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.

2.1 Spring-Algorithmus - Einleitung und Verwendung

Das Problem der Darstellung eines Graphen existiert schon sehr lange. Es basiert auf der Platzierung der Kanten sowie Knoten um eine möglichst ästhetische Zeichnung des Graphen zu erhalten, die gut lesbar und verständlich ist. ([3] Seite 2-3) Um dies zu erreichen gibt es verschiedenste Ansätze. Im Folgenden wird es um ein Verfahren gehen, welches auf ein Modell der Physik zurückgreift.
Verschiedenste Partikel ziehen sich an oder stoßen sich ab. Man kann sich diese anziehende und abstoßende Kraft als eine Feder vorstellen. Existiert eine anziehende Kraft zwischen den Partikel so ist diese Feder gespannt, bei einer abstoßenden Kraft belastet. Nimmt man sich diese Modell zugrunde, so stehen die Partikel für Knoten und die Federn für Kanten. Diese richten sich anhand ihrer Kraft solange aus, bis keine weitere Energie mehr im System ist. ([4] Seite 1130 ff.) Das führt zu einer ästhetischen Zeichnung des Graphen. In der Abbildung 2.1 wird dieser Prozess einmal dargestellt.

2.2 Spring-Algorithmus - Pseudocode

Zwischen jedem Knotenpaar wird eine abstoßende Kraft f_{r} berechnet. Alle benachbarten Knoten erhalten eine anziehende Kraft f_{a}. Zwei Knoten $u, v \in V$sind benachbart wenn $uv \in E$ 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

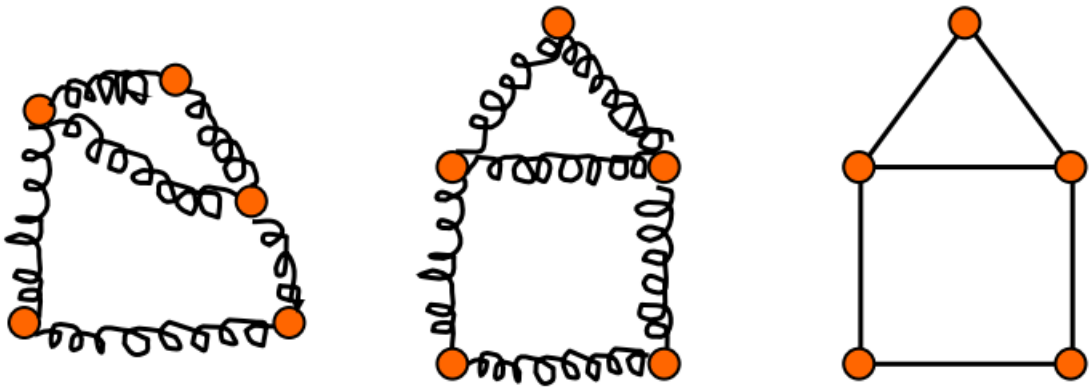

Abbildung 2.1: Darstellung der Kanten als Federn

2. benachbarten Knoten eine anziehende Kraft zuweisen

3. jeden Knoten seiner neuen Kraft nach bewegen

Diese drei Schritte werden werden so oft wiederholt bis das System keine übrige Energie mehr hat und sich kein Knoten mehr bewegt.
Fruchterman und Reingold haben ihre Funktionen f_r und f_a aus ihrer Arbeit 1984 wie folgt definiert:

$$f_\mathrm{a}(d) = d^2/k$$

$$f_\mathrm{r}(d) = -k^2/d \tag{2.1}$$

während k für die optimale Distanz zweier Knoten steht:

$$k = \sqrt{Area/|V|} \tag{2.2}$$

Area ist die zur Verfügung stehende Fläche:

$$Area = W * L \tag{2.3}$$

Eingabe: $G := (V, E)$
Ausgabe: $G := (V, E)$ mit besserer Positionierung
for $i := 1 \leq iterations$ **do**
for v **in** V **do**
$v.disp := 0$;
for (u **in** V) **do**
if $u \neq v$ **then**
$\Delta := v.pos - u.pos$;
$v.disp := v.disp + (\Delta/|\Delta|) * f_{\mathrm{r}}(|\Delta|))$;
end if
end for
end for

for e **in** E **do**
$\Delta := e.v.pos - e.u.pos$;
$e.v.disp := e.v.disp - (\Delta/|\Delta|) * f_{\mathrm{a}}(|\Delta|))$;
$e.u.disp := e.u.disp + (\Delta/|\Delta|) * f_{\mathrm{a}}(|\Delta|))$;
end for

for v **in** V **do**
$v.pos := v.pos + (v.disp/|v.disp|) * min(v.disp, t)$;
$v.pos.x := min(W/2, max(-W/2, v.pos.x))$;
$v.pos.y := min(L/2, max(\breve{}L/2, v.pos.y))$
end for
end for

Algorithmus 2.1: Fruchterman und Reingolds Algorithmus

2.3 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.