Balancing Performance

Nils Henrik Seitz* (Mat.Nr. 218205308)

Faculty of Computer Science
University of Rostock

Abstract

This is abstract.

Greedy approach realized by generator functions.

Pipeline. Basically our approach is described by Luboschik et al.[1].

1. Updating Coords of Vertex and Edges (from iGraph)
2. Conflict Detection
3. Adaptation/Optimization
4. Visualization

1. Labeling

1.1. Vertices

Beim Labeling der Knoten sind als geometrische Repräsentation ein Kreis, das heißt, x, y-Koordinaten und ein Radius, sowie der entsprechende Name des Knotens durch die Daten von `iGraph.js` gegeben.

Nun ist es Aufgabe des Labelings, den Text des Labels erkennbar und visuell ansprechend in Nähe des dazugehörigen Knotens zu platzieren. Die Breite und die Höhe eines Labels werden von der ausgewählten Schriftart (siehe Configuration) beeinflusst. Die Breite eines Labels hängt zusätzlich von dem entsprechenden Namen bzw. der Anzahl der Buchstaben ab.

So ergeben sich die Höhe und Breite des Labels (sowie der Text selbst) und es muss eine Position in x, y-Koordinaten in Abhängigkeit von dem entsprechenden Knoten gefunden werden. Zur Generierung der Position eines potentiellen Labels werden nacheinander verschiedene Verfahren verwendet, um möglichst effizient Labelpositionen zu generieren mit möglichst wenig Überdeckung der einzelnen potentiellen Positionen und so, dass niemals der dazugehörige Knoten überdeckt wird:

*ns464@uni-rostock.de

1.1.1. 4-Position-Model

Das 4-Position-Model positioniert die potentiellen Labels möglichst nahe an dem korrespondieren Knoten in den Himmelsrichtungen Nord-Ost, Nord-West, Süd-Ost und Süd-West. Die erste Position ist Nord-Ost, also rechts-oben, und von dort an wird entgegen x, y-Koordinaten ein weiteres Label generiert (wenn nötig).

Dieses und das nachfolgendene Verfahren sind durch die gängingen Standards der Kartografie inspiriert. Auch dort werden eben beschriebene Positionen generiert, allerdings in leicht veränderter Reihenfolge. ZITAT BITTE

1.1.2. 8-Position-Model

Im 8-Position-Model werden die potentiellen Labelpositionen in den Haupthimmelsrichtungen Ost, Nord, West, Süd generiert. Startpunkt ist Osten, also rechts, und wieder wird entgegen des Uhrzeigersinns ggf. ein weiteres Label generiert.

Weiterhin werden die Nord-/Süd-Positonen im Abstand einer Labelhöhe und die Ost-/West-Positonen im Abstand einer Labelbreite vom dazugehörigen Knoten positioniert. Dies wird gemacht, um Überdeckung mit den zuvor im 4-Position-Model generierten Positionen zu vermeiden, da, wenn zuvor das 4-Position-Model schon keine Position finden konnte, das 8-Position-Model ohne den Abstand dann wahrscheinlich ebenfalls keine Position finden würde.

Das führt dazu, dass vor allem die Ost-/West-Position eher weit entfernt von dem korrespondieren Knoten sind (da Labeltexte bzw. Namen in der Regel eher breit als hoch sind). Die Kartografie verwendet diese Positionen nicht. ZITAT BITTE

In der Implementation wurde dieses Problem dadurch behoben, dass eine "Hilfslinie", als visueller Indikator, das Label mit seinem entsprechenden Knoten verbindet, sodass die Zugehörigkeit schneller ersichtlich wird. Diese Technik wird im Spiral-Model nochmals angewandt, da hier dasselbe Problem besteht.

1.1.3. Slider-Model

Das Slider-Model ist ein aufwendigeres Verfahren als die vorhergehenden beiden. Die Idee hier ist, die Eckpunkte des 4-Position-Model zu nehmen und dann Labelpostionen dazwischen zu generieren, in dem man das potentielle Label stückweise zum nächsten Eckpunkt verschiebt. Der Inkrement-Wert zum Verschieben ist konstant (siehe Magic Constants).

Wie auch beim 4-Position-Model in die erste Position die Nord-Ost-Ecke und von dort aus wird parallel zur y-Achse die Position verändert bis die Süd-Ost-Ecke erreicht wird. Da angekommen wird dann parallel zur x-Achse in Richtung der Süd-West-Ecke verschoben, usw.

Das Verschieben erfolgt hier offensichtlich im Uhrzeigersinn, um dem Trend der vorherigen beiden Verfahren entgegenzuwirken.

1.1.4. Spiral-Model

Dieses Verfahren funktioniert gänzlich anders bisher genannten, da diese versuchen, Labelpositionen *karthesisch* zu finden, also durch Verschiebungen

bezüglich der x, y-Koordinaten ausgehend vom Mittelpunkt des entsprechenden Knotens. Das Spiral-Model versucht, Labelpositonen *polar* zu finden:

$$s(m) = \begin{pmatrix} d \cdot \cos(2\pi\sqrt{\frac{m}{m_{max}}} \cdot c) \\ \sin(2\pi\sqrt{\frac{m}{m_{max}}} \cdot c) \end{pmatrix} \cdot \sqrt{\frac{m}{m_{max}}} \cdot r, \quad m \in \{1, \ldots, m_{max}\}$$

Das heißt, potentielle Labelposition werden hier spiralförmig generiert, also in ansteigendem Abstand vom dazugehörigen Knoten und in fairer Verteilung hinsichtlich der Richtung der Labels, da die Spirale auch den Winkel fortwährend inkrementiert. Auch ist die Idee wieder, den Trend von zuvor zu durchbrechen und auf eine unkonventiellere Art Labelposition zu suchen in Regionen, die zuvor noch nicht beachtet wurden.

Die Parameter der Gleichung beeinflussen die Form und Ausrichtung der Spirale:

- d: Orientierung der Spirale (im/gegen den Uhrzeigersinn), ($d \in \{-1, 1\}$)
- c: Krümmung bzw. Anzahl der Rotation der Spirale, ($c \in \mathbb{N}$)
- m_{max}: Maximalanzahl der Punkte in der Spirale, ($m_{max} \in \mathbb{N}$)
- m: Der jeweilige m-te Punkt der Spirale

Ergänzend zu den vorherigen Verfahren ist es sinnvoll, auch das Spiral-Model zu nutzen, da hier schnell Abstand vom dazugehörigen Knoten gewonnen werden kann. Dies ist wichtig, da die Verfahren zuvor vorrangig in der Nähe des Knoten versuchen, eine Position zu finden. Das Spiral-Model wird aber nur genutzt, wenn die Verfahren dabei erfolglos geblieben sind.

Wie auch beim 8-Position-Model wird, ob des erweiterten Abstands zum korrespondieren Knoten, eine Hilfslinie genutzt, um die Verbindung für gefundene Labelpositionen und ihre Knoten deutlich zu machen.

1.2. Edges

identity etc.

1.2.1. Midpoint

2. Conflict Detection

2.1. Bounding Boxes

comparisions, totally left, right, above, below

2.2. Quadtree

less comparisions, logarithmic stuff, etc.

3. Adaptation/Optimization

3.1. Zoom Mode

Big Bounding box around, no quadtree used

3.2. Performance Modes

Depending on visible amount, more or less complex procedures are used.

4. Visualization

4.1. Drawing

something with HTML canvas

4.2. Animation

3 modes depending on (visible now X visible before) (both not visible doesnt matter)

5. Configuration

5.1. GUI

small gui blending in with iGraph, toggle labels, in-/decrease fontsize and amount of labels shown.

5.2. Magic Constants

`consts.js` for all magic constants, easily configurable

6. About the project

6.1. Technical Details

generator functions used, maps for fast access

6.2. Known Bugs

Flying Labels, Not Bug but consequence of implementation: Possible to have no labeling when everything is super dense

6.3. Suggestions

more conservative on animation side of things (keep positon as long as possible) =¿ comes from implementation to always try to label most important stuff first better conflict detection with circle/bbox for Vertices better labeling for edges according to standards (leads to increased complixity in conflict detection for most cases)

References

[1] Martin Luboschik, Heidrun Schumann, and Hilko Cords. Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Transactions on Visualization and Computer Graphics*, 14 (6):1237–1244, 2008. ISSN 1077-2626. doi:10.1109/tvcg.2008.152. URL https://doi.org/10.1109/TVCG.2008.152.

Figure 1: Sample image caption.