Automatisches Zeichen von Graphen

Gruppe 3&6, Übungsblatt 6, Aufgabe 2

Jan Beisenkamp, Philipp Büdenbender, Martin Gronemann, Moritz Schallaböck und Bernd Zey

Dortmund, February 28, 2008

Agenda

1. Das Paper
2. Die Implementierung

1. Das Paper

Vorstellen der wesentlichen Ideen

Einleitung

Ein Ausblick auf das Paper

- Aufgabe:
  - Implementieren des Plaritätstest aus der Publikation
    - Algorithms for multi-level graph planarity testing and layout (Patrick Healy & Ago Kuusik, 12 Feb. 2004)

- Multi Level Graphen sollen
  - Auf (Level-)Planarität getestet werden
  - Sauber gezeichnet werden

- Oft verwendet bei Multi Level Graph Algorithmen werden drei Stufen
  1. Erstellen von Levels (Disjunkte Teilmengen)
  2. Finden einer Permutation, die Kantenkreuzungen minimiert
  3. Zuweisen der Knoten auf bestimme Positionen für ein gutes Layout

- Hier behandelt: Punkt 2 – Finden einer Permutation

Der Vertex-Exchange Graph

Die Erklärung des Graphen

- Ein Vertex-Exchange Graph ist der Graph G‘ = (V‘,E‘).
  - Knotenmenge $V' = V_1' \cup V_2' \cup \ldots \cup V_p'$
    - $V_r' = \{(v,w) | v,w \in V_r\}$
  - Kantenmenge $E' = E_1' \cup E_2' \cup \ldots \cup E_{p-1}'$
    - $E_r' = \{(e,f) | e,f \in E_r, e = (t,u), f = (w,v), (t,w) \in V_r', (u,v) \in V_{r+1}'\}$

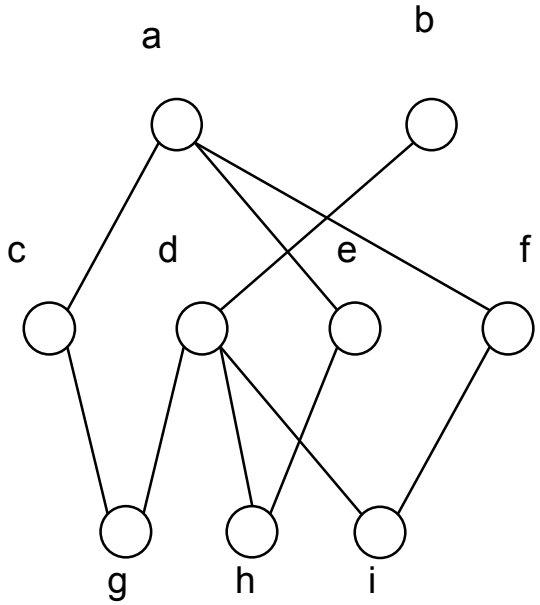
- Der Graph ist also eine Gruppierung des Ursprungsgraphen G(V,E) der unterschiedlichen Knoten eines Levels.
- Die Kanten sind wie oben verbunden, wenn die Knoten in dem Ursprungsgraphen auch verbunden sind.

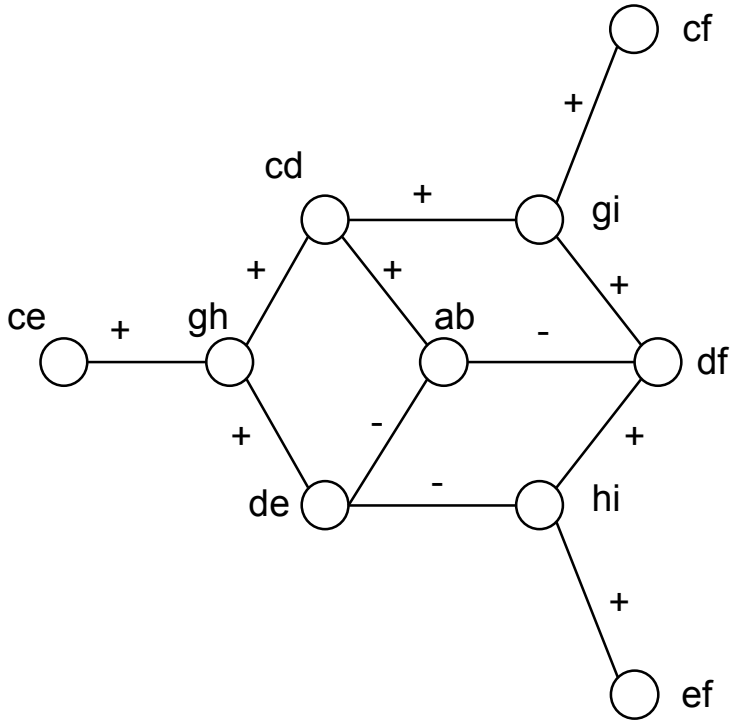
- Im folgenden werden Kanten mit + und – benannt. Eine Kante wird + Kante, wenn sich die zugehörigen Kanten in G nicht kreuzen und – umgekehrt.

Der Vertex-Exchange Graph

Die Erklärung des Graphen

» Beispiele für die Benennung:





Der Vertex-Exchange Graph

Eigenschaften des Graphen

- Zwei Knoten $v(v_1, v_2)$ und $w(w_1, w_2)$ sind **Adjazent** in G‘, wenn im Graphen G die Knoten v_1 und v_2 auf einem Level liegen und die Knoten mit einer Kante verbunden sind, also entweder (v_1, w_1) und (v_2, w_2) oder (v_1, w_2) und (v_1, w_2) verbunden sind.
- Zwei Knoten v und w sind in der gleichen **Zusammenhangskomponente** von G‘ wenn diese in einer Sequenz von Knoten liegen und so verbunden sind, wie oben definiert.
- Wenn dann nun der letzte Knoten einer Sequenz wieder mit dem ersten Knoten verbunden ist, so haben wir einen **Kreis**.

- **Definition:** Ein Kreis wird ***gerader Graph*** genannt, wenn die Kreuzungen bei diesem Teilgraphen gerade sind, ansonten ist es ein ***ungerader Graph***.

- **Korollar 12:** Ein Graph ist **Level-Planar**, wenn sein Vertex-Exchange Graph G‘ **keine** ungeraden benannten Kreise enthält.

Der Algorithmus

Einleitung

- Der Algorithmus testet die Level Planarität
- Es werden nun die Kanten wie folgt markiert:
  - Wenn $v(x_{ij}^{r})$ und $v(x_{kl}^{s})$ durch eine + Kante verbunden sind, so ist $x_{ij}^{r} = x_{kl}^{s}$ (analog für – Kante)
- Erste Kante wird zufällig aus dem Graphen ausgewält
- Die weitere Markierung wird mit einem DFS Algorithmus gemacht:
  - Wenn ein Knoten über eine + Kante erreicht wird, so erhält er als Wert den des Vorgängers, sonst die Negation
  - Wenn Knoten ein zweites mal besucht wird, so wird der alte Wert mit dem neuen Wert verglichen. Wenn diese unterschiedlich sind -> nicht Level-Planar
- Wenn jede Zusammenhangskomponente bei dem Algorithmus den Wert true liefert, so hat der gesamte Vertex-Exchange Graph keine ungeraden benannten Kreise und somit ist der Eingabe Graph Level Planar

Der Algorithmus

Pseudocode

Level PlanarityDFS (G‘,v,b)

```
1.  IF value(v) = unknown THEN
2.    value(v) = b
3.    FOR ALL Knoten w adjazent zu v in G‘ DO
4.      IF label((v,w)) = ‚+‘ THEN
5.        result = LevelPlanarityDFS (G‘, w, b)
6.      ELSE
7.        result = LevelPlanarityDFS(G‘, w, ¬b)
8.      END IF
9.      IF result = false THEN
10.       RETURN false
11.     END IF
12.   END FOR
13. ELSE IF value(v) ≠ b THEN
14.   RETURN false
15. END IF
16. RETURN true
```

2. Die Implementierung

Wie das ganze umgesetzt wurde

Die Implementierung

Pseudocode...

```
LevelPlanarTestDFS(u, v)
    Wenn Paar (u,v) schon besucht return
    Wenn Paar (v,u) schon besucht return nicht planar
    Sonst markiere (u,v) als besucht
    Für alle adjazenten Paare (a,b) mit a != b
      Wenn LevelPlanarTestDFS(a,b) == nicht planar
          return nicht planar
    return
```

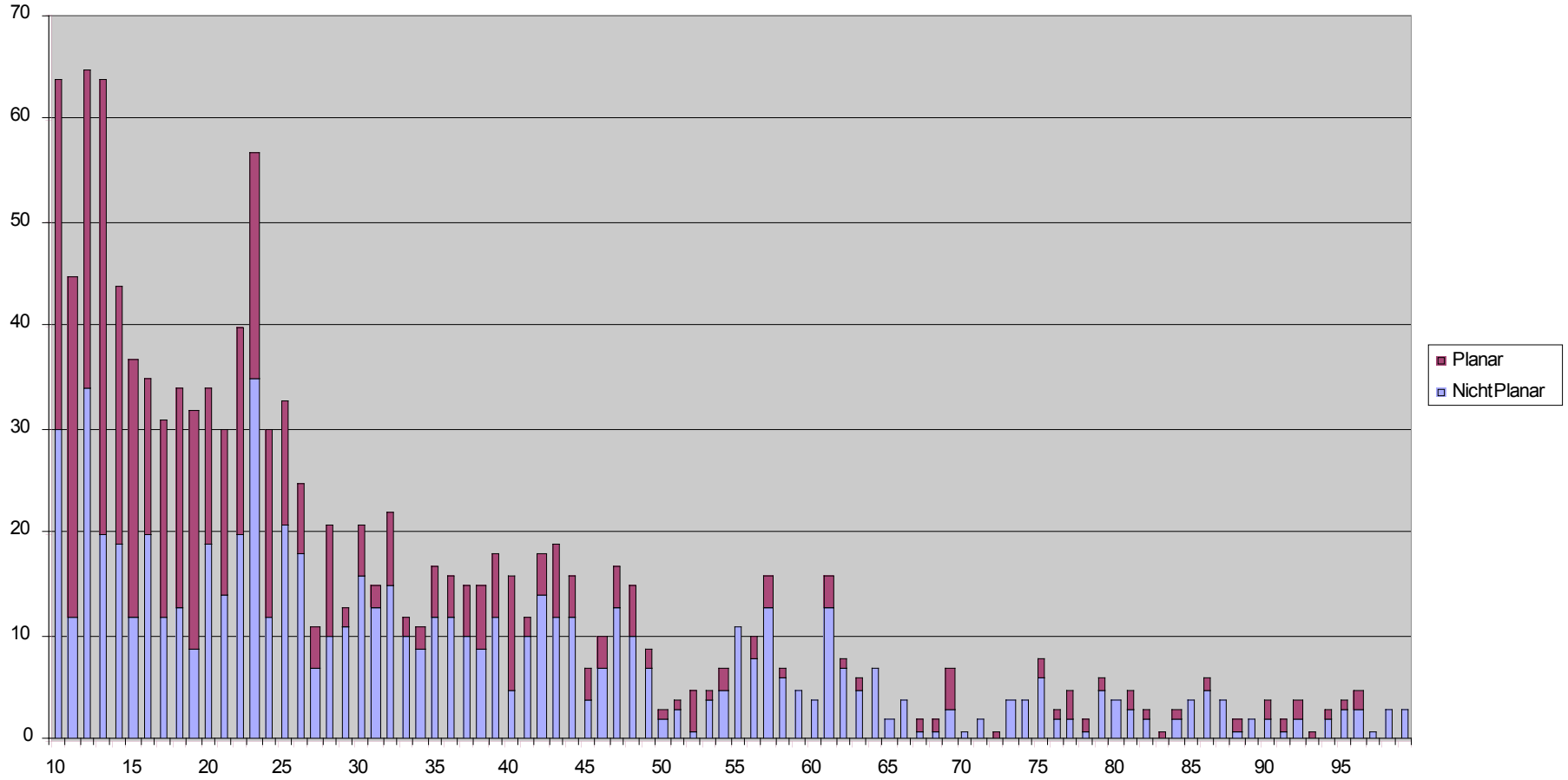
Die Implementierung

Pseudocode...

```
LevelPlanarTest(LevelGraph G)
    Für alle unbesuchten Knotenpaare (a,b) mit level(a) == level(b)
      Wenn LevelPlanarTestDFS(a,b) == nicht planar
          return nicht planar
    return planar
```

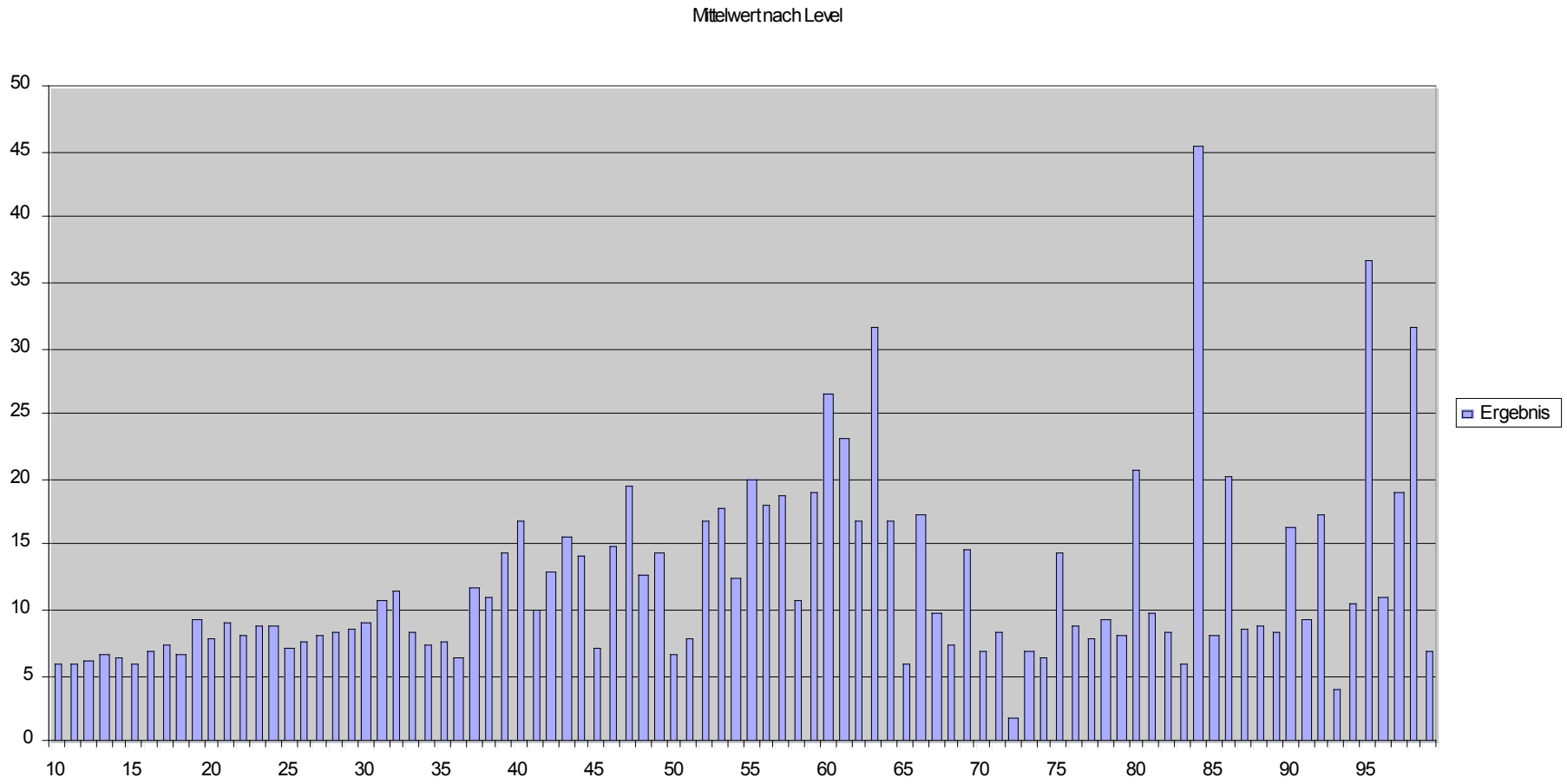
Ergebnisse

Planar / Nicht Planar



Ergebnisse

Durchschnitte Level pro Knotenanzahl



Die Ergebnisse

- Vertex Exchange Graph als Datenstruktur wird nicht benötigt
  - Reihenfolge der beiden Knoten als Parameter reicht aus
- AT&T-Graphen:
  - 1277 Graphen insgesamt
  - 538 level-planare Graphen

Das war‘s!

vielen Dank für eure Aufmerksamkeit!

Gruppe 4&6

TU Dortmund
Fakultät Informatik
LS11 Algorithm Engineering