Implementierung eines Algorithmus für das glatt-orthogonale Zeichnen planarer Graphen

Bernhard Häussner

Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Informatik I
Effiziente Algorithmen und wissensbasierte Systeme

Betreuer:
Prof. Dr. Alexander Wolff
Dipl.-Inf. Philipp Kindermann

Bachelor-Kolloquium am 12. Mai 2014

Implementierung eines Algorithmus für das glatt-orthogonale Zeichnen planarer Graphen

- Zeichnen von Graphen: Knoten platzieren, Kanten zeichnen
- glatt-orthogonal: besonders übersichtlich
- Planarität: Keine Überschneidungen von Kanten
- Algorithmus: Alam et al. [ABK$^+$14]
- Implementierung: Test mit vielen Graphen

Ein Liniennetzplan

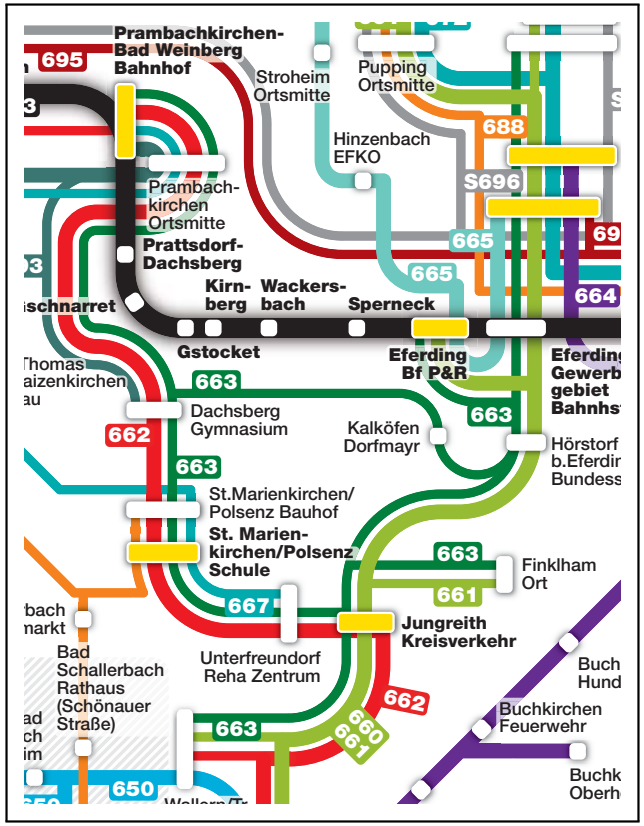


Abbildung: Ein Ausschnitt aus dem Liniennetzplan *Eferding – Grieskirchen – Wels*, Helge Waldherr, 2014. `http://www.ooevv.at/upload/content/blaetterkatalog/Liniennetzplan/EF-GR-WE/assets/common/downloads/Liniennetzplan-GR-EF.pdf`

Vom Liniennetzplan zur glatt-orthogonalen planaren Zeichnung

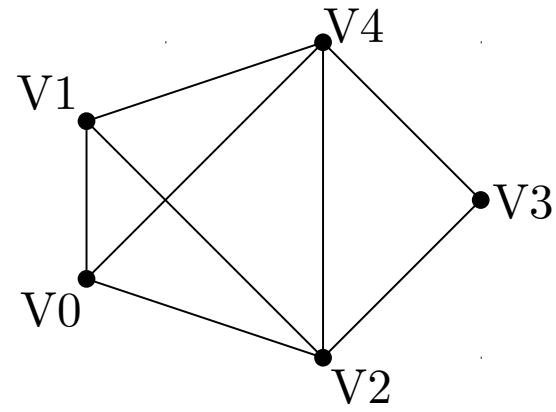
- Liniennetzpläne für jedermann ohne Vorwissen benutzbar
- Abstraktion: Haltestellen $\mapsto$ Knoten, Strecken $\mapsto$ Kanten
- Reduktion: Keine Beschriftungen, Überschneidungen oder parallele Strecken
- Ziel: Platzierung der Knoten, rundes Zeichnen der Kanten
- Überschneidungen der glatt-orthogonalen Kanten vs. Komplexität

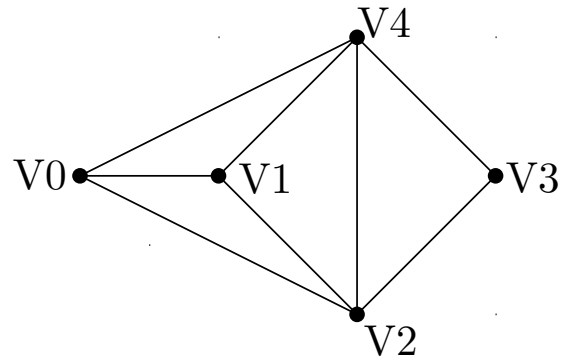
Graphen

- Tupel $G = (V, E)$ von Mengen, wobei $E \subset \mathcal{P}(V)$
- Ungerichtete Graphen
- Einfache Graphen
- Zusammenhängende Graphen
- Bsp: $G_E = (V_E, E_E)$; $V_E = \{0, 1, 2, 3, 4\}$; $E_E = \{\{0, 1\}, \{0, 2\}, \{0, 4\}, \{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

Planare Einbettung

- Planarer Graph: kann ohne Überschneidungen gezeichnet werden
- Kanten können sich an den Knoten berühren
- Zeichnen mit planarer Einbettung durch Reihenfolge der Kanten in Adjazenzliste
- Berechnen mit Algorithmus von Brandes [Bra09]
- Ausführlicher Pseudocode, leicht zu implementieren





st-Ordnung

- Reihenfolge der Knoten
- je min. ein Vorgänger und Nachfolger
- s erster, t letzter Knoten
- Voraussetzung: Graph zweifach zusammenhängend
- Berechnung mit dem Algorithmus von Even und Tarjan [ET76]
- Zeichenalgorithmus platziert Knoten in Reihenfolge der *st*-Ordnung

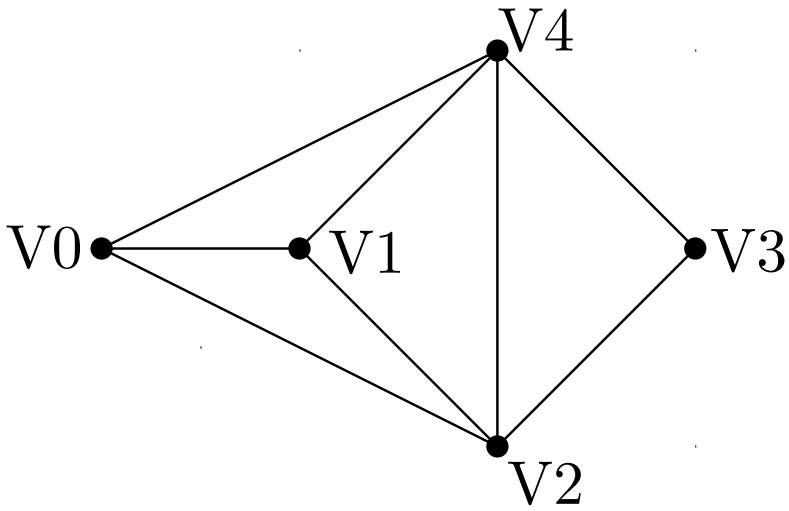


Abbildung: Knotenzahlen sind *st*-Ordnung. Vertauscht man V1 und V3, so hat V1 nur nachfolgende Knoten.

Orthogonale Layouts

- Knoten auf einem Gitter
- Kanten nur horizontal und vertikal
- Vier Ports für Kanten an jedem Knoten
- Maximaler Grad ist 4
- Berechnung mit Algorithmus von Liu et al. [LMS98].

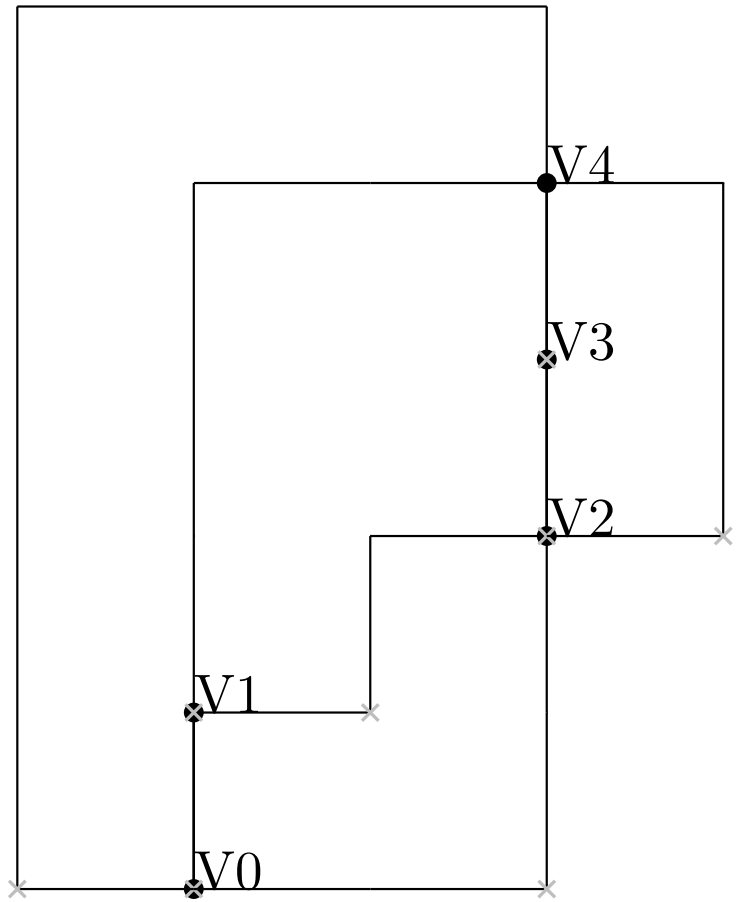


Abbildung: Orthogonales Layout des Beispielgraphen

Der Algorithmus von Liu et al.

- Spezielle Portzuweisung aus planarer Einbettung
- Kante hat max. 3 Segmente
- Von unten nach oben
- Reihenfolge der *st*-Ordnung
- Kantenpartition in ein- und ausgehend *st*-Ordnung

Portzuweisung im Algorithmus von Liu et al.

Portzuweisung eingehender Kanten je Eingangsgrad. Fett sind eingehende Kanten, gestrichelt mögliche ausgehende Kanten.

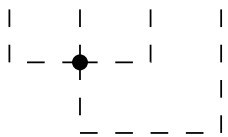


Abbildung: 0

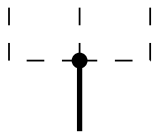


Abbildung: 1

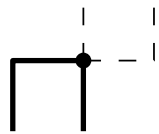


Abbildung: 2

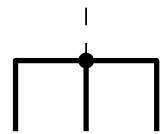


Abbildung: 3

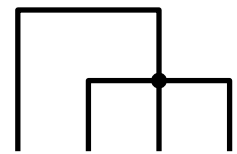
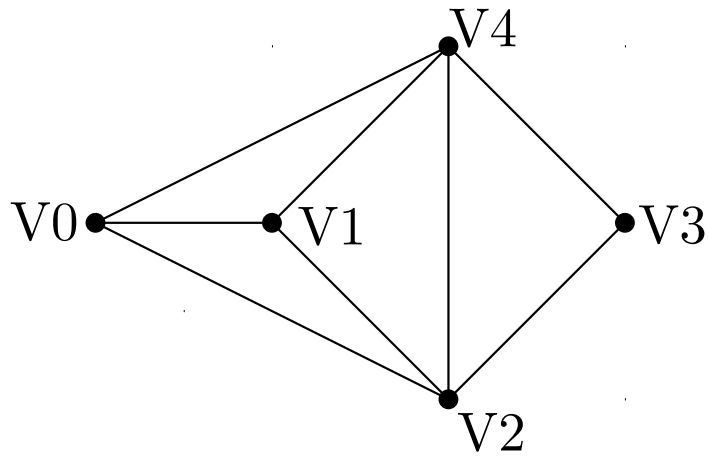


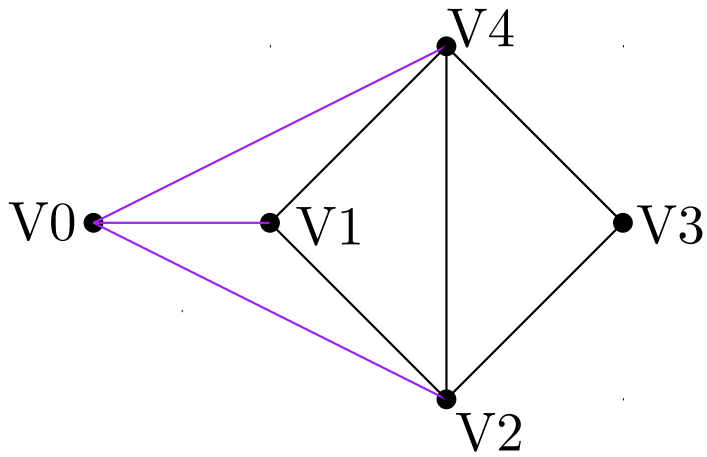
Abbildung: 4

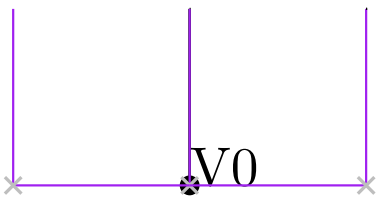
Beispiel zum Algorithmus von Liu et al.



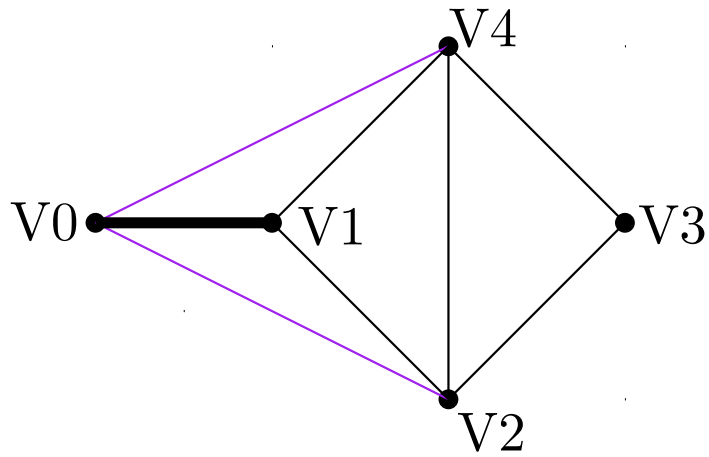
V0

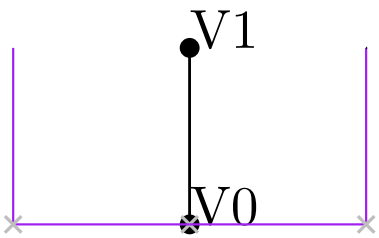
Beispiel zum Algorithmus von Liu et al.



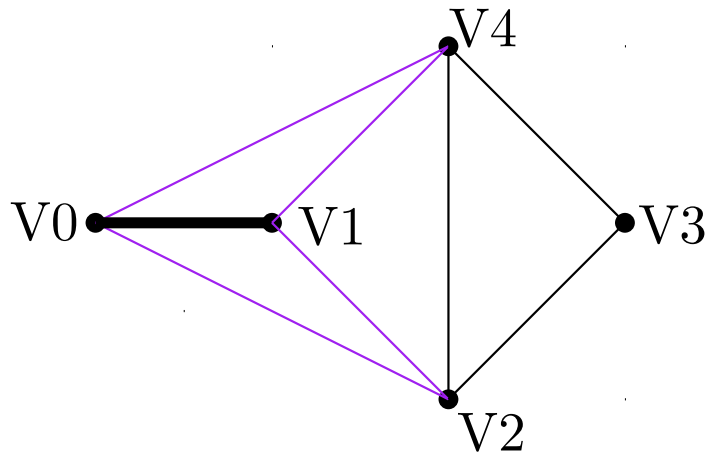


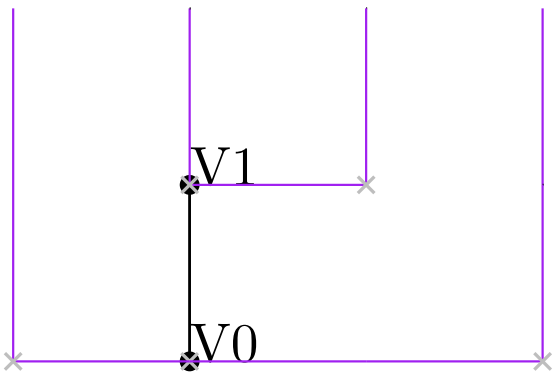
Beispiel zum Algorithmus von Liu et al.



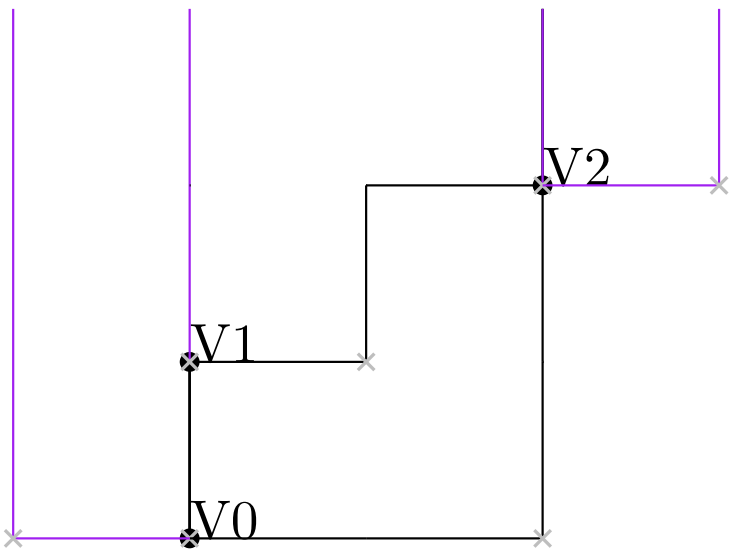


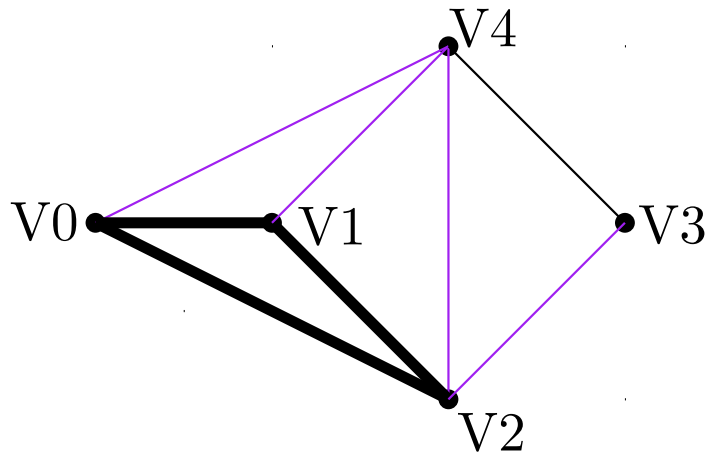
Beispiel zum Algorithmus von Liu et al.



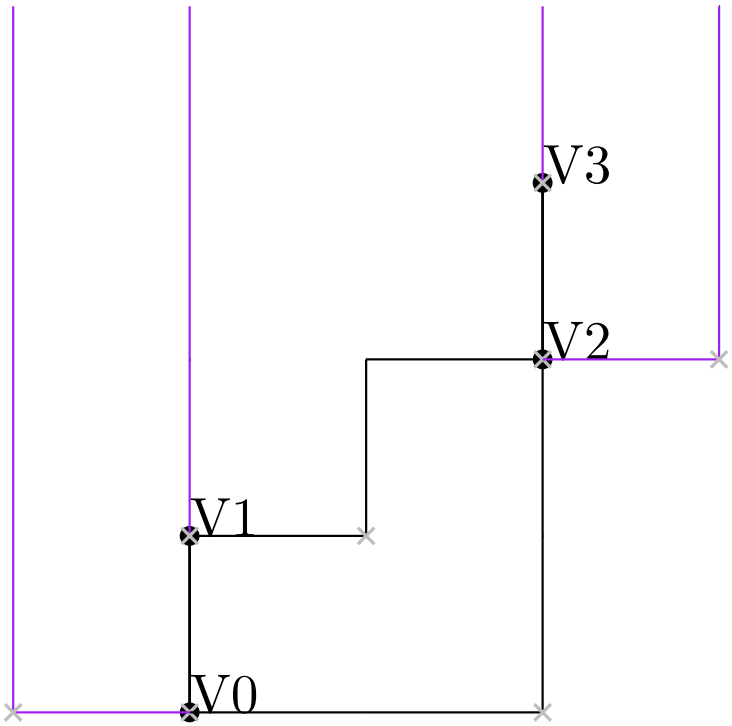


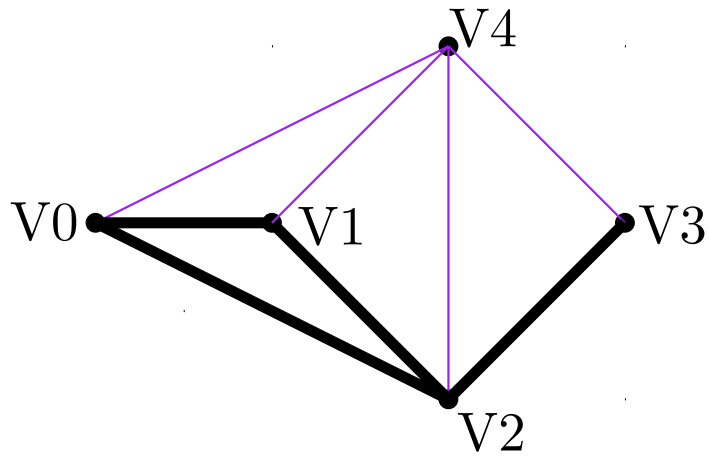
Beispiel zum Algorithmus von Liu et al.



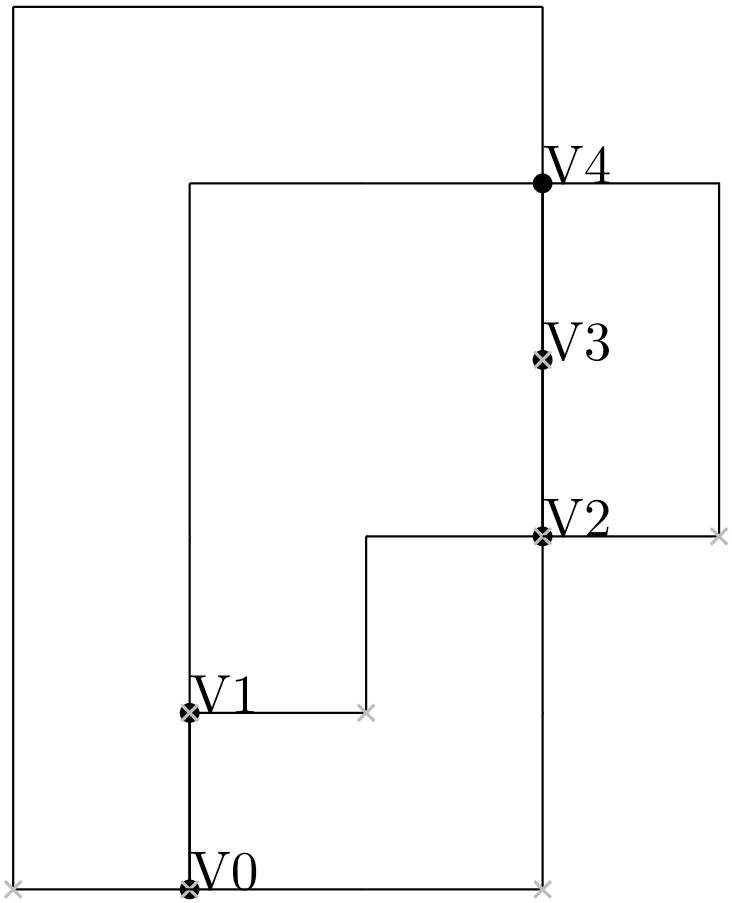


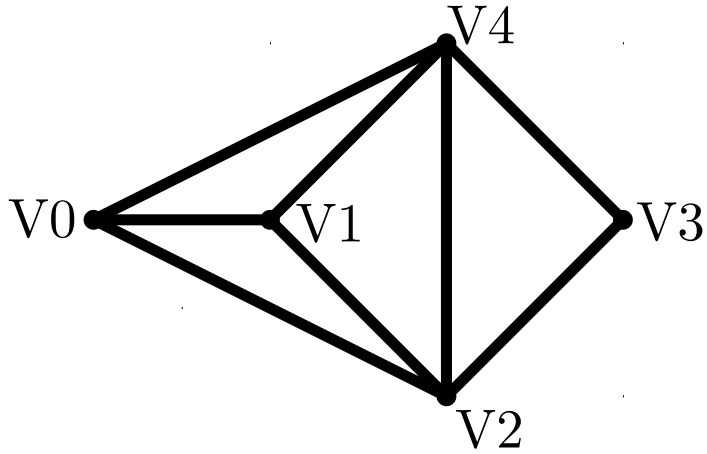
Beispiel zum Algorithmus von Liu et al.





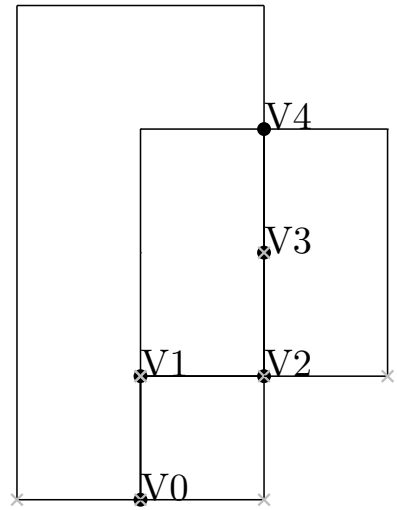
Beispiel zum Algorithmus von Liu et al.

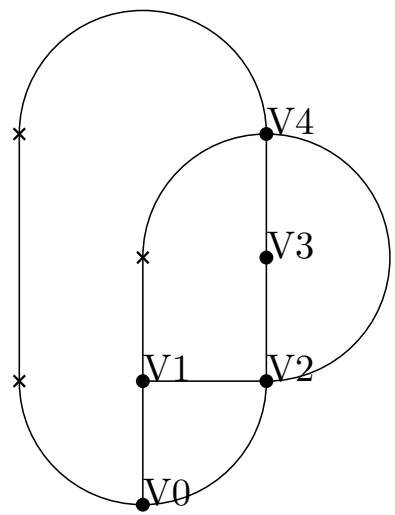




Glatt-orthogonale Zeichungen

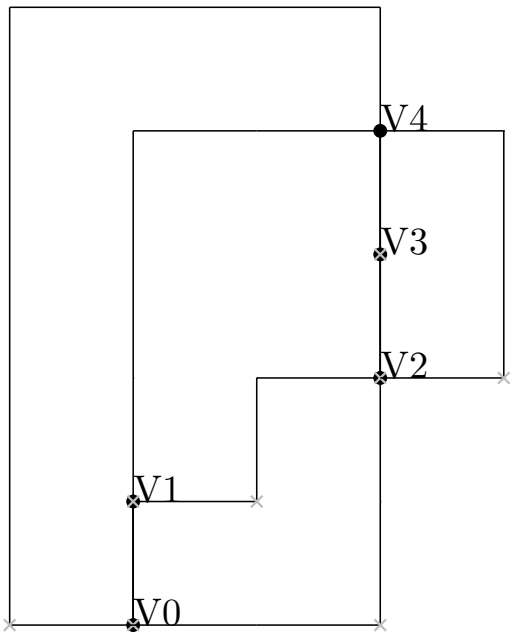
- Bekos et. al. [BKKS13]
- orthogonal: Koordinatengitter; horizontale, vertikale Segmente, Ports
- Glatt: Keine Knicke
- Mit Kreisbögen abgerundet
- Tangentialer Übergang

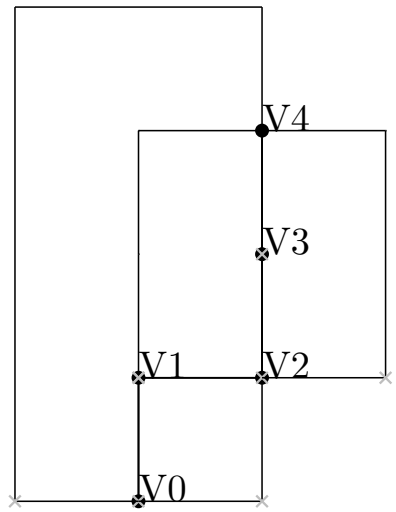




Entfernung stufenförmiger Kanten

- Stufenförmige Kanten werden S-förmige Kanten
- Alle Stufen-Knoten auf ein Plateau
- Höhe des Plateaus berechnen
- Mehrere Plateaus mit selber Höhe möglich





Problem: Neue Überschneidungen

- Abrunden führt zu Überschneidungen
- Kreisbögen so, dass Kanten nur zwei Segmente haben
- Graph auseinander ziehen, Platz schaffen
- An horizontalen Segmenten teilen

Wo kann es zu Überschneidungen kommen?

- Innen an C- und L-Kanten: Plateaus nach oben schieben
- Außen an C-Kanten: Graph auseinander ziehen
- Alle C- und L-Kanten müssen horizontales Segment haben

Kollisionen mit C-Kante vermeiden

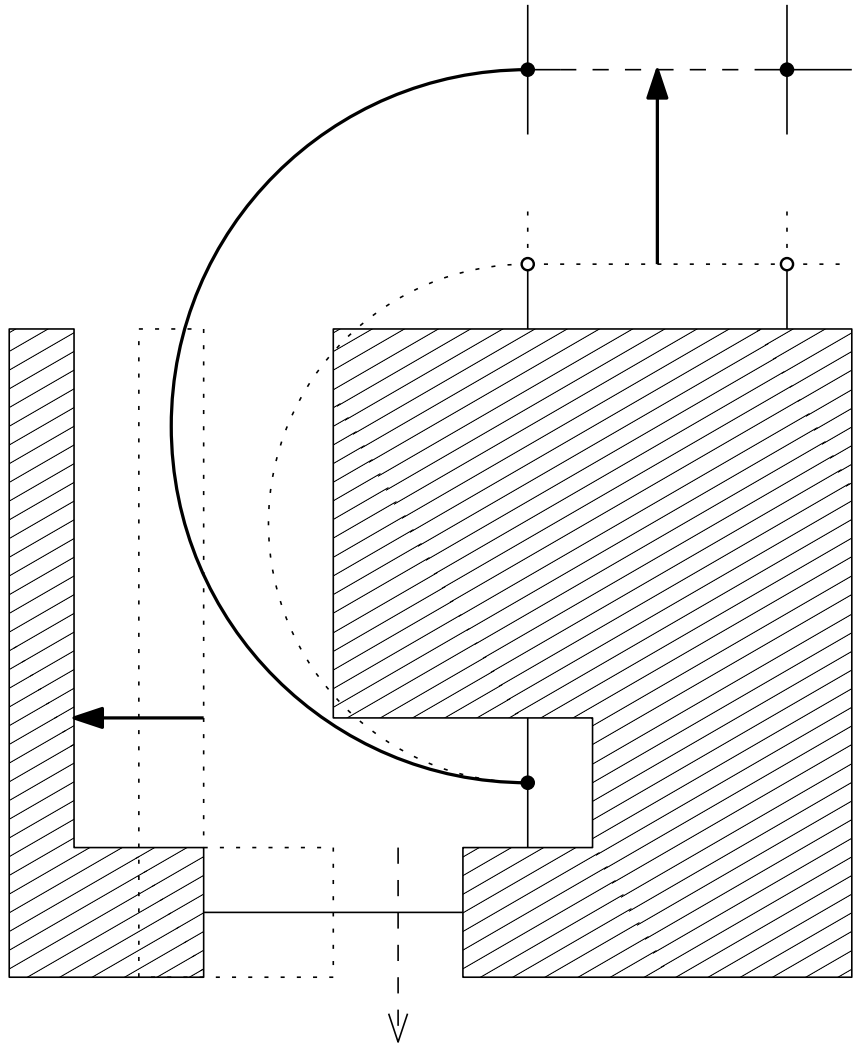


Abbildung: Maßnahmen zur Vermeidung von Kollisionen mit C-Kante.

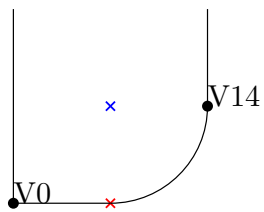
Schnitte durch Graphen finden

- Immer an einem Knoten, in einem Quadranten
- C-, I- und L-Kanten an horizontalen Segmenten durchqueren
- Kein vertikales Segment durchqueren
- Kante folgen oder durchschneiden

Beispieldurchlauf

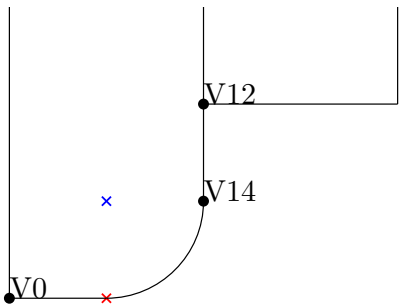
snap #0 placing first vertex

Beispieldurchlauf



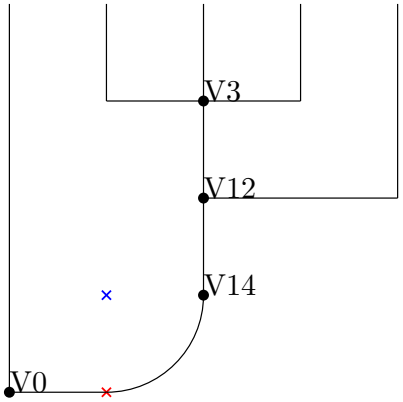
snap #1 placing initial vertex of tier

Beispieldurchlauf



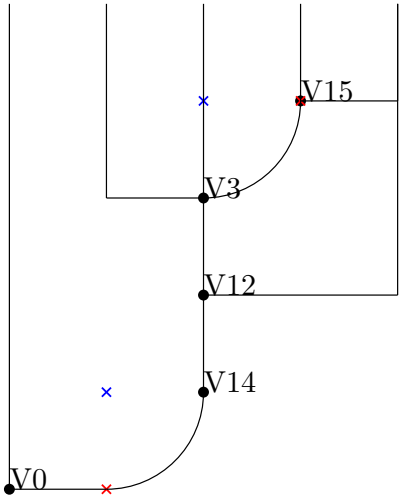
snap #2 placing initial vertex of tier

Beispieldurchlauf



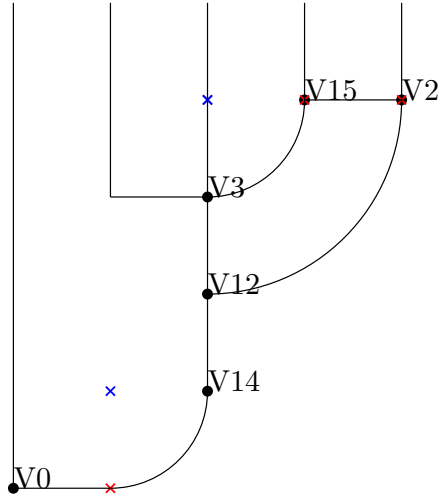
snap #3 placing initial vertex of tier

Beispieldurchlauf



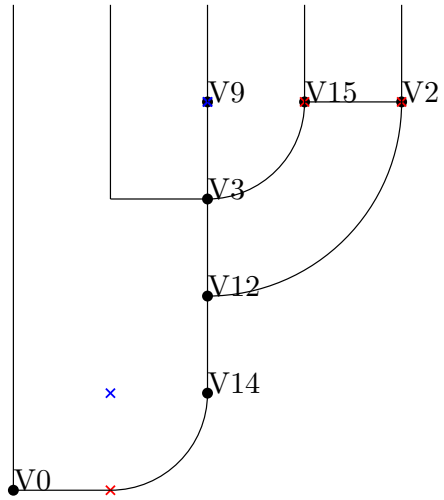
snap #4 placing initial vertex of tier

Beispieldurchlauf



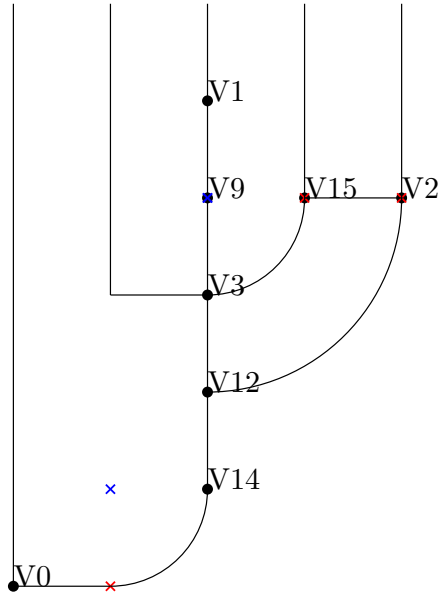
snap #5 placing last vertex of tier

Beispieldurchlauf



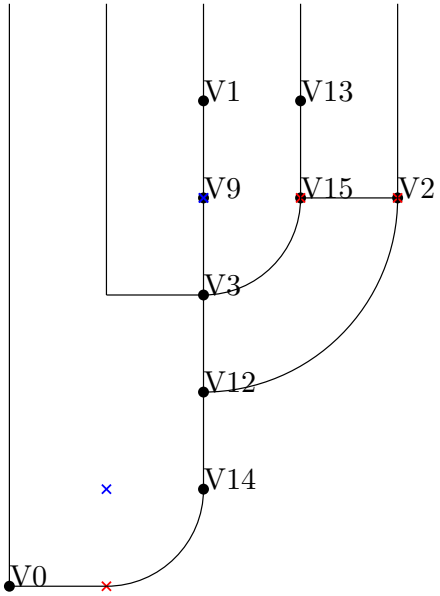
snap #6 placing initial vertex of tier

Beispieldurchlauf



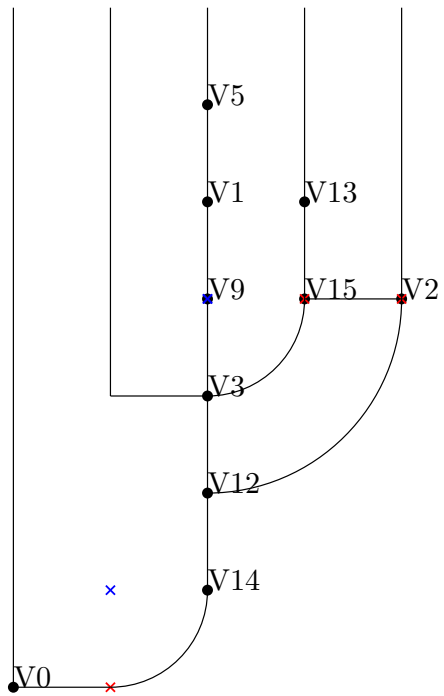
snap #7 placing initial vertex of tier

Beispieldurchlauf



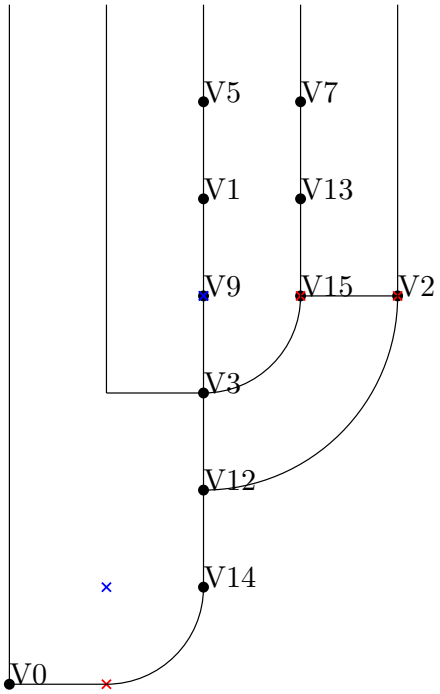
snap #8 placing initial vertex of tier

Beispieldurchlauf



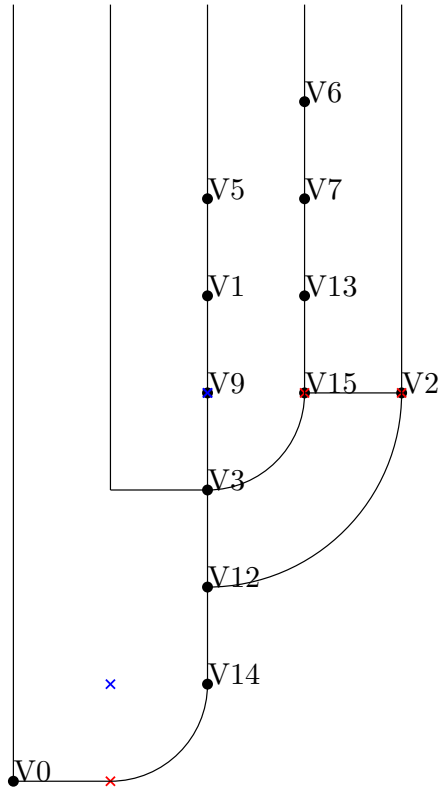
snap #9 placing initial vertex of tier

Beispieldurchlauf



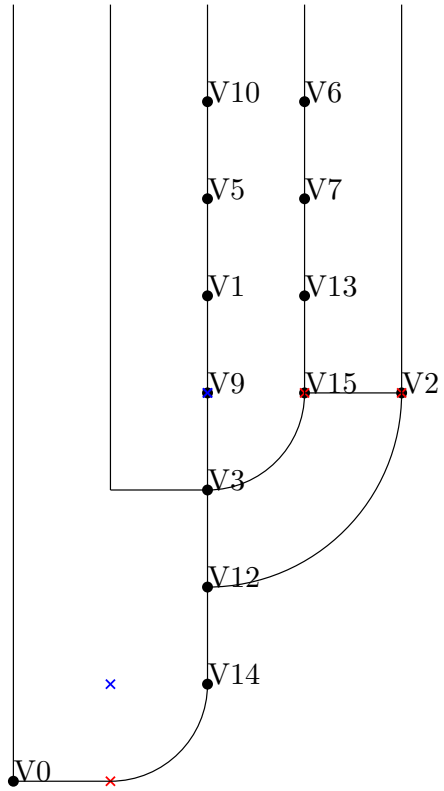
snap #10 placing initial vertex of tier

Beispieldurchlauf



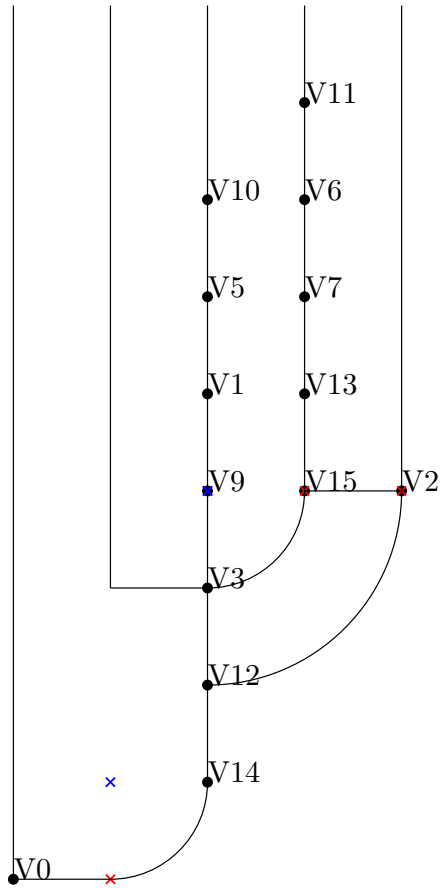
snap #11 placing initial vertex of tier

Beispieldurchlauf



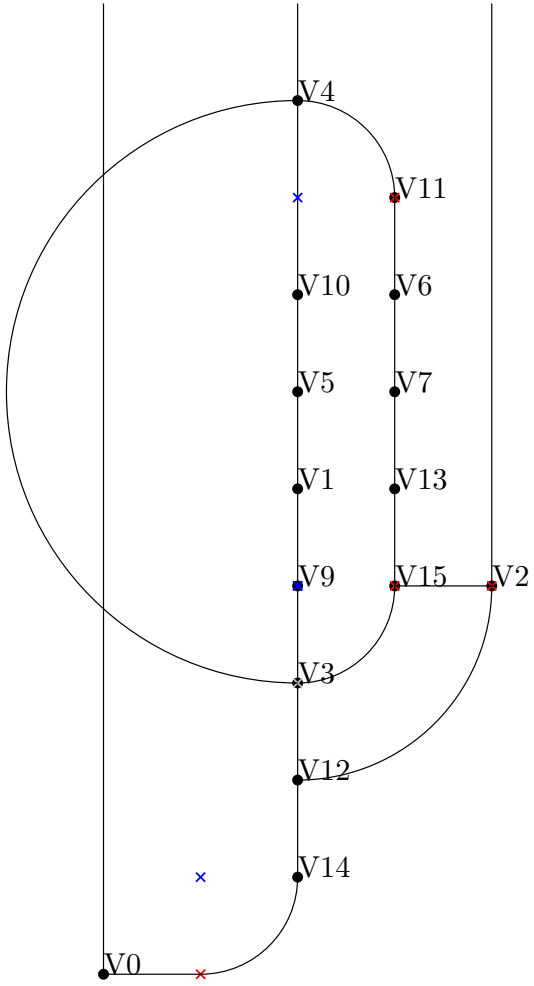
snap #12 placing initial vertex of tier

Beispieldurchlauf



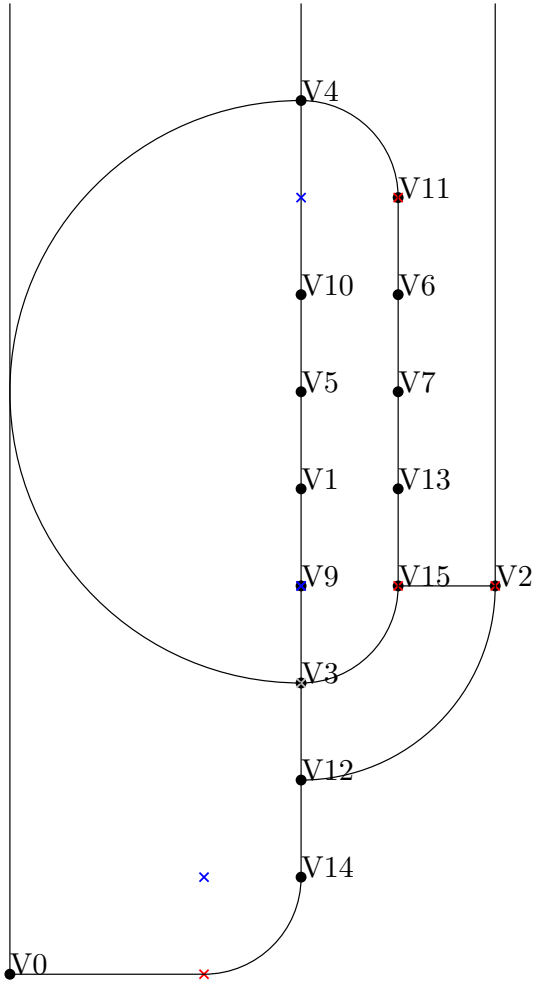
snap #13 placing initial vertex of tier

Beispieldurchlauf



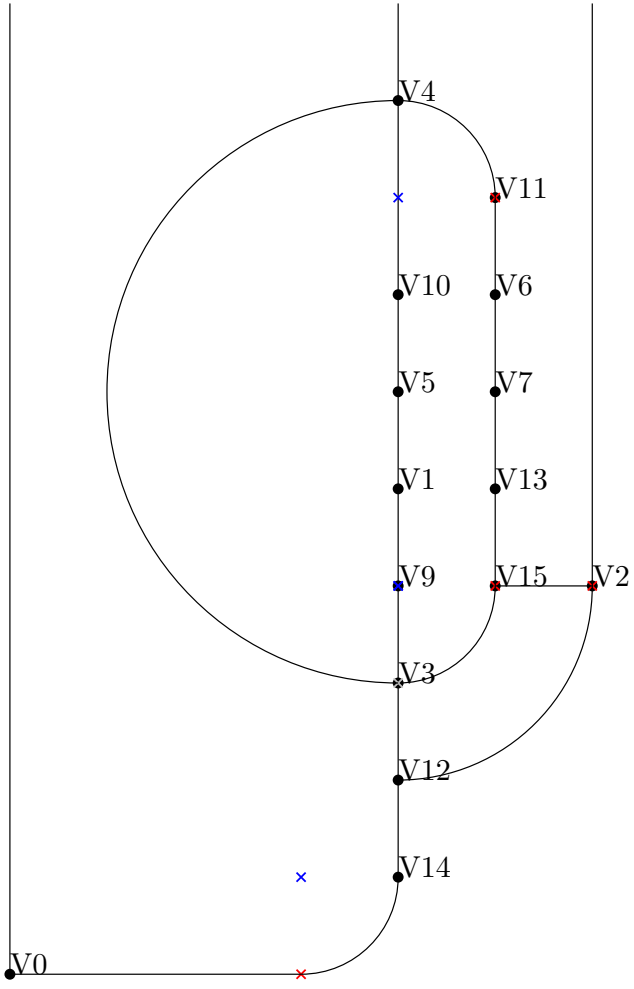
snap #14 placing initial vertex of tier

Beispieldurchlauf



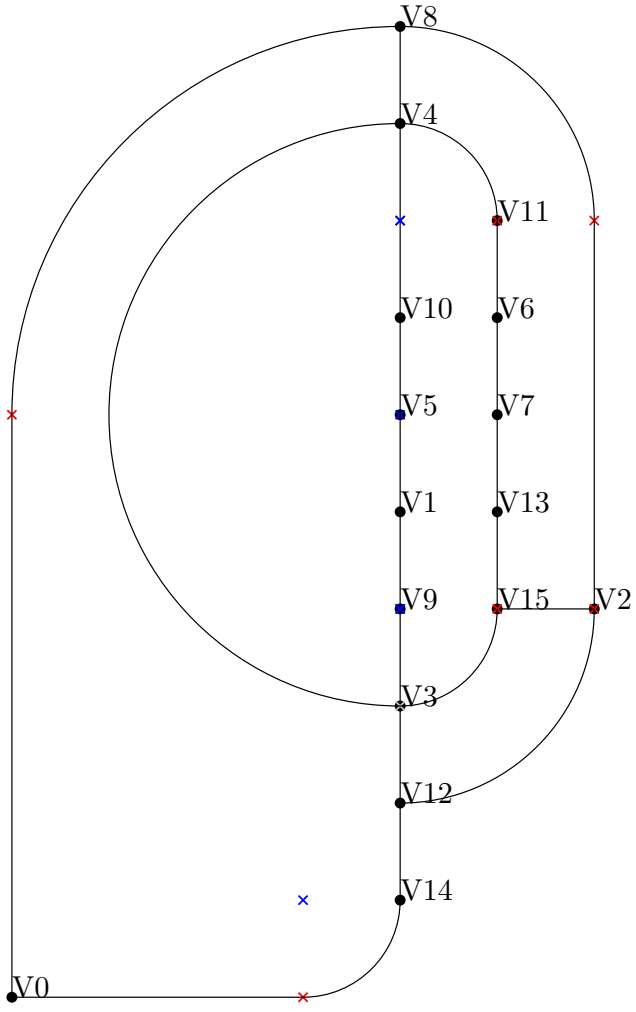
snap #15 moving stuff outside to avoid collision with Lmost edge at V4

Beispieldurchlauf



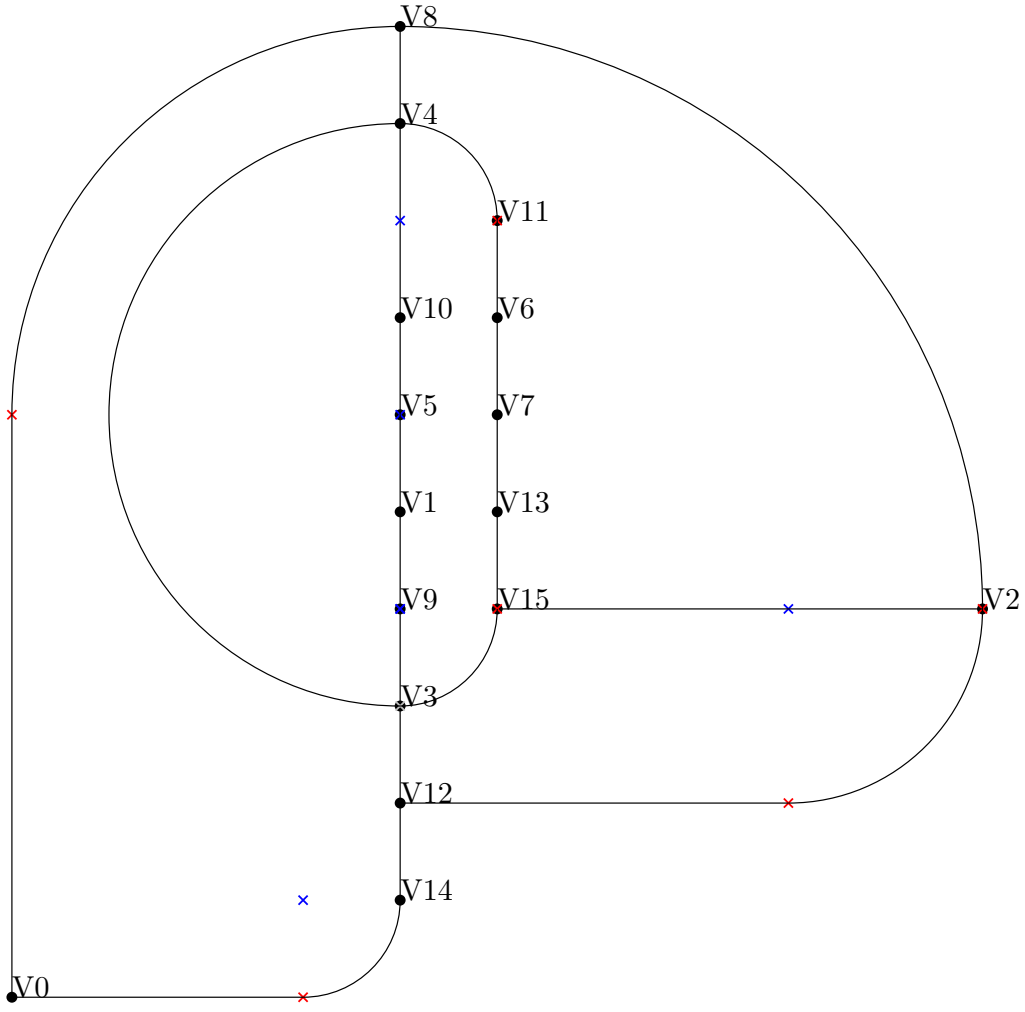

```
snap #16 moving stuff outside to avoid collision with Lmost edge at V4
```

Beispieldurchlauf



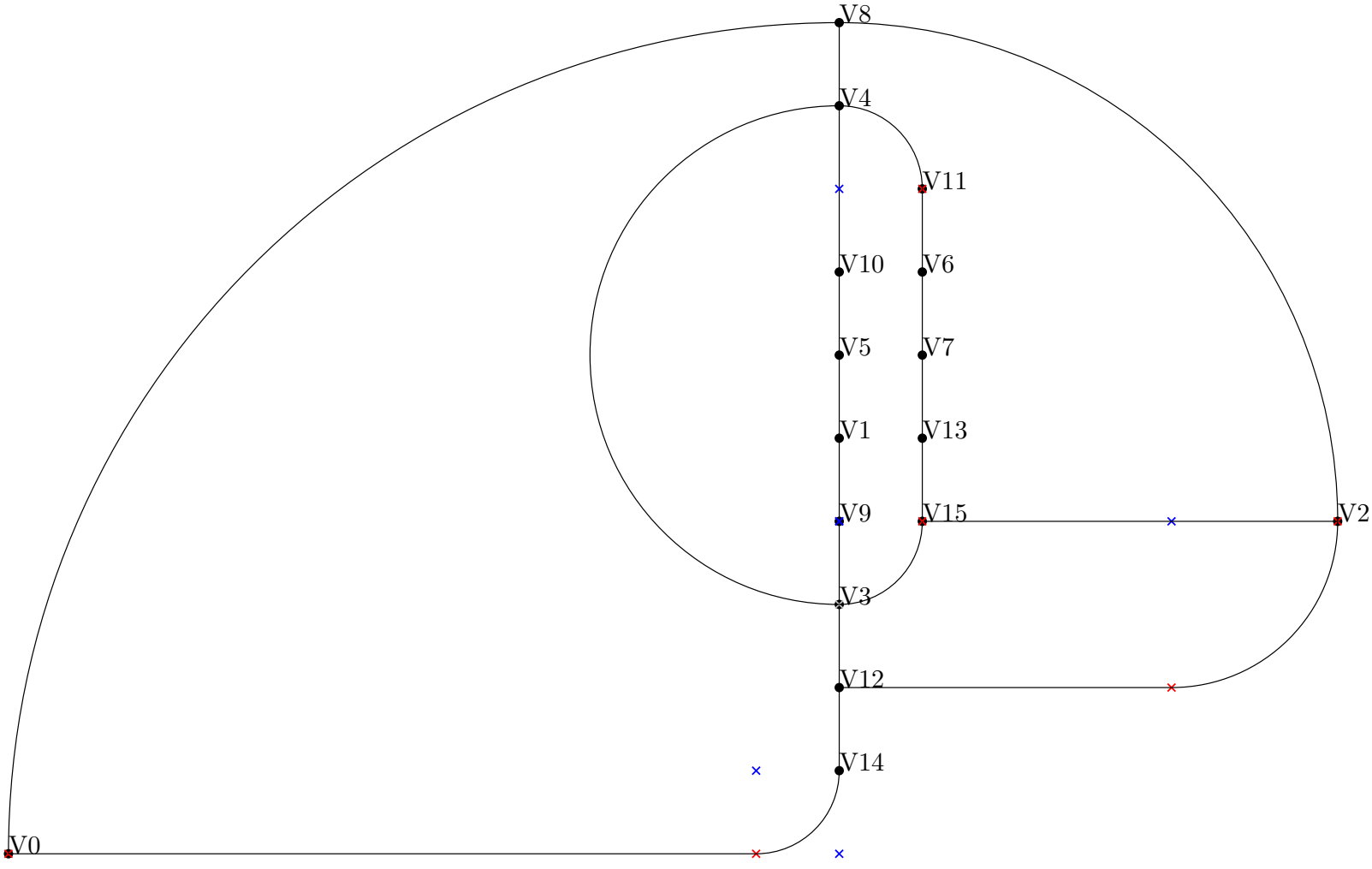
snap #17 placing initial vertex of tier

Beispieldurchlauf



snap #18 adjusting slope of Rmost edge at V8

Beispieldurchlauf



snap #19 adjusting slope of Lmost edge at V8

Eigene Anpassungen für die Implementierung

- Schnitte nur anhand der Ports, statt geometrisch
- Zeichnen von glatt-orthogonalen Kanten
- Finden von Überschneidungen von Kreissegmenten und Linien

Optimierungen:

- Plateaus mit gleicher Höhe
- Korrektur der Steigung der Kanten nur wenn nötig
- Keine G-Kanten

Plateaus mit gleicher Höhe

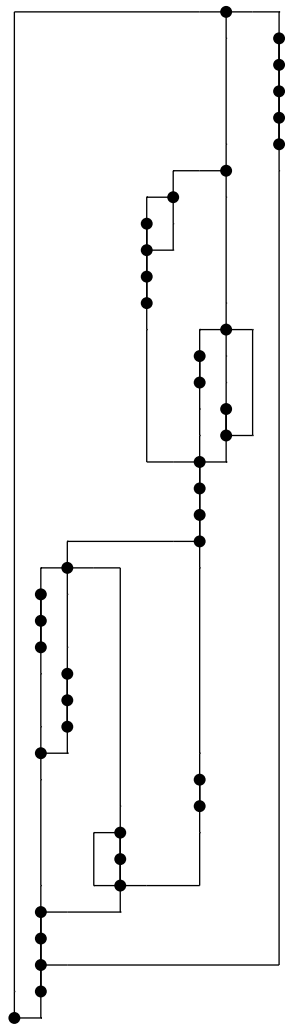


Abbildung: Ohne Komprimierung
Höhe 38

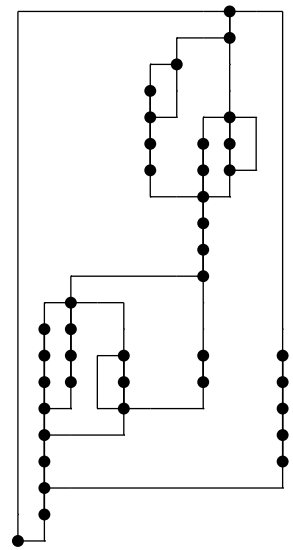


Abbildung: Mit Komprimierung
Höhe 20

Korrektur der Steigung der Kanten nur wenn nötig

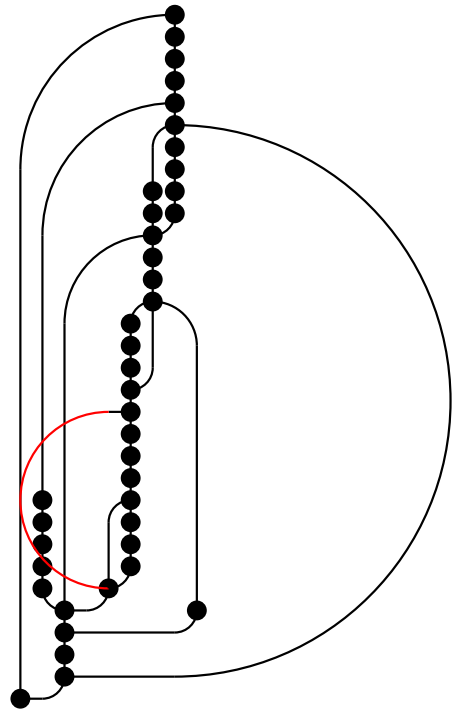
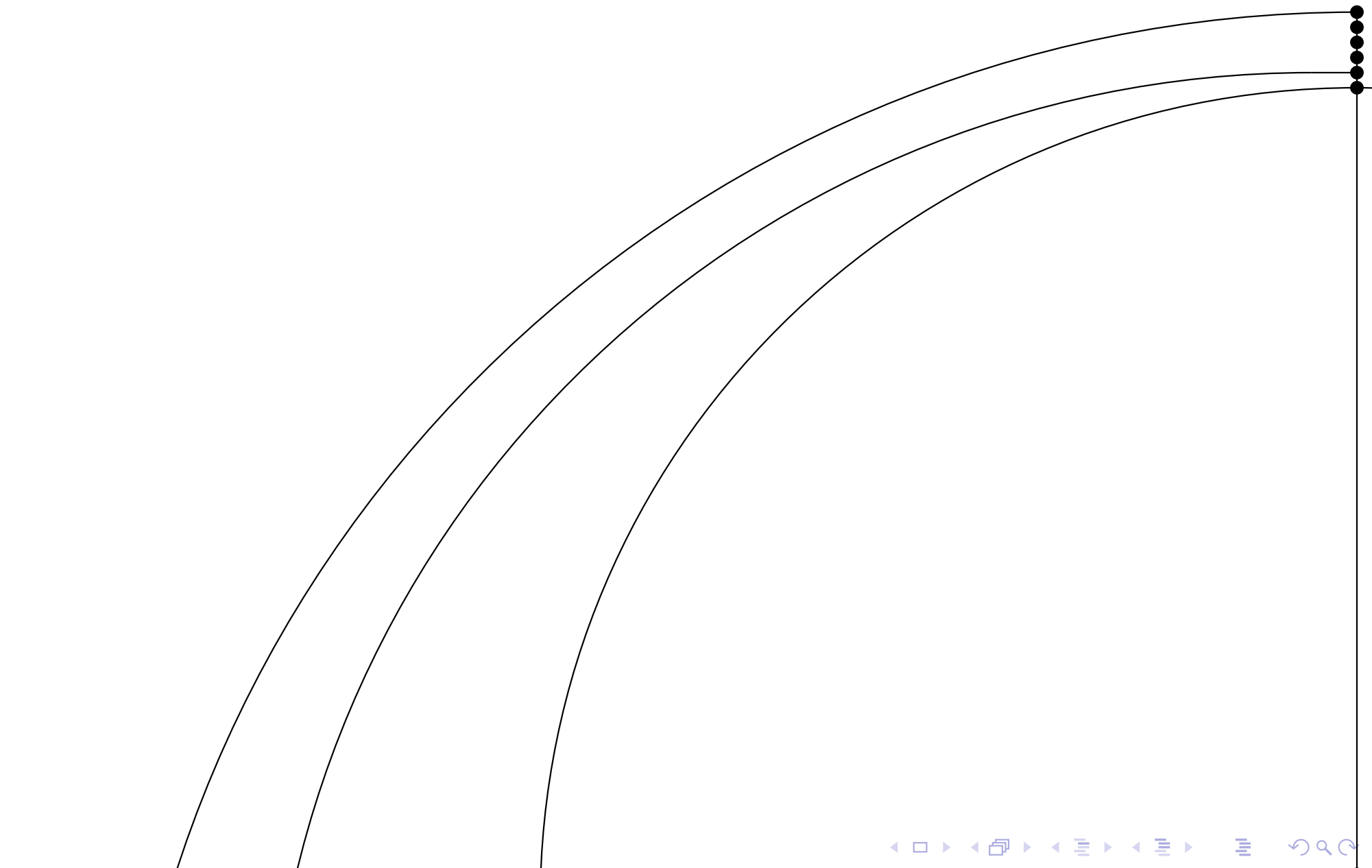


Abbildung: Graph mit Überschneidung

Korrektur der Steigung der Kanten nur wenn nötig



Korrektur der Steigung der Kanten nur wenn nötig

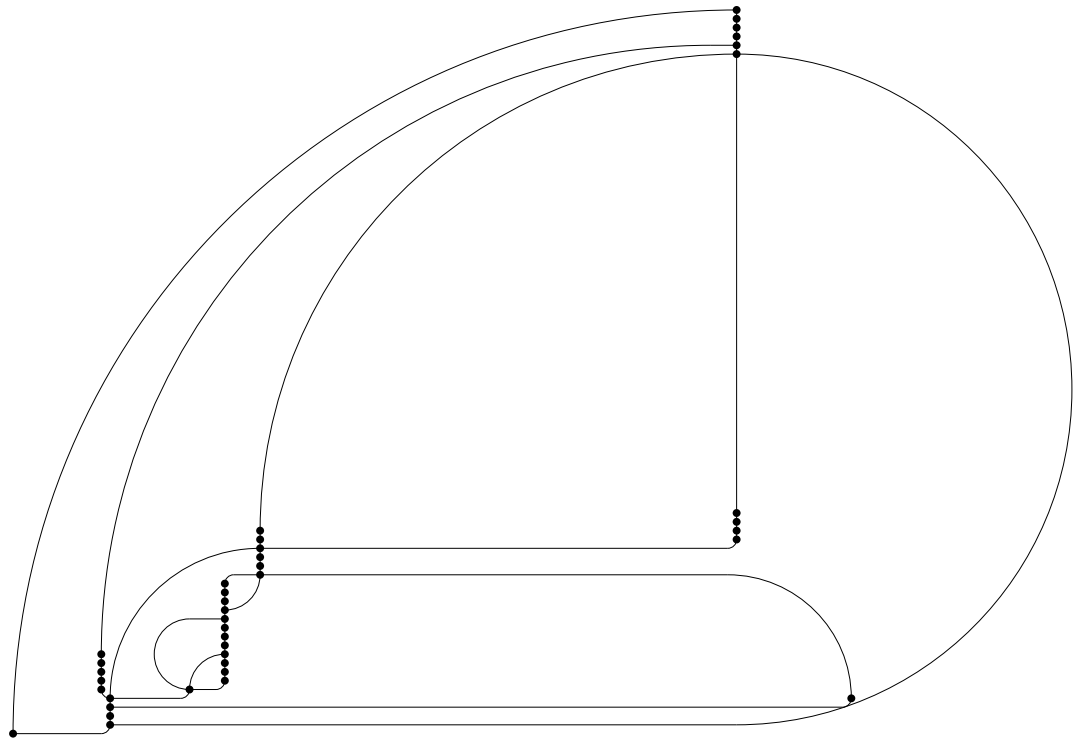


Abbildung: Graph ohne Überschneidung (40%)

Korrektur der Steigung der Kanten nur wenn nötig

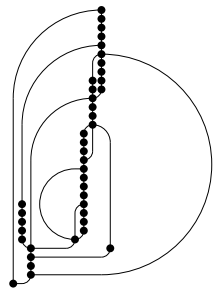


Abbildung: Korrektur der Steigung ausgesetzt (40%)

Korrektur der Steigung der Kanten nur wenn nötig

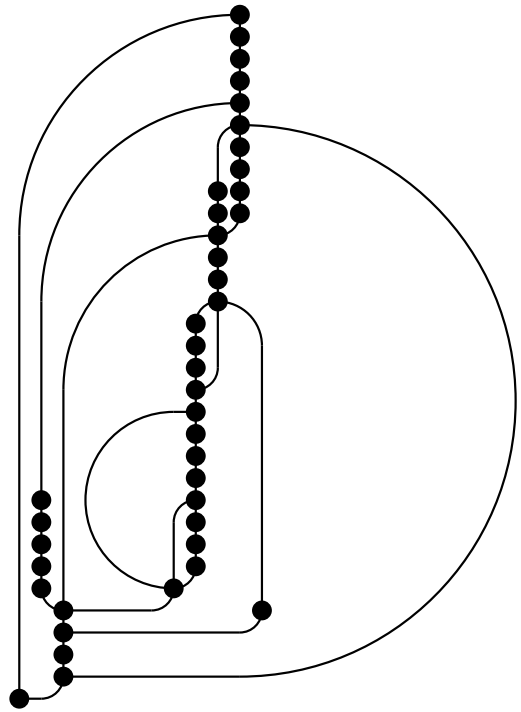


Abbildung: Korrektur der Steigung ausgesetzt

Platzbedarf von G-Kanten

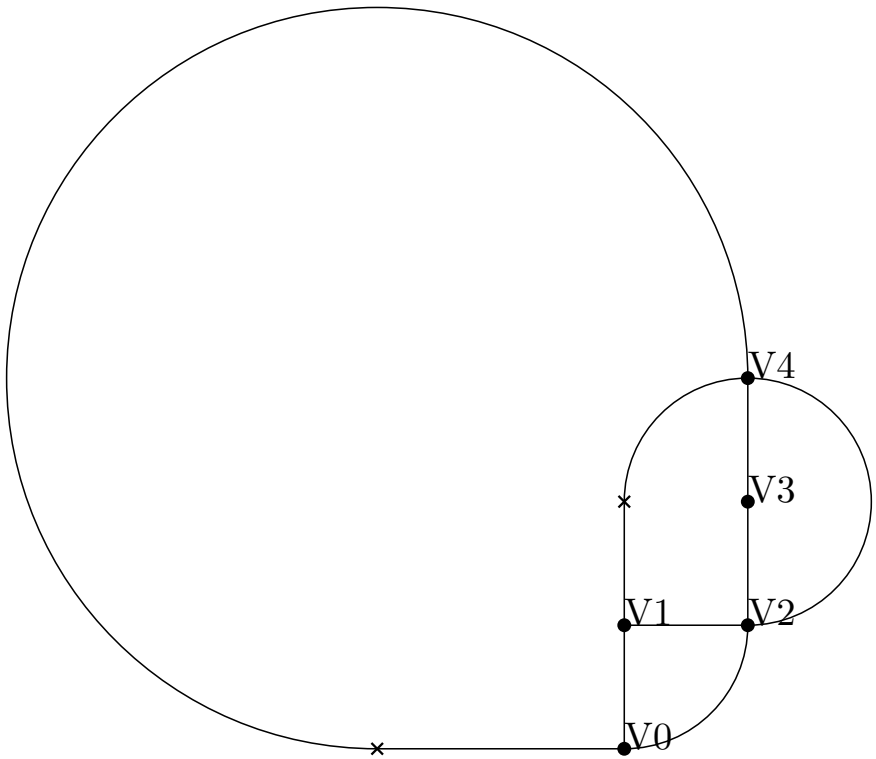


Abbildung: Mit Kante $\{0, 4\}$ aus zwei Segmenten.

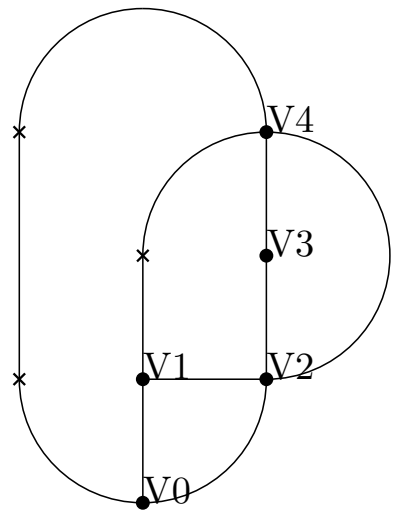


Abbildung: Mit Kante $\{0, 4\}$ aus drei Segmenten.

Bewertungskriterien für glatt-orthogonale Zeichnungen

- Fläche in Gittereinheiten
- Fläche in Abhängigkeit der Anzahl Knoten
- Anzahl Segmente in der Zeichnung

Die Testgraphen

- 844 Rome-Graphen
- Oft zwischen 11 und 20 Knoten, durchschnittlich 18,2
- Bis zu 51 Knoten
- Nur 4-planar und zweifach zusammenhängend

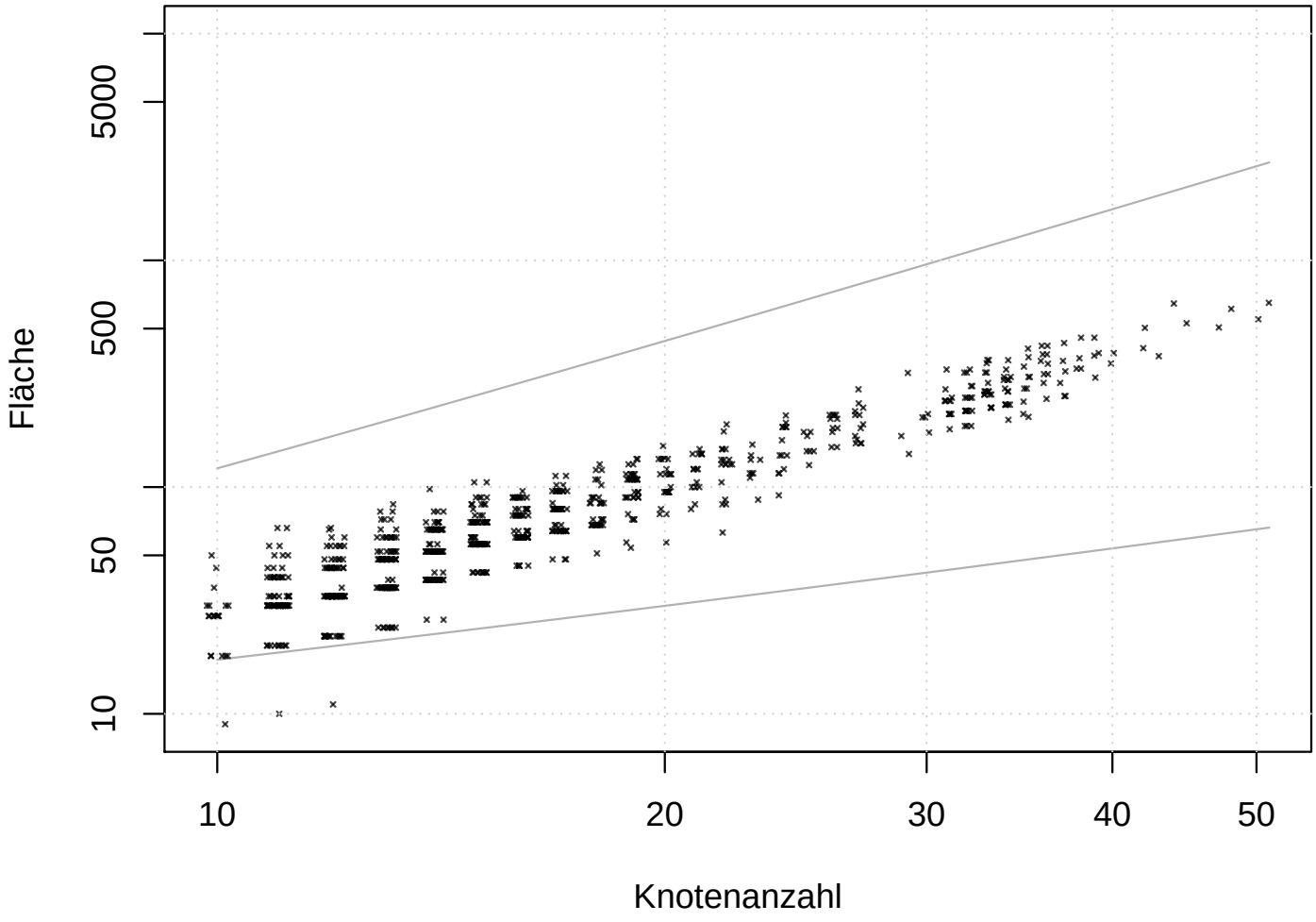


Abbildung: OC_3-Zeichnungen ohne Kompression

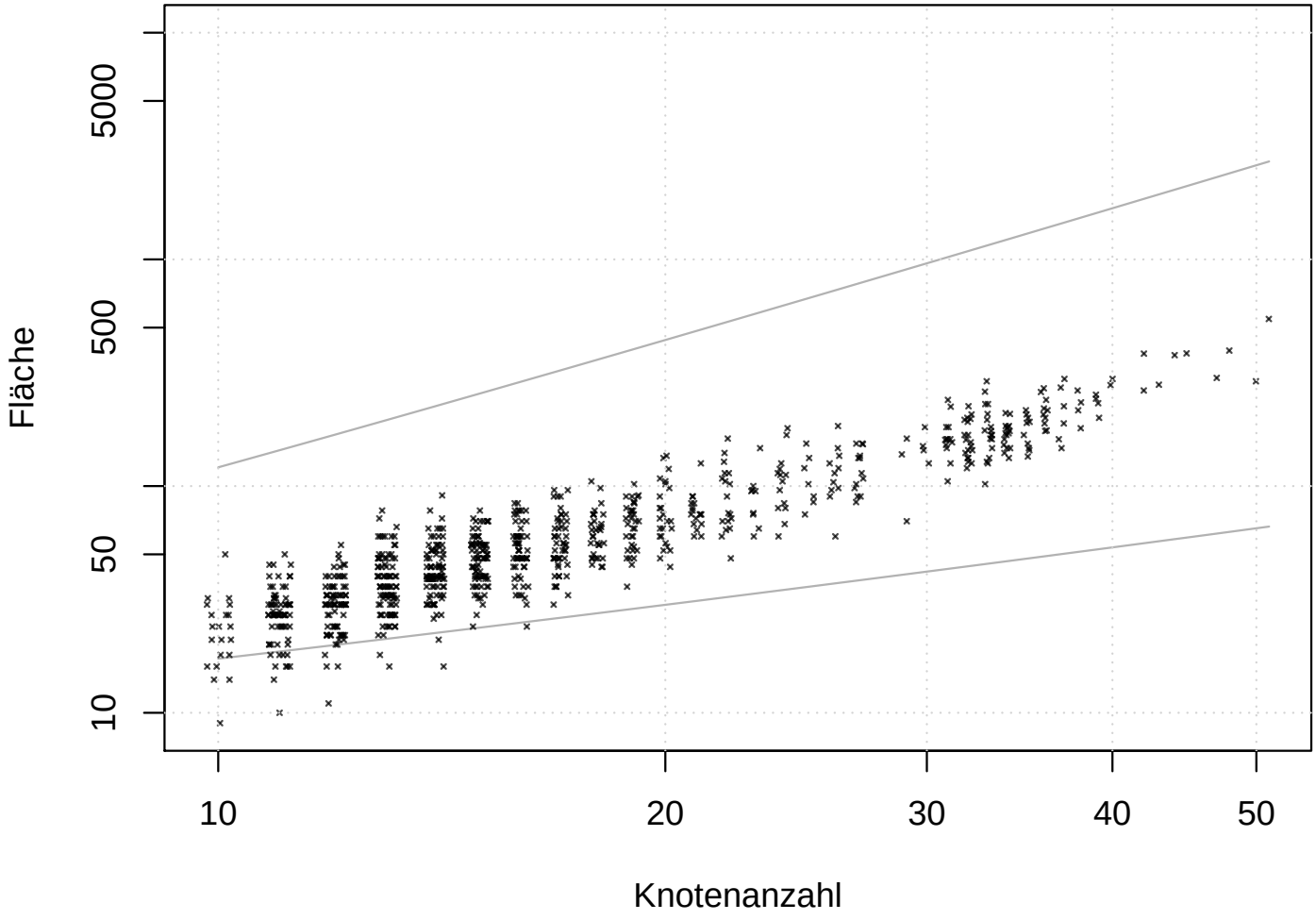


Abbildung: OC_3-Zeichnungen nach Kompression, ohne Stufen

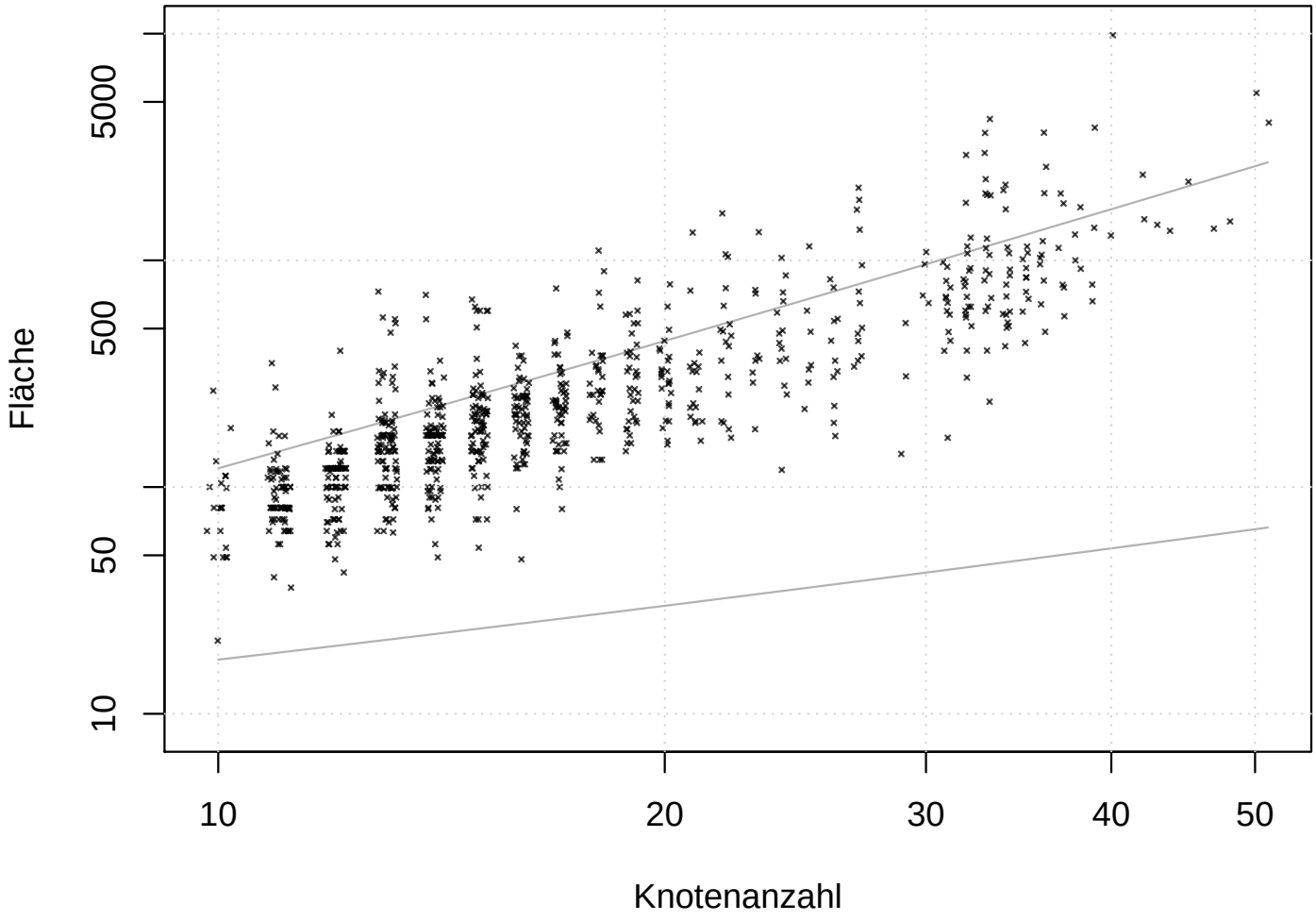


Abbildung: SC_2-Zeichnungen, Steigung aller L-Kanten berichtigt

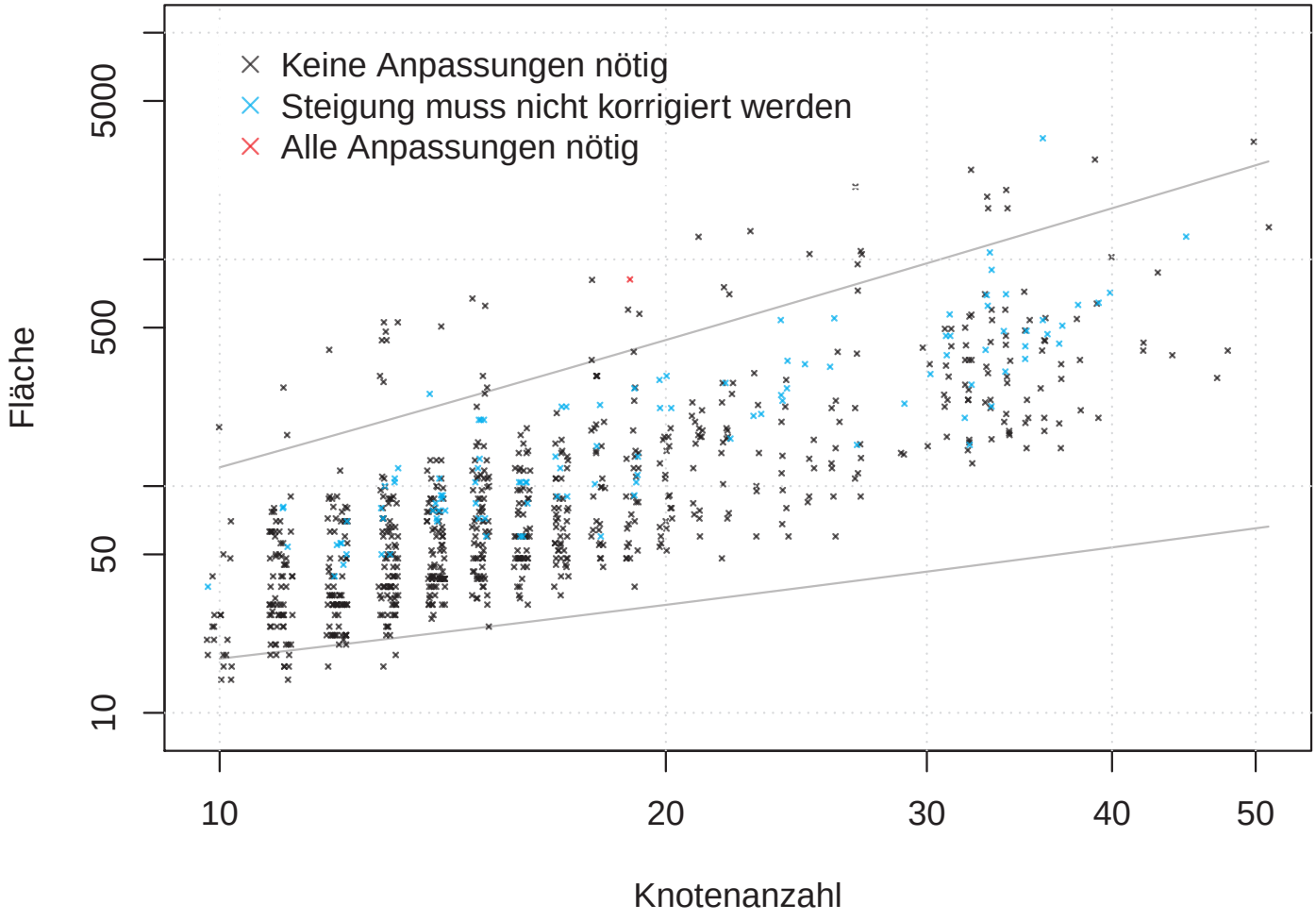


Abbildung: SC_2-Zeichnungen, Steigung L-Kanten bei Bedarf berichtigt

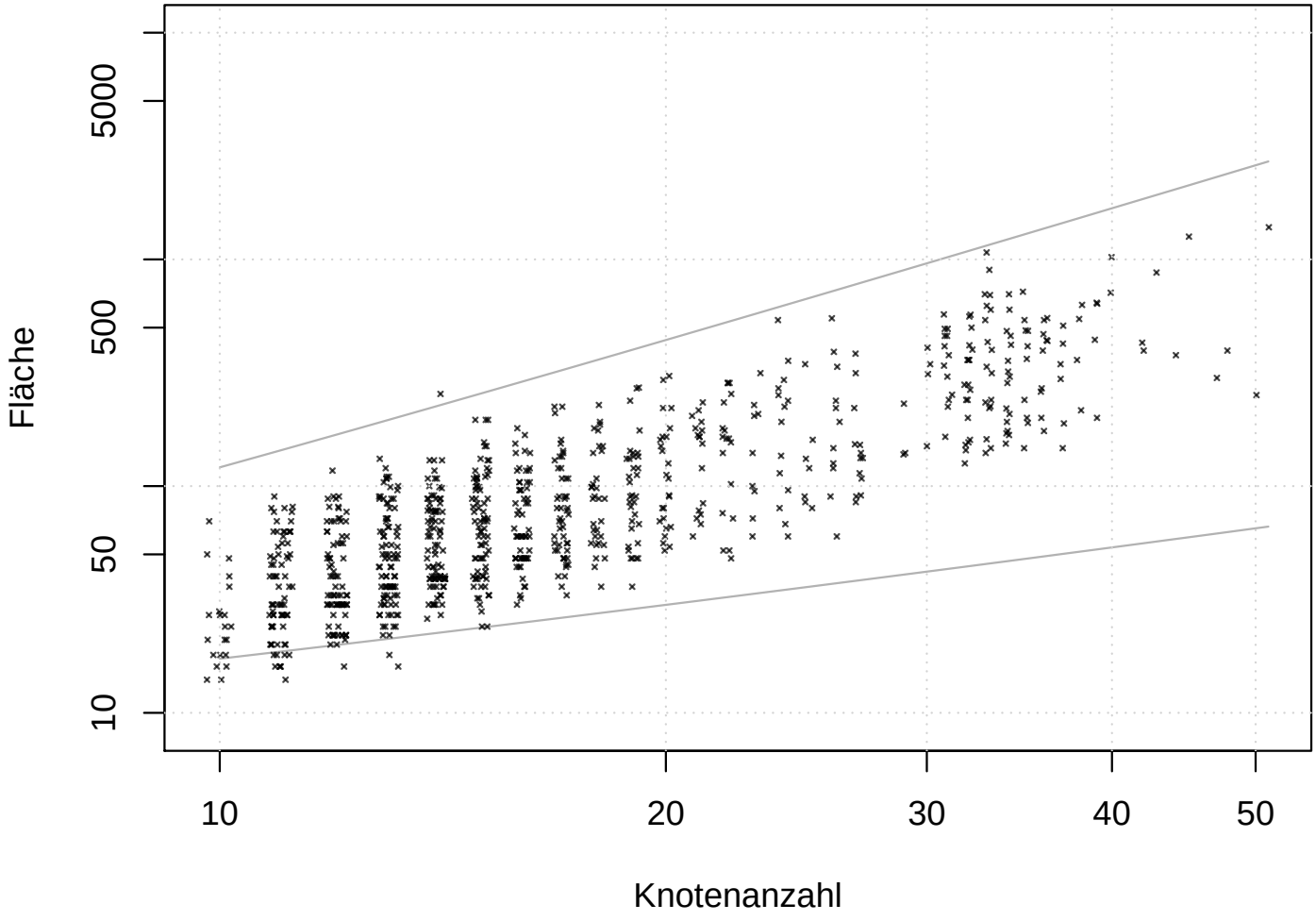


Abbildung: SC_2-Zeichnungen, ohne G-Kanten

Ausblick

- Bessere Laufzeit bei Kollisionsprüfung
- Noch platzsparendere Zeichnungen
- Untersuchung allgemeinerer Graphen
  - Nicht planar
  - nicht zweifach zusammenhängend
  - nicht zusammenhängend
  - höherer Grad als 4
  - ...
- Anwendung: automatischer Liniennetzplan

Literatur


Muhammad Jawaherul Alam, Michael A. Bekos, Michael Kaufmann, Philipp Kindermann, Stephen G. Kobourov und Alexander Wolff: Smooth Orthogonal Drawings of Planar Graphs.

In: Alberto Pardo und Alfredo Viola (Herausgeber): *LATIN 2014: Theoretical Informatics*, Band 8392 der Reihe *Lecture Notes in Computer Science*, Seiten 144–155. Springer, Heidelberg, 2014.


Michael A. Bekos, Michael Kaufmann, Stephen G. Kobourov und Antonios Symvonis: Smooth Orthogonal Layouts.

In: Walter Didimo und Maurizio Patrignani (Herausgeber): *Graph Drawing*, Band 7704 der Reihe *Lecture Notes in Computer Science*, Seiten 150–161. Springer, Heidelberg, 2013.


Ulrik Brandes: The left-right planarity test, 2009.

`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.217.9208&rep=rep1&type=pdf`, besucht: April 2014.


Shimon Even und Robert E. Tarjan: Computing an st-Numbering.

Theoretical Comp. Sci., 2(3):339–344, 1976.


Yanpei Liu, Aurora Morgana und Bruno Simeone: A Linear Algorithm for 2-bend Embeddings of Planar Graphs in the Two-dimensional Grid.

Discrete Appl. Math., 81(1–3):69–91, 1998.