Karlsruher Institut für Technologie
Fakultät für Informatik
ITI Wagner

Algorithmen zur Visualisierung von Graphen
WS 2013/14
Tamara Mchedlidze, Martin Nöllenburg

Siebtes Übungsblatt

Ausgabe: 28. Januar 2014
Abgabe: Keine, Besprechung am 5. Februar 2014

1 Fehlstände Zählen

(a) Sei $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ eine Permutation. Ein Paar (i, j) mit $1 \leq i < j \leq n$ heißt Inversion, wenn $\pi(i) > \pi(j)$. Entwerfen Sie einen Algorithmus, der die Anzahl der Inversionen einer Permutation von n Elementen in $O(n \log n)$ Zeit berechnet.
Hinweis: Denken Sie an Sortieralgorithmen wie Mergesort.

(b) Gegeben sei ein einfacher, bipartiter Graph $G = (V, E)$, dessen Knoten gemäß der Bipartition auf zwei parallele Geraden verteilt sind. Die Knoten seien disjunkt und die Kanten geradlinig gezeichnet. Entwerfen Sie einen Algorithmus, der die Anzahl der Kreuzungen in einer Laufzeit $O(|E| \log |V|)$ bestimmt. Begründen Sie, warum es nicht möglich ist, in dieser worst-case-Laufzeit alle Kreuzungen (d.h. die Paare betroffener Kanten) *auszugeben.*

2 Kreuzungen bei Lagenlayouts

Zeigen Sie, dass die Baryzenter-Heuristik zur einseitigen Kreuzungsreduktion die optimale Lösung liefert, falls diese keine Kreuzung enthält.

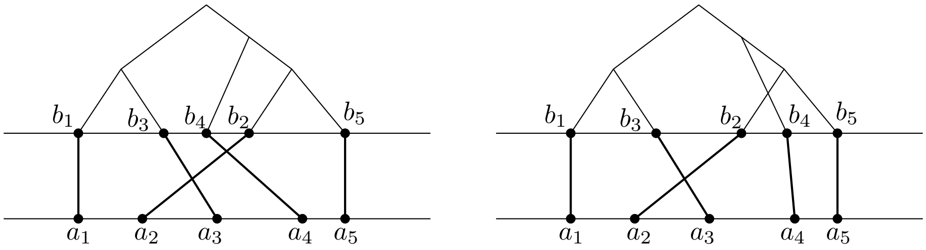
3 Tanglegram Layout

Das einseitige Kreuzungsminimierungsproblem in einem „Zweilagengraph“ ist bereits NP-schwer. Betrachte das folgende verwandte Problem:

Problem 3.1 (One Tree Crossing Minimization (OTCM)). *Gegeben sei ein perfektes Matching M auf $2n$ Knoten $A = \{a_1, \ldots, a_n\}$ und $B = \{b_1, \ldots, b_n\}$, mit Matchingkanten $E = \{(a_i, b_i) \mid i = 1, \ldots, n\}$. Bestimme für eine feste Permutation π_1 der Menge A eine Permutation π_2 der Menge B, die die Anzahl der Kreuzungen von M minimiert, wobei A und B geordnet nach π_1 und π_2 auf zwei horizontalen Geraden platziert sind. Dabei muss π_2* kompatibel *zu einem Binärbaum T mit Blattmenge B sein.*

Die Permutation π_2 heißt dabei *kompatibel* zu T, wenn eine planare Zeichnung von T in der Halbebene $y \geq 0$ existiert, die die Blätter in der Reihenfolge π_2 auf der x-Achse platziert und alle inneren Knoten im Bereich $y > 0$.

bitte umblättern



kompatible Permutation π_2 Permutation π_2 nicht kompatibel

(a) Zeigen Sie, dass das Problem 3.1 in polynomieller Zeit gelöst werden kann. Welchen asymptotischen Zeitaufwand hat Ihr Algorithmus?

Hinweis: Verwenden Sie z.B. dynamische Programmierung.

(b) Angenommen beide Permutationen π_1 und π_2 sind variabel und jeweils durch Kompatibilität mit zwei Binärbäumen T_1 und T_2 auf den Blättern A bzw. B eingeschränkt. Zeigen Sie, dass in polynomieller Zeit entschieden werden kann, ob das Matching kreuzungsfrei gezeichnet werden kann.

4 Metro Maps

(a) Beim Zeichnen von Metro Maps können neben Knickzahl, Kantenlänge und Erhaltung der relativen Lage andere Optimierungskriterien relevant sein. Modifizieren Sie das in der Vorlesung vorgestellte ILP so, dass zusätzlich der Umfang der Bounding-Box bzw. die Breite bei festgelegter Höhe minimiert wird. Warum ist die Fläche der Zeichnung kein geeignetes Optimierungskriterium?

(b) Zur Minimierung der Knickkosten entlang von Metrolinien wurde folgende Kostenfunktion definiert:

$$\mathrm{cost}_{\mathrm{bend}} = \sum_{L \in \mathcal{L}} \sum_{uv, vw \in L} \mathrm{bend}(u, v, w).$$

Dabei steigt die Kostenfunktion $\mathrm{bend}(u, v, w)$ mit steigendem Knickwinkel zwischen den Kanten uv und vw linear an ($45° : 1, 90° : 2, 135° : 3$). Überlegen Sie sich, wie man diese Kostenfunktion im ILP-Modell umsetzen kann. Sie können dazu z.B. die für jede Kante uv definierte Richtungsvariable $\mathrm{dir}(u, v)$ verwenden. Wie lässt sich eine beliebige monoton steigende Kostenfunktion modellieren?

(c) Zur Minimierung der Kantenlänge wurde in der Vorlesung die folgende Kostenfunktion definiert.

$$\mathrm{cost}_{\mathrm{len}} = \sum_{\{u,v\} \in E} \mathrm{len}(\{u, v\})$$

Geben Sie lineare Nebenbedingungen an, die sicherstellen, dass die Variable $\mathrm{len}(\{u, v\})$ der Länge der Kante $\{u, v\}$ bezüglich der Normen L_1, L_∞ und L_2 entspricht. Verwenden sie dazu nur die Positionen $(x(u), y(u))$ und $(x(v), y(v))$ von u bzw. v.