Julius-Maximilians-Universität Würzburg
Institut für Informatik

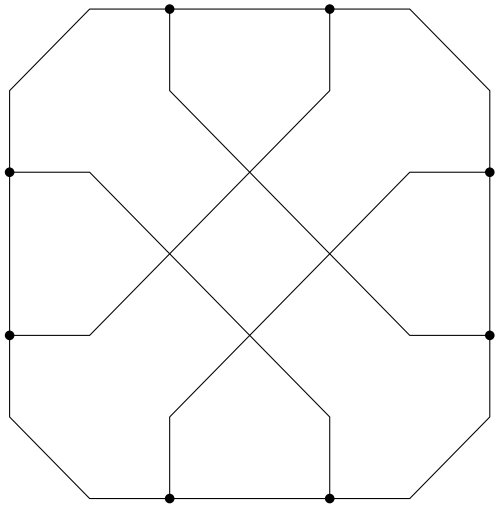
Schieforthogonale Zeichnungen von Graphen

Masterarbeit

zur Erlangung des Hochschulgrades

Master of Science (M.Sc.)

MA/Bilder/Titelbild.eps MA/Bilder/Titelbild.eps



vorgelegt von

Wadim Reimche

Tag der Einreichung: 4. Mai 2015

Betreuer:
Prof. Dr. Alexander Wolff
Dipl.-Inf. Philipp Kindermann
Lehrstuhl für Informatik I

Inhaltsverzeichnis

Kapitel 1

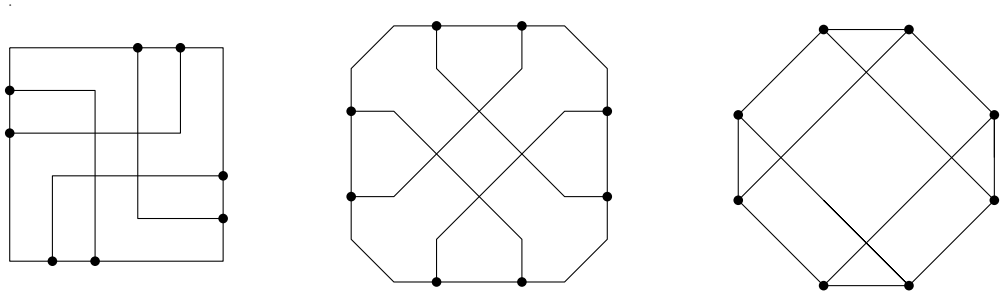
Einleitung

1.1 Motivation

Das Graphzeichnen ist ein interessantes Teilgebiet der Graphentheorie, bei dem man, wie das Wort schon sagt, versucht Graphen benutzerfreundlich oder nach bestimmten Regeln „schön“ darzustellen.

In der Graphentheorie unterscheidet man planare und nicht-planare Graphen. Planare Graphen sind Graphen, die ohne Kreuzungen gezeichnet werden können. Eine häufig verwendete Zeichenart ist die orthogonale Zeichnung, die wir im Folgenden auch mit OG-Zeichnung abkürzen. Bei der OG-Zeichnung werden ausschließlich horizontale und vertikale Kantenzüge (siehe Abb. 1.1(a)) verwendet und so eine Zeichnung erlaubt es auch nicht-planare Graphen zu zeichnen. Diese Arbeit handelt von schieforthogonalen Zeichnungen (eng. slanted orthogonal drawing), die das erste mal im Jahr 2013 von Bekos et al. [2] in „*Slanted Orthogonal Drawings*“ beschrieben wurden. Schieforthogonale Zeichnungen, welche wir im Weiteren auch SLOG-Zeichnung nennen, erlauben im Gegensatz zu OG-Zeichnungen auch diagonale Kantenzüge (siehe Abb. 1.1(b)). Diese Art von Zeichnung wird im nächsten Kapitel genauer definiert.

Die OG-Zeichnung hat den Nachteil, dass man Kreuzungen nicht so gut von Knoten unterscheiden kann. Die SLOG-Zeichnung versucht dieses Problem zu umgehen, indem sie diagonale Kantensegmente einführt, auf denen alle Kreuzungen



(a) (b) (c)

Abbildung 1.1: (a) orthogonal (b) schieforthogonal (c) oktilinear

liegen müssen (siehe Abb. 1.1(b)). Diese Kreuzungen unterscheiden sich dann von den Knoten, weil aus ihnen lediglich horizontale und vertikale Kantenzüge hinausgehen. Bei beiden Modellen arbeitet man mit sogenannten Respräsentationen, welche festlegen in welcher Reihenfolge die Kanten Knicke aufweisen. Hierbei werden nicht die Längen der Kantenzüge festgelegt. Aus diesen Repräsentationen kann man jedoch, wie wir später sehen werden, nicht direkt eine Zeichnung erzeugen.

Wir werden schnell feststellen, dass es nicht leicht ist, automatisiert SLOG-Zeichnungen aus beliebigen Repräsentationen zu generieren.

1.2 Bisheriger Stand der Forschung

Aus der oben genannten Arbeit entstand ein offenes Problem, welches sich mit der Zeichenbarkeit von knickminimalen schieforthogonalen Repräsentationen befasst [2]. Bei dem ursprünglichen orthogonalen Modell ist bereits bekannt, dass sich aus einer Repräsentation stets eine Zeichnung generieren lässt, die diese verwirklicht. Dies wurde von Roberto Tamassia gezeigt [8].

Allgemein ist das Gebiet der OG-Zeichnungen in unterschiedlichen Ausprägungen seit etwa 1980 ein Thema, das ausgiebig erforscht wurde. Anwendungen der OG-Zeichnungen liegen z.B. in Schaltkreisdiagrammen oder in der Grundrissplanung [4],[9]. Es wurde auch ein anderer Ansatz verfolgt um die OG-Zeichnungen zu veranschaulichen. Bei den „Smooth Orthogonal Drawings“ ersetzt man Knicke durch Kreissegmente, welche zusätzlich zu orthogonalen Kantensegmenten verwendet werden können [1].

Es gibt zudem ein erweitertes oktilineares Modell, bei dem man auch diagonale Zugänge zu Knoten zulässt (siehe Abb. 1.1(c)). Somit können die Knoten bis zu acht Nachbarn haben. Im Gegensatz dazu haben Knoten im orthogonalen Modell bis zu vier Nachbarn. Dieses Modell findet z.B. Anwendung bei Zeichnungen von U-Bahnplänen und Straßenbahnplänen [7]. Bei dem oktiliniaren Modell ist bereits bekannt, dass nicht jede Repräsentation gezeichnet werden kann [3]. Es ist jedoch noch nicht bekannt wie es mit der Zeichenbarkeit einer SLOG-Repräsentation aussieht.
Hierfür wurde eine Heuristik beschrieben, welche diese mit einer gewissen Anzahl an zusätzlichen Knicken zeichnen kann [2].

1.3 Problemstellung und mein Beitrag

Bekos et al. [2] haben gezeigt, dass es möglich ist mit einem Algorithmus aus einer knickminimalen orthogonalen Repräsentation eine knickminimale SLOG-Repräsentation zu generieren und umgekehrt. Die Einbettung, die zugrunde liegt, ist jedoch stets vorgegeben. Denn nach Garg und Tamassia [6] ist es i.A. ein NP-schweres Problem, zu einem Graphen eine orthogonale Einbettung zu finden, die knickminimal unter allen Einbettungen ist. Jedoch ist es nicht klar, ob jede SLOG-Repräsentation eines Graphen eine Zeichnung induziert. Um dieses Pro-

blem zu lösen, könnte man versuchen, einen Zeichenalgorithmus anzugeben, der jede SLOG-Repräsentation zeichnet und somit diese Tatsache beweisen.

Alternativ kann man das Problem rein theoretisch nicht-konstruktiv mit einem Beweis lösen. Es wurde bereits ein Algorithmus in Form eines Linearen Programms angegeben, welcher wahrscheinlich alle SLOG-Repräsentation zeichnen kann [2]. Jedoch fehlt noch der Beweis, dass dies tatsächlich der Fall ist.

In dieser Arbeit werden wir mit Hilfe von sogenannten Cuts eine Heuristik beschreiben, welche versucht die Repräsentation mit wenigen Änderungen in eine Zeichnung umzuwandeln.

Hierbei verknüpfen wir Elemente von vorhandenen Heuristiken und Algorithmen um eine Heursitik zu beschreiben, welche unter anderem die knickminimale SLOG-Repräsentation sehr genau zeichnet und sie nur durch Hinzufügen von wenigen Paaren von Rechts- und Linksknicken verändert. In der Nachbearbeitung können dann mit einer allgemeinen Methode zum Finden von Schnitten eventuell weitere solche überschüssigen Knickkombinationen aus der Zeichnung entfernt werden.

Diese Heuristik wurde von mir zudem noch in Java, für eine Analyse an mehreren Testmengen von Graphen, implementiert.

Kapitel 2

Orthogonale und schieforthogonale Zeichnungen

In diesem Kapitel werden die grundlegenden Begriffe dieser Arbeit definiert und mit einigen Beispielen anschaulich gemacht. Zudem werden einige Zusammenhänge zwischen den definierten Begriffen dargestellt.

2.1 Definitionen

In dieser Arbeit beschäftigen wir uns ausnahmslos mit endlichen Graphen ohne Mehrfachkanten und ohne Schlaufen.

Als ersten Schritt erläutern wir was genau eine Zeichnung eines Graphen ist.

2.1.1 Definition *Eine* ***Zeichnung*** Γ *ist eine Darstellung eines Graphen* $G = (V, E)$ *in der Zeichenebene, auch euklidische Zahlenebene genannt, die folgende Eigenschaften erfüllt:*

(i) Jeder Knoten wird durch einen Punkt in der Zeichenebene dargestellt.

(ii) Die Kanten werden durch offene Jordankurven repräsentiert, die ihre adjazenten Knoten als Endpunkte haben.

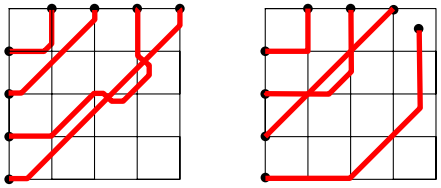
(iii) Zwei solche Jordankurven schneiden sich in maximal endlich vielen Punkten. Falls diese einen Endpunkt gemeinsam haben, dürfen sie sich nicht schneiden.

(iv) Die Punkte, welche die Knoten repräsentieren, sind paarweise disjunkt zu den Jordankurven.

Zeichnungen von Graphen konnten wir bereits in der Einleitung in Abb. 1.1 sehen.

Im Folgenden definieren wir eine besondere Art der Zeichnung, die schieforthogonale Zeichnung.

2.1.2 Definition *Eine Zeichnung* Γ *eines Graphen* $G = (V, E)$ *in der euklidischen Zahlenebene ist genau dann eine* ***schieforthogonale Zeichnung*** *oder* ***SLOG-Zeichnung****, wenn sie folgende Kriterien erfüllt:*



(a) (b)

Abbildung 2.1: (a) zulässige Kanten (b) nicht-zulässige Kanten und Knoten

(i) *Alle Knoten liegen auf ganzzahligen Gitterpunkten.*

(ii) *Alle Kanten verbinden die zugehörigen Knoten und bestehen aus Kombinationen von horizontalen, vertikalen und diagonalen Segmenten.*

(iii) *Ein diagonales Kantensegment ist niemals inzident zu einem Knoten.*

(iv) *Kantenkreuzungen bestehen stets aus Kreuzungen von zwei diagonalen Segmenten.*

(v) *Zwei aufeinanderfolgende Kantensegmente bilden einen Winkel von mindestens* 135°.

2.1.3 Bemerkung Aus *(ii)* und *(iii)* folgt sofort, dass nur Graphen mit einem Maximalgrad von 4, d.h. jeder Knoten hat maximal 4 Nachbarn, eine schieforthogonale Zeichnung haben können.

Mit Hilfe einer Abbildung erläutern wir, was genau eine schieforthogonale Zeichnung zulässt und was nicht: Die obere Definition hat zur Folge, dass man Kreuzungen viel leichter erkennt, da an der Kreuzungsstelle wegen *(iii)* keine Knoten liegen können, wie man bei den unteren beiden Kanten in Abb. 2.1(a) sieht. Jedoch können wir in der Zeichnung auch erkennen, dass Kanten unschön werden können, da wir nicht gefordert haben, dass Knicke auf ganzzahligen Gitterpunkten liegen müssen. Abb. 2.1(b) zeigt nicht erlaubte Kanten, die von oben nach unten (d.h. gesehen vom linken Endpunkt der Kanten) *(v)*, *(iv)*, *(iii)* und *(iv)* verletzen, und als letztes verletzt der Knoten oben das Kriterium *(i)*.

Als nächstes definieren wir eine Einbettung eines Graphen, welche als Grundlage für die Zeichnung eines Graphen dient.

2.1.4 Definition *Sei G ein Graph. Dann ist eine* ***Einbettung*** *eine Zuweisung eines Vektors r_v zu jedem Knoten v des Graphen, in dem jeder Nachbar von v genau einmal vorkommt.*
Eine Zeichnung, deren zu den Nachbarn gehörige Kanten in der Reihenfolge des Vektors den Knoten gegen den Uhrzeigersinn verlassen, ***respektiert*** *die Einbettung. Hierbei stellt man sich den Vektor zirkulär vor, also kann bei jedem Knoten begonnen werden, solange die Reihenfolge danach passt.*
Existiert eine ebene Zeichnung, also eine Zeichnung ohne Kantenkreuzungen, die eine Einbettung respektiert, so nennt man diese eine ***planare Einbettung****.*

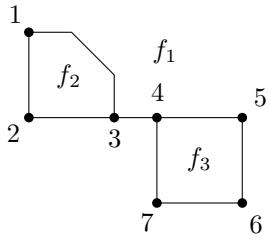


Abbildung 2.2: schieforthogonale Zeichnung

2.1.5 Bemerkung Im Falle der planaren Einbettung kann man äquivalent jeder Facette f einen Vektor r_f zuweisen, in den alle angrenzenden Knoten eingetragen werden.
Eine Zeichnung respektiert genau dann so eine planare Einbettung, wenn bei allen Facetten die Knoten gegen den Uhrzeigersinn in der Reihenfolge des Vektors am Rand der Facette verlaufen. Insbesondere kommt nach dem untersten Knoten im Vektor wiederum der oberste Knoten.
Hierbei kann es passieren, dass Knoten bei der Facette mehrfach aufgelistet werden, wie man im folgenden Beispiel sehen kann.

2.1.6 Beispiel Wir betrachten die folgenden zwei planaren Einbettungen eines Graphen mit $V = \{1, 2, \ldots, 7\}$. Zunächst zeigen wir eine planare Einbettung nach Def. 2.1.4:

$$r_1 = (2, 3);\ r_2 = (3, 1);\ r_3 = (1, 2, 4);\ r_4 = (3, 7, 5);$$
$$r_5 = (4, 6);\ r_6 = (5, 7);\ r_7 = (4, 6)$$

Nun die zweite planare Einbettung nach Bem. 2.1.5:

$$r_{f_1} = (1, 3, 4, 5, 6, 7, 4, 3, 2);\ r_{f_2} = (1, 2, 3);\ r_{f_3} = (4, 7, 6, 5)$$

In Abb. 2.2 sieht man eine schieforthogonale Zeichnung, die beide planare Einbettungen respektiert und zeigt, dass diese somit das Gleiche darstellen.

Das nächste Beispiel soll den Unterschied zwischen planaren Einbettungen und beliebigen Einbettungen verdeutlichen.

2.1.7 Beispiel Nun betrachten wir den vollständigen Graphen K_4 mit $V = \{1, 2, 3, 4\}$. Eine Einbettung könnte z.B. wie folgt aussehen:

$$r_1 = (2, 3, 4);\ r_2 = (3, 4, 1);\ r_3 = (4, 1, 2);\ r_4 = (1, 2, 3)$$

In Abb. 2.3 sehen wir eine Zeichnung, die diese Einbettung respektiert. An dieser Zeichnung kann man auch erkennen, dass eine Einbettung eines planaren Graphen nicht unbedingt eine planare Einbettung sein muss. Denn die Kreuzung von $(1, 3)$ und $(2, 4)$ kann nicht vermieden werden, wenn die Einbettung respektiert wird. Eine planare Einbettung wäre zum Beispiel die Folgende:

$$r_1 = (2, 3, 4);\ r_2 = (3, 1, 4);\ r_3 = (4, 1, 2);\ r_4 = (2, 1, 3)$$

Diese können wir planar zeichnen, indem wir die Kante $(2, 4)$ in Abb.2.3 außen herum führen.

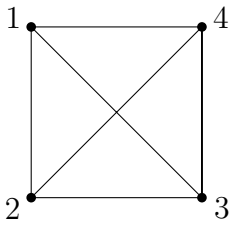


Abbildung 2.3: Zeichnung vom K_4

Sei nun $G = (V, E)$ ein Graph, bei dem jeder Knoten höchstens vier Nachbarn hat, mit einer dazugehörigen Einbettung. Damit wir im Weiteren Kreuzungen besser kontrollieren können, **planarisieren** wir diesen Graphen. Hierbei Dies erreichen wir indem wir auf jede Kreuzung von zwei Kanten in der Einbettung einen **Dummy-Knoten** legen. Diese Knoten bezeichnen wir auch als **c-Knoten** (*eng.: crossing*) und die Knoten aus V als **echte Knoten** oder **r-Knoten** (*eng.: real*), wie es bereits Tamassia gemacht hat [8].
Insbesondere hat unser Graph G dann die Form $G = (R \cup C, E')$, wobei R die Menge der r-Knoten ist, C die Menge der c-Knoten und E' die modifizierte Kantenmenge, die durch Teilen der Kanten an den c-Knoten entsteht. Abhängig von den Endpunkten sprechen wir dann auch von **cc-, rc-** oder **rr-Kanten** dieses Graphen. Dies liefert uns nun einen planaren Graphen.
Nachdem wir nun unseren Graphen planarisiert haben, verwenden wir die ebene Einbettung für die orthogonale Repräsentation.

2.1.8 Definition *Sei $G = (V, E)$ ein planarer zusammenhängender Graph mit einer Facettenmenge F.*
*Eine **orthogonale Repräsentation** oder **OG-Repräsentation** von G ist eine Menge von Listen H(f) für jede Facette $f \in F$, mit Einträgen der Form (e, s, a), wobei $e \in V$, s eine binäre evtl. leere Zeichenkette und $a \in \{90, 180, 270, 360\}$ ist. Diese Listen stellen die Anordnung der Knoten im Uhrzeigersinn an der Facette dar. Die binäre Zeichenkette beschreibt welche Knicke, in welcher Reihenfolge, die Kante, die auf den Knoten e in f folgt, haben soll. Eine* 0 *enstspricht einem Innenwinkel von* $90°$ *und eine* 1 *einem Innenwinkel von* $270°$*. Die Zahl a gibt den Innenwinkel zur nächsten Kante der Facette an.*
Für $h \in H(f)$ bezeichnen wir mit $e[h]$, $s[h]$ und $a[h]$, die zu h gehörigen Einträge. Mit $[h]_0$ und $[h]_1$ bezeichnen wir die Anzahl der 0*-er bzw.* 1*-er in $s[h]$.*
Diese Menge von Listen muss weiterhin folgenden Eigenschaften genügen:

(R1) Seien h und h' zwei Listeneinträge, welche dieselbe Kante repräsentieren, dann erhält man durch Umkehren von $s[h]$ und Vertauschen aller Einträge von 1 *zu* 0 *und umgekehrt, die Zeichenkette $s[h']$.*

(R2) Für alle $f \in F$ gilt folgende Gleichung, wobei f_1 die Außenfacette ist:

$$\sum_{h \in H(f)} \Big([h]_0 - [h]_1 + 2 - a[h]/90 \Big) = \begin{cases} -4 & \text{falls } f = f_1 \text{ gilt,} \\ +4 & \text{sonst.} \end{cases}$$

(R3) Der Gesamtwinkel an jedem Knoten des Graphen über alle anliegenden Facetten addiert beträgt stets $360°$.

In dieser Arbeit werden wir häufig den Begriff der SLOG-Repräsentation verwenden, welcher zu dieser Definition bis auf einige Details identisch ist, jedoch eine Beschreibung einer SLOG-Zeichnung mit Dummy-Knoten darstellen soll und nicht die einer orthogonalen Zeichnung.

2.1.9 Definition *Sei* $G = (R \cup C, E)$ *ein planarisierter zusammenhängender Graph mit einer Facettenmenge* F*, wobei* C *die Dummy-Knoten und* R *die echten Knoten sind.*
Eine ***schieforthogonale Repräsentation*** *oder* ***SLOG-Repräsentation*** *von* G *ist eine Menge von Listen H(f) für jede Facette* $f \in F$*, mit Einträgen der Form* (e, s, a)*, wobei* $e \in R \cup C$*,* s *eine evtl. leere binäre Zeichenkette und* $a \in \{90, 180, 270, 360\}$ *ist.*
Diese Listen stellen die Anordnung der Knoten im Uhrzeigersinn an der Facette dar. Die binäre Zeichenkette beschreibt, welche Knicke in welcher Reihenfolge die Kante, die auf den Knoten e *in* f *folgt, haben soll. Eine* 0 *enstspricht einem Innenwinkel von* $135°$ *und eine* 1 *einem Innenwinkel von* $225°$*. Die Zahl* a *gibt den Innenwinkel zur nächsten Kante der Facette an.*
Für $h \in H(f)$ *bezeichnen wir mit* $e[h]$*,* $s[h]$ *und* $a[h]$*, die zu* h *gehörigen Einträge. Mit* $[h]_0$ *und* $[h]_1$ *bezeichnen wir die Anzahl der* 0*-er bzw.* 1*-er in* $s[h]$*.*
Diese Menge von Listen muss weiterhin folgenden Eigenschaften genügen:

(R1) Seien h *und* h' *zwei Listeneinträge, welche dieselbe Kante repräsentieren, dann erhält man durch Umkehren von* $s[h]$ *und Vertauschen aller Einträge von* 1 *zu* 0 *und umgekehrt, die Zeichenkette* $s[h']$*.*

(R2) Sei h *ein Listeneintrag und die zu* h *gehörige Kante eine* rr*- oder* cc*-Kante, dann ist* $[h]_0 - [h]_1$ *eine gerade Zahl.*

(R3) Sei h *ein Listeneintrag und die zu* h *gehörige Kante eine* rc*-Kante, dann ist* $[h]_1 - [h]_0$ *eine ungerade Zahl.*

(R4) Für alle $f \in F$ *gilt folgende Gleichung, wobei* f_1 *die Außenfacette ist:*

$$\sum_{h \in H(f)} \Big(([h]_0 - [h]_1)/2 + 2 - a[h]/90 \Big) = \begin{cases} -4 & \text{falls } f = f_1 \text{ gilt,} \\ +4 & \text{sonst.} \end{cases}$$

(R5) Der Gesamtwinkel an jedem Knoten des Graphen über alle anliegenden Facetten addiert beträgt stets $360°$.

Hierbei ist zu beachten, dass eine Zeichnung einer solchen SLOG-Repräsentation eine SLOG-Zeichnung ist, welche zunächst 2.1.2(iii) nicht erfüllt, also sind Knoten eventuell inzident zu diagonalen Segmenten. Das gilt aber nur für die c-Knoten, welche Kreuzungen entsprechen. Deshalb sprechen wir bei solchen Zeichnungen auch von **PSLOG-Zeichnungen** (eng. planarized slanted orthogonal

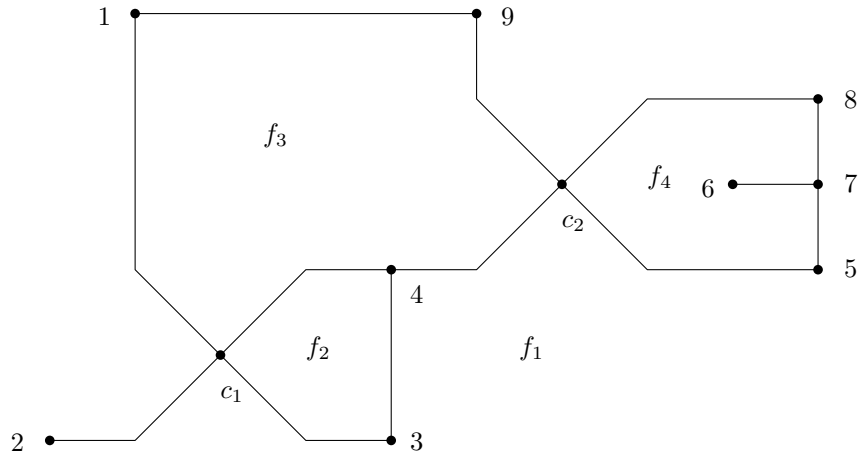


Abbildung 2.4: PSLOG-Zeichnung

drawing), da sie nach Entfernen der c-Knoten alle Eigenschaften der SLOG-Zeichnungen erfüllen.
Die vorhergehende Definition und die geforderten Bedingungen an die Listen machen wir uns am Besten an einem Beispiel klar.

2.1.10 Beispiel Sei $G = (R \cup C, E)$ mit $R = \{1, 2, \ldots, 9\}$ und $C = \{c_1, c_2\}$ gegeben. Die Kantenmenge E liegt durch die SLOG-Repräsentation fest, also geben wir sie hier nicht explizit an.
Zu der Facettenmenge $F = \{f_1, f_2, f_3, f_4\}$ geben wir nun deren SLOG-Repräsentation an:

$$\begin{aligned}
H(f_1) =&((1,1,90),(c_1,0,360),(2,1,90),(c_1,1,270),(3,\epsilon,90),(4,1,90),\\
&(c_2,1,270),(5,\epsilon,180),(7,\epsilon,270),(8,1,90),(c_2,0,270),(9,\epsilon,270)),\\
H(f_2) =&((c_1,0,90),(4,\epsilon,90),(3,0,90)),\\
H(f_3) =&((1,\epsilon,90),(9,1,90),(c_2,0,180),(4,1,90),(c_1,0,90)),\\
H(f_4) =&((c_2,0,90),(8,\epsilon,90),(7,\epsilon,360),(6,\epsilon,90),(7,\epsilon,90),(5,0,90)).
\end{aligned}$$

Dass dies eine SLOG-Repräsentation ist, wobei f_1 die Außenfacette ist, lässt sich leicht nachprüfen. Abb. 2.4 zeigt eine PSLOG-Zeichnung dieser Repräsentation:

2.2 Zusammenhänge zwischen SLOG und OG

Die vorhergehenenden Definitionen liefern uns einen Einblick in die zentralen Begriffe dieser Arbeit. In diesem Absatz werden wir Aussagen über die im letzten Kapitel definierten Begriffe formulieren, um ein besseres Gefühl für sie zu bekommen. Hierfür beweisen wir zunächst ein Lemma, welches die SLOG-Repräsentation und die PSLOG-Zeichnung in Verbindung bringt.

2.2.1 Lemma *Eine PSLOG-Zeichnung eines zusammenhängenden Graphen induziert stets eine SLOG-Repräsentation. D.h. durch Erzeugen der Listen $H(f)$,*

abhängig von der Zeichnung, erhalten wir stets eine SLOG-Repräsentation nach 2.1.9.

Beweis: Für den Beweis gehen wir die Eigenschaften aus Def. 2.1.9 durch und zeigen, dass diese für Listen, die durch die PSLOG-Zeichnung erzeugt wurden erfüllt sind.
Die Eigenschaft (R1) ist erfüllt, weil ein 135° Winkel an einem Knick stets als Gegenwinkel einen 225° Winkel hat und demnach die Listen genau richtig erzeugt werden. Die Eigenschaften (R2) und (R3) sind erfüllt, da ein Knick den Verlauf der Kante um 45° verändert und unsere Zeichnung an c-Knoten diagonale Eingänge bzw. Ports hat und an r-Knoten horizontale oder vertikale Ports. Für die Eigenschaft (R4) verwenden wir die allgemeine Formel für die Innenwinkelsumme in einem Polygon, welche bekanntlich $(n-2)\cdot 180°$ ist, wobei n die Anzahl der Ecken ist. Manuelles Ausrechnen auf der linken Seite durch Summieren der Winkel liefert folgende Gleichungskette.

$$\begin{aligned}
&\sum_{h\in H(f)} \left(135\,[h]_0 + 225\,[h]_1 + a\,[h]\,\right) = \Big(\sum_{h\in H(f)} \left([h]_0 + [h]_1 + 1\right) - 2\Big)180 \\
&\Leftrightarrow 2\cdot 180 = \sum_{h\in H(f)} \Big((180-135)\,[h]_0 + (180-225)\,[h]_1 + 180 - a\,[h]\,\Big) \\
&\Leftrightarrow 4 = \sum_{h\in H(f)} \Big(([h]_0 - [h]_1)/2 + 2 - a\,[h]\,/90\Big)
\end{aligned}$$

Dies entspricht genau der Eigenschaft (R4) für Innenfacetten. Da die Außenwinkelsumme bei einem Polygon um 720° größer ist, als die Innenwinkelsumme, erhält man auf diese Art und Weise -4 auf der linken Seite. Die Eigenschaft (R5) ist offensichtlich erfüllt. Daraus schließen wir, dass wir durch diese Listen eine SLOG-Repräsentation erhalten. □

Zu dem Zusammenhang zwischen den SLOG- und den OG-Repräsentationen können wir alle relevanten Resultate nachlesen [2]. Wir geben im Folgenden eine Zusammenfassung dieser Ergebnisse.

Es ist möglich, mit einem effizienten Algorithmus und gegebener Einbettung eine SLOG-Repräsentation zu berechnen, welche die Einbettung respektiert und zudem die Gesamtanzahl der Knicke auf den Kanten minimiert. So eine Repräsentation nennen wir im Weiteren auch **knickminimale** SLOG-Repräsentation.

Dieser Algorithmus ist zudem konstruktiv und basiert auf dem Flussalgorithmus von Tamassia zur Bestimmung von knickminimalen OG-Repräsentationen [8].

Zudem hat eine knickminimale SLOG-Repräsentation mindestens doppelt so viele Knicke wie die zur Einbettung gehörende knickminimale OG-Repräsentation. Nun haben wir gesehen, dass es gewisse Ähnlichkeiten zwischen diesen beiden Arten der Darstellung gibt und diese somit oft mit ähnlichen Methoden behandelt werden können.

Kapitel 3

Schnitte in schieforthogonalen Zeichnungen

In diesem Abschnitt befassen wir uns mit einer Methode zur Knickreduzierung in schieforthogonalen Zeichnungen. Diese Methode basiert größtenteils auf der Idee des „Moving“ aus einem Artikel von Fößmeier et al [5].

Beim Moving wird der Graph mit Hilfe eines Schnittes durch den Graphen kontrahiert, um die Fläche zu reduzieren. Wir werden etwas ähnliches verwenden, um Knicke aus einer PSLOG-Zeichnungen zu entfernen.

3.1 Definition von Schnitten

Die im Folgenden beschriebene Methode wurde bereits von Bekos et al. [2] mit der selben Zielsetzung wie hier verwendet. Hierfür definieren wir zunächst einen Schnitt in einer PSLOG-Zeichnung.

3.1.1 Definition *Sei S eine PSLOG-Zeichnung.*
*Ein **Schnitt** ist eine Kurve durch den Graphen, die folgende Eigenschaften erfüllt:*

(i) Die Kurve ist x-monoton, startet links von der Zeichnung und endet rechts von der Zeichnung.

(ii) Die Kurve schneidet den Graphen mit genau einer Ausnahme in nur vertikalen Kantensegmenten. Die Ausnahme ist ein Schnitt durch ein diagonales oder horizontales Segment e.

*Wir werden dann auch von einem **Schnitt durch e** oder einem **x-monotonen Schnitt** sprechen. Eine Kurve welche die gegensätzlichen Eigenschaften erfüllt, d.h. wir ersetzen x-monoton durch y-monoton, horizontal durch vertikal und umgekehrt und der Schnitt startet oberhalb der Zeichnung und endet unterhalb der Zeichnung, wird auch Schnitt genannt.*

Solche Schnitte können uns eine Möglichkeit liefern, zwei Knicke aus der Zeichnung zu entfernen, indem wir den Teil des Graphen unter bzw. links vom Schnitt

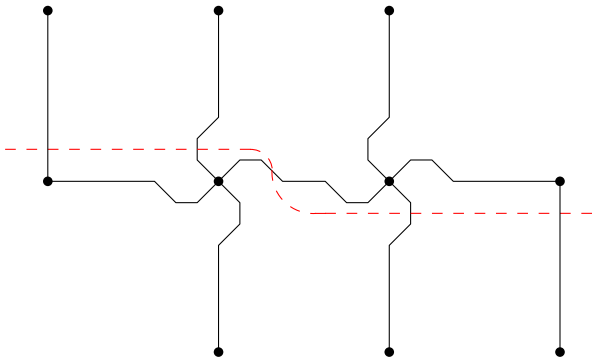


Abbildung 3.1: Beispiel eines Schnittes

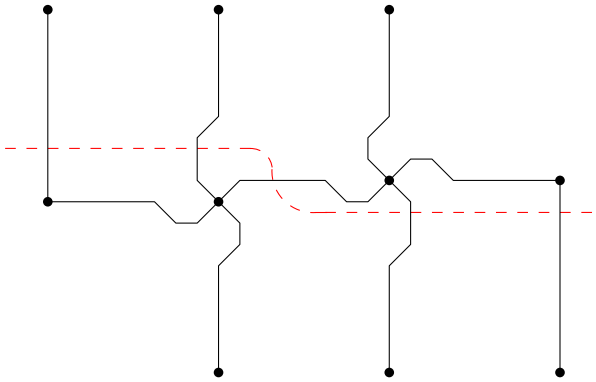


Abbildung 3.2: Verschiebung eines Teils der Zeichnung

nach unten bzw. links verlängern. Analog können wir auch den anderen Teil in die andere Richtung verlängern, was bis auf Verschiebung die gleiche Zeichnung erzeugt.
Hierbei spielen Kantensegmente mit zwei verschiedenen Arten von anliegenden Knicken eine wichtige Rolle. Wir bezeichnen diese Knickkombination, unabhängig davon ob ein Rechts- auf einen Linksknick folgt oder umgekehrt, als **RL-Knick**, da rechts und links abhängig von der Facette ist aus der wir darauf schauen.

Die Anwendung eines solchen Schnittes an einem RL-Knick zeigen wir in einem Beispiel.

3.1.2 Beispiel Wir betrachten den Schnitt, welcher in Abb. 3.1 zu sehen ist:
Um mit Hilfe eines Schnittes Knicke zu reduzieren, werden wir Schnitte wählen, bei denen das Kantensegment e, welches nach 3.1.1(ii) die Ausnahme ist, zwei gleich verlaufende Nachbarsegmente hat. Die Idee ist dabei den Schnitt so zu legen, dass e mit seinen Nachbarsegmenten zu einem Segment nach der Verschiebung verschmilzt.
Das Ergebnis der Verschiebung des unteren Teils der Zeichnung wird in Abb. 3.2 gezeigt: Diese Zeichnung hat nun zwei Knicke weniger als die Ursprüngliche, jedoch hat sie zwangsweise einen größeren Flächenbedarf.
Analog ist es möglich vertikale oder horizontale Kantensegmente mit der richtigen Wahl des Schnittes aus der Zeichnung zu entfernen um die Knickanzahl zu

reduzieren.

Wir werden im folgenden Kapitel nur mit Schnitten arbeiten, die uns ermöglichen Knicke in der Zeichnung zu entfernen. Diese Schnitte müssen, wie im vorherigen Beispiel verdeutlicht wurde, gewisse Zusatzeigenschaften erfüllen. Auf diese gehen wir in der folgenden Definition näher ein.

3.1.3 Definition *Ein Schnitt durch ein Kantensegment e ist* ***dehnbar****, wenn er folgende Eigenschaften erfüllt:*

(i) e liegt innerhalb eines RL-Knicks.

(ii) Der Schnitt verläuft so, dass er von den Knicken aus (i) jeweils den 225° Winkel sieht. (d.h. das Lot des Schnittes auf den Knick ist auf der Seite des 225° Winkels).

(iii) Der Schnitt ist x-monoton, wenn e oder ein Nachbarsegment von e horizontal verläuft, ansonsten y-monoton.

Ein dehnbarer Schnitt erlaubt es das Kantensegment e durch Verlängern der anderen Segmente, die durch den Schnitt verlaufen, zu entfernen.

3.2 Schneidbare Zeichnungen

Damit wir in einem Graphen garantieren können, dass wir beliebige Kantensegmente in RL-Knicken durch dehnbare Schnitte entfernen können, definieren wir folgende Eigenschaft von PSLOG-Zeichnungen:

3.2.1 Definition *Eine PSLOG-Zeichnung ist* ***schneidbar****, wenn es durch jedes Kantensegment in einem RL-Knick einen dehnbaren Schnitt gibt.*

Zunächst merken wir an, dass alle PSLOG-Zeichnungen ohne c-Knoten schneidbar sind, wie uns später klar wird.
Sobald man jedoch anfängt, die Anzahl der c-Knoten und die Anzahl der dazugehörigen Diagonalen zu erhöhen, wird es schwerer, diese Eigenschaft aufrecht zu erhalten, da die dehnbaren Schnitte eines Segments e nicht durch diagonale Segmente, abgesehen von e, verlaufen können.

Nun betrachten wir eine Klasse von PSLOG-Zeichnungen, welche stets schneidbar sind, wie wir danach beweisen werden.

3.2.2 Definition *Eine PSLOG-Zeichnung heißt* ***Windmühlenzeichnung****, wenn folgende Eigenschaften erfüllt sind:*

(i) An jedem c-Knoten ist in allen Richtungen der gleiche Knick mit gleichem Abstand zum Knoten anliegend.

(ii) In dem Quadrat, das durch Verbinden dieser 4 Knicke entsteht sind nur diese 4 Kantensegmente und der dazugehörige c-Knoten gezeichnet.

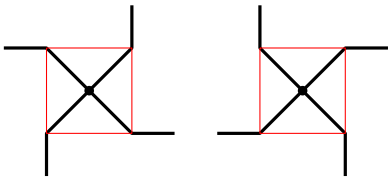


Abbildung 3.3: c-Knoten einer Windmühlenzeichnung

MA/Bilder/Linie.eps MA/Bilder/Linie.eps

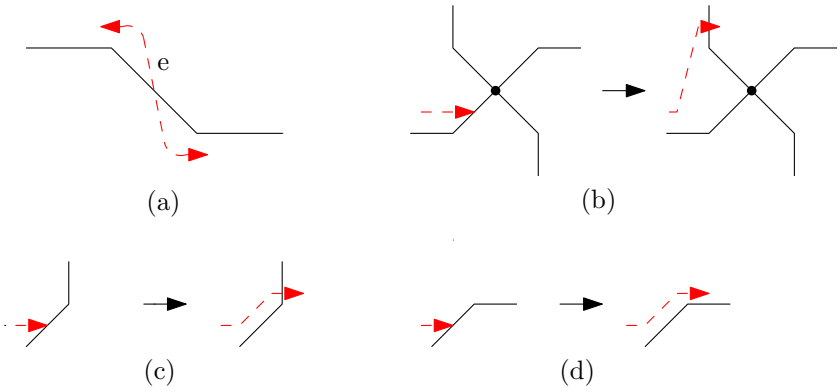


Abbildung 3.4: (a) Kurve (b),(c) Facette verlassen (d) Linie verlängern

Der Name entsteht dadurch, dass die Umgebung der c-Knoten, wie eine Windmühle ausschaut (siehe Abb. 3.3). Das rote Quadrat zeigt den Bereich aus (ii) an.

Hier ist anzumerken, dass PSLOG-Zeichnungen ohne c-Knoten insbesondere auch Windmühlenzeichnungen sind und demnach auch folgenden Satz erfüllen.

3.2.3 Satz *Windmühlenzeichnungen sind schneidbar.*

Beweis: Zunächst zeigen wir die Aussage für alle Kantensegmente in einem RL-Knick, die horizontal verlaufen oder deren Nachbarn horizontal sind. Die Aussage für die Übrigen folgt durch Drehen der Zeichnung um 90°, da wir dann wieder im oberen Fall sind. Zurückdrehen liefert dann den y-monotonen Schnitt.
Sei e ein solches Kantensegment, welches einen x-monotonen dehnbaren Schnitt benötigt.
Wir starten mit einer Linie, welche die Eigenschaften (ii) und (iii) aus 3.1.3 erfüllt. Die Eigenschaft (i) ist nämlich nach Definition von e erfüllt. Diese Linie verläuft bisher nur durch das Kantensegment e (siehe Abb. 3.4(a)).

Was dann zu zeigen bleibt ist, dass ein beliebiges Ende unserer bisherigen Kurve durch die Zeichnung verlängerbar ist und durch vertikale Segmente die Zeichnung

verlassen kann.
Hierfür betrachten wir das rechte Ende der Kurve, da die Argumentation für das andere Ende symmetrisch ist. Wir gehen hierfür in Schritten vor: Wir verlängern diese Kurve nach rechts bis wir auf die Zeichnung treffen. Nun gibt es genau drei mögliche Teile der Zeichnung, die wir treffen können.

(1) vertikales Segment: Durch dieses können wir hindurch und unsere Facette verlassen.

(2) Knoten oder Knickstelle: Dies lässt sich vermeiden wenn wir vor dem Kontakt die Kurve minimal verschieben, da der Knoten oder die Knickstelle nur ein Punkt ist.

(3) diagonales Segment: Hier müssen wir nun betrachten ob es ein diagonales Segment ist, das in x-Richtung an einem c-Knoten anliegt.
Falls ja, können wir aus dem Quadrat der Windmühle zurückgehen (siehe Def. 3.2.2(ii)), da unser Segment e nicht im Quadrat liegen darf ist das möglich. Daraufhin können wir die Facette durch das linke vertikale Segment der Windühle verlassen (siehe Abb. 3.4(b)).
In dem anderen Fall macht die Kante an dieser Stelle einen Knick,falls sie vertikal wird verlassen wir die Facette durch diesen vertikalen Teil (siehe Abb. 3.4(c)). Ansonsten wird sie horizontal und wir weichen der Diagonale aus. Daraufhin verlängern wir nach rechts bis wir wieder die Zeichnung treffen und befinden uns wieder in der Ausgangssituation von oben (siehe Abb. 3.4(d)).

Nachdem wir eine Facette auf diese Weise verlassen haben, starten wir wieder beim Verlängern, solange bis der Schnitt außerhalb der Zeichnung ist. Wegen der Endlichkeit des zugrundeliegenden Graphen und somit auch der Zeichnung bricht dieser Prozess ab. Nachdem wir diese Schritte für beide Enden durchgeführt haben, erhalten wir unseren dehnbaren Schnitt von e, was zu zeigen war. □

Dieser Satz spielt eine wichtige Rolle im folgenden Kapitel, da wir darin Knicke in bereits vorhandenen Zeichnungen mit dehnbaren Schnitten reduzieren wollen.

3.3 Dehnbare Schnitte in PSLOG-Zeichnungen

Wir haben nun im vorherigen Absatz gezeigt, dass es Graphenklassen gibt, bei denen es durch Segmente mit RL-Knicken stets dehnbare Schnitte gibt.
Allgemein ist es jedoch nicht trivial festzustellen, ob es zu einem bestimmtem Kantensegment e in einem RL-Knick einen dehnbaren Schnitt gibt.

Im Folgenden formulieren wir einen Algorithmus, der feststellt ob es einen dehnbaren Schnitt durch e gibt und dieses Segment somit entfernt werden könnte.

Hierfür stellen wir zunächst fest, dass es ausreicht x-monotone Schnitte zu betrachten, da y-monotone Schnitte mit symmetrischen Argumenten behandelt werden können. Zudem genügt es, wenn wir das eine Ende des Schnittes betrachten und mit unserem Algorithmus lediglich zeigen, wie man dieses durch vertikale

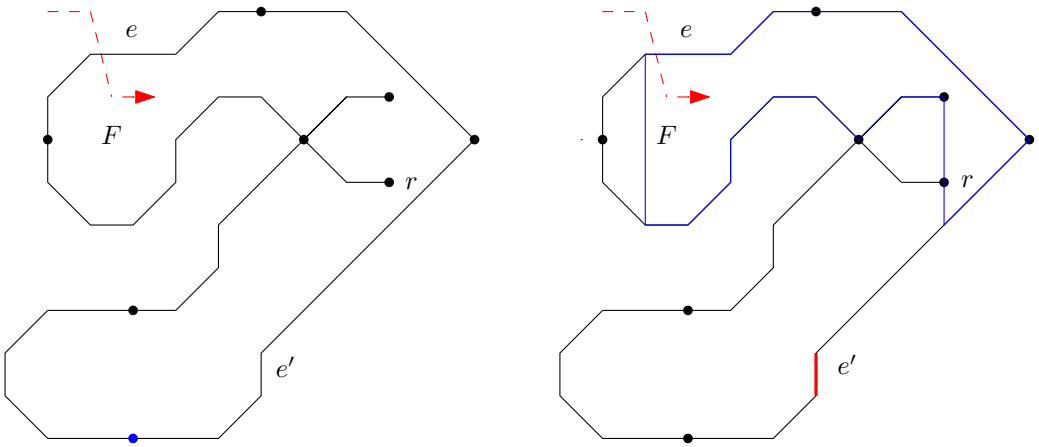


Abbildung 3.5: (a) Ausschnitt einer PSLOG-Zeichnung (b) Markierter Bereich für vertikales Segment

Segmente, wenn möglich, neben die Zeichnung führt. Das andere Ende lässt sich wiederum mit symmetrischen Argumenten neben die Zeichnung führen. Um unser Problem anschaulich zu machen zeigen wir zunächst, welchen Bereich der Facette der Algorithmus nach einem vertikalen Segment durchsuchen muss. Abb. 3.5(a) zeigt einen Auschnitt einer PSLOG-Zeichnung, welcher das Segment e und den Anfang eines x-monotonen Schnittes durch sie zeigt. In Abb. 3.5(b) am blauen Polygon sehen wir den Bereich, der nach dem vertikalen Segment durchsucht wird. Dieser muss überall eine echt größere x-Koordinate als das linke Ende von e haben. Außerdem müssen wir darauf achten, dass der Schnitt x-monoton verläuft und wir somit z.B. nicht um den Knoten r laufen können, um durch e' die Facette F zu verlassen.

Der Übersicht wegen legen wir fest, dass unsere Zeichnung in einer Form vorliegt, in der jeder Knoten auf einem ganzzahligen Gitterpunkt liegt und der Verlauf der Kanten dem Programm durch Koordinaten der Knicke vorliegt.

Nun beschreiben wir unseren Algorithmus.

Algorithmus: Dehnbarkeitstest
Eingabe: Kantensegment e in einem RL-Knick in einer PSLOG-Zeichnung.
Ausgabe: Dehnbarer Schnitt durch e, falls einer existiert.

(1) Bestimmen des Polygons, welches mit einem Schnitt durch e erreicht werden kann. Speichere danach alle Segmente, durch die der Schnitt verlaufen kann, in einer Liste L.

(2) Rekursive Aufrufe von Dehnbarkeitstest für alle Segmente in L, wobei das Ziel ist den Schnitt aus der Zeichnung zu führen.

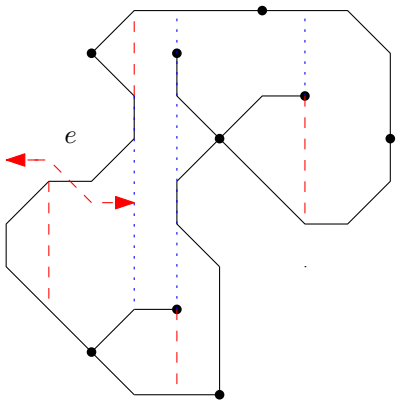
(3) Rufe Dehnbarkeitstest für die von e aus andere Richtung auf, falls die ersten zwei Schritte erfolgreich waren.

Falls hier der Schnitt auch nach außen geführt werden kann, erhalten wir durch Zusammenführen beider Richtungen den gesuchten Schnitt.

Beschreibung des Algorithmus:

(1): Zunächst sei F die zu e gehörige Facette, in der unser Schnitt in positive x-Richtung verlaufen soll. Zudem beschreiben wir die Lage des linken Endes von e (dies ist wohldefiniert, da der Schnitt x-monoton und e demnach horizontal oder diagonal ist) mit den Koordinaten $L = (x', y')$.
Hierfür bestimmen wir zunächst das blaue Polygon, welches man z.B. in Abb. 3.5(b) sehen kann. Dafür legen wir eine Kopie der Facette F an und konstruieren daraus unser gewünschtes Polygon. Daraufhin fällen wir das Lot von L innerhalb von F. Also suchen wir ein Randsegment von F, welches die Gerade $x = x'$ mit der größten y-Koordinate, jedoch mit $y < y'$ schneidet (siehe Abb. 3.6). Vertikale Segmente mit dieser x-Koordinate werden hierbei übergangen. Diesen Schnittpunkt nennen wir S.
Nachdem dieser Schnittpunkt gefunden ist, entfernen wir den Teil des Graphen zwischen L und dem gefundenen Schnittpunkt aus F. Stattdessen fügen wir eine senkrechte Gerade von L nach S ein.
Danach durchlaufen wir weiter F und suchen eben solche Segmente, die von x-monoton zu nicht mehr x-monoton wechseln, wobei rechts davon das Innere der Facette ist. Dann fällen wir von diesem Wendepunkt ein Lot nach unten wie zuvor. Danach fällen wir ein Lot nach oben. Hierfür suchen wir einen Randpunkt von F, mit der kleinsten y-Koordinate von allen Randbereichen der Facette, die über unserem Wendepunkt liegen (siehe Abb. 3.6).
Wenn bei dem Lotfällprozess kein Lotfußpunkt gefunden wird, heißt es, dass wir uns auf der Außenfacette befinden und man zudem entlang des Lotes rechts aus der Zeichnung herauskommt. In diesem Fall ist eine Richtung des Schnittes gefunden und wir können bei (3) weitermachen.
Als nächstes betrachten wir die Teile der Facette, welche links zwischen den jeweiligen Lotendpunkten liegen, und ersetzen jedes dieser Teile durch das Lot, falls das Kantensegment e nicht darin liegt. Dies machen wir, da man in diesem Fall diese Segmente von F nicht mit dem Cut durch e erreichen kann.
Nachdem wir dies für alle Wendepunkte in F getan haben, erhalten wir das Polygon, welches von e mit Schnitten erreicht werden kann.
Als letztes erstellen wir für F eine Liste L_F, in die wir alle vertikalen Segmente des Polygons eintragen von denen aus sich links F befindet. Diese Segmente können wir nämlich mit unserem Cut von e aus erreichen.

(2): Falls L_F nicht leer ist, entfernen wir ein Segment daraus und rufen Dehnbarkeitstest indem wir das e oben durch dieses Segment erstezen. Da dies jedoch ein vertikales Segment ist, ist das linke Ende bei (1) das untere Ende und es kann passieren, dass im ersten Schritt kein Lot nach unten gefällt wird. Dies erfüllt dann jedoch nicht die Endbedingung in (1).
In dem Fall, dass L_F leer ist, gehen wir wenn möglich eine Facette zurück durch ein vertikales Segment und machen dann wieder bei (2) weiter. Falls wir damit



MA/Bilder/Lotbsp.eps MA/Bilder/Lotbsp.eps

Abbildung 3.6: Lote fällen: Rote Segmente ersetzen die Teile des Graphen die links davon sind

bereits zu der Facette vom ursprünglichen e gelangen und dort das L_F leer ist, gibt es keinen dehnbaren Schnitt durch e.

(3): Wir rufen nun sozusagen den Algorithmus auf, der nun spiegelverkehrt Schritt 1 durchführt, da wir nun den Teil des Cuts in die negative x-Richtung von e suchen. Wenn der Algorithmus irgendwann bei (1) terminiert, indem man keinen Lotfußpunkt findet, dann erhalten wir einen Schnitt, der durch die Kantensegmente definiert ist, die unser Algorithmus beidseitig durchlaufen hat.

Der formale Beweis dieses Algorithmus wird hier weggelassen, weil dieser vom Aufbau sehr ähnlich zu der bereits beschriebenen Beschreibung ist.

Kapitel 4

Heuristik für fast optimale Zeichnungen

4.1 Vorarbeit

Da wir im Allgemeinen noch nicht wissen, ob jede SLOG-Repräsentation zeichenbar ist, wollen wir versuchen, eine PSLOG-Zeichnung zu finden, die der SLOG-Repräsentation so nah wie möglich ist. Dies haben bereits Bekos et al. [2] beschrieben und sind zu folgendem Ergebnis gekommen:

4.1.1 Satz *Sei eine SLOG-Repräsentation eines planarisierten Graphen G mit Maximalgrad 4 gegeben, dann können wir effizient eine schieforthogonale Zeichnung mit $O(n^2)$ Flächenbedarf berechnen mit*

(i) optimaler Anzahl von Halbknicken an rr-Kanten und rc-Kanten ohne Knicke und

(ii) höchstens zwei zusätzliche Halbknicke an cc-Kanten und rc-Kanten mit Knicken.

Bei diesem Satz ist zu beachten, dass bei der dort beschriebenen Zeichnung lediglich die Einbettung der ursprünglichen SLOG-Repräsentation respektiert wird und nicht direkt die Repräsentation selbst. Es können sich gewisse Winkel zwischen r-Knoten verändern, was zu einer anderen Zeichnung führt. Dies liegt daran, dass man bei dieser Heuristik von der zugrundeliegenden knickminimalen orthogonalen Repräsentation ausgeht, welche Unterschiede in Winkeln zwischen Knoten aufweisen kann.
Ein großer Vorteil der dort verwendeten Konstruktion ist der quadratische Flächenbedarf, da dieser bei knickminimalen Zeichnungen leider exponentiell sein kann, wie auch von Bekos et al. [2] bewiesen wurde.

4.2 Hilfslemmata

Für die später folgende Heuristik benötigen wir einige Hilfsüberlegungen in Form von Lemmata. Eines davon befasst sich mit der Lagenzuweisung, was der Festle-

gung der Ausrichtung von Kantensegmenten in der Zeichenebene entspricht.

4.2.1 Lemma *Die Festlegung der Lage eines einem Knoten angrenzenden Kantensegmentes einer SLOG-Repräsentation in einer Zeichnung (z.B. von links nach rechts oder von rechts oben nach links unten) legt die Lage aller Kantensegmente des Graphen fest.*
Hierbei ist zu beachten, ob das Kantensegment diagonal ist oder senk- bzw. waagrecht, da wir an Dummy-Knoten diagonale Ports und senk- bzw. waagrecht an echten Knoten haben.

Beweis: Für den Beweis wählen wir ein beliebiges Kantensegment s und zeigen, dass dessen Lage durch die Wahl der Lage eines beliebigen anderen Segments s' festgelegt ist.
Da unser zugrunde liegende Graph zusammenhängend ist, existiert ein Pfad von s zu s', welcher durch Kantensegmente und eventuell Knoten verläuft. Die Lagen der Kantensegmente der Kante von s werden bereits durch einen Listeneintrag einer Nachbarfacette festgelegt, da die Binärstrings die Art der Knicke und somit auch die Lagen festlegen. □

Das nächste Lemma zeigt eine Eigenschaft von knickminimalen SLOG-Repräsentationen.

4.2.2 Lemma *Eine knickminimale SLOG-Repräsentation eines Graphen enthält nur monotone Binärstrings, d.h. in diesen Strings kommen jeweils ausschließlich 1-er oder 0-er vor.*

Beweis: Angenommen, unsere knickminimale SLOG-Repräsentation enthält bei einem Kantenzug sowohl 1er als auch 0er. Dann gibt es eine 01- oder 10-Folge im Binärstring, die wir nun mit ihrem Gegenstück aus den Binärstrings entfernen. Wir zeigen, dass nach Def. 2.1.9 immernoch eine SLOG-Repräsentation vorliegt.
Die Eigenschaft (R1) ist erfüllt, da wir das Gegenstück auch entfernt haben.
(R2), (R3) und (R4) sind noch erfüllt, da eine Entfernung einer 0 und einer 1 nichts an dem Term $[h]_1 - [h]_0$ verändert. Der Term $a\,[h]$ bleibt dadurch auch unverändert.
An der Eigenschaft (R5) wurde nichts geändert.
Nun haben wir zwei Knicke aus der SLOG-Repräsentation entfernt und es liegt immernoch eine SLOG-Repräsentationen vor. Dies ist ein Widerspruch zur Knickminimalität. □

4.2.3 Bemerkung Die Rückrichtung von Lemma 4.2.2 ist nicht richtig, wie man bereits an einem Graphen mit zwei r-Knoten, einer Kante und einem Knick sieht. Deswegen sprechen wir später von einer größeren Menge von SLOG-Repräsentationen, wenn sie lediglich diese Eigenschaft haben.

4.3 Rotationsbasierte Heuristik

In diesem Teil geben wir unsere Heuristik in Form eines Algorithmus an. Hierbei erhalten wir als Eingabe SLOG-Repräsentationen mit nur monotonen Binärstrings.

Dies ist keine große Einschränkung, da Bekos et al. [2] gezeigt haben, dass man zu beliebigen Einbettungen effizient knickminimale schieforthogonale Repräsentationen berechnen kann. Nach Lemma 4.2.2 haben diese Repräsentationen monotone Binärstrings.
Unser Ziel wird es hierbei sein eine Zeichnung zu generieren, die auf jeden Fall die Einbettung respektiert und weitestgehend die ursprüngliche SLOG-Repräsentation erhält, d.h. die Knicke und Winkel werden soweit möglich aus der SLOG-Repräsentation übernommen.
Zudem entspricht die Lage der aus den Knoten ausgehenden Kantensegmente genau denen, die wir durch Festlegung einer Kante nach Lemma 4.2.1 erhalten.
Nun beschreiben wir den Ablauf unserer Heuristik:

Algorithmus: Rotationsbasierter Algorithmus
Eingabe: Schieforthogonale Repräsentation S mit monotonen Binärstrings von einem Graphen G.
Ausgabe: Eine schieforthogonale Zeichnung von G.

(1) Lege die Lage eines Kantensegments fest und speichere diese.

(2) Erzeuge durch Rotation der c-Knoten eine orthogonale Repräsentation basierend auf S und ersetze die Halbknicke durch Knicke.

(3) Zeichne die orthogonale Repräsentation mit einem Flussnetzwerk.

(4) Rotiere die c-Knoten in der orthogonalen Zeichnung wiederum in eine Richtung, sodass sie stimmig mit der Lagenzuordnung aus (1) sind und man eine PSLOG-Zeichnung erhält.

(5) Entferne Knicke mit dehnbaren Schnitten, welche garantieren, dass die Zeichnung schneidbar bleibt.

(6) Entferne Knicke mit dehnbaren Schnitten, die eventuell einen Teil der Schneidbarkeit der Zeichnung zerstören.

Nachdem wir eine kurze Beschreibung des Algorithmus geliefert haben, betrachten wir nun jeden Schritt nochmal in detaillierter Form.

(1): Festlegen einer Lage liefert uns nach Lemma 4.2.1 eine Ausrichtung jedes Kantensegments der Repräsentation. Insbesondere werden wir die Ports an den r-Knoten später in (3) wieder herstellen.

(2): Um aus S eine gültige orthogonale Repräsentation zu erhalten rotieren wir die c-Knoten, damit diese orthogonale Ports bekommen.

Hierfür analysieren wir zunächst welche Rotation geschickter ist, um unserer Zeichnung weniger Knicke hinzuzufügen. Dafür betrachten wir die Art der Binärstrings, die mit einem c-Knoten beginnen, und gefolgt werden von einem r-Knoten.

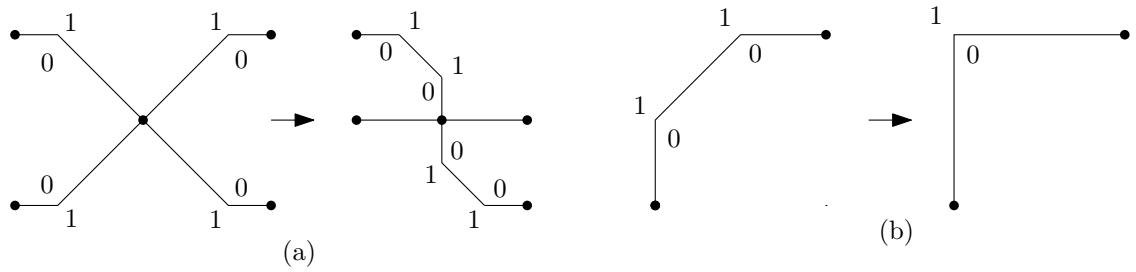


Abbildung 4.1: (a) Drehung des c-Knotens (b) Halbieren der Einträge

Hierbei zählen wir, ob es mehr Binärstrings unter diesen Kanten gibt, die aus Einsen oder Nullen bestehen. Leere Strings sind nicht möglich, da diese Einträge *rc*-Knoten repräsentieren. Falls es mehr oder gleich viele Einträge mit Nullen gibt rotieren wir im Uhrzeigersinn, ansonsten umgekehrt.

Im Uhrzeigersinn bedeutet, dass wir folgende Schritte nacheinander für jeden c-Knoten c_i des Graphen durchführen: In jeder Facette an der c_i liegt, suchen wir den Listeneintrag, der mit diesem Knoten beginnt. Falls dieser Eintrag als erstes Zeichen des Binärstrings eine 0 hat, entfernen wir diese, ansonsten fügen wir eine 1 als erstes Zeichen hinzu (siehe Abb. 4.1(a)). Im Falle der Rotation gegen den Uhrzeigersinn vertauschen wir bei diesem Prozess die Rolle von 0 und 1. Danach ändern wir diese Kante von der anderen Seite analog ab, sodass die Bedingung (R1) in Def. 2.1.9 weiterhin erhalten bleibt. Nachdem wir dies für alle Knoten gemacht haben, erhalten wir zwischen allen Knoten des Graphen Binärstrings mit einer geraden Anzahl an Einsen oder Nullen.

Dies liegt daran, dass wir an rc-Kanten genau eine Änderung vorgenommen haben und diese somit gerade Stringlängen haben. An cc-Kanten haben wir zwei Änderungen vorgenommen, die sich gegenseitig aufheben, sodass sich an diesen nichts geändert hat. Das können wir damit begründen, dass die Kante zwischen den c-Knoten einen monotonen Binärstring hat.
In den Strings wurden nach Konstruktion bisher lediglich Zeichen eingefügt, die nicht unterschiedlich zu den bereits vorhandenen waren.
Als nächstes entfernen wir nun in jedem Binärstring aller Listen die Hälfte der Zeichen und erhalten als Ergebnis eine orthogonale Repräsentation, was wir im Folgenden beweisen (siehe Abb. 4.1(b)).

Beweis: Wir zeigen zunächst, dass uns der erste Schritt der Rotation der c-Knoten eine schieforthogonale Repräsentation liefert, wobei die Knotenmenge danach lediglich Knoten mit orthogonalen Ports enthält. Die c-Knoten hatten zu Beginn

diagonale Ports, also werden diese Ports durch das Einfügen oder Entfernen eines Halbknicks zu einem orthogonalen Port.
Nun zeigen wir, dass das Halbieren der Halbknicke durch Halbieren der Strings eine orthogonale Repräsentation liefert. Zuvor galt die Gleichung:

$$\sum_{h \in H(f)} \left(([h]_0 - [h]_1)/2 + 2 - a\,[h]\,/90 \right) = \begin{cases} -4 & \text{falls } f = f_1 \text{ gilt,} \\ +4 & \text{sonst.} \end{cases}$$

Nachdem danach $[h]_0$ und $[h]_1$ halbiert wurden erhalten wir genau die Bedingung aus Def. 2.1.8 (R2) :

$$\sum_{h \in H(f)} \left([h]_0 - [h]_1 + 2 - a\,[h]\,/90 \right) = \begin{cases} -4 & \text{falls } f = f_1 \text{ gilt,} \\ +4 & \text{sonst.} \end{cases}$$

Also erhalten wir nach dieser Änderung eine orthogonale Repräsentation.

□

(3): Zum Zeichnen der orthogonalen Repräsentation verwenden wir das Flussnetzwerk von Tamassia [8]. Dieser Algorithmus wird darin als $FIND\_LENGTHS$ bezeichnet und liefert eine Zeichnung, bei der sowohl die Knicke, als auch die Knoten auf ganzzahligen Gitterpunkten liegen. Wir müssen lediglich anpassen, dass die Portzuweisungen an den früheren r-Knoten mit denen aus Schritt (1) übereinstimmen. Hierfür müssen wir die Zeichnung nur noch rotieren, sodass die Ports an einem Knoten übereinstimmen. Dies liegt daran, dass Schritt (2) nichts an der Lage der r-Knoten zueinander verändert hat.

(4): Damit wir die folgenden Operationen besser durchführen können, verachtfachen wir zunächst die Größe unserer Zeichnung, sodass alle Knoten und Knicke auf Gitterpunkten liegen, welche durch 8 teilbar sind. Damit wir im Weiteren eine PSLOG-Zeichnung erhalten, welche möglichst genau die SLOG-Repräsentation respektiert, ersetzen wir jeden Knick durch zwei Halbknicke, wobei das mittlere Stück zwischen den Halbknicken die Länge $\sqrt{2}$ hat (das entspricht der Länge der Diagonale eines Quadrats mit Seitenlänge 1). Danach rotieren wir in der Zeichnung nun die früheren c-Knoten (die ja orthogonale Ports in der orthogonalen Zeichnung haben) entgegen der Richtung aus (2), um die gewünschten Lagen der c-Knoten zu erhalten. Hierbei fügen wir sogenannte Löffel ein, welche von Bekos et al. [2] verwendet wurden (siehe Abb. 4.2).

Dabei ist zu beachten, dass die Knoten oben und links jeweils acht Einheiten von dem c-Knoten entfernt sind, demnach die Löffel stets eine Höhe von genau 1 haben. Dies wird für die Schnitte in den weiteren Schritten relevant.

Nach diesem Schritt entsprechen die Ports an allen Knoten genau denen aus Schritt (1) und sie werden von nun an nicht mehr verändert.

(5): In diesem Schritt entfernen wir möglichst viele Knicke, die wir durch die Löffel eingefügt haben. Dies erfolgt mit dehnbaren Schnitten entsprechend Def. 3.1.1.

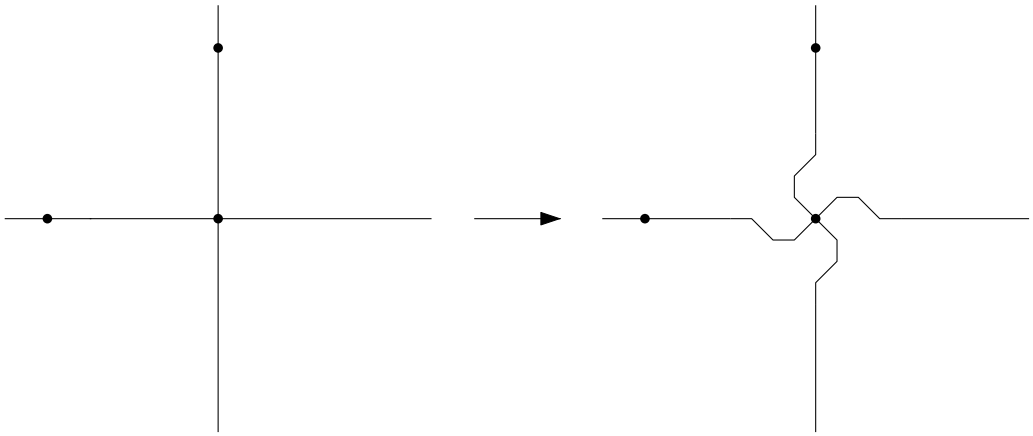


Abbildung 4.2: Löffel einfügen

Hierfür verwenden wir Satz 3.2.3, um dehnbare Schnitte zu garantieren, da unsere Zeichnung nach Einfügen der Löffel an jedem Knoten zu einer Windmühlenzeichnung wurde.

Nun können wir wie in Bsp. 3.1.2 nacheinander die zwei hinteren Knicke an jedem Löffel entfernen, da unsere Zeichnung nach einer solchen Entfernung weiterhin eine Windmühlenzeichnung ist. Hierzu legen wir einen dehnbaren Schnitt durch das diagonale Kantensegment des Löffels, das nicht am c-Knoten liegt.

(6): Im letzten Schritt betrachten wir zunächst Kantensegmente in unserer Zeichnung, welche mit Hilfe eines dehnbaren Schnitts entfernt werden könnten. Das sind in unserem Falle noch alle Segmente in RL-Knicken, da wir immernoch eine Windmühlenzeichnung vorliegen haben. Je nachdem ob die Meisten dieser Segmente einen x-monotonen Schnitt benötigen oder einen y-monotonen entfernen wir die Mehrheit von diesen durch dehnbare Schnitte.

Dies können wir tun, da ein x-monotoner Schnitt keine vertikalen Segmente entfernt, sondern nur horizontale oder diagonale durch welche der folgende x-monotone Schnitt nicht verlaufen darf.

Insbesondere lässt sich die Argumentation für die Durchführbarkeit eines x-monotonen Schnittes, wie im Beweis von Satz 3.2.3 immer noch genauso durchführen. Analog verläuft die Argumentation wiederum für die y-monotonen Schnitte, falls diese der Mehrheit entsprechen.

4.4 Heuristik an einem Beispiel

Nachdem wir nun unsere Heuristik in Worten beschrieben haben, betrachten wir in diesem Absatz den Ablauf der einzelnen Zwischenschritte an einem speziellen Beispiel, das möglichst viele Szenarien darstellen soll.

Hierfür betrachten wir den Graphen G mit der Knotenmenge

$$R \cup C = \{1, 2, \ldots, 7\} \cup \{c_1, c_2, c_3\}$$

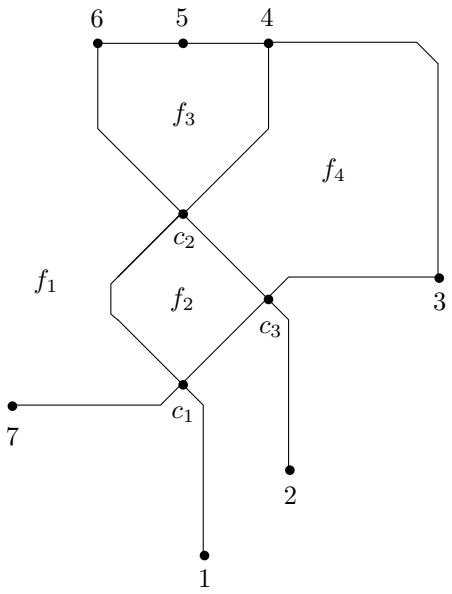


Abbildung 4.3: PSLOG-Zeichnung, die S respektiert

und der Facettenmenge $\{f_1, f_2, f_3, f_4\}$.
Hierzu geben wie eine SLOG-Repräsentation S an, wobei ϵ den leeren String repräsentiert und bedeutet, dass keine Knicke auf dieser Kante sind.

$$
\begin{aligned}
H(f_1) = ((1,1,90),(c_1,\epsilon,90),(c_3,0,360),(2,1,90),(c_3,0,270),(3,11,180),\\
(4,\epsilon,180),(5,\epsilon,270),(6,1,90),(c_2,11,90),(c_1,0,360),(7,1,90),(c_1,0,360)),\\
H(f_2) = ((c_1,00,90),(c_2,\epsilon,90),(c_3,\epsilon,90)),\\
H(f_3) = ((c_2,0,90),(6,\epsilon,180),(5,\epsilon,90),(4,0,90)),\\
H(f_4) = ((c_2,1,90),(4,00,90),(3,1,90),(c_3,\epsilon,90)).
\end{aligned}
$$

Nachdem dies nicht sehr anschaulich ist, zeigen wir in Abb. 4.3 eine PSLOG-Zeichnung, die unsere SLOG-Repräsentation respektiert. Unser Ziel wird es sein, dieser mit Hilfe unserer Heuristik so nahe wie möglich zu kommen. Als nächstes beschreiben wir Schritt für Schritt, was unsere Heuristik mit S macht.

(1): Die Lagenzuordnung können wir z.B. so wie in Abb. 4.3 festlegen. Hierfür können wir sagen, dass die Kante aus dem Knoten 1 hinaus nach oben verläuft. Damit ist die Ausrichtung aller Kantensegmente wie in dieser Zeichnung.

(2): In diesem Schritt verändern wir die Binärstrings in der SLOG-Repräsentation. Wir können direkt durch Zählen feststellen, dass wir eine Rotation im Uhrzeigersinn durchführen müssen. Die folgenden Listen zeigen den Schritt vor der Halbierung der Binärstrings, wobei die unterstrichenen Binärstrings ein- oder zweimal

abgeändert wurden:

$$H(f_1) = ((1, \underline{\epsilon}, 90), (c_1, \underline{\epsilon}, 90), (c_3, \underline{\epsilon}, 360), (2, \underline{\epsilon}, 90), (c_3, \underline{\epsilon}, 270), (3, 11, 180),$$
$$(4, \epsilon, 180), (5, \epsilon, 270), (6, \underline{\epsilon}, 90), (c_2, \underline{11}, 90), (c_1, \underline{\epsilon}, 360), (7, \underline{\epsilon}, 90), (c_1, \underline{\epsilon}, 360)),$$
$$H(f_2) = ((c_1, \underline{00}, 90), (c_2, \underline{\epsilon}, 90), (c_3, \underline{\epsilon}, 90)),$$
$$H(f_3) = ((c_2, \underline{\epsilon}, 90), (6, \epsilon, 180), (5, \epsilon, 90), (4, \underline{00}, 90)),$$
$$H(f_4) = ((c_2, 11, 90), (4, 00, 90), (3, \underline{\epsilon}, 90), (c_3, \underline{\epsilon}, 90)).$$

Nach Halbieren der Binärstrings in dieser Liste erhalten wir dann die gewünschte orthogonale Repräsentation.

(3): Wenn wir nun durch $FIND\_LENGTHS$ unsere SLOG-Repräsentation zeichnen lassen und sie rotieren, erhalten wir im Wesentlichen eine Zeichnung wie in Abb. 4.4, wobei die genauen Längen der Segmente keine Relevanz für uns haben.

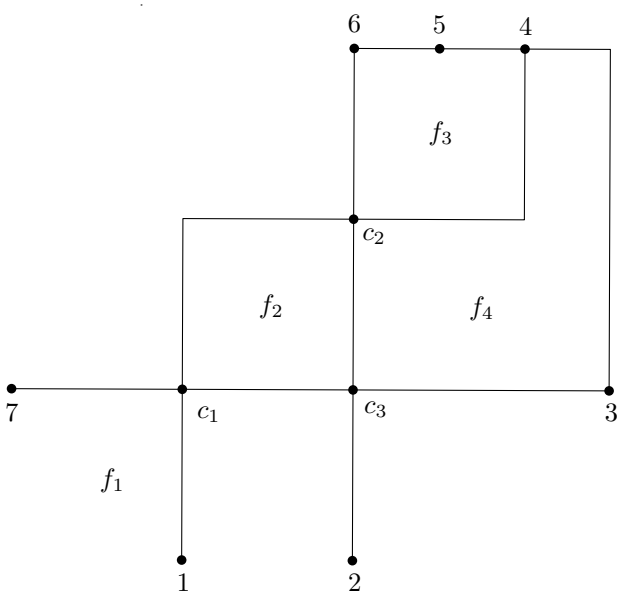


Abbildung 4.4: orthogonale Zeichnung

(4): In diesem Schritt machen wir durch Korrigieren der Knicke und Einfügen von Löffeln (siehe Abb. 4.5) aus dieser orthogonalen Zeichnung wieder eine PSLOG-Zeichnung.

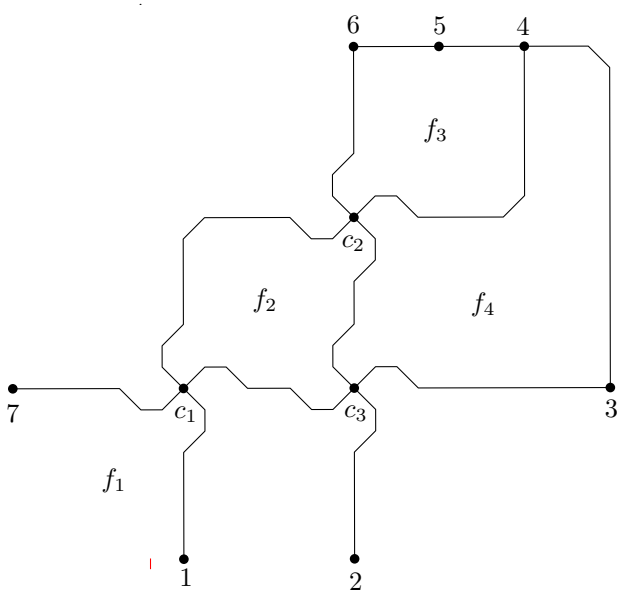


Abbildung 4.5: erzeugte PSLOG-Zeichnung

Wenn man diese Zeichnung mit der aus Abb. 4.3 vergleicht, dann sieht man, dass die Portzuweisungen übereinstimmen und die rr-Kanten bereits die gleiche Anzahl von Knicken haben.

(5): Nachdem wir nun eine PSLOG-Zeichnung haben, bearbeiten wir diese mit dehnbaren Schnitten, um die überschüssigen Knicke wieder zu entfernen.
In Abb. 4.6 können wir vier solcher Schnitte sehen, durch die wir nacheinander die diagonalen Segmente, die durch den Schnitt gehen, entfernen.

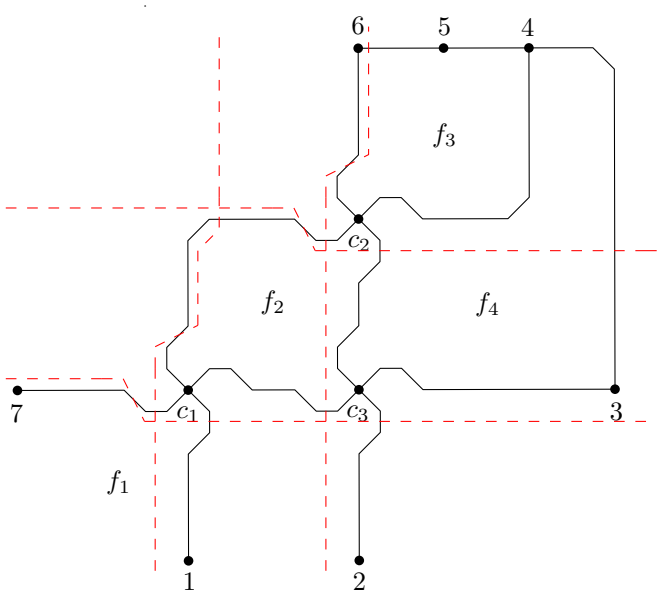


Abbildung 4.6: Anwendung von dehnbaren Schnitten

Wenn wir nun alle hinteren Diagonalsegmente der Löffel entfernt haben, sieht

unsere Zeichnung wie in Abb. 4.7 aus.

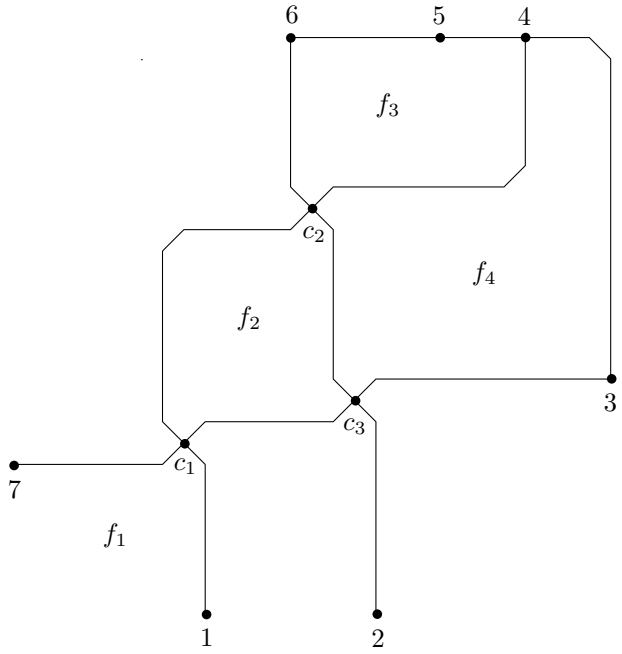


Abbildung 4.7: alle hinteren Diagonalsegmente entfernt

(6): Als nächstes betrachten wir alle übrigen Kantensegemente in einem RL-Knick. Zunächst entscheiden wir ob mehr x-monotone oder y-monotone dehnbare Schnitte benötigt werden, um diese Segmente zu entfernen.

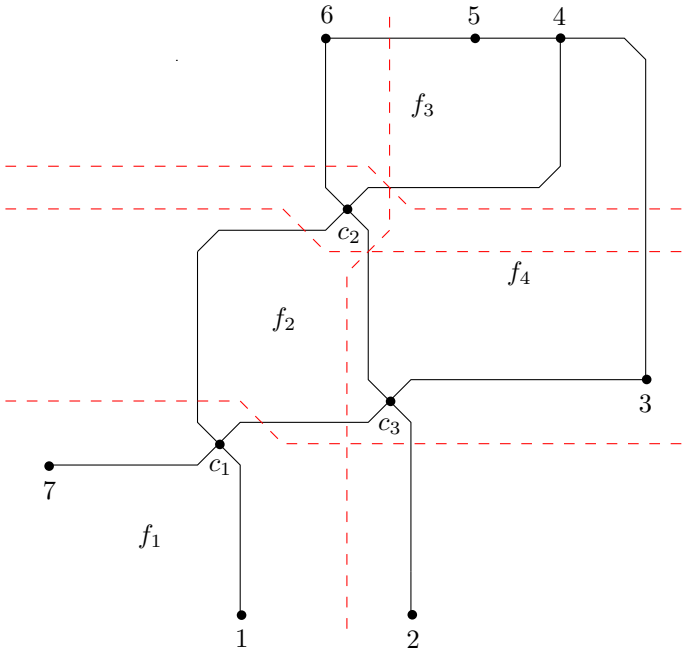


Abbildung 4.8: alle übrigen Schnitte

In unserem Beispiel können wir in Abb. 4.8 erkennen, dass es drei x-monotone und einen y-monotonen Schnitt gibt. Also entscheiden wir uns, die Segmente zu entfernen, die x-monotone Schnitte benötigen. Das Ergebnis sieht wie in Abb. 4.9 aus.

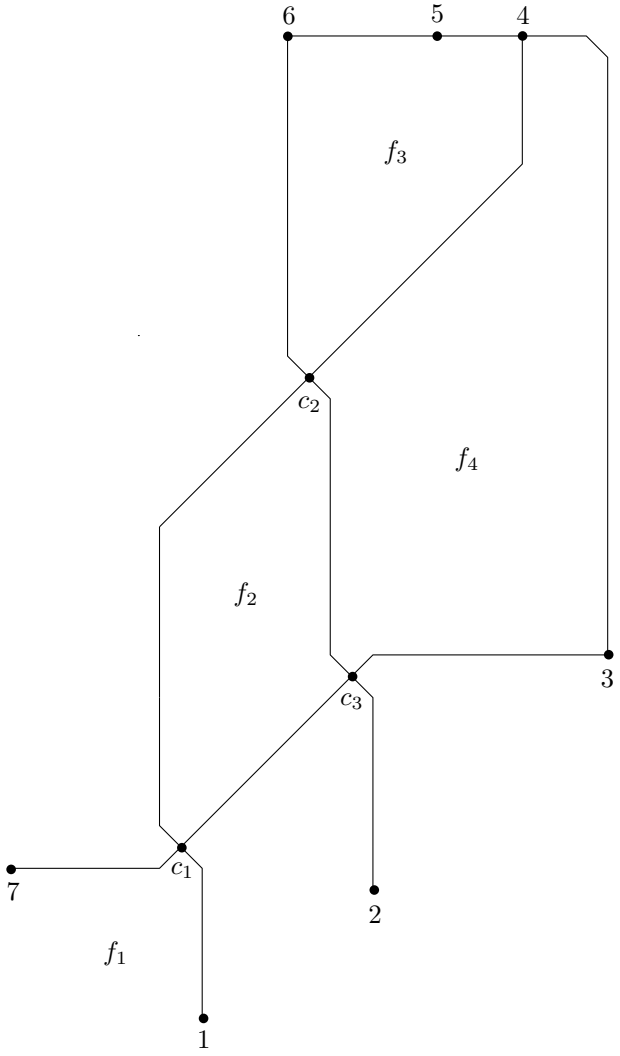


Abbildung 4.9: endgültige PSLOG-Zeichnung

Zum Einen ist hier besonders auffällig, dass der Flächenbedarf durch lediglich drei Vergrößerungen stark zugenommen hat. Darauf werden wir im folgenden Abschnitt bei dem es um die Eigenschaften unserer generierten Zeichnung geht, nochmal explizit eingehen.

Zum Anderen fällt auf, dass man mit dieser Methode der Beseitigung von Knicken in diesem Beispiel nun tatsächlich keine PSLOG-Zeichnung generieren kann, die komplett die SLOG-Repräsentation S respektiert. Dies liegt daran, dass es keinen dehnbaren Schnitt durch das Kantensegment zwischen c_2 und c_3 gibt, da man f_2 nicht durch ein horizontales Segment nach unten oder oben verlassen kann.

4.5 Eigenschaften der erzeugten Zeichnung

Dieser Abschnitt behandelt die Güte unserer Zeichnung im Vergleich mit der zugrunde liegenden SLOG-Repräsentation, welche gezeichnet werden soll.

Zunächst formulieren und beweisen wir einige Eigenschaften unserer erzeugten Zeichnung.

4.5.1 Satz *Sei S die Eingabe des rotationsbasierten Algorithmus aus 4.3 und D die am Ende entstandene PSLOG-Zeichnung von S. Zudem sei RC die Anzahl der rc-Kanten, CC die Anzahl der cc-Kanten in S, und k die Anzahl der Knicke, um welche D von S abweicht. Dann gelten folgende Aussagen:*

(i) $k \leq \frac{1}{2}RC + CC$.

(ii) Die k überschüssigen Knicke sind RL-Knicke, anliegend an c-Knoten, auf unterschiedlichen Kanten.

(iii) Wenn man aus D die überschüssigen k Knicke entfernen würde, dann würde D die SLOG-Repräsentation S respektieren.

(iv) Der Flächenbedarf der PSLOG-Zeichnung liegt nach Schritt 5 des Algorithmus in $O(n^2)$, wobei n die Summe der Knoten und der Knicke ist.

Beweis: (i): Hierfür zeigen wir zunächst, dass genau die Knicke, welche in Schritt 6 nicht entfernt wurden, den überschüssigen K Knicken entsprechen.

Zunächst stellen wir fest, dass Abweichungen allgemein nur an c-Knoten zu erwarten sind, da unser Algorithmus bei rr-Kanten und bei den anderen Kantensegmenten diese lediglich zu orthogonalen Knicken in Schritt 2 und danach wieder zurück zu Halbknicken in Schritt 4 transformiert.

Nun zeigen wir, dass eine rc-Kante e aus D, bei der in Schritt 2 durch die Rotation ein Knick entfernt wurde, S respektiert (d.h. die Kante hat die gleichen Knicke in der Zeichnung wie in S). Hierfür zählen wir die Knicke von e im Laufe des Algorithmus. In Schritt 2 wird ein Knick nach Voraussetzung entfernt und beim Einfügen der Löffel erhält e drei zusätzliche Knicke. Schritt 5 entfernt die Diagonale in jedem Löffel und senkt somit die Knickanzahl wiederum um 2. Also erhalten wir am Ende genau die gleiche Knickanzahl wie in S, da Schritt 6 nur RL-Knicke bearbeitet und diese nicht mehr vorliegen. Die Art der Knicke ist auch gleich, da sie bis auf den bereits entfernten Knick im Löffel gleich gelassen wurden.

Mit analoger Argumentation lässt sich zeigen, dass sich die Knickanzahl bei den anderen rc-Kanten bis Schritt 5 um zwei erhöht hat, und diese somit, wegen übereinstimmenden Ports von S, einen RL-Knick am c-Knoten enthalten. Bei cc-Kanten kann man beide Argumente wiederverwenden, da stets ein Ende in Schritt 2 einen Knick mehr bekommt und das andere einen weniger, weil ihre Binärstrings einmal von einer Facette und einmal von der anderen Facette betrachtet werden. Insgesamt hat eine cc-Kante also nach Schritt 5 einen RL-Knick mehr als in S. Nach Schritt 5 haben wir also folgende Ungleichung:

$$k \leq 2(\frac{1}{2}RC + CC)$$

MA/Bilder/ObereSchranke.eps MA/Bilder/ObereSchranke.eps

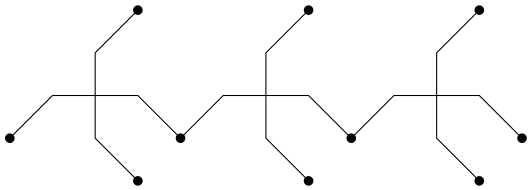


Abbildung 4.10: Nach rechts erweiterbare Graphenklasse

Der Faktor 1/2 vor RC kommt daher, dass wir in Schritt 2 unsere Rotation bewusst so gewählt haben, dass bei mindestens der Hälfte der Knoten keine Kante eingefügt wird.

Schritt 6 entfernt dann nochmals mindestens die Hälfte der nun vorhandenen RL-Knicke und wir erhalten somit:

$$k \leq \frac{1}{2}(2(\frac{1}{2}RC + CC)) = \frac{1}{2}RC + CC$$

(ii): Diese Aussage wurde zuvor bereits mitbewiesen, da angegeben wurde, wie die RL-Knicke zustande kommen.

(iii): Hier ist lediglich zu sagen, dass zum Respektieren einer SLOG-Repräsentation die Knicke der Kanten und die Winkel an den Knoten mit dieser übereinstimmen müssen. Mit Aussage *(ii)* und der Erhaltung der Portzuweisungen in Schritt 3 des Algorithmus ist klar, dass die Entfernung dieser RL-Knicke eine Zeichnung die S respektiert generiert.

(iv): Die grundlegende Zeichnung, die von $FIND\_LENGTHS$ erzeugt wird, hat den erwünschten Flächenbedarf [8]. In Schritt 4 vervielfachen wir einmalig die Fläche und in Schritt 5 vergrößert jede Entfernung der hinteren Diagonalen des Löffels die Zeichnung um 1 in eine Richtung. Dieser Schritt wird $4\,|C|$ mal durchgeführt, also ist unser Flächenbedarf danach immer noch $O(n^2)$. □

4.5.2 Bemerkung Die obere Schranke in Satz 4.5.1 ist mit den SLOG-Repräsentationen zu der Graphenklasse aus Abb. 4.10 scharf.

Das liegt daran, dass eine beliebige Rotation an jedem Kreuzungsknoten zwei Knicke erzeugt, welche nicht parallel zueinander sind. Diese können somit nicht gleichzeitig in Schritt 6 der rotationsbasierten Heuristik entfernt werden. Also bleibt an jeweils 4 rc-Knoten, ein RL-Knick übrig.

4.5.3 Bemerkung Die Aussage *(iv)* ist so formuliert, da wir nach Schritt 6 keine polynomielle Schranke für die Fläche angeben können. Die Heursitik ist darauf ausgelegt, dass man theoretisch, falls der Flächenbedarf eine große Rolle spielt Schritt 6 weglassen kann. Dies führt im Worst-Case zu einer Verdoppelung der Extraknicke, wie man im Beweis von *(i)* sehen kann. Weiteres zum Flächenbedarf wird im folgendem Absatz erläutert.

Wir hatten am Anfang des Kapitels in Satz 4.1.1 dargestellt, was mit einer anderen Heuristik bereits gezeigt wurde.

Unsere Heuristik hat durch ihren etwas anderen Ansatz nach dem letzten Schritt keinen garantierten polynomiellen Flächenbedarf, jedoch ist das bei einer

gewissen Knickanzahl auch notwendig. Dies hängt damit zusammen, dass knickminimale SLOG-Zeichnungen im Allgemeinen keinen polynomiellen Flächenbedarf haben [2] und wir diesen demnach aufgeben müssen, wenn wir Knicke soweit wie möglich reduzieren wollen.

Zudem ist es bei unserer Heuristik so, dass wir in vielen Fällen, weit mehr Knicke reduzieren können, als uns die Schranken angeben. Das liegt daran, dass wir in unserem Algorithmus zuerst unsere Rotation in Schritt 2 wählen. Dies kann theoretisch bereits dazu führen, dass wir am Ende keine Extraknicke haben. Als zweites treffen wir in Schritt 6 die Entscheidung, welche Doppelknicke noch entfernt werden. Dabei kann es auch passieren, dass wir unter Umständen alle RL-Knicke entfernen.

Kapitel 5

Nachbearbeitung der Heursitik

In diesem Abschnitt beschreiben wir weitere Möglichkeiten, um das Ergebnis der rotationsbasierten Heuristik zu verbessern.

5.1 Knickreduzierung

Die erste Art und Weise zum Bearbeiten der Ausgabe unserer Heuristik ist eine allgemeine Methode zum Entfernen von RL-Knicken. Hierzu hatten wir in Kap. 3.3 den Algorithmus `Dehnbarkeitstest`, den wir auf alle Segmente der RL-Knicke in unserer entstehenden Zeichnung anwenden können. Hierbei haben wir keine Garantie, dass weitere Kantensegmente entfernt werden können, wie man an Beispiel 4.4 sehen kann. Jedoch kann es sein, dass wir bei symmetrischen Graphen sehr viele weitere Knicke entfernen können.

Zur Laufzeit können wir sagen, dass `Dehnbarkeitstest` für jedes Kantensegment eine asymptotische Laufzeit von $O(n^2)$ hat, wobei n die Anzahl der Knoten des Graphen sind. Also erhalten wir insgesamt eine Laufzeit von $O(k \cdot n^2)$, wobei k die Anzahl der RL-Knicke ist, die noch in unserer Zeichnung vorhanden sind. Diese Zahl wurde im vorherigen Abschnitt nach oben beschränkt.

5.2 Flächenminimierung

Zur Verringerung der Gesamtfläche kann man eine Standardtechnik verwenden, welche auch für orthogonale Zeichnungen verwendet wird. Die Methode des Moving wurde bereits zu Beginn von Kapitel 3 erläutert. Diese Methode sucht Kurven durch den Graphen, welche gewissen Eigenschaften genügen müssen. Anhand dieser kann man dann ähnlich wie bei den hier verwendeten Schnitten die Zeichnung verkleinern anstatt sie zu vergrößern. Die genauen Details hierzu werden bei der Definition von „area-saving moving-line“ in [5] beschrieben.

Kapitel 6

Experimentelle Analyse der Heuristik

Die zuvor beschriebene Heuristik wurde weitestgehend in Java implementiert und dann mit Hilfe von RStudio Grafiken und Berechnungen erstellt. Als Testmenge für die im Weiteren beschriebene Analyse wurden von uns die Rome-Graphen verwendet. Hierbei handelt es sich um eine Datenbank von 11534 Graphen, von der Seite www.graphdrawing.org, welche oft als Testmenge dienen.

6.1 Aufbau der Implementierung

Die Heuristik wurde in Java programmiert und als zusätzliche Datenbanken wurden die yFiles von yWorks und der Gurobi LP Solver verwendet. Da wir als Eingabe eine SLOG-Repräsentation benötigen, bearbeiteten wir die Romegraphen, welche lediglich die Knoten und die Kanten als Mengen ohne Zusatzinformation enthalten. Hierfür haben wir für diese Graphen zuerst ein Layout mit einem kräftebasierten Algorithmus aus den yFiles berechnen lassen. Danach wurde einen modifizierter Tamassia darauf angewendet, welcher bereits von Bekos et al. [2] verwendet wurde. Dieser berechnete dann eine knickminimale SLOG-Repräsentation, welche als Eingabe für unsere Heuristik diente. Der Algorithmus hat nebenbei noch mit einem Linearen Programm eine Zeichnung generiert, welche wir im Weiteren jedoch nicht benötigen werden.

An dieser Stelle möchte ich einen großen Dank an Robert Krug einen Mitautor von „ Slanted Orthogonal Drawings“[2] aussprechen, da er mir ihre Implementierung zur Verfügung gestellt hat und mir stets bei Fragen helfend zur Seite stand.

Die weiteren Schritte der rotationsbasierten Heuristik wurden nur an den Ports, der Kanten und an den Binärstrings der SLOG-Repräsentation durchgeführt. Zuerst wurden hierbei die Löffel eingefügt und daraufhin die hinteren entstandenen RL-Knicke wieder entfernt. Als letztes wurde mit Hilfe der Ports festgestellt, ob es mehr RL-Knicke gibt, die x-monotone Schnitte benötigen. Falls dies der Fall war wurden alle solchen RL-Knicke entfernt und ansonsten wurden alle, die y-monotone Schnitte benötigen, entfernt.

Für die Analyse wurden einige im nächsten Absatz genannten Variablen des Graphen zwischendrin oder danach in Textdateien gespeichert.

6.2 Analyse der Heuristik

Für die Analyse wurden wie bereits erwähnt die 11534 Graphen aus der Rome-Datenbank herangezogen. Wir haben daraus drei Testmengen an Graphen generiert. Die erste und kleinste entsteht durch Filtern der Graphan nach zusammenhängenden nicht-planaren Maximalgrad-4 Graphen. Dies liefert eine Testmenge von 83 Graphen, welche im folgenden mit T83 abgekürzt wird. Diese ist für eine Analyse sehr interessant, weil unsere Heuristik bei planaren Graphen keine Änderung durchführt. Als Obermenge von T83 erhalten wir 1122 Graphen (T1122). Bei diesen wurde der Filter für nicht-planare Graphen nicht verwendet. Diese Testmenge wurde auch für die Analyse, auf die sich Satz 4.1.1 bezieht, verwendet. Wobei bei dieser Analyse von Bekos et al. [2] alle planaren Graphen auch eben eingebettet wurden. Bei uns wurden 362 davon nicht eben eingebettet und diese Tatsache erzeugt eine gewisse Abweichung von der anderen Analyse.

Als meiner Meinung nach interessanteste Testmenge der Analyse hat sich jedoch eine andere Testmenge herausgestellt. Hierzu haben wir für alle Graphen, bei allen Knoten mit Grad größer 4 solange Kanten entfernt bis sie nur noch Grad 4 hatten. Hierbei haben wir stets die Kanten gewählt, bei welchen der zugehörige Nachbarknoten den höchsten Grad hat, um den Zusammenhang möglichst lange zu erhalten. Nach einer Filterung der nun veränderten Graphen haben wir 3108 nicht-planare, zusammenhängende Graphen mit Maximalgrad 4 erhalten. Da beim Einlesen ein paar Graphen Fehler erzeugten, dienten dann letzendlich von den 3108 Graphen 2995 Graphen als Testmenge (T2995).

Im Weiteren zeigen wir zwei Tabellen von einigen wichtigen Variablen zu den drei beschriebenen Testmengen. Wir werden danach diese Spalten und deren Aussage näher erläutern.

Menge	Knoten	∅ Knoten	Dichte	∅ Kreuzungen	∅Verhältnis Schnittart
T83	11-56	34,3	0,62	5,7	1,88
T1122	10-56	19,6	0,54	0,9	1,37
T2995	11-100	61,5	0,61	17,7	1,36

Menge	∅ HK. knickminimal	∅ HK. Heuristik	∅ HK. obere Schranke
T83	20,8	27,8	32,2
T1122	4,0	4,9	5,7
T2995	54,0	81,4	89,6

Das Symbol ∅ in den Tabellen steht für den Mittelwert und die Abkürzung HK für Halbknicke.

Knoten, ∅ Knoten: An diesem Wert sehen wir bereits einen großen Unter-

MA/Bilder/Knoten.pdf MA/Bilder/Knoten.pdf

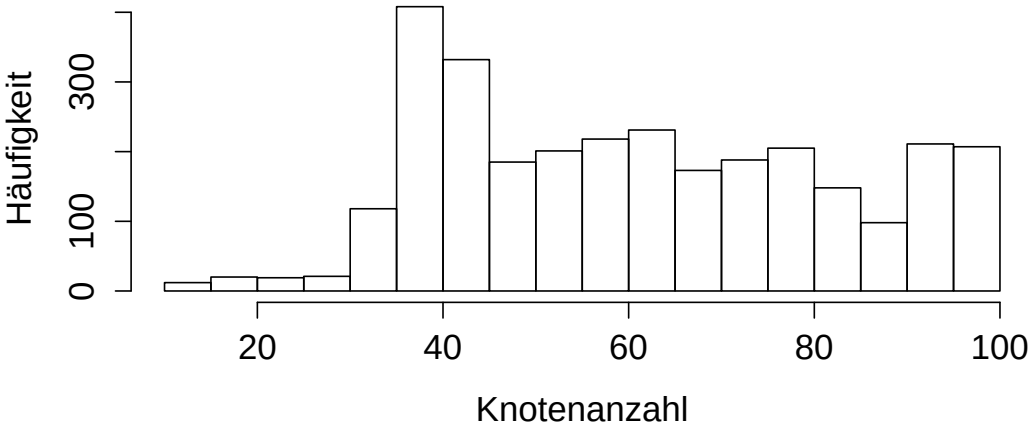


Abbildung 6.1: Verteilung der Knotenanzahl in R2995

schied in der Größe der Graphen dieser Datenmengen. Vor allem T1122 hat sehr wenig Knoten, was vor allem an der Planarität der meisten Graphen liegt. Eine genauere Darstellung der Knotenanzahl kann man in Abb. 6.1 sehen.

Hierbei fällt auf, dass so geringe Knotenanzahlen wie bei dem Durschnschnitt von T1122 fast gar nicht auftreten, was große Auswirkungen auf die Werte in der zweiten Tabelle hat.

Dichte: Hier wird mit der Dichte der Quotient aus der Kantenmenge und der maximalen Kantenmenge eines 4-regulären Graphs mit der gleichen Knotenanzahl bezeichnet. Dies soll zeigen, dass T83 und T2995 strukturell bis auf die Größe der Graphen ähnlich aufgebaut sind.

∅ **Kreuzungen:** Unsere Heuristik führt an rr-Kanten insgesamt keine Veränderung durch. Das meiste geschieht anliegend an c-Knoten, also Kreuzungsknoten. Demnach wird T2295 mit durchschnittlich 17,7 Kreuzungen, die aussagekräftigste Menge für unsere Heuristik sein. Bei T1122 sind 794 Graphen eben eingebettet und haben somit keine c-Knoten. Hier ist noch anzumerken, dass die c-Knoten nicht in die vorigen Knotenberechnungen eingegangen sind, da sie danach erst erzeugt wurden.

∅**Verhältnis Schnittart:** Dieser Wert entpricht dem Durchschnitt der Maxima von $(\frac{x-monoton}{y-monoton}, \frac{y-monoton}{x-monoton})$ über alle Graphen. Hierbei bezeichnen wir mit $x - monoton$ die Anzahl der RL-Knicke, die nach Schritt 5 unserer Heursitik einen x-monotonen Schnitt benötigen. $y - monoton$ wird analog definiert. Falls einer dieser Werte 0 annimmt nehmen wir lediglich den größeren der beiden. Unsere Heuristik ist besser, wenn dieses Verhältnis möglichst groß ist, da die Merhheit dieser RL-Knicke dann entfernt werden kann.

In der Praxis nähert sich dieser Wert bei zunehmender Knotengröße leider eher dem Wert 1, wie wir in Abb. 6.2 an T2295 sehen können.

Dafür kann es vielerlei Gründe geben, wobei wir den wahrscheinlich Wichtigsten hier nennen werden. Es ist sicherlich sinnvoll anzunehmen, dass ein RL-Knick

keine Präferenzen hat welche Art von Knick er benötigt. Also können wir das stochastisch als eine Münze modellieren, die wir so oft werfen wie es RL-Knicke in der Zeichnung gibt. Dieses Modell genügt dem starken Gesetz der großen Zahlen und somit konvergiert das oben definierte Verhältnis bei steigender Knickanzahl gegen 1.

Die Werte der Menge T83 schwanken wegen der kleinen Datenmenge extrem

MA/Bilder/Verh.pdf MA/Bilder/Verh.pdf

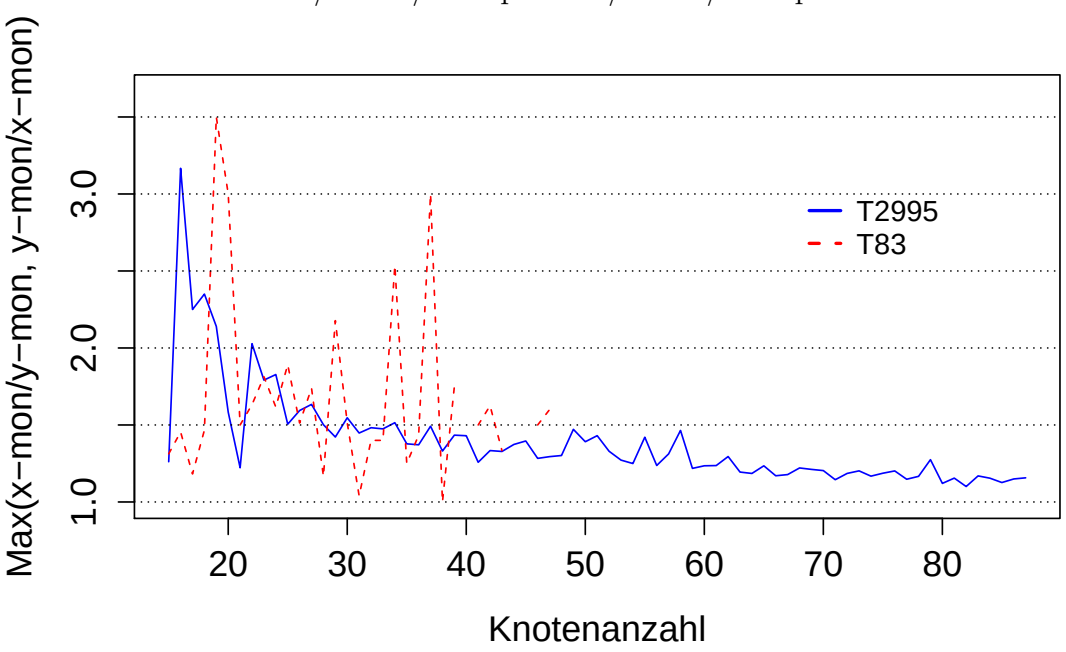


Abbildung 6.2: Verhältnis der Schnitte bezüglich der Knotenanzahl

MA/Bilder/Knicke.pdf MA/Bilder/Knicke.pdf

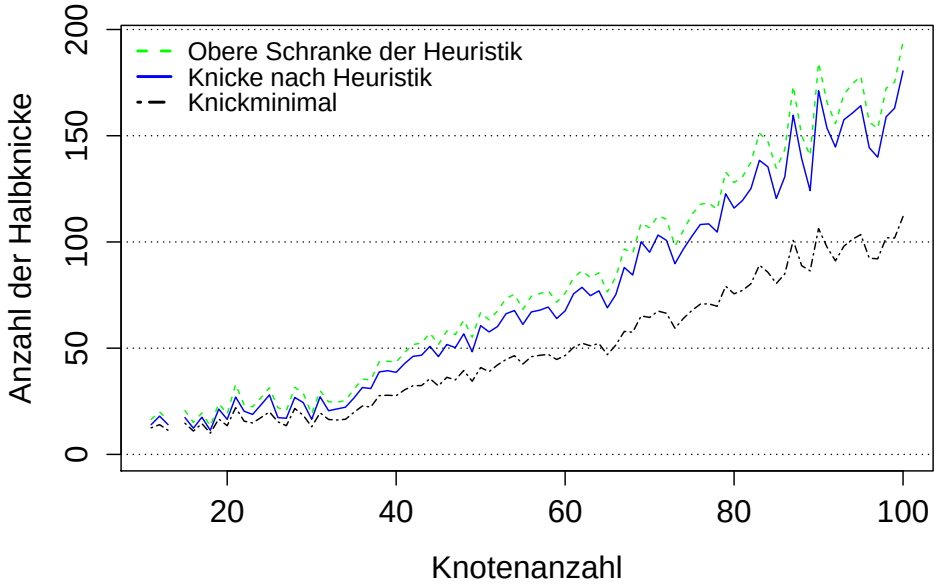


Abbildung 6.3: Knickanzahl bei T2995 im Vergleich zur Knotenanzahl

und liefern nicht direkt ein Anzeichen einer konsistenten Abnahme bei höherer Knotenzahl.

Als nächstes kommen wir zu den drei wichtigsten Werten, welche die Güte unserer Heursitik anhand ihrer Knickanzahl im Vergleich zur knickminimalen Zeichnung zeigt.

∅ HK. knickminimal, ∅ HK. Heuristik, ∅ HK. obere Schranke: Der erste Wert gibt die Halbknickanzahl einer knickminimalen SLOG-Repräsentation an. Der zweite ist die Anzahl der Halbknicke in der Ausgabe unserer Heuristik und der letzte Wert entspricht der Knickanzahl der oberen Schranke in Satz 4.5.1*(i)*.

Wir können sehen, dass wir einen ähnlich großen Wert in T2995 unter der oberen Schranke bleiben wie bei T83. Dies liegt vor allem daran, dass sich die Schnittarten eher ausgleichen, wie man in Abb. 6.2 bei großen Graphen sieht. Die Tendenz der drei Werte können wir in Abb. 6.3 sehen.

In Abb. 6.3 können wir wiederum sehen, dass die Heuristik in etwa konstant unter ihrer oberen Schranke bleibt. Bei der Testmenge T1122 haben wir im Schnitt 0.9 Extraknicke pro Graph, was wiederum an der Planarität der meisten Graphen dieser Menge liegt.

Im folgenden Absatz gehen wir im Vergleich zur Heuristik von Bekos et al. [2] nochmal näher auf diese Werte ein.

6.3 Vergleich mit der löffelbasierten Heuristik

Am Anfang von Kapitel 4 in Satz 4.1.1 hatten wir bereits die theoretische Aussage der auf Löffel basierenden Heuristik von Bekos et al. [2] gezeigt.

Da dieser Satz zwischen rc-Kanten ohne Knick und welchen mit Knicken unterscheidet, können wir diese Aussage nicht direkt, bezüglich der Knickanzahl, mit unserer Aussage aus Satz 4.5.1 vergleichen. Unsere Heursitik versucht asymmetrische Konstellationen in der Zeichnung auszunutzen um möglichst viele Knicke zu entfernen. Einen Teil den die löffelbasierte Heuristik garantiert ist der quadratische Flächenbedarf, welchen unser Algorithmus nach Schritt 6 nicht mehr garantieren kann.

Die andere Heuristik wurde von Bekos et al. auch auf T1122 getestet und in Abb. 6.4 kann man ein Bild aus dieser Analyse im Vergleich zu unserer Analyse auf T1122 sehen (siehe Abb. 6.5).

Ein signifikanter Unterschied zwischen den Analysen ist jedoch, dass wir nicht jeden planaren Graphen eben eingebettet haben und somit 362 Graphen mit Kreuzungen erhalten. Hiervon haben jedoch 153 nur einen c-Knoten und können fast immer knickminimal durch unsere Heuristik berechnet werden. Der Quetient zwischen Knicken nach der Heuristik zu den Knicken der knickminimalen Zechnung liegt bei der Löffelbasierten auf T1122 bei 1.18. Bei unserer Heuristik liegt der Quetient bei 1.21.

MA/Bilder/KnickeT.pdf MA/Bilder/KnickeT.pdf

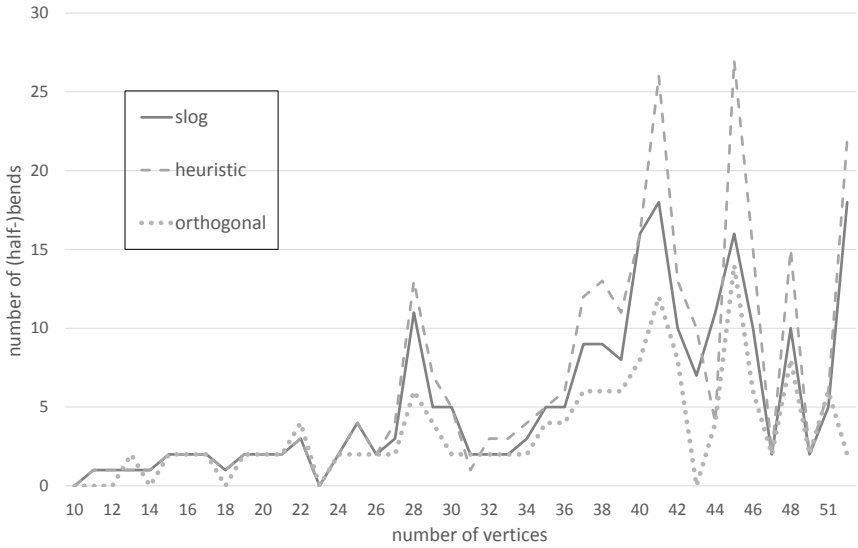


Abbildung 6.4: Bild aus der Analyse der löffelbasierten Heuristik auf T1122

MA/Bilder/Knicke2.pdf MA/Bilder/Knicke2.pdf

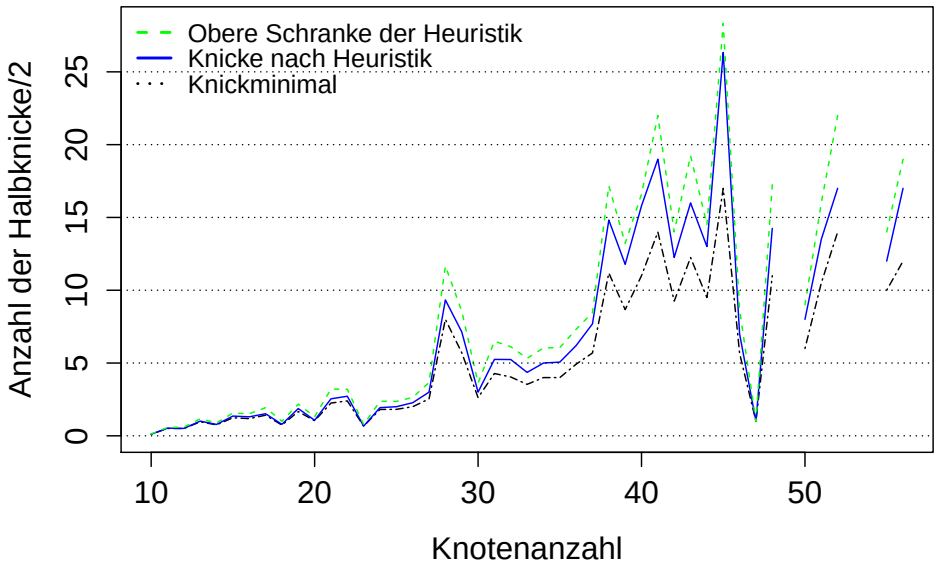


Abbildung 6.5: Anzahl der Knicke(Anzahl der Halbknicke/2) auf T1122

MA/Bilder/GrossesBspKM.pdf MA/Bilder/GrossesBspKM.pdf

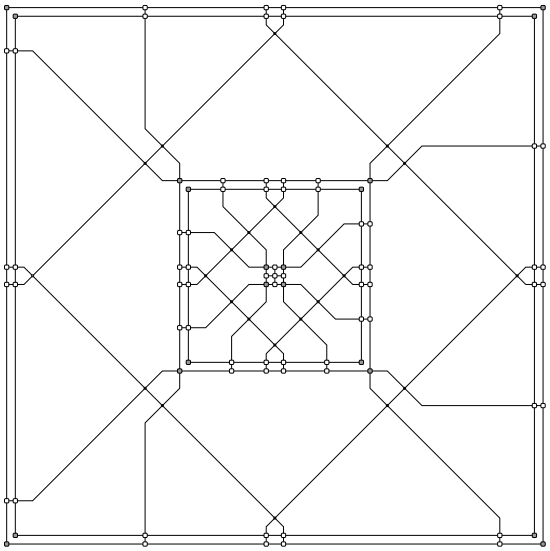


Abbildung 6.6: Knickminimale Zeichnung eines Graphen

Wir möchten nun ein Beispiel zeigen, an dem wir sehen können, dass die rotationsbasierte und die löffelbasierte Heuristik unterschiedliche Zeichnungen erzeugen. In der Zeichnung in Abb. 6.6, welche aus [2] entnommen ist sieht man eine knickminimale Zeichnung.

Wenn unser Algorithmus die zugrunde liegende Repräsentation dieser Zeichnung als Eingabe erhält, generiert er nach der Nachbearbeitung, wie in Kapitel 5 beschrieben, die Zeichnung aus Abb. 6.7. Die Knickanzahl in der Zeichnung entspricht vor der Nachbearbeitung genau unserer oberen Schranke, da die Eingaberepräsentation sehr symmetrisch ist und wir deshalb bei beiden Rotationen nur genau die Hälfte der RL- Knicke entfernen, beziehungsweise zu Beginn verhindern können. Eine Zeichnung von dem gleichen Graphen, nach der Löffelbasierten Heuristik, wird in Abb. 6.8 gezeigt.

Als letztes Beispiel zeigen wir noch eine Zeichnung eines Graphen, welche durch unsere rotationsbasierte Heuristik knickminimal gezeichnet wird (siehe Abb. 6.9). Dies liegt daran, dass die erste Roation in Schritt 2 keine orthogonalen Knicke auf rc- Kanten erzeugt und somit vor Schritt 6 nur noch die vier RL-Knicke auf den cc-Kanten übrig bleiben. Da diese Knicke gleich ausgerichtet sind werden sie alle zusammen durch vier weitere Schnitte entfernt.

MA/Bilder/GGKM.eps MA/Bilder/GGKM.eps

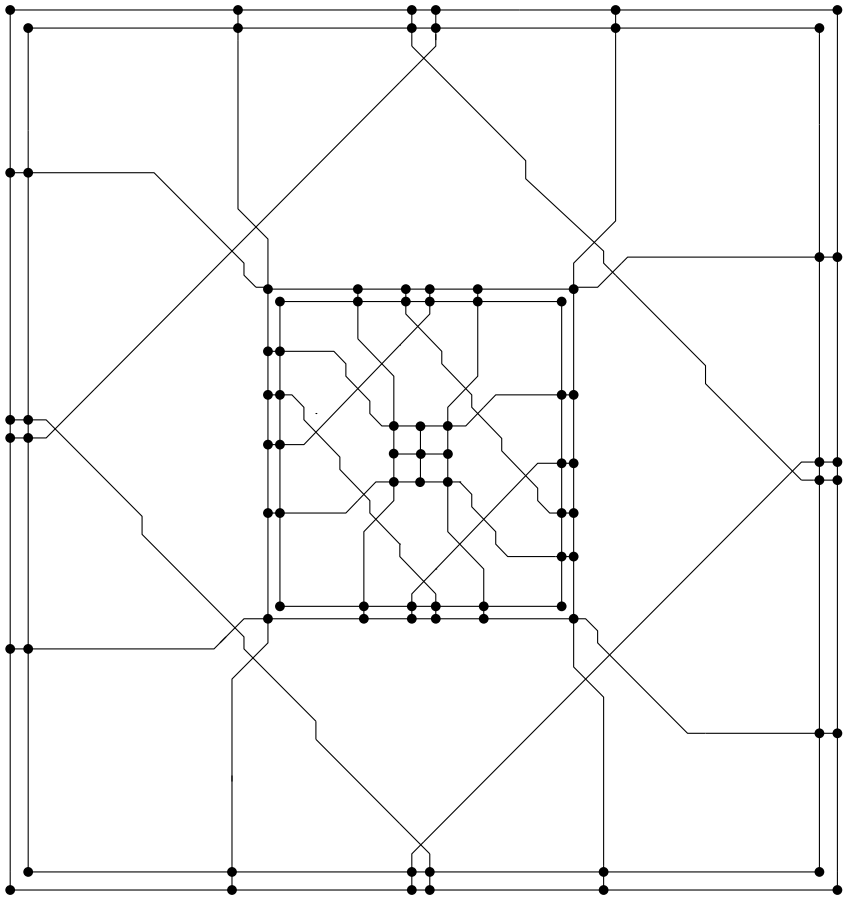


Abbildung 6.7: Zeichnung nach unserer Heuristik und Nachbearbeitung

MA/Bilder/GrossesBspHT.pdf MA/Bilder/GrossesBspHT.pdf

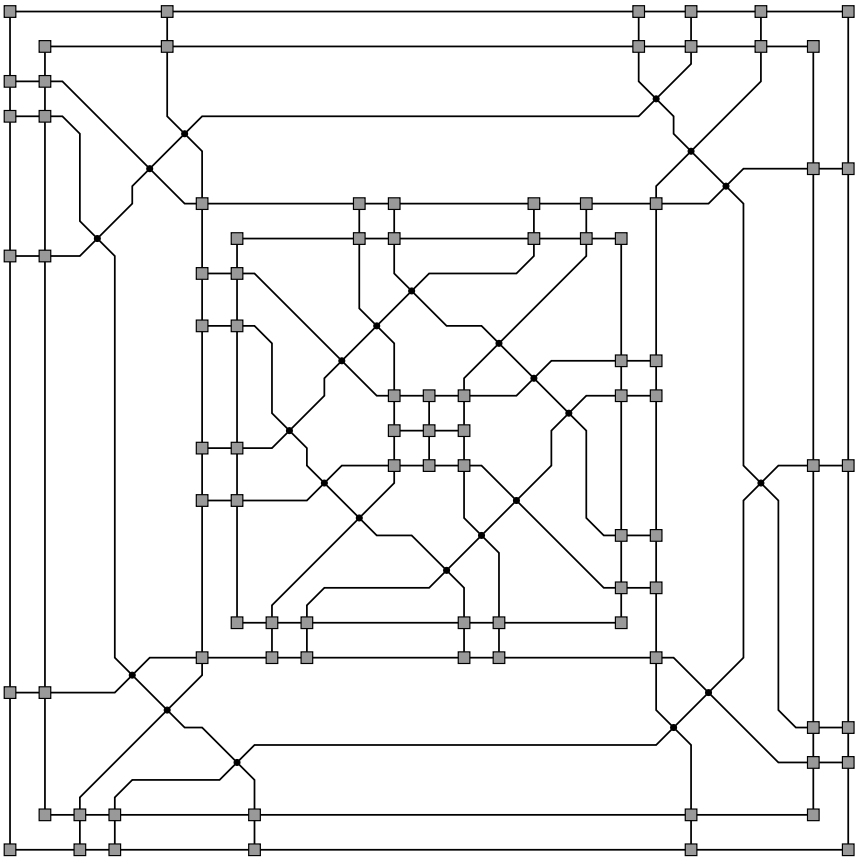


Abbildung 6.8: Zeichnung des Graphen nach der Heuristik von Bekos et al.

MA/Bilder/BspKM.eps MA/Bilder/BspKM.eps

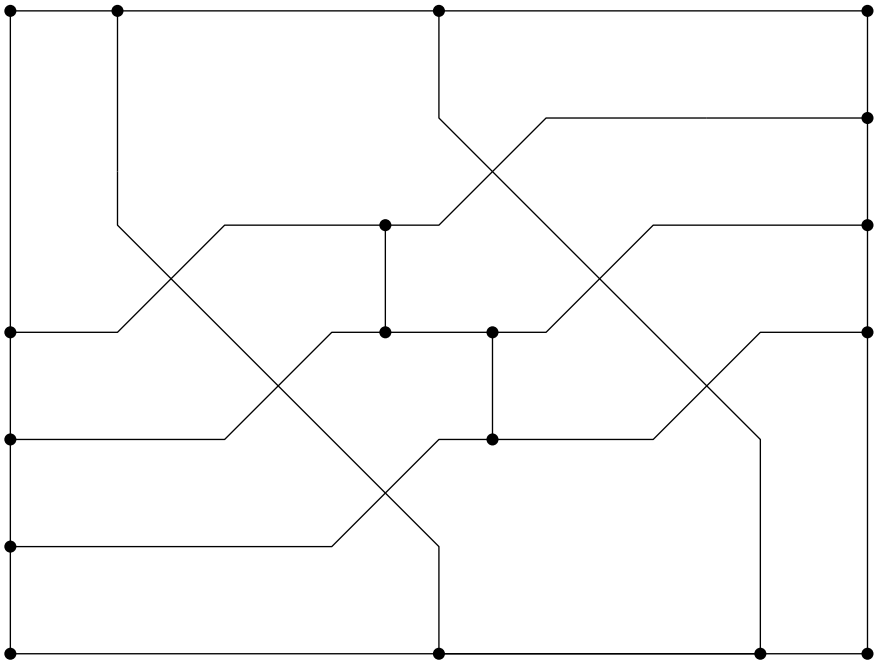


Abbildung 6.9: Zeichnung, die unsere Heursitik knickminimal erzeugt

Kapitel 7

Fazit und offene Fragen

In dieser Arbeit wurde zunächst eine Methode zum Bearbeiten von schieforthogonalen Zeichnungen in Form von Schnitten gezeigt. Zudem wurde ein Algorithmus angegeben um dehnbare Schnitte in Graphen zu finden, falls solche existieren.

Daraufhin haben wir eine rotationsbasierte Heuristik angegeben, welche eine schieforthogonale Repräsentation mit so wenig Veränderungen wie möglich zeichnet. Als Ergebnis der rotationsbasierten Heuristik erhielten wir eine Zeichnung, die sich lediglich um RL-Knicke von der vorgegebenen Repräsentation unterscheidet.

Daraufhin wurde, um eine Analyse der Heuristik zu erstellen, ein Java Programm geschrieben, welches die Heuristik an Graphen simuliert. Dieses Programm wurde dann an mehreren Testmengen bezüglich der Abweichung der entstehenden Knicke von einer knickminimalen Zeichnung untersucht. Die entstehende Knickanzahl unserer Heuristik ist sehr von der Struktur der vorliegenden SLOG-Repräsentation abhängig. Zudem hat ein Graph, mit einer festen Einbettung, im Normalfall viele verschiedene knickminimale SLOG-Repräsentationen. Also könnten wir uns folgende Frage stellen. Gibt es eine effiziente Methode, mit der wir eine der knickminimalen Repräsentationen als Eingabe auswählen können, welche unter allen diesen Repräsentationen, die Ausgabe mit den wenigsten Extraknicken erzeugt? Eine weitere Frage, die entsteht, da wir nun zwei unterschiedliche Heuristiken vorliegen haben, ist ob man die Ergebnisse dieser beiden Heuristiken zu einer besseren Heuristik kombinieren kann. Eine Analyse der löffelbasierten Heuristik auf der Menge T2295 könnte aufzeigen, wie gut diese auf einer großen Menge nicht-planarer Graphen mit vielen Kreuzungen ist.

Zudem bleibt weiterhin die Frage offen ob jede SLOG-Repräsentation gezeichnet werden kann.

Literaturverzeichnis

[1] Muhammad Jawaherul Alam, Michael A. Bekos, Michael Kaufmann, Philipp Kindermann, Stephen G. Kobourov, and Alexander Wolff. Smooth orthogonal drawings of planar graphs. In A. Pardo and A. Viola, editors, *Proc. 11th Latin American Sympos. on Theoretical Informatics (LATIN'14)*, volume 8392 of *Lecture Notes in Computer Science*, pages 144–155. Springer-Verlag, 2014. To appear.

[2] Michael A. Bekos, Michael Kaufmann, Robert Krug, Thorsten Ludwig, Stefan Näher, and Vincenzo Roselli. Slanted orthogonal drawings: Model, algorithms and evaluations. *Journal of Graph Algorithms and Applications*, 18(3):459–489, 2014.

[3] Hans L. Bodlaender and Gerard Tel. A note on rectilinearity and angular resolution. *Journal of Graph Algorithms and Applications*, 8(1):89–94, 2004.

[4] Charles E.Leiserson. Area-efficient graph layouts. In *Proceedings of the 21st Annual Symposium on Computer Science*, pages 270–281, 1980.

[5] Ulrich Fößmeier, Carsten Heß, and Michael Kaufmann. On improving orthogonal drawings: The 4M-algorithm. *Lecture Notes in Computer Science*, 1547:125–137, 1998.

[6] Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. In Roberto Tamassia and IoannisG. Tollis, editors, *Proceedings of the 13th Symposium on Graph Drawing*, volume 894 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 1995.

[7] Martin Nollenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *Visualization and Computer Graphics, IEEE Transactions on*, 17(5):626–641, 2011.

[8] Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

[9] Roberto Tamassia and Ioannis.G. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, 1989.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

_____ _____

Ort, Datum Wadim Reimche