

## Bachelor-Arbeit

### MetroMap-Layout-konforme Visualisierung von Höchstspannungsnetzen

Robin Möhring

24. Juli 2017

#### **Gutachter:**

Prof. Dr. Heinrich Müller

M.Sc. Dominic Siedhoff

Lehrstuhl Informatik VII  
Graphische Systeme  
TU Dortmund



# Inhaltsverzeichnis

<b>Mathematische Notation</b>	<b>1</b>
<b>1 Einleitung</b>	<b>3</b>
1.1 Motivation und Hintergrund . . . . .	3
1.2 Aufbau der Arbeit . . . . .	3
<b>2 Spring-Algorithmus</b>	<b>5</b>
2.1 Spring-Algorithmus - Vorgehen . . . . .	6
2.2 Spring-Algorithmus - Erweiterbarkeit . . . . .	8
<b>3 Das Kapitel 3</b>	<b>11</b>
3.1 Kapitel 3 - Unterkapitel 1 . . . . .	11
3.2 Kapitel 3 - Unterkapitel 2 . . . . .	11
<b>A Weitere Informationen</b>	<b>13</b>
<b>Abbildungsverzeichnis</b>	<b>15</b>
<b>Algorithmenverzeichnis</b>	<b>17</b>
<b>Quellcodeverzeichnis</b>	<b>19</b>
<b>Literaturverzeichnis</b>	<b>21</b>



# 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.

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.

## **1.2 Aufbau der Arbeit**



Abbildung 1.1: Karte des deutschen Höchstspannungsnetzes



Abbildung 1.2: MetroMap der Bahnverbindungen in Singapur



## 2 Spring-Algorithmus

In diesem Kapitel werden die Grundlagen des Spring-Algorithmus erklärt. Es geht um die Verwendung dieser Algorithmen, das grundsätzliche Vorgehen und die Erweiterbarkeit.

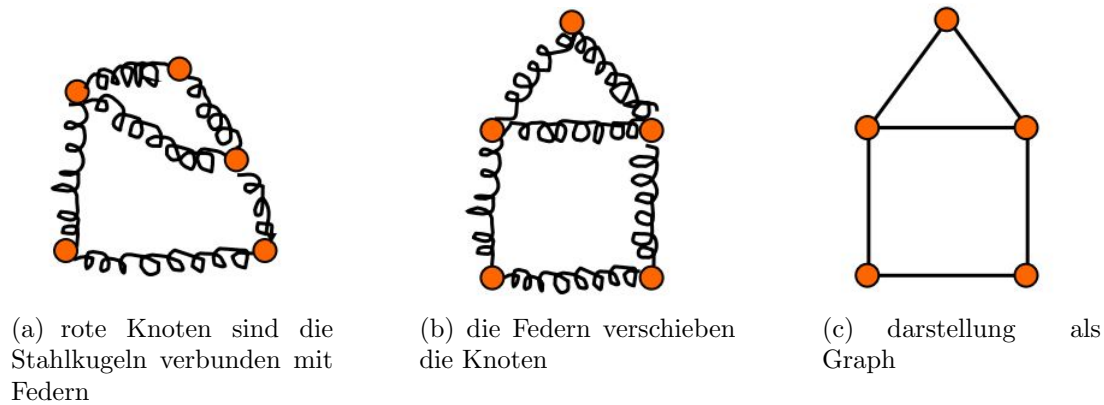
Das Problem der Darstellung basiert auf der Platzierung der Kanten sowie Knoten um eine möglichst ästhetische Zeichnung des Graphen zu erhalten, die gut lesbar und verständlich ist. Die grundlegenden Kriterien für eine ästhetischen Zeichnung eines Graphen sind:

1. die Knoten sollten gleichmäßig verteilt werden
2. minimale Überschneidung der Kanten
3. einheitliche Länge der Kanten
4. möglichst symmetrisch
5. alles soll sich innerhalb der Fläche befinden[3].

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.

Im Folgenden wird es um ein Verfahren gehen, welches auf ein Modell der Physik basiert, indem Stahlkugeln für die Knoten, und Federn für die Kanten stehen. Durch die Federn wirken Kräfte auf die Kugeln ein, wodurch sich diese verschieben. Dies passiert solange, bis die Federn ihre optimale Länge haben. Die optimale Länge  $k$  wird durch die Anzahl der Knoten sowie der zur Verfügung stehender Fläche  $A$  bestimmt:

$$k = \sqrt{A/|V|} \tag{2.1}$$



**Abbildung 2.1:** Darstellung als Stahlkugeln und Federn eines Graphen

$$A = W * L. \quad (2.2)$$

$W$  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.

Bildet die bisherige Platzierung der Kanten und Knoten einen planaren Graphen, so wird die neue die 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.

## 2.1 Spring-Algorithmus - Vorgehen

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

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

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

```

for  $i := 1 \leq \text{iterations}$  do
  for  $v_i \in V$  do
     $d_{v_i} := 0$ ;
    for  $(v_j \in V)$  do
      if  $v_i \neq v_j$  then
         $\Delta := p_{v_i} - p_{v_j}$ ;
         $d_{v_i} := d_{v_i} + (\Delta/|\Delta|) \cdot f_{v_i, v_j}^r(|\Delta|)$ ;
      end if
    end for
  end for

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

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

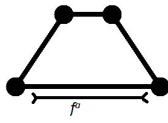
```

**Algorithmus 2.1:** Spring-Algorithmus

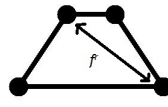
$$f_{v_i, v_j}^a(x) = x^2/k \quad (2.3)$$

$$f_{v_i, v_j}^r(x) = -k^2/x. \quad (2.4)$$

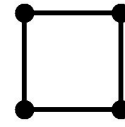
Der Vektor  $d_{v_i}$  gibt an in welcher Richtung der Knoten  $v_i$  bewegt wird. Während der Vektor  $p_{v_i}$  auf die momentane Position des Knoten  $v_i$  zeigt. Der Vektor  $\Delta$  ist die Differenz zwischen  $p_{v_i}$  und  $p_{v_j}$ , der Richtungsvektor von  $v_i$  nach  $v_j$ . Die Variable *iterations* gibt die Anzahl der Durchläufe an. Bei jedem neuen Durchlauf wird der Vektor  $d_v$  eines jeden Knotens  $v$  auf 0 gesetzt. Mit zwei For-Schleifen geht man jede Knotenkombination durch und berechnet die abstoßende Kraft  $f_{v_i, v_j}^r$  mithilfe der



(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 2.2:** Wirkende Kräfte im Graphen

Länge des  $\Delta$  Vektors. Anschließend wird der Bewegungsvektor auf den Einheitsvektor von  $\Delta$  multipliziert mit der berechneten Kraft gesetzt. Dadurch drücken sich die Knoten direkt voneinander weg.

Der zweite Schritt besteht darin die anziehenden Kräfte zwischen den benachbarten Knoten zu berechnen. Dazu wird jede Kante einmal durchgegangen und der Bewegungsvektor von jedem betroffenen Knoten wird wie bei der abstoßenden Kraft aufaddiert.

Im letzten Schritt des Algorithmus werden die Knoten nun in Richtung ihres Bewegungsvektors bewegt. Es wird darauf geachtet, dass die Knoten dabei nicht das Bild verlassen. 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 10 Iterationen wären dadurch keine weiteren Bewegungen der Knoten mehr möglich.

## 2.2 Spring-Algorithmus - Erweiterbarkeit

Eine besondere Eigenschaft dieses Algorithmus ist die leichte Erweiterbarkeit. Er kann leicht an viele verschiedene Probleme angepasst werden. Der Algorithmus 2.1 ist bereits eine erste Erweiterung des ursprünglichen Algorithmus. Beim Bewegen der Knoten im dritten Schritt wird sichergestellt, dass sich die Knoten weiterhin auf der zur Verfügung stehenden Fläche befinden.

Ebenfalls modifizierbar sind die Funktionen  $f_{v_i, v_j}^a$  und  $f_{v_i, v_j}^r$ . Im Kapitel 3 wird dies genutzt um den Graphen weiter anzupassen. Es ist leicht möglich noch weitere Kräfte

miteinzubeziehen oder sie ganz anderen Knoten zuzuweisen. Im späteren Verlauf wird es auch Knoten geben, die sich nicht bewegen sollen, weil sie zum Beispiel größere Städte oder zentrale Verbindungspunkte sind.



## **3 Das Kapitel 3**

### **3.1 Kapitel 3 - Unterkapitel 1**

### **3.2 Kapitel 3 - Unterkapitel 2**





## **A Weitere Informationen**



# Abbildungsverzeichnis

1.1	Karte des deutschen Höchstspannungsnetzes . . . . .	4
1.2	MetroMap der Bahnverbindungen in Singapur . . . . .	4
2.1	Darstellung als Stahlkugeln und Federn eines Graphen . . . . .	6
2.2	Wirkende Kräfte im Graphen . . . . .	8



# Algorithmenverzeichnis

2.1	Ein Algorithmus . . . . .	7
-----	---------------------------	---



# Quellcodeverzeichnis





# Literaturverzeichnis

- [1] ABRAMOWSKI, S.; MÜLLER, H.: *Geometrisches Modellieren*. Mannheim: B.I. Wissenschaftsverlag, 1991 (Reihe Informatik)
- [2] MÜLLER, H.; WEICHERT, F.: *Vorkurs Informatik: Der Einstieg ins Informatikstudium*. 2. Auflage. Vieweg+Teubner Verlag | Springer Fachmedien Wiesbaden GmbH, 2011