen ebenso nicht in eine perfekte MetroMap um. Orthogonalität wird komplett ignoriert. Durch das

zufällige Platzieren der Knoten, wird jegliche Topologie des Graphen entfernt und alle anderen ausmachenden Komponenten einer MetroMap werden nicht erstellt. Ausschließlich das Platzieren der Knoten und Kanten sind beim Erstellen des neuen Layouts von Bedeutung, dh. es wird bei der Platzierung auch nicht auf mögliche Positionen der Labels geachtet. Das Ziel ist es demnach nun den Algorithmus den neuen Ansprüchen anzupassen und ein ansprechendes Layout des Ausschnittes des Höchstspannungsnetzes zu erstellen. Dieser Ausschnitt kann anschließend beliebig erweitert werden oder das komplette Höchstspannungsnetz umfassen.

4.2 Nachteile des Spring-Embedders auf der Anwendung des Graphen

Im Folgenden wird der Algorithmus unmodifiziert auf den Graphen angewendet um deutlich zu machen, wo die grundsätzlichen Probleme dieses Algorithmus liegen und was verbessert werden muss. Dazu wird dieser mit einer verschiedenen Anzahl an Iterationen auf den Graphen angewendet und anschließend ausgewertet. Existierten keine Bewegungseinschränkungen für die Knoten und würde der Algorithmus so auf den Graphen angewendet werden, würden viele unerwünschte Nebeneffekte eintreten, die den Graphen noch unästhetischer machen. In den Abbildung 4.1/2/3 wird der resultierende Graph dargestellt nach jeweils 10, 25 und 50 Iterationen.

Nach 10 Iterationen des Algorithmus wird eine deutliche Verbesserung der Abstände der Knoten sichtbar. Das resultiert aus der abstoßenden Kraft der Knoten zueinander. Jedoch treten bei 10 Iterationen schon große Probleme mit der Überschneidung von Kanten auf, was eine nicht planare Darstellung des Graphen zur Folge hat. Dies sollte jedoch unter allen Umständen vermieden werden, denn wie es auf der Abbildung 4.1 schon leicht zu sehen ist, wird durch das Überschneiden von Knoten der Graph sehr unübersichtlich und es wird schwer zu sagen, zu welcher Kante welcher Knoten gehört. Da die vergrößerten Abstände der Knoten auf den langen Verbindungen zu einer deutlichen Verbesserung der Ästhetik führen, sieht der resultierende Graph nach 10 Iterationen, trotz der leichten Überschneidung von Kanten, schon deutlich übersichtlicher aus.

Nach weiteren Iterationen des Algorithmus wird die Darstellung zunehmend schlechter und es kommt zu etlichen Überschneidungen der Kanten. Die Knoten stoßen sich weiter ab und sie fangen an sich an die Grenzen der Oberfläche zu bewegen, welches

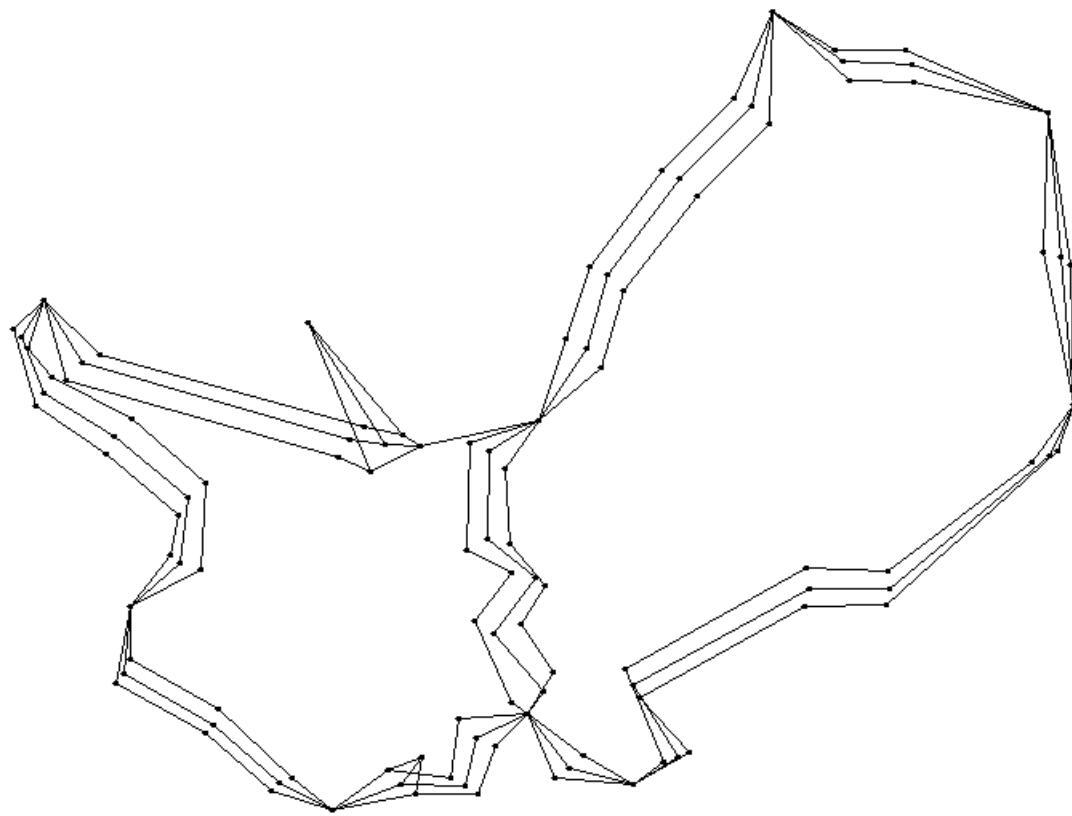


Abbildung 4.1: Dortmunder MetroMap [<http://www.urbanrail.net/eu/de/do/dortmund-map.png>]

auf der Abbildung 4.2 nach 25 Iterationen am linken, rechten und unteren Rand gut zu sehen ist. Die Knoten fangen an sich dort zu sammeln und trotz ihrer abstoßenden Kraft zueinander wird das Sammeln zunehmend schlimmer. Der ursprüngliche Graph, welcher aus dem Höchstspannungsnetz modelliert wurde, ist noch immer wiederzuerkennen. Die beiden markanten großen Flächen innerhalb des Graphen, welche der ursprüngliche Graph auch besaß, sind noch deutlich zu erkennen. Der Zweck des Algorithmus die Knoten gleichmäßig zu verteilen mit einer optimalen Distanz zu einander wird langsam deutlich durch die größeren Abstände der Knoten.

Nach 50 Iterationen, welches auf Abbildug 4.3 zu sehen ist, sind fast alle Knoten bereits deutlich von ihrer Startpositionen bewegt worden. Ursprüngliche Muster, Merkmale oder die Wiedererkennbarkeit dieser Zeichnung im Vergleich des Graphen vor Anwendung des Algorithmus, wird immer schwerer. Die äußeren Grenzen sind sehr deutlich am Rande der Oberfläche zu sehen. Die Knoten sammeln sich dort, da sie sich nicht weiter nach außen entfernen können. Die anderen Knoten fangen an

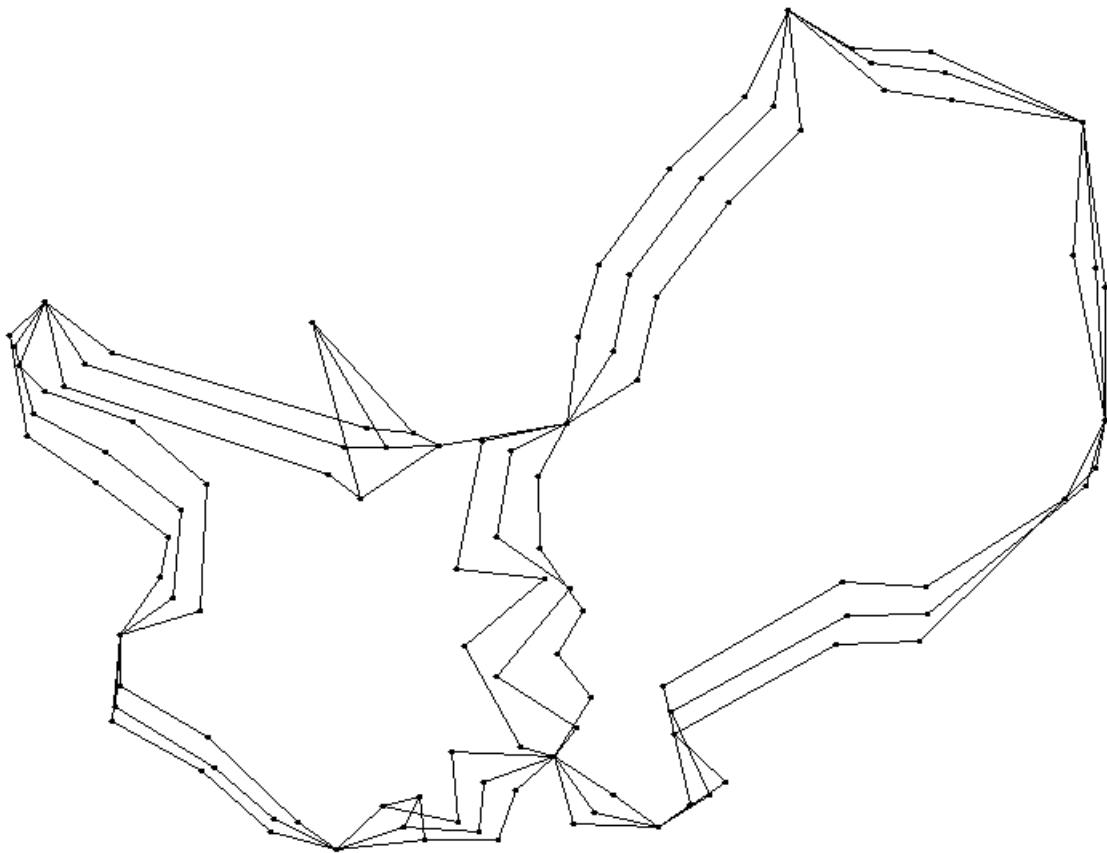


Abbildung 4.2: Dortmunder MetroMap [<http://www.urbanrail.net/eu/de/do/dortmund-map.png>]

sich in den beiden offenen Flächen des Graphen zu verteilen und zu Positionieren. Es scheint als wäre die Oberfläche nicht groß genug für den Algorithmus mit der gegebenen Anzahl an Knoten, doch auch eine deutliche Verringerung der Knoten führt zum selben Ergebnis, da die optimale Distanz zwischen zwei Knoten noch deutlich zu groß ist. Die Knoten versuchen sich ausnahmslos auf der Oberfläche mit perfekten Abständen zu einander zu verteilen. Dieser Vorgang ist jedoch nicht gewollt, nicht bei dem Layout unseres Graphen noch bei einer grundsätzlichen MetroMap. Es gibt auch sehr viele Ausreißer, die ein lokales Minima nicht mehr überwinden können. Das bedeutet, der Algorithmus wird niemals, eine erst schlechtere Positionierung wählen um anschließend eine bessere zu bekommen. Er wird stets versuchen von Iteration nach Iteration eine bessere Positionierung anhand der Kräfte zu erhalten, auch wenn dies zur Folge hat, dass die Knoten sich möglicherweise nicht optimal Positionieren.

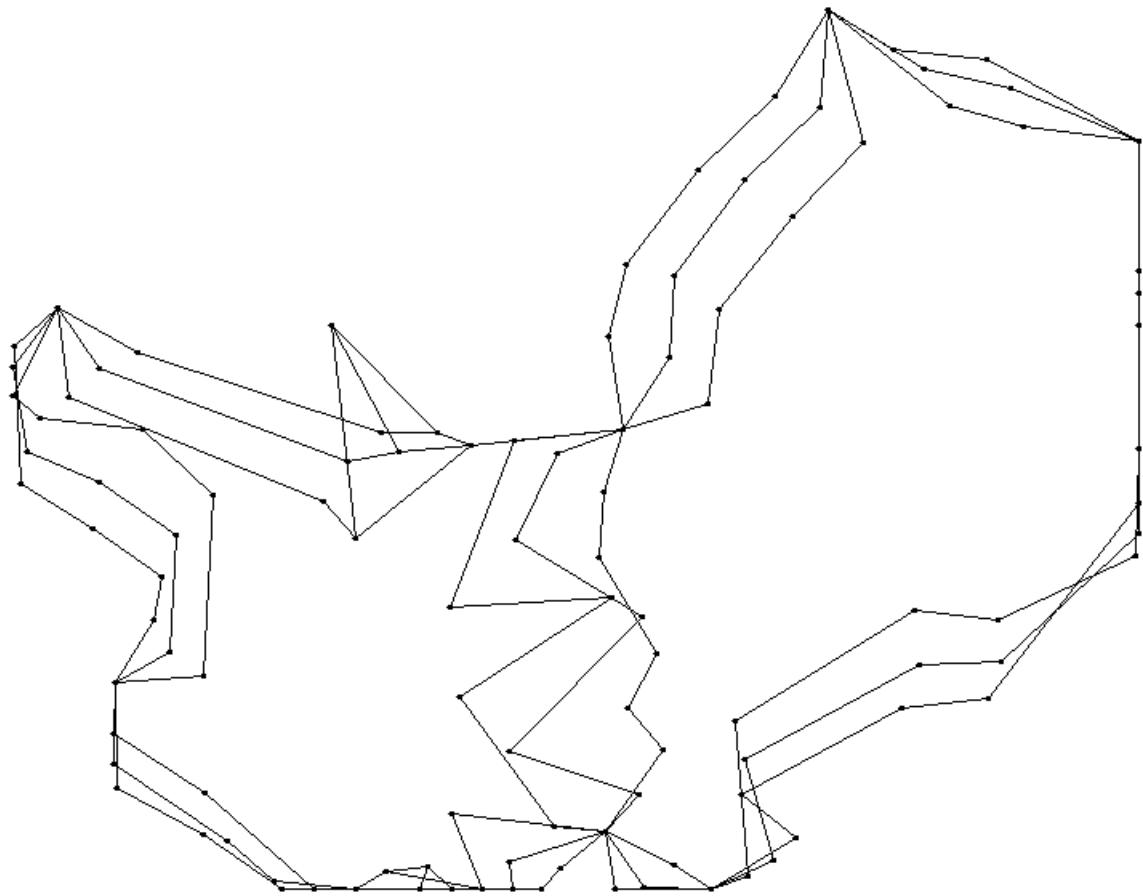


Abbildung 4.3: Dortmunder MetroMap [<http://www.urbanrail.net/eu/de/do/dortmund-map.png>]

4.3 Anpassung des Spring-Embedders zur Anwendung auf das modellierte Höchstspannungsnetz

Zum Anwenden auf das Höchstspannungsnetz war der Algorithmus in seiner ursprünglichen Form nicht geeignet, besonders nicht mit der zur Hinzunahme der erweiterten Regeln die zum Erstellen des Layouts gelten. Nun geht es demnach darum, den Algorithmus so anzupassen, dass er ein passendes Layout des Graphen liefert, der die Anforderung besser erfüllt. Der Algorithmus kann demnach auf viele verschiedene Arten erweitert oder verändert werden.

Eine bereits deutliche Besserung des Layouts führt eine Veränderung der optimalen Distanz. Die Konstante C bei der Berechnung der optimalen Distanz $k = C\sqrt{A/|V|}$ sollte so gewählt werden, dass sie die optimale Distanz zwischen zwei benachbarten

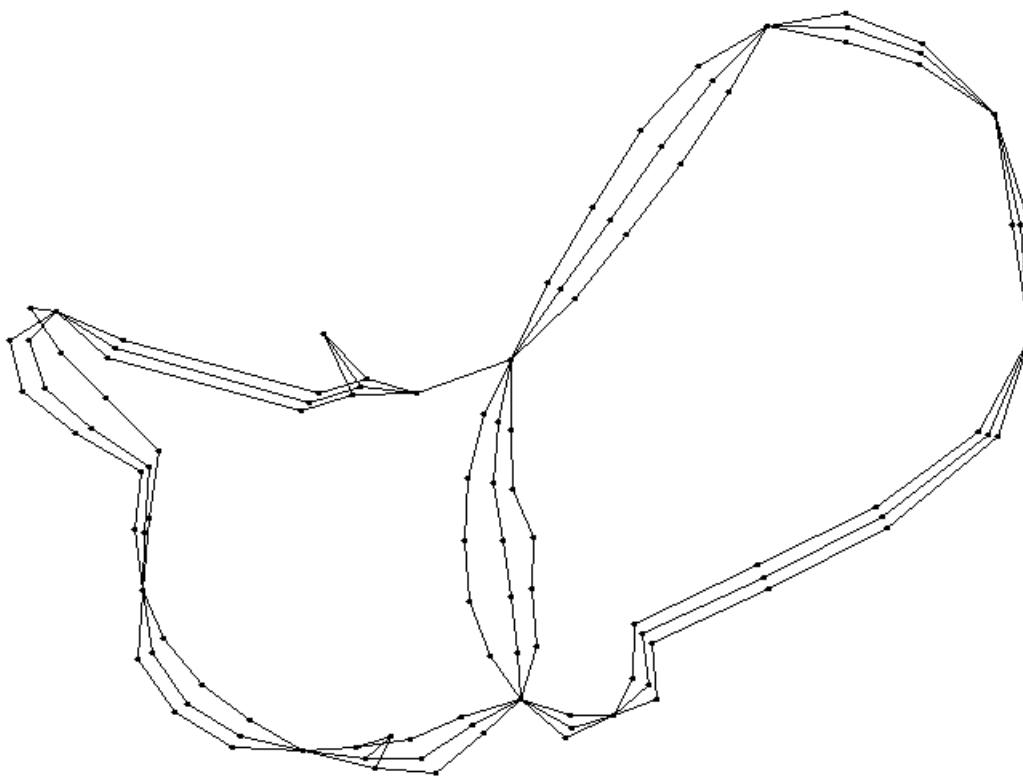


Abbildung 4.4: Dortmunder MetroMap [<http://www.urbanrail.net/eu/de/do/dortmund-map.png>]

</

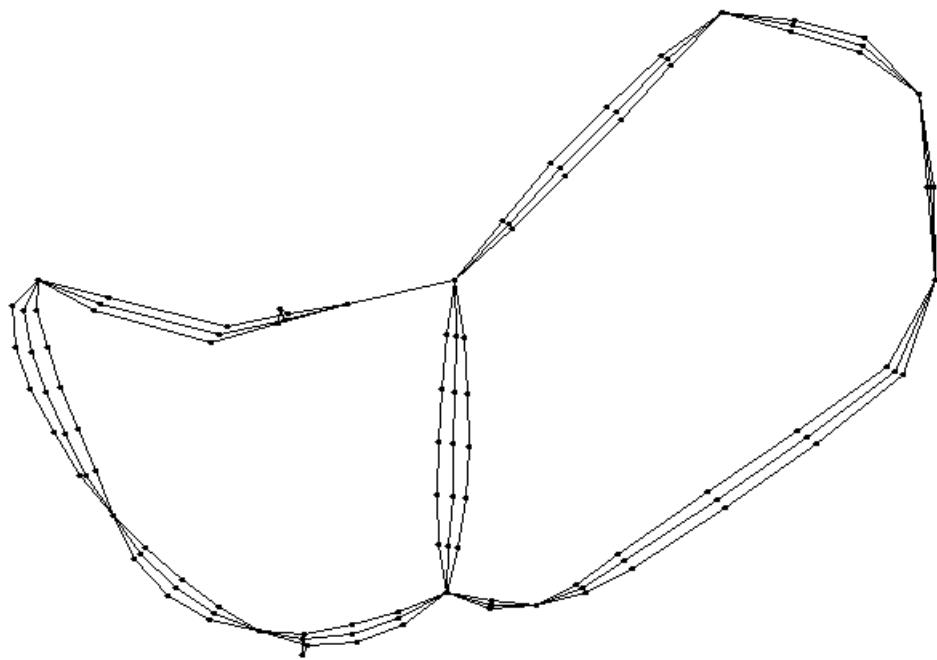


Abbildung 4.5: Dortmund MetroMap [<http://www.urbanrail.net/eu/de/do/dortmund-map.png>]

die Distanz zweier benachbarten Knoten so klein ist, dass sie sich lieber weiter verringert, als dass sich Knoten abstoßen wollen. Ein positiver Effekt der daraus folgt ist, dass sich kein Knoten mehr am Rand der Oberfläche sammelt. Ebenfalls sehr deutlich wird der komplette Verlust der topologischen Eigenschaften der Verbindungen. Jegliches umgangenes Hindernis ist in der Zeichnung nicht mehr zu sehen. Die Knoten haben sich komplett von ihrem Ursprung entfernt, dass die kompletten Merkmale und Eigenarten verschwunden sind. Es muss darauf geachtet werden, dass diese noch zu erkennen sind und die optimale Distanz zweier benachbarter Knoten nicht zu sehr verringert wird.

Es können demnach viele Probleme nur mit Anpassung der optimalen Distanz gelöst werden, doch es folgen daraus Probleme die den Verlust der Topologie und geographischen Merkmale zur Folge haben. Ebenfalls wird die Parallelität der Leiterseile, welche einen guten Überblick der Verbindung verschafft, durch die optimale Distanz verringert.

Durch das Verändern der wirkenden Kräfte, können wiederum Nachteile der Veränderung der optimalen Distanz ausgeglichen werden. Wie bereits angesprochen,

Eingabe: $G := (V, E)$
Ausgabe: $G := (V, E)$

```

1: for  $i := 1 \leq \text{iterations}$  do
2:   ..
3:   for  $e_{v_i, v_j} \in E$  do
4:      $\Delta := p_{v_i} - p_{v_i}^0;$ 
5:      $d_{v_i} := d_{v_i} - (\Delta / |\Delta|) \cdot f_{v_i, p_{v_i}^0}^a(|\Delta|);$ 
6:   end for
7:   ..
8: end for
```

Algorithmus 4.1: Erweiterung des Spring-Algorithmus

verliert die abstoßende Kraft zwischen den Knoten ihre Relevanz im Vergleich zur optimalen Distanz, sofern diese zu gering wird. Es ist sehr leicht diese Kraft etwas zu modifizieren und höher zu setzen und damit wieder mehr Konturen in das Layout bekommen. Die Topologie wird mehr beachtet und dennoch sammeln sich die Knoten nicht mehr Rande der Oberfläche. Wird die abstoßende Kraft des Algorithmus um etwa 40% erhöht, so gibt es jedoch weitaus mehr Ausreißer, die die eigentliche Struktur komplett verlassen und nach außen driften.

Um das Problem der Ausreißer besser in den Griff zu bekommen und dennoch wieder mehr Topologie in dem Graphen zu bringen, ist es besser eine gänzlich neue Kraft der Abstoßenden und Anziehenden hinzuzufügen. Eine Kraft die nur sicherstellt, dass die Topologie des Graphen beibehalten wird. Das bedeutet die Knoten dürfen sich nicht zu weit von ihrer ursprünglichen Position entfernen. Die Kraft sollte stärker wirken, je weiter die Knoten von ihrer Startposition entfernt sind. Ähnlich der anziehenden Kraft zwischen zwei Knoten, die bereits vorhanden ist. Es müssen sich demnach alle Knoten ihre Startposition merken und stets mit ihrer momentanen Position vergleichen. Mithilfe dieser beiden Informationen, kann eine weitere anziehende Kraft hinzugefügt werden, welche mit diesen beiden Positionen arbeitet. Das Vorgehen dabei ist relativ simpel. Wird die anziehende Kraft so modifiziert, dass der zweite Knoten nicht mehr ein Knoten ist, sondern die ursprüngliche Position p^0 des ersten Knotens. So würde es nun eine anziehende Kraft von einem Knoten zu seiner Startposition geben. Diese Kraft muss natürlich auch jede Iteration auf jedem Knoten wirken und genau wie bei der ursprünglichen anziehenden Kraft, dem Vektor des Knotens, welcher die Richtung der Bewegung bestimmt hinzugefügt werden. Dem Algorithmus aus 3.1 wird demnach eine neue anziehende Kraft hinzugefügt,

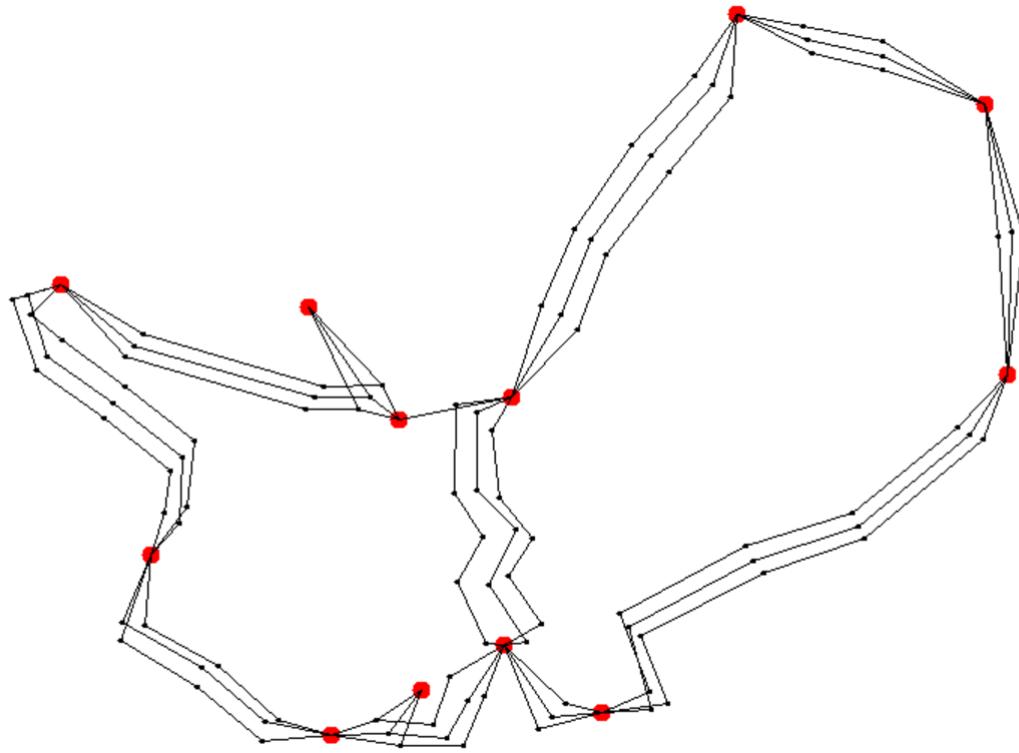


Abbildung 4.6: Dortmunder MetroMap [<http://www.urbanrail.net/eu/de/do/dortmund-map.png>]

welche im Algorithmus 4.1 durch Zeile 3-6 veranschaulicht wird. Sie ist sehr ähnlich der anziehenden Kraft, nur dass der zweite Knoten durch die ursprüngliche Position p^0 ersetzt wird.

Es existieren nun jedoch wieder zu viele anziehende Kräfte, sodass die abstoßende Kraft in Relevanz zu schwach wirkt und die Knoten sich wieder näher zueinander hinziehen als sie sollten. Um das Problem der Ausreißer weitestgehend zu verhindern, wird die abstoßende Kraft nicht erhöht, sondern die beiden anziehenden Kräfte verringert. Experimentell hat sich bewährt, dass eine Schwächung der anziehenden Kraft zwischen benachbarten Knoten von etwa 20% und eine Schwächung von 50% der haltenden Kraft, die einen Knoten an ihre ursprüngliche Position hält, die schönsten Layouts bringt, die sich am besten dem Problem angepasst haben.

Auf der Abbildung 4.6 ist das Endergebnis der Arbeit zu sehen. Fast alle Knoten haben einen guten Abstand zueinander, welches auch in Abbildung 4.7 dargestellt wird. Die Verbindungen sind noch sehr ähnlich ihrem ursprünglichen geographischen Verlauf. Es ist kein Problem zu erkennen, wo die umgangenen Hindernisse sich befanden.

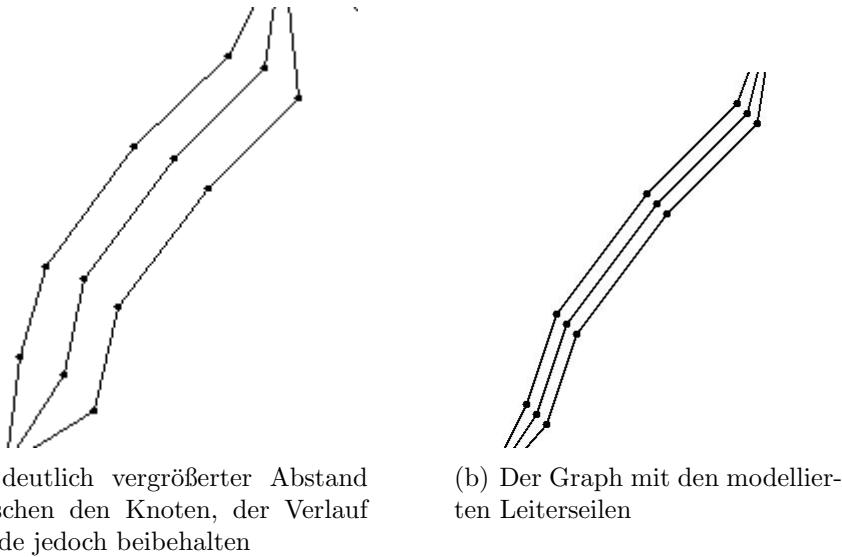
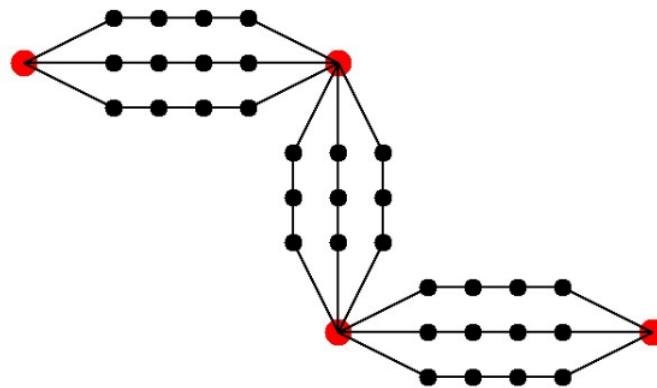


Abbildung 4.7: Modellierung der Leiterseile

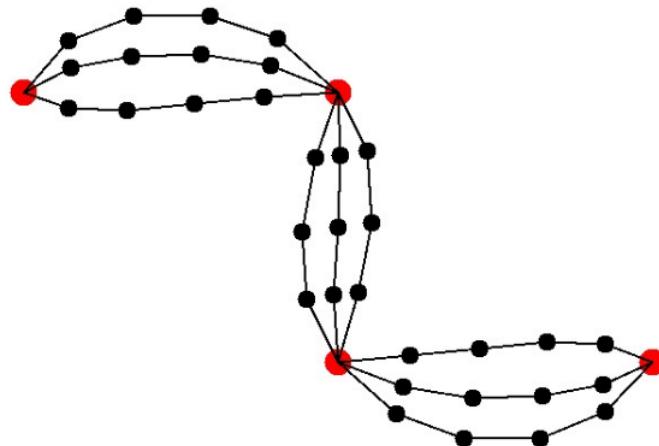
den. Drei Ausreißer sind noch zu sehen, welche sich jedoch nur sehr schwer entfernen lassen. Die Parallelität zwischen den Leiterseilen ist sehr gut zu sehen. Es ist nun möglich bei jeder Verbindung die Anzahl der Leiterseile direkt der Zeichnung zu entnehmen.

4.4 Bildung nahezu paralleler Leiterseile mittels dem Spring-Embedder

Um eine noch bessere Sichtbarkeit der Parallelität zu schaffen, wurde eine weitere Kraft hinzugefügt. Diese Kraft verändert nicht nur die Parallelität der Leiterseile, sondern hat auch einen positiven Einfluss auf die Knoten, die in direkter Verbindung mit den größeren unbeweglichen Städten stehen. Das eigentliche Ziel war es die Bildung von Kurven in den Verbindungen zu verhindern. Durch die optimale Distanz bildet der Algorithmus eine runde Positionierung der Knoten um die unbeweglichen Städte. In der Abbildung 4.8 wird dieses Vorgehen anhand eines Beispielgraphen verdeutlicht. Die Städte und somit unbeweglichen Knoten sind rot. Die anderen Knoten haben eine kurvige Form in den Verbindungen. Dadurch wurde die Parallelität der Leiterseile aufgehoben. Ein Ziel war es jedoch, die Leiterseile so parallel wie möglich zu zeichnen, damit es leicht möglich ist zu sehen, in welcher Anzahl diese auftreten und wie diese verlaufen. Das folgende Verfahren verschiebt die Knoten so, dass keine kurvigen Kanten durch Anwendung des Algorithmus auftreten.



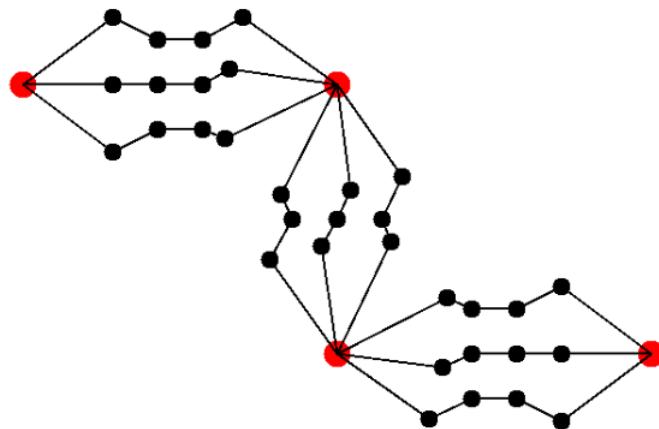
(a) Die roten Knoten repräsentieren unbewegliche Städte, zwischen ihnen verlaufen genau drei Leiterseile, die noch einen parallelen Verlauf aufweisen



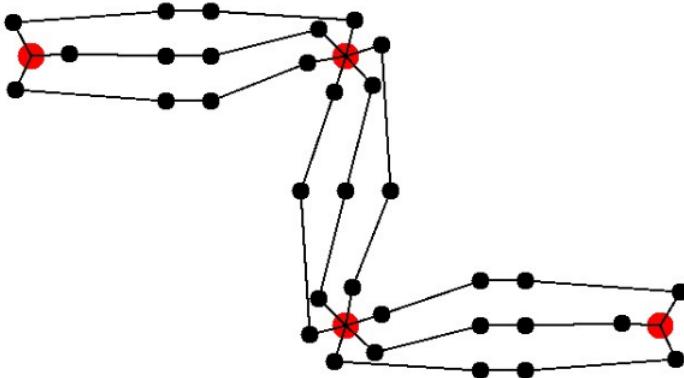
(b) Nach der Anwendung des Algorithmus auf den Graphen geht die parallele Struktur verloren

Abbildung 4.8: Aus der parallelen Struktur des Graphen wird durch den Algorithmus eine kurvige

Es ist eine ziemliche Herausforderung, einem Algorithmus, der grundsätzlich eher kurvige Strukturen erzeugt dazu zu bringen parallel zu bleiben. Die Grundidee für das Folgende Vorgehen war es, die Städte, die direkt an einem Knoten liegen, sich anziehen zu lassen. Dadurch ist die bestmögliche Lösung, zur Platzierung der Knoten, eine indem die Knoten den geringsten Abstand zu einander haben und genau das ist der Fall, wenn sie sich direkt auf einer Gerade befinden. Ziehen sich diese jeweiligen Knoten an, so passiert es, dass sich die Knoten an den Städten zu nahe kommen, was eine unschönes Layout zur Folge hätte.



(a) Die roten Knoten repräsentieren unbewegliche Städte, zwischen ihnen verlaufen genau drei Leiterseile, die noch einen parallelen Verlauf aufweisen



(b) Nach der Anwendung des Algorithmus auf den Graphen geht die parallele Struktur verloren

Abbildung 4.9: Aus der parallelen Struktur des Graphen wird durch den Algorithmus eine kurvige

Der erste Schritt besteht darin, die Knoten, die in direkter Verbindung zu einem unbeweglichen Knoten stehen, sich abstoßen zu lassen. Diese Knoten sollen sich rundum an ihrem unbeweglichen Knoten verteilen. In der Abbildung 4.9(a) wird dargestellt wie sich die Knoten jeweils um den unbeweglichen Knoten verteilen und sich stark voneinander abstoßen. Damit sie sich jedoch nicht zu weit entfernen, wurde die optimale Distanz zwischen dem beweglichen und unbeweglichen Knoten sehr stark gesetzt, was zur Folge hat, dass sich diese auch wirklich um den Knoten verteilen und sich nicht immer weiter davon entfernen. Dies ist zu sehen in der Abbildung

4.9(b). Besonders gut zu sehen ist die perfekt symmetrische Verteilung der beweglichen Knoten um den unbeweglichen herum. Diese Knoten stoßen sich gegenseitig ab, werden aber stark durch den beweglichen angezogen, was diese Struktur zur Folge hat. Auf die anderen Knoten wird erst später ein justierende Kraft angewendet, deswegen liegen diese noch auf ihrer ursprünglichen Position. Durch diesen Ansatz müssen die noch unbewegten Knoten am Ende nur gut zwischen den zwei bis jetzt bewegten Knoten verteilt werden.

Da die Knoten nun sehr symmetrisch um die Städte positioniert wurden, müssen nun die Knoten gefunden werden, die sich direkt an einem unbeweglichen Knoten befinden und ihren direkten Partner am anderen Ende des Leiterseiles. Wurden diese beiden Knoten gefunden, die sich auf einem Leiterseil befinden und sich auch durch eine Kante in direkter Verbindung mit einem unbeweglichen Knoten stehen, so ist es möglich ihnen eine sehr leichte anziehende Kraft zu geben. Diese anziehende Kraft sorgt dafür, dass sich die Knoten nun nicht mehr nur um ihre Städte verteilen. Es wird die Struktur der Leiterseile wieder deutlicher. Dieser Vorgang wird in der Abbildung 4.10(a) gezeigt. Anstatt, dass sie sich nur noch um den Knoten verteilen, führt die leichte Einführung einer anziehenden Kraft nun dazu, dass sie wieder die Leiterseil ähnliche Form bekommen.

Die nicht bewegten Knoten in der Abbildung 4.10(a) werden nun auf die Verbindung ihrer Leiterseile gebracht und mit genug Abstand, damit sie sich gut verteilen. Dies führt zu der Abbildung 4.10(b), wo sich die bisher nicht bewegten Knoten nun zwischen dem Start- und Endknoten, welche den Start und das Ende des Leiterseiles modellieren, positionieren. Dies passiert nur durch die Einführung der anziehenden und abstoßenden Kraft, wie es bereits im Algorithmus 3.1 passiert ist. Hier ist bereits gut zu erkennen, dass die Leiterseile wieder sehr gerade verlaufen und keine Kurve bilden.

Im letzten Schritt wird versucht, die Leiterseile sich etwas paralleler ausrichten zu lassen, indem die optimale Distanz zwischen den Knoten verringert wurde, was zur Folge hat, dass sich die Knoten besser auf den Leiterseilen verteilen und sich generell etwas mehr zusammenziehen. Dieser Schritt ist in der Abbildung 4.10(c) zu sehen. Die Leiterseile dieses Graphen liegen nun fast komplett parallel zueinander und die anfänglich kurvige Form ist verschwunden.

4.5 Einbettung der Knoten in Felder

Es ist möglich, die Oberfläche in einzelne kleine Felder einzuteilen. In jedes dieser Felder passt anschließend genau ein Knoten. Sind nun alle Feder gleich groß und die komplette Oberfläche symmetrisch eingeteilt, so ist das Überlappen von Knoten ausgeschlossen. Voraussetzung dafür ist:

1. es existiert immer genau nur ein Knoten pro Feld
2. es gibt nicht mehr Knoten als Feder.

Das Vermeiden von überlappenden Knoten ist kein großes Problem, denn die Knoten würden sich durch ihre abstoßende Kraft wieder von einander entfernen, doch das Layout wird, durch die strikte Verteilung der Knoten auf feste Felder, deutlich schöner und übersichtlicher. Die Dortmunder MetroMap in der Abbildug 2.11 veranschaulicht dies einmal. Die Knoten liegen alle in symmetrisch eingeteilten Feldern, was eine ordentliche Darstellung der Informationen ermöglicht. Werden die Knoten dabei so positioniert, dass nur noch Knoten mit orthogonal ein- und ausgehenden Kanten existieren, so trägt dies zu einem deutlich besseren Layout bei. Der Graph in der Abbildung 4.8(a) hat teilweise eine solche Darstellung, die Knoten liegen alle in Feldern aufgeteilt, jedoch sind einige Kanten nicht orthogonal zum Knoten. Diese Anordnung der Knoten sieht übersichtlicher aus, als die Anordnung in Abbildung 4.8(b), indem die Knoten bereits ohne Einteilung in Feldern positioniert wurden.

Ein wesentlicher Nachteil dieser Einteilung ist es, dass den Knoten anschließend nicht mehr die komplette Fläche zur Positionierung zur Verfügung steht. Ein Beispiel: existiert ein Knoten v_i mit der Position $(5/5)$ und ein Knoten v_j mit der Position $(15/15)$ so ist eine Positionierung der Knoten, um möglicherweise eine Steigerung der Ästhetik zu erhalten, in den kompletten Bereich um diese Felder gesperrt. Ist die Feldgröße nun etwa $10 \cdot 10$ und die Knoten würden immer auf der Position direkt mittig liegen, so könnte es kein Knoten auf den Punkt $(1/1)$, $(18/14)$ oder $(4/5)$ geben. Anstatt 100 möglichen Positionierungen in dem Bereich $(0/0)$ bis $(10/10)$, existiert nur noch eine. Das führt zu einem enormen Verlust der Genauigkeit und verhindert mögliche bessere Positionierungen.

Die Einbindung dieses Ansatzes in den Spring-Embedder ist auch sehr schwer. Es müsste zum einen entschieden werden, welche Größe die einzelnen Felder haben,

dann müsste ein initiales Layout gefunden werden, indem sich die Knoten vor Anwendung des Algorithmus schon befinden in dieser Struktur befinden. Knoten müssten ebenso in der Lage sein, sich um größere Strecken pro Iteration bewegen zu können, denn sonst wären Knoten, die umringt von anderen Knoten liegen überhaupt nicht in der Lage sich neu zu positionieren. Ganze Sammlungen von Knoten in einem Bereich, wären nicht in der Lage sich ihren Kräften nach zu bewegen, was dazu führen würde, dass weitaus mehr Iterationen gebraucht werden. Die äußeren Knoten würden sich erst weiter nach außen verteilen, anschließend können die inneren Knoten ihren Platz einnehmen und so weiter. Daraus folgend wurde dieser Ansatz nicht realisiert, es ist jedoch trotzdem ein wichtiger Aspekt einer MetroMap, die Knoten möglichst orthogonal zu platzieren und sollte erwähnt werden.

4.6 Evaluierung und Fazit

Nun folgt die Bewertung, des modifizierten Algorithmus aus Kapitel 4.3, welcher das Layout in Abbildung 4.6 erstellt hat. Die Anforderung bestand darin, ein MetroMap-konformes-Layout zur Visualisierung des Höchstspannungsnetzes zu erstellen. Dabei mussten viele besondere Eigenschaften des Höchstspannungsnetzes berücksichtigt werden, die sich zum Teil sehr von einem Bus- oder Bahnnetz unterscheiden und damit auch die resultierende MetroMap beeinflusste. Der wesentliche Aspekt des Labelings, welcher auch einen sehr großen Einfluss auf die Ästhetik einer MetroMap hat, musste dabei nicht bearbeitet werden.

Anfangend mit der Modellierung des Höchstspannungsnetzes als einfachen Graphen, waren die Anzahl der Leiterseile von besonderer Bedeutung. Das Erhalten eines Graphen aus einem Teil des Höchstspannungsnetzes war ein sehr arbeitsintensives Vorgehen. Die Positionen der einzelnen Städte wurden aus der Karte abgelesen und ein jeweils repräsentierender Knoten wurde erstellt. Das bedeutet es mussten alle Knoten manuell erstellt werden und die Kanten ebenso. Dieses Verfahren könnte deutlich verbessert werden, indem die Knoten automatisch abgelesen und erstellt werden, was ein deutlich schnelleres Erhalten eines repräsentierten Graphen zur Folge hätte. Würde dies deutlich schneller gehen, so wäre es möglich anstatt eines Teiles auch das komplette Höchstspannungsnetz zu modellieren. Es Bedarf demnach noch die Implementierung einer Möglichkeit die Knoten und Kanten automatisch der Karte zu entnehmen.

Die Wahl auf den Spring-Embedder zur Bewältigung der Anforderungen fiel schnell, denn er war:

- schnell und leicht zu implementieren,
- sehr effizient,
- anpassbar sowie modifizierbar.

Diese Eigenschaften wurden bereits bei der ersten Implementierung bemerkbar und ließen es schnell zu, kleinere Graphen mittels des Algorithmus in eine visuell ansprechendere Darstellung zu bringen. Auf den repräsentierenden Graphen des Höchstspannungsnetz jedoch angewendet, waren viele Veränderungen und Anpassungen nötig um einen bessere Ästhetik zu erhalten. Hier wurden vor allem die allgemeinen Unterschiede zur MetroMap deutlich, diese waren:

- eine deutlich striktere Beibehaltung der topologischen und geographischen Eigenschaft des Graphen,
- das Verteilen der Knoten auf die komplette Oberfläche soll nicht stattfinden,
- es existieren Knoten, die sich nicht bewegen lassen,
- durch das modellieren der Leiterseile existiert eine bestimmte Struktur innerhalb des Graphen, die noch immer sichtbar bleiben soll.

Nun musste der Algorithmus den Anforderungen entsprechend angepasst werden. Dabei gab es Punkte die waren sehr leicht umzusetzen, wie das modellieren von unbeweglichen Städten, die beim bewegen der Knoten einfach ausgelassen wurden, jedoch gab es auch kompliziertere Sachen, wie die Beibehaltung der Topologie und der geographischen Eigenschaften.

Die Knoten an ihre Startposition zu binden, mittels einer anziehenden Kraft, hat schnell und gut geklappt. Das Ergebnis war nicht optimal, genügte jedoch den Anforderungen, eine gewisse Wiedererkennbarkeit zu erhalten, neben der, die, die unbeweglichen Städte brachten.

Das sich die Knoten nicht überall Verteilen, wurde sehr gut gelöst, durch die Optimierung der optimalen Distanz. Diese war nach einigem experimentieren gefunden und die Knoten sammelten sich nicht mehr Rande des Bildes nach der Anwendung

des Algorithmus.

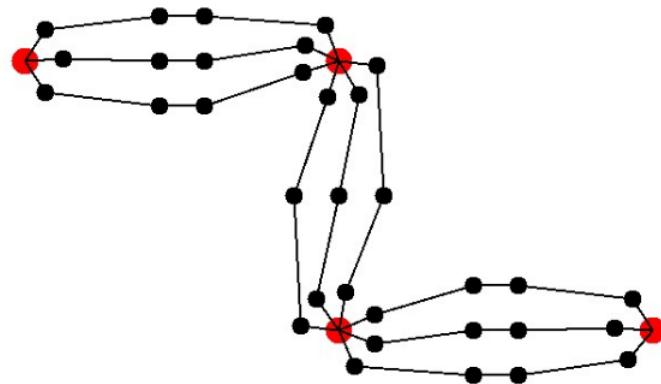
Das Modellieren und die gute Darstellung der Leiterseile, war die größte Herausforderung, die es zu lösen galt. Davon abgesehen, dass die genaue Anzahl der Leiterseile nicht benutzt wurde, sondern stets als drei angenommen war, war es sehr schwierig mit dieser Struktur umzugehen. Der Algorithmus musste den Abstand der einzelnen Leiterseile untereinander stark vergrößern, durfte dabei aber den eigentlichen Verlauf der Verbindungen nicht verlieren. Ebenso sollten die Verläufe nicht zu einer Kurve werden, was mittels dem Vorgehen in Kapitel 4.4 gut gelöst wurde.

Das Auftreten der Ausreißer, fiel erst bei der Anwendung auf den Graphen des Höchstspannungsnetzes auf. Sie entstanden, bei den kleineren Graphen mit einem zufälligen Layout und weniger Modifikationen, nicht. Je mehr die Relevanz der optimalen Distanz aus dem Algorithmus genommen wurde, desto mehr Ausreißer entstanden. Die optimale Distanz stand demnach in direkter Verbindung mit der Bildung der Ausreißer und bildeten sich zu viele, so musste die optimale Distanz wieder angepasst werden. Mit der Anpassung der optimalen Distanz war es jedoch oft nicht möglich die Ausreißer komplett zu vermeiden. Im fertigen Graphen, waren auch noch wenige Knoten zu sehen, welche ein leichtes Verhalten zum Ausreißer aufweisen. Auch durch weiteres experimentieren mit den Kräften, war es nicht möglich diese komplett zu verhindern.

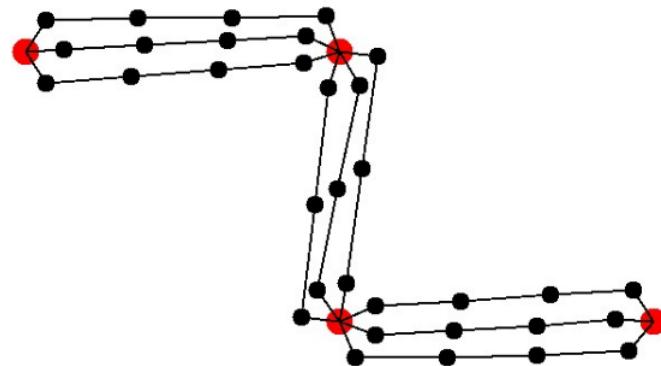
Letztendlich war die Wahl des Algorithmus den Spring-Embedder zu nehmen, zur Visualisierung des Höchstspannungsnetzes, durch die vielen möglichen Modifikationen, eine Gute. Die Laufzeit dieses Algorithmus beträgt in O-Notation $|V|^2 + |E| + |V|$. Diese kommt zur Stande durch das zweimalige Durchgehen aller Knoten, zur Berechnung der abstoßenden Kraft, somit $|V|^2$. Die anziehende Kraft benötigt einen Durchgang der Liste aller Kanten was $|E|$ entspricht. Das letztendliche Bewegen der Knoten passiert in $|V|$. Der Algorithmus gehört damit zu den schnellsten, sofern dieser unmodifiziert bleibt. Durch unüberlegte Erweiterungen, wie die in Kapitel 4.4 kann die Laufzeit jedoch deutlich schlechter werden. In dem ersten Ansatz das Problem so zu lösen, mussten die Kanten drei mal jeweils durchgegangen werden, eine Laufzeit von $|E|^3$ also. Diese resultiere vor allem aus der Findung der beiden wichtigen Knoten, indem über die Kanten iteriert wurde.

Aus dieser Arbeit ist zu schließen, dass sich Algorithmus mit den richtigen Anpas-

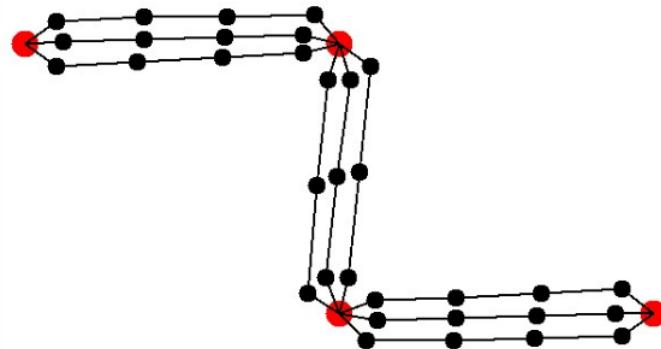
sungen auch für viele andere Probleme, die sich mit der Visualisierung eines Graphen beschäftigen, benutzen lassen könnte. Das Layout des fertigen Graphen kann noch weiter verbessert werden. Ein Beispiel wurde bereits, mit der Orthogonalität der einzelnen Knoten zueinander, angesprochen. Jedoch entspricht der resultierende Graph bereits jetzt, bis auf einige Ausreißer, den an ihn gestellten Anforderungen.



(a) Die roten Knoten repräsentieren unbewegliche Städte, zwischen ihnen verlaufen genau drei Leiterseile, die noch einen parallelen Verlauf aufweisen



(b) Nach der Anwendung des Algorithmus auf den Graphen geht die parallele Struktur verloren



(c) Die roten Knoten repräsentieren unbewegliche Städte, zwischen ihnen verlaufen genau drei Leiterseile, die noch einen parallelen Verlauf aufweisen

Abbildung 4.10: Aus der parallelen Struktur des Graphen wird durch den Algorithmus eine kurvige

A Weitere Informationen

Abbildungsverzeichnis

1.1	Karte des deutschen Höchstspannungsnetzes	5
1.2	MetroMap der Bahnverbindungen in Singapur	5
2.1	Prinzip der Stromerzeugung	8
2.2	Die vier verschiedenen Leitungstypen	9
2.3	Karte des deutschen Höchstspannungsnetzes	10
2.4	Karte des europäischen Verbundnetzes	11
2.5	Freileitungsmast mit Leitern und Erdseil	12
2.6	Einfacher Graph mit Städten als Knoten und deren Höchstspannungsleitungen als Kanten	14
2.7	Verschiedene Zeichnungen eines planaren Graphen	15
2.8	Übergang von der Karte des Höchstspannungsnetzes zu einem Graphen	20
2.9	Modellierung der Leiterseile	21
2.10	Erzeugung der neuen Knoten zur Modellierung der Leiterseile	22
2.11	Dortmunder MetroMap	22
2.12	8 Mögliche Platzierungen der Beschriftung	23
3.1	Darstellung als Stahlkugeln und Federn eines Graphen	26
3.2	Eine Beispielhafte Abbildung der Knoten mit einer nahezu perfekten Verteilung	29
3.3	Wirkende Kräfte im Graphen	32
3.4	Verhältnis der Kräfte zueinander und deren Einfluss auf die optimale Distanz	33
3.5	Lokales Minima-Problem	34
3.6	Bildung von Ausreißer	35
3.7	Resultat von Ballenbildung und dessen Auswirkung auf das Layout .	36
4.1	Dortmunder MetroMap	41
4.2	Dortmunder MetroMap	42
4.3	Dortmunder MetroMap	43
4.4	Dortmunder MetroMap	44

4.5	Dortmunder MetroMap	45
4.6	Dortmunder MetroMap	47
4.7	Vergleich der Leiterseile	48
4.8	Aus der parallelen Struktur des Graphen wird durch den Algorithmus eine kurvige	49
4.9	Aus der parallelen Struktur des Graphen wird durch den Algorithmus eine kurvige	50
4.10	Aus der parallelen Struktur des Graphen wird durch den Algorithmus eine kurvige	57

Algorithmenverzeichnis

3.1	Spring-Algorithmus	30
4.1	Erweiterung des Spring-Algorithmus	46

Quellcodeverzeichnis

Literaturverzeichnis