Aufgabe 3.1 | Visuelle Eigenschaften von Graphen

XXX

Aufgabe 3.2 | Visual Clutter

Kamada-Kawai

- Rahmenbedingungen
  - ungerichtete, gewichtete Graphen
  - gerade Kanten, keine Beschränkung bei Positionierung der Knoten
  - zusammenhängender Graph (sonst: Zusammenhangskomponenten einzeln zeichnen)
- Kriterien
  - Anzahl Kantenkreuzungen reduzieren
  - gleichmäßige Verteilung von Knoten und Kanten (wichtiger für menschliches Verständnis)
- Idee
  - Stahlringe werden durch Federn zusammengehalten
  - Ziel: Minimierung der Energie dieses Systems
  - ideale Distanz zwischen zwei Knoten proportional zur Länge eines kürzesten Pfades zwischen ihnen
- Algorithmus
  - Berechnung der Distanz $d_{i,j}$ (Shortest Path Algorithm - Floyd), in gewichteten Graphen: Distanz = Summe der Gewichte
  - Berechnung der Länge $l_{i,j}$ (aus Distanz und optimaler Länge)
  - Berechnung der Federstärke $k_{i,j}$ (aus Distanz)
  - Bestimmung der initialen Knotenpositionen
  - Schrittweise Minimierung der Energie: Verschieben eines Knotens in stabile Position

  Laufzeit: $O(|V|^3 + T * |V|)$, T = Anzahl der Iterationen
- Besonderheiten
  - symmetrische Graphen werden auch symmetrisch angeordnet
  - isomorphe Graphen werden auf dieselbe Art und Weise dargestellt (evtl. verschoben, gedreht oder gespiegelt)

Fruchtermann-Reingold

- Rahmenbedingungen
  - ungerichtete Graphen
  - gerade Kanten, Graph begrenzt durch festen Rahmen
- Kriterien
  - gleichmäßige Verteilung von Knoten und Kanten innerhalb des Rahmens
  - Anzahl Kantenkreuzungen reduzieren
  - Streben nach symmetrischer Darstellung
  - Nachbarknoten sollen nah beieinander liegen, Knoten sollen aber nicht zu eng zusammen gezeichnet werden
- Idee
  - Knoten verhalten sich wie Atomteilchen und üben anziehende und abstoßende Kräfte aufeinander aus
  - zur Vereinfachung: anziehende Kräfte nur zwischen Nachbarknoten, abstoßende Kräfte zwischen allen Knoten
  - Geschwindigkeit statt Beschleunigung, statisches Gleichgewicht statt dynamischem Gleichgewicht
- Algorithmus
  - Berechnung der abstoßenden Kräfte
  - Berechnung der anziehenden Kräfte
  - Verschiebung begrenzen durch Temperatur
  - Teilchen am Rahmen stoppen
  - Sonderfall: zwei Knoten mit derselben Position $\rightarrow$ behandeln als wären sie dicht beieinander, dann wirken große abstoßende Kräfte

  Laufzeit: O(T * $(|V|^2 + |E| + |V||E|)$), T = Anzahl der Iterationen = O(|V|) oder O(|E|)
- Variante des Algorithmus
  - Einteilung in Gitter
  - abstoßende Kräfte nur zwischen Knoten benachbarter Quadrate
  - besser: abstoßende Kräfte zwischen Knoten innerhalb eines Kreises um Ausgangsknoten

  Laufzeit: O(T * $(|V| + |E| + |V||E|)$), T = Anzahl der Iterationen = O(|V|) oder O(|E|)

- Ergebnisse
  - symmetrische Graphen werden besonders gut dargestellt
  - nicht alle symmetrischen, planaren Graphen werden optimal dargestellt
  - isomorphe Graphen werden auf dieselbe Art und Weise dargestellt (evtl. verschoben, gedreht oder gespiegelt)
  - relativ viele Kantenkreuzungen bei Bäumen, Wurzel nicht zu erkennen
  - sehr unterschiedlich gute Ergebnisse bei dreidimensional wirkenden Graphen
  - Variante: Dreidimensionales Layout
    * Layouting in drei Dimensionen, Projektion in die Ebene
    * keine Verbesserung der Ergebnisse im Vergleich zum 2D-Algorithmus
    * Auswahl der Projektion durch Benutzer
  - Zusammenhangskomponenten driften auseinander und verbreiten sich am Rand → einzelne Berechnungen (wie bei Kamada-Kawai) oder Gitter-Variante
  - sehr schnelle Berechnungen

Gemeinsamkeiten & Unterschiede

- Layouting ungerichteter Graphen mit geraden Kanten, Kamada-Kawai unterstützt auch gewichtete Graphen, Fruchtermann-Reingold ordnet Graphen innerhalb eines festen Rahmens an
- Ziel beider Verfahren: gleichmäßige Verteilung der Knoten und Kanten, Kamada-Kawai dabei eher fixiert auf Vermeidung von Kantenkreuzungen
- verschiedene Ideen liegen zugrunde: System aus Federn ↔ Kräfte zwischen Atomteilchen
- verschiedene Laufzeiten, Fruchtermann-Reingold wesentlich schneller
- gute Darstellungen symmetrischer und isomorpher Graphen, Fruchtermann-Reingold Probleme mit komplexeren Graphen

Aufgabe 3.3 | Graphen übersichtlicher gestalten

Einfärben von Knoten oder Kanten

Gruppieren von Knoten und Kanten

Verschiedene Knotendarstellungen

Aufgabe 3.4 | Visualisierung mit Gephi

Für die Visualisierungen mit Gephi wird hier der Datensatz der Facebook-Freundschaften der empfohlenen Seite verwendet, der aus 10 Netzwerken zusammengesetzt wurde [1].

Gephi beginnt nach dem Import des Datensatzes mit einem zufälligen Layout, das die Knoten innerhalb einer rechteckigen Fläche anordnet, siehe Abbildung 1. In dieser Darstellung sind bei Datensätzen solcher Größe vermutlich selten Aussagen über die Daten zu treffen.

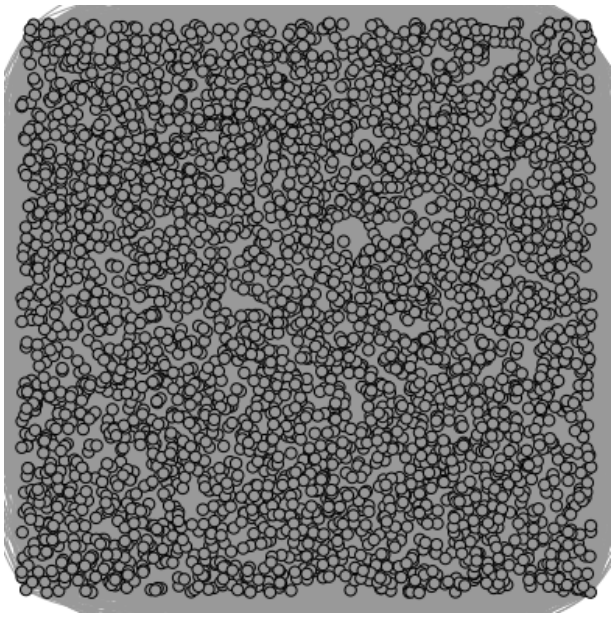

Abbildung 1: Grafik direkt nach Import der Daten

In erster Linie kann die Visualisierung durch ein Layout verbessert werden, das nach den Beziehungen zwischen den Daten ausgerichtet ist. Die Erfahrung zeigt, dass oft eine Kombination verschiedener Algorithmen ästhetischere Abbildungen ergibt, aus denen bereits erste Erkenntnisse gewonnen werden können. Abbildung 2 zeigt ein solches Layout.

Zusätzlich können Knoten und Kanten noch eingefärbt werden. Zu den zahlreichen Möglichkeiten gehören unter anderem (Ein-/Ausgangs-)Grad, Kantengewicht oder Zentralitätsmaße.

[1] Größere Datensätze waren im CIP-Pool leider nicht zu importieren. Selbst mit diesem relativ kleinen Datensatz gab es immer wieder Probleme mit dem begrenzten Speicherplatz. Auf meinem Windows-PC wollte Gephi leider nicht starten.

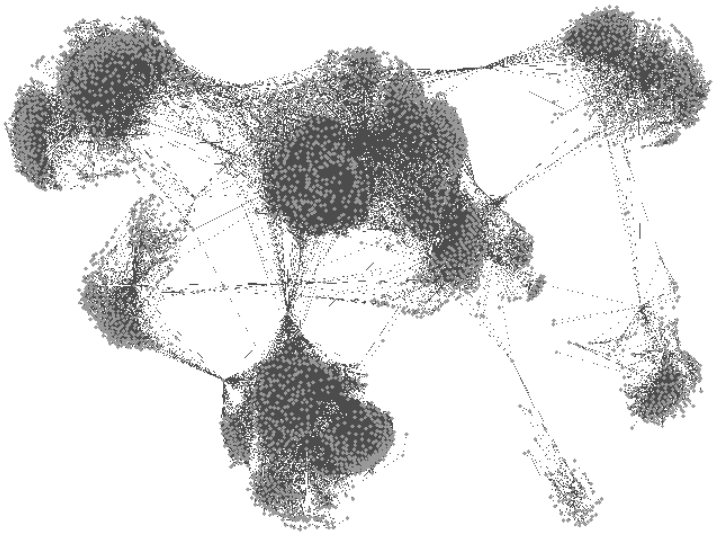

Abbildung 2: Grafik nach Anwendung von Layout-Algorithmen

Am nützlichsten hat sich für diesen Datensatz die Färbung der Knoten nach *Modularity* erwiesen, wie in Abbildung 3 zu sehen.

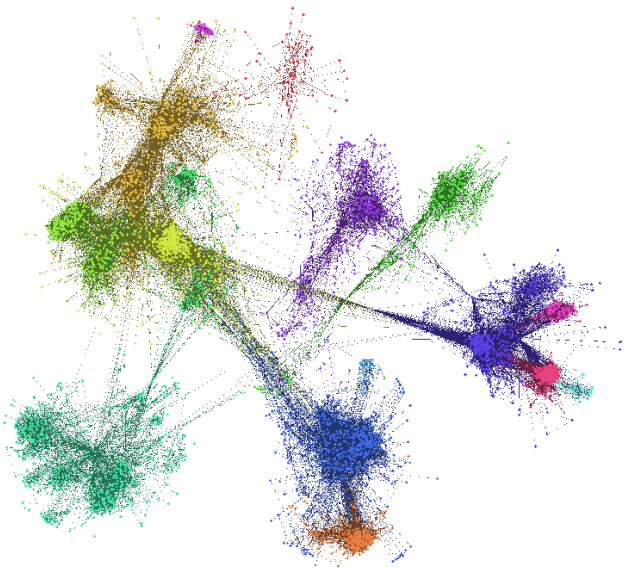

Abbildung 3: Grafik nach Anwendung von Layout-Algorithmen und Einfärben der Knoten