Universität Konstanz

Mathematisch-naturwissenschaftliche Sektion

Wissenschaftlichen Abschlussarbeit

Algorithmen zur Visualisierung planarer, partitionierter Graphen

Thomas Schank [1]

7. Januar 2001

[1] Thomas.Schank@uni-konstanz.de www.fmi.uni-konstanz.de/~schank

Inhaltsverzeichnis

Kapitel 1

Einleitung und Überblick

Dies ist die Ausarbeitung der zum Studium geforderten Wissenschaftlichen Arbeit für das Lehramt an Gymnasien.

Es geht hierbei um das automatisierte Zeichnen planarer Graphen. Die Arbeit wurde durch zwei Artikel [3] und [4] von Therese Biedl inspiriert. In diesen geht es unter anderem um sogenannte LL-Zeichnungen. Bei diesen wird ein partitionierter Graph $G = (A \cup B, E)$ so gezeichnet, dass alle Knoten aus A auf einer geraden Linie und alle Knoten aus B auf einer dazu parallelen Linie gezeichnet werden, zudem werden alle Kanten zwischen den Knoten geradlinig gezeichnet.

In [3] beschreibt T. Biedl LL-Zeichnungen partitionierter planarer Graphen und wie man sie in linearer Zeit anfertigen kann. Diese Ergebnisse werden hier in Kapitel 3 kurz vorgestellt, nachdem in Kapitel 2 die wichtigsten Grundlagen zur Graphentheorie kurz beschrieben wurden.

In [3] wird auch die sogenannte LLL-Zeichnung erwähnt, hierbei wird ein partitionierter Graph $G = (A \cup B)$ auf drei Linien gezeichnet, wobei alle Knoten aus A auf der mittleren Linie und jene aus B auf zwei dazu parallelen Linien gezeichnet werden. Direkt aufbauend auf der Arbeit von T. Biedl befasst sich Kapitel 6 mit einem Spezialfall dieser partitionierten LLL-Zeichnungen.

Kapitel 4 befasst sich mit LL-Zeichnungen von unpartitionierten Bäumen und Kapitel 5 darauf aufbauend mit allgemeinen unpartitionierten LL-Zeichnungen. Auch diese können als Spezialfall der LLL-Zeichnung aufgefasst werden, wenn nämlich die Menge A leer ist.

Kapitel 7 gibt noch einen kurzen Ausblick, wie die gewonnen Ergebnisse möglicher Weise für das Zeichnen von allgemeinen LLL-Zeichnungen hilfreich sein könnten.

Danksagung

An dieser Stelle möchte ich mich bei Prof. Dr. Dorothea Wagner bedanken die diese Arbeit ermöglicht hat. Insbesondere möchte ich mich bei Sabine Cornelsen bedanken, die Korrektur las und während dem Entstehen dieser Arbeit wertvolle Hinweise gab.

Kapitel 2

Grundlagen

Hier folgt eine kurze Beschreibung, der für diesen Aufsatz wichtigsten Graphentheoretischen Grundlagen. Manches wurde um diese Einführung kurz zu halten stark vereinfacht. Eine ausführliche Ausarbeitung findet man zum Beispiel in den Büchern von Jungnickel [1] und Swamy / Thulasiraman [2].

2.1 Graphen und Subgraphen

Ein Graph $G = (V, E)$ ist ein Paar aus zwei endlichen Mengen V und E. Die Elemente von V werden als Knoten (engl. vertices) und die Elemente aus E als Kanten (engl. edges) bezeichnet. Jede Kante $e \in E$ entspricht dabei einer Verbindung zweier Knoten in V. Man notiert $e = (v_i, v_j) \in E$ wenn es in G eine Kante zwischen den Knoten $v_i, v_j \in V$ gibt. e ist dann inzident zu v_i und v_j und die beiden Knoten v_i und v_j sind adjazent zueinander. Mit dem Grad eines Knotens bezeichnet man die Anzahl der zu dem Knoten inzidenten Kanten. Die Gesamtzahl der Knoten bezeichnet man mit $n = |V|$ und die Gesamtzahl der Kanten eines Graphen mit $m = |E|$.

Ein Graph in dem es keine Kanten zwischen dem gleichen Knoten gibt (d.h. $\forall v_i \in V : (v_i, v_i) \notin E$), und in dem keine Kante doppelt vorkommt (d.h. falls $e_a = (v_i, v_j)$ und $e_b = (v_i, v_j)$ folgt $e_a = e_b$), heißt einfacher Graph. Man spricht weiterhin von ungerichteten Graphen wenn zwischen der Richtung der Kanten nicht unterschieden wird, d.h. $(v_i, v_j) = (v_j, v_i)$.

Zur Veranschaulichung werden Graphen oft in der Ebene gezeichnet. Knoten werden als Punkte und Kanten als Linien zwischen den Knoten gezeichnet. Abbildung 2.1 zeigt Darstellungen einiger prominenter Graphen.

$G' = (V', E')$ heißt Subgraph von $G = (V, E)$ wenn $V' \subset V$, $E' \subset E$ und $e = (v_i, v_j) \in E' \Rightarrow v_i, v_j \in V'$ gilt, man schreibt dann $G' \subset G$. G' heißt knoten-induzierter (oder einfach induzierter) Subgraph von G wenn $V' \subset V$

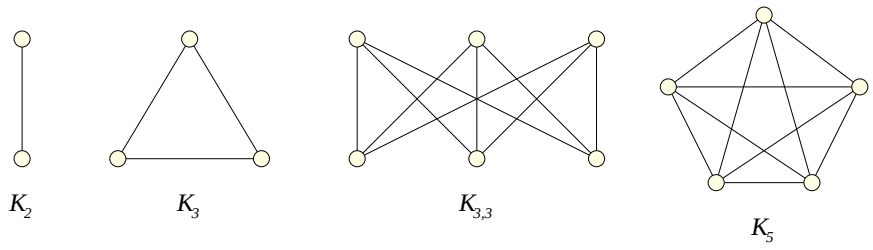

Abbildung 2.1: Einige Graphen

und $E' = \{(v_i, v_j) : v_i, v_j \in V' \wedge (v_i, v_j) \in E\}$.

Wenn später von einem Graphen V' die rede ist, dann ist der durch V' induzierte Subgraph gemeint. Eine nicht unbedingt übliche Abkürzung, die aber einige Schreibarbeit spart. Man wird im Zusammenhang immer erkennen ob von einer Knotenmenge oder von einem Graphen die Rede ist.

2.2 Wege, Pfade, Zykel, Bäume und Zusammenhang

Ein Weg in G ist eine Folge von disjunkten Kanten, wobei zwei Aufeinander folgende Kanten jeweils durch den gleichen Knoten verbunden, sind z.B. $(v_1, v_4), (v_4, v_5), (v_5, v_9), (v_9, v_4)$. Alternativ könnte man das auch durch eine Folge von Knoten und Kanten $v_1, (v_1, v_4), v_4, (v_4, v_5), v_5, (v_5, v_9), v_9, (v_9, v_4)$ oder nur durch eine Folge von Knoten v_1, v_4, v_5, v_9, v_4 notieren (bei letzterem wird implizit angenommen dass die Kanten zwischen aufeinander folgenden Knoten existieren). Falls auch die Knoten disjunkt sind spricht man von einem Pfad. Falls der Erste Knoten gleich dem Letzten ist, so spricht man von einem Zykel. Ein Zykel der Knoten und Kantendisjunkt ist heißt Kreis. Falls es zwischen zwei beliebigen Punkte eines Graphen einen Weg gibt, dann nennt man den Graphen zusammenhängend. Wenn ein zusammenhängender Graph durch die Wegnahme von einem Knote v_i in zwei oder mehrere Zusammenhangs-Komponenten zerfällt, dann nennt man v_i einen trennenden Knoten (engl. cut-vertex).

Falls ein Graph nach Herausnahme eines beliebigen Knotens zusammenhängend ist, dann nennt man ihn zweifach zusammenhängend. Ein zusammenhängender Graph, der keinen Kreis hat, heißt Baum.

2.3 Planare Graphen

Ein Graph der sich so in der Ebene zeichnen lässt, dass sich die Kanten nur in den Knoten berühren, zu denen sie inzident sind, heißt planar. Eine solche Zeichnung teilt die Ebene in zusammenhängende Gebiete, die als Facetten bezeichnet werden.

Die Anordnung der Kanten um ein Knoten charakterisiert eine planare Zeichnung, man spricht von einer planaren Einbettung.

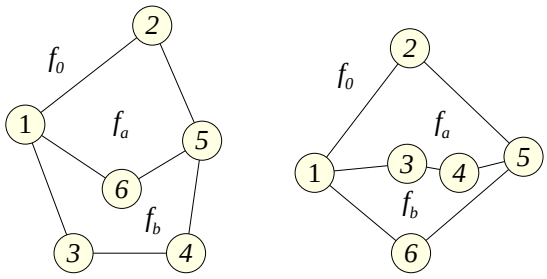

Abbildung 2.2: Planarer Graph mit zwei Einbettungen

Zu dem gleichen Graphen kann es mehrere planare Einbettungen geben, siehe Abbildung 2.2. Daher ist zwischen einem planaren Graphen und seiner Einbettung zu unterscheiden.

Ein nicht planarer Graph kann auf einer anderen Oberfläche trotzdem einbettbar sein, ohne dass sich die Kanten schneiden. Ein planarer Graph lässt sich aber immer auf einer Kugeloberfläche einbetten, und umgekehrt ist ein auf einer Kugeloberfläche ohne sich kreuzenden Kanten einbettbarer Graph auch planar. Die in einer planaren Zeichnung ausgezeichnete äußere Facette, meist mit f_0 bezeichnet, verliert auf der Kugeloberfläche ihre besondere Stellung.

Ein planarer Graph bei dem alle Knoten zu einer Facette inzident sind heißt outerplanar. Bei einer Zeichnung eines outerplanaren Graphen wird meist die ausgezeichnete Facette als äußere gewählt, daher rührt auch der Name outerplanar.

Nicht alle Graphen sind planar. Die wichtigsten beiden nicht planaren Graphen sind der $K_{3,3}$ und der K_5 siehe Abbildung 2.3.

Es ist recht intuitiv einzusehen, dass jeder Graph, der einen der beiden Graphen $K_{3,3}$ oder K_5 enthält selbst nicht planar sein kann. Enthält steht hier für eine folge von Operationen, bei der, ausgehend von einem Subgraph, zwei verbundene Knoten kontrahiert werden und die Verbindungskante weg fällt. Für genaueres sei wieder auf [1] verwiesen. Man sieht sehr schnell welche von

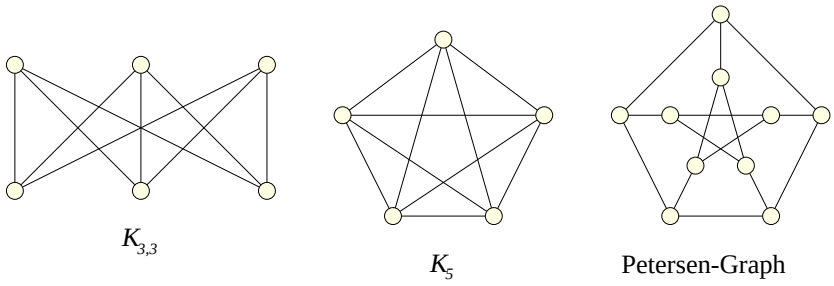

Abbildung 2.3: Nichtplanare Graphen

diesen Operationen auszuführen sind um zum Beispiel vom Petersen Graph zum K_5 zu kommen.

Erstaunlicher ist, dass sogar dir Umkehrung gilt: Ein Graph ist genau dann Planar, wenn er keinen Subgraph enthält der zum $K_{3,3}$ oder zum K_5 kontrahierbar ist. Dieser Satz wurde von Wagner 1937 gezeigt [5]. In der Literatur findet man häufiger eine andere, 7 Jahre ältere Form unter dem Namen *Satz von Kuratowski* [6].

Kapitel 3

LL-Zeichnung nach Biedl

3.1 LL-Zeichnungen - Einführung

Gegeben sei ein planarer Graph $G = (V, E)$ mit einer Partition $V = A \cup B$. Welche Bedingungen müssen gelten, damit sich dieser Graph so zeichnen lässt, dass alle Knoten aus A auf einer Linie und alle Knoten aus B auf einer dazu parallelen Linie gezeichnet und zudem alle Kanten geradlinig und kreuzungsfrei sind? Eine solche Zeichnung soll LL-Zeichnung heißen. Abbildung 3.1 zeigt ein Beispiel einer solchen Zeichnung.

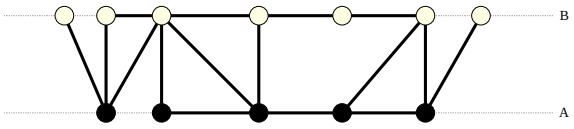

Abbildung 3.1: LL-Zeichnung

Therese Biedl hat in [3] hinreichende und notwendige Bedingungen aufgestellt, sowie einen Algorithmus aufgezeigt der eine solche Zeichnung in Linearzeit anfertigt. Hier folgt eine Beschreibungen der wesentlichen Ideen.

Wir bemerken, dass es nicht zu allen Partitionen planarer Graphen eine solche LL-Zeichnung gibt, Abbildung 3.2 zeigt ein Beispiel.

Wir stellen zunächst notwendige Bedingungen auf und zeigen dann, dass diese auch hinreichend sind.

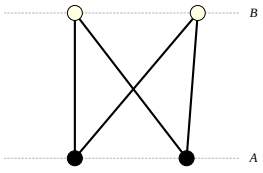

Abbildung 3.2: Beispielgraph, der keine LL-Zeichnung hat

3.2 Notwendige-Bedingungen

Sei ein partitionierter Graph $G = (A \cup B, E)$ mit einer LL-Zeichnung gegeben. Die Knoten der Partitionen seien dabei von links nach rechts durchnummeriert wie etwa in Abbildung 3.3.

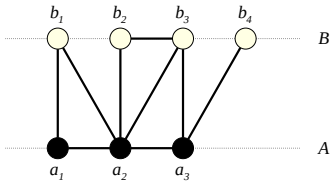

Abbildung 3.3: Graph G_{LL}

Nun werden dem Graphen zwei weitere Knoten v_a und v_b zugefügt, sowie Kanten zwischen v_a und allen Knoten aus A und zwischen v_b und allen Knoten aus B. Desweiteren die Kante (v_a, v_b). Der so erhaltene Graph soll G^+_{LL} heißen. Abbildung 3.4 zeigt ein Beispiel wobei v_a eine Position links und oberhalb der A-Knoten gezeichnet wurde und v_b entsprechend eine Position links und unterhalb aller B-Knoten.

Damit gelten folgende Eigenschaften der Zeichnung von G^+_{LL}: keine der zugefügten Kanten schneiden sich gegenseitig oder schon vorhandene Kanten. Die alle Kanten berühren sich weiterhin nur in den Knoten. Daraus können wir folgendes Lemma schließen:

Lemma 3.2.1
Eine planare LL-Zeichnung kann nur existieren, wenn folgendes gilt:

- G^+_{LL} *ist planar*
- *es gibt eine planare Einbettung von* G^+_{LL} *so, dass jedes Dreieck, dem* v_b *oder* v_a *angehört eine Facette ist.*

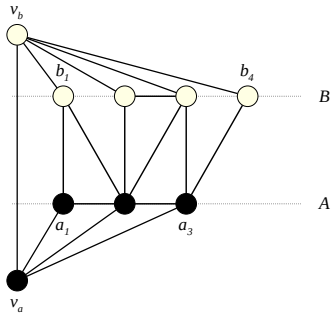

Abbildung 3.4: Graph G^+_{LL}

3.3 Hinreichende Bedingungen

Nun wollen wir zeigen, dass die Bedingungen von Korollar 3.2.1 auch hinreichend sind. Sei also ein planarer Graph G^+_{LL} mit einer Einbettung gegeben, so dass jedes Dreieck, das v_a oder v_b enthält eine Facette ist. Wir nummerieren zunächst die Knoten neu: jene um v_a gegen den Uhrzeigersinn und jene um v_b mit dem Uhrzeigersinn, jeweils beginnend mit bei der Kante (v_a, v_b) siehe auch Abbildung 3.5.

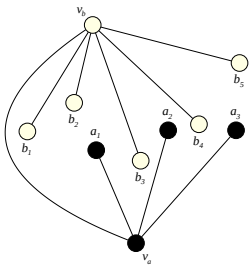

Abbildung 3.5: Graph G^+_{LL}

Wir zeigen folgendes Lemma:

Lemma 3.3.1
Wenn alle Knoten der Reihenfolge ihrer Nummerierung nach auf das gewünschte Raster gesetzt und alle Kanten geradlinig gezeichnet werden, erhalten wir eine planare LL-Zeichnung.

Es ist nun zu zeigen, dass sich keine zwei Kanten (a_i, b_k), (a_j, b_l) schneiden und dass es keinen Knoten a_j gibt der auf einer Kante zwischen (a_i, a_k)

liegt. Wir fügen nun weitere Kanten der Einbettung zu, erhalten dabei aber die Planarität der Zeichnung: Kanten zwischen allen (a_i, a_{i+1}) und analog zwischen (b_i, b_{i+1}) falls sie nicht schon vorhanden waren sowie die Kanten (a_1, b_1) und (a_{max}, b_{max}). Um die Planarität zu erhalten werden diese Kanten sehr nahe entlang schon vorhandenen gelegt, siehe Abbildung 3.6.

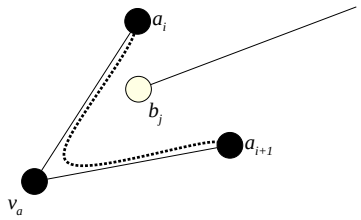

Abbildung 3.6: Hinzufügen von Kanten zu G^+_{LL}

Der komplette Graph G^{++}_{LL} auf dem Zweilinienraster sieht dann wie in Abbildung 3.7 aus. Die fetten, längsgestrichelten Kanten sind nur vorhanden wenn sie schon vor dem zufügen der weiteren Kanten vorhanden waren.

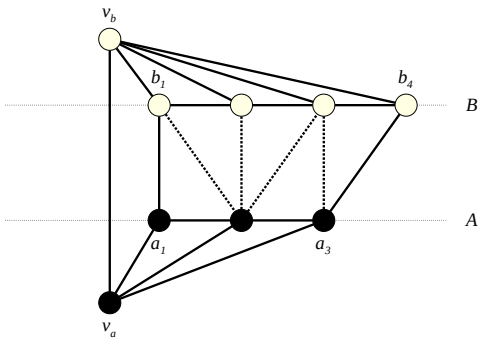

Abbildung 3.7: kompletter Graph G^{++}_{LL}

Wenn es zwei sich schneidende Kanten (a_i, b_k), (a_j, b_l) geben würde, so würde man in dem erweiterten Graph eine Unterteilung des nicht planaren Graphen $K_{3,3}$ finden, siehe Abbildung 3.8. Da aber nur Kanten unter Erhalt der Planarität zugefügt wurden ergibt sich eine Widerspruch zur vorausgesetzten Planarität.

Analog finden wir auch eine Unterteilung des $K_{3,3}$, wenn wir annehmen es gäbe eine lange Kante (a_i, a_j) mit $j > i + 1$. Dazu wird ein weiterer Knoten

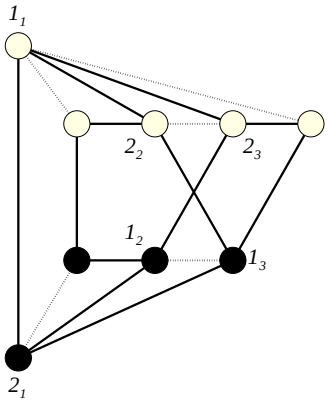

Abbildung 3.8: Unterteilung des $K_{3,3}$ in G_{LL}^{++} (fette ungestrichelte Kanten)

in das Dreieck v_a, v_i, v_j gesetzt und durch Kanten mit diesen Knoten verbunden. Da jedes solche Dreieck nach Voraussetzung eine Facette ist, bleibt die Planarität durch diese Operation erhalten. Abbildung 3.9 zeigt eine Unterteilung des $K_{3,3}$ in einem auf diese Art erzeugten Graph, wiederum ein Widerspruch zur vorausgesetzten Planarität.

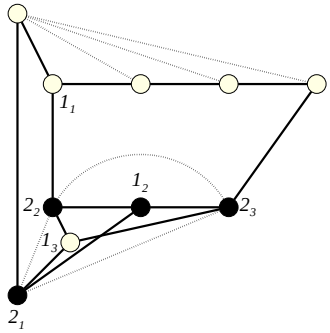

Abbildung 3.9: Unterteilung des $K_{3,3}$ in G_{LL}^{++}

3.4 Zeitverhalten

Ob ein Graph planar ist, kann in linearer Zeit bestimmt werden [7]. Zum Finden aller Dreiecke die v_a bzw. v_b enthalten, schaut man alle Kanten an und prüft ob es von den adjazenten Knoten Kanten zu v_a bzw. v_b gibt, ist also

linear zur Anzahl der Kanten m, was bei planaren Graphen wiederum linear zur Anzahl der Knoten ist. Bleibt noch zu testen, ob es eine Einbettung gibt in der die oben genannten Dreiecke Facetten sind. Dazu addiert man zu jedem Dreieck einen Knoten und von diesem Kanten zu allen Knoten des Dreiecks. Für diesen Graph berechnet man dann eine planare Einbettung, was in Linearzeit möglich ist, und entfernt anschließend wieder die zufügten Knoten. Genaueres und vor allem den Beweis der Korrektheit dieses Vorgehens findet man in [4].

Damit gilt folgender Satz:

Satz 3.4.1
Es gibt genau dann eine planare LL-Zeichnung zu einem partitionierten planaren Graphen $G = (V, E)$ mit $V = A \cup B$, wenn die Bedingungen von Lemma 3.2.1 erfüllt sind. Eine solche LL-Zeichnung kann in linearer Zeit berechnet werden.

Kapitel 4

LL-Bäume

Gegeben sein ein Baum B. Wann lässt sich dieser so zeichnen, dass all seine Knoten auf zwei parallelen Linien liegen und all seine Kanten geradlinig und kreuzungsfrei gezeichnet sind? Gibt es einen Algorithmus, der die Zeichnung, falls möglich, in linear-Zeit anfertigt?

Im folgenden werden wir diese Fragen positiv beantworten. In Kapitel 5 werden wir auf diese Ergebnisse zurückgreifen wenn wir allgemeine unpartitionierte LL-Zeichnungen behandeln.

Analog zum vorigen Kapitel wollen wir auch hier von LL-Zeichnungen (unpartitionierter Bäume) sprechen. Ein Baum, der diese Eigenschaft hat, soll LL-Baum heißen.

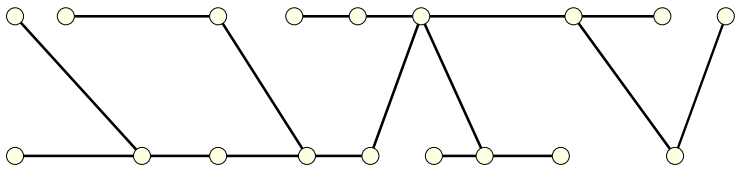

Abbildung 4.1: Beispiel eines LL-Baumes

4.1 Baum-Lemma

Die gesamte Theorie der LL-Bäume dreht sich um folgendes Lemma:

Lemma 4.1.1
Ein Baum hat eine LL-Zeichnung genau dann, wenn es einen einfachen Pfad P in B gibt, so dass gilt: die Zusammenhangskomponenten, gegeben durch $B \setminus P$, sind einfache Pfade.

Definition 4.1.2
Die Menge aller dieser Pfade wollen wir mit $\hat{P}_B$ bezeichnen. Also für alle $P_i \in \hat{P}_B$ sind die Zusammenhangskomponenten gegeben durch $B \setminus P_i$ einfache Pfade.

Definition 4.1.3
Die ersten und letzten Knoten auf den beiden Linien wollen wir als Eckknoten bezeichnen.

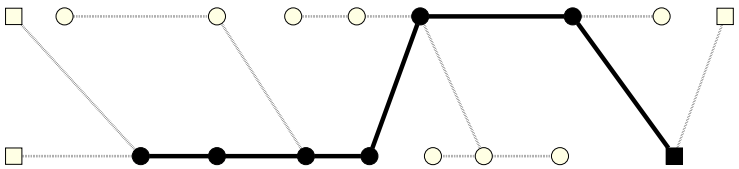

Abbildung 4.2: Pfad und Eckknoten

Beweis:
$\Rightarrow$

Sei eine Einbettung eines Baumes auf zwei Linien mit den genannten Eigenschaften gegeben. Wir können den Pfad P auf folgende Weise bestimmen: Wir wählen einen der Eckknoten als Anfangsknoten v_a aus, d.h. $P = \{v_a\}$. Solange es Komponenten $B \setminus P$ gibt die keine einfachen Pfade sind, wird P folgendermaßen erweitert: nehme jenen Knoten zu P hinzu welcher in der nicht-Pfad Komponente von $B \setminus P$ liegt und welcher zum letzten zu P hinzugefügten Knoten inzident ist. Mit Tiefensuche und einer geeigneten Datenstruktur lässt sich der Pfad P in linearer Zeit bestimmen.

Es ist noch zu zeigen, dass es dabei immer nur eine Komponente geben kann, die kein einfacher Pfad ist. Nehmen wir an es gäbe zwei davon. Da sie keine einfachen Pfade in $B \setminus P$ sind, haben sie Knoten auf beiden Linien. Da weiterhin v_a ein Eckknoten ist, müssen sich die beiden Komponenten schneiden oder berühren.

$\Leftarrow$

Gegeben sei ein Baum B mit einem Pfad P, so dass die Komponenten $B \setminus P$ einfache Pfade sind. Zeichne P auf eine Linie. Da B ein Baum ist, gibt es genau eine Verbindungskante zwischen P und den Pfad-Komponenten; diese lassen sich somit in der Reihenfolge wie sie an P angeschlossen sind auf die zweite Linie zeichnen und einfach mit P verbinden. Die Zeichnung lässt sich in linearer Zeit anfertigen. Eine solche Zeichnung kann dann wie Abbildung 4.3 aussehen.

□

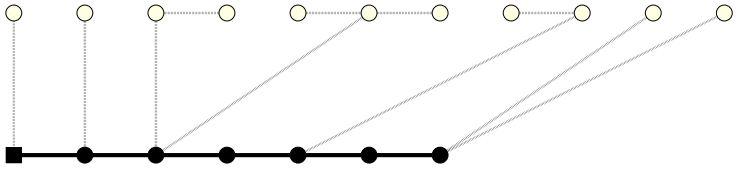

Abbildung 4.3: Gezeichnerter Graph

4.2 Zeitverhalten und Anfertigen der Zeichnung

Lemma 4.2.1
Gegeben sei ein Baum B. Es kann in linearer Zeit entschieden werden, ob es eine Einbettung auf zwei Linien gibt und falls ja, kann diese in linearer Zeit gezeichnet werden.

Nach der obigen Behauptung ist zu Entscheiden, ob es den Pfad P mit genannten Eigenschaften gibt, und diesen dann gegebenenfalls in Linear-Zeit zu berechnen.

Wir werden den Pfad P in drei Schritten berechnen. Im ersten Schritt werden wir einen Knoten v_m in B suchen zu dem es ein $P_i \in \hat{P}_B$ gibt, so dass gilt $v_m \in P_i$. Möglicherweise gibt es ein $P_i = \{v_m\}$ und wir sind fertig. Andernfalls bestimmen wir im nächsten Schritt einen zweiten zu v_m adjazenten Knoten v_n der ebenfalls in P_i liegen muss. Die Kante (v_n, v_m) teilt B in zwei Teilbäume. Im letzten Schritt bestimmen wir von v_n respektive v_m den Teilpfad P_n bzw. P_m im jeweiligen Teilbaum. Die Vereinigung der beiden Pfade liefert uns dann ein $P \in \hat{P}_B$.

Schritt 1 Der Pseudocode von Schritt 1 ist unter Algorithmus 1 angegeben. Wir starten bei einem beliebigen Blatt $v_b \in B$ und suchen von dort aus den ersten Knoten mit Grad größer oder gleich drei. Diesen bezeichnen wir mit v_s. Falls es keinen Solchen Knoten gibt, ist B schon ein einfacher Pfad und P kann, wie im Algorithmus, gleich dem zu dem Startblatt adjazenten Knoten gesetzt werden. Falls der Grad von v_s gleich vier oder größer ist wird, wie in Abbildung 4.4, $v_m = v_s$ gesetzt . Bleibt noch der Fall, dass $d(v_s) = 3$. Falls $P = \{v_s\} \in \hat{P}_B$ sind wir wiederum fertig. Ansonsten suchen wir eine zu v_s inzidente Komponente die in $B \setminus \{v_s\}$ kein einfacher Pfad ist. Der zu

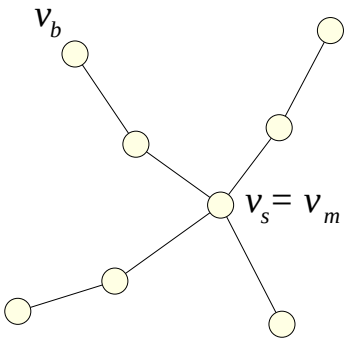

Abbildung 4.4:

v_s inzidente Knoten dieser Komponente bezeichnen wir mit v_m und setzen $P = \{v_m\}$, siehe auch Abbildung 4.5.

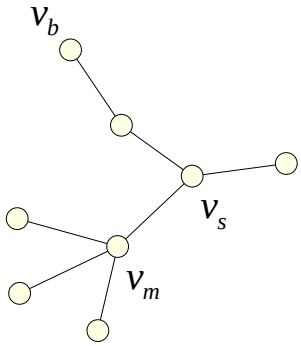

Abbildung 4.5:

Beweis: (Korrektheit von Schritt 1) Zu zeigen ist, wenn $\hat{P}_B$ nicht leer ist, dann gibt es einen Pfad $P \in \hat{P}_B$ für den gilt $v_m \in P$. Falls $d(v_s) = 1$, besteht B selbst nur aus einem einfachen Pfad. In $\hat{P}_B$ liegen dann unter anderem alle $\{v\}$ für $v \in V$, also insbesondere auch der letzte Knoten v_m des Pfades. Falls $d(v_s) = 4$, dann muss v_s jedem P_i angehören, sonst wären $B \setminus P_i$ keine einfachen Pfade oder P_i kein zusammenhängender Pfad. Falls $d(v_s) = 3$ und $P = \{v_s\} \notin \hat{P}_B$, dann gibt es einen an v_s angeschlossenen Teilbaum, der kein einfacher Pfad ist. Der zu v_s adjazente Knoten v_m dieses Baumes muss dann in allen $P_i \in \hat{P}_B$ liegen¸ sonst wäre die Zusammenhangs-Komponente inklusive v_m in Richtung v_s schon kein einfacher Pfad. □

Schritt 2 Wenn nach dem ersten Schritt nun noch nicht $\{v_m\} \in \hat{P}_B$ gilt, dann suchen wir im zweiten Schritt (siehe Algorithmus 2) einen zweiten Knoten $\{v_n\}$, der ebenfalls wie v_m in P liegt. Dazu untersuchen wir einfach die Zusammenhangskomponenten $B \setminus \{v_m\}$. Sobald wir eine gefunden haben die kein einfacher Pfad ist setzen wir v_n gleich dem Knoten in dieser Komponente, der zu v_m adjazent ist. Abbildung 4.6 zeigt ein Beispiel, in dem zufällig $v_m = v_s$ ist.

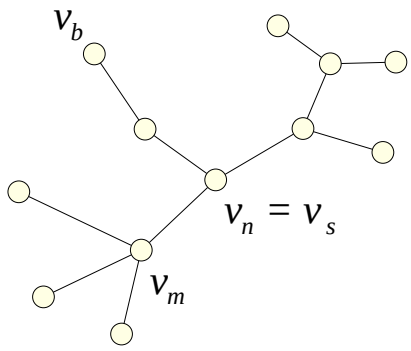

Abbildung 4.6:

Beweis: (Korrektheit von Schritt zwei) Zu zeigen ist, dass $v_n \in P_i$ falls $v_m \in P_i$. Da die Komponente in der v_n liegt kein einfacher Pfad ist muss mindestens ein Knoten aus ihr zu P_i gehören, da P_i ja ein Pfad ist und $p_m \in P_i$, dann muss auch der adjazente Knoten zu v_m in P_i liegen. □

Schritt 3 Zur Vorbereitung zu Schritt drei teilen wir den Baum B an der Kante (v_m, v_n) in zwei Teilbäume B_m respektive B_n. Für jeden Teilbaum berechnen wir dann den Pfad $P_n \in \hat{P}_{B_n}$ z.B. für B_n, für den v_n der erste Knoten von P_n ist. Der Algorithmus ist in 3 festgehalten, ähnlich wir in Algorithmus 2 wird der nächste Knoten durch die Restkomponente bestimmt, die kein einfacher Pfad ist. Falls es einmal zwei solcher Restkomponenten geben sollte, führt das zum Abbruch. Es gibt dann kein P mit den gewünschten Eigenschaften. Im positiven Fall gilt aber $P = P_n \cup P_m \in \hat{P}_B$.

Beweis: (Korrektheit von Schritt drei) Zu zeigen ist, dass P_n die gewünschte Eigenschaften hat, d.h. Zusammenhangs-Komponenten $B_n \setminus P_n$ sind einfache Pfade. Das ist unmittelbar aus dem Algorithmus ersichtlich. □

Zeitverhalten Mit der richtigen Datenstruktur lassen sich die Algorithmen in Linearzeit durchführen. Vor Algorithmus 1 führt man eine Tiefen-

suche mit v_b als Wurzel durch. Dabei merkt man sich in einer geeigneten Datenstruktur wie die oder der Teilbaum an jeder Kante unterhalb jedes Knotens $v \in V$ aussieht (i.e. ist es ein einfacher Pfad oder nicht, wie viele Kanten sind unterhalb eines Knotens angeschlossen). Ein solcher Vordurchlauf mit Tiefen-suche benötigt $O(n)$ Zeit. Danach benötigt Algorithmus 1 auch nur lineare Zeit, da immer nur Informationen über die jeweils unter einem aktuellen Knoten liegende Teilbäume abgerufen werden müssen. Auch für Algorithmus 2 führt man wiederum einen Vordurchlauf mit v_m als Wurzel durch den Baum durch. Da v_n in diesem Baum unterhalb von v_m liegt, benötigt Algorithmus 3 keinen eigenen Vordurchlauf.

Mit einer kleinen Änderung von Algorithmus 2 kann der zweite Vordurchlauf entfallen. Dazu muss in der Schleife über die inzidenten Knoten von v_m der Knoten v_s als erstes überprüft werden. Es gibt dann den Fall, dass $v_n = v_s$ gesetzt wird. Nach dem ersten Vordurchlauf mit v_b als Wurzel wissen wir, dass über v_n nur der einfache Pfad $v_b, ..., v_n$ liegt. Der Teilbaum mit v_m als Wurzel liegt unterhalb von v_n und ist auch schon berechnet (siehe Abbildung 4.7). Im zweiten Fall, $v_n \neq v_s$, liegt über v_m eine Komponente die in $B \setminus \{v_m\}$

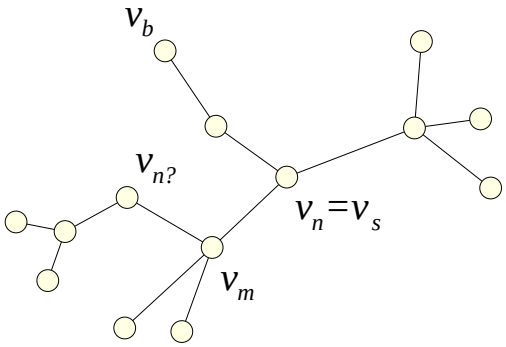

Abbildung 4.7:

ein einfacher Pfad ist (siehe Abbildung 4.8). Der Teilbaum unterhalb v_n liegt dann innerhalb des Teilbaums von v_m und wurde ebenfalls schon berechnet.

Eingabe : Baum B, Blatt v_b
Ausgabe: ein Knoten der P angehört
Initialisierung: v_c = inzidenter Knoten zu v_b, $v_o = v_b$;
$P = \emptyset$;
while $d(v_c) = 2$ **do**
  $v_d = v \in V$ so dass $\exists (v, v_c) \in E, v \neq v_o$;
  $v_o = v_c$;
  $v_c = v_d$;
end
$v_s = v_c$;
if $d(v_s) = 1$ **then** $v_m = v_s$;
if $d(v_s) \geq 4$ **then**
  $v_m = v_s$;
end
if $d(v_s) = 3$ **then**
  if $\{K \subset B \setminus \{v_c\} : K$ *zusammenhängend, K kein einfacher Pfad* $\} = \emptyset$ **then**
    $v_m = v_c$;
  end
  else
    $v_s = v_c$;
    foreach v_c *inzident to* v_s **do**
      if $K \subset B \setminus \{v_s\}$, $v_c \in K$,K *zusammenhängend, K kein einfacher Pfad* **then**
        $v_m = v_c$;
        EXIT ;
      end
    end
  end
end
RETURN v_m;

Algorithmus 1: Schritt 1, berechne v_m

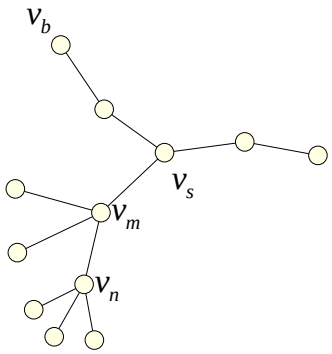

Abbildung 4.8:

Eingabe : Baum B, Knoten v_m in P
Ausgabe: zweiter Knoten v_n in P oder -1 Falls es diesen nicht gibt
foreach $v_c \in V$*:* v_c *inzident* v_m **do**
  if $K \in B \setminus \{v_m\}, K$ *zusammenhängend,* $v_c \in K$*,* K *ist kein einfacher Pfad* **then**
    $v_n = v_c$;
    RETURN v_n;
  end
end
EXIT -1;

Algorithmus 2: Schritt 2, berechne v_n

Eingabe : Baum B, Knoten $v_a \in P$

Ausgabe: Pfad $P \in B$ so dass Zusammenhangs-Komponenten $B \setminus P$ einfache Pfade sind, wobei v_a der erste Knoten in P ist; -1 falls es diesen nicht gibt

Initialisierung: $v_c = v_a$, $P = \{v_a\}$;
while $\{K \subset B \setminus P : K$ *zusammenhängend,* K *inzident* v_c*,* K *kein einfacher Pfad* $\} \neq \emptyset$ **do**
    $V_n = \emptyset$;
    foreach $v_i \in V : \exists (v_i, v_c) \in E; v_i \notin P$ **do**
        if $K \subset B \setminus \{v_c\}$*,* K *zusammenhängend,* $v_i \in K$*,* K *kein einfacher Pfad* **then** $V_n = V_n \cup \{v_i\}$;
    end
    if $|V_n| > 1$ **then** EXIT -1;
    else
        $v_c = v_i \in V_n$;
        $P = P \cup \{v_c\}$;
    end
end
RETURN P;

Algorithmus 3: Schritt 3, berechne P von v_a

Kapitel 5

Allgemeine unpartitionierte LL-Zeichnungen

In diesem Kapitel wollen wir folgender Fragestellung nachgehen: Wann ist es möglich einen planaren Graphen $G = (V, E)$ so zu Zeichnen, dass alle Knoten auf zwei Linien liegen und alle Kanten geradlinig gezeichnet werden. Abbildung 5.1 zeigt eine solche Zeichnung.

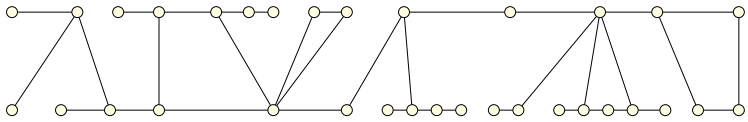

Abbildung 5.1: Beispiel

Nach einigen Definitionen und weiteren Vorbereitungen in 5.1 und 5.2 werden wir in 5.3 das eigentliche Thema behandeln. Wir werden dabei den gegebenen Graphen in verschiedene Teilgraphen aufspalten, so dass sich diese gesondert behandeln lassen. Die in Kapitel 4 behandelten LL-Bäume können zum Beispiel solche Teilgraphen sein.

5.1 Anschlussknoten

Gegeben sei ein outerplanarer Graph $G = (V, E)$. Wir suchen darin jene Knoten, die zweifache Zusammenhangskomponenten von einfachen Zusammenhangskomponenten trennen.

Definition 5.1.1
Unter einer zweifachen Zusammenhangskomponente des Graphen $G = (V, E)$, verstehen wir einen Subgraph $G_S \subset G$ der mindestens drei Knoten hat, $|V_S| \geq 3$, und der nach Wegnahme eines beliebigen Knotens immer noch zusammenhängend ist. Meist sind zweifache Zusammenhangskomponenten in der Literatur etwas allgemeiner definiert. Die obige Version ist für unsere Zwecke aber günstiger.

Definition 5.1.2
Unter einem Anschlussknoten v_a eines einfachen, zusammenhängenden Graphen $G = (V, E)$ verstehen wir trennende Knoten (Artikulationsknoten), die einer zweifachen Zusammenhangskomponente angehören. Die Menge aller Anschlussknoten eins Graphen wollen wir mit V_A bezeichnen. $V_A = \{v_a \in V : v_a$ Artikulationsknoten, $\exists$ zweifache Zusammenhangskomponente G_S mit $v_a \in G_S\}$.

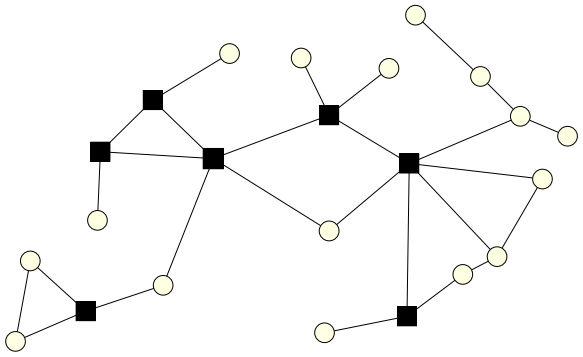

Abbildung 5.2: Beispiel Anschlussknoten V_A

Bestimmen der Anschlussknoten in $O(n)$

Wir gehen von einer planaren Einbettung aus, bei der alle Knoten zu der äußeren Facette inzident sind. Beginnend mit einer beliebigen Kante laufen

wir einmal entlang der äußeren Facette um den Graph herum. Wir erreichen dabei jeden Knoten mindestens einmal.

Falls ein Knoten trennend ist, erreichen wir ihn nach Umlauf einer Komponente, die er vom Rest-Graph trennt, ein zweites mal. Um alle trennenden Knoten zu finden müssen wir beim einmaligem Umlauf nur vermerken wie oft jeder Knoten erreicht wurde. (Diese Zahl gibt an in wie viele Komponenten der Graph ohne diesen Knoten zerfällt).

Ähnliches gilt auch für die Kanten. Trennende Kanten, ohne die der Graph nicht mehr zusammenhängend ist, werden Brücken genannt. Kanten die nicht zur äußeren Facette inzident sind können keine Brücken sein, da der Graph ja outerplanar eingebettet ist. Beim Umlauf um die äußere Facette werden genau die Kanten zweimal durchlaufen, die den Graphen trennen. Das bedeutet, alle Kanten, die nur einmal durchlaufen werden, gehören einer zweifachen Zusammenhangskomponente an. Um die Knoten zu bestimmen, die zu einer zweifachen Zusammenhangskomponente gehören, müssen wir somit nur überprüfen, ob es adjazente Kanten gibt, die beim Umlaufen nur in eine Richtung passiert wurden.

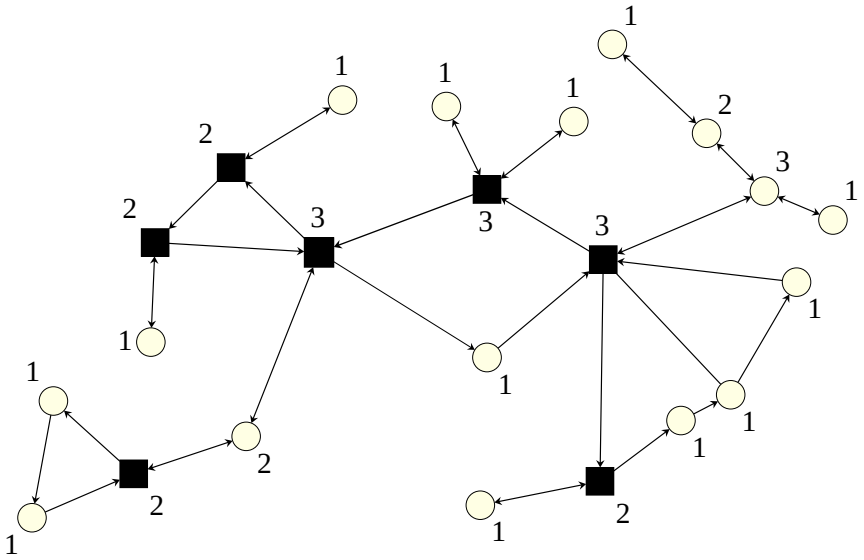

Abbildung 5.3:

Abbildung 5.3 zeigt den Graphen aus dem Beispiel, nachdem er einmal gegen den Uhrzeigersinn umlaufen wurde. Wie oft ein Knoten dabei erreicht wurde ist mit der markierten Nummer festgehalten. In beide Richtungen

durchlaufene Kanten haben Pfeile an beiden Enden der Kante , einfach durchlaufene nur einen, passend zur Umlauf-Richtung.

Lemma 5.1.3
Die Menge der Anschlussknoten V_A eines Graphen mit outerplanarer Einbettung kann in Linearzeit bestimmt werden.

Beweis: Die Korrektheit ist aus dem schon geschriebenen ersichtlich. Der Zeitaufwand für den Umlauf liegt in der Größenordnung der Anzahl der Kanten, diese ist linear zur Anzahl der Knoten da der Graph planar ist. Danach muss zunächst für jeden Knoten nachgeschaut werden wie oft er erreicht wurde. Dies liegt in $O(n)$. Zu überprüfen ob ein Knoten inzident zu einer zweifach durchlaufenen Kante ist liegt in der Größe der adjazenten Kanten des Knotens. Man überlegt sich aber, dass beim Nachschauen aller Knoten jede Kante dabei höchstens zweimal aufgerufen wird. Da es linear viele Kanten gibt ist dieser Aufwand somit auch linear. □

5.2 Einseitige Komponenten

Gegeben sei ein outerplanarer Graph G mit einer Menge von Anschlussknoten $A \subset V$.

Wir sind hier an jenen Teilmengen von G interessiert, die adjazent zu einem Knoten aus $v_a \in A$ sind und die im Graph ohne diesen Knoten einen einfachen Pfad bilden. Das entspricht: gesucht sind alle Komponenten aus $G \setminus \{v_a\} : v_a \in A$ die einfache Pfade sind. Wir wollen die Menge dieser Komponenten *Einseitige Komponenten* nennen und sie mit K^* bezeichnen und die durch v_a implizierten Elemente daraus mit $k_{v_a i}$. Möglicherweise wird es Untermengen in K^* geben die sich beinhalten, K sei dann jene dir nur disjunkte Elemente enthält, welche aber maximal sind.

Abbildung 5.4 zeigt ein Beispiel eines Graphen in dem die Knoten der einseitigen Komponenten schwarz markiert sind.

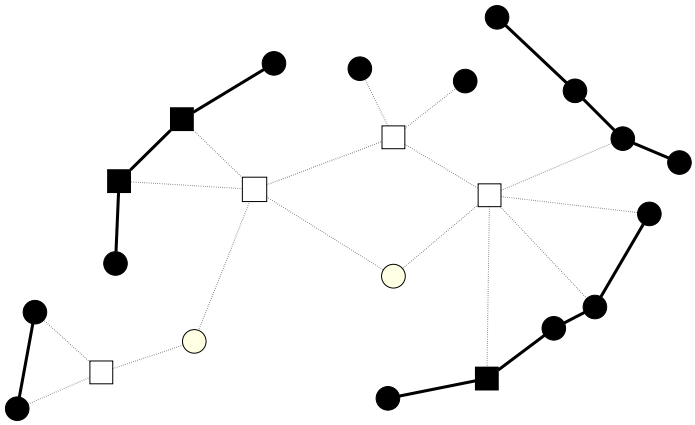

Abbildung 5.4: Einseitige Komponenten

Finden der Komponenten in $O(n^2)$

Eine einfache Möglichkeit diese Komponenten zu bestimmen ist, für alle $v_i \in V_A$ alle Zusammenhangskomponenten, gegeben durch $G \setminus \{v_i\}$, auf Pfadeigenschaft zu testen. Dazu beginnen wir z.B. bei einem beliebigen Knoten $v \in V \setminus \{v_i\}$. Da V_A linear zu V ist und für jeden $v_i \in V_a$ einmal eine Tiefensuche über den gesamten Graphen (ohne v_i) gemacht, wird liegt die Gesamtlaufzeit in $O(n^2)$.

Finden der Komponenten in $O(n)$

Um die Komponenten in linearer Zeit zu bestimmen, nutzen wir die Outerplanarität des Graphen und folgendes Lemma:

Lemma 5.2.1
In jeder Einseitigen Komponente gibt es genau keinen, einen oder zwei Anschlussknoten.

Die Korrektheit dieser Aussagen überlegt man sich leicht. Abbildung 5.5 zeigt Beispiele der drei Klassen von Komponenten, die es gibt: mit keinem, einem und zwei Anschlussknoten. Der untere Knoten ist, jener der die Komponente mit dem Restgraph verbindet.

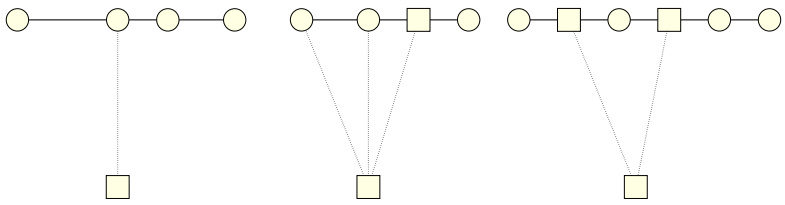

Abbildung 5.5: Einseitige Komponenten

Wir gehen nun von einer outerplanaren Einbettung aus und laufen einmal um den Graph, entlang der äußeren Facette herum. In einer Liste vermerken wir welche Knoten erreicht wurden.

Abbildung 5.6 zeigt unseren Graphen noch einmal mit nummerierten Knoten. Die Liste würde beginnend mit der Kante $(1, 3)$, gegen den Uhrzeigersinn umlaufen, wie folgt aussehen: {**3**, **20**, 22, **20**, 21, 19, 14, **3**, 15, 17, 15, 16, 18, 16, 15, **3**, **2**, 12, **2**, 13, **2**, **0**, **4**, 11, **4**, **5**, 5, 10, **5**, **0**, 6, **7**, 8, 9, **7**, 6, **0**, 1}. Die Anschlussknoten sind hervorgehoben. Im Folgenden werden wir Überlegungen zu dem Graph anstellen, die mit Bedingungen der Liste verknüpft sind. Da die Liste in Wirklichkeit einem Zykel entspricht (die äußere Facette), muss beim erreichen des Endes der Liste am Anfang fortgefahren werden. Bis zu welchem Eintrag in der Liste dann gegangen werden muss, hängt von der Art der Bedingung ab die Überprüft wird. Eine einfache Möglichkeit dieses Problem zu umgehen, besteht darin, die Liste zu verdoppeln. Das heißt am Ende einfach nochmal die gesamte Liste anhängen. Im allgemeinen wird sich dadurch die Laufzeit verdoppeln, was für uns keine Rolle spielen wird. Von solch einer modifizierten Liste wollen wir im Folgenden ausgehen.

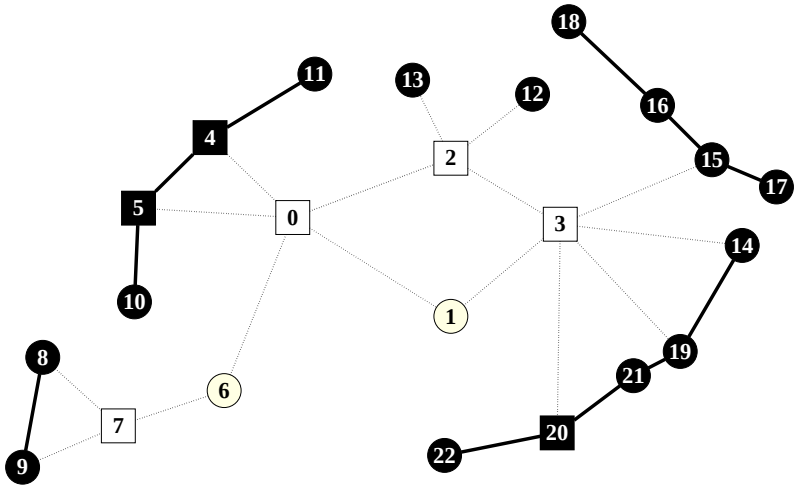

Abbildung 5.6: Liste

Einseitige Komponenten, können nun nur solche sein, die in der Liste zwischen dem gleichen Anschlussknoten notiert sind. Mit dem obigen Lemma lässt sich eine verschärfte Bedingung erstellen:

Korollar 5.2.2
Es können nur jene Komponenten einseitig sein, deren Knoten der Liste zwischen dem gleichen Anschlussknoten notiert sind und in denen maximal zwei weitere Anschlussknoten vorkommen.

Durch kurze Überlegung lässt sich auch diese Bedingung noch verschärfen:

Lemma 5.2.3
Wenn in der Teilliste, die einer Komponente entspricht alle Knoten bis auf die Anschlussknoten gestrichen werden, dann besteht die Teilliste noch genau aus Null, zwei oder vier (Anschluss-) Knoten, welche immer Paarweise aufeinander folgend vorkommen.

Mit dem genannten Korollar und Lemma können wird nun anhand der Liste einfach überprüfen welche Komponenten möglicherweise einseitig sind. Dazu laufen wir die Liste entlang und merken uns immer die letzten sechs Anschlussknoten, die wir passiert haben. Bei jedem Anschlussknoten v_a den wir passieren, schauen wir, ob v_a schon unter den letzten fünf passierten Anschlussknoten war und ob dazwischen Paare von keinem, einem oder zwei anderen Anschlussknoten stehen. Falls ja, dann überprüfen wir, die Komponente bestehend aus den Knoten,die zwischen v_a stehen, auf Pfadeigenschaft

in $G \setminus \{v_a\}$. Das Verfahren ist in Algorithmus 4 nochmals explizit zur Verdeutlichung notiert.

Als Beispiel gehen wir wieder von der obigen Liste aus. Wir würden uns zunächst die Anschlussknoten 3 und 20 merken. Der nächste Anschlussknoten wäre wieder 20 und die Komponente bestehend aus dem Knoten 22 würde auf Pfadeigenschaft in $G \setminus \{22\}$ überprüft werden. Als nächster Anschlussknoten kommt wieder 3 in der Liste vor und die Komponente aus den Knoten 20, 22, 21, 19 und 14 würde in $G \setminus \{3\}$ auf Pfadeigenschaft überprüft werden.

Beweis Korrektheit

Geht aus den Lemmas bzw. Korrolars hervor.

Laufzeit

Das Anfertigen der Liste ist linear zur Anzahl der Kanten der äußeren Facette. Da der Graph planar ist gibt es auch nur $O(n)$ viele Kanten. Das Finden der potentiell einseitigen Komponenten ist linear zur Größe der Liste. Für jeden neuen Anschlussknoten werden nur die sechs letzten angeschaut. Bleibt noch zu überlegen, dass das Finden zusammen mit dem Überprüfen ob eine Komponente tatsächlich ein einfacher Pfad ist auch linear ist. Ist so weil zwischen dem gleichen Anschlussknoten immer eine Komponenten steht. Und weil maximal zwei innere Komponenten überprüft werden.

Eingabe : doppelte Knotenliste L_V
Ausgabe: Einseitige Komponenten
Datenstruktur: Stack von Knotenzeiger $A[i]$ mit mit Zugriff auf die letzten 6 Elemente $i \in \{1, .., 6\}$;
Initialisierung: $K^* = \emptyset$;
foreach $v_i \in L_V$ **do**
  if $v_i \in V_A$ **then**
    push$(\&v_i, A)$;
    if $*A[1] == *A[2]$ **then**
      if *die Menge* $\{L_V[A[1]+1], .., L_V[A[2]-1]\}$ *induziert Pfad* **then**
        push $(\{L_V[A[1]+1], .., L_V[A[4]-1]\}, K^*)$;
      end
    end
    if $*A[1] == *A[4]$ and $*A[2] == *A[3]$ **then**
      if *die Menge* $\{L_V[A[1]+1], .., L_V[A[4]-1]\}$ *induziert Pfad* **then**
        push $(\{L_V[A[1]+1], .., L_V[A[4]-1]\}, K^*)$;
      end
    end
    if $*A[1] == *A[6]$ and $*A[2] == *A[3]$ and $*A[4] == *A[5]$ **then**
      if *die Menge* $\{L_V[A[1]+1], .., L_V[A[6]-1]\}$ *induziert Pfad* **then**
        push $(\{L_V[A[1]+1], .., L_V[A[6]-1]\}, K^*)$;
      end
    end
  end
end
RETURN K^*;

Algorithmus 4: Berechne einseitige Komponenten

5.3 Unpartitionierte LL-Zeichnungen

Nach den Vorbereitungen beginnen wir nun mit den eigentlichen unpartitionierten LL-Zeichnungen. Zunächst nochmal die Problemstellung: Gegeben sei ein planarer Graph $G = (V, E)$. Wann ist es möglich diesen auf zwei Linien zu zeichnen, so dass alle Knoten auf einem Raster auf den Linien liegen und alle Kanten geradlinig und kreuzungsfrei gezeichnet werden?

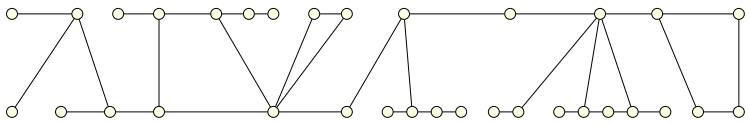

Abbildung 5.7: Beispiel

Im folgenden wollen wir uns notwendige und hinreichende Bedingungen dazu überlegen sowie eine Möglichkeit aufzeigen eine solche Zeichnung in Linearzeit zu Bestimmen.

Vorbereitung und Definitionen

Um notwendige Kriterien angeben zu können müssen wir zunächst einige Vorarbeit leisten.

Anschlussknoten und zweifache Zusammenhangskomponenten Zunächst bestimmen wir die Menge der Anschlussknoten $V_A \subset V$ und die zweifachen Zusammenhangskomponenten von G. Wie sich das in linearer Zeit realisieren lässt haben wir schon in Kapitel 5.1 gesehen.

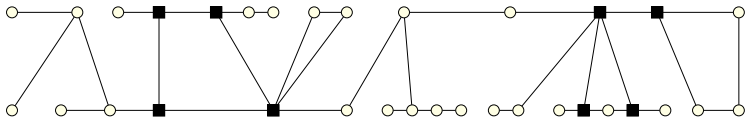

Abbildung 5.8: Anschlussknoten

Einseitige Komponenten Im nächsten Schritt bestimmen wir alle jene Teilkomponenten getrennt von einem (!) Anschlussknoten zum Restgraph, die sich auf einer Linie zeichnen lassen. Dies ist ebenfalls wie in Kapitel 5.2 beschrieben in Linearzeit möglich. Mit G_{EK} wird die Menge dieser Einseitigen

Komponenten bezeichnet, V_{EK} soll dabei die korrespondierende Knotenmenge sein.

Wir werde noch eine kleine Verfeinerung brauchen, nämlich genau jene einseitigen Komponenten, die zusammen mit ihrem Anschlussknoten einfache Pfade sind. Diese lassen sich aus V_{EK} recht einfach bestimmen, etwa durch testen ob der durch $V_{EK_i} \cup \{a_{EK_i}\}$ induzierte Subgraph ein einfacher Pfad ist. Hierbei ist V_{EK_i} eine Komponente aus V_{EK} und a_{EK_i} der adjazente Anschlussknoten. Die Pfadkomponenten wollen wir V_{EKP} und die nicht Pfadkomponenten V_{EKN} nennen. Somit gilt $V_{EK} = V_{EKN} \cup V_{EKP}$.

Zweifache Zusammenhangskomponenten Im weiteren wollen wir nur solche zweifache Zusammenhangskomponenten betrachten, die keine Knoten haben die in einer der Einseitigen Komponenten angehört. Die dadurch gegebene Knoten-Menge wollen wir mit $V_Z \subset V$ bezeichnen.

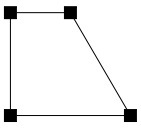

Abbildung 5.9: zweifache Zusammenhangskomponente induziert aus $V_{Z_i} \subset V_Z$

Bäume Sei $E_B = E \setminus (E_{EK} \cup E_Z \cup \{(u,v) : u \in V_{EK}, v \in V_A\})$ und $V_B = \{v \in V : \exists e \in E_B : v \text{ inzident zu } e\}$ also jene Knoten die zu irgend einer Kante in E_B inzident sind. Die Zusammenhangskomponenten in G_B sind dann Bäume. Darunter kann es welche geben, die einen Anschlussknoten nicht als Blatt enthalten. Teilbäume darin sollen maximale Zusammenhangskomponenten sein, bei denen ein Anschlussknoten aber immer nur als Blatt vorkommt.

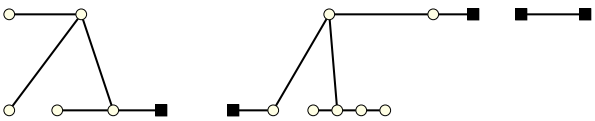

Abbildung 5.10: Teilbäume des Graphen

Definition 5.3.1 (Beidlinige Komponenten)
Die Menge der zweifachen Zusammenhangskomponenten zusammen mit den Teilbäumen wollen wir beidlinige Komponenten nennen: $V_{BK} = V_Z \cup V_B$.

Definition 5.3.2 (Verbindungsknoten)
Ein Anschlussknoten heißt Verbindungsknoten, wenn er zu mehr als einer beidlinigen Komponente angehört.

5.3.1 Notwendige Bedingungen

Sei ein zusammenhängender Graph $G = (V, E)$ mit einer geradlinigen zwei Linien-Zeichnung Gegeben. Dann gelten folgende Lemmata und Korollare.

Lemma 5.3.3 (Outerplanarität)
Jeder auf zwei Linien, kreuzungsfrei und gradlinig gezeichnete Graph ist outerplanar.

Beweis: Da alle Knoten auf den zwei Linien gezeichnet sind und keine Kante oberhalb der oberen Linie und unterhalb der unteren Linie gezeichnet sind, sind alle Knoten zu der äußeren Facette inzident. □

Lemma 5.3.4 (lineare Anordnung)
Wenn wir den Graph ohne die Knoten der einseitigen Komponenten betrachten, also den Graph induziert durch $V \setminus V_{EK}$, dann gilt

- *der Restgraph ist zusammenhängend*
- *es gibt eine lineare Anordnung (Pfad von Komponenten) in dem Restgraph, in der zweifache Zusammenhangskomponenten und Teilbäume aufeinander folgend vorkommen.*

Korollar 5.3.5
Aus Lemma 5.3.4 folgt, dass eine beidlinige Komponente maximal an zwei weitere beidlinige Komponenten angeschlossen ist, (diese sind in der Zeichnung links bzw. rechts von dieser zu finden).

Da wir den Teilbaum auf zwei Linien zeichnen wollen folgt außerdem aus Lemma 5.3.4:

Korollar 5.3.6 (Bäume)
1. *jeder Teilbaum hat maximal zwei Anschlussknoten*
2. *es gibt einen Pfad P in jedem Baum mit folgenden Eigenschaften:*
   (a) *die Komponenten $B \setminus P$ sind einfache Pfade.*

(b) alle Anschlussknoten sind am Anfang bzw. Ende dieses Pfades.

Lemma 5.3.7 (zweifache Zusammenhangskomponenten)
Es gilt:

1. *es gibt maximal vier Anschlussknoten*
2. *wenn die Komponente an zwei weitere beidlinige Komponenten angeschlossen ist, dann ist sie das in zwei verschiedenen Anschlussknoten*
3. *für die Zeichnung gilt:*
   (a) *jeder Anschlussknoten ist ein äußerster Knoten dieser Komponenten auf einer Linie*
   (b) *für jede Seite (d.h. für die Anschlussknoten auf der linken / rechten Seite) gilt*
      i. *es ist maximal eine zweifache Zusammenhangskomponente oder maximal ein Teilbaum zugeordnet*
      ii. *wenn einem der Anschlussknoten eine zweifache Zusammenhangskomponente, ein Teilbaum und oder mehrere einseitige Komponenten aus G_{EKN} zugeordnet sind, dann ist dem zweiten Anschlussknoten höchstens eine Komponente aus G_{EKP} zugeordnet.*

Beweis:

1. Die Anschlussknoten müssen auf den vier Ecken der Komponente liegen.
2. Annahme die Komponente G_{BK} wäre mit in einem Anschlussknoten a_k an zwei weitere Beidlinige Angeschlossen. Da die Komponenten selbst auf beiden Linien Knoten haben oder an solche angeschlossen sind (bestimmte 'Arten' von Bäumen) hat G_{BK} außer a_k alle Knoten auf der anderen Linie als a_k, dann wäre G_{BK} aber eine einseitige Komponente oder zerfällt in mehrere einseitige Komponenten.
3. Es würde sonst Überkreuzungen von Kanten geben.

□

5.3.2 Zeichnen einer zweifachen Zusammenhangskomponente

Um eine Zeichnung zu bestimmen wird folgendes Lemma dienlich sein:

Lemma 5.3.8 (Eindeutigkeit der Einbettung)
Die Einbettung einer outerplanaren, zweifachen Zusammenhangskomponente ist bis auf Inversion festgelegt, d.h. die Folge der Knoten um die äußere Facette ist festgelegt. Diese Folge lässt sich in linearer Zeit bestimmen.

Beweis: Seien V_{ZK_i} die Knoten der Komponente und G_{ZK}, der induzierte davon Subgraph. Es wird ein Knoten v_x und Kanten von ihm zu allen Knoten $v \in V_{ZK_i}$ zu G_{ZK_i} zugefügt. Der so entstandene Graph $G^+_{ZK_i}$ ist nun dreifach zusammenhängend und eine planare Einbettung folgend eindeutig. Es wird eine planare Einbettung in Linearzeit berechnet. Nach dem Entfernen von v_x erhält man eine eindeutige outerplanare Einbettung von G_{ZK_i}. □

Wir behandeln zwei Fälle getrennt: den Fall, dass es keinen Anschlussknoten und den Fall, dass es einen oder mehr Anschlussknoten gibt. Zunächst soll es mindestens ein Anschlussknoten geben:

Zeichnung einer zweifachen Zusammenhangskomponente mit Anschlussknoten

Gegeben sei eine zweifache Zusammenhangskomponente ZK mit ihren Anschlussknoten A_{ZK} und den angegliederten Komponenten (die Klassifizierung jener). Die Position eines der Anschlussknoten relativ zum Rest der Komponente sei vorgegeben. Wir nennen diesen Knoten a_1. Also zum Beispiel a_1 soll auf der unteren Linie und der linken Seite der Komponente zugeordnet sein, wie in Abbildung 5.11.

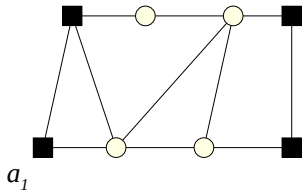

Abbildung 5.11: Zeichnung mit a_1 links unten

Nun werden die Bedingungen 1 und 2 aus Lemma 5.3.7 überprüft, sie sind einbettungsunabhängig. Danach wird eine outerplanare Einbettung bestimmt, siehe den Beweis von Lemma 5.3.8, die eine zyklische Anordnung

der Knoten entlang der äußeren Facette vorgibt. Daraus lässt sich nun eine Zeichnung berechnen, indem man den vorgegebenen Knoten auf einen Eckpunkt setzt und dann im Uhrzeigersinn versucht die Knoten aufeinander folgenden zu setzen. Wenn ein Anschlussknoten erreicht wird oder die bereits einer Linie zugeordneten Knoten plus dem nächsten Knoten kein Pfad mehr induzieren, muss die Linie gewechselt werden. Dies ist in Algorithmus 5 genauer beschrieben. Nach dem der Algorithmus im Erfolgsfall die Anordnung der Komponente auf der Linie zurückgegeben hat, werden die restlichen Bedingungen von Lemma 5.3.7 überprüft. Im negativen Fall wird das gleiche Verfahren mit umgekehrtem Umlaufsinn $(v_1 = a_1, v_n, v_{n-1}, ..., v_2)$ wiederholt. Falls auch dies nicht zum Erfolg führt, gibt es die gewünschte Zeichnung nicht.

Korrektheit Bei genau einem oder genau vier Anschlussknoten ist direkt einsehbar, dass dieses Verfahren das Gewünschte liefert. Bei genau zwei oder drei ist vorstellbar, dass nur Zeichnungen gefunden werden, bei denen zwei Anschlussknoten mit Komponenten aus V_{EKN} oder zweiseitige Komponenten angeschlossen sind auf eine Seite gelegt werden, obwohl es noch weitere Zeichnungen gibt, bei denen diese auf verschiedenen Seiten liegen.

Man kann nun den Algorithmus so modifizieren, dass er es nicht zulässt, dass zwei solche Knoten auf die gleiche Seite gelegt werden. Durch einige Überlegungen sieht man aber auch, dass die zweimalige Anwendung mit beiden Umlaufrichtungen mit Algorithmus 5 trotzdem die gewünschte Zeichnung liefert, falls sie existiert.

Beweis: Sei o.B.d.A. a_1 auf der linken unteren Ecke und a_2 auf der linken oberen und beide sind zu Komponenten aus V_{EKN} oder zu beidlinigen Komponenten adjazent. Nehmen wir an, dass es eine zulässige Zeichnung gibt, die unser Verfahren nicht gefunden hat. Diese muss dann so sein, dass a_2 alleine auf der oberen Linie gezeichnet ist, nur dann ist a_2 auch zur anderen Seite, in diesem Fall der rechen, frei. Dann induziert aber $V_{ZK} \setminus \{a_2\}$ einen Pfad. Somit ist eine Zeichnung mit a_2 rechts unten gültig, diese hätte der Algorithmus aber beim umgekehrten Durchlauf gefunden. □

Zeichnung einer zweifachen Zusammenhangskomponente ohne Anschlussknoten

Nun machen wir uns noch Gedanken, wie wir zu einer Zeichnung kommen, falls es gar keine Anschlussknoten gibt. Im vorigen Fall wechselte der Algorithmus die Linie falls eine Anschlussknoten auftrat oder falls die schon auf die Linie gezeichneten Knoten plus den nächsten Knoten keinen Pfad

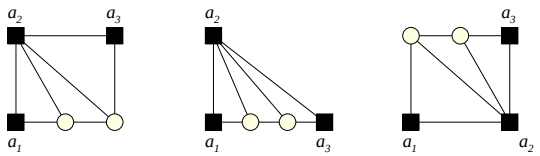

Abbildung 5.12: Beispiel zu Beweis 5.3.2,

Eingabe : Zweifache Komponente $V_{ZK_i} = |n|$, Anordnung der Knoten $v_1 = a_1, v_2, ..., v_n$, Eckpunkt von a_1
Ausgabe: Zeichnung der Komponente falls möglich
setze $v_1 = a_1$ auf gewünschten Eckknoten ;
setze v_2 im Uhrzeigersinn auf nächste Position ;
if a_1 *links unten oder rechts oben* **then** setze $P = \{v_2\}$;
else setze $P = \{v_1, v_2\}$;
for $i = \{3, ..., n\}$ **do**
  if *(* $v_{i-1} \in A$ *und* v_{i-1} *nicht einziger Knoten auf Linie) oder Graph induziert durch* $L = P \cup \{v_i\}$ *kein Pfad* **then**
    wechsle Linie;
    setze v_i auf nächste Position;
    setze $P = \{v_i\}$;
    for $j = \{i, .., n\}$ **do**
      if *Graph induziert durch* $L = P \cup \{v_j\}$ *kein Pfad* **then**
        EXIT not possible;
      end
      setze v_j auf nächste Position;
      setze $P = P \cup \{v_j\}$;
    end
    RETURN Zeichnung;
  end
  setze v_i auf nächste Position;
  setze $P = P \cup \{v_i\}$;
end
RETURN Zeichnung;

Algorithmus 5: Zeichnung von ZK mit Anschlussknoten

mehr induzieren. Wir modifizieren den Algorithmus so, dass wir mit einem beliebigen Knoten beginnen aber die Linie zweimal anstatt einmal wechseln dürfen. So muss der Anfangsknoten nicht unbedingt auf einem Eckpunkt liegen, es ist aber Notwendig zu überprüfen ob die Knoten, auf der Linie des Anfangsknotens einen Pfad induzieren. Siehe dazu auch Algorithmus 6 und Abbildung 5.13.

Eingabe : Zweifache Komponente $V_{ZK_i} = |n|$;
Zyklische Anordnung der Knoten $v_1 = a_1, v_2, ..., v_n$
Ausgabe: Zeichnung der Komponente falls möglich
setze v_1 auf obere Linie ;
setze v_2 im Uhrzeigersinn auf nächste Position ;
setze $P = \{v_1, v_2\}$, $i = 2$;
while $P \cup \{v_{i+1}\}$ *induziert Pfad* **do**
  $i++$;
  setze v_i auf nächste Position;
  $P = P \cup \{v_i\}$;
end
wechsle Linie;
setze v_{i+1} auf nächste Position;
$j = i + 2$;
if $v_j = v_1$ **then** RETURN Zeichnung;
$P = \{v_j\}$;
while $P \cup \{v_{j+1}\}$ *induziert Pfad* **do**
  $j++$;
  setze v_j auf nächste Position;
  if $v_{j+1} = v_1$ **then** RETURN Zeichnung;
  $P = P \cup \{v_j\}$;
end
wechsle Linie;
if $v_{j+1}, ..., v_1$ *induziert Pfad* **then** RETURN Zeichnung;
else EXIT not possible

Algorithmus 6: Zeichnung von ZK ohne Anschlussknoten)

5.3.3 Konstruktion der Zeichnung

Wir zeichnen den Graph induktiv über die zweifachen Zusammenhangskomponenten und Teilbäume. Zunächst müssen wir nach einer Anfangs-

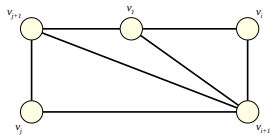

Abbildung 5.13: Gezeichnete Komponente ZK die keine Anschlussknoten hat

Komponente suchen. Dazu nehmen wir den aus $V \setminus V_{EK}$ induzierten Subgraphen und kontrahieren in ihm jede zweifache Zusammenhangskomponente und jeden Teilbaum zu jeweils einem Knoten. Nach Lemma 5.3.4 muss der so entstandene Graph ein einfacher Pfad sein. Der Anfangs- und End-Knoten entspricht jeweils einer Komponente, die am Anfang bzw. Ende der Zeichnung stehen wird.

Die Erste Komponente: Induktionsanfang Die erste Komponente ist entweder ein Teilbaum oder eine zweifache Zusammenhangskomponente. Behandeln wir erst den Fall des Teilbaumes.

Teilbaum Im ersten Teilbaum kommen maximal zwei Anschlussknoten vor, und wiederum maximal einer davon ist ein Verbindungsknoten zu einer Komponente der linearen Anordnung. Falls kein Anschlussknoten enthalten ist (in diesem Fall besteht der ganze Graph nur aus diesem Baum) so können wir direkt die Ergebnisse aus Kapitel 4 übernehmen. Falls es ein oder zwei Anschlussknoten gibt entfällt das Suchen eines Knotens der dem ausgezeichneten Pfad P angehört. Die Anschlussknoten sind dann der Anfangs bzw. Endknoten des ausgezeichneten Pfades im Teilbaum. Mit dieser Einschränkung können wir wieder die Algorithmen aus Kapitel 4 anwenden. Falls es die gewünschte Zeichnung gibt sind die Anschlussknoten nun die ersten bzw letzten auf einer der beiden Linien, und die adjazenten einseitigen Komponenten können einfach dazu gezeichnet werden, so dass alle ihre Knoten auf der anderen Linie liegen. Falls es einen Verbindungsknoten gibt so soll die Zeichnung so geändert werden, dass dieser rechts von allen anderen Knoten liegt. Das Ergebnis solch einer Zeichnung könnte wie in Abbildung 5.14 aussehen.

Zweifache Zusammenhangskomponente Falls es keinen Anschlussknoten gibt, dann besteht der ganze Graph nur aus dieser Komponente. Die Zeichnung kann direkt aus Abschnitt 5.3.2 übernommen werden. Im anderen

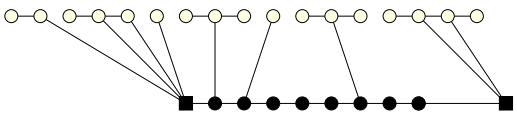

Abbildung 5.14: Zeichnung des ersten Teilbaumes

Fall sei a_1 der Verbindungsknoten oder, falls nicht vorhanden, ein beliebiger Anschlussknoten. Wir setzen a_1 o.B.d.A auf die linke untere Ecke, berechnen ob es eine Zeichnung gibt wie in Abschnitt 5.3.2. Wir zeichnen diese, dann nach links, so dass die Knoten auf dem Raster liegen. Die Angeschlossenen Komponenten aus V_{EKP_i} werden dann auf der gleichen Linie wie der zugehörige Anschlussknoten fortsetzend gezeichnet. Zuletzt werden die Komponenten gegeben durch V_{EKN_i} auf der Linie gegenüber ihrem Anschlussknoten gezeichnet und a_1 auf die Position rechts von allen anderen Knoten, auf der gleichen Linie, auf der er schon gezeichnet war. Die entstandene Zeichnung könnte wie Abbildung 5.15 aussehen.

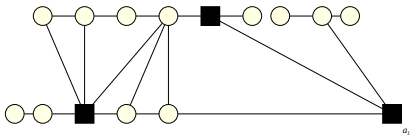

Abbildung 5.15: Zeichnung der ersten zweifachen Zusammenhangskomponente

Weitere Komponenten: Induktionsschritt

Teilbaum Der erste Verbindungsknoten a_1 ist vorgegeben. Mit ihm wird wie dem Fall, dass die erste Komponente ein Teilbaum ist, der ausgezeichnete Pfad berechnet bei dem a_1 der erste Knoten und falls vorhanden a_2 der letzte Knoten ist. Dieser Pfad wird dann von a_1 auf der gleichen Linie wie a_1 bis zum letzten Knoten v_l eingebettet. Die weiteren angeschlossenen Knoten werden der Reihenfolge nach beginnend auf der gleichen Höhe wie a_1 eingebettet. Falls danach a_2 nicht mehr der letzte Knoten ist wird er nach rechts verschoben. Siehe Abbildung 5.16.

Zweifache Zusammenhangskomponente Die Zeichnung erfolgt analog zur ersten Komponente. a_1 ist vorgegeben der Rest rechts davon ge-

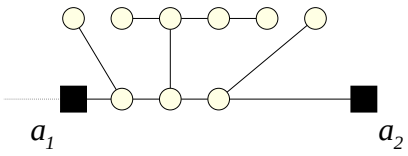

Abbildung 5.16: Zeichnung eines Teilbaumes

zeichnet. Ob es überhaupt eine Zeichnung gibt,die zyklische Anordnung der Knoten und welche Knoten auf den Ecken liegen wird wie in in Abschnitt 5.3.2 gezeigt ermittelt. Danach ist die Zeichnung nur noch durch Strecken so anzupassen, dass alle Knoten auf dem Raster liegen. Falls es einen zweiten Verbindungsknoten a_v gibt so wird dieser analog zum Teilbaum auf die äußerste Position rechts von allen anderen Knoten verschoben.

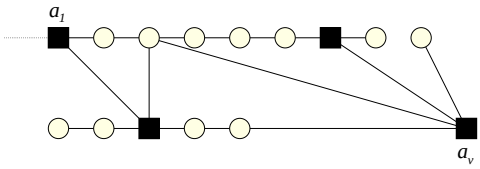

Abbildung 5.17: Zeichnung einer zweifachen Zusammenhangskomponente

Kapitel 6

LLL-Zeichnung mit zusammenhängendem A-Pfad

6.1 Einfache LLL-Zeichnung mit zusammenhängendem A-Pfad

Gegeben sei ein planarer Graph $G = (V, E)$ mit einer Partition $V = A \cup B$, wobei A ein Pfad $(a_1, a_2), ..., (a_{n-1}, a_n)$ sei. Ziel ist eine geradlinige, planare Zeichnung wobei alle Knoten von A auf einer mittleren und alle Knoten von B auf einer parallelen Linie darüber bzw. darunter liegen. Kanten von der oberen zur unteren B Linie sind nicht erlaubt.

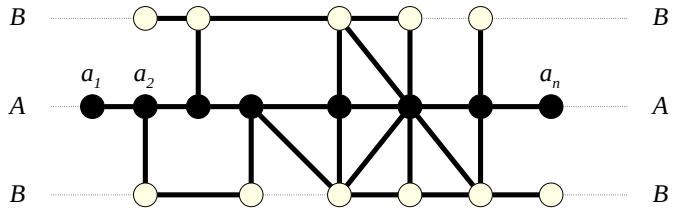

Abbildung 6.1: Beispiel für einen LLL-Zeichnung mit zusammenhängendem A-Pfad

Definition 6.1.1 (Flügelkomponenten)
Sei $F_i \subset V$ die Knotenmenge einer Zusammenhangskomponente in B, $a_{i_{min}} \in A$ der zu F_i adjazente Knoten aus A mit kleinstem Index und analog $a_{i_{max}}$ jener mit dem größtem. Die Flügelkomponente FK_i ist dann definiert durch die

Knotenmenge $F_i \cup \{a_{i_{min}}, a_{i_{min}+1}, ..., a_{i_{max}}\}$ (falls $a_{i_{max}} = a_{i_{min}}$ wird natürlich nur ein Knoten a_i dazu gezählt).

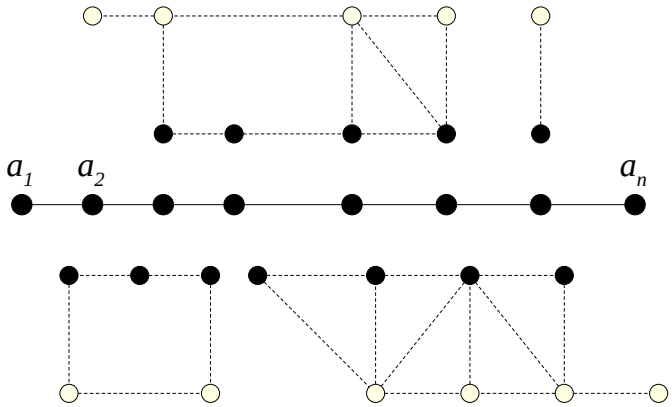

Abbildung 6.2: Die Flügel von G

Eine Kante aus A heißt belegt, wenn sie einer Flügelkomponente angehört, 2-fach belegt wenn sie zwei verschiedenen Flügelkomponenten angehört und n-fach belegt, wenn sie n verschiedenen Flügelkomponenten angehört. Ein Knoten $a_i \in A$ heißt verfügbar, wenn er kleinster oder größter A-Knoten einer Flügelkomponente ist.

Lemma 6.1.2
Ein Graph $G = (A \cup B, E)$ hat genau dann eine LLL-Zeichnung mit Zusammenhängendem A-Pfad und keinen Kanten zwischen den B-Linien wenn folgendes gilt:

- *jede Flügelkomponente hat eine LL-Zeichnung*
- *jede Kante (a_i, a_{i+1}) ist maximal 2-fach Belegt*
- *jeder Knoten aus A der mehr als 2 Komponenten angehört ist verfügbar*

Beweis:

- $\Rightarrow$ ist klar
- $\Leftarrow$ siehe unten

□

6.2 Testen der Bedingungen und Vorbereiten der Zeichnung

Zuerst werden die Flügelkomponenten FK_i berechnet. Von einem beliebigen Knoten aus B starten wir eine Breitensuche entlang aller Kanten, stoppen aber bei Knoten aus A. Dabei vermerken wir jeweils den kleinsten $a_{i_{min}}$ und größten Knoten $a_{i_{max}}$ die wir dabei erreichen. Wenn alle Kanten und Knoten erreicht wurden wird zu den ermittelten Knoten aus B die Knoten $a_{i_{min}}, ..., a_{i_{max}}$ zu der Komponente dazu genommen. Die Knotenmenge einer Flügelkomponente ist somit bestimmt. In einer geeigneten Datenstruktur wird die Belegung der Kanten zwischen $a_{i_{min}}$ und $a_{i_{max}}$ um eins erhöht. Dies wird solange wiederholt bis alle Knoten aus B zu Flügelkomponenten zugeordnet sind.

Jetzt können die Bedingungen von Lemma 6.1.2 überprüft werden. Dabei wird beim Testen auf LL-Zeichnung der Pfad vermerkt entlang dem die Knoten aus $B \cap FK_i$ gesetzt werden.

6.3 Anfertigen der Zeichnung

Zuerst werden alle Knoten aus A ihrer Anordnung nach beginnend mit a_1 auf die mittlere Linie gezeichnet. Beginnend bei a_1 werden die Flügelkomponenten induktiv dazu gezeichnet.

Bei einem Knoten a_i seien alle Flügelkomponenten deren kleinster A-Knoten kleiner als a_i ist schon gezeichnet. Als nächste Komponente wird jene gezeichnet deren kleinster A-Knoten a_i ist und deren größter A-Knoten der mit kleinstem Index unter allen diesen Komponenten ist. Die Komponente wird dann auf jene B-Linie gelegt, deren (a_i, a_{i+1}) Kante noch nicht belegt ist. Nachdem alle Komponenten, die a_i als ihren kleinsten A-Knoten haben in dieser weise gezeichnet wurden wird zum nächsten Knoten a_{i+1} übergegangen.

Beweis: [Korrektheit der Zeichnung] Die Korrektheit sieht durch Induktion über die im Zeichnungsverfahren hinzugefügten Komponenten. □

6.4 Zeitverhalten

Lemma 6.4.1 (Zeitverhalten)
Die Zeichnung lässt sich in linearer Zeit Anfertigen.

Beweis: Das Ermitteln jeder Flügelkomponente ist linear zur Anzahl der Kanten der Komponente, welche wegen der Planarität linear zur Anzahl der

Knoten ist. Somit können alle Flügelkomponenten in $O(n)$ Zeit bestimmt werden. Das Testen auf LL-Zeichnung ist in linearer Zeit möglich, siehe auch [3]. Das Anfertigen der Zeichnung ist in linearer Zeit möglich. Man beachte dabei, dass es bei jedem Knoten a_i maximal zwei Komponenten gibt, die bei diesem beginnen und Kanten im A-Pfad belegen. Somit ist es nicht nötig die in Frage kommenden Komponenten nach ihrem kleinsten $a_{i_{max}}$ zu sortieren. □

Kapitel 7

Ausblick

Sei $G = (A \cup B, E)$ ein zusammenhängender Graph. Eine allgemeine LLL-Zeichnung ist eine Zeichnung, wobei die Knoten aus A auf einer mittleren Linie und die Knoten aus B auf zwei Linien darüber und darunter liegen.

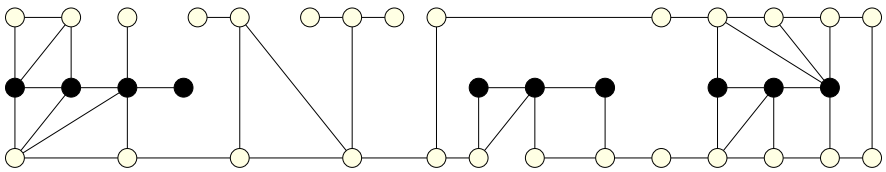

Abbildung 7.1: allgemeine LLL-Zeichnung

Wenn man nun geeignete Knoten auswählt und den Graph bei diesen trennt, zerfällt er in einfache Pfade, partitionierte und unpartitionierte LL-Zeichnungen und den LLL-Zeichnungen mit zusammenhängendem A-Pfad wie in Kapitel 6. Um allgemeine LLL-Zeichnungen anzufertigen ist es nun noch nötig solche Trennknoten zu finden, die es ermöglichen die Teilkomponenten einzeln zu behandeln und dann auch wieder zu einer Zeichnung zusammenfügen lassen. Es ist offen, ob dieses Problem mit vernünftigem Zeitaufwand zu realisieren ist.

Literaturverzeichnis

[1] Dieter Jungnickel, Graphen, Netzwerke und Algorithmen ISBN-3-411-14263-4

[2] Swamy / Thulasiraman, Graphs, Networks and Algorithms ISBN-0-471-03503-3

[3] T.Biedl. Drawing Planar Partitions I: LL-Drawings And LH-Drawings.*Rutcor Research Report RRR 11-98*, March, 1998

[4] T.Biedl. Drawing Planar Partitions III: Two constrained embeddings. *Rutcor Research Report RRR 13-98*, February 1998

[5] K. Wagner, Über eine Eigenschaft der ebenen Komplexe, *Math. Ann.*, Vol 114, 570-590 (1937)

[6] C. Kuratowski, Sur le problème des courbes gauches en topologie, *Fund.Math.*, Vol 15, 271-283 (1930)

[7] J.E. Hopcroft and R.E. Tarjan. Efficient planarity testing. *Journal of the Association for Computing Machinery* 21(4), October 1974

Erklärung:

Ich erkläre, dass ich die Arbeit selbstständig und nur mit den angegebenen Hilfsmittel angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, durch Angabe der Quellen als Entlehnungen kenntlich gemacht worden sind.

Konstanz, den 7.1.2001 (Thomas Schank)