1 Graph Drawing: Layered Digraphs

1.1 Einführung

Eine wichtige Teilklasse von Graphen bei der Visualisierung sind gerichtete Graphen: durch sie werden z.B. zeitliche Abläufe oder (Produktions-)prozesse modelliert. Eine Visualisierung durch Kräfte-basiertes Zeichnen ist hier in der Regel ungeeignet; vielmehr sind Verfahren erforderlich, die die Kanten entlang einer *Hauptrichtung* orientieren und so den Fortgang, und die Phasen eines Prozesses sichtbar machen. Im folgenden gehen wir davon aus, dass die Hauptrichtung von oben nach unten verläuft. Dies führt zu folgenden ästhetischen Kriterien:

1. Kanten sollten nach Möglichkeit entlang der Hauptrichtung orientiert sein.
2. a) Knoten sollten gleichmässig verteilt sein.
   b) Lange Kanten sollten vermieden werden.
3. Es sollten möglichst wenige Kantenkreuzungen auftreten.
4. Kanten sollten so senkrecht wie möglich gezeichnet werden.

Wie "üblich" beim Graphzeichnen ist es offensichtlich unmöglich alle Kriterien gleichzeitig zu optimieren. In der Literatur hat sich deswegen ein Phasen-weises Vorgehensweise durchgesetzt, bei der in jeder Phase eines der Kriterien optimiert wird.

1.2 Entfernung von Kreisen

Ziel dieser Phase is es, so wenige Kanten wie möglich zu *invertieren* (d.h. aus einer Kanten (u, v) wird (v, u)), um den Graphen *azyklisch* zu machen. Das ist für die Ebenenberechnung der 2. Phase notwendig. Das Problem firmiert oft unter dem Namen *feedback arc set problem*:

Definition 1 feedback arc set problem*: Finde eine Menge min. Grösse $E_f \subset E$, so dass der Graph $(V, E \setminus E_f) = (V, E_{acyc})$ azyklisch ist.*

Die Kanten in $E \setminus E_{acyc}$ werden für die folgenden Berechnungen invertiert. Nach Ende der Layoutberechnung werden invertierten Kanten dann wieder zurückgedreht, um den Graphen korrekt darzustellen.Unglücklicherweise ist das Problem NP-hart. Im folgenden werden daher kurz einige Heuristiken beschrieben.

1.2.1 Schnelle Heuristiken

Eine äquivalente Formulierung des *feedback arc set* Problems ist die des *unweighted linear ordering* Problems:

Definition 2 unweighted linear ordering problem: *Finde eine Permutation bzw. Ordnung $o : V \to \{1, ..., |V|\}$ der Knoten von G, die die Anzahl der Kanten $(u, v) \in E : o(u) \geq o(v)$ minimiert.*

Die triviale Heuristik berechnet eine Ordnung durch eine Tiefen- bzw. Breitensuche, und löscht die Kanten mit $(u, v) \in E : o(u) \geq o(v)$. Zwar benötigt diese Heuristik nur Linearzeit ($O(m + n)$), offensichtlich ist die Anzahl der zu löschenden Kanten so aber überhaupt nicht beschränkbar.

Die Heuristik von Algorithmus 1.1 ist ebenso Laufzeit-effizient, garantiert aber dass $|E \setminus E_f| \geq 1/2 * |E|$.

Algorithmus 1.1

---

func GreedyAcyclicSet

$E_{acyc} = \emptyset$

for all $v \in V$ {
  if $outdeg(v) \geq indeg(v)$ **then**
    Füge die ausgehenden Kanten zu E_{acyc} hinzu.
  else
    Füge die eingehenden Kanten zu E_{acyc} hinzu.
  fi
  Lösche alle, zu v indizenten Kanten aus G.
}

Im Folgenden gehen wir davon aus, das der Graph azyklisch ist.

1.3 Ebenenberechnung

Jedem Knoten wird ein Level bzw. eine Ebenenzugehörigkeit zugewiesen, und zwar so, dass möglichst alle Kanten in *Hauptrichtung* orientiert sind. Die Ebeneneinteilung einer Knotenmenge wird dabei genau dann als *ordentlich* bezeichnet, wenn es nur Kanten zwischen Knoten benachbarter Ebenen gibt. Um eine ordentliche Einteilung erreichen zu können ist es oft notwendig *dummy*-Knoten in Kanten zuzulassen. Die Anzahl der *dummy*-Knoten sollte klein gehalten werden, denn

- die Zeichnunghöhe,
- die Summe aller Kantenlängen,
- die Anzahl der Kantenknicke und
- die Rechenzeit der folgenden Schritte

werden mehr oder weniger (Rechenzeit) proportional dazu schlechter. Algorithmus 1.2 berechnet eine Ebeneneinteilung minimaler Höhe in linearer Zeit.

Tip: Für die Implementierung kann eine topologische Sortierung hilfreich sein.

Algorithmus 1.2

func MinHeightLayering

Platziere alle Knoten v mit $outdeg(v) = 0$ auf Ebene 1.

Berechne $Ebene(u)$ rekursiv durch

$Ebene(u) = max\{i|v \in N_+(u)undEbene(v) = i\} + 1$
wobei
$N_+(u) = \{v \in V\ \exists(u, v) \in E\}$

Die resultierende Ebeneneinteilung kann allerdings sehr breit sein. Andererseits ist die Berechnung von Ebeneneinteilungen minimaler Höhe $mitvorgegebenerBreite$ NP-vollständig. Heuristiken für dieses Problem implizieren gewöhnlich das Lösen linearer Gleichungssysteme, und sind im Rahmen des Praktikums zu umfangreich.

Nun müssen für die folgende Kreuzungsminimierung noch in allen Kanten, die Knoten nicht benachbarter Ebenen verbinden, *dummy*-Knoten eingefüegt werden: eine Kante (u, v), die Knoten u, v aus den Ebene i, j enthält, wird durch die Kanten und Knoten $(u, d_1), .., (d_{i-j-1}, v)$ ersetzt, wobei die *dummy*-Knoten d_k auf den Ebenen $Ebene(d_k) = i + k$ liegen. Diese *dummy*-Knoten werden nach Abschluss der Layoutberechnung, beim Zeichnen als Kanten mit Knicken bzw. als offene Polygone dargestellt.

1.4 Kreuzungsreduzierung

Für jede Ebene wird eine *Ordnung* bzw. eine Permutation ihrer enthaltenen Knoten berechnet. Die Ordnung soll so berechnet werden, dass Kriterium 3, d.h. die Minimierung der Kantenkreuzungen optimiert wird. Dabei hängt die Anzahl der Kreuzungen nicht von der absoluten horizontalen Position, sondern nur von der relativen Position, d.h. vom Rang des Knotens in der Ebene ab. Wie so oft ist das resultierende Problem NP-hart, sogar, wenn man sich auf 2 Ebenen beschränkt, und die Permutation der Knoten auf der einen Ebene fixiert. Deshalb wir im Folgenden die *barycenter*-Heuristik vorgestellt, deren Laufzeit linear ist.

Definition 3 *Als* barycenter *eines Knotens v bezeichnet man den Durchschnitt der x-Koordinaten bzw. der Ränge seiner Nachbarn $N(v) = \{u : \{u, v\} \in E\}$:*

$$bary(v) = \frac{\sum_{u \in N(v)} xpos(u)}{deg(v)}$$

Wie viele andere Strategien basiert sie auf der *paarweisen Optimierung der Kreuzungen zwischen benachbarten Ebenen.* Dabei wird mit einer zufälligen Permutation auf der untersten Ebene L_1 begonnen, und die Kreuzungen zwischen den Ebenen $L_1 \leftrightarrow L_2, ..., L_{h-1} \leftrightarrow L_h$ sukzessiv reduziert. Dabei wird die Permutation der Knoten der Ebene L_i fixiert, und die Knoten der Ebene L_{i+1} werden nach einem definierten Mass (z.B.

Median, *barycenter*) geordnet. Ist die höchste Ebene L_h erreicht oszilliert das Verfahren wieder zurück: $L_h \leftrightarrow L_{h-1}, ..., L_2 \leftrightarrow L_1$. Dies geht solange, bis keine Verbesserung mehr erreicht wird.

1.5 x-Koordinatenberechnung

Nachdem eine Ordnung der Kanten berechnet wurde, können Knoten noch unter Einhaltung der Ordnung auf der x-Achse verschoben werden. Dies ist vor allem bei der Minimierung von Kantenknicken, sowie bei Graphen mit unterschiedlich grossen Knoten von Bedeutung. Da die oben beschriebene barycenter bzw. die Median Heuristik schon eine gewisse Berücksichtigung der Kantenknicke bieten und eine Implementierung verhältnismässig aufwendig wäre, verzichten wir im Praktikum auf eine weitere Optimierung der x-Koordinaten, und verweisen bei Interesse auf [2]

Literatur

[1] Battista G., Eades P., Tamassia R., Tollis I.: *Graph Drawing*, Prentice Hall, 1999.

[2] D. Wagner, M. Kauffmann: *Drawing Graphs*, Springer, 2001.