technische universität
dortmund

fakultät für informatik

Bachelor-Arbeit

MetroMap-Layout-konforme Visualisierung von Höchstspannungsnetzen

Robin Möhring

23. Juli 2017

Gutachter:

Prof. Dr. Heinrich Müller

M.Sc. Dominic Siedhoff

Lehrstuhl Informatik VII
Graphische Systeme
TU Dortmund

Inhaltsverzeichnis

Mathematische Notation

Notation	Bedeutung
$\mathbb{N}$	Menge der natürlichen Zahlen $1, 2, 3, \ldots$
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}^d$	d-dimensionaler Raum
$\mathcal{M} = \{m_1, \ldots, m_N\}$	ungeordnete Menge $\mathcal{M}$ von N Elementen m_i
$\mathcal{M} = \langle m_1, \ldots, m_N \rangle$	geordnete Menge $\mathcal{M}$ von N Elementen m_i
$\mathbf{v}$	Vektor $\mathbf{v} = (v_1, \ldots, v_n)^T$ 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

1.2 Aufbau der Arbeit

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:

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

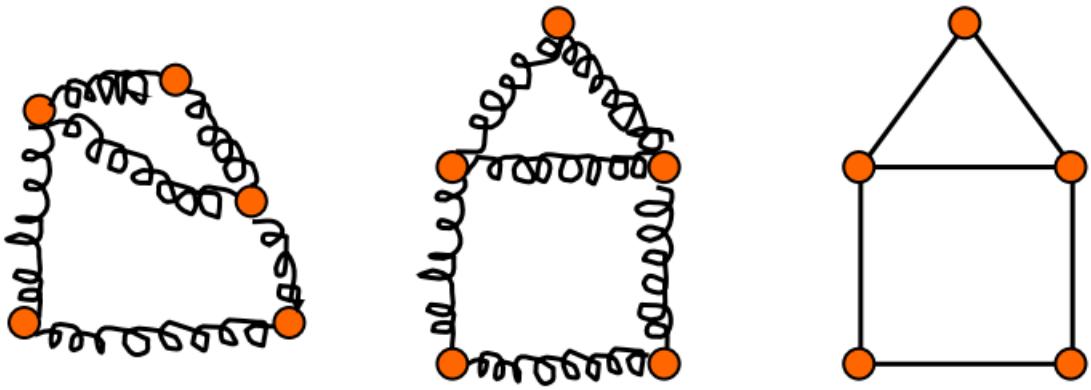

Abbildung 2.1: Darstellung der Kanten als Federn

$$A = W * L. \tag{2.2}$$

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 $f^r_{v_i,v_j}$ berechnet. Alle benachbarten Knoten erhalten eine anziehende Kraft $f^a_{v_i,v_j}$. 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 werden so oft wiederholt bis keine weitere Bewegung mehr stattfindet. Die Funktionen $f^a_{v_i,v_j}$ und $f^r_{v_i,v_j}$ sind wie folgt definiert:

Eingabe: $G := (V, E)$
Ausgabe: $G := (V, E)$
for $i := 1 \leq iterations$ **do**
for $v_i \in V$ **do**
$d_{v_i} := 0$;
for $(v_j \in V)$ **do**
if $v_i \neq v_j$ **then**
$\Delta := p_{v_i} - p_{v_j}$;
$d_{v_i} := d_{v_i} + (\Delta/|\Delta|) \cdot f^r_{v_i,v_j}(|\Delta|))$;
end if
end for
end for

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

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

Algorithmus 2.1: Fruchterman und Reingolds Algorithmus

$$f^a_{v_i,v_j}(x) = x^2/k$$

$$f^r_{v_i,v_j}(x) = -k^2/x. \tag{2.3}$$

Der Vektor d_{v_i} gibt an in welcher Richtung man den Knoten v_i bewegen möchte. 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 0 gesetzt. Mit zwei For-Schleifen geht man jede Knotenkombination durch und berechnet die abstoßende Kraft $f^r_{v_i,v_j}$ mithilfe

der Länge des Δ Vektors. Anschließend wird der Bewegungsvektor auf den Einheitsvektor von Δ 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.

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.

3 Das Kapitel 3

3.1 Kapitel 3 - Unterkapitel 1

3.2 Kapitel 3 - Unterkapitel 2

A Weitere Informationen

Abbildungsverzeichnis

Algorithmenverzeichnis

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