UNIVERSITÄT DORTMUND

Kreuzungsminimales Einfügen einer Kante in einen aufwärtsplanaren sT-Graphen

Hoi-Ming Wong

Algorithm Engineering Report
TR06-1-007
August 2006
ISSN 1864-4503

Universität Dortmund
Fachbereich Informatik
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

Diplomarbeit

Kreuzungsminimales Einfügen einer Kante in einen aufwärtsplanaren sT-Graphen

Hoi-Ming Wong
August 2006

INTERNE BERICHTE
INTERNAL REPORTS

Diplomarbeit am
Fachbereich Informatik
der Universität Dortmund
Betreuer:
Prof. Dr. Petra Mutzel
Dipl.-Inform. Carsten Gutwenger

Inhaltsverzeichnis

Vorwort

Zu Beginn der Arbeit möchte ich mich bei einigen Personen für ihre Unterstützung bedanken. Ich bedanke mich bei meinem Betreuer Carsten Gutwenger, der mir stets hilfsbereit zur Seite stand und bei Prof. Dr. Petra Mutzel für die Annahme der Diplomarbeit. Bedanken möchte ich mich auch bei meiner Familie und meinen Freunden für deren Unterstützung.

1 Einleitung

1.1 Einleitung

Graphen werden in vielen Gebieten eingesetzt. Sei es zur Darstellung von Prozessen, Flüssen, komplexen Systemen, mathematische Strukturen oder zur Modellierung. Entsprechend existieren viele und vielseitige Applikationen auf dem Markt. Hier seien nur CAD-Programme, UML Modellierungswerkzeuge, Programme zum Schaltkreisdesign oder Programme zum Projektmanagement erwähnt. Besonders wichtig hierbei ist die Darstellung der Graphen, womit wir in das Gebiet des Graphenzeichnens kommen. Das *Graphenzeichnen* beschäftigt sich unter anderem mit dem Entwurf von Algorithmen zur Visualisierung von Graphen, mit dem Ziel ein Graph möglichst „schön“ zu zeichnen. Dabei gilt ein Graph als schön gezeichnet, wenn die Zeichnung übersichtlich ist und semantische Zusammenhänge sich leicht daraus ablesen lassen. Es hängt von vielen Faktoren ab, ob ein Graph schön gezeichnet ist oder nicht. Obwohl dies auch sehr stark von der Person des Betrachters abhängt, haben sich dennoch einige Kriterien herauskristallisiert:

- Gleichmäßige Verteilung der Knoten des Graphen auf der Zeichenfläche.
  Zu viele Knoten auf einer kleinen Fläche führen zur Unübersichtlichkeit.

- Minimales Überkreuzen der Kanten des Graphen.
  Damit lassen sich die Kanten einfacher verfolgen.

- Eine generelle Richtung der gerichteten Kanten.
  D.h. möglichst viele gerichtete Kanten sollen in eine gemeinsame Richtung zeigen. Zusammenhänge werden dadurch schneller erkannt.

- Planares zeichnen.

- Symmetrie.
  Aus der Symmetrie können wichtige semantische Zusammenhänge abgeleitet werden. Auch sind symmetrische Zeichnungen übersichtlicher als nicht symmetrische.

- Kanten sollten kurz sein.
  Dadurch kann man sie einfacher verfolgen.

- Zeichenfläche sollten minimal sein.
  Dadurch wird eine kompakte Darstellung erreicht.

Die Liste ist keinesfalls vollständig und sollte uns nur einen Einblick geben. Ziel ist es, die gewünschten Kriterien optimal zu erfüllen. Das ist jedoch nicht ohne weiteres möglich, da die Optimierung einiger Kriterien, wie z.B. Kreuzungsminimalität oder die platzoptimale Zuweisung der Knoten in einem Schicht-Layout, NP-schwer ist [20, 13]. Deshalb greift man häufig auf Heuristiken zurück und gibt sich mit suboptimalen Lösungen zufrieden.

Ein recht erfolgreicher Ansatz, Graphen übersichtlich darzustellen, ist das Planarisieren. Auch wenn nicht alle Graphen planar zeichenbar sind, kann man durch Einfügen von zusätzlichen Knoten, sogenannten *Dummy-Knoten*, an den Kreuzungen den Graph planar gestalten. Ziel ist es, immer möglichst wenige dieser Dummy-Knoten einzufügen (Kreuzungsminimierung).

Bei der Klasse der azyklischen, gerichteten Graphen (DAG) fordert man nicht Planarität, sondern die Aufwärtsplanarität des Graphen. Ein Graph ist *aufwärtsplanar*, wenn er eine planare Zeichnung besitzt, deren gerichtete Kanten monoton steigend in die vertikale Richtung zeigen. Aufwärtsplanarisierung ist ein relativ neuer Ansatz zur Zeichnung von DAGs. Klassischerweise werden DAGs mit der Sugiyma-Methode gezeichnet [35, 13]. Diese Methode hat jedoch den Nachteil, dass die Knoten auf fest zugeordneten Schichten verteilt werden, was zu unnötige Kreuzungen führen kann. Da sich die Planarisierung bei ungerichteten Graphen als eine sehr erfolgreiche Methode zur Kreuzungsminimierung erwiesen hat, erhofft man sich den gleichen Erfolg auch von der Aufwärtsplanarisierung von DAGs.

Der Aufwärtsplanarisierungsprozess kann in zwei Phasen eingeteilt werden. In der ersten Phase wird ein maximaler aufwärtsplanarer Untergraph berechnet und in der zweiten Phase werden die Kanten, die bei der Berechnung des Untergraphen entfernt wurden, wieder (möglichst kreuzungsminimal) eingefügt. Das Problem der zweiten Phase wird auch *Edge Insertion Problem* genannt.

In dieser Diplomarbeit untersuchen wir das *Edge-Insertion-Problem* für die Klasse der DAGs mit genau einer Quelle, welche auch kurz als sT-Graphen bezeichnet werden. Die Diplomarbeit wird unter anderem durch folgende Tatsachen motiviert:

1. Das *Edge-Insertion-Problem* spielt eine wichtigen Rolle in der Aufwärtsplanarisierung von DAGs.

2. Der Aufwärtsplanaritätstest ist für die Klasse der sT-Graphen effizient durchführbar [7], während der Test für allgemeine DAGs NP-schwer ist [21].

3. Das *Edge-Insertion-Problem* für sT-Graphen wurde noch nicht untersucht. Die guten Ergebnisse aus den Arbeiten von Gutwenger et al. [25], die das Problem für ungerichtete, planare Graphen untersucht haben, lassen vermuten, dass ähnliche Ergebnisse auch für gerichtete Graphen zu erwarten sind.

1.2 Gliederung

Die Diplomarbeit ist wie folgt aufgebaut:
In Kapitel 2 werden einige grundlegende Begriffe der Graphentheorie wiedergegeben. Insbesondere liegt der Schwerpunkt auf der Aufwärtsplanarisierung. Eine wichtige und häufig benutze Datenstruktur, von der wir in dieser Arbeit Gebrauch machen werden, ist der SPQR-Baum. Diese Datenstruktur wird in Kapitel 2.2 genauer erläutert. Die Grundlage der Diplomarbeit bilden die beiden Veröffentlichungen „*Inserting an Edge into a Planar Graph*“ von Gutwenger et al. und „*Optimal Upward Planarity Testing of Single-Source Digraphs*“ von Di Battista et al. Die Kernideen und wesentlichen Aussagen dieser beiden Veröffentlichungen werden in Kapitel 2.3 und 2.4 beschrieben.

Der klassiche Zeichenalgorithmus von Sugiyma, der bisher zum Zeichnen von DAGs verwendet wird, wird in Kapitel 2.5 erläutert. Eine formale Beschreibung sowie eine erste Analyse des *Edge-Insertion-Problems* für sT-Graphen wird in Kapitel 3 gegeben. Aus dieser Analyse folgend werden dann Untersuchungen zur Spiegelung von Graphkomponenten durchgeführt.

In Kapitel 4 wird eine Datenstruktur vorgestellt, die auf einen SPQR-Baum basiert und implizit alle aufwärtsplanaren Einbettungen eines 2-zusammenhängenden sT-Graphen enthält.

Ein Lösungsansatz für das *Edge-Insertion-Problem* für eingebettete sT-Graphen wird in Kapitel 5 vorgestellt.

Basierend auf den Ergebnissen der Analyse aus Kapitel 3 und des Lösungsansatzes des *Egde-Insertion-Problems* für eingebettete sT-Graphen in Kapitel 5 wird in Kapitel 6 ein erweiterter Lösungsansatz beschrieben.

In Kapitel 7 wird schließlich eine Zusammenfassung der wesentlichen Ergebnisse dieser Diplomarbeit gegeben. Desweiteren werden noch offene Fragen aufgezeigt.

2 Grundlagen

In diesem Kapitel werden einige Grundlagen der Graphentheorie erläutert. Insbesondere wird auf die Datenstruktur SPQR-Baum, den Algorithmus zum kreuzungsminimalen Einfügen einer Kante von Gutwenger et al. [25] und den Aufwärtsplanaritätstest für sT-Graphen von Di Battista et al. [7] eingegangen. Der klassische Algorithmus [13] zum Zeichnen von DAGs wird im letzten Abschnitt erläutert.
Für grundlegenden Algorithmen und Datenstrukturen, die hier wegen des Umfangs nicht erläutert werden, verweisen wir auf das Buch „*Introduction to Algorithms*“ [10].

2.1 Graphen

Als Referenz für die Abschnitte 2.1.1 und 2.1.2 dient das Buch „*Handbook of Graph Theorie*“ [23] und die Einführung „*Upward Planarity Testing*“ von Garg und Tamassia [22].

2.1.1 Grundlegende Definitionen der Graphentheorie

Definition 2.1. Ein *ungerichteter Graph* ist ein Paar $G = (V, E)$ bestehend aus einer Menge V von *Knoten* und einer Menge E ungeordneter Paare (u, v) mit $u, v \in V$, die *Kanten* genannt werden. Die Knoten u und v werden als *Endpunkte* der Kante (u, v) bezeichnet.

Wenn der Knoten v ein Endpunkt einer Kante e ist, so ist v *inzident* zu e oder e ist inzident zu v. Sind zwei Knoten u und v durch eine Kante verbunden, so ist u *adjazent* zu v und umgekehrt.

Eine wichtige Klasse von Graphen bilden die gerichteten Graphen/Digraphen (Directed Graphs).

Definition 2.2. Ein *gerichteter Graph*, kurz *Digraph* genannt, ist ein Paar $G = (V, A)$ bestehend aus einer Menge V von *Knoten* und einer Menge A geordneter Paare (u, v) mit $u, v \in V$, die *gerichtete Kanten* oder *Bögen* genannt werden. u ist der *Startpunkt/Startknoten* und v der *Zielpunkt/Zielknoten* der Kante (u, v). (Die Kante wird symbolisch auch mit $u \to v$ beschrieben.)

Der *zugrundeliegende ungerichtete Graph* (engl. *underlying, undirected graph*) eines Digraphen ist der Graph, der daraus resultiert, indem man die Kanten des Digraphen als ungerichtete Kanten interpretiert.

Ein *(ungerichteter) Pfad* $P = v_0, e_1, v_1, \ldots, e_n, v_n$ ist eine alternierende Sequenz von Knoten und Kanten, so dass für alle $j = 1, \ldots, n$ die Knoten v_{j-1} und v_j die Endpunkte der Kante e_j sind[1]. Ferner sind alle Knoten und Kanten des Pfades disjunkt. Sind alle Kanten e_j von v_{j-1} nach v_j gerichtet, so handelt es sich um einen *gerichteten Pfad*. Ein Pfad von u nach v wird auch symbolisch mit $u \rightsquigarrow v$ beschrieben. Wenn kein solcher Pfad existiert, dann wird auch symbolisch $u \nrightarrow v$ benutzt. Existiert in einem Graphen weder ein Weg von u nach v noch ein Weg von v nach u, dann wird das durch $u \nleftrightarrow v$ beschrieben. Ein Knoten u *dominiert* einen Knoten v, wenn ein gerichteter Pfad von u nach v existiert (vergl. [7]).

Ein *Kreis/Zyklus* ist ein Pfad $v_0, e_1, v_1, \ldots, e_n, v_n$ plus die Kante (v_n, v_0). Die kreisfreien/azyklischen Digraphen werden auch kurz *DAGs* (*directed acyclic graph*) genannt.

Der *Ingrad* eine Knotens v ist die Anzahl der Kanten mit v als Zielknoten. Entsprechend ist der *Outgrad* von v die Anzahl der Kanten mit v als Startknoten. Der Ingrad bzw. Outgrad von v wird mit $in(v)$ bzw. $out(v)$ notiert. Ein Knoten in einem Digraphen G mit Ingrad gleich Null wird *Quelle* von G genannt. Ein Knoten mit Outgrad gleich Null ist eine *Senke* von G.

Definition 2.3. Ein *sT-Graph* ist ein DAG mit genau einer Quelle s.

Ein sT-Graph wird als *st-Graph* bezeichnet, wenn er genau eine Senke besitzt und noch zusätzlich die gerichtete Kante (s, t) enthält. Klar einzusehen ist, dass in einem sT-Graphen die Quelle s alle anderen Knoten dominiert.

Definition 2.4. Ein Graph $G' = (V', E')$ ist ein *Teilgraph* von $G = (V, E)$, wenn $V' \subseteq V$ und $E' \subseteq E$ gilt. Ein Graph $G' = (V', E')$ ist ein *Untergraph* von $G = (V, E)$, wenn $E' \subseteq E$ ist.

Ein Graph ist *zusammenhängend*, wenn zwischen jedem Paar von Knoten ein ungerichteter Pfad existiert. Ein k-fach zusammenhängender Graph wird wie folgt definiert:

Definition 2.5. Ein Graph $G = (V, E)$ ist *k-fach (Knoten) zusammenhängend*, wenn $|V| > k$ und $G - X$ ist zusammenhängend für jede $X \subseteq V$ mit $|X| < k$.

Im Kontext eines Digraphen G bezieht sich mit k-fachzusammenhängend — wenn nicht anders erwähnt — auf den zugrundeliegenden Graphen von G. Ein maximaler zusammenhängender, nicht leerer Teilgraph K von G ist eine *Komponente* von G. Ein *Schnittknoten* eines zusammenhängenden Graphen G ist ein Knoten,

[1] Ein Pfad wird auch eindeutig beschrieben, indem entweder eine Sequenz von Kanten oder Knoten angegeben wird.

durch dessen Entfernung sich die Anzahl der zusammenhängenden Komponenten des Graphen erhöht. Müssen erst zwei Knoten entfernt werden, damit sich die Anzahl der zusammenhängenden Komponenten erhöht, so werden die beiden Knoten als *Separationspaar* des Graphen G bezeichnet. Ein maximaler zusammenhängender Graph ohne Schnittknoten wird auch als *Block* oder *2-Zusammenhangskomponente* genannt.

2.1.2 Einbettung und Aufwärtsplanarität

Nachdem in Abschnitt 2.1.1 die grundlegenden Definitionen abgehandelt wurden, wenden wir uns hier den Aufwärtsplanaritätseigenschaften der Digraphen zu.

Ein Graph G ist *planar*, wenn es eine Zeichnung von G gibt, die keine Kreuzungen von Kanten enthält. Eine *Aufwärtszeichnung* eines gerichteten Graphen G ist eine Zeichnung von G bei der alle Kanten streng monoton in vertikale Richtung steigend gezeichnet sind.

Definition 2.6. Ein Digraph G ist *aufwärtsplanar*, wenn es eine planare Aufwärtszeichnung von G gibt.

Eine planare Zeichnung eines Graphen partitioniert die Ebene in verschiedene Gebiete. Diese Gebiete sind durch Kanten getrennt und werden *Faces*[2] genannt. Ein Face kann für eine gegebene planare Zeichnung durch die Folge der begrenzenden Kanten eindeutig beschrieben werden. Um Beschreibungen nicht unnötig zu verkomplizieren, wird mit Face auch diese Kantenaufzählung bezeichnet, falls das eindeutig aus dem entsprechenden Kontext hervorgeht. Mit *innerem Face* wird ein inneres Gebiet und mit *äußerem Face* wird das Gebiet bezeichnet, das nicht durch Kanten beschränkt ist.

Zwar gibt es unendlich viele Möglichkeiten einen planaren Graphen zu zeichnen, doch mit Hilfe der Kantenordnung in jedem Knoten lässt sich die Zeichnungen eines planaren Graphen in endlich viele Äquivalenzklassen partitionieren[3]. Diese Äquivalenzklassen heißen *Einbettungen*.

1. *Schwache Äquivalenz*
   Zwei planare Zeichnungen D_1 und D_2 eines Graphen G sind *schwach äquivalent*, wenn für jeden Knoten v in G die zirkulare Aufzählung der inzidenten Kanten um v im Uhrzeigersinn in D_1 die gleiche ist wie in D_2.

2. *Starke Äquivalenz*
   Zwei Zeichnungen sind stark äquivalent, wenn sie schwach äquivalent sind und das gleiche äußere Face besitzen.

[2] Die englische Bezeichnung wird hier übernommen.
[3] Selbstverständlich muss die Anzahl der Knoten des planaren Graphen endlich sein.

Definition 2.7. Die *kombinatorischen Einbettungen* eines Graphen sind die Äquivalenzklassen der schwachen Äquivalenz. Die *planaren Einbettungen* eines Graphen sind die Äquivalenzklassen der starken Äquivalenz.

Definition 2.8. Eine (planare) Einbettung wird als *aufwärtsplanare Einbettung* bezeichnet, wenn sie eine planare Aufwärtszeichnug besitzt.

Da ein Digraph genau dann eine Aufwärtszeichnung besitzt, wenn er azyklisch ist [7], kann ein einfacher Test, ob ein Digraph eine Aufwärtszeichnung besitzt, mittels einer Tiefensuche durchgeführt werden. Hopcroft und Tarjan [26] haben gezeigt, dass der Planaritätstest effizient durchgeführt werden kann. Interessant ist, dass der Aufwärtsplanaritätstest für allgemeine Digraphen NP-vollständig ist [21], obwohl beide Tests jeder für sich keine Probleme bereiteten.

Satz 2.1 ([21])**.** *Der Aufwärtsplanaritätstest für Digraphen ist NP-vollständig.*

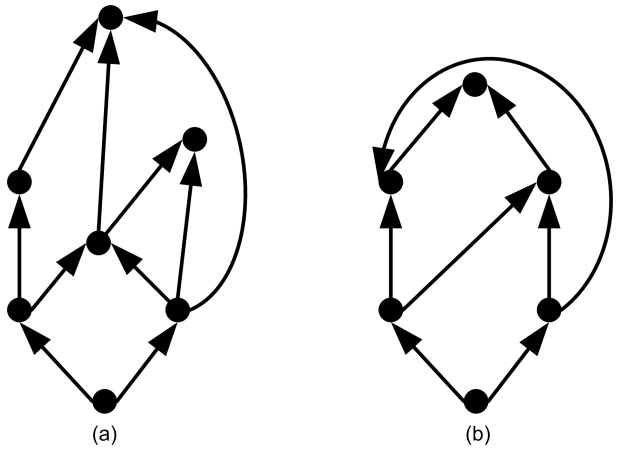


Abbildung 2.1: (a) ein aufwärtsplanarer sT-Graph; (b) ein azyklischer und planarer, aber nicht aufwärtsplanarer Graph;

Trotz dieses negativen Ergebnisses können einige Klassen von Graphen effizient auf Aufwärtsplanarität getestet werden.

Satz 2.2. *Die folgenden Klassen von Digraphen sind aufwärtsplanar.*

- *Planare st-Graphen [3, 30].*
- *Planare* bipartite Digraphen *[2]. Das sind Graphen, deren Knotenmenge V sich in zwei disjunkte Teilmengen A und B partitionieren lassen, so dass zwischen zwei beliebigen Knoten, die beide zusammen entweder in A oder B zu finden sind, keine Kanten existieren.*

- *Digraphen, deren zugrundeliegender ungerichteter Graph einen Wald bildet [33].*
- Serien-parallele Digraphen *[8]. Diese Graphen werden wie folgt rekursiv definiert:*
  - *Zwei Knoten, die mit einer gerichteten Kanten verbunden sind, bilden einen serie-parallelen Digraphen.*
  - *Seien G' und G zwei serien-parallele Digraphen. Verschmilzt man die Senke von G mit der Quelle von G', so erhält man wieder einen serien-parallelen Digraphen (Serielle Komposition).*
  - *Seien G' und G zwei serien-parallele Digraphen. Verschmilzt man die Senken sowie die Quellen beider Graphen miteinander, so ist der resultierende Graph wieder ein serien-paralleler Digraph (Parallele Komposition).*

Wichtig für den Aufwärtsplanarisierungprozess sind die Aufwärtsplanaritätstests. Mit Hilfe dieser Tests kann z.B. ein (möglichst großer) aufwärtsplanarer Untergraph berechnet werden. Die Berechnung kann wie folgt durchgeführt werden: Es wird mit einem leeren Graph G' begonnen. Eine Kante e wird zufällig aus E gewählt — wobei jede Kante in E die gleiche Wahrscheinlichkeit besitzen ausgewählt zu werden — und in G' eingefügt. Danach wird ein Aufwärtsplanaritätstest auf G' durchgeführt. Ist G' nicht aufwärtsplanar, so wird die Kante e wieder aus G' entfernt, ansonsten wird die nächste Kante gewählt. Dieser Vorgang wird iterativ solange fortgeführt, bis alle Kanten von E ausgewählt wurden. Häufig wird eine „multi start“ Variante dieser Methode angewandt. Hierbei wird die obige Methode mehrmals angewendet, um so eine Menge von Lösungen zu bekommen. Anschließend wird aus dieser Menge die beste Lösung gewählt.

Der folgende Satz gibt einige Ergebnisse zu Aufwärtsplanaritätstests wieder.

Satz 2.3. *Es gilt:*

(a) Der Aufwärtsplanaritätstest eines eingebetteten sT-Graphen $G = (V, A)$ kann in Zeit $\mathcal{O}(|V| + |E|)$ durchgeführt werden [7].

(b) Der Aufwärtsplanaritätstest eines eingebetteten Digraphen $G = (V, A)$ (ohne Multikanten) kann in Zeit $\mathcal{O}(|V|^2)$ durchgeführt werden [6].

(c) Der Aufwärtsplanaritätstest für 3-zusammenhängende Digraphen ohne Multikanten kann in Zeit $\mathcal{O}(|V|^2)$ durchgeführt werden. [6].

(d) Der Aufwärtsplanaritätstest für außenplanare (engl. outerplanar) Graphen — also Digraphen, die eine planare Zeichnung besitzen, so dass alle Knoten am äußeren Face liegen — kann in Zeit $\mathcal{O}(|V|^2)$ durchgeführt werden [33].

Für eine feste Einbettung ist der Aufwärtsplanaritätstest für allgemeine Digraphen kein Problem. Schwierig wird es erst dann, wenn keine feste Einbettung vorgegeben wurde. Dann muss im ungünstigsten Fall über alle Einbettungen, deren Anzahl exponentiell sein kann, getestet werden. Durch die Tatsache, dass 3-zusammenhängende Graphen genau zwei Einbettungen besitzen, kann (c) des obigen Satzes auf (b) zurückgeführt werden.

Der nachfolgende Satz von Di Battista, Tamassia und Kelly [3, 30] gibt eine Charakterisierung aufwärtsplanarer Graphen an.

Satz 2.4. *Ein Digraph G ist genau dann aufwärtsplanar, wenn er ein Untergraph eines planaren st-Graphen ist.*

Für planare *st*-Graphen existieren Zeichenverfahren für Aufwärtszeichnungen (siehe z.B. [5]). Um diese zu benutzen, werden aufwärtsplanare Digraphen durch Hinzufügen von Kanten zu einem planaren *st*-Graphen erweitert. Dieser *st*-Graph wird dann aufwärtsplanar gezeichnet und die hinzugefügten Kanten werden wieder entfernt.

2.2 Der SPQR-Baum

SPQR-Bäume wurden von Di Battista und Tamassia [4] eingeführt und werden in vielen Algorithmen zum automatischen Zeichnen von Graphen genutzt. Ein SPQR-Baum repräsentiert die Dekompositionen eines 2-zusammenhängenden Graphen in seine 3-Zusammenhangskomponenten [4]. Die Datenstruktur bietet eine kompakte Darstellung aller Einbettungen eines 2-zusammenhängenden planaren Graphen. Die nachfolgenden Definitionen und Fakten orientieren sich an Gutwenger et al., Bertolazzi et al. sowie an der Dissertation von Weiskircher [7, 25, 36].

2.2.1 Grundlagen und Definitionen

Ein *Split-Paar* ist entweder ein Separationspaar oder zwei adjazente Knoten. Eine *Split-Komponente* eines Split-Paares $\{u, v\}$ ist entweder die Kante (u, v) oder ein maximaler Untergraph K von G, so dass $\{u, v\}$ kein Split-Paar von G ist.
Sei nun $\{s, t\}$ ein Split-Paar von G. Ein *maximales Split-Paar* $\{u, v\}$ bezüglich $\{s, t\}$ ist ein Split-Paar ungleich $\{s, t\}$, so dass für jedes andere Split-Paar $\{u', v'\}$ von G eine Split-Komponente von $\{u', v'\}$ existiert, die die Knoten u, v, s und t enthält.

Sei nun $G = (V, E)$ ein 2-zusammenhängender Graph und $e = (s, t)$ eine Kante von G, auch *Referenzkante* genannt. Ein SPQR-Baum $\mathcal{T}$ von G bezüglich der Kante e

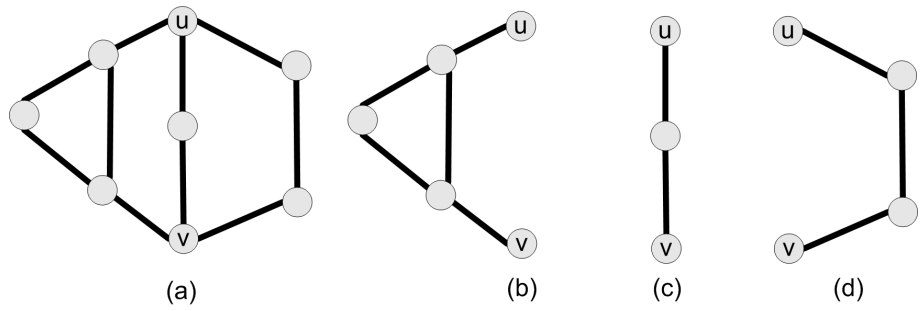


Abbildung 2.2: Ein 2-zusammenhängender Graph (a) und seine Split-Komponenten (b), (c), (d) bezüglich des Split-Paares $\{u, v\}$. Jeder Knoten außer u und v wird eindeutig einer Split-Komponente zugewiesen.

ist ein gewurzelter und geordneter Baum, dessen Knoten sich in vier Typen, nämlich S-, P-, Q- und R-Knoten, einteilen lassen. Jeder Knoten μ des Baumes $\mathcal{T}$ ist mit einem 2-zusammenhängenden Multigraphen — *Skeleton* von μ genannt (Notation: $skeleton(\mu)$) — assoziiert. Der SPQR-Baum wird rekursiv wie folgt definiert:

Trivialer Fall: Wenn G nur aus zwei parallelen Kanten besteht, die die Knoten s und t verbinden, dann enthält $\mathcal{T}$ einen einzelnen Q-Knoten. Das Skeleton dieses Q-Knoten ist der Graph G selbst.

Paralleler Fall: Wenn das Split-Paar $\{s, t\}$ den Graphen G in die Split-Komponenten $C_1, \ldots, C_k$, $k > 2$, zerlegt, dann ist die Wurzel μ von $\mathcal{T}$ ein P-Knoten. Das Skeleton von μ besteht aus k Kanten $e = e_1, \ldots, e_k$ mit den gemeinsamen Endpunkten s und t. Die Kinder von μ werden durch die SPQR-Bäume der Graphen $C_i \cup e_i$ mit e_i als Referenzkante rekursiv definiert.

Serieller Fall: Wenn das Split-Paar $\{s, t\}$ den Graphen G in genau zwei Split-Komponenten zerlegt, dann ist die Wurzel μ von $\mathcal{T}$ ein S-Knoten. Eine der Split-Komponenten ist e und die andere sei G'. Enthält G' die Schnittknoten $v_1, \ldots, v_{k-1}$, die G in die Blöcke $B_1, \ldots, B_k$ (in der Reihenfolge von s nach t) partitionieren, so ist das Skeleton von μ der Kreis $e_0 = e = (s, t), e_1 = (s, v_1), \ldots, e_k = (v_{k-1}, t)$. Die Kinder von μ werden durch die SPQR-Bäume der Graphen $B_i \cup e_i$ mit e_i als Referenzkante rekursiv definiert.

Starrer Fall: Wenn keiner der obigen Fälle zutrifft, dann ist die Wurzel μ von $\mathcal{T}$ ein R-Knoten. Seien $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, mit $k \geq 1$, die maximalen Split-Paare von G bezüglich $\{s, t\}$. Für alle $i = 1, \ldots, k$ sei U_i die Vereinigung aller Split-Komponente des Split-Paares $\{s_i, t_i\}$, außer der Komponente, die die Referenzkante e enthält. Das Skeleton von μ erhält man, indem die Graphkomponente U_i durch die Kante $e_i = (s_i, t_i)$ ersetzt wird. Die Kinder von μ werden durch die SPQR-Bäume der Graphen $U_i \cup e_i$, mit e_i als Referenzkante, rekursiv definiert.

Abgesehen vom trivialen Fall hat μ die Kinderknoten $\mu_1, \ldots, \mu_k$. Die Kante e_i der obigen Definition wird *virtuelle Kante* des Knoten μ_i in $skeleton(\mu)$ genannt. Anders herum ist e_i auch die virtuelle Kante von μ in $skeleton(\mu_i)$.
Zu jeder virtuellen Kante eines Skeletons gibt es einen entsprechenden Knoten von $\mathcal{T}$ sowie eine Graphkomponenente von G. Ersteres wird als *pertinenter Knoten*[4] und letzteres als *pertinenter Graph* bezeichnet.
Ein Knoten v ist in einer virtuellen Kante e *lokalisiert*, wenn v ein Knoten des pertinenten Graphen von e ist. Entsprechend ist v im Knoten μ, der mit e assoziiert ist, lokalisiert.

Die Abbildung 2.3 illustriert einige Fälle der obigen Definition.
In (a) ist der serielle Fall dargestellt. Die Schnittknoten v_1 und v_2 partitionieren den Graphen in die Blöcke B_1, B_2 und B_3. Diese Blöcke werden im Skeleton des S-Knoten durch die Kanten e_1, e_2 und e_3 ersetzt. Sie bilden zusammen mit der Referenzkante $e = (s, t)$ einen Kreis.

In (b) ist der parallele Fall dargestellt. Die parallelen Split-Komponenten C_1, C_2 und C_3 werden im Skeleton durch die Kanten e_2, e_3 und e_4 ersetzt. Das Skeleton ist ein Multigraph mit zwei Knoten.

Der starre Fall ist in (c) dargestellt. U_1 bis U_5 bilden die Vereinigung der Split-Komponenten der entsprechenden Split-Paare. Im Skeleton des R-Knotens werden diese Komponenten durch die Kanten e_1 bis e_5 ersetzt. Das Skeleton selbst ist ein 3-zusammenhängender Graph.

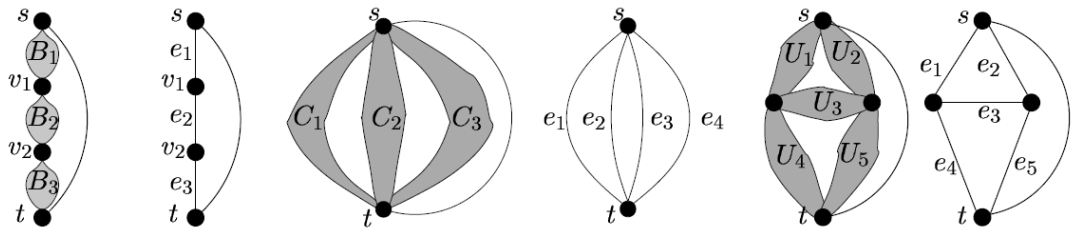


Abbildung 2.3: Illustration zur Definition des SPQR-Baumes. Zeichnung aus [36]

2.2.2 Eigenschaften

Nach der formalen Definition im vorherigen Abschnitt wird hier näher auf die Eigenschaften des SPQR-Baumes eingegangen. Für die Skeletons eines SPQR-Baums

[4] engl.: pertinent=zugehörig. Hier wird die englische Bezeichnung übernommen.

gilt:

- Das Skeleton eines R-Knotens ist ein 3-zusammenhängender Graph.
- Das Skeleton eines S-Knotens ist ein Kreis.
- Das Skeleton eines P-Knotens ist ein Multigraph mit mindestens drei Kanten.
- Das Skeleton eines Q-Knoten ist ein Multigraph mit zwei Kanten.

Eine virtuelle Kante e eines Skeletons ist mit ihrem pertineten Graphen assoziiert. Die Kante e steht quasi „stellvertretend“ für diese Graphkomponente. Wird für einen Graphen $G = (V, E)$ eine (kombinatorische) Einbettung festgelegt, so wird durch diese Einbettung auch die Einbettung aller Skeletons des SPQR-Baumes $\mathcal{T}$ festgelegt. Umgekehrt wird durch die Festlegung der Einbettung der einzelnen Skeletons von $\mathcal{T}$ eine Einbettung von G definiert. Somit bildet das Produkt aller Kombinationen der Einbettungen der Skeletons die Gesamtheit der (kombinatorischen) Einbettungen von G.

Die Anzahl der Einbettungen eines einzelnen Skeletons im Falle eines R-Knotens ist genau zwei und im Falle eines S-, und Q- Knotens ist sie genau eins. Im Falle des P-Knotens beträgt sie $(k-1)!$, wobei k die Anzahl der virtuellen Kanten des Skeletons ist.

Wird $\mathcal{T}$ an einem Q-Knoten μ, der mit einer Kante e' assoziiert ist, gewurzelt, so repräsentiert $\mathcal{T}$ alle Einbettungen mit e' an dem äußeren Face von G. Sind alle Einbettungen der Skeletons festgelegt, kann die Zeichnung mit der Einbettung, die $\mathcal{T}$ darstellt, rekursiv wie folgt vonstatten gehen:

1. Zeichne das Skeleton von μ mit der Kante e' am äußeren Face.
2. Für jedes Kind ν von μ:
   - Sei e die virtuelle Kante von ν in $skeleton(\mu)$ und H der pertinete Graph von ν mit der Kante e.
   - Zeichne rekursiv H mit der Referenzkante e an dem äußeren Face.
   - Ersetze die virtuelle Kante e in $skeleton(\nu)$ durch H ohne die Kante e.

Ein SPQR-Baum hat genau $|E|$ Q-Knoten. Die Anzahl der S-, P, und R-Knoten liegt in der Größenordnung $\mathcal{O}(|V|)$. Die Gesamtzahl der Knoten in den Skeletons liegt in der Größenordnung $\mathcal{O}(|V| + |E|)$. Somit hat ein SPQR-Baum lineare Größe in Bezug auf die Anzahl der Knoten und Kanten eines Graphen. Mit Hilfe des Algorithmus von Hopcroft und Tarjan [27][5] zur Zerlegung eines Graphen in seine 3-Zusammenhangskomponenten, kann die Datenstruktur effizient in Zeit $\mathcal{O}(|V| + |E|)$ berechnet werden.

[5] Der Algorithmus hatte einige Fehler. Gutwenger und Mutzel präsentierten in [24] eine fehlerfreie Version.

2.3 Kreuzungsminimales Einfügen einer Kante für ungerichtete Graphen

Gutwenger, Mutzel und Weiskircher untersuchten das *Edge-Insertion-Problem* für planare Graphen (siehe [25]). Ihnen ist es gelungen, einen effizienten Algorithmus zu entwerfen, der eine Kante kreuzungsminimal in einen planaren Graphen einfügt. Die Idee zur Lösung basiert auf einem zentralen Lemma, das besagt, dass die Traversierungskosten einer mit einer Graphkomponente K assoziierten virtuellen Kante e unabhängig von der Einbettung von K ist. Mit diesem Ergebnis, sowie einem SPQR-Baum, der die Einbettung des Graphen implizit aufzählt, wird dann mit Hilfe eines Routing-Graphen ein optimaler Einfügepfad berechnet. In diesem Abschnitt werden wir die wesentlichen Ergebnisse daraus skizzieren.

2.3.1 Grundlagen und Definitionen

Zum einfacheren Verständnis der späteren Beschreibungen werden hier einige Definitionen aus [25] wiedergegeben.

Definition 2.9. Sei G ein zusammenhängender Graph (primaler Graph) und Γ eine kombinatorische Einbettung von G. Der *duale Graph* Γ^D von G in Abhängigkeit der Einbettung Γ ist wie folgt definiert:

1. Für jedes Face $f \in \Gamma$ gibt es einen Knoten $f^D \in \Gamma^D$.
2. Für jede Kante $e \in \Gamma$ gibt es eine Kante $e^D \in \Gamma^D$, mit $e^D = (f_1, f_2)$, wenn e eine Kante der Grenze zwischen den Faces f_1, $f_2 \in \Gamma$ ist.
3. Das sind alle Kanten und Knoten.

Definition 2.10. Sei $G = (V, E)$ ein zusammenhängender Graph und sei Γ eine Einbettung von G. Ferner seien x und y zwei nicht adjazente Knoten aus G. Dann ist $e_1, \ldots, e_k$ ein *Einfügepfad* des Knotenpaares x und y für die Einbettung Γ, wenn entweder $k = 0$ ist und ein Face f in Γ existiert, das x und y enthält[6], oder die folgenden Bedingungen sind erfüllt:

1. $e_1, \ldots, e_k \in E$.
2. Es gibt ein Face f_x in Γ, das x und die Kante e_1 enthält.
3. Es gibt ein Face f_y in Γ, das y und die Kante e_k enthält.
4. $e_1^D, \ldots, e_k^D$ ist ein Pfad in Γ^D von f_x^D nach f_y^D.

[6] Ein Face f enthält den Knoten v, wenn eine Kante aus f existiert, die v als einer ihrer Endpunkte enthält.

Aus dem dualen Graphen wird ein Routing-Graph konstruiert, der zur Berechnung des Einfügepfades des Knotenpaares x und y benötigt wird.

Definition 2.11. Seien F_x und F_y die Mengen der Faces, die den Knoten x bzw. y enthalten. Der *Routing-Graph* Γ^R der Einbettung Γ und der Knoten (x, y) ist der duale Graph von Γ mit den zusätzlich Knoten x und y, sowie die Kanten (x, f_x), für alle $f_x \in F_x$, und der Kanten (y, f_y) für alle $f_y \in F_y$.

Der Einfügepfad gibt eine Menge von Kanten im primalen Graphen G an, die gekreuzt werden, wenn ein Pfad von Knoten x nach Knoten y in G konstruiert wird. Dieser Einfügepfad ist eindeutig assoziiert mit einem Pfad im Routing-Graphen von G. Ziel ist es, einen Pfad unter allen Einbettungen von G zu finden, der eine minimale Anzahl an Kanten kreuzt. Dieser Pfad wird *optimaler Einfügepfad* genannt.
Die *Kosten* eines Einfügepfades sind die Anzahl der Kanten in dem Pfad. Wenn die Kanten im Routing-Graphen (die auch Kanten des dualen Graphen sind) alle die Länge eins und sämtliche nicht duale Kanten alle die Länge null besitzen, dann sind die Kosten eines Einfügepfades gleichbedeutend mit der Länge des assoziierten Pfades im Routing-Graphen.

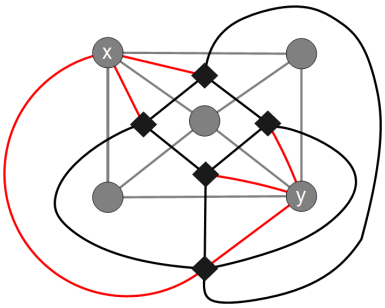


Abbildung 2.4: Der primale Graph ist in grau und der dazugehörige duale Graph in schwarz gezeichnet. Der Routing-Graph besteht aus dem dualen Graphen, den Knoten x und y und allen rot gezeichneten Kanten.

Die Abbildung 2.4 illustriert die obigen Definitionen. Der primale Graph mit der Einbettung Γ ist in grau und der duale Graph Γ^D in schwarz gezeichnet. Zu jedem Face von G existiert ein Knoten in Γ^D und zu jeder Kante von Γ gibt es eine eindeutige Kante im dualen Graphen. Der Routing-Graph besteht aus dem dualen Graphen, den Knoten x und y und allen rot gezeichneten Kanten, die das Knotenpaar x und y mit allen ihren benachbarten Faces verbindet.

2.3.2 Einfügen einer Kante in einen 2-zusammenhängenden Graphen

Zunächst wird das Problem auf planare 2-zusammenhängende Graphen beschränkt. Der Vorteil bei dieser Klasse von Graphen ist, dass alle Einbettungen durch einen SPQR-Baum implizit aufgezählt werden können. Später wird mit Hilfe der Zerlegung eines zusammenhängenden Graphen in seine Blöcke die Lösung des Problems auf allgemein zusammenhängende, planare Graphen erweitert.

Sei nun μ ein Knoten in einem SPQR-Baum $\mathcal{T}$ und e eine virtuelle Kante von $skeleton(\mu)$. Mit $expansion^+(e)$ wird der pertinente Graph von e plus der Kante e bezeichnet. Sei Π eine beliebige Einbettung von $expansion^+(e)$ und f_1 und f_2 zwei Faces in Π, die e als gemeinsame Grenze haben. In dem dualen Graphen Π^D von Π gibt es dann zwei Knoten f_1^D und f_2^D, die mit f_1 und f_2, und eine Kante e^D, die mit e assoziiert sind. Mit $P(\Pi^D, e)$ wird in Π^D der kürzeste f_1^D und f_2^D verbindende Pfad, der die Kante e^D nicht enthält, notiert.

Mit der obigen Notation kann nun der Begriff Traversierungskosten definiert werden: Die *Traversierungskosten* $c(e)$ einer virtuellen Kante eines Skeletons sind die Länge des Pfades $P(\Pi^D, e)$ für eine beliebige Einbettung von $expansion^+(e)$. Diese Definition ist durch das folgende Lemma gerechtfertigt.

Lemma 2.1 (aus [25]). *Sei μ ein Knoten eines SPQR-Baumes $\mathcal{T}$ und e eine virtuelle Kante von $skeleton(\mu)$. Dann ist die Länge des Pfades $P(\Pi^D, e)$ unabhängig von der Einbettung von $expansion^+(e)$.*

Beweis aus [25]. Sei ν der pertinente Knoten von e und $\mathcal{T}_e$ der Teilbaum von $\mathcal{T}$ mit ν als Wurzel. Die Aussage wird mittels Induktion über die Höhe h des Teilbaumes $\mathcal{T}_e$ bewiesen.
Induktionsanfang: $h = 1$
ν ist ein Q-Knoten. $expansion^+(e)$ ist ein Kreis mit zwei Kanten und hat somit genau eine Einbettung. Die Länge von $P(\Pi^D, e)$ ist eins.
Induktionsschritt: $h = k > 1$
Sei die Behauptung korrekt für alle Teilbäume mit $h < k$. Wegen $k > 1$ ist ν entweder ein S-, P-, oder R-Knoten. Sei e_i eine beliebige Kante von $skeleton(\nu)$, die nicht die Referenzkante e_{ref} ist. Der zu e_i assoziierte Teilbaum $\mathcal{T}_{e_i}$ von $\mathcal{T}$ hat eine Höhe kleiner als k. Somit ist nach Induktionsvoraussetzung $c(e_i)$ unabhängig von der Einbettung von $expansion^+(e_i)$.

Ist ν ein S-Knoten, so ist $skeleton(\nu)$ ein Kreis mit den Kanten $e_{ref}, e_1, \ldots, e_l$ mit $l \geq 2$. $c(e)$ ist das minimale $P(\Pi^D, e_i)$ mit $1 \leq i \leq l$. Dies ist unabhängig von der Einbettung von $expansion^+(e)$.

Ist ν ein P-Knoten, so besteht $skeleton(\nu)$ aus den parallelen Kanten $e_{ref}, e_1, \ldots, e_l$ mit $l \geq 2$. Alle Kanten außer e_{ref} müssen traversiert werden. Somit ist $c(e)$ die

Summe aller $P(\Pi^D, e_i)$ mit $1 \leq i \leq l$. Dies ist unabhängig von der Einbettung von $expansion^+(e)$.

Ist ν ein R-Knoten, so ist $skeleton(\nu)$ ein 3-zusammenhängender planarer Graph S mit den Kanten $e_{ref}, e_1, \ldots, e_l$ mit $l \geq 5$. $c(e)$ lässt sich wie folgt berechnen: Zuerst wird ein dualer Graph S^D zu S konstruiert. Jede duale Kante e_i^D zu e_i hat dabei die Länge $c(e_i)$. Dann wird ein kürzester Pfad berechnet, der die zwei Faces, die durch e_{ref} getrennt werden, miteinander verbindet. Dabei darf die Kante e_{ref} nicht in dem kürzesten Pfad sein. $c(e)$ ist dann die Länge dieses kürzesten Pfades. Da S nur zwei Einbettungen besitzt, wobei die zweite Einbettung durch eine Spiegelung der ersten entsteht, ist der kürzeste Weg unabhängig von der Einbettung. □

Lemma 2.1 ist die Basis für den folgenden Algorithmus, der einen optimalen Einfügepfad in einem planaren 2-zusammenhängenden Graph berechnet. Dieser Algorithmus wird später benutzt um einen optimalen Einfügepfad für allgemein planare Graphen zu berechnen.

Die Laufzeit des Algorithmus 1 ist in der Größenordnung $\mathcal{O}(|V| + |E|)$ und setzt sich wie folgt zusammen:

1. Die Berechnungszeit von $\mathcal{T}$ ist in $\mathcal{O}(|V| + |E|)$ (siehe [24]).

2. Mit Hilfe einer Breitensuche auf $\mathcal{T}$ kann die Berechnung des kürzesten Pfades $\mu_1, \ldots, \mu_k$ in $\mathcal{T}$ in Zeit $\mathcal{O}(|V| + |E|)$ realisiert werden.

3. Die Konstruktion von S_i ist linear in der Größe von $skeleton(\mu_i)$. Da die gesamte Größe aller Skeletons linear in der Größe von G ist, gilt somit die Laufzeit von $\mathcal{O}(|V| + |E|)$.

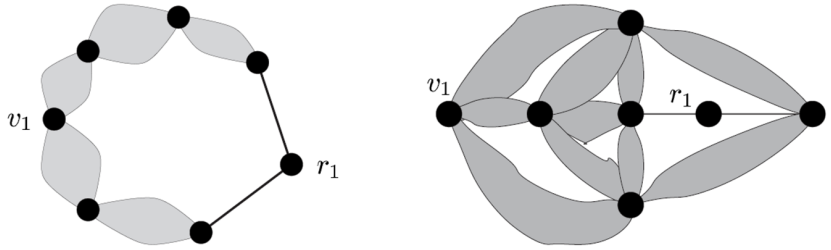
4. Die Größe von $G_i = (V_i, E_i \cup M_i)$ beträgt $\mathcal{O}(|E_i|)$ mit $|V_i| < |E_i|$. Dabei ist M_i die Menge der markierten Kanten und E_i die Menge der Kanten, die durch den Split-Prozess entstanden sind. Somit ist die Konstruktion einer beliebigen Einbettung von G_i und die Konstruktion des dazugehörigen Routing-Graphen in $\mathcal{O}(|E_i|)$ möglich. Die Berechnung des kürzesten Pfades, der x_i^1 mit x_i^2 verbindet, kann mit einer Breitensuche in $\mathcal{O}(|E_i|)$ realisiert werden. Die Laufzeit, über alle G_i summiert, liegt in der Größenordnung $\mathcal{O}(|E|)$.

Somit hat der Algorithmus eine Laufzeit von $\mathcal{O}(|V| + |E|)$.

Gutwenger et al. bewiesen die Korrektheit des Algorithmus, indem sie zuerst durch Induktion über die Pfadlänge $\mu_1, \ldots, \mu_i$ zeigten, dass eine Einbettung von G existiert, in der der berechnete Pfad ein Einfügepfad ist. Danach zeigten sie, dass für den kürzesten Einfügepfad P_{min} und den vom Algorithmus berechneten Pfad P_A die Ungleichung $P_{min} \geq P_A$ gilt. Der folgende Satz 2.5 hält dieses Ergebnis fest.

Algorithmus 1 aus [25]. Berechnet einen optimalen Einfügepfad für ein Paar von nicht adjazenten Knoten x und y in einem 2-zusammenhängenden Graphen G.

procedure OPTIMALBLOCKINSERTER(Graph G, Knoten x, Knoten y)
  Berechne den SPQR-Baum $\mathcal{T}$ von G;
  Berechne μ_1 bzw. μ_k deren Skeleton den Knoten x bzw. y enthält;
  Berechne den kürzesten Pfad $\mu_1, \ldots, \mu_k$ in $\mathcal{T}$;
  for $i := 1, \ldots, k$ **do**
    $S_i := skeleton(\mu_i)$;
    if x ist in S_i **then**
      $x_i^1 := x$;
    else
      Splitte die Kante in S_i, in der x lokalisiert ist,
      indem ein neuer Knoten y_i^1 eingeführt wird;
      Markiere die beiden neuen Kanten,
      die durch den Split-Prozess entstanden sind;
      $x_i^1 := y_i^1$;
    end if
    if y ist in S_i **then**
      $x_i^2 := y$;
    else
      Splitte die Kante in S_i, in der y lokalisiert ist,
      indem ein neuer Knoten y_i^2 eingeführt wird;
      Markiere die beiden durch den Split-Prozess entstandenen Kanten;
      $x_i^2 := y_i^2$;
    end if
    Sei G_i der Graph, der daraus entsteht, dass die unmarkierten Kanten
    von S_i durch ihren pertinenten Graphen ersetzt werden.
    if μ_i ist kein R-Knoten **then**
      $p_i := \emptyset$;
    else
      Berechne eine beliebige Einbettung Γ_i von G_i;
      Berechne den Routing-Graphen Γ^R von Γ_i, x_i^1, x_i^2;
      Berechne den kürzesten Pfad $e_0^D, \ldots, e_{l+1}^D$ in Γ^R zwischen
      x_i^1 und x_i^2;
      $p_i := e_1, \ldots, e_l$, wobei e_j die primale Kante von e_j^D ist;
    end if
  end for
  return $p_1 + \cdots + p_k$;
end procedure



(a) (b)

Abbildung 2.5: Illustration zum Algorithmus 1 für den Fall $i = 1$: (a) μ_i ist ein S-Knoten; (b) μ_i ist ein R-Knoten; (Bild aus [25])

Satz 2.5 (aus [25]). *Sei $G = (V, E)$ ein planarer, 2-zusammenhängender Graph und seien x und y zwei nicht adjazente Knoten aus G. Algorithmus 1 berechnet einen optimalen Einfügepfad für x und y in Laufzeit von $\mathcal{O}(|V| + |E|)$.*

2.3.3 Einfügen einer Kante in einen zusammenhängenden Graphen

Der Algorithmus 1 berechnet nur einen optimalen Einfügepfad für einen 2-zusammenhängenden Graphen. Mit Hilfe des sogenannten Block-Knoten-Baumes kann der Algorithmus auf planare, zusammenhängende Graphen erweitert werden.

Definition 2.12. Sei $G = (V, E)$ ein zusammenhängender Graph. Der *Block-Knoten-Baum* $\mathcal{B}$ entsteht, indem jeder Block von G durch einen Knoten B — auch *B-Knoten* genannt — ersetzt wird. Zu jedem Knoten von G wird ein Knoten v — auch *V-Knoten* genannt — erzeugt. Ist v ein Knoten des Blockes B, so werden v und B mit der Kante (v, B) in $\mathcal{B}$ verbunden. Ein *Repräsentant* von $v \in G$ in einem Block B ist entweder v selbst, wenn $v \in B$ gilt, oder der erste Schnittknoten c auf dem eindeutigen Weg von B nach v in $\mathcal{B}$.

Der Algorithmus zur Berechnung eines Einfügepfads in einem planaren, zusammenhängenden Graphen lokalisiert erst die Blöcke, in denen die Knoten x und y zu finden sind, und berechnet dann einen Pfad $P_{\mathcal{B}}$ in $\mathcal{B}$, der die beiden Blöcke verbindet. Der Pfad $P_{\mathcal{B}}$ beginnt mit dem Knoten x, gefolgt von einer Sequenz $B_1, c_1, \ldots, B_{k-1}, c_{k-1}, B_k$ und endet mit dem Knoten y. In jedem Block B_i von $P_{\mathcal{B}}$ wird der Einfügepfad P_i, der die Repräsentanten von x und y in B_i verbindet, mit dem Algorithmus 1 berechnet. Zuletzt werden die so berechneten Pfade konkateniert. Der Pseudocode dieses Verfahrens ist in Algorithmus 2 angegeben. Die Korrektheit

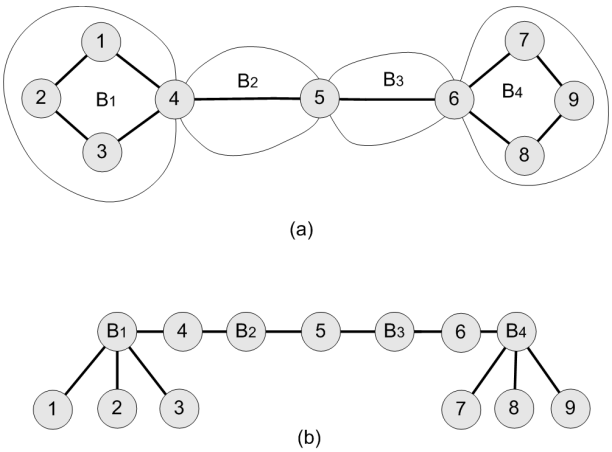


Abbildung 2.6: (a) ein planarer Graph mit den Schnittknoten 4, 5, 6 sowie der Blöcken B_1, B_2, B_3, B_4; (b) der Block-Knoten-Baum des Graphen;

und die Laufzeitanalyse kann in [25] nachgelesen werden.

Die Abbildung 2.6 zeigt einen Graphen mit dem dazugehörigen Block-Knoten-Baum. Zu jedem Block existiert ein B-Knoten in dem Block-Knoten-Baum. Der Weg von Knoten 2 zu Knoten 9 in dem Baum ist $2, B_1, 4, B_2, 5, B_3, 6, B_4, 9$.

Algorithmus 2 aus [25]. Berechnet einen optimalen Einfügepfad für ein Paar von nicht adjazenten Knoten x und y in einem zusammenhängenden Graphen G.

procedure OPTIMALINSERTER(Graph G, Knoten x, Knoten y)
  Berechne den Block-Knoten-Baum $\mathcal{B}$ von G;
  Berechne den Pfad $x, B_1, c_1, \ldots, B_{k-1}, c_{k-1}, B_k, y$ von x nach y in $\mathcal{B}$;
  for $1, \ldots, k$ **do**
    Sei x_i und y_i die Repräsentanten von x und y in B_i.
    Berechne den Einfügepfad p_i von x_i nach y_i in B_i mit Algorithmus 1;
  end for
  return $p_1 + \cdots + p_k$;
end procedure

Satz 2.6 (aus [25])**.** *Sei $G = (V, E)$ ein zusammenhängender Graph und seien x und y zwei nicht adjazente Knoten aus G. Algorithmus 2 berechnet einen optimalen Einfügepfad für die Knoten x und y in G in Laufzeit $\mathcal{O}(|V| + |E|)$.*

2.4 Aufwärtsplanaritätstest für sT-Graphen

Der Aufwärtsplanaritätstest für sT-Graphen von Bertolazzi et al. [7] bildet zusammen mit den Ergebnissen von Gutwenger et al. (siehe Kapitel 2.3) die Basis für den Lösungsansatz dieser Diplomarbeit.
Der erste effiziente Aufwärtsplanaritätstest für sT-Graphen mit einer Laufzeit von $\mathcal{O}(|V|^2)$ (ohne Multikanten) wurde von Hutton und Lubiw [28] veröffentlicht. Ihre Idee wurde von Bertolazzi et al. aufgegriffen und weiterentwickelt. Als Ergebnis entstand ein optimaler Aufwärtsplanaritätstest mit einer Laufzeit von $\mathcal{O}(|V| + |E|)$. In diesem Kapitel werden die wesentlichen Ergebnisse daraus zusammengefasst.

2.4.1 Grundlagen und Definitionen

In diesem Abschnitt werden die Grundlagen wiedergegeben, die zum Verständnis des späteren Algorithmus benötigt werden.

Definition 2.13. Sei $G = (V, A)$ ein DAG. Der Graph G ist *expandiert*, wenn für jeden Knoten $v \in V$ entweder $in(v) \leq 1$ oder $out(v) \leq 1$ gilt.

Ein *Expansionsgraph* kann aus einem azyklischen Digraphen G konstruiert werden, in dem jeder Knoten v von G mit den eingehenden Kanten $e_1^{in}, \ldots, e_k^{in}$ und den ausgehenden Kanten $e_1^{out}, \ldots, e_j^{out}$ in zwei Knoten aufgesplittet wird. D.h. es wird noch ein zusätzlicher Knoten v' konstruiert, der mit v durch die gerichtete Kante $e = (v', v)$ verbunden ist. v' enthält alle eingehenden Kanten und v enthält alle ausgehenden Kanten. Der Expansionsprozess zerstört nicht die Aufwärtsplanarität eines DAGs.

Lemma 2.2 (aus [7])**.** *Ein DAG G ist genau dann aufwärtsplanar, wenn der zu G gehörende Expansionsgraph G_{exp} aufwärtsplanar ist.*

Mit G/e wird eine *Kontraktion* notiert. Dies ist eine Operation auf einem Graphen, bei der eine Kante $e = (x, y)$ durch Verschmelzen der Knoten x und y entfernt wird. Eine *gerichtete Unterteilung* — notiert mit $sub(G, e)$ — ist eine Operation, in der ein Knoten z in eine Kante $e = (x, y)$ eingefügt wird. Das Ergebnis ist $x \to z \to y$.
Zwei Digraphen G_1 und G_2 sind *homöomorph* zueinander, wenn beide das Ergebnis einer endlichen Anwendung der gerichteten Unterteilungsoperation auf einen Digraph G sind.
Zwei Graphen sind *isomorph* zueinander, wenn sie bis auf die Umbenennung der Knoten gleich sind (Strukturgleichheit).
Ein Graph G_{minor} ist Minor eines Graphen G, wenn G_{minor} isomorph zu einem Graphen G' ist und G' aus G durch eine Anzahl von Kontraktionsoperationen entsteht.
Ein *Peak* ist die Struktur $u \to t \leftarrow v$ (siehe Abbildung 2.7 (b)). Die Knoten v und u werden *Basis(knoten)* und t, die *Spitze* des Peaks genannt. Ein *Valley* mit der Quelle

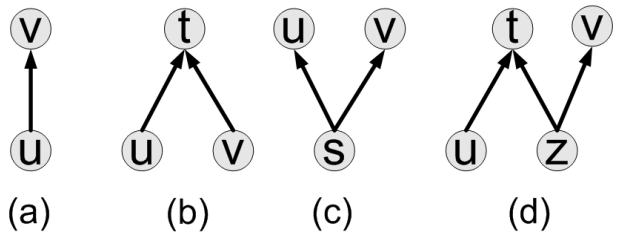


Abbildung 2.7: (a) eine gerichtete Kante; (b) ein Peak mit den Basis Knoten u und v und der Spitze t; (c) ein Valley mit den Spitzen u und v; (d) ein Zig-Zag;

s und den beiden Spitzen u und v ist die Struktur $a \leftarrow s \rightarrow b$ (siehe Abbildung 2.7 (c)). Ein *Zig-Zag* ist die Struktur $u \rightarrow t \leftarrow z \rightarrow v$ (siehe Abbildung 2.7 (d)).

Peak, Valley, gerichtete Kante und Zig-Zag sind Strukturen, die die „Form“ einer Graphkomponente wiedergeben. Ein Graph homöomorph zu einer dieser vier Strukturen behält nach einer endlichen Anwendung der Operation $sub(G, e)$ diese homöomorphe Eigenschaft bei. Diese Tatsache spielt eine besondere Rolle, wie sich noch herausstellen wird.
Das folgende Lemma gibt einige hilfreiche Fakten über die „Form“ von Graphkomponenten wieder.

Lemma 2.3 (aus [7]). *Sei $G = (V, A)$ ein sT-Graph und $u, v \in V$.*

1. *Wenn in G $u \nleftrightarrow v$ gilt, dann existiert in G ein Untergraph der homöomorph zu einem Valley mit den Spitzen u und v ist.*

2. *Sei $\{u, v\}$ ein Split-Paar von G und K eine Split-Komponente des Split-Paares, so dass v ein innerer Knoten und u eine Quelle von K ist, dann enthält K einen Untergraphen, der homöomorph ist zu einem Peak mit u und v als Basisknoten.*

3. *Sei nun G ein 2-zusammenhängender Digraph, dann ist weder u noch v ein Schnittknoten in den Split-Komponenten des Split-Paares.*

Sei G ein 2-zusammenhängender sT-Graph und K eine Split-Komponente des Split-Paares $\{u, v\}$. Mit K° wird der Graph $K - \{u, v\}$ bezeichnet und mit H der Teilgraph, der dadurch entsteht, dass alle Knoten außer u und v von K aus G entfernt werden (kurz $H = G - K$).

Die Idee von Lubiv und Hutton ist nun wie folgt: Statt den gesamten Graphen zu betrachten, betrachtet man nur einen „Teilauschnitt“, d.h. einen Minor von G. Gemäß Lemma 2.3 werden einige Teilgraphen durch eine zu ihnen homöomorphe Strukturen ersetzt. Diese Strukturen sind die oben beschriebenen Peak, Valley, gerichtete Kante und Zig-Zag. Es wird sich herausstellen, dass es sogar ausreicht für

den Zweck des Aufwärtsplanaritätstest statt der vier Strukturen nur zwei, nämlich Peaks und gerichtete Kanten, zu verwenden.

Die nachfolgenden Regeln geben an, wann eine Split-Komponente von $\{u, v\}$ durch einen Peak, $u \to \circ \leftarrow v$, oder durch eine gerichtete Kante, $u \to v$ bzw. $v \to u$, ersetzt wird. Im Falle eines Peaks wird ein Dummy-Knoten (die Spitze des Peaks) erzeugt. Dieser Knoten ist kein Knoten von G.

Ersetzungsregeln für die Minoren-Konstruktion[7]

- *Regel* $\mathcal{R}_1$ (Quelle/Quelle): u und v sind Quellen in K.
  K wird durch den Peak $u \to \circ \leftarrow v$ ersetzt.

- *Regel* $\mathcal{R}_2$ (Quelle/Senke):

  1. u ist Quelle und v ist Senke in K und $s \notin K^\circ$:
     K wird durch eine gerichtete Kante $u \to v$ ersetzt.
  2. u ist Quelle und v ist Senke in K und $s \in K^\circ$.
     K wird durch den Peak $u \to \circ \leftarrow v$ ersetzt.

- *Regel* $\mathcal{R}_3$ (Quelle/innerer Knoten):

  1. u ist Quelle, v ist innerer Knoten von K, $s \notin K^\circ$ und v ist Quelle in H:
     K wird durch eine gerichtete Kante $u \to v$ ersetzt.
  2. u ist Quelle, v ist innerer Knoten von K und $s \in K^\circ$:
     K wird durch den Peak $u \to \circ \leftarrow v$ ersetzt.
  3. u ist Quelle, v ist innerer Knoten von K und v ist nicht Quelle in H:
     K wird durch den Peak $u \to \circ \leftarrow v$ ersetzt.

- *Regel* $\mathcal{R}_4$ (nicht Quelle/nicht Quelle):
  Falls u Quelle in H ist, wird K durch die Kante $u \to v$ ersetzt, ansonsten wird K durch die Kante $u \leftarrow v$ ersetzt.

Die Peaks und die gerichteten Kanten, die K in G nach den obigen Regeln ersetzen, werden *gerichtete virtuelle Kanten* genannt und mit $d(K, G - K)$ notiert. Wie man leicht feststellen kann, ist der resultierende Graph nach dem Ersetzungsprozess ein Minor von G.

Sei $\mathcal{T}$ ein SPQR-Baum von G. Die oben beschriebene Minoren-Konstruktion kann auf die Skeletons von $\mathcal{T}$ erweitert werden. Ein Skeleton ist ein *sT-Skeleton*, wenn alle seine virtuellen Kanten durch gerichtete virtuelle Kanten ersetzt wurden. Die Ersetzungsregeln werden dabei auf die pertinenten Graphen der virtuellen Kanten

[7] Eine Illustration der Regeln ist in der Abbildung 3.3 zu sehen.

von $\mathcal{T}$ angewandt.

Abbildung 2.8 illustriert die Minoren-Konstruktion. Für die Komponente K_1 wird die Regel $\mathcal{R}_4$ und für die Komponente K_2 wird die Regel $\mathcal{R}_{2a}$ angewandt. Beide Komponenten werden jeweils durch eine gerichtete Kante ersetzt.

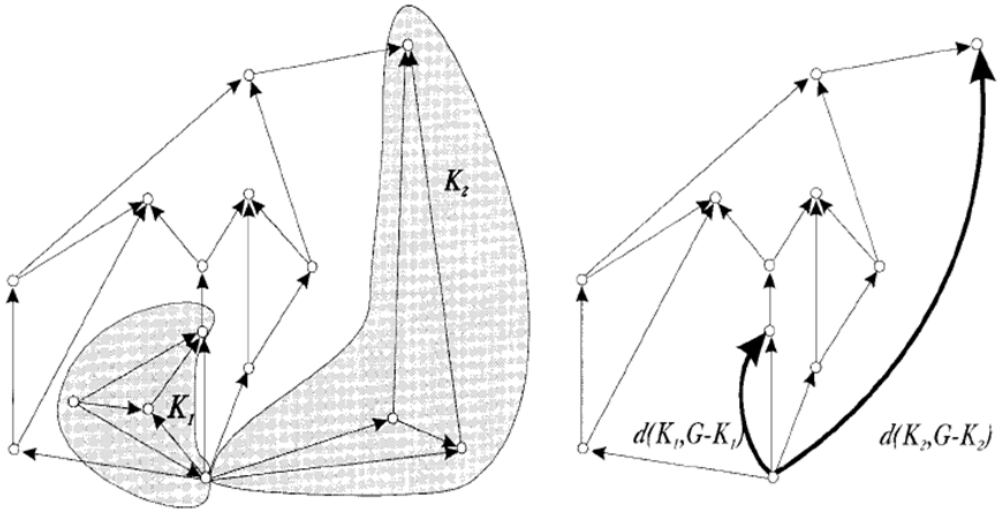


Abbildung 2.8: Minoren-Konstruktion mit den Regeln R_1 bis R_4.(Bild aus [7])

Charakterisierung von aufwärtsplanaren Einbettungen

In [6] geben Betolazzi und Di Battista eine Charakterisierung von aufwärtsplanaren Einbettungen eines Digraphen an. Diese hilfreiche Charakterisierung wird im Folgenden skizziert.

Sei Γ eine aufwärtsplanare Einbettung eines Digraphen G und D eine Zeichnung von Γ. Eine Quelle s bzw. eine Senke t von G ist einem Face f von D *zugewiesen*, wenn der Winkel, der durch die beiden zu s bzw. t inzidenten Kanten in f gebildet wird, größer ist als π. Bildlich gesprochen bedeutet dies, dass die beiden Kanten einer Quelle/Senke eine Spitze bilden. Wenn diese Spitze in ein Face „sticht“, dann wird die Quelle/Senke diesem Face zugeordnet (siehe Abbildung 2.9). Wenn die Quellen und Senken im äußeren Face liegen, dann ist klar, dass sie nur dem äußeren Face zugewiesen werden können, wenn die Einbettung aufwärtplanar ist. Bei einer aufwärtsplanaren Einbettung ist die Anzahl der Zuweisungen an einem Face begrenzt. Die Begrenzung wird durch die *Kapazität* c des Faces angegeben. Sie ist wie folgt definiert:

$$\begin{aligned} c(f) &= n_f - 1 \quad \text{falls } f \text{ innerer Face ist} \\ c(f) &= n_f + 1 \quad \text{falls } f \text{ äußerer Face ist} \end{aligned}$$

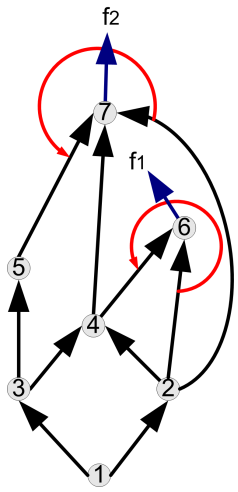


Abbildung 2.9: Zuweisung von Senken: Die Senke 6 wird dem Face f_1 zugewiesen, da der Winkel, den die Kanten $(4, 6)$ und $(2, 6)$ bilden, größer ist als π. Die Spitze sticht in das Face f_1. Entsprechend wird die Senke 7 dem äußeren Face f_2 zugewiesen.

Dabei ist n_f die Anzahl der Knoten, die Senken in f sind. Diese Senken werden auch *(Senken-)Switches* von f genannt. Hierbei ist zu beachten, das die Senken-Switches nicht notwendigerweise Senken in G sind.

Mit $\mathcal{A}(f)$ wird die Menge von Senken und Quelle bezeichnet, die dem Face f zugewiesen wird. Eine Zuweisung $\mathcal{A}$ in einer Zeichnung von Γ ist *aufwärtskonsistent*, wenn folgende Bedingung erfüllt sind:

1. Jede Quelle und Senke von G wird genau einem Face zugewiesen.
2. Die Anzahl $|\mathcal{A}(f)|$ der Zuweisungen eines Faces f ist gleich seiner Kapazität $c(f)$.

Der letzte Punkt wird auch *Kapazitätsgleichung* des Faces f genannt.

Eine aufwärtskonsistente Zuweisung ist nicht hinreichend, um eine aufwärtsplanare Einbettung zu charakterisieren. Notwendig ist die Bimodalität eines Graphen. Hierbei ist G *bimodal*, wenn für jeden Knoten von G die zirkuläre Aufzählung der inzidenten Kanten in höchstens zwei Mengen, nämlich die Menge der eingehenden und die Menge ausgehenden Kanten, partitionieren lässt. Die Menge der planaren, azyklischen Einbettungen eines bimodalen Graphen werden *Kandidaten* für eine aufwärtsplanare Einbettung genannt. Insbesondere ist eine planare Einbettung eines expandierten Digraphen ein Kandidat.

Lemma 2.4 (aus [6])**.** *Sei G ein Digraph und Γ_G eine Einbettung von G, so dass Γ_G ein Kandidat für eine aufwärtsplanare Einbettung ist. Sei f' ein Face von Γ_G und*

$\mathcal{A}$ eine Zuweisung. Wenn für jedes Face $f \neq f'$ die Anzahl der Zuweisungen $\mathcal{A}(f)$ gleich der Kapazität $c(f)$ ist, dann ist die Anzahl der Zuweisungen von f' durch $\mathcal{A}$ gleich $c(f')$.

Satz 2.7 (aus [6]). *Sei Γ eine Einbettung eines Digraphen G und ein Kandidat für eine aufwärtsplanare Einbettung. Die Einbettung Γ ist genau dann aufwärtsplanar, wenn sie eine aufwärstkonsistente Zuweisung $\mathcal{A}$ besitzt.*

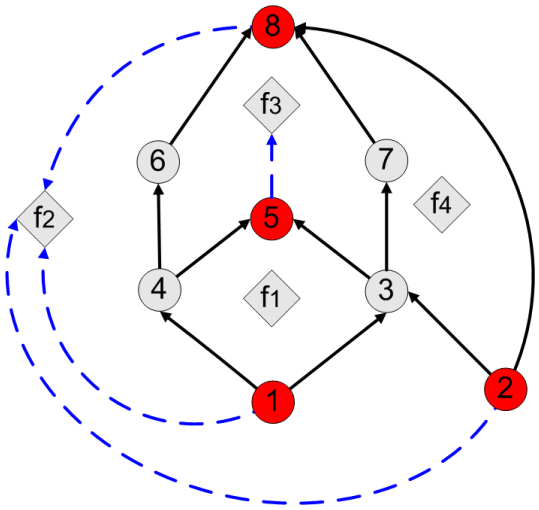


Abbildung 2.10: Beispiel einer aufwärtskonsistenten Zuweisung.

Die Abbildung 2.10 zeigt einen aufwärtsplanaren Graphen mit einer aufwärtskonsistenten Zuweisung. Das Face f_2 besitzt zwei Senken-Switches, nämlich die Knoten 3 und 8. Da f_2 äußeres Face ist, ist $c(f_2) = 3$. Ähnlich lassen sich die Kapazitäten der anderen Faces berechnen. Man erhält dann $c(f_1) = 0$, $c(f_3) = 1$ und $c(f_4) = 0$. Die blauen Kanten in der Abbildung zeigen die Zuweisungen der Quellen und Senken des Graphen zu den jeweiligen Faces.

2.4.2 Satz zum Aufwärtsplanaritätstest von sT-Graphen

In diesem Abschnitt geben wir die Beweise zu zwei Lemmata wieder, die zum zentralen Satz 2.8 führen. Aus diesem Satz wird dann der Aufwärtsplanaritätstest für sT-Graphen abgeleitet.

Nach Lemma 2.2 genügt es zu zeigen, dass der Expansionsgraph eines azyklischen sT-Graphen aufwärtsplanar ist. Die Einbettungen eines azyklischen Expansionsgraphen sind automatisch Kandidaten für eine aufwärtsplanare Einbettung und erfüllen somit eine notwendige Voraussetzung. Durch die Restriktion des In- und Outgrad eines Knotens, nämlich entweder $in(v) \leq 1$ oder $out(v) \leq 1$, besitzen expandierte Graphen eine eingeschränkte Struktur. Diese Einschränkung wird in den späteren Beweisen angewendet.

Lemma 2.5 (aus [7]). *Sei G ein sT-Graph und G' ein Digraph homöomorph zu einem Untergraphen G_{sub} von G. Dann gilt:*

1. *Ist G azyklisch, dann ist auch G' azyklisch.*
2. *Ist G expandiert, dann ist auch G' expandiert.*
3. *Ist G aufwärtsplanar, dann ist auch G' aufwärtsplanar.*
4. *Ist G_{sub} ein sT-Graph, dann ist auch G' ein sT-Graph.*

Die Graphen, die homöomorph zu einem Untergraphen eines sT-Graphen G sind, sind auch sT-Graphen und besitzen im Bezug auf Kreisfreiheit und Aufwärtsplanarität die gleichen Eigenschaften wie G. Diese Eigenschaft erlaubt den Rückschluss, dass wenn G' nicht aufwärtsplanar ist, dann ist auch G nicht aufwärtsplanar. Den Graph G' zu finden, der homöomorph zu einem Untergraphen von G ist, ist „umständlich“. Das gleich nachfolgende Lemma 2.8 zeigt, dass statt G' ein Minor von G genügt um den gleichen Rückschluss zu ziehen. Um dies zeigen zu können, werden noch einige Hilfslemmata benötigt.

Lemma 2.6 (aus [28]). *Seien $G = (V, A)$ ein Digraph und $e = (u, v) \in A$ mit $out(u) = 1$ und $in(v) = 1$. Sei ferner $G' = G/e$. Dann gilt:*

1. *Ist G azyklisch, dann ist auch G' azyklisch.*
2. *Ist G aufwärtsplanar, dann ist auch G' aufwärtsplanar.*

Lemma 2.7 (aus [7]). *Sei Γ eine aufwärtsplanare Einbettung eines aufwärtsplanaren sT-Graphen $G = (V, A)$ und $e = (s, u) \in A$ eine Kante des äußeren Faces α von Γ. Ferner sei P ein Valley mit $u \leftarrow s' \rightarrow s$ und $G' = G/e \cup P$. Dann ist G' ein sT-Graph und hat eine aufwärtsplanare Einbettung Γ' mit P eingebettet am äußeren Face α.*

In Abbildung 2.11 ist der Sachverhalt von Lemma 2.7 dargestellt. Die gestrichelte Kante (s, u) wird durch das Valley $u \leftarrow s' \rightarrow s$ ersetzt. Der resultierende Graph ist wieder aufwärtsplanar.

Einer der Grundsteine zum Beweis von Satz 2.8 bildet das folgende Lemma. Informal lässt sich dieses Lemma wie folgt beschreiben: Eine Komponente K eines expandierten sT-Digraphen G_{exp} wird durch eine virtuelle Kante d_K ersetzt. Das Ergebnis ist ein Minor G_{minor} von G_{exp}. Wenn G_{exp} aufwärtsplanar ist, dann ist es auch der Minor G_{minor}. Mehr noch, der Minor ist wie G_{exp} ein expandierter sT-Graph. Wenn die Quelle s von G_{exp} in K lokalisiert ist, dann ist offensichtlich, dass die virtuelle Kante d_K, nach außen gezeichnet werden muss. In dem Beweis wird die „Form“ von K als eine Struktur P_K identifiziert. Es stellt sich heraus, dass es

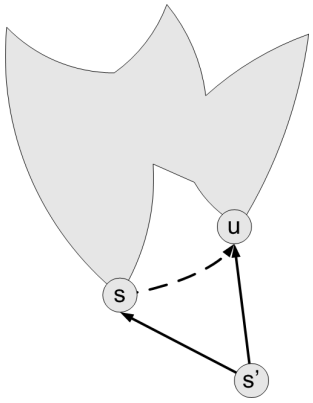


Abbildung 2.11: Illustration zu Lemma 2.7.

reicht, P_K als Peak, gerichtete Kante, Valley oder Zig-Zag zu identifizieren. Bei der Minorenkonstruktion genügt es K stellvertretend durch ein Peak oder eine gerichtete Kante zu ersetzen. Aus diesem Grunde werden in den Ersetzungsregeln $\mathcal{R}_1$ bis $\mathcal{R}_4$ nur Peaks und gerichtete Kanten verwendet.

Lemma 2.8 (aus [7]). *Sei G_{exp} ein Expansionsgraph eines sT-Graphen G, $\{u, v\}$ ein Split-Paar von G_{exp} und K eine Split-Komponente des Split-Paares. Sei $H := G - K$ und sei $d_K := d(K, H)$ eine gerichtete virtuelle Kante, die mit K assoziiert ist und $H' := H \cup d_K$. Dann gilt:*

(a) H' ist ein azyklischer, expandierter sT-Graph.

(b) Ist G_{exp} aufwärtsplanar, dann ist auch H' aufwärtsplanar.

(c) Ist G_{exp} aufwärtsplanar mit der Quelle $s \in K^\circ := K - \{u, v\}$, dann hat H' eine aufwärtsplanare Einbettung mit d_K am äußeren Face.

Beweis aus [7]. Es sind nur vier Fälle möglich. Jeder Fall korrespondiert zu den Ersetzungsregeln $\mathcal{R}_1$-$\mathcal{R}_4$. Für jeden der vier Fälle wird nun gezeigt, dass (a)-(c) gilt. Dabei ist in allen vier Fällen der Beweis zu (c) trivial, weil bei einer aufwärtsplanaren Einbettung von G_{exp} die Quelle s stets am äußeren Face eingebettet ist.

$\mathcal{R}_1$: u und v sind Quellen in K
d_K ist der Peak $u \rightarrow \circ \leftarrow v$. Nach Lemma 2.3 enthält K eine Struktur P_K homöomorph zu einem Peak mit den Basisknoten u und v. Somit ist H' homöomorph zu einem Untergraphen von G_{exp} und Lemma 2.5 kann angewendet werden.

(a) Nach Lemma 2.5 ist H' ein expandierter, azyklischer Graph. Da G_{exp} ein sT-Graph ist, können u und v nicht beide gleichzeitig Quelle in H' sein. Also hat H' nur eine Quelle und ist somit ein sT-Graph.

(b) Nach Lemma 2.5 ist H' aufwärtsplanar.

$\mathcal{R}_{2a}$: u ist Quelle, v ist Senke in K und $s \notin K^\circ$
d_K ist die gerichtete Kante $u \to v$. Da $s \notin K^\circ$ und u die einzige Quelle in K ist, muss es eine Struktur $P_K := u \rightsquigarrow v$ in K geben. Diese ist homöomorph zu der gerichteten Kante (u, v). Somit ist H' homöomorph zu einem Untergraphen von G_{exp} und Lemma 2.5 kann angewendet werden.

(a) Nach Lemma 2.5 ist H' ein expandierter, azyklischer Graph. Wegen der Kante d_K ist v keine Quelle in H'. Aus G_{exp} ist ein sT-Graph folgt H' ist ein sT-Graph.

(b) Nach Lemma 2.5 ist H' aufwärtsplanar.

$\mathcal{R}_{2b}$: u ist Quelle, v ist Senke in K und $s \in K^\circ$
d_K ist der Peak $u \to \circ \leftarrow v$. Es ist $s \in K^\circ$ und $s \neq u$. Nach Lemma 2.3 existiert ein von s und u verschiedener Knoten t und zwei knotendisjunkte Pfade $s \rightsquigarrow t$ und $u \rightsquigarrow t$. Da u nicht Quelle in H sein kann, ist v die einzige Quelle von H. Es gibt somit den Pfad $s \rightsquigarrow v$ in K und den Pfad $v \rightsquigarrow u$ in H. Diese Fakten implizieren zwei knotendisjunkte Pfade $s \rightsquigarrow v$ und $u \rightsquigarrow t$.

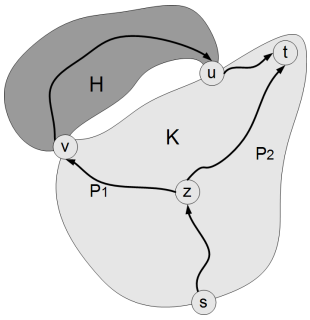


Abbildung 2.12: Abbildung zum Beweis von Punkt $\mathcal{R}_{2b}$.

Sei nun $P_1 := s \rightsquigarrow v$ und $P_2 = s \rightsquigarrow t$ und z der letzte gemeinsame Knoten der beiden Pfade. Dann existiert eine Struktur P_K. Dieser ist homöomorph zu dem Zig-Zag $u \to t \leftarrow z \to v$ (siehe Abbildung 2.12). Daraus folgt, dass in G_{exp} ein Untergraph existiert, der homöomorph zu $H'' := H \cup (z, v) \cup (z, t) \cup (u, t)$ ist. Wird nun eine Kontraktionsoperation auf die Kante (z, v) angewandt, so ist das Ergebnis $H''/(z, v) = H \cup d_K = H'$. Nach Lemma 2.6 ist H' wieder aufwärtsplanar.

(a) Nach Lemma 2.5 ist H'' ein expandierter azyklischer Graph. Zusammen mit Lemma 2.6 folgt, dass H' azyklisch ist. Nach Konstruktion haben alle Knoten außer v in H'' den gleichen In- und Outgrad und v ist (die einzige) Quelle in H'. Also ist H' auch expandiert.

(b) Nach Lemma 2.5 ist H'' aufwärtsplanar. Nach Lemma 2.6 ist somit auch H' aufwärtsplanar.

$\mathcal{R}_{3a}$: u ist Quelle, v ist innerer Knoten in K, $s \notin K^\circ$ und v ist Quelle in H
d_K ist die gerichtete Kante $u \to v$. u ist die einzige Quelle in K, somit gibt es den Pfad $u \rightsquigarrow v$ in K. Der Beweis für (a) und (b) läuft analog wie für $\mathcal{R}_{2a}$.

$\mathcal{R}_{3b}$: u ist Quelle, v ist innerer Knoten in K, $s \in K^\circ$ und v ist Quelle in H
d_K ist der Peak $u \to \circ \leftarrow v$. Die Komponente K enthält zwei Quellen s und v. Der Beweis ist analog zu $\mathcal{R}_{2b}$.

$\mathcal{R}_{3c}$: u ist Quelle, v ist innerer Knoten in K und nicht Quelle in H
Aus der Tatsache, dass v keine Quelle in H ist, folgt $s \notin K^\circ$ und u ist die einzige Quelle in K. Nach Lemma 2.3 enthält K eine Struktur P_K homöomorph zu einem Peak mit den Basisknoten u und v. Somit ist H' homöomorph zu einem Untergraphen von G_{exp} und Lemma 2.5 kann angewendet werden.

(a) Nach Lemma 2.5 ist H' ein expandierter, azyklischer Graph. Wenn u identisch mit s ist, dann hat H' nur eine Quelle. Wenn u nicht identisch ist mit s, dann ist er nicht Quelle in H und H' hat nur eine Quelle. Aus diesen beiden Tatsachen folgt, dass H' ein sT-Graph ist.

(b) Nach Lemma 2.5 ist H' aufwärtsplanar.

$\mathcal{R}_4$: u und v sind beide keine Quelle in K
Dann ist u oder v Quelle in H. Da die Rollen von u und v hier vertauschbar sind, wird ohne Beschränkung der Allgemeinheit angenommen, dass u die Quelle von H ist. d_K ist die gerichtete Kante $u \to v$. Folgende Fälle sind möglich:

$u \nleftrightarrow v$: Nach Lemma 2.3 enthält K eine Struktur P_K homöomorph zu einem Valley mit den Spitzen u und v. Also ist der Graph $H'' := H \cup (s,v) \cup (s,u)$ homöomorph zu G_{exp}. Durch Kontraktion von (s,u) erhält man $H'' = H \cup d_K = H'$.

(a) Nach Lemma 2.5 ist H'' azyklisch und expandiert. Zusammen mit Lemma 2.6 folgt, dass H' azyklisch ist. Nach Konstruktion haben alle Knoten außer u in H'' den gleichen In- und Outgrad und u ist (die einzige) Quelle in H'. Also ist H' auch expandiert.

(b) Nach Lemma 2.5 ist H'' aufwärtsplanar. Zusammen mit Lemma 2.6 folgt H' ist aufwärtsplanar.

$u \rightsquigarrow v$: Es gibt dann eine Struktur P_K in K homöomorph zu der gerichteten Kante (u,v).

(a) Nach Lemma 2.5 ist H' ein expandierter azyklischer Digraph. u ist die einzige Quelle in H und somit auch die einzige Quelle in $H' = H \cup d_K$.

(b) Nach Lemma 2.5 ist H' aufwärtsplanar.

$v \rightsquigarrow u$: Durch die Existenz des Pfades $v \rightsquigarrow u$ ist v Quelle in H, sonst gäbe es einen Zyklus in G_{exp}. Somit sind u und v Quellen in H.

(a) Da H azyklisch und expandiert ist und zwei Quellen u und v besitzt, ist $H' = H \cup d_K$ auch azyklisch und expandiert. Durch die Kante d_K ist v nicht Quelle in H'. Somit hat H' nur die Quelle u.

(b) Sei $H'' := H \cup (v, u)$. H'' ist homöomorph zu einem Untergraphen von G_{exp}. Nach Lemma 2.5 existiert eine Einbettung Γ von H'' mit (v, u) am äußeren Face. Ferner ist H'' ein expandierter sT-Graph mit Quelle v. Aus H'' wird nun $\widetilde{H}$ konstruiert, indem die Kante (v, u) durch den Valley $u \leftarrow s' \rightarrow v$ ersetzt wird. Der (Dummy-)Knoten s' wird zusätzlich in $\widetilde{H}$ eingefügt. H' wird nun aus $\widetilde{H}$ durch Kontraktion der Kante (s', u) gebildet. Diese Operationen ändern die aufwärtsplanaren Eigenschaften nicht (siehe Lemma 2.5 und 2.6). Aus Lemma 2.5 folgt $\widetilde{H}$ ist aufwärtsplanar. Zusammen mit Lemma 2.6 folgt H' ist aufwärtsplanar.

Somit folgt die Behauptung von Lemma 2.8. □

In Lemma 2.8 wird ein Minor konstruiert, indem nur eine Split-Komponente durch eine virtuelle Kante ersetzt wird. Durch Induktion über die Anzahl der Komponenten K_i kann ohne weiteres gezeigt werden, dass die Ergebnisse von Lemma 2.8 auch dann gilt, wenn mehrere Komponenten durch entsprechende virtuelle Kanten ersetzt werden.

Lemma 2.9 (aus [7]). *Sei G_{exp} ein expandierter sT-Graph und $K_1, \ldots, K_m$ die Split-Komponenten der Split-Paare $\{u_1, v_1\}, \ldots, \{u_l, v_l\}$ von G_{exp}. Ferner sei $G_{minor} := G - K_1 - \cdots - K_m \cup d_{K_1} \cup \cdots \cup d_{K_m}$ ein Minor von G mit $d_{K_i} := d(K_i, G - K_i)$. Dann gilt:*

(a) G_{minor} ist ein sT-Digraph.

(b) Ist G aufwärtsplanar, dann ist auch G_{minor} aufwärtsplanar.

(c) Ist G aufwärtsplanar mit der Quelle $s \in K_i^\circ$, für $1 \leq i \leq m$, dann hat G_{minor} eine aufwärtsplanare Einbettung mit d_{K_i} am äußeren Face.

Für den Beweis des zweiten Grundsteins für Satz 2.8, dem Lemma 2.13, wird eine Reihe von Hilfslemmata benötigt, die hier ohne Beweis angeben wird. Mit P_K wird die Struktur, die die „Form“ einer Graphkomponente widerspiegelt (siehe Beweis zu Lemma 2.8) bezeichnet und mit $\Gamma_H \subset \Gamma_G$ wird die Einbettung Γ_H der Komponente H, die in der Einbettung Γ_G enthalten ist, notiert.

Lemma 2.10 (aus [7]). *Sei G_{exp} ein planarer Expansionsgraph eines sT-Graphen G mit der Quelle s, $\{u, v\}$ ein Split-Paar von G_{exp}, K eine Split-Komponente des Split-Paares. Sei $H = G - K$, $d_K = d(K, H)$ und $d_H = d(H, K)$ die gerichtete virtuelle Kante von K bzw. H. Ferner sei $\widetilde{H} = H \cup P_K$ und $H' = H \cup d_K$ ein Minor von G_{exp} mit der planaren Aufwärtseinbettung $\Gamma_{H'}$, so dass d_K am äußeren Face von $\Gamma_{H'}$ eingebettet ist, falls $s \in K^\circ$ ist. Sei α_H ein Face der Einbettung Γ_H von H, die d_K enthält. Dann gilt:*

(a) $\widetilde{H}$ ist ein expandierter, azyklischer Digraph.

(b) $\widetilde{H}$ ist ein sT-Graph und besitzt eine planare Aufwärtseinbettung $\Gamma_{\widetilde{H}}$ mit $\Gamma_H \subset \Gamma_{\widetilde{H}}$ und P_K ist eingebettet in α_H.

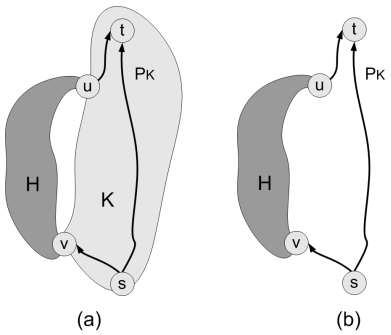


Abbildung 2.13: Illustration des Lemma 2.10: (a) der Graph G_{exp} mit den Komponenten H und K; (b) der Graph $\widetilde{H}$, bestehend aus H und der Struktur P_K;

Die Abbildung 2.13 illustriert Lemma 2.10. In (a) ist der Graph G_{exp} dargestellt. Die Komponente H hat die Einbettung Γ_H. In (b) ist die Komponente H mit der Einbettung Γ_H sowie die Struktur P_K dargestellt. Sie bilden den Graphen $\widetilde{H}$ mit der Einbettung $\Gamma_{\widetilde{H}}$.

Wir schauen uns die Split-Komponenten H und K eines Split-Paares $\{u, v\}$ eines planaren, expandierten sT-Graphen G_{exp} genauer an (siehe Abbildung 2.14). In den Komponenten von H und K gibt es jeweils eine Struktur P_H bzw P_K, die mit ihrer jeweiligen gerichteten virtuellen Kante $d_K = d(K, H)$ und $d_H = d(H, K)$ assoziiert sind. Diese beiden Strukturen haben die gemeinsamen Knoten u und v und bilden einen Kreis $C = P_K \cup P_H$. Sei wieder $\widetilde{H} = H \cup P_K$ und $\widetilde{K} = K \cup P_H$. Seien G^* der Teilgraph, der von C eingeschlossen wird, K^*, der Teilgraph in K von G^* plus die Struktur P_H, und H^*, der Teilgraph in H von G^* plus die Struktur P_K. Durch eine gegebene aufwärtsplanare Einbettung $\Gamma_{G_{exp}}$ sind die Einbettungen der Graphkomponenten von G_{exp} festgelegt. Diese aufwärtsplanaren Einbettungen der

Komponenten werden mit Γ_{komp} notiert und es gilt:

$$\Gamma_{komp} \subset \Gamma_{G_{exp}} \text{ für } komp \in \{\widetilde{H}, \widetilde{K}, H^*, K^*, G^*\}.$$

Eingeschlossen in C ist das Face γ_{G^*}. Dieses enthält die Knoten u und v (siehe Abbildung 2.14 (c)). Für die äußeren Faces α_{H^*} von H^*, α_{K^*} von K^* und α_{G^*} der entsprechenden Einbettung kann folgendes beobachtet werden:

$$\alpha_{H^*} = \alpha_{K^*} = \alpha_{G^*} = C = P_H \cup P_K.$$

Wir geben nun einige Fakten über die Komponenten H^*, K^* und G^* wieder.

Lemma 2.11 (aus [7]). *Sei s^* eine Quelle in K^* bzw. H^*, wie oben beschrieben. Dann gilt:*

(a) $s^* \in C$.

(b) Der Digraph K^* bzw. H^* ist ein expandierter sT-Graph.

Lemma 2.12 (aus [7]). *Sei Γ_{G^*} eine planare Einbettung von G^* mit P_H und P_K eingebettet am äußeren Face. Dann ist G^* ein expandierter Graph und Γ_{G^*} ist eine Aufwärtseinbettung. Insbesondere ist Γ_{G^*} eine aufwärtsplanare Einbettung.*

Den zweiten Grundstein zum Beweis von Satz 2.8 bildet das folgenden Lemma.

Lemma 2.13. *Sei G_{exp} ein expandierter, planarer sT-Graph mit der Quelle s, $\{u, v\}$ ein Split-Paar von G_{exp}, K eine Split-Komponente des Split-Paares mit $s \in K$. Sei $H = G - K$, $d_K = d(K, H)$ und $d_H = d(H, K)$ die gerichtete virtuelle Kante von K bzw. H. Wenn die folgenden Bedingungen gelten, dann ist G_{exp} aufwärtsplanar.*

1. *Der Minor $K' = K \cup d_H$ ist aufwärtsplanar.*
2. *Der Minor $H' = H \cup d_K$ besitzt eine planare Aufwärtseinbettung mit d_K am äußeren Face.*

Beweisskizze aus [7]. Nach Voraussetzung sind H' und K' aufwärtsplanar. Nach Lemma 2.10 sind auch die Einbettungen $\Gamma_{\widetilde{H}}$ und $\Gamma_{\widetilde{K}}$ (definiert wie oben) aufwärtsplanar. Aus diesen beiden Einbettungen kann eine Einbettung $\Gamma_{G_{exp}}$ für G_{exp} konstruiert werden mit $\Gamma_{G^*} \subseteq \Gamma_{G_{exp}}$. Mit Ausnahme von zwei Faces, die mit $\gamma_{G_{exp}}$ und $\beta_{G_{exp}}$ bezeichnet werden, können alle anderen Faces von $\Gamma_{G_{exp}}$ eindeutig zu $\Gamma_{\widetilde{K}}$ oder zu $\Gamma_{\widetilde{H}}$ zugewiesen werden. Das Face $\gamma_{G_{exp}}$ ist identisch mit dem Face γ_{G^*} von G^* und ist somit ein inneres Face (siehe Abbildung 2.14(c)). Die Knoten u und v sind die „Nahtstellen“ der Komponenten H und K. Um zu zeigen, dass $\Gamma_{G_{exp}}$ aufwärtsplanar ist, muss nach Lemma 2.7 gezeigt werden, dass eine konsistente aufwärtsplanare Zuweisung $\mathcal{A}_{G_{exp}}$ existiert. Diese Zuweisung lässt sich aus den Zuweisungen von $\Gamma_{\widetilde{H}}$, $\Gamma_{\widetilde{K}}$ und Γ_{G^*} konstruieren. Zusammen mit der Tatsache, das $\Gamma_{G_{exp}}$ ein Kandidat für

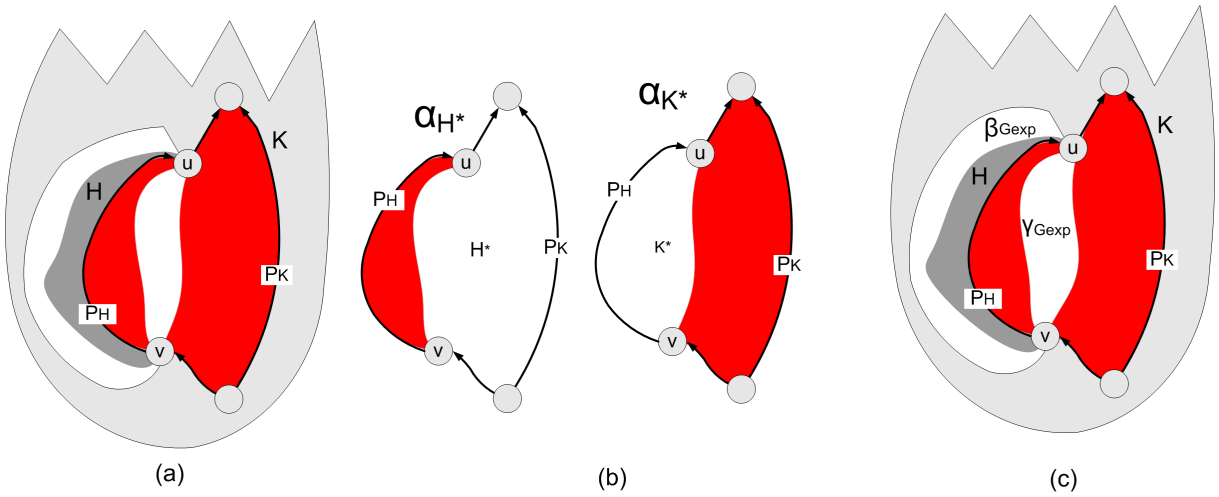


Abbildung 2.14: (a) eine planare Einbettung Γ_G des Graphen G mit den Split-Komponenten H und K von $\{u, v\}$. In H und K sind jeweils die Strukturen P_H und P_K eingezeichnet. Diese beiden Strukturen bilden einen ungerichteten Kreis C. Eingeschlossen in C ist G^* (rot gezeichnet). (b) die Einbettung Γ_{H^*} und Γ_{K^*}; Für die beiden äußeren Faces α_{H^*} und α_{K^*} gilt: $\alpha_{H^*} = \alpha_{K^*} = C = P_H \cup P_K$. (c) Darstellung der Faces $\gamma_{G_{exp}}$ und $\beta_{G_{exp}}$;

eine aufwärtsplanare Einbettung ist, folgt dann die Behauptung von Lemma 2.13.

Seien $\mathcal{A}_{\Gamma_{\widetilde{H}}}$, $\mathcal{A}_{\Gamma_{\widetilde{K}}}$ und $\mathcal{A}_{\Gamma_{G^*}}$ die aufwärtskonsistenten Zuweisungen der entsprechenden Graphkomponenten. Da die Einbettungen $\Gamma_{\widetilde{H}}$ und $\Gamma_{\widetilde{K}}$ aufwärtsplanar sind, muss es diese Zuweisungen geben. Dies gilt auch für G^*, da er nach Lemma 2.12 aufwärtsplanar ist. Aus den drei Zuweisungen wird eine Zuweisung $\mathcal{A}_{G_{exp}}$ für G_{exp} wie folgt konstruiert:

1. $\mathcal{A}_{G_{exp}}(\gamma_{G_{exp}}) = \mathcal{A}_{G^*}(\gamma_{G^*})$.
2. $\mathcal{A}_{G_{exp}}(f) = \mathcal{A}_{\widetilde{K}}(f)$ für alle $f \in \Gamma_{\widetilde{K}}$.
3. $\mathcal{A}_{G_{exp}}(f) = \mathcal{A}_{\widetilde{H}}(f)$ für alle $f \in \Gamma_{\widetilde{H}}$.
4. Die restlichen verbliebenen Senken und eventuell die Quelle s werden dann Face $\beta_{G_{exp}}$ zugewiesen.

Problematisch für die Zuweisung sind die Knoten u und v. Da sie Nahtstellen sind, fallen sie sowohl unter $\mathcal{A}_{G^*}$, unter $\mathcal{A}_{\widetilde{K}}$ als auch unter $\mathcal{A}_{\widetilde{H}}$. Somit können sie an mehreren Faces zugewiesen werden. Um zu zeigen, dass das nicht der Fall sein kann, muss die folgende drei Punkte bewiesen werden. Dann kann daraus gefolgert werden, dass $\mathcal{A}_{G_{exp}}$ aufwärtskonsistent ist[8].

(a) Falls u oder v Senken sind, so werden sie von $\mathcal{A}_{G_{exp}}$ zu genau einen einen Face f von $\Gamma_{G_{exp}}$ zugewiesen.

(b) Falls u oder v innere Knoten sind, werden sie von $\mathcal{A}_{G_{exp}}$ keinem Face zugewiesen.

(c) Die Anzahl der Zuweisungen für das Face f entspricht genau der Kapazität $c(f)$.

□

Unter bestimmten Voraussetzungen ist der Minor eines Minors eines Digraphen G wiederum ein Minor von G, wie das folgende Lemma zeigt.

Lemma 2.14 (aus [7]). *Sei G_{minor} ein Minor eines Digraphen G, $\{u, v\}$ ein Split-Paar von G, das auch Split-Paar von G_{minor} ist. Seien die Quellen von G auch Quellen von G_{minor}. Ferner sei $\widetilde{K}$ eine Split-Komponente von G_{minor} des Split-Paares $\{u, v\}$ und K eine zu $\widetilde{K}$ korrespondierende Komponente, die man erhält, indem alle gerichteten virtuellen Kanten von $\widetilde{K}$ durch ihre entsprechenden Komponenten in G ersetzt werden. Es gilt dann $d(\widetilde{K}, G_{minor} - \widetilde{K}) = d(K, G - K)$*

[8] Der Beweis kann in [7] nachgelesen werden.

Wir haben nun alle Lemmata zusammen, die zum Beweis des folgende Satzes nötig sind.

Satz 2.8. *Sei G_{exp} der Expansionsgraph eines sT-Graphen G und $\mathcal{T}$ der SPQR-Baum von G_{exp}. G ist genau dann aufwärtsplanar, wenn $\mathcal{T}$ an einem Q-Knoten μ gewurzelt werden kann, so dass folgendes gilt:*

1. *μ enthält die Quelle s von G und*
2. *alle sT-Skeletons von $\mathcal{T}$ haben eine aufwärtsplanare Einbettung mit der virtuellen Referenzkante am äußeren Face.*

Der Beweis der Richtung „⇒“ geht aus Lemma 2.9 hervor. Die Rückrichtung „⇐“ lässt sich mit Induktion über die Knotenanzahl des SPQR-Baumes $\mathcal{T}$ mit Hilfe von Lemma 2.13 und Lemma 2.14 zeigen.

Aus diesem Satz wird im nächsten Abschnitt der Algorithmus zum Testen von Aufwärtsplanarität eines sT-Graphen abgeleitet.

2.4.3 Algorithmus zum Aufwärtsplanaritätstest

Der Algorithmus 3 zum Testen von Aufwärtsplanarität von sT-Graphen wird aus dem Satz 2.8 abgeleitet. Als Eingabe benötigt er einen 2-zusammenhängenden sT-Graphen. Ein allgemeiner sT-Graph kann ohne weiteres in Blöcke zerlegt werden. Sind alle seine Blöcke aufwärtsplanar, dann ist auch der sT-Graph aufwärtsplanar.

Im Algorithmus wird zuerst der Eingabegraph $G = (V, A)$ zu einem Expansionsgraph G_{exp} transformiert. G_{exp} wird dann auf die notwendigen Bedingungen der Kreisfreiheit und der Planarität getestet. Der Expansionsprozess und der Test auf Kreisfreiheit benötigt eine Laufzeit von $\mathcal{O}(|V + |E|)$. Der Planaritätstest ist nach [26] in $\mathcal{O}(|V| + |E|)$ realisierbar. Nach dem Testen der notwendigen Bedingungen wird der SPQR-Baum $\mathcal{T}$ von G_{exp} mit den sT-Skeletons konstruiert. Dieser Prozess benötigt eine Laufzeit von $\mathcal{O}(|V| + |E|)$.

Im Algorithmus werden die S-, P- und Q-Knoten nicht auf Aufwärtsplanarität getestet. Der Grund hierfür ist, dass in einem planaren, azyklischen, expandierten sT-Graphen die sT-Skeletons immer aufwärtsplanar sind und die virtuelle Referenzkante immer nach außen gezeichnet werden kann.(Für eine detailierte Analyse der Knoten siehe Kapitel 4.) Nur für die R-Knoten wird ein Aufwärtsplanaritätstest vollzogen. Für sie gilt: Die Anzahl der Einbettungen eines R-Knotens beträgt genau zwei und die Summe der Knoten in allen R-Skeletons ist höchstens $\mathcal{O}(|V|)$. Nach Satz 2.3 lässt sich der Aufwärtsplanaritätstest für alle R-Knoten in $\mathcal{O}(|V| + |E|)$ bewerkstelligen.

Durch das Markieren der virtuellen Kanten in einem R-Knoten μ_R werden die Kanten kenntlich gemacht, die nach außen gezeichnet werden können. Nach Satz 2.8 können nur sie Referenzkanten sein. Folglich dürfen die $\mathcal{T}$-Knoten ν_R^i, die mit den

unmarkierten Kanten e_i assoziiert sind, keine Elternknoten von μ_R sein. Dementsprechend werden die Kanten von $\mathcal{T}$, die mit e_i assoziiert sind, so gerichtet, dass die ν_R^i Kinder von μ_R sind.

Die Knoten von $\mathcal{T}$, deren Skeleton s nicht enthält, müssen Kinder eines benachbarten Knotens sein, deren pertinenter Graph s enthält. Wenn $\mathcal{T}$ an einem Q-Knoten μ_Q, mit $s \in skeleton(\mu_Q)$, gewurzelt wird, dann erzwingt diese Bedingung zusammen mit der oben beschriebenen $\mathcal{T}$-Kantenausrichtung, dass in der Referenzkante jedes $\mathcal{T}$-Knotens κ mit $s \notin skeleton(\kappa)$ stets s lokalisiert ist. Besitzen die $\mathcal{T}$-Kanten nicht eine einheitliche Ausrichtung, dann kann eine Referenzkante nicht nach außen gezeichnet werden. Der Eingabegraph ist folglich nicht aufwärtsplanar.

Eine Überprüfung, ob so eine $\mathcal{T}$-Kantenausrichtung und Wurzelung existiert, kann in einer Laufzeit von $\mathcal{O}(|V|+|E|)$ realisiert werden. Somit hat der Algorithmus eine Gesamtlaufzeit von $\mathcal{O}(|V|+|E|)$.

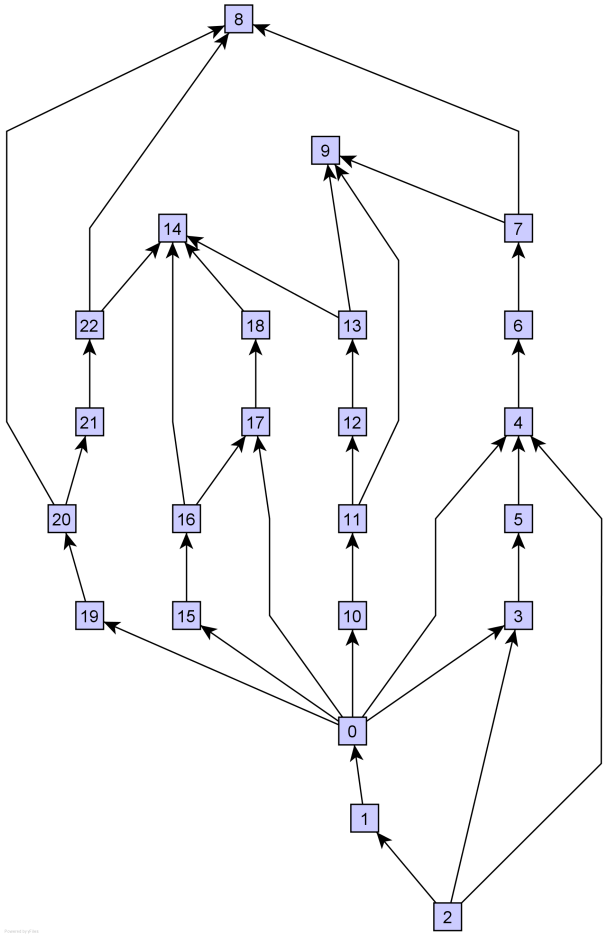


Abbildung 2.15: Ein aufwärtsplanarer, expandierter sT-Graph.

Algorithmus 3 aus [7]. Aufwärtsplanaritätstest für 2-zusammenhängenden sT-Graphen.

Eingabe: 2-zusammenhängender sT-Graph G mit Quelle s.
Ausgabe: **true** oder **false**
function IsUpwardPlanar(G)
    Konstruiere Expansionsgraph G_{exp} von G;
    if G_{exp} nicht planar **or** nicht azyklisch **then**
        return false;
    end if
    Konstruiere den SPQR-Baum $\mathcal{T}$ von G_{exp};
    for each virtuelle Kante e eines Skeletons **do**
        Identifiziere die Endpunkte von e als Quelle, Senke oder inneren Knoten in dem pertinenten Graph von e;
        Merke, ob der pertinente Graph von e die Quelle s enthält;
    end for
    Berechne die sT-Skeletons;
    for all R-Knoten μ **do**
        Wähle eine beliebige Einbettung Γ von $skeleton(\mu)$;
        if Γ ist nicht aufwärtsplanar **then**
            return false;
        end if
        Markiere die virtuellen Kanten von $skeleton(\mu)$,
        deren Endpunkte im äußeren Face einer planaren Aufwärtszeichnung
        $skeleton(\mu)$ liegen können.
        for each nicht markierte virtuelle Kante e **do**
            Richte die zu e assoziierte Kante in $\mathcal{T}$, so
            dass sie μ als Zielknoten hat;
        end for
    end for
    for each Knoten μ von $\mathcal{T}$ **do**
        if $s \notin \mu$ **then**
            Sei ν der Nachbarknoten von μ, dessen pertinenter Graph s enthält.
            Setze ν als Zielknoten der Kante μ und ν;
        end if
    end for
    Berechne, ob $\mathcal{T}$ an einem Q-Knoten μ_Q, mit $s \in \mu_Q$, gewurzelt werden kann, so dass alle Kanten von $\mathcal{T}$ eindeutig zur Wurzel ausgerichtet sind.
    if es gibt so eine Wurzelung **then**
        return true;
    else
        return false;
    end if
end function

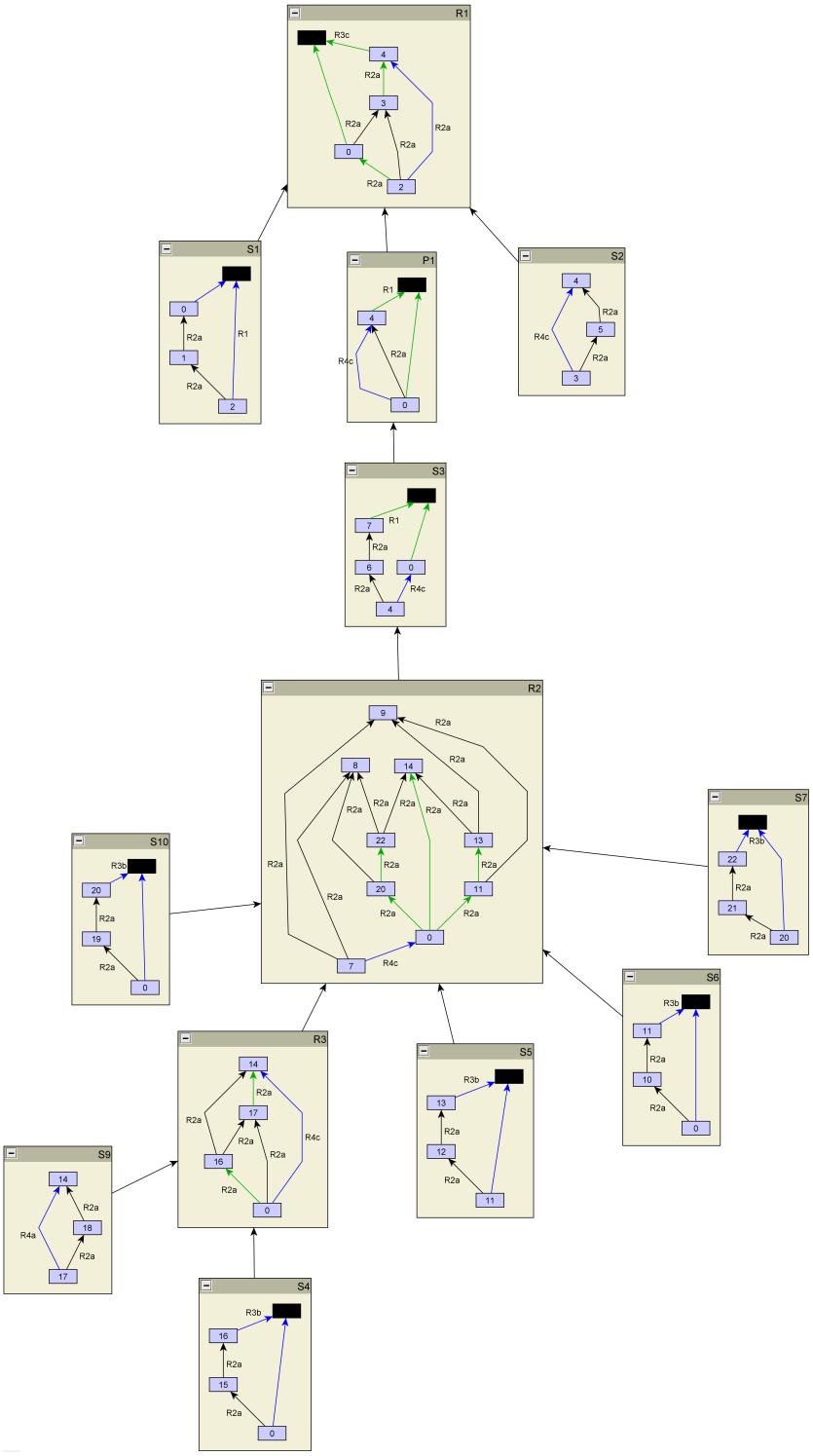


Abbildung 2.16: Der SPQR-Baum mit *sT*-Skeletons zum Beispielgraphen aus der Abbildung 2.15.

Abbildung 2.16 zeigt einen SPQR-Baum mit sT-Skeletons des Beispielgraphen aus der Abbildung 2.15. Der Baum ist an der Kante $(2, 4)$ gewurzelt. Die Q-Knoten des Baumes wurden aus Übersichtsgründen weggelassen. Die virtuelle Referenzkante ist in blau gezeichnet. Die Kanten, die mit einem Q-Knoten assoziiert sind, sind schwarz gezeichnet und die anderen gerichteten virtuellen Kanten in grün gehalten. Neben den gerichteten virtuellen Kanten stehen die entsprechenden Ersetzungsregeln. Die schwarzen Knoten stellen die Dummy-Knoten der Peaks dar.

2.5 Das Zeichenverfahren von Sugiyama

Das Zeichenverfahren von Sugiyama [35, 13] ist das klassische Verfahren zum Zeichnen von Digraphen. Das Verfahren ist in mehrere Phasen unterteilt. In diesen Phasen werden bestimmte ästhetische Kriterien optimiert. Einige dieser Phasen beinhalten NP-schwere Probleme, zu deren Lösung eine Vielzahl von Heuristiken existieren. Deshalb spricht man auch von einem Framework für das Zeichnen von Digraphen. Das Framework ist in Algorithmus 4 angegeben. Die einzelnen Phasen werden in den nachfolgenden Abschnitten erläutert.

Algorithmus 4 Sugiyamas Framework zum Zeichnen von Digraphen.

Phase 1: Entfernen von Kreisen

Phase 2: Schichtenzuweisung

Phase 3: Kreuzungsminimierung

Phase 4: Zuweisung der horizontalen Positionen der Knoten

2.5.1 Entfernen von Kreisen

Der Eingabedigraph wird in dieser Phase kreisfrei gemacht, da die späteren Phasen einen azyklischen Digraphen voraussetzen. Um dies zu erreichen, wird eine Menge von Kanten M_{rev} berechnet, durch deren Umkehrung der Digraph azyklisch wird. Dieser Umkehrprozess wird nach Phase 4 wieder rückgängig gemacht. Um eine gute Zeichnung zu bekommen, sollte die Kardinalität von M_{rev} möglichst klein bleiben. Dieses Problem ist auch als *feedback arc set* bekannt und ist NP-schwer [20]. Zur Lösung dieses Problems bieten sich folgende Heuristiken an:

Sei $G = (V, A)$ für die folgenden Beschreibungen ein zusammenhängender Digraph.

Tiefensuche-Heuristik [13]: G wird mittels einer Tiefensuche durchlaufen. Dabei werden die Kanten von G in zwei Mengen, die Menge der Vorwärstkanten

und die Menge der Rückwärstkanten, partitioniert. Die Rückwärtskanten werden dann umgekehrt. Diese Heuristik zeichnet sich durch ihre Einfachheit und Schnelligkeit (Laufzeit $\mathcal{O}(|V|+|A|)$) aus. Die Ergebnisse sind jedoch nicht sehr befriedigend.

Greedy-Heuristik [12]: Die zentrale Idee der Greedy-Heuristik ist die Annahme, dass Knoten mit hohem Outgrad eher weiter unten gezeichnet werden als Knoten mit geringem Outgrad. Dadurch erhofft man sich, dass höher gelegene Knoten weniger Kanten besitzen, die zu einer tiefer gelegenen Kante zeigen.

Die Knoten des Graphen werden nach ihrem Outgrad sortiert. Gemäß der Sortierungsreihenfolge werden die Knoten nur mit ausgehenden Kanten nacheinander in einem zu Beginn leeren Graphen hinzugefügt. Die Kanten, die zu einem Knoten mit einer höheren Sortierungsnummer führen, werden umgekehrt. Der so konstruierte Graph enthält keine Kreise. Die Laufzeit dieses Verfahrens ist in der Größenordnung $\mathcal{O}(|V| + |A|)$. Die Ergebnisse dieses Verfahrens sind deutlich besser als die der Tiefensuche-Heuristik.

Divide und Conquer Heuristik [12]: Ähnlich wie bei dem Greedy-Verfahren wird auch hier eine Reihenfolge der Knoten ermittelt, indem jeder Knoten eine Prioritätsnummer aus $1, \ldots, |V|$ zugewiesen bekommt. Gemäß dieser Reihenfolge wird dann wie bei dem Greedy-Verfahren ein kreisfreier Graph konstruiert. Die Reihenfolge wird rekursiv wie folgt ermittelt: Der Graph G wird in zwei disjunkte Mengen G_1 und G_2 partitioniert, wobei für alle Knoten $u \in G_1$ und $v \in G_2$ gilt: $out(u) \geq out(v)$. Ist die Kardinalität von V gerade, so wird der Graph G in zwei gleich große Mengen eingeteilt. Ist sie ungerade, so besteht die Menge G_1 aus genau einem Knoten v_{max}. v_{max} besitzt den größten Outgrad von allen Knoten von G. Die Menge G_2 besteht aus den restlichen Knoten. Dem Knoten v_{max} wird die größte noch verfügbare Prioritätsnummer zugewiesen. Das ganze wird dann rekursiv auf G_1 und G_2 fortgesetzt, bis die Mengen leer sind.

Dieses Verfahren ist etwas langsamer als die oben beschriebenen Verfahren. Tests haben ergeben, dass bei dünnen Graphen[9] die Laufzeit um den Faktor vier langsamer ist. Die Menge M_{rev} enthält im Schnitt 20% weniger Kanten als die Greedy-Heuristik.

Zum Schluß dieses Abschnitts sei noch angemerkt, dass für die Klassen der planaren Digraphen das *feedback arc set* Problem in einer Laufzeit von $\mathcal{O}(|V|^5)$ optimal lösbar ist [17].

[9] Als *dünne Graphen* wird hier Graphen bezeichnet, deren Kantenanzahl in der Größenordnung ihrer Knoten liegen.

2.5.2 Schichtenzuweisung

Nachdem die Kreise des Graphen entfernt wurden, ist die Aufgabe in Phase 2, die Knoten möglichst gleichmäßig auf der Zeichenfläche zu verteilen und gleichzeitig darauf zu achten, dass die Zeichnung nicht unnötig viel Zeichenfläche einnimmt. Ein zu großer Zeichenflächenverbrauch birgt die Gefahr, dass Knoten zu weit auseinander liegen. Die Verfolgung der Kanten durch das menschliche Auge wird dadurch erschwert.

Um eine Verteilung der Knoten auf der Zeichenfläche systematisch gestalten zu können, benutzt Sugiyama das Schichtlayout. In einem *Schichtlayout* werden die Knoten eines Digraphen $G = (V, A)$ in h Mengen(Schichten) $L_1, \ldots, L_h$ partitioniert, so dass für jede Kante $e = (u, v)$ gilt: Ist $u \in M_i$ und $v \in L_j$ dann ist $i < j$. Die *Höhe* des Graphen ist h. Die *Breite* w ist die größte Kardinalität der L_i, $i \in \{1, \ldots, h\}$. Die *Spannweite* von e ist die Anzahl der Schichten, die zwischen u und v liegen. Ein Graph ist *proper*[10] wenn alle Kanten eine Spannweite von eins haben. Dies wird erreicht, indem Dymmy-Knoten in Kanten mit einer Spannweite größer als eins eingefügt werden. Die Aufgabe, eine Partitionierung von G in $L_1, \ldots, L_h$ zu finden, die den obigen Anforderungen genügt, ist NP-schwer [13]. Das Problem kann nämlich auf das *Multiprocessor-Scheduling Problem* reduziert werden. Die Aufgabe hierbei ist es n Aufgaben an w Prozessoren so zu verteilen, dass sie alle in der Zeit h erledigt werden können. Folgende Heuristiken werden für die Verteilung der Knoten auf die Schichten herangezogen:

Längste Pfad Heuristik [13]: Hier werden alle Senken von G in der obersten Schicht (L_h) platziert. Alle anderen Knoten werden gemäß ihrem längsten Abstand zu den Senken in den entsprechenden Schichten platziert. Der Abstand ist hier die Anzahl der Kanten des längsten Pfades.

Der Vorteil dieses Verfahrens ist, dass mit Hilfe einer Tiefensuche auf G die Lösungen berechnet werden können. Ferner ist die Höhe des Graphen minimal, da alle Knoten stets den kürzesten Abstand zur Senke haben. Das Verfahren erzeugt jedoch Graphen mit hoher Breite, da sich viele Knoten an den unteren Schichten ansammeln.

Coffman-Graham-Heuristik [9]: Die Coffman-Graham-Heuristik wurde ursprünglich für das *multiprocessor scheduling* Problem entwickelt. Wegen der nahen Verwandschaft dieses Problems mit der Suche nach einer passende Partitionierung kann die Heuristik ohne weiteres zur Lösung der Schichtenzuweisung herangezogen werden. Die Coffman-Graham-Heuristik berechnet für jeden Knoten eine Prioritätsnummer. Die Knoten mit vielen Vorgängern bekommen eine hohe und Knoten mit wenigen Vorgängern bekommen eine niedrige Prioritätsnummer. Nach Vergabe der Prioritätsnummer für alle Knoten und nach

[10] englisch: proper=korrekt, ordentlich

Festlegung der maximalen Breite w bekommt jeder Knoten eine Schicht zugewiesen. Die Zuweisung läuft wie folgt: Weise, beginnend mit $i = h$, dem Knoten v mit der größten Prioritätsnummer, dessen Vorgänger noch nicht gezeichnet wurden, die Schicht L_i zu. Im Falle, dass L_i schon w Knoten zugewiesen bekommen hat, weise v die Schicht L_{i-1} zu und fahre iterativ fort mit $i = i - 1$ bis alle Knoten eine Zuweisung bekommen haben.
Nach [31] gilt für die Höhe des von der Coffman-Graham-Heuristik berechneten Graphen die folgende Ungleichung:

$$h \leq (2 - \frac{2}{w})h_{min}$$

Dabei ist h_{min} die minimale Höhe des geschichteten Graphen, dessen maximale Schichtbreite w beträgt.

Neben der Abschätzung für die Höhe des geschichteten Graphen hat diese Heuristik noch den Vorteil, dass die Breite w als Parameter eingegeben werden kann. Die Heuristik hat jedoch den Nachteil, dass es viele Dummy-Knoten erzeugt. Negativ wirkt sich auch die Laufzeit aus, da die Methode einen Graphen voraussetzt, der keine transitiven Kanten enthält. Um dies zu erreichen muss eine Transformation durchgeführt werden, die eine Laufzeit von $\mathcal{O}(|V|^2)$ mit sich bringt. Das gesamte Verfahren wird von dieser Laufzeit dominiert.

Methode von Gansner et al. [18]: Die Dummy-Knoten entstehen, wenn eine Kante eine Spannweite von mehr als eins hat. Dann wird für jede Schicht, die die Kante überspannt, ein Dummy-Knoten erzeugt. Sie blähen den Graphen unnötig auf und verlangsamen dadurch die Berechnung. In der endgültigen Zeichnung werden Dummy-Knoten nicht gezeichnet. Stattdessen werden sie durch potentielle Knickpunkte ersetzt. Eine Verfolgung der Kante durch das Auge kann dadurch erschwert werden.

Die Methode von Gansner et al. setzt hier an und berechnet eine Schichtenzuordnung, so dass der resultierende geschichtete Graph eine minimale Anzahl von Dummy-Knoten besitzt. Die Minimierung der Anzahl der Dummy-Knoten ist gleichbedeutend mit der Minimierung der Länge der Kanten. Die Autoren benutzen hierzu Lineare Programmierung und formulieren das Problem wie folgt: Sei $y(v)$ eine Funktion, die jedem Knoten $v \in G$ eine y-Koordinate zuweist. Für diese Funktion gilt:

- $y(v)$ ist eine ganze Zahl für alle v.
- $y(v) \geq 1$ für alle v.
- $y(u) - y(v) \geq 1$ für alle Kanten (u, v).

Die Funktion $f(y) = \sum_{u \to v}(y(u) - y(v) - 1)$ gibt dann die Summe aller Längen der Kanten an. Gesucht wird eine Funktion $y(v)$, die f minimiert. Gansner

et al. haben mit einer Variante des Simplexverfahrens, das in der Praxis als effizient gilt, dieses Problem gelöst.

2.5.3 Kreuzungsminimierung

Ein Graph mit vielen Überkreuzungen von Kanten ist unübersichtlich. Eine Minimierung der Kreuzungen ist notwendig, um eine gute Zeichnung zu ermöglichen. Die Anzahl der Kreuzungen hängt nicht von der Position der Knoten ab, sondern von ihrer Reihenfolge in den jeweiligen Schichten. Die Aufgabe in dieser Phase ist eine Ordnung der Knoten für alle Schichten zu finden, so dass der Graph eine minimale Anzahl von Kantenkreuzungen besitzt. Dieses Problem ist als *crossing number* bekannt und ist NP-schwer. Es sei angemerkt, dass bereits das Minimieren der Kantenkreuzungen für zwei Schichten NP-schwer ist [19]. Lösungsansätze für dieses Problem basieren meistens auf lokaler Optimierung. Lokal bedeutet hier, dass nur zwei benachbarte Schichten betrachtet werden. Die Kreuzungen zwischen diesen beiden Schichten werden dann minimiert. Hierbei geht man wie folgt vor: Zuerst legt man die Knotenordnung für eine Schicht L_i fest. Die Ordnung der benachbarten Schicht L_{i+1} wird dann berechnet, so dass die Kreuzungsanzahl zwischen den beiden Schichten möglichst gering ist. Dieses Verfahren, *layer by layer sweep* genannt, wird für $i \in \{1, \ldots, h-1\}$ angewandt und solange iteriert bis keine Verbesserung mehr erkennbar ist. Für die Kreuzungsminimierung zwischen zwei Schichten werden dann z.B. folgende Heuristiken benutzt:

Barycenter-Heuristik [34]: Diese Heuristik sowie die nachfolgend beschriebene Median-Heuristik, basieren auf der intuitiven Annahme, dass jeder Knoten möglichst „nahe“ bei seinem Vorgänger- bzw. Nachfolgerknoten liegen sollte. Hat zum Beispiel ein Knoten $v \in L_i$ m Nachfolgerknoten in der Schicht L_{i+1} mit den Barycenter-Werten $x_1, \ldots, x_m$, so ist der neue Barycenter-Wert von v der Durchschnitt $\frac{x_1+\cdots+x_m}{m}$. Die neue Ordnung der Knoten in der Schicht L_{i+1} wird durch die sortierten Barycenter-Werte bestimmt.

Median-Heuristiks [14]: Die Median-Heuristik basiert auf der gleichen Idee wie die Barycenter-Heuristik. Jedoch wird statt der Durchschnittskoordinate die Mediankoordinate der Nachfolgerknoten benutzt. Die Barycenter- und Median-Heuristik zeichnen sich durch ihre Einfachheit und Schnelligkeit aus [11, 32]. Tests haben gezeigt, dass für dichte Graphen die Ergebnisse nahe am Optimum liegen [15].

Knotentausch-Heuristik [11]: Hier werden Paare von Knoten in der selben Schicht vertauscht und dann ihre Wirkung auf die Anzahl der Kreuzungen festgestellt. Die Vertauschung wird rückgängig gemacht, wenn keine Verbesserung festgestellt werden kann. Dies wird solange durchgeführt bis kein Paar mehr existiert, das zu einer Verbesserung beiträgt. Da die Anzahl verschiedener Paare genau

$\binom{|V|}{2}$ beträgt, ist der Aufwand nicht allzu groß.
Die Experimente mit diesem Verfahren lassen darauf schließen, dass es sehr gute Ergebnisse mit zufällig erstellten Graphen produziert.

Hybrid-Methode [32, 18]: Die Hybrid-Methode kombiniert alle drei genannten Heuristiken. Zuerst werden die Knoten nach ihrem Median geordnet. Die Barycenter-Heuristik wird angewandt, falls mehrere Knoten mit dem gleichen Median existiert. Zuletzt wird die Knotentausch-Heuristik benutzt. Dieses Verfahren hat sich in der Praxis sehr bewährt.

Jünger et al. haben verschiedene Heuristiken sowie Methoden zur exakten Berechnung der Knotenordnung für die 2-Schicht-Kreuzungsminimierung untersucht [29]. Es stellt sich heraus, dass eine optimale Lösung selbst für große Instanzen (eine Schicht mit bis zu 60 Knoten) schnell und ohne Heuristik berechnet werden kann.

2.5.4 Zuweisung der horizontalen Positionen

In Phase 2 wird jeden Knonten v die y-Koordinate berechnet, indem v einer Schicht L_i zugewiesen wird. Die Ordnung der Knoten in den Schichten wurde in Phase 3 berechnet.

In dieser Phase werden nun die x-Koordinaten der Knoten berechnet, ohne die Ergebnisse der vorherigen Phasen zunichte zu machen. Gleichzeitig sollte die Zeichnung durch die Platzierung der Knoten kompakt sein. Schwierigkeiten bereiten in dieser Phase die Kanten mit großer Spannweite. Damit die Kanten möglichst gerade gezeichnet werden, müssen geeignete x-Koordinaten für die Dummy-Knoten gefunden werden.

Die Abbildung 2.17 gibt eine Illustration dieser Sachverhalte wieder. In (a) ist eine Zeichnung mit nicht optimaler x-Koordinatenzuweisung dargestellt. Dem gegenüber steht die Zeichnung (b) mit verbesserter x-Koordinatenzuweisung.

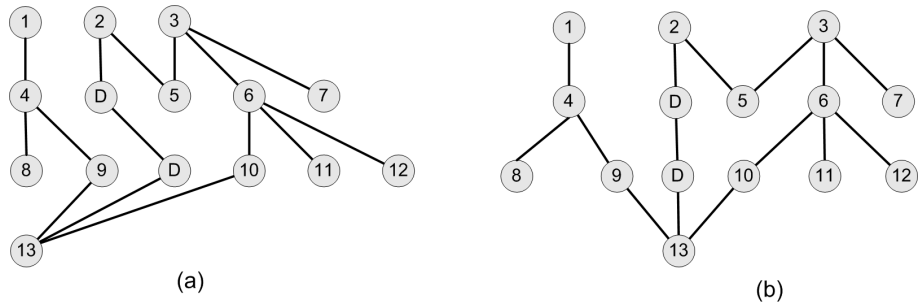


Abbildung 2.17: Phase 4 des Sugiyama-Algorithmus: (a) eine Zeichnung ohne Optimierung der x-Koordinaten; (b) eine Zeichnung, deren x-Koordinaten mit der QP-Methode ermittelt wurden.

Im Folgenden wird eine Lösung zur Berechnung der Koordinaten vorgestellt:

QP-Layout Methode [35]: Das Kürzel QP steht für *Quadratic Programming*. Bei diesem Verfahren wird das Problem als Optimierungsproblem formuliert und mit den Methoden der quadratischen Programmierung gelöst. Die Methode ist jedoch rechenintensiv.

Sei v ein Knoten in der Schicht L_i, $a_1, \ldots, a_k \in L_{i+1}$ die Nachfolger und $b_1, \ldots, b_j \in L_{i-1}$ die Vorgänger von v. Mit x_v wird die x-Koordinate von Knoten v notiert. Der Abstand von v zu einem seiner Nachfolger a_i wird definiert als $(x_v - x_{a_i})^2$. Sei f_1 die Funktion, die die Summe dieser Abstände über alle Knoten des Graphen berechnet. Der *untere Schwerpunkt* x^v_{us} von v ist der Durchschnitt $x^v_{us} = (x_{a_1} + \cdots + x_{a_k})/k$ der Koordinaten der Vorgänger von v. Entsprechend ist der *obere Schwerpunkt* x^v_{os} der Durchschnitt der Koordinaten der Nachfolger. Der Abstand von v zum unteren Schwerpunkt wird definiert als $(x^v_{us} - x_v)^2$, wenn $in(v) > 1$, ansonsten null. Der Abstand von v zum oberen Schwerpunkt wird definiert als $(x^v_{os} - x_v)^2$, wenn $out(v) > 1$, ansonsten null. Sei f_2 die Funktion, die die Summe beider Abstände über alle Knoten berechnet. Die Aufgabe der Optimierung ist nun die Funktion $f = cf_1 + (1 - c)f_2$, mit $0 < c \leq 1$, zu minimieren. Die Minimierung von f_1 bewirkt eine Minimierung der Abstände der Knoten. Die Minimierung von f_2 bewirkt, dass der Knoten v nicht allzu weit von seinen Nachfolgern und Vorgängern gezeichnet wird. Bei der Optimierung müssen dabei einige Randbedingungen beachtet werden, wie z.B. die Ausdehnung der Knoten oder der minimale Abstand. Diese Randbedingungen können als Bedingung der Optimierung formuliert werden.

2.5.5 Vor- und Nachteile

Das Zeichenverfahren von Sugiyama zeichnet sich durch seine Universalität aus. Es kann alle Digraphen Klassen zeichnen. Die Einteilung des Verfahrens in vier Phasen erlaubt eine große Flexibilität. Hinzu kommt, dass für jede Phase genügend bewährte und schnelle heuristische Lösungensmethoden existieren. In der Praxis hat sich das Verfahren als recht effizient erwiesen. Hinzu kommt, dass keine bessere Alternativen zum Zeichnen von Digraphen existieren, die praktikabel sind.

Eine der großen Schwächen des Verfahrens ist die Schichtung des Graphen. Durch die Schichtung können zusätzliche Kreuzungen entstehen. Selbst wenn in der Phase 3 eine optimale Lösung gefunden wird, bleiben diese Kreuzungen in der endgültigen Zeichnung bestehen. Dies verdeutlicht Abbildung 2.18. (a) zeigt eine Zeichnung eines Graphen mit minimaler Kreuzungsanzahl. (b) zeigt eine Zeichnung mit Schichtung bei Anwendung der *Längste Pfad Heuristik*. Die Kreuzung in (b) lässt sich nicht mehr durch Permutation der Knoten in den jeweiligen Schichten entfernen.

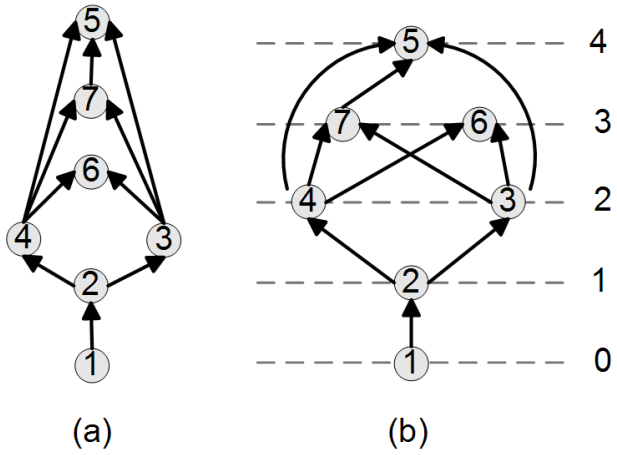


Abbildung 2.18: Nachteil durch die Schichtung bei Sugiyama-Verfahren: (a) eine kreuzungsfreie Zeichnung; (b) eine Zeichnung mit einer Schichtung der Knoten;

3 Aufgabenstellung und Problemanalyse

In diesem Kapitel wird eine Beschreibung des zu lösenden Problems und dessen Einordnung in das Gebiet des automatischen Zeichnens von Graphen angegeben. Es wird eine Untersuchung der Eigenschaften und Strukturen von Graphkomponenten durchgeführt. Insbesondere wird auf die Spiegelung der Einbettungen einer Komponente eingegangen.

3.1 Motivation und Problembeschreibung

In Abschnitt 2.5 wurde die klassische Zeichenmethode von Sugiyama vorgestellt. Eine alternative Zeichenmethode für DAGs bietet der Ansatz der Aufwärtsplanarisierung. Dieser Ansatz lässt sich in zwei Phasen einteilen:

1. Berechne aus dem Ursprungsgraphen G einen aufwärtsplanaren Untergraphen G_{up}, bei dem eine möglichst kleine Anzahl von Kanten, die Kreuzungen verursachen, entfernt werden.

2. Füge die entfernten Kanten wieder nacheinander so ein, dass der Graph, der durch Ersetzung der Kreuzungen durch Dummy-Knoten entsteht, wieder aufwärtsplanar ist.

In dieser Diplomarbeit konzentrieren wir uns auf die Aufwärtsplanarisierung von sT-Graphen. Genau genommen untersuchen wir Phase 2 des Aufwärtsplanarisierungsprozesses. Die Aufgabe (oder auch Problem) dieser Phase wird *Edge-Insertion-Problem* genannt. Formal lässt sich das Problem wir folgt beschreiben.

Problem 3.1 (Edge-Insertion)**.** *Gegeben sei ein zusammenhängender, aufwärtsplanarer sT-Graph $G = (V, A)$ mit der Quelle s und ein Paar von Knoten (x, y) mit $x, y \in V$. Finde eine aufwärtsplanare Einbettung Γ_{min} von G, so dass das Einfügen der Kante $e = (x, y)$ in Γ_{min} eine minimale Anzahl an Kreuzungen unter allen möglichen aufwärtsplanaren Einbettungen verursacht.*

Für die Phase 2 wird meistens ein Schichtungsverfahren benutzt, welches dem des Sugiyma-Ansatzes (siehe z.B. [16]) ähnelt. Hierbei wird zuerst eine Schichtung des

aufwärtsplanaren Untergraphen durchgeführt, danach wird ein passender Routing-Graph konstruiert. In diesem Routing-Graphen wird der kürzeste Einfügepfad berechnet. Dieses Verfahren wird für alle einzufügenden Kanten wiederholt. Das Resultat ist ein aufwärtsplanarer Graph. Die Schwäche des Verfahrens liegt in der Schichtung des Untergraphen. Wie beim Sugiyama-Ansatz kann die Schichtung unnötige Kreuzungen erzeugen. Hier setzt unsere Diplomarbeit an. Wir werden einen Ansatz zur Berechnung eines Einfügepfades entwickeln, der nicht auf einer Schichtung basiert.

Die Motivation zur Untersuchung des *Edge-Insertion-Problem* leitet sich nicht nur aus der Tatsache ab, dass das Problem eine wichtige Rolle in der Aufwärtsplanarisierung einnimmt. Zusätzliche Motivation bieten auch die Resultate aus den Arbeiten von Gutwenger et al. (siehe Kapitel 2.3 und [25]). Sie haben das *Edge-Insertion-Problem* für ungerichteten Graphen untersucht und in Experimenten gezeigt, dass durch das von ihnen entwickelte Verfahren durchschnittlich eine relative Verbesserung der Kreuzungsanzahl von 14,4% erzielt werden kann[1], was merklich zu Verbesserung der Zeichnung eines Graphen beiträgt.

3.2 Problemanalyse

Im Gegensatz zum ungerichteten Fall (siehe Abschnitt 2.3) genügt es im gerichteten Fall nicht einen kreuzungsminimalen Einfügepfad zu berechnen. Zusätzlich muss noch die Randbedingung gelten, dass der Einfügepfad die Aufwärtsplanarität des Ursprungsgraphen nicht zerstören darf. Einen Einfügepfad mit dieser Eigenschaft nennen wir einen *aufwärtskonsistenten Einfügepfad*. Durch diese Restriktion können wir an einem einfachen Beispiel zeigen, dass die Traversierungskosten einer Split-Komponente K eines Split-Paares $\{u, v\}$ eines Digraphen G nicht unabhängig sind von der Einbettung von G. Dieses Beispiel ist in Abbildung 3.1 illustriert. In (a) ist der ungerichtete Fall dargestellt. Hier ist der optimale Einfügepfad unabhängig von der Einbettung der Komponente K. Es werden drei Kanten gekreuzt. Der gleiche Pfad würde in einem Digraphen die aufwärtsplanaren Eigenschaften zerstören. In (b) ist der optimale Einfügepfad im gerichteten Fall dargestellt. Es werden hier vier Kanten gekreuzt. In (c) wurde die Einbettung von K aus der Abbildung (b) geändert. Die Anzahl der gekreuzten Kanten beträgt fünf. Die Traversierung einer Graphkomponente in einem Digraph kann folglich von der Einbettung abhängen. Zusätzlich kann sie davon abhängen, ob ein Pfad von x nach y berechnet werden soll, oder von y nach x. Dies verdeutlicht Abbildung 3.1 (d), wo der Pfad von y nach x verläuft. In (e) ist ein Beispiel dargestellt, wo der optimale Einfügepfad im gerichteten Fall immer mehr Kosten als im ungerichteten Fall verursacht. Da K eine 3-zusammenhängende Komponente ist, gilt dies sogar für alle aufwärtsplanare Einbettungen von K. Ferner ist K symmetrisch, so dass die Kosten von x nach y

[1] Als Testgraphen diente die Graphensammlung von Di Battista et al.[1]

gleich den Kosten von y nach x sind. Somit existiert kein Pfad von x nach y und umgekehrt, der weniger Kosten verursacht als im ungerichteten Fall.

Es existieren auch aufwärtsplanare Einbettungen, die keine aufwärtskonsistenten Einfügepfade besitzen (siehe Abbildung 3.2). Hier muss erst eine geeignete Einbettung gefunden werden. Die Tatsache, dass es exponentiell viele aufwärtsplanare Einbettungen geben kann, erschwert das Problem zusätzlich. Um eine Lösung zu finden, ist eine Betrachtung über alle aufwärtsplanaren Einbettungen nötig. Hierfür wird eine ähnliche Datenstruktur wie den SPQR-Baum für planare 2-zusammenhängende Graphen benötigt.

Angenommen, eine geeignete aufwärtsplanare Einbettung mit einer Vielzahl an aufwärtskonsistenten Einfügepfaden wurde gefunden. Dann stellt sich die Frage, wie man nun die aufwärtskonsistenten Einfügepfade von den nicht aufwärtskonsistenten Einfügepfaden unterscheidet. Insbesondere sind wir an der Berechnung des kürzesten aufwärtskonsistenten Einfügepfades für diese Einbettung interessiert.

Diese Überlegungen führen zu drei Problemen, die gelöst werden müssen.

1. Wie kann eine Datenstruktur zur Aufzählung aller aufwärtsplanaren Einbettungen konstruiert werden?
2. Wie kann aus den vielen aufwärtsplanaren Einbettungen eine geeignete Einbettung gefunden werden?
3. Wie kann ein kürzester Einfügepfad für eine gegebene Einbettung berechnet werden?

Dabei lassen sich die letzten beiden Probleme vermutlich nicht getrennt voneinander betrachten.

3.3 Eigenschaften der Split-Komponenten

Die sT-Skeletons werden konstruiert, indem die Split-Komponenten durch Peaks oder gerichtete Kanten ersetzt werden. Jede Split-Komponente ist dabei eindeutig mit einer der Ersetzungsregeln assoziiert. In diesem Abschnitt untersuchen wir diese Komponenten genauer. Insbesondere gehen wir auf die Möglichkeit der Spiegelung der einzelnen Komponenten ein. Wir beginnen mit der Struktur der Split-Komponente.

3.3.1 Struktur der Split-Komponenten

Sei G_{exp} ein expandierter, 2-zusammenhängender sT-Graph mit der Quelle s, $\{u, v\}$ ein Split-Paar von G_{exp}, K eine Split-Komponente des Split-Paares und $H := G_{exp} -$

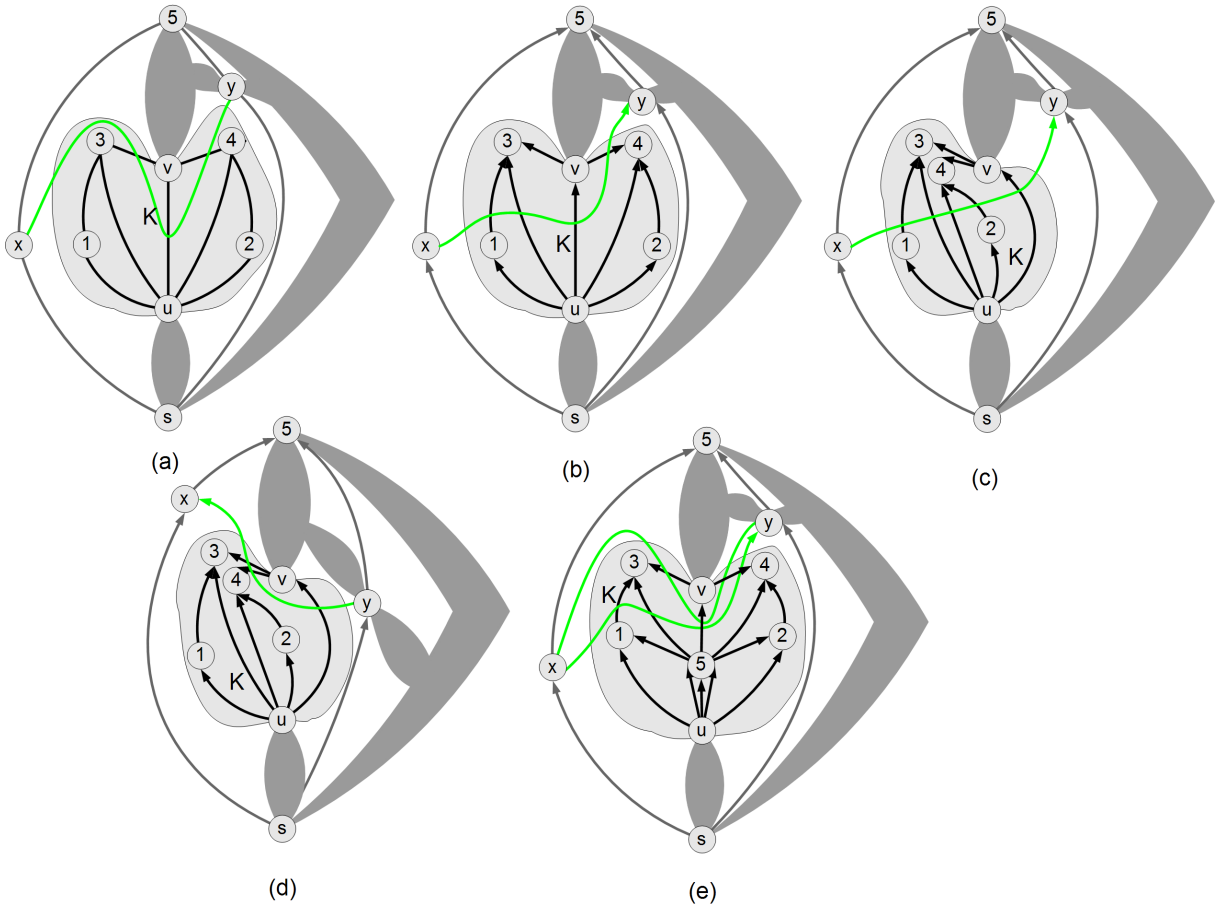


Abbildung 3.1: Abhängigkeit der Traversierungskosten einer Graphkomponente von der Einbettung: Die ausgemalten Graphkomponenten symbolisieren sehr dichte Teilgraphen, deren Traversierung hohe Kosten verursachen. (a) der ungerichtete Fall; (b) ein optimaler Einfügepfad von x nach y; (c) der gleiche Graph wie in (b) nur mit einer anderen Einbettung; (d) die gleiche Einbettung wie (c), wobei der Pfad von y nach x verläuft; (e) Ein Beispiel, wo der optimale Einfügepfad im gerichteten Fall immer mehr kostet als im ungerichteten Fall, egal ob die Kante von x nach y oder umgekehrt gerichtet ist.

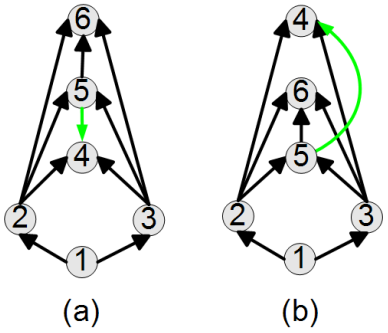


Abbildung 3.2: (a) eine Einbettung ohne aufwärtskonsistente Einfügepfade; (b) Durch die Wahl einer anderen Einbettung existiert nun ein aufwärtskonsistenter Einfügepfad.

K. Ferner sei $K^\circ := K - \{u, v\}$.

Wir untersuchen nun die Split-Komponenten des Split-Paares $\{u, v\}$. Dabei werden die Komponenten nach den Ersetzungsregeln $\mathcal{R}_1$ bis $\mathcal{R}_4$ klassifiziert. Die Ersetzungsregel, mit der die gerichtete virtuelle Kante e bzw. die Komponente K assoziiert ist, wird mit $rule(e)$ bzw. $rule(K)$ notiert.

$rule(K) = \mathcal{R}_1$:
u und v sind beide Quellen der Komponente K (siehe Abbildung 3.3 (R1)).

- $\mathcal{E}_{1a}$: Es gibt eine Struktur P_K in K homöomorph zu dem Peak $u \to t \leftarrow v$ (siehe Kapitel 2.4).
- $\mathcal{E}_{1b}$: Die Quelle s ist in H. Wäre sie in K°, so dominiere s nicht u und v.
- $\mathcal{E}_{1c}$: Da u und v beide Quellen in K sind, muss in K folglich $u \nleftrightarrow v$ gelten.

$rule(K) = \mathcal{R}_{2a}$:
u ist Quelle und v ist Senke in der Komponente K. Die Quelle s ist nicht in K° lokalisiert (siehe Abbildung 3.3 (R2a)).

- $\mathcal{E}_{2a}$: Es gibt eine Struktur P_K in K homöomorph zu der Kante $u \to v$ (siehe Kapitel 2.4).
- $\mathcal{E}_{2b}$: u dominiert alle Knoten in K, da er die einzige Quelle in K ist.
- $\mathcal{E}_{2c}$: Es gibt keinen Pfad homöomorph zu einen Peak $u \to \circ \leftarrow v$ in K, da v eine Senke in K ist.

$rule(K) = \mathcal{R}_{2b}$:
u ist Quelle und v ist Senke in der Komponente K. Die Quelle s ist in K° lokalisiert (siehe Abbildung 3.3 (R2b)).

$\mathcal{E}_{2d}$: Es gibt eine Struktur P_K in K homöomorph zum Zig-Zag $u \to t \leftarrow z \to v$ mit $t, z \in K$ und eventuell $z = s$ (siehe Kapitel 2.4).

$\mathcal{E}_{2e}$: Es gilt $s \nrightarrow u$ in K, da u eine Quelle in K ist.

$\mathcal{E}_{2f}$: Es gilt $v \rightsquigarrow u$ in H, sonst dominiert s nicht u.

$\mathcal{E}_{2g}$: u ist keine Quelle in H. Dies geht aus $\mathcal{E}_{2f}$ hervor.

$\mathcal{E}_{2h}$: Es gilt $u \not\leftrightarrow v$ in K. Dies geht aus der Eigenschaft $\mathcal{E}_{2f}$ zusammen mit der Voraussetzung, dass v keine Senke in K ist, und der Kreisfreiheit von G_{exp} hervor.

$\mathcal{E}_{2i}$: Es gibt keinen Pfad in K homöomorph zu einem Peak $u \to \circ \leftarrow v$, da v eine Senke in K ist.

$\mathcal{E}_{2j}$: v ist die einzige Quelle in H und dominiert alle Knoten von H. Wäre dies nicht der Fall, dann besäße H keine Quelle.

$rule(K) = \mathcal{R}_{3a}$:
u ist Quelle und v ist innerer Knoten der Komponente K. Zusätzlich ist v noch Quelle in H und s ist nicht in K° lokalisiert (siehe Abbildung 3.3 (R3a)).

$\mathcal{E}_{3a}$: Es gibt einen Pfad in K homöomorph zum Peak $u \to t \leftarrow v$, $t \in K$ (siehe Kapitel 2.4) sowie einen Untergraphen homöomorph zum „Dreieck“ $t \leftarrow u \to v \to t$.

$\mathcal{E}_{3b}$: u ist die einzige Quelle in K und dominiert alle Knoten in K.

$\mathcal{E}_{3c}$: Es gilt $u \nrightarrow v$ in H, da v eine Quelle in H ist.

$\mathcal{E}_{3d}$: Es gibt eine Struktur P_H homöomorph zu dem Peak $s \to \circ \leftarrow v$ in H, weil v eine Quelle von H ist und somit mindestens einen Knoten dominiert. Dieser wird auch von s dominiert.

$rule(K) = \mathcal{R}_{3b}$:
u ist Quelle und v innerer Knoten der Komponente K. Die Quelle s ist in K° lokalisiert (siehe Abbildung 3.3 (R3b)).

$\mathcal{E}_{3e}$: Es gibt eine Struktur P_K in K homöomorph zum Zig-Zag $u \to t \leftarrow z \to v$ mit $t, z \in K$. Dabei kann $z = s$ sein. (siehe Kapitel 2.4).

$\mathcal{E}_{3f}$: Es gibt den Pfad $v \rightsquigarrow u$ in H, sonst kann s nicht u dominieren.

$\mathcal{E}_{3g}$: Es gilt $u \not\leftrightarrow v$ in K. Existiert ein Pfad $u \rightsquigarrow v$, so gäbe er zusammen mit $\mathcal{E}_{3f}$ ein Kreis in G_{exp}. Da u eine Quelle in K ist, gibt es den Pfad $v \rightsquigarrow u$ in K nicht.

$\mathcal{E}_{3h}$: v ist die einzige Quelle von H und dominiert alle Knoten von H, da u wegen $\mathcal{E}_{3f}$ keine Quelle in H ist.

$rule(K) = \mathcal{R}_{3c}$:
u ist Quelle und v innerer Knoten der Komponente K. Zusätzlich ist v nicht Quelle in H (siehe Abbildung 3.3 (R3c)).

$\mathcal{E}_{3i}$: Es gibt einen Pfad in K homöomorph zum Peak $u \to t \leftarrow v$, $t \in K$ (siehe Kapitel 2.4).

$\mathcal{E}_{3j}$: v ist Senke in H, da er nach Voraussetzung keine Quelle ist und durch die Ingrad/Outgrad Restriktion eines Expansionsgraphen kein innerer Knoten in H sein kann.

$\mathcal{E}_{3k}$: u ist die einzige Quelle in K und dominiert alle Knoten in K.

$\mathcal{E}_{3l}$: Es gibt einen Pfad $u \rightsquigarrow v$ in K, da u die einzige Quelle und v ein innerer Knoten in K ist.

$\mathcal{E}_{3m}$: Es gibt einen Pfad $s \rightsquigarrow v$ in H. Trivial, da v eine Senke in H ist.

$rule(K) = \mathcal{R}_{4}$:
u und v sind nicht Quellen der Komponente K (siehe Abbildung 3.3 (R4a), (R4b) und (R4c).

$\mathcal{E}_{4a}$: u oder/und v sind Quellen in H.

3.3.2 Spiegelung einer Graphkomponente

In diesem Abschnitt werden wir untersuchen, unter welchen Umständen eine Komponente eines 2-zusammenhängenden sT-Graphen bei einer gegebenen aufwärtsplanaren Einbettung gespiegelt werden kann, so dass die aus der Spiegelung resultierende Einbettung aufwärtsplanar ist.

Seien G_{exp}, $\{u, v\}$, H und K wie im letzten Abschnitt definiert. Sei $\Gamma_{G_{exp}}$ eine aufwärtsplanare Einbettung von G_{exp} und $\mathcal{A}$ eine aufwärtskonsistente Zuweisung der Einbettung $\Gamma_{G_{exp}}$. Durch $\Gamma_{G_{exp}}$ werden auf H und K die Einbettungen $\Gamma_H, \Gamma_K \subset \Gamma_{G_{exp}}$ induziert. Im Folgenden wird das äußere Face von Γ_H mit α_H und das äußere Face von Γ_K mit α_K notiert.

Die Komponente K wird durch zwei Faces F^l und F^r von H getrennt (siehe Abbildung 3.4). F^l lässt sich in zwei Face-Abschnitte f^l_H und f^l_K unterteilen. f^l_H enthält nur Kanten aus H und f^l_K enthält nur Kanten aus K ohne die Kanten, die u und v als Endknoten besitzen. Entsprechend wird Face F^r in Face-Abschnitten f^r_H und f^r_K unterteilt. Somit sind die Knoten u und v nur in den Face-Abschnitten f^l_H und f^r_H enthalten.

Sei Γ'_K die Einbettung von K, die aus der Spiegelung der Einbettung Γ_K resultiert. (Eine Spiegelung von Γ_K kann durch die Umkehrung sämtlicher zirkulärer Kantenaufzählungen der Knoten in der Einbettung Γ_K durchgeführt werden.) Sei $\Gamma'_{G_{exp}}$ die

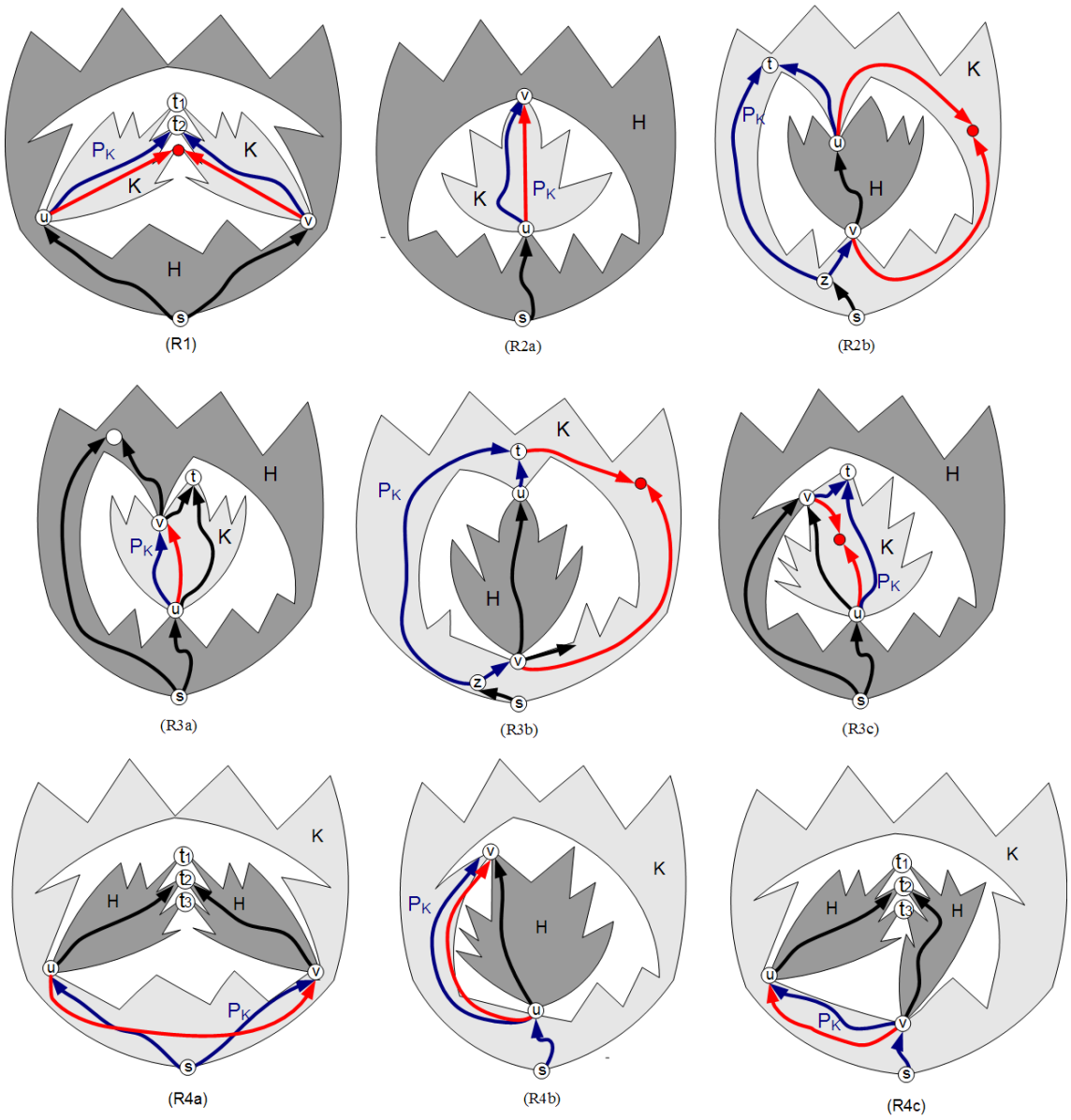


Abbildung 3.3: Illustration der Graphkomponenten: Jede Abbildungen ist mit einer entsprechenden Regel assoziiert. Die gerichteten virtuellen Kanten sind in rot gezeichnet. Die Abbildungen (R4a), (R4b) und (R4c) zeigen den Fall wenn in K $u \nleftrightarrow v$, $u \rightsquigarrow v$ und $v \rightsquigarrow u$ gilt. In allen drei Fällen ist u eine Quelle von H.

resultierende Einbettung aus $\Gamma_{G_{exp}}$, bei der nur die Einbettung Γ_K gespiegelt wird. Das nach der Spiegelung zu F^l bzw. F^r korrespondierende Face wird mit F'^l bzw. F'^r beschrieben. Face F'^l besteht aus der Kantensequenz von f_H^l und der umgekehrten Kantensequenz $f_K'^r$ von f_K^r. Face F'^r besteht aus der Kantensequenz von f_H^r und der umgekehrten Kantensequenz $f_K'^l$ von f_K^l.

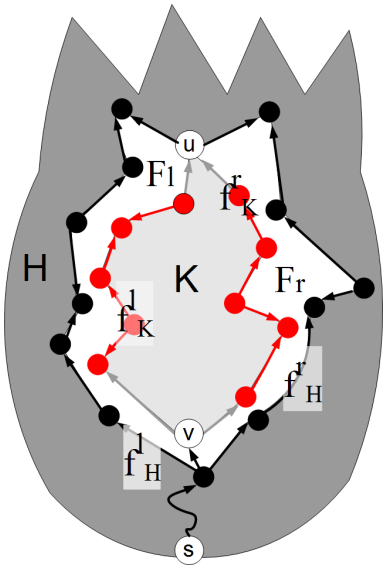


Abbildung 3.4: Illustration der Face-Notationen: Zu beachten ist, dass die Rollen von H und K vertauscht werden können und sowohl F_l als auch F_r können äußeres Face sein.

Im letzten Abschnitt wurde festgestellt, dass die Kosten des Einfügepfades von Knoten x nach Knoten y davon abhängen, ob der gerichtete Pfad von x nach y oder von y nach x führt. Bezogen auf die Traversierung einer Graphkomponente K bedeutet das, dass die Traversierungskosten von K davon abhängen, ob K von F^l nach F^r oder umgekehrt traversiert wird. Das ist dann der Fall, wenn K eine Struktur P_K beinhaltet, die homöomorph ist zu einem Peak $u \rightarrow t \leftarrow v$ mit $t \in K$. Dann kann dieser Pfad in zwei Teilstücke partitioniert werden. Der erste Teilpfad enthält alle Knoten, die von u dominiert werden, der zweite enthält die restlichen Knoten. Je nach dem in welches Face die Spitze des Peaks hinein sticht, kann einer der beiden Teilpfade nicht traversiert werden. Anschaulich verdeutlicht dies die Abbildung 3.5. Der Peak $v \rightarrow 3 \leftarrow u$ sticht in das Face F_r hinein. Alle Kanten der Komponente K können ohne weiteres vom Einfügepfad gekreuzt werden. Sticht die Spitze hingegen in Face F^l, darf der Einfügepfad z.B. nicht die Kante $v \rightarrow 3$ kreuzen. Die eventuell kostengünstigeren Einfügepfade können dadurch blockiert werden. Um das zu verhindern, wird die Komponente K so gespiegelt, dass die Kanten, die vorher nicht

passierbar waren, wieder vom Einfügepfad gekreuzt werden. Die Abbildung 3.5 illustriert diese Überlegung. In dieser Abbildung wird die Einbettung aus der Abbildung 3.1 (c) gespiegelt. Statt fünf werden hier nur drei Kanten gekreuzt.

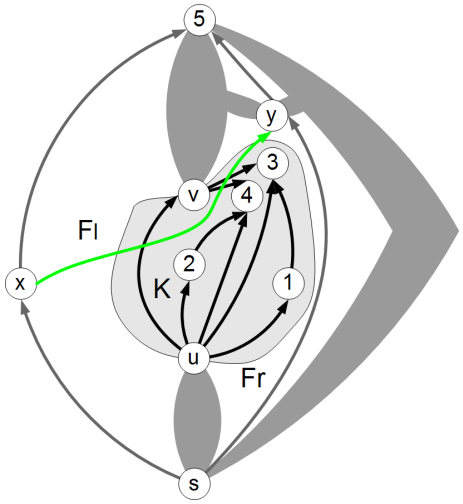


Abbildung 3.5: Minimierung der Traversierungskosten durch Spiegelung der Einbettung Γ_K aus Abbildung 3.1 (c).

Es stellt sich nun die Frage, welche Graphkomponenten bei einer gegebenen Einbettung gespiegelt werden können, so dass die daraus resultierende Einbettung aufwärtsplanar bleibt. Um dieser Frage nachgehen zu können, sind einige Vorüberlegungen notwendig.

Vorüberlegungen zur Spiegelung von Graphkomponenten

Wir beginnen mit einige Fakten und Vorüberlegungen über die Faces einer aufwärtsplanaren Einbettung $\Gamma_{G_{exp}}$. Wie wir sehen werden, spielt die Anzahl der Senken in einem Face eine wichtige Rolle bei der Entscheidung, ob eine Graphkomponente gespiegelt werden kann oder nicht.

Sei $\alpha_{G_{exp}}$ ein äußeres Face und $\beta_{G_{exp}}$ ein inneres Face der Einbettung $\Gamma_{G_{exp}}$ (siehe Abbildung 3.6 (a) und (b)). Durch eine einfache Überlegung können wir folgendes feststellen (siehe auch [7]):

Beobachtung 3.1. *Für äußere und innere Faces einer aufwärtsplanaren Einbettung $\Gamma_{G_{exp}}$ gilt:*

(a) Alle Senken-Switches im äußeren Face $\alpha_{G_{exp}}$ sind auch Senken in G_{exp}. Diese Senken werden von $\mathcal{A}$ dem Face $\alpha_{G_{exp}}$ zugewiesen (siehe Abbildung 3.6 (a)).

(b) In einem inneren Face $\beta_{G_{exp}}$ gibt es genau ein ausgezeichnete *Senken-Switch t der nicht an $\beta_{G_{exp}}$ zugewiesen wird (siehe Abbildung 3.6 (b)).*

(c) Alle Senken-Switches ungleich t von $\beta_{G_{exp}}$ sind Senken in G_{exp}.

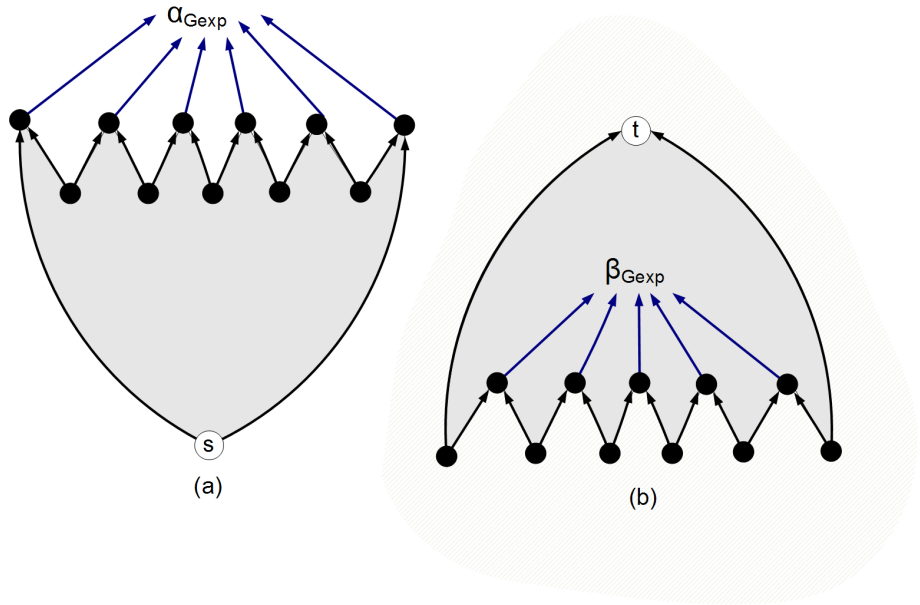


Abbildung 3.6: Schematische Darstellung eines inneren Faces $\alpha_{G_{exp}}$ (a) und eines äußeren Faces $\beta_{G_{exp}}$ (b) und die Zuweisung der Senken. (a) Alle Senken-Switches des äußeren Faces sind auch Senken des sT-Graphen. Sie werden dem äußeren Face durch eine aufwärtskonsistente Zuweisung $\mathcal{A}$ zugewiesen. (b) Mit Ausnahme von einem Senken-Switch t sind alle Senken-Switches Senke des sT-Graphen. Sie werden dem inneren Face zugewiesen, in das sie hineinstechen.

Für die weiteren Untersuchungen wird sich das folgende Hilfslemma als nützlich erweisen.

Lemma 3.1. *Seien $\Gamma_{G_{exp}}$, Γ_K, $\Gamma'_{G_{exp}}$ und $\{u, v\}$ wie oben beschrieben. Sei die Komponente K ein sT-Graph mit $s \notin K^\circ := K - \{u, v\}$. Wenn die Eigenschaft von u und v als Senken-Switch bzw. nicht Senken-Switch in F^l und/oder F^r nach der Spiegelung in F'^l und/oder F'^r beibehalten wird, dann ist $\Gamma'_{G_{exp}}$ eine aufwärtsplanare Einbettung.*

Beweis. Seien $c_{F^l}(u)$, $c_{F^r}(u)$, $c_{F'^l}(u)$ und $c_{F'^r}(u)$ die Kapazitäten von u zu den Faces F^l, F^r, F'^l und F'^r. Seien $c_{F^l}(v)$, $c_{F^r}(v)$, $c_{F'^l}(v)$ und $c_{F'^r}(v)$ die Kapazitäten von v zu den Faces F^l, F^r, F'^l und F'^r. Aus den Voraussetzungen folgen $c_{F^l}(u) = c_{F'^l}(u)$, $c_{F^r}(u) = c_{F'^r}(u)$, $c_{F^l}(v) = c_{F'^l}(v)$ und $c_{F^r}(v) = c_{F'^r}(v)$.

Nach der Spiegelung bleiben die Faces von Γ_H mit Ausnahme von F^l und F^r unverändert. Die Kantenaufzählung der Faces in Γ_K sind umgedreht, so dass zu jedem Face $f \in \Gamma_K$ ein eindeutig korrespondierendes Face $f' \in \Gamma'_K$ existiert.

Seien $|f|_F$ die Anzahl der Senken des Face-Abschnittes f, die an dem Face F zugewiesen wird und $c_F(f)$ der Beitrag von f zur Kapazität von F. Für den Beweis des Lemmas werden die folgende zwei Fälle unterschieden:

Fall 1: F^l und F^r sind innere Faces von $\Gamma_{G_{exp}}$
Die Abbildung 3.7 gibt eine Darstellung dieses Falles wieder. Die Face-Abschnitte f^l_K und f^r_K sind Kantensequenzen aus dem äußeren Face α_K. Nach Beobachtung 3.1 sind alle Senken-Switches in den beiden Face-Abschnitten auch Senken in G_{exp}. Die Zuweisung $\mathcal{A}$ weist die Senken von f^l_K dem Face F^l und die Senken von f^r_K dem Face F^r zu. Somit ist $|f^l_K|_{F^l} = c_{F^l}(f^l_K)$ und $|f^r_K|_{F^r} = c_{F^r}(f^r_K)$. Es lässt sich ferner beobachten, dass die Knoten u und v von $\mathcal{A}$ weder an F^l noch an F^r zugewiesen werden, da sie nicht in deren Gebieten hineinstechen können. Wir konstruieren nun aus $\mathcal{A}$ die Zuweisung $\mathcal{A}'$ wie folgt:

1. Für jedes Face $f \in \Gamma_H$ mit $f \notin \{F'^l, F'^r\}$ ist $\mathcal{A}'(f) := \mathcal{A}(f)$.
2. $\mathcal{A}'$ weist die Senken des Face-Abschnitts f^l_H und f^r_H wie $\mathcal{A}$ zu. Falls $\mathcal{A}$ sie an F^l oder F^r zuweist, dann weist $\mathcal{A}'$ sie an F'^l oder F'^r zu.
3. Sei $f' \in \Gamma'_K$ und $f \in \Gamma_K$ das zu f' korrespondierende Face. Ferner enthält f' und f keine Kanten aus f^l_K und f^r_K: $\mathcal{A}'(f') := \mathcal{A}(f)$
4. $\mathcal{A}'$ weist die Senken von f^l_K an F'^r und die Senken von f^r_K an F'^l zu.

Nach der Konstruktion gilt für alles Faces $f \in \Gamma_{G'_{exp}}$ mit $f \notin \{F'^l, F'^r\}$ die Kapazitätsgleichung. Das ist offensichtlich, da für f die Zuweisung von $\mathcal{A}$ benutzt wird und $\mathcal{A}$ nach Voraussetzung aufwärtskonsistent ist. Wir betrachten nun die Kapazitätsgleichungen von F^l, F'^l, F^r und F'^r. Vor der Spiegelung gilt:

$$\underbrace{|f^l_H|_{F^l} + |f^l_K|_{F^l}}_{=|\mathcal{A}(F^l)|} = \underbrace{c_{F^l}(f^l_H) + |f^l_K|_{F^l} + c_{F^l}(u) + c_{F^l}(v) - 1}_{=c(F^l)}$$

$$\underbrace{|f^r_H|_{F^r} + |f^r_K|_{F^r}}_{=|\mathcal{A}(F^r)|} = \underbrace{c_{F^r}(f^r_H) + |f^r_K|_{F^r} + c_{F^r}(u) + c_{F^r}(v) - 1}_{=c(F^r)}$$

Nach der Spiegelung folgt aus der Konstruktion von $\mathcal{A}'$:

$$\underbrace{\overbrace{|f^l_H|_{F^l}}^{=|f'^l_H|_{F'^l}} + \overbrace{|f^r_K|_{F^r}}^{=|f'^l_K|_{F'^l}}}_{=|\mathcal{A}'(F^l)|} = \underbrace{\overbrace{c_{F^l}(f^l_H)}^{c_{F'^l}(f'^l_H)} + |f^r_K|_{F^r} + \overbrace{c_{F^l}(u)}^{=c_{F'^l}(u)} + \overbrace{c_{F^l}(v)}^{=c_{F'^l}(v)} - 1}_{=c(F'^l)}$$

$$\underbrace{\overbrace{|f_H^r|_{F^r}}^{=|f_H'^r|_{F'^r}} + \overbrace{|f_K^l|_{F^l}}^{=|f_K'^r|_{F'^r}}}_{=|\mathcal{A}'(F'r)|} = \underbrace{\overbrace{c_{F^r}(f_H^r)}^{=c_{F'^r}(f_H'^r)} + |f_K^l|_{F^l} + \overbrace{c_{F^r}(u)}^{=c_{F'^r}(u)} + \overbrace{c_{F'^r}(v)}^{=c_{F'^r}(v)} - 1}_{=c(F'r)}$$

Somit gilt auch die Kapazitätsgleichung für F'^l und F'^r. $\mathcal{A}'$ ist eine aufwärtskonsistente Zuweisung. Da alle Einbettungen von G_{exp} Kandidaten einer aufwärtsplanaren Einbettung sind, ist nach Satz 2.7 die Einbettung $\Gamma'_{G_{exp}}$ aufwärtsplanar.

Fall 2: F^l oder F^r ist das äußere Face von $\Gamma_{G_{exp}}$
Ohne Beschränkung der Allgemeinheit kann angenommen werden, dass F^r der äußere Face ist. Der Beweis, dass F^l äußeres Face ist, geht bis auf die Umbenennung der Face-Abschnitte analog.

Es gilt wieder $|f_K^l|_{F^l} = c_{F^l}(f_K^l)$ und $|f_K^r|_{F^r} = c_{F^r}(f_K^r)$. Falls u oder v eine Senke ist, dann ist u oder v ein Senken-Switch des äußeren Face und $\mathcal{A}$ weist ihn F^r zu. Die Konstruktion von $\mathcal{A}'$ und die Kapazitätsgleichung für F^l und F'^l sind identisch mit Fall 1. Die Kapazitätsgleichung für F^r vor der Spiegelung ist

$$\underbrace{|f_H^r|_{F^r} + |f_K^r|_{F^r}}_{=|\mathcal{A}(F^r)|} = \underbrace{c_{F^r}(f_H^r) + |f_K^r|_{F^r} + c_{F^r}(u) + c_{F^r}(v) + 1}_{=c(F^r)}$$

und nach der Spiegelung ist

$$\underbrace{\overbrace{|f_H^r|_{F^r}}^{=|f_H'^r|_{F'^r}} + \overbrace{|f_K^l|_{F^l}}^{=|f_K'^r|_{F'^r}}}_{=|\mathcal{A}'(F'r)|} = \underbrace{\overbrace{c_{F^r}(f_H^r)}^{=c_{F'^r}(f_H'^r)} + |f_K^l|_{F^l} + \overbrace{c_{F^r}(u)}^{=c_{F'^r}(u)} + \overbrace{c_{F^r}(v)}^{=c_{F'^r}(v)} + 1}_{=c(F'r)}.$$

Somit ist $\mathcal{A}'$ eine aufwärtskonsiste Einbettung und nach Satz 2.7 ist die Einbettung $\Gamma'_{G_{exp}}$ aufwärtsplanar. □

Eine Illustration zum Beweis von Lemma 3.1 gibt die Abbildung 3.7. In (a) ist die Komponente K vor der Spiegelung dargestellt. Die Senken des Face-Abschnitts f_l^K werden dem Face F_l zugewiesen. Der Beitrag zur Kapazität von f_l^K zu F_l ist gleich der Anzahl der Zuweisungen von f_l^K an F_l. In (b) wird Γ_K gespiegelt. Die Face-Abschnitte f_l^K und f_r^K vertauschen ihre Positionen. Eine aufwärtskonsistente Zuweisung $\mathcal{A}'$ kann aus $\mathcal{A}$ konstruiert werden, indem die Zuweisung für Faces F_l und F_r in den Abschnitten f_l und f_r entsprechend geändert wird.

Eine Hilfsstruktur für die weitere Untersuchungen

Als nächstes betrachten wir die Hilfsstruktur $\mathcal{H}$ aus der Abbildung 3.8 (a). $\mathcal{H}$ besitzt zwei aufwärtsplanare Einbettungen, wobei die zweite Einbettung eine Spiegelung

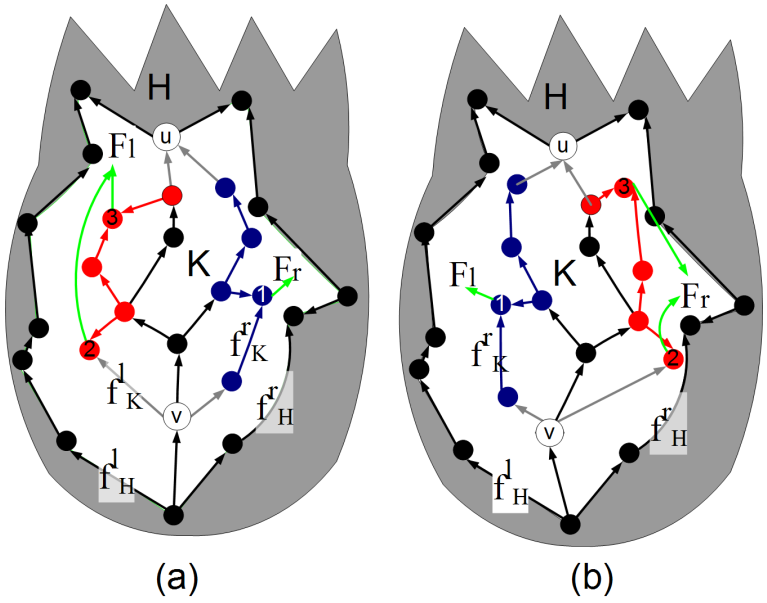


Abbildung 3.7: Zuordnung der Senken bei einer Spielgelung.

der ersten Einbettung ist. Sei A der Teilgraph von $\mathcal{H}$ ohne den Knoten s, die Kante $s \to v$ und $s \to v$. Für eine gegebene aufwärtsplanare Einbettung $\Gamma_{\mathcal{H}}$ von $\mathcal{H}$ kann die von $\Gamma_{\mathcal{H}}$ induzierte Einbettung Γ_A auf A offensichtlich nicht gespiegelt werden, so dass die daraus resultierende Einbettung $\Gamma'_{\mathcal{H}}$ aufwärtsplanar ist (siehe Abbildung 3.8 (b)).

Beobachtung 3.2. *Für die Struktur $\mathcal{H}$ gilt:*

(a) Ein Digraph G mit einem Untergraph homöomorph zu $\mathcal{H}$ ist nicht aufwärtsplanar, wenn $\mathcal{H}$ nicht aufwärtsplanar gezeichnet werden kann.

(b) Sei $\Gamma_{\mathcal{H}}$ und Γ_A wie oben beschrieben. Sei $\Gamma'_{\mathcal{H}}$ die Einbettung von $\mathcal{H}$, die aus der Spiegelung von Γ_A entsteht. Dann ist $\Gamma'_{\mathcal{H}}$ keine aufwärtsplanare Einbettung.

Im Folgenden wird die Komponente K nach den Ersetzungsregeln $\mathcal{R}_1$ bis $\mathcal{R}_4$ klassifiziert und untersucht.

K ist mit $\mathcal{R}_1$ assoziiert

Nach Eigenschaft $\mathcal{E}_{1a}$ gibt es eine Struktur P_K homöomorph zum Peak $u \to t \leftarrow v$ in K. In H existiert eine Struktur P_H homöomorph zu einem Valley $u \leftarrow s \to v$ (siehe Abbildung 3.3 (R1)). In einer aufwärtsplanaren Zeichnung der Einbettung $\Gamma_{G_{exp}}$ ist der Knoten t höher gezeichnet als u und v. Durch eine Spiegelung von Γ_K kann $\Gamma'_{G_{exp}}$ im Allgemeinen nicht aufwärtsplanar gezeichnet werden, da dann

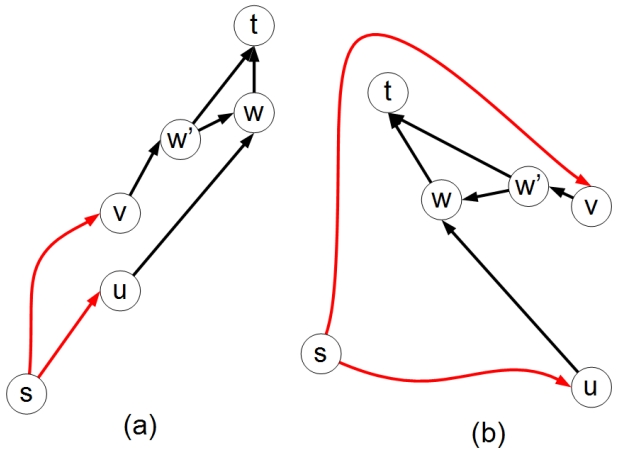


Abbildung 3.8: (a) die Hilfsstruktur $\mathcal{H}$ mit einer aufwärtsplanaren Einbettung Γ_H; (b) die nicht aufwärtsplanare Einbettung $\Gamma'_{\mathcal{H}}$; $\Gamma'_{\mathcal{H}}$ entsteht durch eine Spiegelung der von $\Gamma_{\mathcal{H}}$ induzierten Einbettung Γ_A auf A.

die Struktur P_H verhindert, dass t höher gezeichnet werden kann als u und v (siehe Abbildung 3.9). Das folgende Lemma gibt an, unter welchen Bedingungen die Komponente K gespiegelt werden kann, so dass die resultierende Einbettung eine aufwärtsplanare Zeichnung besitzt.

Lemma 3.2. *Sei $rule(K) = \mathcal{R}_1$. Dann ist $\Gamma'_{G_{exp}}$ genau dann aufwärtsplanar, wenn in K kein innerer Knoten existiert, der gleichzeitig von u und v dominiert wird.*

Beweis. „⇒“ (indirekt)
Angenommen es existiert ein innerer Knoten w in K, der von u und v gleichzeitig dominiert wird. Dann gibt es eine Struktur P_K in K homöomorph zum Teilgraphen A von $\mathcal{H}$ (siehe Abbildung 3.8). Nach $\mathcal{E}_{1a}$ gibt es in H eine Struktur homöomorph zu einem Valley $u \leftarrow s \rightarrow v$. P_K und P_H bilden zusammen einen Untergraphen homöomorph zu der Struktur $\mathcal{H}$. Aus der Beobachtung 3.2 folgt, dass durch die Spiegelung von Γ_K die Einbettung $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar sein kann.

„⇐“ Angenommen in K gibt es keinen Knoten, der gleichzeitig von u und v dominiert wird. Dann lässt sich K in zwei Teilgraphen K_u und K_v aufteilen. K_u enthält alle von u dominierten und K_v alle von v dominierten Knoten. Die Knoten $t_1, \ldots, t_n$, die gleichzeitig in K_u und K_v sind, sind die einzigen Knoten, die sowohl von u als auch von v dominiert werden. Aus der Annahme lässt sich ableiten, dass sie alle Senken in G_{exp} sind. Sie sind sozusagen die Nahtstellen zwischen K_u und K_v. Der Knoten u (bzw. v) ist die einzige Quelle von K_u (bzw. K_v). Somit sind die beiden Teilgraphen sT-Graphen.

Die Zuweisung $\mathcal{A}$ weist F^l keinen der Knoten $t_1, \ldots, t_n$ und F^r genau einen die-

ser Knoten zu. Sei ohne Einschränkung der Allgemeinheit t_1 der Knoten, der F^r zugewiesen wurde. Wir „spalten“ die Knoten $t_2, \ldots, t_n$ auf, indem wir zusätzliche zu t_i korrespondierende Dummy-Knoten t'_i, $2 \leq i \leq n$ einführen. t_i wird nur von u und t'_i wird nur von v dominiert. t_1 ist der einzige Knoten der gleichzeitig von u und v dominiert wird. Die Einbettung des so entstandenen Graphen, die zu $\Gamma_{G_{exp}}$ korrespondiert, nennen wir Γ_{G^*} und die Komponente, die zu K korrespondiert nenne wir K^*. Offensichtlich ist Γ_{G^*} aufwärtsplanar. Die Einbettungen $\Gamma_{K_u} \subset \Gamma_K$ und $\Gamma_{K_v} \subset \Gamma_K$ von K_u und K_v bleiben unverändert. Die Untersuchungen der Knoten u und t_1 sowie der Knoten v und t'_1 zeigen, dass Lemma 3.1 für die Spiegelung von Γ_{K_u} und Γ_{K_v} angewendet werden kann. Die Spiegelung der beiden Einbettungen bedeutet eine Spiegelung der Einbettung Γ_{K^*}, die durch Γ_{K_u} und Γ_{K_v} auf K^* induziert wurde. Nun wird der Spaltungsprozess rückgängig gemacht, indem die Knoten t_i und t'_i mit $1 \leq i \leq n$ zu einem gemeinsamen Knoten verschmolzen wird. Die Einbettung $\Gamma'_{G_{exp}}$, die daraus resultiert, ist aufwärtsplanar. □

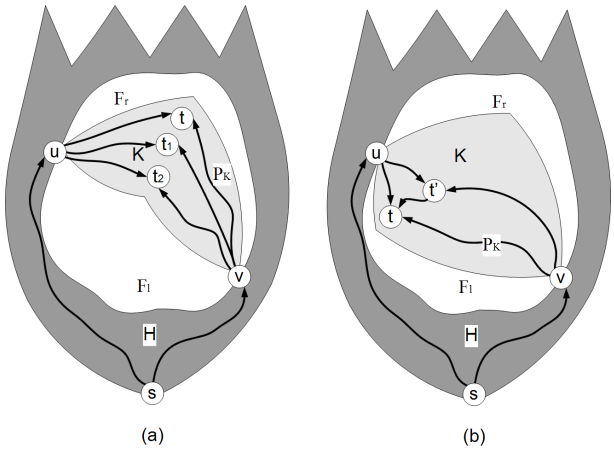


Abbildung 3.9: Spiegelung einer mit $\mathcal{R}_1$ assoziierten Komponente. Durch die Spiegelung verändert sich die Zuweisung des Knotens t des Peaks $u \rightarrow t \leftarrow v$. Der Peak sticht in das Face F_l. Das Valley $v \leftarrow s \rightarrow u$ verhindert eine aufwärtsplanare Zeichnung.

Um festzustellen, ob Γ_K gespiegelt werden kann, so dass die resultierende Einbettung $\Gamma'_{G_{exp}}$ aufwärtsplanar ist, wird die Komponente K von u aus traversiert und alle erreichten inneren Knoten markiert. Danach wird K von v aus traversiert. Wird dann ein markierter Knoten erreicht, so kann K nicht in unserem Sinne gespiegelt werden. Dieses Ergebnis wird im folgenden Korollar festgehalten.

Korollar 3.1. *Seien $rule(K) = \mathcal{R}_1$, n die Anzahl der Knoten und m die Anzahl der Kanten von K. Seien Γ_K und $\Gamma'_{G_{exp}}$ wie oben beschrieben. Dann kann in Zeit $\mathcal{O}(n+m)$ festgestellt werden, ob $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K eine aufwärtsplanare Einbettung ist.*

K ist mit $\mathcal{R}_{2a}$ assoziiert

Nach Eigenschaft $\mathcal{E}_{2b}$ ist u die einzige Quelle in K. Die Komponente K ist somit ein sT-Graph. Ferner ist nach Voraussetzung von $\mathcal{R}_{2a}$ der Knoten v eine Senke und $s \notin K^\circ$. Wenn u ein Senken-Switch bzw. nicht Senken-Switch in F^l und/oder F^r ist, dann ist u nach der Spiegelung Senken-Switch bzw. nicht Senken-Switch in F'^l und/oder F'^r. Der Grund hierfür ist, dass u eine Quelle von K ist. Für v gilt das gleiche, da er eine Senke in K ist. Somit können wir Lemma 3.1 anwenden.

Korollar 3.2. *Sei $rule(K) = \mathcal{R}_{2a}$. Dann ist die resultierende Einbettung $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K aufwärtsplanar.*

K ist mit $\mathcal{R}_{2b}$ assoziiert

v ist nach Eigenschaft $\mathcal{E}_{2j}$ die einzige Quelle von H. Somit ist H ein sT-Graph. Da u Quelle in K und v Quelle in H ist, können wir feststellen, dass durch eine Spiegelung die Eigenschaft der beiden Knoten als Senken-Switch bzw. nicht Senken-Switch in den Faces F^l und/oder F^r nach der Spiegelung in F'^l und/oder F'^r sich nicht verändert hat. Es gilt zusätzlich $s \in K^\circ$ ist $s \notin H$. Somit kann Lemma 3.1 angewendet und Γ_H gespiegelt werden. Sei $\Gamma''_{G_{exp}}$ die resultierende aufwärtsplanare Einbettung aus der Spiegelung von Γ_H. $\Gamma''_{G_{exp}}$ wird nun gespiegelt. Die resultierende Einbettung $\Gamma'_{G_{exp}}$ ist wieder aufwärtsplanar. Die Doppelspiegelung entspricht einer Spiegelung von Γ_K in $\Gamma_{G_{exp}}$.

Korollar 3.3. *Sei $rule(K) = \mathcal{R}_{2b}$. Dann ist die resultierende Einbettung $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K aufwärtsplanar.*

K ist mit $\mathcal{R}_{3a}$ assoziiert

u ist nach Eigenschaft $\mathcal{E}_{3b}$ die einzige Quelle in K. K ist also ein sT-Graph. Wenn u ein Senken-Switch bzw. nicht Senken-Switch in F^l und/oder F^r ist, dann ist u nach der Spiegelung von Γ_K Senken-Switch bzw. nicht Senken-Switch in F'^l und/oder F'^r, da er die Quelle von K ist. Für v gilt das gleiche, da er die Quelle in H ist. Ferner gilt $s \notin K^\circ$. Somit kann Lemma 3.1 angewendet werden.

Korollar 3.4. *Sei $rule(K) = \mathcal{R}_{3a}$. Dann ist die resultierende Einbettung $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K aufwärtsplanar.*

K ist mit $\mathcal{R}_{3b}$ assoziiert

H ist nach Eigenschaft $\mathcal{E}_{3h}$ ein sT-Graph. v ist die Quelle von H und u ist eine Quelle von K. Wenn u ein Senken-Switch bzw. nicht Senken-Switch in F^l oder F^r ist, dann ist u nach der Spiegelung von Γ_K Senken-Switch bzw. nicht Senken-Switch in F'^l oder F'^r, da er die Quelle von K ist. Für v gilt das gleiche, da v die Quelle

in H ist. Ferner ist $s \notin H$, da nach Voraussetzung der Regel $\mathcal{R}_{3b}$ $s \in K^\circ$ gilt. Somit kann Lemma 3.1 angewendet und Γ_H gespiegelt werden. Sei $\Gamma''_{G_{exp}}$ die resultierende aufwärtsplanare Einbettung aus der Spiegelung von Γ_H. $\Gamma''_{G_{exp}}$ wird nun gespiegelt. Die resultierende Einbettung $\Gamma'_{G_{exp}}$ ist wieder aufwärtsplanar. Die Doppelspiegelung entspricht einer Spiegelung von Γ_K in $\Gamma_{G_{exp}}$.

Korollar 3.5. *Sei $rule(K) = \mathcal{R}_{3b}$. Dann ist die resultierende Einbettung $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K aufwärtsplanar.*

K ist mit $\mathcal{R}_{3c}$ assoziiert

Nach $\mathcal{E}_{3i}$ existiert eine Struktur P_K homöomorph zum Peak $u \rightarrow t \leftarrow v$, $t \in K$. Ferner existiert in H eine Struktur P_H homöomorph zu einem Valley $u \leftarrow s \rightarrow v$. Im Folgenden wird die Face-Notation aus der Abbildung 3.10 benutzt. In Γ_{Gexp} weist $\mathcal{A}$ den Knoten t dem Face F^r zu. Im Allgemeinen verhindert die Struktur P_H nach der Spiegelung von Γ_K, dass der Knoten t aus der Struktur P_K höher gezeichnet werden kann als u und v. Es gibt aber Ausnahmen wie das folgende Lemma zeigt.

Lemma 3.3. *Sei $rule(K) = \mathcal{R}_{3c}$. Dann ist $\Gamma'_{G_{exp}}$ genau dann aufwärtsplanar, wenn die folgenden Bedingungen erfüllt sind:*

(1) Es gibt in K keinen inneren Knoten, der gleichzeitig von u und v dominiert wird.

(2) K kann durch das Split-Paar $\{u, v\}$ in zwei Komponenten K' und K^* zerlegt werden, so dass kein innerer Knoten in K^* von v dominiert wird.

Beweis. „⇒“ (indirekt)
Angenommen die Bedingung (1) gilt nicht. Es gibt einen innerer Knoten w in K, der gleichzeitig von u und v dominiert wird. Dann gibt es eine Struktur P_K in K homöomorph zum Teilgraph A der Hilfsstruktur $\mathcal{H}$. Es muss einen Pfad $s \rightsquigarrow u$ in H geben, sonst dominiert H nicht u. Zusammen mit dem Pfad $s \rightsquigarrow v$ (siehe Eingenschaft $\mathcal{E}_{3m}$) bilden sie eine Struktur P_H homöomorph zu einem Valley. P_H und P_K bilden dann zusammen einen Untergraphen homöomorph zu der Hilfsstruktur $\mathcal{H}$ (siehe Abbildung 3.8). Nach Beobachtung 3.2 ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar.
Angenommen die Bedingung (2) gilt nicht. Dann gibt es zwei mögliche Fälle.

- K kann nicht in K^* und K' durch das Split-Paar u und v zerlegt werden, weil ein Knoten w existiert, der sowohl ein Knoten eines Pfades $u \rightsquigarrow v$ als auch ein Knoten in einer Struktur homöomorph zu einem Peak $u \rightarrow t \leftarrow v$ ist. w verhindert sozusagen das Zerlegen von K in K^* und K'. Er bildet zusammen mit u, v und t eine Struktur $S := u \rightsquigarrow w \rightsquigarrow u \cup v \rightsquigarrow t \cup w \rightsquigarrow t$. Die von der Einbettung Γ_K auf S induzierte Einbettung kann nicht gespiegelt werden, da

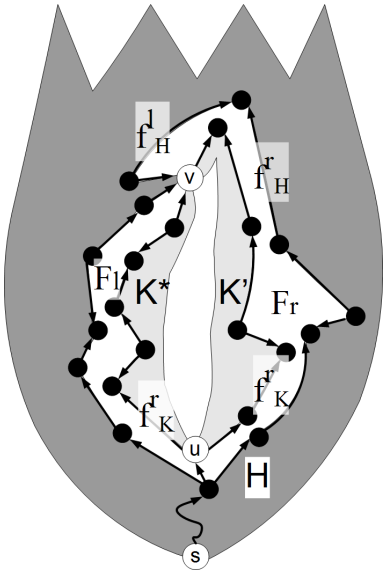


Abbildung 3.10: Illustration zum Beweis 3.3. Die Komponente K wird in zwei Komponenten K' und K^* aufgeteilt.

das Valley P_H in H zusammen mit S verhindert, dass der Knoten t nach der Spiegelung höher gezeichnet werden kann als u und v. Somit ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar.

- Die Komponente K kann durch das Split-Paar u und v in die Komponenten K^* und K' zerlegt werden. In K^* existiert jedoch ein innerer Knoten w' der gleichzeitig von u und v dominiert wird. Dann existiert eine Struktur in K^* homöomorph zum Teilgraphen A der Hilfsstruktur $\mathcal{H}$. Zusammen mit P_H bildet dann A eine Struktur homöomorph zur Hilfsstruktur $\mathcal{H}$. Somit ist nach Beobachtung 3.2 $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar.

„⇐“ Alle Knoten in K^* werden von u dominiert. Ferner ist v eine Senke in K^* (siehe Abbildung 3.10) und es gilt $s \notin K^* - \{u, v\}$. Somit ist K^* mit der Regel $\mathcal{R}_{2a}$ assoziiert. Bei der Spiegelung von K^* wird kann Korollar 3.2 angewendet werden.

u und v sind Quellen in K' und es ist $s \notin K' - \{u, v\}$. Nach Voraussetzung enthält K' keinen inneren Knoten, der gleichzeitig von u und v dominiert wird. Somit ist K' mit der Regel $\mathcal{R}_1$ assoziiert. Bei der Spiegelung von K' kann Lemma 3.2 nun angewendet werden.

Sei $\Gamma'_{G_{exp}}$ die Einbettung, die daraus entsteht, indem zuerst Γ_{K^*} und dann $\Gamma_{K'}$ gespiegelt wird. $\Gamma'_{G_{exp}}$ ist aufwärtsplanar, wie wir oben gezeigt haben. Was noch gezeigt werden muss ist, dass die Spiegelung der Einbettungen Γ_{K^*} und $\Gamma_{K'}$ das gleiche bewirkt wie die Spiegelung von Γ_K in $\Gamma_{G_{exp}}$.

Durch die Spiegelung von K^* und K' werden die zirkulären Kantenaufzählungen der Knoten in K, mit Ausnahme der Knoten u und v, umkehrt. Sei $e_1, \ldots, e_i, e_{i+1}, \ldots, e_l$ die Kantenordnung von u vor der Spiegelung von K' und K^*, wobei $e_1, \ldots, e_i$ Kanten aus K' und $e_{i+1}, \ldots, e_l$ Kanten aus K^* sind. Nach der Spiegelung von K' und K^* ist $e_i, \ldots, e_1, e_l, \ldots, e_{i+1}$ die neue Kantenordnung von u. Das ist genau die Umkehrung der Kantenordnung vor der Spiegelung. Analog gilt dies für den Knoten v. □

Mit dem folgenden Verfahren kann getestet werden, ob K gespiegelt werden kann. Zuerst wird K in Abhängigkeit des Split-Paares $\{u, v\}$ in die Split-Komponenten $K_1, \ldots, K_n$ zerlegt. Die einzelnen Komponenten werden erst von v aus traversiert und alle innere Knoten markiert. Dann werden sie von u aus traversiert. Wenn dabei ein markierter Knoten erreicht wird, dann gibt es einen Knoten, der sowohl von u als auch von v dominiert wird, und $\Gamma'_{G_{exp}}$ ist nicht aufwärtsplanar. Ansonsten werden die Komponenten aus $K_1, \ldots, K_n$, die keine von v dominierten Knoten enthalten, zu der Komponente K^* und die übrigen zu K' hinzugefügt. Dabei darf weder K^* noch K' leer sein.

Korollar 3.6. *Sei $rule(K) = \mathcal{R}_{3c}$, n die Anzahl der Knoten und m die Anzahl der Kanten von K. Seien Γ_K und $\Gamma'_{G_{exp}}$ wie oben beschrieben. Dann kann in Zeit $\mathcal{O}(n+m)$ festgestellt werden, ob $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K eine aufwärtsplanare Einbettung ist.*

K ist mit $\mathcal{R}_4$ assoziiert

Nach $\mathcal{E}_{4a}$ ist u oder v Quelle in H. Die Rollen von u und v sind vertauschbar. Ohne Beschränkung der Allgemeinheit sei u eine Quelle in H. Folgende drei Fälle werden unterschieden:

v ist Senke in H: Die Komponente H ist ein sT-Graph. u ist die einzige Quelle und v eine Senke in H. Die Quelle s ist nicht in H. Folglich ist H mit Regel $\mathcal{R}_{2a}$ assoziiert. Bei der Spiegelung von Γ_H kann Korollar 3.2 angewendet werden. Sei $\Gamma''_{G_{exp}}$ die resultierende aufwärtsplanare Einbettung nach der Spiegelung von Γ_H. $\Gamma''_{G_{exp}}$ wird nun gespiegelt. Das Ergebnis dieser Spiegelung ist die aufwärtsplanare Einbettung $\Gamma'_{G_{exp}}$.

Korollar 3.7. *Sei $rule(K) = \mathcal{R}_4$. Ferner sei u die Quelle und v eine Senke von H. Dann ist die resultierende Einbettung $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K aufwärtsplanar.*

v ist innerer Knoten in H: Wegen der Ingrad/Outgrad Restriktion eines Expansionsgraphen ist v kein innerer Knoten in K. Folglich ist er Senke in K. In H existiert eine Struktur P_H homöomorph zu dem Peak $u \rightarrow t \leftarrow v$. Es muss den Pfad $s \rightsquigarrow v$ in K geben, da v eine Senke in K ist. Dies impliziert, dass

es eine Struktur P_K homöomorph zu dem Valley $u \leftarrow s \rightarrow v$ geben muss. Im Allgemeinen verhindert die Struktur P_K bei der Spiegelung von Γ_K, dass der Knoten t aus P_H höher gezeichnet werden kann als u und v. Es gibt jedoch wieder Ausnahmen.

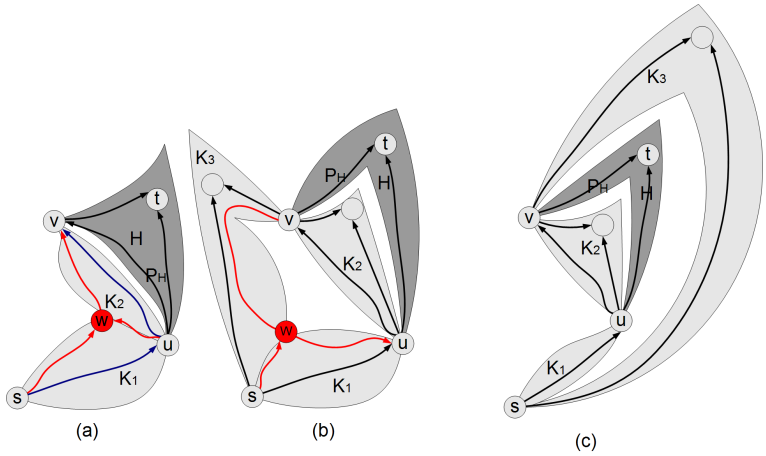


Abbildung 3.11: (a) Illustration zum Beweis von Lemma 3.4: (b) Illustration zum Beweis von Lemma 3.5. Der Knoten w bildet mit den roten Kanten eine Struktur S. Bei der Spiegelung von Γ_K wird S mitgespiegelt und er verhindert dann, dass der Knoten t höher gezeichnet werden kann als u und v. (c) Spiegelung des Teilgraphen K_3, falls die Struktur S nicht existiert.

Lemma 3.4. *Sei rule$(K) = \mathcal{R}_4$, u eine Quelle und v ein innerer Knoten in H. $\Gamma'_{G_{exp}}$ ist genau dann aufwärtsplanar, wenn die folgenden Bedingungen erfüllt sind:*

(1) K enthält einen Teilgraphen K_1, so dass s die Quelle und u ein Knoten in K_1 ist.

(2) Falls ein Pfad $u \rightsquigarrow v$ in K existiert, dann enthält K den Teilgraphen K_2, so dass u die Quelle und v eine Senke von K_2 ist.

(3) Falls ein Pfad $s \rightsquigarrow v$ in K existiert, der u nicht enthält, dann enthält K einen Teilgraphen K_3, so dass s die Quelle und v eine Senke in K_3 ist.

(4) Es gilt $K = K_1 \cup K_2 \cup K_3$, $K_1 \cap K_2 = \{u\}$, falls K_2 existiert, $K_1 \cap K_3 = \{s\}$, falls K_3 existiert und $K_2 \cap K_3 = \{v\}$, falls K_2 und K_3 existieren.

Beweis. „⇒“ (indirekt)
zu (1):
Nach Voraussetzung muss es einen Pfad von s nach u geben. Der Teilgraph,

der diesen Pfad enthält, ist der Teilgraph K_1. Trivialerweise ist s Quelle und u ein Knoten in K_1. Folglich ist (1) immer erfüllt.
zu (2):
Wenn ein Pfad $u \rightsquigarrow v$ in K existiert, dann existiert auch der Teilgraph K_2, der diesen Pfad enthält. Der Knoten v ist nach Voraussetzung Senke in K, also auch Senke in K_2. Angenommen (2) ist nicht erfüllt. Dann ist u keine Quelle in K_2. Folglich ist u ein innerer Knoten in K_2 und es existiert ein Pfad $s \rightsquigarrow u \rightsquigarrow v$ in K_2. Somit ist $K_1 = K_2$. Es gibt dann einen Knoten w der verhindert, dass K in die Teilgraphen K_1 und K_2 zerlegt werden kann. Folglich existiert ein Pfad von $s \rightsquigarrow w$, der den Knoten u nicht enthält und ein Pfad $w \rightsquigarrow v$. Zusätzlich existiert entweder ein Pfad $u \rightsquigarrow w$ oder ein Pfad $w \rightsquigarrow u$ in K_2. Diese drei Pfade bilden eine Struktur S (siehe Abbildung 3.11 (a)). Bei einer Spiegelung von Γ_K verhindert S, dass der Knoten t aus P_H höher gezeichnet werden kann als u und v. Somit ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar.
zu (3):
Wenn ein Pfad $s \rightsquigarrow v$ existiert, dann existiert auch die Komponente K_3. Diese enthält den Pfad $s \rightsquigarrow v$ und die Quelle s. Trivialerweise ist s Quelle in K_3. Nach Voraussetzung ist v Senke in K, also auch Senke in K_3. Folglich ist (3) immer erfüllt.
zu (4):
Angenommen (4) ist nicht erfüllt. Dann gibt es drei mögliche Fälle:

(i) Es gibt die Teilgraphen K_1 und K_2 mit $K = K_1 \cup K_2$ und es gibt einen Knoten w, so dass $K_1 \cap K_2 = \{w, u\}$. Dieser Fall wurde schon unter (2) behandelt.

(ii) Es gibt die Teilgraphen K_1 und K_3 mit $K = K_1 \cup K_3$ und es gibt einen Knoten w, so dass $K_1 \cap K_3 = \{w, u\}$. Es existiert wieder die Struktur S aus (2), die bei der Spiegelung verhindert, dass der Knoten t aus der Struktur P_H höher gezeichnet werden kann als u und v. Somit ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar.

(iii) Es gibt die Teilgraphen K_1, K_2 und K_3 mit $K = K_1 \cup K_2 \cup K_3$ und es gibt einen Knoten w, so dass $K_1 \cap K_2 = \{w, u\}$, $K_1 \cap K_3 = \{w, u\}$ oder $K_1 \cap K_2 \cap K_3 = \{w, u\}$. Falls $K_1 \cap K_2 = \{w, u\}$ ist, dann gilt (i) und falls $K_1 \cap K_3 = \{w, u\}$ ist, dann gilt (ii). Den letzten Fall können wir entweder auf (i) oder auf (ii) zurückführen. Somit ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar.

„⇐“ Seien (1) und (4) erfüllt, sowie (2) und/oder (3). Wir betrachten die Teilgraphen K_1, K_2 und K_3.

In K_1 ist der Knoten u keine Quelle, da sonst kein Pfad von s nach u führt. u ist sowohl Quelle in K_2, falls K_2 existiert, als auch Quelle in H. s ist die einzige Quelle in K_1. Trivialerweise ist $s \notin K_1^\circ := K_1 - \{s, u\}$. Folglich gilt

$rule(K_1) = \mathcal{R}_{2a}$, wenn v eine Senke in K_1 ist oder $rule(K_1) = \mathcal{R}_{3a}$, wenn v ein innerer Knoten in K_1 ist. Nach Korollar 3.2 oder 3.7 ist die resultierende Einbettung nach der Spiegelung der Einbettung $\Gamma_{K_1} \subset \Gamma_K$ aufwärtsplanar.

Falls der Teilgraph K_2 existiert, ist u die einzige Quelle und v eine Senke in K_2. Ferner ist $s \notin K_2^\circ := K_2 - \{u, v\}$. Somit gilt $rule(K_2) = \mathcal{R}_{2a}$. Nach Korollar 3.2 ist die resultierende Einbettung nach der Spiegelung der Einbettung $\Gamma_{K_2} \subset \Gamma_K$ aufwärtsplanar.

Falls der Teilgraph K_3 existiert, ist s eine Quelle und v eine Senke in K_3. Trivialerweise ist $s \notin K_3^\circ := K_3 - \{s, v\}$. Somit gilt $rule(K_3) = \mathcal{R}_{2a}$. Nach Korollar 3.2 ist die resultierende Einbettung nach der Spiegelung der Einbettung $\Gamma_{K_2} \subset \Gamma_K$ aufwärtsplanar.

Eine Untersuchung der Kantenordnung der Knoten s, u und v zeigt (wie in Lemma 3.3), dass die Spiegelung der drei Komponenten gleichbedeutend ist mit der Spiegelung von Γ_K und $\Gamma_{G_{exp}}$. □

Um festzustellen, ob $\Gamma'_{G_{exp}}$ nach der Spiegelung von K aufwärtsplanar ist, kann man wie folgt vorgehen: Zuerst wird K von u aus traversiert und die erreichten Knoten markiert. Die ausgehenden Kanten von u werden entfernt und K wird von s aus traversiert. Falls ein markierter Knoten erreicht wird, ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar, da dieser Knoten ungleich u ist und sowohl zu K_1 als auch zu K_2 gehört. Anschließend werden alle Markierungen gelöscht. s wird von K entfernt und der zugrundeliegende Graph von $K - \{s\}$ wird von u aus traversiert. Falls dabei der Knoten v erreicht wird, ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar, da ein Knoten ungleich s existiert der sowohl zu K_1 als auch zu K_3 gehört. Die entfernten Kanten und der Knoten s werden wieder in K eingefügt.

Korollar 3.8. *Sei $rule(K) = \mathcal{R}_4$, u eine Quelle und v ein innerer Knoten in H. Sei ferner n die Anzahl der Knoten und m die Anzahl der Kanten von K. Dann kann in Zeit $\mathcal{O}(n+m)$ festgestellt werden, ob $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K eine aufwärtsplanare Einbettung ist.*

v ist Quelle in H: In H existiert eine Struktur P_H homöomorph zu dem Peak $u \to t \leftarrow v$. In K gibt es eine Struktur P_K homöomorph zum Valley $u \leftarrow s \to v$. Wie in dem Unterfall, wenn v innerer Knoten in H ist, verhindert P_K nach der Spiegelung von Γ_K, dass eine aufwärtsplanare Zeichnung existiert. Es gibt wieder Ausnahmen.

Lemma 3.5. *Sei $rule(K) = \mathcal{R}_4$ und u und v sind Quellen in K. $\Gamma'_{G_{exp}}$ ist genau dann aufwärtsplanar, wenn die folgenden Bedingungen erfüllt sind:*

(1) K enthält einen Teilgraphen K_1, so dass s die Quelle und u ein Knoten von K_1 ist.

(2) Falls ein Pfad $u \rightsquigarrow v$ in K existiert, dann enthält K einen Teilgraphen K_2, so dass u die Quelle und v ein Knoten von K_2 ist.

(3) Falls ein ungerichteter Pfad $s \rightsquigarrow v$ existiert, der u nicht enthält, dann gibt es einen Teilgraphen K_3, so dass s die Quelle und v ein Knoten in K_3 ist.

(4) Es gilt $K = K_1 \cup K_2 \cup K_3$, $K_1 \cap K_2 = \{u\}$, falls K_2 existiert, $K_1 \cap K_3 = \{s\}$, falls K_3 existiert und $K_2 \cap K_3 = \{v\}$, falls K_2 und K_3 existieren.

Beweis. „⇒“ (indirekt)
zu (1):
Nach Voraussetzung muss es einen Pfad von s nach u geben. Der Teilgraph, der diesen Pfad enthält, ist der Teilgraph K_1. Trivialerweise ist s Quelle und u ein Knoten in K_1. Folglich ist (1) immer erfüllt.
zu (2):
Falls ein Pfad von u nach v in K existiert, dann gibt es auch einen Teilgraphen K_2, der diesen Pfad enthält. Der Knoten v ist dann in K_2 enthalten. Angenommen (2) gilt nicht, dann ist u keine Quelle von K_2. Folglich ist u ein innerer Knoten in K_2 und es gibt einen Pfad $s \rightsquigarrow u \rightsquigarrow v$. Also ist $K_1 = K_2$. Es gibt dann einen Knoten w mit $s \rightsquigarrow w$, der den Knoten u nicht enthält und einen Pfad $w \rightsquigarrow v$. Zusätzlich existiert entweder ein Pfad $u \rightsquigarrow w$ oder ein Pfad $w \rightsquigarrow u$ in K_2. Diese drei Pfade bilden eine Struktur S. Bei einer Spiegelung von Γ_K verhindert S, dass der Knoten t aus P_H höher gezeichnet werden kann als u und v. Somit ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar.
zu (3):
Falls es einen Pfad von s nach v gibt, dann gibt es auch einen Teilgraphen K_3, der diesen Pfad enthält. Trivialerweise ist s Quelle und v ein Knoten in K_3. Folglich ist (3) immer erfüllt, wenn ein Pfad von s nach v existiert.
zu (4):
Angenommen (4) ist nicht erfüllt. In diesem Fall unterscheiden wir drei Fälle:

(i) Es gibt die Teilgraphen K_1 und K_2 mit $K = K_1 \cup K_2$ und es gibt einen Knoten w, so dass $K_1 \cap K_2 = \{w, u\}$. Dieser Fall wurde schon unter (2) behandelt.

(ii) Es gibt die Teilgraphen K_1 und K_3 mit $K = K_1 \cup K_3$ und es gibt einen Knoten w, so dass $K_1 \cap K_3 = \{w, u\}$ ist. Es existiert wieder die Struktur S aus (2), die bei der Spiegelung verhindert, dass der Knoten t aus der Struktur P_H höher gezeichnet werden kann als u und v. Somit ist $\Gamma'_{G_{exp}}$ nicht aufwärtsplanar.

(iii) Es gibt die Teilgraphen K_1, K_2 und K_3 mit $K = K_1 \cup K_2 \cup K_3$ und es gibt einen Knoten w, so dass $K_1 \cap K_2 = \{w, u\}$, $K_1 \cap K_3 = \{w, u\}$ oder $K_1 \cup K_2 \cup K_3 = \{w, u\}$. Falls $K_1 \cap K_2 = \{w, u\}$ ist, dann gilt (i) und falls

$K_1 \cap K_3 = \{w, u\}$ ist, dann gilt (ii). Den letzten Fall können wir entweder auf (i) oder auf (ii) zurückführen.

„⇐“ Wir betrachten die Teilgraphen K_1, K_2 und K_3.

In K_1 ist der Knoten u keine Quelle, da sonst kein Pfad von s nach u führt. u ist sowohl Quelle in K_2 — falls K_2 existiert — als auch in H. s ist die einzige Quelle in K_1. Trivialerweise ist $s \notin K_1^\circ := K_1 - \{s, u\}$. Folglich gilt $rule(K_1) = \mathcal{R}_{2a}$, wenn v eine Senke in K_1 ist oder $rule(K_1) = \mathcal{R}_{3a}$ wenn v ein innerer Knoten in K_1 ist. Nach Korollar 3.2 oder 3.7 ist die resultierende Einbettung nach der Spiegelung der Einbettung $\Gamma_{K_1} \subset \Gamma_K$ aufwärtsplanar.

Falls die Komponente K_2 existiert, dann existiert ein Pfad $u \rightsquigarrow v$ in K_2 und v ist keine Quelle in K_2. u ist die einzige Quelle in K_2 und $s \notin K_2^\circ := K_2 - \{u, v\}$. Falls v eine Senke in K_2 ist, dann ist $rule(K_2) = \mathcal{R}_{2a}$. Falls v ein innerer Knoten in K_2 ist, dann ist $rule(K_2) = \mathcal{R}_{3a}$. Die resultierende Einbettung nach der Spiegelung der Einbettung $\Gamma_{K_2} \subset \Gamma_K$ ist nach Korollar 3.2 oder 3.7 aufwärtsplanar.

Falls K_3 existiert, dann gilt trivialerweise $s \notin K_3^\circ := K_3 - \{s, v\}$. Wir unterscheiden hier drei Fälle:

- v ist eine Quelle in K_3:
  Dann ist $rule(K_3) = \mathcal{R}_1$. Da in K_3 kein Pfad von s nach v existiert, ist die resultierende Einbettung aus der Spiegelung von $\Gamma_{K_3} \subset \Gamma_K$ aufwärtsplanar. Eine aufwärtsplanare Zeichnung kann schematisch wie in Abbildung 3.11 (c) gezeichnet werden.
- v ist eine Senke in K_3:
  Dann ist $rule(K_3) = \mathcal{R}_{2a}$ und die resultierende Einbettung aus der Spiegelung von $\Gamma_{K_3} \subset \Gamma_K$ ist nach Korollar 3.2 aufwärtsplanar.
- v ist ein innerer Knoten in K_3:
  Wegen der Ingrad/Outgrad Restriktion darf es keinen Weg von u nach v geben. Somit existiert der Teilgraph K_2 nicht. v ist eine Quelle in $G_{exp} - K_3$. Folglich ist $rule(K_3) = \mathcal{R}_{3a}$. Die resultierende Einbettung aus der Spiegelung von $\Gamma_{K_3} \subset \Gamma_K$ ist nach Korollar 3.7 aufwärtsplanar.

Eine Untersuchung der Kantenordnung der Knoten s, u und v zeigt (wie in dem Beweis von Lemma 3.3), dass die Spiegelung der drei Komponenten gleichbedeutend ist mit der Spiegelung von K in $\Gamma_{G_{exp}}$. □

Analog kann es wie in dem Fall, dass v innerer Knoten in H ist, vorgegangen werden, um festzustellen, ob $\Gamma'_{G_{exp}}$ aufwärtsplanar ist.

Korollar 3.9. *Seien $rule(K) = \mathcal{R}_4$, u und v Quellen in H, n die Anzahl der Knoten und m die Anzahl der Kanten von K. Dann kann in Zeit $\mathcal{O}(n + m)$*

festgestellt werden, ob $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K eine aufwärtsplanare Einbettung ist.

Die Komponenten, für die die Ausnahmen in den Fällen $\mathcal{R}_1$, $\mathcal{R}_{3c}$ und $\mathcal{R}_4$ gelten, benötigen keine Spiegelung, wenn davon ausgangen wird, dass die Split-Komponenten der Split-Paare von K schon entsprechend gespiegelt wurden. Eine genaue Betrachtung der Ausnahmen zeigt nämlich, dass K keinen Pfad homöomorph zum Peak $u \rightarrow \circ \leftarrow v$ beinhaltet.

Die Ergebnisse dieses Abschnittes werden nun in einem Satz zusammengefasst.

Satz 3.1. *Seien G_{exp} der Expansionsgraph eines sT-Graphen G, $\{u, v\}$ ein Split-Paar von G_{exp}, K eine Split-Komponente des Split-Paares mit m Kanten und n Knoten und $H := G_{exp} - K$. Ferner sei $\Gamma_{G_{exp}}$ eine aufwärtsplanare Einbettung von G_{exp}, $\Gamma_K \subset \Gamma_{G_{exp}}$ und $\Gamma'_{G_{exp}}$ die resultierende Einbettung aus der Spiegelung von Γ_K. Dann gilt:*

(a) Wenn $rule(K) = \mathcal{R}_{2a}$, $rule(K) = \mathcal{R}_{2b}$, $rule(K) = \mathcal{R}_{3a}$ oder $rule(K) = \mathcal{R}_{3b}$ ist, dann ist $\Gamma'_{G_{exp}}$ aufwärtsplanar.

(b) Wenn $rule(K) = \mathcal{R}_4$, u eine Quelle und v eine Senke in H ist, dann ist $\Gamma'_{G_{exp}}$ aufwärtsplanar.

(c) Wenn $rule(K) = \mathcal{R}_1$ oder $rule(K) = \mathcal{R}_{3c}$ ist, dann kann in Zeit von $\mathcal{O}(n+m)$ festgestellt werden, ob $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K aufwärtsplanar ist.

(d) Wenn $rule(K) = \mathcal{R}_4$, u eine Quelle und v ein innerer Knoten oder eine Quelle in H ist, dann kann in Zeit von $\mathcal{O}(n+m)$ festgestellt werden, ob $\Gamma'_{G_{exp}}$ nach der Spiegelung von Γ_K aufwärtsplanar ist.

4 Aufzählung der aufwärtsplanaren Einbettungen

Ein SPQR-Baum eines 2-zusammenhängenden Graphen G enthält implizit alle planare Einbettungen von G. Nach den Überlegungen in Kapitel 3.2 wäre eine ähnlich Datenstruktur für die Aufzählung aller aufwärtsplanaren Einbettungen für einen Digraphen, dessen zugrundeliegender ungerichteter Graph 2-zusammenhängend ist, hilfreich. In diesem Kapitel werden wir, basierend auf der Arbeit von Bertolazzi et al. [7], so eine Datenstruktur beschreiben.

Sei nun in diesem Kapitel $G_{exp} = (V, A)$ der Expansionsgraph eines sT-Graphen G, dessen zugrundeliegender ungerichteter Graph 2-zusammenhängend ist. Sei ferner $\mathcal{T}$ der SPQR-Baum von G_{exp}.

4.1 Der P-Knoten

Die SPQR-Bäume, die wir in den nachfolgenden Kapiteln betrachten, sind alle SPQR-Bäume eines Expansionsgraphen. Die Expansionsgraphen haben eine besondere Eigenschaft, nämlich die Ingrad/Outgrad Restriktion. Wir werden in diesem Abschnitt untersuchen, inwiefern diese Restriktion Einfluss auf die Struktur eines P-Knotens hat. Der P-Knoten ist deshalb von Interesse, weil sein Skeleton $(k-1)!$ Einbettungen besitzt, wobei k die Anzahl der virtuellen Kanten des Skeletons ist. Mit der Untersuchung wollen wir feststellen, welche der $(k-1)!$ Einbettungen zu einer aufwärtsplanaren Einbettung von G führt.

Sei μ ein P-Knoten von $\mathcal{T}$ und $\{e_1, \ldots, e_k\}$ mit $k \geq 3$ die Menge der gerichteten virtuellen Kanten des sT-Skeletons S von μ. Zu jeder Kante e_i ist eine Split-Komponente K_i des Split-Paares $\{u, v\}$ von G_{exp} assoziiert. Wie die Komponenten können die gerichtete virtuelle Kanten mit Hilfe der Ersetzungsregeln $\mathcal{R}_1$ bis $\mathcal{R}_4$ klassifiziert werden. Mit Hilfe der Ingrad/Outgrad Restriktion eines Expansionsgraphen G_{exp}, nämlich entweder $in(v) \geq 1$ oder $out(v) \geq 1$ für $v \in V$, wird nun gezeigt, dass die virtuellen Kanten von μ nicht mit beliebigen Ersetzungsregel $\mathcal{R}_1$ bis $\mathcal{R}_4$ assoziiert sein kann. Dabei wird sukzessiv die nicht möglichen Skeletons eines P-Knotens ausgeschlossen.

Lemma 4.1. *Sei μ und $\{e_1, \ldots, e_k\}$ mit $k \geq 3$ wie oben beschrieben. Dann gilt für die gerichteten Kanten des sT-Skeletons S von μ genau eine der folgenden Kombinationen:*

I. $rule(e_1) = \mathcal{R}_{2b}$ *und*

- *(a)* $rule(e_i) = \mathcal{R}_{2a}$ *für* $2 \leq i \leq k$ *oder*
- *(b)* $rule(e_2) = \mathcal{R}_{2a}$ *und* $rule(e_i) = \mathcal{R}_1$ *für* $3 \leq i \leq k$ *oder*
- *(c)* $rule(e_2) = \mathcal{R}_{3a}$ *und* $rule(e_i) = \mathcal{R}_1$ *für* $3 \leq i \leq k$.

II. $rule(e_1) = \mathcal{R}_{3b}$ *und*

- *(a)* $rule(e_i) = \mathcal{R}_{2a}$ *mit* $2 \leq i \leq k$ *oder*
- *(b)* $rule(e_2) = \mathcal{R}_{2a}$ *und* $rule(e_i) = \mathcal{R}_1$ *für* $3 \leq i \leq k$ *oder*
- *(c)* $rule(e_2) = \mathcal{R}_{3a}$ *und* $rule(e_i) = \mathcal{R}_1$ *für* $3 \leq i \leq k$.

III. $rule(e_1) = \mathcal{R}_4$ *und*

- *(a)* $rule(e_i) = \mathcal{R}_1$ *für* $2 \leq i \leq k$.
- *(b)* $rule(e_i) = \mathcal{R}_{2a}$ *für* $2 \leq i \leq k$ *oder*
- *(c)* $rule(e_2) = \mathcal{R}_1$ *und* $rule(e_i) = \mathcal{R}_{2a}$ *für* $3 \leq i \leq k$ *oder*
- *(d)* $rule(e_2) = \mathcal{R}_{3c}$ *und* $rule(e_i) = \mathcal{R}_{2a}$ *für* $3 \leq i \leq k$.

IV. $u = s$ *und*

- *(a)* $rule(e_i) = \mathcal{R}_{2a}$ *für* $2 \leq i \leq k$ *oder*
- *(b)* $rule(e_2) = \mathcal{R}_1$ *und* $rule(e_i) = \mathcal{R}_{2a}$ *für* $3 \leq i \leq k$ *oder*
- *(c)* $rule(e_2) = \mathcal{R}_{3c}$ *und* $rule(e_i) = \mathcal{R}_{2a}$ *für* $3 \leq i \leq k$ *oder*
- *(d)* $rule(e_2) = \mathcal{R}_{2a}$ *und* $rule(e_i) = \mathcal{R}_1$ *für* $3 \leq i \leq k$ *oder*
- *(e)* $rule(e_2) = \mathcal{R}_{3a}$ *und* $rule(e_i) = \mathcal{R}_1$ *für* $3 \leq i \leq k$.

Beweis. Sämtliche nicht möglichen Kombinationen der gerichteten virtuellen Kanten in S werden nun mit Fallunterscheidungen ausgeschlossen. Dazu werden nacheinander die Kanten e_1 bis e_k sowie der Ingrad und der Outgrad der Split-Knoten u und v in Betracht gezogen.
Seien $K_1, \ldots, K_k$ die Komponenten der gerichteten virtuellen Kanten $e_1, \cdots, e_k$ mit $k \geq 3$ und $K_i^\circ := K_i - \{u, v\}$ für $1 \leq i \leq k$.
Fall I:
Die gerichtete virtuelle Kante e_1 ist ein Peak. u ist Quelle, v ist Senke in K_1 und $s \in K_1$. Da $s \notin K_1^\circ$ ist, sind u und v nicht identisch mit s. Für den Ingrad und

Outgrad gilt:

$$in(u) \geq 1$$
$$out(v) \geq 1$$

Nicht-Kandidaten für alle restlichen gerichteten virtuellen Kanten sind die Regeln $\mathcal{R}_{2b}$, $\mathcal{R}_{3b}$ und $\mathcal{R}_4$, da sie voraussetzen, dass s in der jeweils mit ihr assoziierten Komponente lokalisiert ist. Auch Regel $\mathcal{R}_{3c}$ kommt nicht in Frage. Denn wenn eine Komponente K_i mit $i \neq 1$ und $rule(K_i) = \mathcal{R}_{3c}$ existiert, dann ist u Senke in $G_{exp} - K_i$. Das widerspricht der Voraussetzung von $\mathcal{R}_{2b}$.

e_2 und ihre möglichen Ersetzungsregeln werden nun betrachtet. Regel $\mathcal{R}_1$ ist kein Kandidat, da sonst s nicht v dominiert. Regel $\mathcal{R}_{3c}$ ist auch kein Kandidat, da u eine Quelle in $G_{exp} - K_2 = K_1$ ist. Es bleiben folgende Möglichkeiten. (I.i) e_2 ist mit Regel $\mathcal{R}_{2a}$ assoziiert oder (I.ii) e_2 ist mit Regel $\mathcal{R}_{3a}$ assoziiert.

(I.i) $rule(e_2) = \mathcal{R}_{2a}$:
Es ist:

$$in(u) \geq 1 \text{ und } out(u) \geq 1$$
$$in(v) \geq 1 \text{ und } out(v) \geq 1$$

Für e_3 gibt es nun folgende Möglichkeiten:

(I.i.i) $rule(e_3) = \mathcal{R}_1$:
Es ist:

$$in(u) \geq 1 \text{ und } out(u) \geq 2$$
$$in(v) \geq 1 \text{ und } out(v) \geq 2$$

Für e_4 stehen folgende Regeln zur Auswahl: $\mathcal{R}_1$, $\mathcal{R}_{2a}$ oder $\mathcal{R}_{3a}$. Eine Untersuchung des Ingrad und Outgrad am Knoten v zeigt, dass $\mathcal{R}_{2a}$ und $\mathcal{R}_{3a}$ wegen der Ingrad/Outgrad Restriktion nicht benutzt werden können. Somit sind e_4 und die restlichen gerichteten virtuellen Kanten mit $\mathcal{R}_1$ assoziiert. Folglich gilt I.(b) des Lemmas.

(I.i.ii) $rule(e_3) = \mathcal{R}_{2a}$:
Es ist:

$$in(u) \geq 2 \text{ und } out(u) \geq 1$$
$$in(v) \geq 1 \text{ und } out(v) \geq 2$$

Für e_4 stehen folgende Regeln zur Auswahl: $\mathcal{R}_1$, $\mathcal{R}_{2a}$ oder $\mathcal{R}_{3a}$. Eine Untersuchung des Ingrad und Outgrad am Knoten v zeigt, dass $\mathcal{R}_1$ und $\mathcal{R}_{3a}$ wegen der Ingrad/Outgrad Restriktion nicht benutzt werden können. Somit sind e_4 und die restlichen gerichteten virtuellen Kanten mit $\mathcal{R}_{2a}$ assoziiert. Folglich gilt I.(a) des Lemmas.

(I.i.iii) $rule(e_3) = \mathcal{R}_{3a}$:
Es ist:

$$in(u) \geq 2 \text{ und } out(u) \geq 2$$
$$in(v) \geq 1 \text{ und } out(v) \geq 2$$

Das verstößt gegen die Ingrad/Outgrad Restriktion eines Expansiongraphen.

(I.ii) $rule(e_2) = \mathcal{R}_{3a}$:
Es ist:
$in(u) \geq 1$ und $out(u) \geq 1$
$in(v) \geq 1$ und $out(v) \geq 2$
Eine weitere Anwendung von $\mathcal{R}_{2a}$ und $\mathcal{R}_{3a}$ ist wegen der Ingrad/Outgrad Restriktion am Knoten v nicht möglich. Folglich kann für e_3 und die restlichen gerichteten virtuellen Kanten nur die Regel $\mathcal{R}_1$ angewandt werden. Es gilt I.(c) des Lemmas.

Fall II. III. und IV.:
Die Ergebnisse von II-IV können wie im Fall I durch sukzessives Betrachten aller relevanten Fälle gezeigt werden. □

Das Lemma 4.1 beschränkt die möglichen Kombinationen eines P-Knotens. Eine einfache Beobachtung zeigt, dass immer eine aufwärtsplanare Einbettung der sT-Skeletons eines P-Knotens existiert. Das ist sogar unabhängig davon, ob der dazugehörige Graph G_{exp} aufwärtsplanar ist oder nicht. Offensichtlich genügt es, wenn G_{exp} ein expandierter, azyklischer Graph ist. Aus diesem Grunde benötigt Algorithmus 3 keine Überprüfung, ob die sT-Skeletons der P-Knoten eine aufwärtsplanare Einbettung mit der Referenzkante am äußeren Face besitzen.

4.2 Eine Datenstruktur zur Aufzählung von aufwärtsplanaren Einbettungen

In diesem Abschnitt werden wir eine Datenstruktur beschreiben, die alle aufwärtsplanaren Einbettungen eines aufwärtsplanaren, 2-zusammenhängenden sT-Graphen enthält. Die Grundlagen hierfür fundieren auf der Arbeit von Betolazzis et al. [7], insbesondere auf Satz 2.8.

Ein SPQR-Baum enthält implizit alle planaren (kombinatorische) Einbettungen eines 2-zusammenhängenden aufwärtsplanaren sT-Graphen. Somit insbesondere auch die aufwärtsplanaren Einbettungen. Es muss lediglich die aufwärtsplanaren Einbettungen von den planaren (kombinatorische) Einbettungen ohne aufwärtsplanare Zeichnung gefiltert werden. Wie sich noch herausstellen wird, bestimmen nur die P-Knoten eines SPQR-Baumes, ob eine planare Einbettung auch aufwärtsplanar ist.

Wir geben zuerst einen Algorithmus zur Konstruktion der gewünschte Datenstruktur an und werden anschließend seine Korrektheit beweisen. Hierfür wird noch folgende Lemma benötigt.

Lemma 4.2. *Seien μ ein Knoten von $\mathcal{T}$, S das sT-Skeleton von μ mit der aufwärtsplanaren Einbettung Γ_S, so dass die gerichtete Referenzkante e_{ref} am äußeren Face Γ_S eingebettet ist, und e eine gerichtete virtuelle Kante von S. Ferner sei Γ_e eine aufwärtsplanare Einbettung von expansion$^+(e)$. Dabei ist e am äußeren Face von Γ_e eingebettet. Sei G_S der Graph, der daraus entsteht, indem die Kante e durch ihren pertinenten Graphen mit der Einbettung Γ_e ersetzt und anschließend die Kante e gelöscht wird. Dann ist die Einbettung Γ_{G_S} von G_S, die durch die Einbettungen Γ_S und Γ_e induziert wird, aufwärtsplanar.*

Beweis. Die Korrektheit des Lemmas folgt direkt aus dem Beweis von Lemma 2.13 in Kapitel 2.4.2. □

Algorithmus 5 Datenstruktur zur Aufzählung der aufwärtsplanaren Einbettungen eines 2-zusammenhängenden sT-Graphen.

Eingabe: 2-zusammenhängender, aufwärtsplanarer sT-Graph G.
Ausgabe: SPQR-Baum $\mathcal{T}$ mit sT-Skeletons, die möglichen Wurzeln $\mu_1, \cdots, \mu_k$ von $\mathcal{T}$.
procedure EnumUpwardPlanarEmbeddings(G)
    Konstruiere Expansionsgraph G_{exp} von G
    Konstruiere den SPQR-Baum $\mathcal{T}$ von G_{exp}
    Berechne die möglichen Wurzel $\mu_1, \cdots, \mu_k$ von $\mathcal{T}$ im Sinne von Satz 2.8
    return $\mathcal{T}$, $\mu_1, \cdots, \mu_k$
end procedure

Satz 4.1. *Die von Algorithmus 5 berechnete Datenstruktur enthält implizit alle aufwärtsplanaren Einbettungen eines 2-zusammenhängenden sT-Graphen $G = (V, A)$. Der Algorithmus hat eine Laufzeit von $\mathcal{O}(|V| + |A|)$.*

Beweis. Wir müssen folgendes zeigen:

(a) Es existiert eine eins zu eins Abbildung der aufwärtsplanaren Einbettungen eines 2-zusammenhängenden Graphen G zu den aufwärtsplanaren Einbettungen des Expansionsgraphen G_{exp} von G.

(b) Sei $\Gamma_{G_{exp}}$ eine beliebige, aufwärtsplanare Einbettung von G_{exp}, dann sind die von $\Gamma_{G_{exp}}$ auf die sT-Skeletons $S_1, \ldots, S_n$ von $\mathcal{T}$ induzierten Einbettungen $\Gamma_{S_1}, \ldots, \Gamma_{S_n}$ aufwärtsplanar. Ferner ist die Referenkante der Skeletons in den jeweiligen äußeren Faces der Einbettungen $\Gamma_{S_1}, \ldots, \Gamma_{S_n}$ zu finden.

(c) Seien die Einbettungen der sT-Skeletons von $\mathcal{T}$ alle aufwärtsplanar, so dass die jeweilige Referenzkante am äußeren Face eingebettet ist. Dann ist die von den Einbettungen der sT-Skeletons induzierten Einbettung $\Gamma_{G_{exp}}$ auf G_{exp} aufwärtsplanar.

zu (a):
Sei Γ_G eine beliebige, aufwärtsplanare Einbettung von G. Eine zu Γ_G korrespondierende Einbettung $\Gamma_{G_{exp}}$ kann durch Expansion der Knoten von G gewonnen werden. Umgekehrt kann durch die Umkehrung des Expansionsprozesses aus $\Gamma_{G_{exp}}$ die Einbettung Γ_G gewonnen werden. Somit kann aus einer Einbettung in G_{exp} eine Einbettung in G konstruiert werden und umgekehrt.

zu (b): (indirekt)
Angenommen $\Gamma_{G_{exp}}$ ist aufwärtsplanar und es existiert ein Knoten μ, dessen sT-Skeleton S nicht aufwärtsplanar ist oder S ist aufwärtsplanar aber die Referenzkante kann nicht in das äußere Face eingebettet werden. Dann kann μ nur einer der folgenden Knoten sein:

μ ist ein Q-Knoten: Ein Q-Knoten hat genau eine Einbettung. Da G_{exp} aufwärtsplanar ist, sind nach Satz 2.8 alle sT-Skeletons der Q-Knoten aufwärtsplanar. μ ist folglich kein Q-Knoten.

μ ist ein S-Knoten: Ein S-Knoten hat genau eine Einbettung. Da G_{exp} aufwärtsplanar ist, sind nach Satz 2.8 alle sT-Skeletons der S-Knoten aufwärtsplanar. μ ist folglich kein S-Knoten.

μ ist ein R-Knoten: Da G_{exp} aufwärtsplanar ist, gibt es nach Satz 2.8 für alle sT-Skeletons der R-Knoten mindestens eine aufwärtsplanare Einbettung mit der Referenzkante am äußeren Face. Da die sT-Skeletons nur zwei Einbettungen besitzen, wobei die zweite Einbettung eine Spiegelung der ersten ist, ist μ kein R-Knoten.

μ ist ein P-Knoten: Wenn eine Einbettung eines P-Knotens existiert, deren Referenzkante nicht nach außen gezeichnet werden kann, dann besitzt $\Gamma_{G_{exp}}$ keine Zeichnung mit s am äußeren Face. Das widerspricht der Voraussetzung, das $\Gamma_{G_{exp}}$ aufwärtsplanar ist. Folglich kann es nur nicht aufwärtsplanare Einbettungen eines P-Knotens geben, deren Referenzkante am äußeren Face eingebettet ist. Nach Lemma 4.1 kann das nur der Fall sein, wenn eine gerichtete virtuelle Kante ein Peak ist, der von zwei gerichteten Kanten eingeschlossen wird. Eine Untersuchung der Ersetzungsregeln zeigt, dass die pertinenten Graphen der beiden gerichteten Kanten entweder eine Struktur homöomorph zu einem Valley oder eine Struktur homöomorph zu einer gerichteten Kante besitzen. Somit schließt diese Struktur den pertinenten Graphen des Peaks ein. Der pertinente Graph des Peaks enthält jedoch einen Knoten t, der höher gezeichnet werden muss als u und v. Somit kann $\Gamma_{G_{exp}}$ nicht aufwärtsplanar sein. μ ist kein P-Knoten.

Somit folgt (b).

zu (c):
Sei $\mathcal{T}_{\mu_0}$ ein SPQR-Baum gemäß Satz 2.8 mit der Wurzel μ_0. μ_0 ist ein Q-Knoten, der zu einer Kante korrespondiert, deren Startknoten die Quelle s ist. Die Wahl des Wurzelknotens von $\mathcal{T}$ hat keinen Einfluss auf die Einbettungen, die die Skeletons von $\mathcal{T}$ induzieren. Jedoch verändern sich bei einer anderen Verwurzelung die Referenzkanten des SPQR-Baumes. Um das zu verhindern, bleiben alle Referenzkanten der sT-Skeletons von $\mathcal{T}_{\mu_0}$ auch Referenzkanten der sT-Skeletons von $\mathcal{T}$.

Seien S ein sT-Skeleton von μ und Γ_S eine aufwärtsplanare Einbettung von S, so dass die gerichtete virtuelle Referenzkante e_{ref} am äußeren Face eingebettet ist. Ferner seien $e_1, \ldots, e_l$ die gerichteten virtuellen Kanten von S. (c) wird nun mit einer Induktion über die Höhe h von $\mathcal{T}$ bewiesen.
Induktionsbeginn: $h = 1$
$\mathcal{T}$ besteht nur aus dem Q-Knoten μ. Der Graph G_{exp} und S besitzen jeweils genau eine Einbettung. Die Einbettung von S ist nach Voraussetzung von (c) aufwärtsplanar.
Induktionsschritt: $h = k > 1$
Seien $e_1, \ldots, e_l$ die gerichteten virtuellen Kanten von S und $\mathcal{T}_1, \ldots, \mathcal{T}_l$, die mit den gerichteten virtuellen Kanten assoziierten Teilbäume von $\mathcal{T}$. Sei Γ_i eine aufwärtsplanare Einbettung des sT-Skeleton von e_i. Ferner sei Γ_{G_i} die Einbettung von $expansion^+(e_i)$. Alle $\mathcal{T}_1, \ldots, \mathcal{T}_l$ haben eine Höhe kleiner als h.
Induktionsvoraussetzung: Γ_{G_i} mit $1 \leq i \leq l$ ist aufwärtsplanar und die Referenzkante ist im äußeren Face eingebettet. Die Einbettung Γ_{G_i}, die durch Γ_i und die Einbettungen der sT-Skeletons von $\mathcal{T}_i$ festgelegt wird, ist aufwärtsplanar.
Sukzessiv wird jede gerichtete virtuelle Kante e_i der Einbettung Γ_S durch G_i mit der Einbettung Γ_{G_i} ersetzt und dann $e_i \in G_i$ gelöscht. Der daraus resultierende Graph sei G'_i mit der Einbettung $\Gamma_{G'_i}$. Nach Lemma 4.2 ist $\Gamma_{G'_i}$ aufwärtsplanar. Es ist $G_{exp} = G'_l$ und $\Gamma_{G_{exp}} = \Gamma_{G'_l}$.

Die Laufzeit geht aus dem Algorithmus 3 hervor. □

Die Einbettungen der sT-Skeletons legen nicht nur eine kombinatorische Einbettung, sondern sie legen auch ein Face $\alpha_{\Gamma_{G_{exp}}}$ fest, das in eine planare Aufwärtszeichnung als äußeres Face gewählt werden kann. Wenn $\Gamma_{G_{exp}}$ die Faces $\alpha^1_{\Gamma_{G_{exp}}}, \ldots, \alpha^k_{\Gamma_{G_{exp}}}$ besitzt, die als äußeres Face eines planaren Aufwärtszeichnung gewählt[1] werden kann, dann kann aus $\Gamma_{G_{exp}}$ k verschiedene aufwärtsplanare Einbettung gewonnen werden.

Durch Spiegelung der R-Knoten und Permutation der virtuellen Kanten der P-

[1] In Kapitel 5 wird näher auf die Wahl des äußeren Face eingegangen.

Knoten kann die (kombinatorische) Einbettung von G_{exp} bestimmt werden, die mindestens eine aufwärtsplanare Einbettung enthält. Wobei bei einem P-Knoten darauf geachtet werden muss, dass die Einbettung durch die Permutation aufwärtsplanar bleibt. Beim Vergleichen der Anzahl der kombinatorischen mit den aufwärtsplanaren Einbettungen eines aufwärtsplanaren, 2-zusammenhängenden Expansionsgraphen, kann festgestellt werden, dass die Anzahl der kombinatorische Einbettungen, die keine Aufwärtszeichnung besitzen, nur durch die P-Knoten bestimmt werden. Ein P-Knoten mit k gerichteten virtuellen Kanten hat $(k-1)!$ (kombinatorische) Einbettungen und mindestens $(k-2)!$ aufwärtsplanare Einbettungen. Zum Schluss dieses Kapitels geben wir noch eine Illustration der möglichen P-Knoten sowie das folgende Korollar an.

Korollar 4.1. *Enthält der SPQR-Baum $\mathcal{T}$ eines aufwärtsplanaren Graphen G_{exp} keinen P-Knoten, so besitzen alle Einbettungen von G_{exp} eine aufwärtsplanare Zeichnung.*

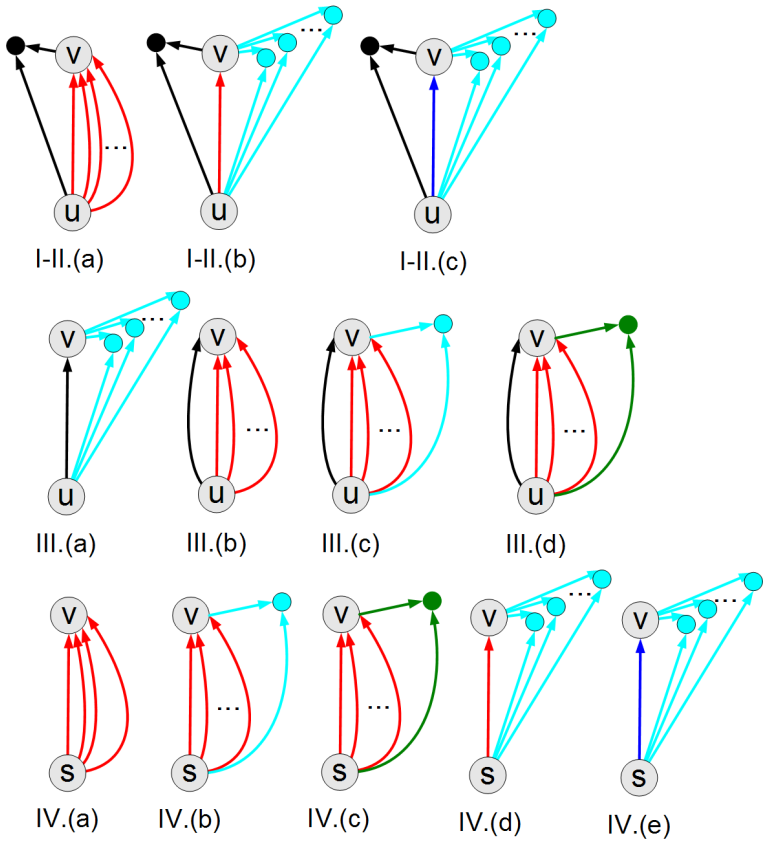


Abbildung 4.1: Mögliche Kombinationen der gerichteten virtuellen Kanten eines P-Knotens nach Lemma 4.1. Türkise virtuelle Kanten sind mit Regel $\mathcal{R}_1$, rote virtuelle Kanten sind mit Regel $\mathcal{R}_{2a}$, blaue virtuelle Kanten sind mit Regel $\mathcal{R}_{3a}$ und grüne virtuelle Kanten sind mit Regel $\mathcal{R}_{3c}$ assoziiert. Die schwarzen Kanten sind die Referenzkanten und sind im Falle I-II mit Regel $\mathcal{R}_{2b}$ oder Regel $\mathcal{R}_{3b}$ und im Falle III mit Regel $\mathcal{R}_4$ assoziiert.

5 Kreuzungsminimales Einfügen einer Kante in einen eingebetteten sT-Graphen

Wir werden zuerst eine Methode zur Berechnung einer aufwärtskonsistenten Zuweisung angeben. Dann werden wir eine Charakterisierung eines aufwärtskonsistenten Pfades für einen sT-Graphen vorstellen. Basierend auf dieser Charakterisierung wird dann einen Algorithmus für das *Edge-Insertion-Problem* angegeben. Dieser Algorithmus berechnet einen kreuzungsminimalen, aufwärtskonsistenten Einfügepfad für eine gegebene Einbettung eines sT-Graphen.

Sei in diesem Kapitel $G = (V, A)$ ein sT-Graph und Γ_G eine aufwärtsplanare Einbettung mit der aufwärtskonsistenten Zuweisung $\mathcal{A}$. Seien α_{Γ_G} ein äußeres von Γ_G und x und y die Anfangs- und Endknoten des zu berechnenden Einfügepfades gemäß Problembeschreibung 3.1.

5.1 Berechnung einer aufwärtskonsistenten Zuweisung

Eine aufwärtskonsistente Zuweisung für einen aufwärtsplanaren, eingebetteten Digraphen kann in einer Zeit von $\mathcal{O}(n + m + r^2)$ berechnet werden [6], wobei r die Anzahl der Senken, n die Anzahl der Knoten und m die Anzahl der Kanten des Digraphen sind. In diesem Abschnitt geben wir eine Methode zur Berechnung einer aufwärtskonsistenten Zuweisung $\mathcal{A}$ eines aufwärtsplanaren sT-Graphen $G = (V, A)$ mit Laufzeit $\mathcal{O}(|V| + |A|)$ an. Die Methode basiert auf einem Satz von Bertolazzi et al. [7]. Folgenden Definition wird für diesen Satz benötigt:

Definition 5.1. Der *Face-Senken-Graph* F von G wird wie folgt konstruiert:

1. Die Knoten von F sind die Faces und Senken-Switches von G.

2. Sei f^* der zur Face f korrespondierende Knoten in F. Der Graph F besitzt eine Kante (f^*, v), wenn v ein Senken-Switch von Face f ist.

Satz 5.1 ([7]). *Sei G ein planarer sT-Graph mit der Einbettung Γ_G und f ein Face von Γ_G. Genau dann hat die Einbettung Γ_G eine aufwärtsplanare Zeichnung mit f als äußeres Face, wenn folgende Bedingungen gelten:*

(a) Der Graph F ist ein Wald.

(b) Genau ein Baum T von F enthält keinen inneren Knoten von G und jeder andere Baum enthält genau einen inneren Knoten von G.

(c) Der zu f korrespondierende Knoten f^* ist in T.

(d) Die Quelle s von G ist in f enthalten.

In einem inneren Face f von Γ_G gibt es nach Beobachtung 3.1 einen ausgezeichneten Senken-Switch t, der nicht von $\mathcal{A}$ an f zugewiesen wird, während alle anderen Senken-Switches von f an f selbst zugewiesen werden. Es kann beobachtet werden, dass jeder Knoten $t' \neq t$ in f, der Senken-Switch in einem zu f adjazenten Face f' ist, nicht an f' zugewiesen wird. t' ist somit der ausgezeichnete Senken-Switch in f'.

Beobachtung 5.1. *Sei t der ausgezeichnete Senken-Switch in f.*

(1) Jeder Knoten $t' \neq t$ eines inneren Faces f, der Senken-Switch in einem zu f adjazenten Face f' ist, wird nicht an f' zugewiesen.

(2) Jeder Knoten v in äußerem Face α_{Γ_G}, der Senken-Switch in einem inneren Face f ist, wird nicht an f zugewiesen.

Mit dieser Beobachtung und mit dem Satz 5.1 kann eine aufwärtskonsistente Zuweisung für Γ_G wie folgt hergeleitet werden:

1. Berechne F und überprüfe Bedingung (b) und (c) des Satzes 5.1. Wenn sie nicht erfüllt ist, dann gibt es keine aufwärtskonsistente Zuweisung.
2. Ein Face f, das s enthält und dessen korrespondierender Knoten $f^* \in F$ in T liegt, kann ein äußeres Face von Γ_G sein. Falls kein solches Face existiert, dann gibt es keine aufwärtskonsistente Zuweisung für Γ_G.
3. Wähle ein äußere Face α_{Γ_G}. Nach Konstruktion eines Face-Senken-Graphen sind die zu f^* adjazenten Senken-Switches alle Senken-Switches des äußeren Face α_{Γ_G}. Da sie auch Senken von G sind, werden sie zusammen mit der Quelle s dem äußeren Face α_{Γ_G} zugewiesen.
4. Markiere alle Knoten des äußeren Faces α_{Γ_G} und markiere α_{Γ_G} als „berechnet“.
5. Für jedes Faces f, das adjazent zu einem als berechnet markierter Face ist und einen markierten Senken-Switch besitzen:
   Weise alle nicht markierten Senken-Switches von f Face f selbst zu und markiere f als „berechnet“.

6. Fahre mit Schritt 5 fort bis alle Faces als „berechnet“ markiert sind.

Jeder dieser einzelnen Schritte kann in Zeit $\mathcal{O}(|V| + |A|)$ durchgeführt werden. Die gesamte Laufzeit beträgt folglich $\mathcal{O}(|V| + |A|)$.

Lemma 5.1. *Die Berechnung einer aufwärtskonsistenten Zuweisung für einen eingebetteten sT-Graphen $G = (V, A)$ kann in Zeit $\mathcal{O}(|V| + |A|)$ durchgeführt werden.*

5.2 Charakterisierung eines aufwärtskonsistenten Einfügepfades

Seien x' und y' zwei nicht adjazente Knoten des Faces f, die hier *Eintritts-* und *Austrittsknoten* in bzw. aus dem Face f genannt werden. Ein Pfad, der f traversiert, kann f über den Knoten x' betreten und dann über den Knoten y' f wieder verlassen. Dabei kann x' mit y' entweder direkt durch die Kanten (x', y') verbunden werden oder sie werden durch einen Pfad verbunden, der innerhalb von f verläuft. Es wird nun den Fall betrachtet, in der x' direkt durch die Kante (x', y') mit y' verbunden ist. Sei $\Gamma_{G'} := \Gamma_G \cup (x', y')$ die Einbettung, die daraus resultiert, indem in der Einbettung Γ_G die Kante (x', y') eingefügt wird. $\Gamma_{G'}$ ist dann eindeutig durch Γ_G und die Kante (x', y') festgelegt.

Falls f ein inneres Face von Γ_G ist, so gibt es nach Beobachtung 3.1 genau einen ausgezeichneten Senken-Switch t in f, der nicht von $\mathcal{A}$ an f zugewiesen wird. f wird nun im Uhrzeigersinn von x' bis zum Knoten t — falls f das äußere Face ist, bis zur Quelle s — durchlaufen. Die Knoten von f werden dabei aufsteigend nummeriert. Danach wird f gegen den Uhrzeigersinn von x' bis zum Knoten t bzw. s durchlaufen und die Knoten werden wieder aufsteigend nummeriert. In beiden Fällen werden mit Ausnahme von s bzw. t die Knoten, die schon eine Nummerierung besitzen, nicht mehr nummeriert[1]. Die Knoten t und s können somit zwei verschiedene Nummern besitzen. Wenn das der Fall ist, so bekommt t bzw. s die größte der beiden Nummern. Die Nummerierung der Knoten von f in Abhängigkeit von x' wird mit $num(f, x')$ beschrieben. Es kann nun festgestellt werden, dass $\Gamma_{G'}$ aufwärtsplanar ist, solange y' ein Senken-Switch in f ist. Wenn y' ein Quellen-Switch in f ist, dann ist die Einbettung $\Gamma_{G'}$ bei jeder beliebigen Wahl des Knoten x' für die Kante (x', y') nicht aufwärtsplanar. Das bedeutet in Bezug auf die Berechnung eines Einfügepfades, dass alle Kanten gesperrt werden muss, die beim Durchlaufen in (gegen) den Uhrzeigersinn des Faces f von x' nach t bzw. s gegen die Laufrichtung zeigen. Diese Beobachtung kann durch $num(f, x')$ ausgedrückt werden.

[1] In einem nicht 2-zusammenhängenden Graphen kann beim durchlaufen eines Face-Abschnittes ein Knoten mehrmals vorkommen.

Beobachtung 5.2. *Seien x' und y' zwei disjunkte, nicht adjazente Knoten eines Faces f. Wenn y' ein innerer Knoten in f ist, dann seien $u \rightarrow y'$ und $y' \rightarrow v_i$[2] die zu y' inzidenten Kanten in f und $\#u$ und $\#v_i$ die Nummer der Knoten u und v_i aus der Nummerierung $num(f, x')$. Sei $\Gamma_{G'} := \Gamma_G \cup (x', y')$. Dann ist $\Gamma_{G'}$ aufwärtsplanar wenn eine der folgenden Bedingungen gilt:*

(a) y' ist Senken-Switch in f und wenn f ein inneres Face ist, dann gilt zusätzlich noch $x' \neq t$.

(b) y' ist ein innerer Knoten in f und $\#v_i > \#u$ und wenn f ein inneres Faces ist, dann gilt zusätzlich noch $x' \neq t$.

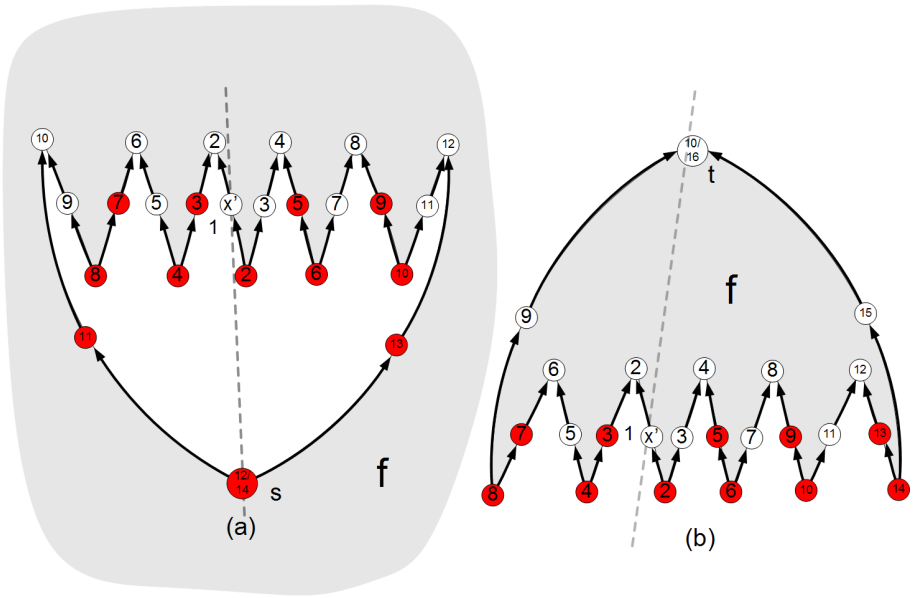


Abbildung 5.1: Die Knotennummerierung in einem äußeren Face (a) und einem inneren Face (b). Die gestrichelte Linie symbolisiert die Aufteilung der Kanten des Faces in zwei Face-Abschnitten. Die roten Knoten sind keine möglichen Zielknoten der Kante $x' \rightarrow y'$.

Basierend auf dieser Beobachtung wird nun eine Charakterisierung eines Einfügepfades für sT-Graphen angegeben.

Lemma 5.2. *Sei P ein Einfügepfad für die beiden nicht adjazenten Knoten x und y und die aufwärtsplanare Einbettung Γ_G. Seien $f_1, \ldots, f_n$ die Faces von Γ_G, die der Pfad P traversiert. Genau dann ist P aufwärtskonsistent, wenn die folgenden Bedingungen gelten:*

[2] In einem nicht 2-zusammenhängender sT-Graph, kann der Knoten y' in Face f der Startknoten von mehrere Kanten, z.B. die Kanten $(y', v_1), \ldots, (y', v_k)$, von f sein. v_i ist dann ein Knoten aus $\{v_1, \ldots, v_k\}$.

(a) *Die Kanten von P kreuzen sich nicht.*

(b) *Für alle Kanten des Pfades P in den jeweiligen Faces $f_1, \ldots, f_n$ gilt die Beobachtung 5.2.*

(c) *Sei f_i ein Face, in dem ein Eintrittsknoten x' nicht direkt durch eine gerichtete Kante mit dem Austrittsknoten y' verbunden ist, sondern über die Knoten $d_1, \ldots, d_k$ (siehe Abbildung 5.2). Dann muss für die Kante (x', d_i) mit $1 \leq i \leq k$ die Beobachtung 5.2 gelten.*

Beweis. „⇒“ Sei P ein aufwärtskonsistenter Einfügepfad. Dann traversiert P die Faces $f_1, \ldots, f_n$.
Angenommen (a) gilt nicht und P enthält einen Kreis. Dann existiert keine aufwärtsplanare Zeichnung für die resultierende Einbettung $\Gamma_{G'} := \Gamma_G \cup P$. Folglich ist P nicht aufwärtskonsistent. Widerspruch!
Angenommen (b) gilt nicht. Dann gibt es ein Face f in dem eine Kante e existiert, so dass für e die Beobachtung 5.2 nicht gilt. Somit ist $\Gamma_{G'} := \Gamma_G \cup e$ nicht aufwärtsplanar. Da P die Kante e enthält, ist P nicht aufwärtskonsistent. Widerspruch!
Angenommen (c) gilt nicht. Dann gibt es ein Face f mit einem Pfad $P^* := x' \rightsquigarrow y'$, der innerhalb von f verläuft, und es gibt einen Knoten $d \in P^*$, so dass die Beobachtung 5.2 für die Kante (x', d) nicht gilt. Somit gibt es einen Pfad in P^* homöomorph zu der Kante (x', d). Folglich ist $\Gamma_{G'} := \Gamma_G \cup P^*$ nicht aufwärtsplanar und, da P den Pfad P^* enthält, ist P nicht aufwärtskonsistent.

„⇐“ Sei P ein azyklischer Pfad von x nach y. Für jede Kante von P in den jeweiligen Faces $f_1, \ldots, f_n$ gilt die Eigenschaft (b) und in jedem Face gilt die Eigenschaft (c). durch Induktion über die Anzahl k der Kanten in P wird nun die Behauptung bewiesen.
Sei $e_1, \ldots, e_k$ die Kantenreihenfolge in P.
Induktionsbeginn: k=1
Der Pfad P besteht aus genau einer Kante. Diese Kante ist $e_1 = (x, y) = P$. Für diese Kante gilt die Eigenschaft (b). Nach Beobachtung 5.2 ist $\Gamma_{G'} := \Gamma_G \cup P$ aufwärtsplanar. Somit ist P aufwärstkonsistent.
Induktionsschritt: k>1
Sei die Behauptung richtig für den Pfad P_{k-1} mit den Kanten $e_1, \ldots, e_{k-1}$. Dann ist die Einbettung $\Gamma_{G'}^{k-1} := \Gamma_G \cup P_{k-1}$ aufwärtsplanar. Sei d der letzte Knoten des Pfades P und sei f das Face, in dem d und y lokalisiert sind. Wir betrachten nun die k-te Kante $e_k = (d, y)$. Diese Kante kreuzt keine andere Kante aus P_{k-1}. Da sonst $P = P_{k-1} + e_k$ einen Kreis enthält. Das widerspricht Eigenschaft (a). Falls d ein Eintrittsknoten ist, dann gilt für e_k nach Voraussetzung Eigenschaft (b) folglich ist $\Gamma_{G'} := \Gamma_{G'}^{k-1} \cup (d, y)$ aufwärtsplanar. Somit ist P aufwärtskonsistent. Falls d keinen Eintrittsknoten und P_{k-1} konkateniert mit e_k nicht aufwärtskonsistent ist, dann gibt es einen Eintrittsknoten $d' \in f$ und einen Pfad $P^* := d' \rightsquigarrow d$. Alle Kanten von P^*

verlaufen innerhalb von f und nach Eigenschaft (a) kreuzen sie sich nicht untereinander. Da $P_{k-1} + e_k$ nicht aufwärtskonsistent ist, existiert mindestens ein Knoten d^* in $P_{k-1} + e_k$, so dass der Pfad $d^* \rightsquigarrow y$ nicht aufwärtskonsistent ist. Es gibt nun zwei Möglichkeiten:

(1) $d^* \notin P^*$:
Da P über den Eintrittsknoten d' verläuft, ist der Pfad $d' \rightsquigarrow y$ auch nicht aufwärtskonsistent. Somit gilt für die Kante (d', y) die Beobachtung 5.2 nicht. Widerspruch zur Eigenschaft (c)!

(2) $d^* \in P^*$:
Der Pfad $d^* \rightsquigarrow y$ ist nicht aufwärtskonsistent. Nach Eigenschaft (c) erfüllen die Kanten (d', d^*) und (d', y) die Beobachtung 5.2. Folglich muss es eine Kante in $d^* \rightsquigarrow y$ geben, welche die Beobachtung 5.2 nicht erfüllt. Widerspruch zur Eigenschaft (b)!

Folglich ist $\Gamma_{G'} := \Gamma_G \cup P$ aufwärtsplanar und somit ist P aufwärtskonsistent. □

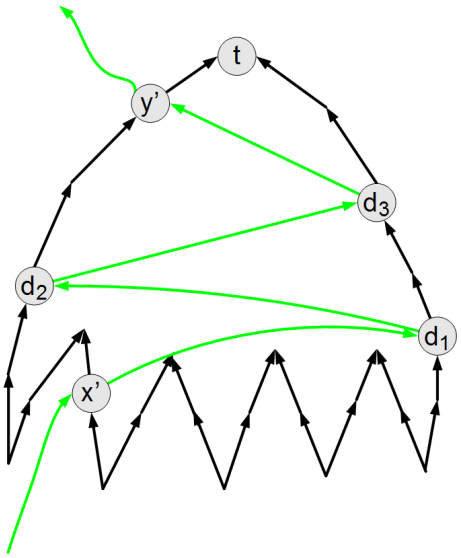


Abbildung 5.2: Illustration zum Lemma 5.2 (c): x' ist der Eintrittsknoten und y' ist der Austrittsknoten in bzw. aus dem Face f. Die beiden Knoten sind nicht direkt durch eine Kante verbunden sondern durch einen Pfad über die Knoten d_1, d_2 und d_3.

Eigenschaften

Der so charakterisierte Einfügepfad hat folgende Eigenschaften.

1. Sämtliche Pfadabschnitte eines aufwärtskonsistenten Pfades sind aufwärtskonsistent.

2. Ein konkatenierter Pfad $x \rightsquigarrow w \rightsquigarrow y$ zweier aufwärtskonsistenter Pfade $x \rightsquigarrow w$ und $w \rightsquigarrow y$ ist nicht zwingend aufwärtskonsistent.

3. Ein Pfadabschnitt $u \rightsquigarrow v$ eines kürzesten aufwärtskonsistenten Einfügepfades $x \rightsquigarrow y$ ist nicht zwingend ein kürzester aufwärtskonsistenter Pfad von u nach v.

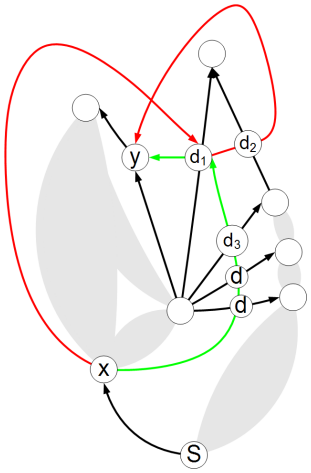


Abbildung 5.3: Illustration zu den Eigenschaften eines aufwärtskonsistenten Einfügepfades: Die ausgemalten Teilgraphen symbolisieren einen dichten Teilgraphen, deren Traversierung hohe Kosten verursachen. Der grüne Pfad ist der kürzeste aufwärtskonsistente Pfad von x nach y. Es werden vier Kanten gekreuzt.

Punkt 1 ist trivial, Punkt 2 und 3 folgt aus dem Beispiel in Abbildung 5.3. Der grüne Pfad ist der kürzeste aufwärtskonsistente Pfad von x nach y. Es werden vier Kanten gekreuzt. Der kürzeste aufwärtskonsistente Pfad zum Knoten d_1 ist die Kante $x \rightarrow d_1$. Dieser Pfad ist kein Pfadabschnitt des grünen Pfades. Der rote Pfad, zeigt die Konkatenation dreier kürzester aufwärtskonsistenter Pfade, $x \rightsquigarrow d_1$, $d_1 \rightsquigarrow d_2$ und $d_2 \rightsquigarrow y$. Er ist jedoch kein aufwärtskonsistenter Einfügepfad.

5.3 Existenz eines aufwärtskonsistenten Pfades

In Kapitel 3.2 haben wir festgestellt, dass durchaus aufwärtsplanare Einbettungen existieren, die keine aufwärtskonsistenten Einfügepfade besitzen. Ein einfaches Ergebnis kann hierzu aus dem Satz 5.1 hergeleitet werden. Sei F^* der gerichtete Face-

Senken-Graph. Dieser kann aus der aufwärtskonsistenten Zuweisung $\mathcal{A}$ der aufwärtsplanaren Einbettung Γ_G konstruiert werden, indem die Kanten des ungerichteten Face-Senken-Graphen F gemäß der Zuweisung der Senken von $\mathcal{A}$ gerichtete werden. D.h., wenn der zum äußeren Face α_{Γ_G} korrespondierenden Knoten v in dem Baum $T \subset F$ ist, dann wird T an v gewurzelt. Alle Kanten werden dann so gerichtet, dass sie in Richtung Wurzel zeigen. Für ein inneres Face f gilt: Wenn t der ausgezeichnete Senken-Switch von f ist, dann gibt es in F eine Kante von f^* nach v_t. f^* ist dabei der zu Face f und v_t der zu t korrespondierenden Knoten in F. Die Kante wird zur einer gerichtete Kante mit dem Startknoten f^* und dem Endknoten v_t. Alle anderen Knoten von F, die zu den Senken-Switches ungleich t von f korrespondieren, werden so gerichtete, dass sie in Richtung v_t zeigen.

Korollar 5.1. *Seien $G = (V, A)$ ein sT-Graph und F^* der gerichtete Face-Senken-Graph einer aufwärtsplanaren Einbettung Γ_G von G. Dann gilt:*

(a) Γ_G besitzt genau dann einen aufwärtskonsistenten Einfügepfad, wenn der Graph $G^+ := G \cup F^* \cup (x, y)$ azyklisch ist.

(b) Sei G nun 2-zusammenhängend und $\mathcal{T}$ ein SPQR-Baum von G. Sei S ein Skeleton eines R-Knotens von $\mathcal{T}$ mit den aufwärtsplanaren Einbettungen $\Gamma_S^1, \ldots, \Gamma_S^k$. Ferner seien x und y zwei Knoten des Skeletons S und $F_S^{*,i}$ der gerichtete Face-Senken-Graph der Einbettung Γ_S^i. Dann hat G keinen aufwärtskonsistenten Einfügepfad für x und y, wenn $S^+ := S \cup F_S^{*,i} \cup (x, y)$ mit $1 \leq i \leq k$ azyklisch ist.

Nach diesem Korollar kann somit in Zeit $\mathcal{O}(|V|+|A|)$ getestet werden, ob eine Einbettung einen aufwärtskonsistenten Einfügepfad besitzt. Der Pseudo-Code hierfür ist in Algorithmus 6 angegeben.

Wir geben nun ein Beispiel für einen Graphen an, der keinen aufwärtskonsistenten Einfügepfad besitzt (siehe Abbildung 5.4). Der Graph ist 3-zuammenhängend und besitzt zwei Faces f_0 und f_1, die als äußeres Face gewählt werden können.

Markieren eines Teilgraphen in Abhängigkeit des Startknotens x des Einfügepfades

Hat Γ_G keinen aufwärtskonsistenten Einfügepfade für x und y, dann wird der Knoten y in einer entsprechenden aufwärtsplanaren Zeichnung von zwei Pfaden $P_l := s \rightsquigarrow x$ und $P_r := s \rightsquigarrow x$ eingeschlossen (siehe Abbildung 5.5). Hierbei kann y ein Knoten eines der beiden Pfade sein. Die Kanten des Teilgraphen G_u, der von P_l und P_r eingeschlossen wird, dürfen nicht vom Einfügepfad gekreuzt werden. Wir markieren diesen Teilgraphen mit Algorithmus 7 und sperren später die Kanten damit kein Einfügepfad durch sie hindurch führt. Der markierte Teilgraph G_u ist dabei der

Algorithmus 6 Überprüfe ob ein aufwärtskonsistenter Einfügepfad existiert.

Eingabe: aufwärtsplanare Einbettung Γ_G von G und die Knoten x und y.
Ausgabe: **true** oder **false**
function HasConsistentInsertionPath(Γ_G, x, y)
    if $\exists x \rightsquigarrow y$ **then**
        return true;
    end if
    Berechne den gerichteten Face-Senken-Graphen F^* von Γ_G;
    Berechne $G^+ := G \cup F^* \cup (x, y)$;
    if G^+ ist kreisfrei **then**
        return true;
    else
        return false;
    end if
end function

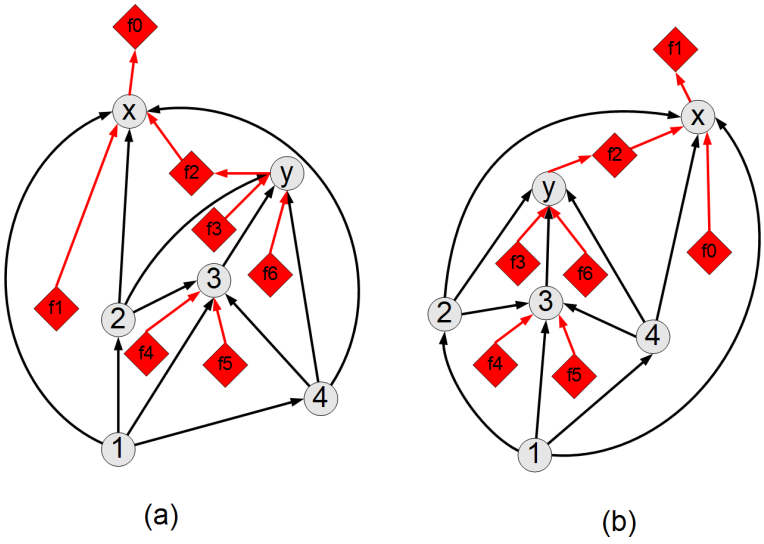


Abbildung 5.4: Ein Graph ohne aufwärtskonsistente Einfügepfade: (a) eine Zeichnung mit f_0 als äußeres Face; (b) eine Zeichnung mit f_1 als äußeres Face. In beiden Fällen sind die Einbettungen identisch. Der gerichtete Face-Senken Graph F^* ist in rot gezeichnet. Das Einfügen der Kante (x, y) verursacht sowohl in (a) als auch in (b) einen Kreis in dem Graphen $G^+ := G \cup F^* \cup (x, y)$.

größtmögliche Teilgraph, der von P_l und P_r eingeschlossen wird. Der Algoritmus hat eine Laufzeit von $\mathcal{O}(|V| + |A|)$ und funktioniert wie folgt:

In Zeile 7-14 werden die linke und die rechte in x eingehenden Kanten e_l und e_r ermittelt. Im Falle, dass x eine Senke ist, können die Kanten e_l und e_r durch eine Kantenaufzählung des Faces f im Uhrzeigersinn eindeutig festgestellt werden. Wenn f nur eine zu x inzidente Kante enthält, dann ist diese Kante sowohl e_l als auch e_r. Falls x keine Senke ist, dann sind e_l und e_r eindeutig bestimmt. Die Kantenordnung der Knoten lässt sich wegen der Bimodalität von G nämlich in zwei Sequenzen partitionieren. Die erste Sequenz besteht aus den eingehenden und die zweite aus den ausgehenden Kanten. e_l ist dann die letzte Kante und e_r die erste Kante in der Sequenz der eingehenden Kanten. Falls nur eine Kante in der Sequenz existiert, dann ist $e_l = e_r$.

Der Algorithmus berechnet die Pfade P_l und P_r, indem er von x aus die beiden Pfade zurück bis zur Quelle s verfolgt. Bei der Berechnung des linken Pfades P_l (Zeile 18-24) wird stets die linke Kante und bei der Berechnung des rechten Pfades P_r (Zeile 26-32) stets die rechte Kante in der Sequenz der eingehenden Kante benutzt. Dadurch wird gewährleistet, dass der Teilgraph, der von P_l und P_r eingeschlossen wird, maximal ist. Bei der Berechnung der beiden Pfade werden die Kanten entfernt, die nicht zum Teilgraphen G_u gehören. G_u wird anschließend traversiert (Zeile 34) und alle erreichten Kanten werden markiert. Die entfernten Kanten werden am Ende wieder zu G hinzugefügt.

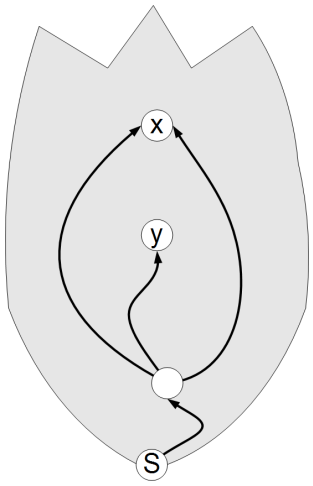


Abbildung 5.5: Es existieren kein aufwärtskonsistenter Einfügepfad, wenn in einer Einbettung mit festgelegtem äußeren Face der Zielknoten y von zwei Pfaden, die von s nach x verlaufen, eingeschlossen wird.

Algorithmus 7 Markiere den gesperrten Teilgraphen unterhalb von x.

Eingabe: aufwärtsplanare Einbettung Γ_G, aufwärtskonsistente Zuweisung $\mathcal{A}$ und Knoten x.
Ausgabe: Markierter, nicht vom Einfügepfad passierbarer Teilgraph G_u.
procedure MARK-$G_u(\Gamma_G, \mathcal{A}, x)$
  if $x =$ Quelle s **then**
    return;
  end if
  if $out(x) = 0$ **then**
    Sei $x \in \mathcal{A}(f)$.
    Identifiziere die linke und rechte eingehende Kante e_l und e_r von x in f;
  else
    Berechne die erste und die letzte Kante e_r und e_l
    in der Sequenz der eingehenden Kanten von x;
    Entferne alle ausgehenden Kanten von x;
  end if
  $w_l :=$ Startknoten von e_l;
  $w_r :=$ Startknoten von e_r;
  $v := w_l$;
  while $v \neq s$ **do**
    Berechne die letzte eingehende Kante e_{in}
    in der Sequenz der eingehenden Kanten von v;
    Entferne die ausgehenden Kanten zwischen e_{in} und e_l;
    $v :=$ Startknoten von e_{in};
    $e_l := e_{in}$;
  end while
  $v := w_r$;
  while $v \neq s$ **do**
    Berechne die erste eingehende Kante e_{in}
    in der Sequenz der eingehenden Kanten von v;
    Entferne die ausgehenden Kanten zwischen e_r und e_{in};
    $v :=$ Startknoten von e_{in};
    $e_r := e_{in}$;
  end while
  Entferne die Kanten zwischen e_r und e_l in s;
  Traversiere G von s aus und markiere alle erreichten Kanten;
  Füge die entfernten Kanten wieder zu G hinzu;
end procedure

5.4 Einfügen einer Kante in einen eingebetteten sT-Graphen

Wir werden nun mit der Charakterisierung von Lemma 5.2 einen Algorithmus zur Berechnung eines kürzesten Einfügepfades herleiten. Wegen der Eigenschaft 2-3 des aufwärtskonsistenten Einfügepfades (siehe Abschnitt 5.2) können nicht ohne weiteres bekannte Algorithmen zur Berechnung kürzester Pfade wie die von Floyd-Wahrschall oder Dijkstra benutzt werden. Diese setzen voraus, dass ein kürzester Pfad auch kürzeste Pfadabschnitte beinhaltet [10]. Damit ein Algorithmus zur Berechnung eines kürzesten Pfades von x nach y angewendet werden kann, müssen einige Randbedingungen festlegt werden. Eine dieser Randbedingungen wurde schon behandelt, nämlich dass der Pfad nicht unterhalb des Knoten x verlaufen darf. Hierzu wurde der Algorithmus 7 entworfen. Im folgenden Abschnitt werden wir einen Algorithmus angeben, der den Teilgraphen G_o oberhalb von y markiert.

Markieren eines Teilgraphen in Abhängigkeit des Zielknotens y des Einfügepfades

Ein Einfügepfad, der oberhalb des Zielknotens y verläuft, kann nicht aufwärtsplanar sein. Deshalb werden alle Kanten und Knoten markiert, die eindeutig oberhalb von y gezeichnet werden müssen. Anders als im Falle des Teilgraphen G_u, wo stets ein durchgehender Pfad von s nach x existiert, ist beim Teilgraphen G_o oberhalb von y nicht immer garantiert, dass einen Pfad von y zu einer Senke t des äußeren Faces existiert. Gegebenenfalls muss so einen Pfad konstruiert werden, falls er nicht vorhanden ist. Dieser Pfad dient später als blockierender Pfad und gewährleistet, dass kein berechneter Einfügepfad über y verläuft. Wie sich später herausstellen wird, verhindert dieser Pfad auch, dass bei der Berechnung eines Einfügepfades einige Kreuzungen wie in der Abbildung 5.3 nicht mehr vorkommen.

Der Pseudo-Code hierfür ist in Algorithmus 9 angegeben. Dieser markiert einen den Teilgraphen G_o oberhalb von y und konstruiert gegebenenfalls einen Pfad von y zur einer Senke im äußeren Face. Er benötigt eine feste Wahl des äußeren Faces und besitzt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.

Randbedingung für das äußere Face

Seien nun $v_1, \ldots, v_k$ die von Algorithmus markierten Knoten des äußeren Faces. Es ist klar, dass die Kanten eines Einfügepfades nicht über die markierten Knoten $v_1, \ldots, v_k$ verlaufen dürfen. Um diese Bedingung zu erfüllen, wird das äußere Face α_{Γ_G} in zwei Abschnitte $\alpha^l_{\Gamma_G}$ und $\alpha^r_{\Gamma_G}$ eingeteilt. Den ersten Abschnitt erhalten wir, indem wir in α_{Γ_G} im Uhrzeigersinn von s aus zu einem markierten Knoten v_i laufen, den zweiten Abschnitt, wenn wir in α_{Γ_G} von s aus gegen den Uhrzeigersinn bis zur

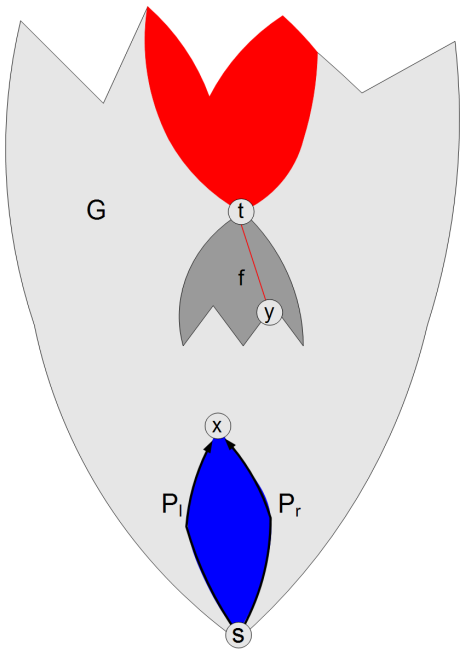


Abbildung 5.6: Illustration zum Algorithmus 9 und 7: Der blaue Teilgraph G_u wird von Algorithmus 7 markiert und symbolisiert alle Knoten und Kanten, die unter dem Knoten x gezeichnet werden müssen. Der rote Teilgraph G_o wird vom Algorithmus 9 markiert. Der Knoten y wird mit dem Knoten t des Faces f verbunden. Dadurch entsteht ein durchgängiger Pfad von y zu einer Senke im äußeren Face. Ein Einfügepfad darf keine Kante im blauen und roten Teilgraphen kreuzen.

Algorithmus 8 Konstruiert eine Kante von y, welche im Face f ist, zu den ausgezeichneten Senken-Switch t von f.

Eingabe: aufwärtskonsistente Zuweisung $\mathcal{A}$, Knoten y und äußeres Face α_{Γ_G}.
Ausgabe: angereicherter Graph G^+
procedure AUGMENTGRAPH($\mathcal{A}$, y, α_{Γ_G})
    if $out(y) \neq 0$ **or** $y \notin \alpha_{\Gamma_G}$ **then**
        Sei $y \in \mathcal{A}(f)$.
        Berechne Senken-Switch $t \in f$, der nicht f zugewiesen wurde.
        $G^+ := G \cup (y, t)$;
        return G^+;
    end if
end procedure

Algorithmus 9 Markiere den Teilgraphen G_o oberhalb von y und konstruiere gegebenenfalls einen Pfad von y bis zu einer Senke im äußeren Face.

Eingabe: aufwärtskonsistente Zuweisung $\mathcal{A}$, Knoten y und äußeres Face α_{Γ_G}.
Ausgabe: markierter und angereicherter Graph G^+
procedure MARK-$G_o(\mathcal{A}, y, \alpha_{\Gamma_G})$
    Traversiere mit einer Breitensuche G von y aus
    und markiere alle erreichten Knoten und Kanten;
    Falls bei der Traversierung eine Senke t erreicht wird,
    dann $G^+ :=$AUGMENTGRAPH$(\mathcal{A}, t, \alpha_{\Gamma_G})$;
    Setze die Traversierung mit dem angereicherten Graphen G^+ fort;
end procedure

einem markierten Knoten v_j laufen. Dabei darf keine Kante im Einfügepfad vom Abschnitt $\alpha^l_{\Gamma_G}$ zum Abschnitt $\alpha^r_{\Gamma_G}$ führen und umgekehrt. Es gilt weiterhin die in Abschnitt 5.2 beschriebene Nummerierung für das äußere Face.

Sperren von Knoten/Kanten

Es muss sichergestellt werden, dass in den Faces, durch die der Einfügepfad P verläuft, die Eigenschaften (a)-(c) des Lemma 5.2 gelten. Es wird nun davon ausgegangen, dass für jede Kante ein entsprechender Dummy-Knoten konstruiert wurde.

Ein Knoten y' ist mit einem Dummy-Knoten x' *benachbart*, wenn x' und y' in dem selben Face f enthalten sind und die Beobachtung 5.2 gilt. x' kann also mit einem zu ihm benachbarten Knoten durch eine gerichtete Kante — mit x' als Startknoten — verbunden werden, ohne die aufwärtsplanare Eigenschaft der daraus resultierenden Einbettung zu zerstören. Mit $adj(f, x')$ werden nun alle benachbarten Dummy-Knoten von x' im Face f und mit $\overline{adj}(f, x')$ alle Dummy-Knoten, die im f liegen, aber nicht mit x' benachbart sind, notiert. $\overline{adj}(f, x')$ enthält folglich die Mengen der Dummy-Knoten, die in Abhängigkeit von x' und f gesperrt werden. Insbesondere gilt $x' \in \overline{adj}(f, x')$.

Kreise entstehen bei der Berechnung des Einfügepfades entweder dann, wenn der Pfad P einen Dummy-Knoten mehr als einmal enthält oder wenn sich zwei Kanten von P kreuzen. Wir werden im folgenden eine Prozedur angeben, die für einen gegebenen Knoten nur solche Knoten zurück gibt, die keine Kreise verursachen. Hierfür wird die Kantenaufzählung eines inneren Faces f betrachtet. Seien x' und y' die Eintritts- und Austrittsknoten des Einfügepfades beim ersten traversieren von f. Sei f' der Face-Abschnitt, der unterhalb der Kante (x', y') liegt (siehe Abbildung 5.7). Falls der Einfügepfad P das Face f noch mal traversiert, dann darf dieser die Dummy-Knoten von f' sowie die Dummy-Knoten von $\overline{adj}(f, x') \cup \overline{adj}(f, y')$ nicht benutzen. Es kann nun beobachtet werden, dass nicht mit x' benachbarte Dummy-

Knoten auch nicht mit y' benachbart sind, wenn sie nicht in f' liegen. Ferner lässt sich feststellen, dass die Knoten von f' alle in der Menge $\overline{adj}(f, x') \cup \overline{adj}(f, y')$ liegen. Verallgemeinern wir diese Feststellung, so gilt: Sind $(x'_1, y'_1), \ldots, (x'_k, y'_k)$ die Kanten eines Einfügepfades, die durch f verlaufen, so sind alle Knoten der Menge $\overline{adj}(f, x'_1, y'_1, \ldots, x'_k, y'_k) := \overline{adj}(f, x'_1) \cup \cdots \cup \overline{adj}(f, x'_k) \cup \overline{adj}(f, y'_1) \cup \cdots \cup \overline{adj}(f, y'_k)$ gesperrt. Wenn zwei Kanten $e_1 = (x'_1, y'_1)$ und $e_2 = (x'_2, y'_2)$ sich in f kreuzen, dann muss einer der Start- oder Endknoten dieser beiden Kanten in der Menge der gesperrten Knoten $\overline{adj}(f, x'_1, y'_1, x'_2, y'_2)$ sein.

Beobachtung 5.3. *Sei (x', y') eine Kante, die durch das Face f verläuft. Dann gilt:*

(a) Alle Knoten in f' sind in $\overline{adj}(f, x', y')$.

(b) Für einen Knoten $d \in f$ mit $d \notin f'$ gilt: $d \in adj(f, x') \Leftrightarrow d \in adj(f, y')$.

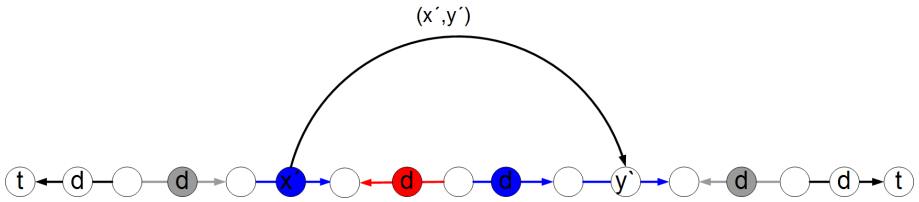


Abbildung 5.7: Zur besseren Darstellung werden die Kanten von f in einer Geraden gezeichnet. Die Kanten/Dummy-Knoten, die wegen Knoten x' gesperrt werden, sind in rot und grau gezeichnet. Die Kanten/Dummy-Knoten, die wegen Knoten y' gesperrt werden, sind in blau und grau gezeichnet. f' ist die Kantensequenz zwischen x' und y'.

Berechnung der gesperrten und nicht gesperrten Dummy-Knoten

Wir geben nun in Algorithmus 10 eine Prozedur an, die in Abhängigkeit eines Knoten v alle benachbarten, nicht gesperrten Dummy-Knoten in einem Face f berechnet. Zusätzlich zum Knoten v und Face f benötigt die Prozedur noch eine aufwärtskonsistente Zuweisung $\mathcal{A}$. Diese Zuweisung ist nötig, um die ausgezeichneten Senken der inneren Faces zu identifizieren und die Nummerierung der Dummy-Knoten in f zu berechnen (Zeile 4). In Zeile 5 wird der Pfad P bis zum Knoten v berechnet. Es wird davon ausgegangen, dass jeder Knoten den Eintrag $pred(v)$ besitzt, der den Vorgänger des Knotens v in P ausweist. Somit kann P und die Knoten $d_1, \ldots, d_k$ berechnet werden (Zeile 5-6). Diese Knoten sind die Schnittknoten des Pfades P mit f. Mit diesem Knoten wird die Menge $\overline{adj}(f, v, d_1, \ldots, d_k)$ berechnet und somit auch alle Dummy-Knoten, die benachbart zu v und nicht gesperrt sind (Zeile 7). Für die Berechnung von $adj(f, v)$ in Zeile 7 muss unterschieden werden, ob f ein inneres oder äußeres Face ist. Es gilt:

- f ist inneres Face:
  Ein Knoten $y' \in f$, der die Beobachtung 5.2 erfüllt, ist in $adj(f, v)$.

- f ist äußeres Face:
  Identifiziere die Face-Abschnitte $\alpha^l_{\Gamma_G}$ und $\alpha^r_{\Gamma_G}$. Falls $v \in \alpha^l_{\Gamma_G}$ bzw. $v \in \alpha^r_{\Gamma_G}$ ist, dann sind alle Knoten in $\alpha^l_{\Gamma_G}$ bzw. $\alpha^r_{\Gamma_G}$, welche die Beobachtung 5.2 erfüllen, in $adj(f, v)$.

Trivialerweise gilt für beide Fälle $\overline{adj}(f, x') = \{v \mid v \in f \text{ und } v \notin adj(f, x')\}$.

Lemma 5.3. *Seien $P := x \rightsquigarrow v$ ein aufwärtskonsistenter Pfad, M die Menge der Knoten, die die Prozedur* AdjDummies$(\mathcal{A}, v, f)$ *zurückgibt und $y' \in M$. Dann ist der Pfad $P^* := x \rightsquigarrow v + (v, y')$ aufwärtskonsistent.*

Beweis. Wir zeigen (a) P^* ist azyklisch, (b) für die Kanten von P^*, die durch f verlaufen, gilt die Beobachtung 5.2 und (c) falls v kein Eintrittsknoten in f ist, dann gilt für die Kanten (v', y') die Beobachtung 5.2. Dabei ist v' der letzte Eintrittsknoten in f. Nach Lemma 5.2 folgt dann die Behauptung.

(a) Angenommen P^* ist nicht azyklisch. Dann muss die Kante (v, y') mindestens eine Kante $e = (d_1, d_2)$ kreuzen oder y' ist ein Knoten von P.
Sei f' der Face-Abschnitt von f unterhalb der Kante e. Damit (v, y') die Kante e kreuzt, muss entweder v oder y' in f' sein. Nach Beobachtung 5.3 sind jedoch alle Knoten von f' in $\overline{adj}(f, d_1, d_2)$. Die Anweisung in Zeile 8 gibt keinen Knoten aus $\overline{adj}(f, d_1, d_2)$ zurück. Widerspruch!
Wenn y' ein Knoten in P ist, dann ist nach der Anweisung in Zeile 6 und 7 der Knoten y' in $\overline{adj}(f, y')$. Widerspruch!

(b) Nach Voraussetzung gilt für alle Kanten von P die Bedingung 5.2. Was noch gezeigt werden muss, ist dass es auch für die Kante (v, y') gilt. Nach Zeile 7 ist y' in der Menge $adj(f, v)$. $adj(f, v)$ ist die Menge der zu y' benachbarten Dummy-Knoten im Face f für die die Beobachtung 5.2 gilt. Also gilt (b).

(c) Angenommen die Kante (v', y') erfüllt die Beobachtung 5.2 nicht, dann ist y' in $\overline{adj}(f, v')$. Gemäß der Anweisung in Zeile 8 wird y' nicht von der Prozedur zurückgegeben. Widerspruch!

□

Wahl des äußeren Faces

Die Kosten eines Einfügepfades für eine gegebene aufwärtsplanare Einbettung können von der Wahl des äußeren Faces abhängen (siehe Abbildung 5.8). Diese Tatsache erschwert die Berechnung eines kostenminimalen Einfügepfades. Wenn $\alpha^1_{\Gamma_G}, \ldots, \alpha^k_{\Gamma_G}$

Algorithmus 10 Berechne eine Liste nicht gesperrter Dummy-Knoten im Face f in Abhängigkeit vom Knoten v.

Eingabe: aufwärtskonsistente Zuweisung $\mathcal{A}$, Knoten v und Face f.
Ausgabe: Eine Liste von Dummy-Knoten, die alle Zielknoten einer Kante $(v, \cdot)$ sein können, so dass für diese Kante in f die Beobachtung 5.2 gilt und kein anderer Kante von ihnen gekreuzt wird.
procedure AdjDummies($\mathcal{A}$, v, f)
    Berechne mit $\mathcal{A}$ die Nummerierung $num(f, v)$;
    Berechne den Pfad P bis zum Knoten v;
    Berechne $d_1, \ldots, d_k$ mit $d_i \in f$ und $d_i \in P$;
    Berechne mit $num(f, v)$ die Mengen $adj(f, v)$ und $\overline{adj}(f, v, d_1 \ldots, d_k)$;
    return $adj(f, v) - \overline{adj}(f, v, d_1 \ldots, d_k)$;
end procedure

die möglichen äußeren Faces von Γ_G sind, so müssen wir für jede Wahl des äußeren Faces $\alpha^i_{\Gamma_G}$ einen kostenminimalen Einfügepfad berechnen und dann aus den Lösungen das Optimum wählen. Die Berechnung der möglichen äußeren Faces kann mit Hilfe von Satz 5.1 realisiert werden. Hierfür muss ein Face-Senken-Graph F von Γ_G konstruiert und der Baum T ermittelt werden. Alle Faces, deren Face-Knoten in T liegen und die Quelle s von G enthalten, sind mögliche äußere Faces der Einbettung Γ_G. Die Berechnung benötigt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.

Der Algorithmus

Der Algorithmus zur Berechnung eines kürzesten, aufwärtskonsistenten Einfügpfades, den wir im Folgenden vorstellen, funktioniert wie folgt:

Zuerst werden die möglichen äußeren Faces $\alpha^1_{\Gamma_G}, \ldots, \alpha^k_{\Gamma_G}$ berechnet. Dann wird für jedes $\alpha^i_{\Gamma_G}$ mit Algorithmus 6 getestet, ob die Einbettung Γ_G mit $\alpha^i_{\Gamma_G}$ als äußeres Face mindestens einen aufwärtskonsistenten Einfügepfad besitzt. Wenn nicht, so wird das nächste äußere Face gewählt. Ansonsten werden die Teilgraphen G_o und G_u mit dem Algorithmen 9 und 7 markiert. Für jede nicht markierte Kante wird ein Dummy-Knoten konstruiert. Anschließend wird Dijkstras Algorithmus zur Berechnung kürzester Pfade angewendet, um einen kürzesten Pfad von x nach y zu berechnen. Dabei führt der berechnete Pfad nur über die Dummy-Knoten, die von der Prozedur AdjDummies berechnet werden. Der so berechnete Pfad wird gespeichert. Diese Prozedur wird dann für alle $\alpha^1_{\Gamma_G}, \ldots, \alpha^k_{\Gamma_G}$ wiederholt. Zum Schluss wird aus den möglichen Lösungen die Kostenminimale ausgegeben.

Algorithmus 11 Initialisierung

```
    Eingabe: Graph G und Knoten x.
 2: procedure Init(G, x)
        Konstruiere für jede nicht markierte Kante einen Dummy-Knoten;
 4:     Markiere y als Dummy-Knoten;
        Lösungen L := ∅;
 6:     for each Dummy d do
            cost(d) := ∞;
 8:         pred(d) :=nil; //Vorgänger von d
            predFace(d) :=nil; //zuletzt besuchtes Face
10:    end for
       cost(x) := 0;
12:    Berechne die zu x adjazenten Faces f_1, ..., f_k;
       for each Faces f_i ∈ {f_1, ..., f_k} do
14:        for each Dummy d ∈ adj(f_i, x) do
               if cost(d) ≠ 1 then
16:                cost(d) := 1;
                   predFace(d) := f_i;
18:            end if
           end for
20:    end for
   end procedure
```

Algorithmus 12 Berechnet einen aufwärtskonsistenten Einfügepfad von Knoten x nach y.

Eingabe: aufwärtsplanare Einbettung Γ_G, Knoten x und y.
Ausgabe: aufwärtskonsistenter Pfad P oder $\emptyset$, falls kein solcher Pfad existiert.
procedure InsertEdgeEmbedded(Γ_G,x, y)
  if HasConsistentInsertionPath(Γ_G, x, y) = **false then**
    return $\emptyset$;
  end if
  Berechne die äußeren Faces $\alpha^1_{\Gamma_G}, \ldots, \alpha^k_{\Gamma_G}$;
  for $\alpha^i_{\Gamma_G} \in \{\alpha^1_{\Gamma_G}, \ldots, \alpha^k_{\Gamma_G}\}$ **do**
    Berechne eine aufwärtskonsistente Zuweisung $\mathcal{A}$;
    Mark-$G_u(\Gamma_G, \mathcal{A}, x)$;
    if y ist nicht markiert **then**
      Mark-$G_o(\mathcal{A}, y, \alpha^i_{\Gamma_G})$;
      Init(G, x);
      Initialisiere Min. Priority-Queue Q mit allen Dummy-Knoten;
      while $Q \neq \emptyset$ **do**
        $d :=$**pop**(Q);
        Berechne Face f adjazent zu d mit $f \neq predFace(d)$;
        Menge $M :=$AdjDummies($\mathcal{A}, d, f$);
        for each $v \in M$ **do**
          if $cost(v) > cost(d) + 1$ **and** $f \neq predFace(d)$ **then**
            $pred(v) := d$;
            $predFace(v) := f$;
          end if
        end for
      end while
      Konstruiere Pfad P_i mit Hilfe von $pred(y)$;
      $\mathcal{L} := \mathcal{L} \cup P_i$;
    end if
    Lösche alle Markierungen und hinzugefügten Kanten;
  end for
  if $\mathcal{L} = \emptyset$ **then**
    return $\emptyset$;
  end if
  return P mit $\min_{P \in \mathcal{L}}\{cost(P)\}$;
end procedure

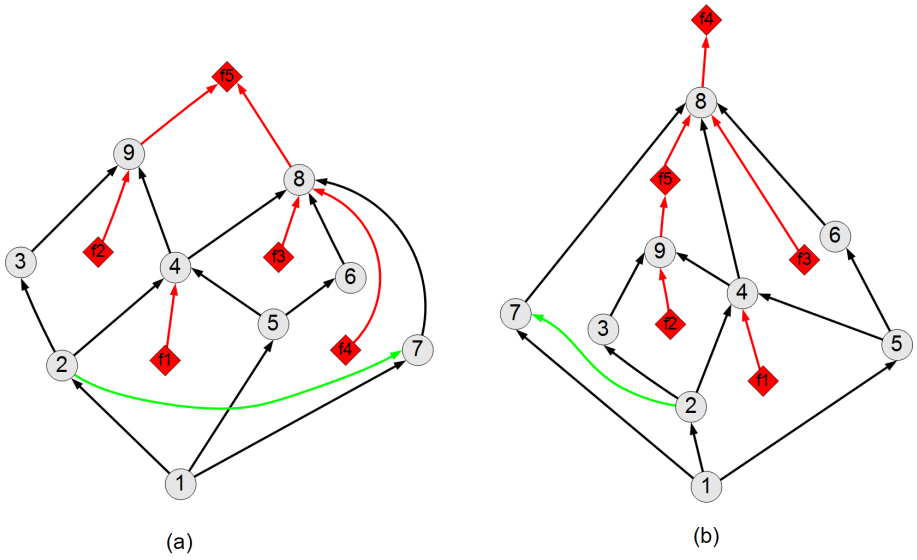


Abbildung 5.8: Die Kreuzungsanzahl eines Einfügepfades kann von der Wahl des äußeren Faces abhängen. (a) f_5 ist äußeres Face; Es wird eine Kante vom grünen Einfügepfad gekreuzt. (b) f_4 ist äußeres Face; Es wird keine Kante vom grünen Einfügepfad gekreuzt.

Korrektheit und Laufzeit

Satz 5.2. *Algorithmus 12 berechnet einen kreuzungsminimalen Einfügepfad für einen eingebetteten sT-Graphen $G = (V, A)$ mit k möglichen äußeren Faces in Zeit $\mathcal{O}(k(|A||V| + |A|^2))$.*

Beweis. Wir müssen folgendes zeigen:

(a) Der berechnete Pfad P_A ist aufwärtskonsistent.

(b) Der Algorithmus berechnet einen Pfad P_A von x nach y falls er existiert.

(c) Der berechnete Pfad ist kostenminimal.

Beweis zu (a): In der Prozedur *Init* wird in den Zeilen 12-18 alle zu x benachbarte Dummy-Knoten berechnet. Sei d_1 ein beliebiger zu x benachbarter Dummy-Knoten. Da $d_1 \in adj(f_i, x)$ ist, gilt für die Kante (x, d_1) die Beobachtung 5.2. Der Pfad $P_A = (x, d_1)$ ist offensichtlich aufwärtskonsistent. Danach findet die weitere Berechnung von P_A in der **While**-Schleife in den Zeilen 15-25 der Hauptprozedur InsertEdgeEmbedded statt. Die weiteren Knoten von P_A bestehen dann aus den von der Prozedur AdjDummies zurückgegebenen Dummy-Knoten (siehe Zeile 15 und 25). Mit Lemma 5.3 kann nun induktiv über die Anzahl k der Kanten des Pfades P_A argumentiert werden.

Induktionsbeginn ***While****-Schleife*:
Sei $P_A = (x, d_1)$ der bis dahin berechnete Pfad. d_1 ist somit in der Queue Q mit $cost(d_1) = 1$. Wenn die **While**-Schleife in den Zeilen 15-25 zum ersten mal betreten wird, dann gibt die Prozedur AdjDummies eine Menge M von Knoten zurück, die zu d_1 benachbart sind. Der Pfad P_A hat nach der Initialisierung genau eine Kante. Sei $d_2 \in M$. Gemäß Lemma 5.3 ist dann der Pfad $P_A + (d_1, d_2)$ aufwärtskonsistent.
Induktionsschritt ***While****-Schleife*:
Wird die Schleife zum n-ten (und letzten) mal durchlaufen, dann ist nach Induktionsbehauptung $P_A^n = (x, d_1), \ldots, (d_{n-1}, d_n)$ ein aufwärtskonsistenter Pfad. Sei $d_{n+1} \in M$. Dann ist der Pfad $P_A^{n+1} = P_A^n + (d_n, d_{n+1})$ gemäß Lemma 5.3 aufwärtsplanar.

Beweis zu (b): Sei P ein aufwärtskonsistenter Pfad von x nach y. Wenn der Algorithmus keinen Pfad von x nach y findet, dann gibt es einen Knoten w mit $P_1 := x \rightsquigarrow w'$, $P_2 := w \rightsquigarrow y$ und $P = P_1 + (w', w) + P_2$, so dass entweder w nach Berechnung von P_1 von der Prozedur AdjDummies gesperrt wird und P_2 nicht mehr berechnet werden kann oder es gibt einen Pfad $P_1' := x \rightsquigarrow w$ mit $cost(P_1') < cost(P_1) + 1$, so dass $P_1' + P_2$ nicht aufwärtskonsistent ist. Letzteres ist in Abbildung 5.3 illustriert. Dort besucht der Algorithmus zuerst den Knoten d_1 und d_1 bekommt die Kosten $cost(d_1) = 1$. Wird später der grüne (richtige) Pfad verfolgt und Knoten d_3 vom Algorithmus besucht, so ist $cost(d_3) = 3$. Der Knoten d_1 ist benachbart zu Knoten d_3. Es gilt aber $cost(d_3) + 1 > cost(d_1)$. Die Kante (d_3, d_1) wird also nicht verfolgt und der richtige Pfad nicht berechnet. Der Knoten d_1 blockiert durch seine niedrigen Kosten eventuell einen (kreuzungsminimalen) aufwärtskonsitenten Pfad von x nach y. Im Folgenden wird gezeigt, dass wenn ein blockierter Knoten existiert, dann gehört dieser Knoten nicht zu einem aufwärtskonsistenten Einfügepfad oder es gibt stets einen anderen (kürzeren) Pfad ohne diesen Knoten. Der Algorithmus wird dann diesen berechnen.

Fall 1: w wird von der Prozedur AdjDummies *gesperrt*
Wenn w von der Prozedur AdjDummies gesperrt wird, dann können wir aus Lemma 5.3 folgern, dass mindestens eine der Eigenschaften (a)-(c) von Lemma 5.2 nicht erfüllt ist.

- Angenommen die Kante (w', w) kreuzt eine Kante $e \in P_1$. Da $P = P_1 + (w', w) + P_2$ aufwärtskonsistent ist, ist er nach Lemma 5.2 insbesondere azyklisch. (w', w) kann folglich keine Kanten kreuzen. Widerspruch!
- Angenommen die Kante (w', w) erfüllt Beobachtung 5.2 nicht. Alle Einfügepfade, die diese Kante enthalten sind nicht aufwärtskonsistent. Da P aufwärtskonsistent ist, erfüllen alle Kanten von P nach Lemma 5.2

die Beobachtung 5.2. Insbesondere gilt das für die Kante (w', w). Widerspruch!

- Sei f das Face, das w' und w enthält und durch das der Pfad P verläuft, und w' ist kein Eintrittsknoten in f. Sei w'' der letzte Eintrittsknoten von P in f.
  Angenommen die Kante (w'', w) erfüllt nicht die Beobachtung 5.2. Dann ist nach Lemma 5.2 P nicht aufwärtskonsistent. Widerspruch zur Voraussetzung!

Folglich können die von der Prozedur AdjDummies gesperrten Knoten nicht zu dem aufwärtskonsistenten Pfad P gehören. (Hier muss man beachten, dass die Menge der gesperrten Knoten abhängig vom gewählten Pfad zum Knoten w ist. Die gesperrten Dummy-Knoten gelten nur für den gerade vom Algorithmus betrachteten Einfügepfad P_A.)

Fall 2: $\exists P_1' = x \rightsquigarrow w : cost(P_1') < cost(P_1) + 1$
Sei (x', w) die letzte Kante des Pfades P_1'. Sei f das Face, das x' und w enthält, und durch das der Pfad P verläuft. Der Pfad P kann f mehrmals traversieren. Sei y' deshalb der letzte Austrittsknoten von P aus dem Face f, so dass der Pfadabschnitt $y' \rightsquigarrow y$ das Face f nicht mehr betritt. Sei f' der Face-Abschnitt, der unterhalb von der Kante (x', w) liegt (siehe Abbildung 5.9). Wir unterscheiden nun die folgende vier Fälle:

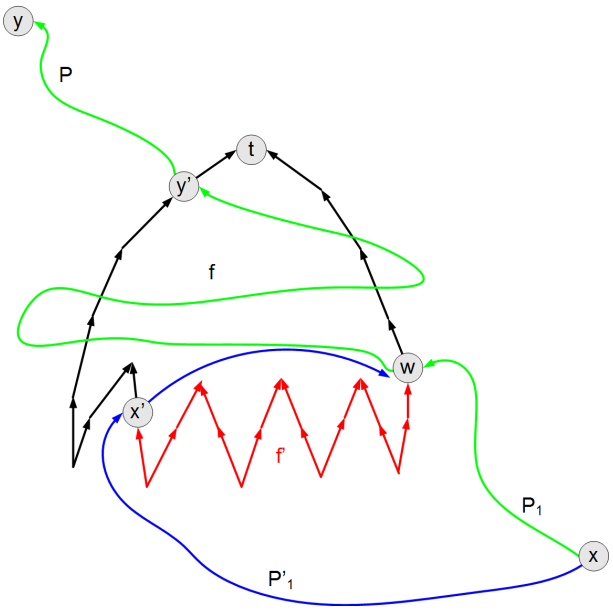


Abbildung 5.9: Illustration zum Beweis 5.2 Fall 2.

(i): $y' \notin f'$ und $y' \in adj(f, x')$
Bei der Untersuchung des Knotens x' wird der Algorithmus alle Knoten aus

$adj(f, x')$ untersuchen (Zeile 19-24). Da $y' \in adj(f, x')$ ist, wird folglich auch die Kante (x', y') betrachtet. Diese Kante führt zum Pfad $x \rightsquigarrow x' + (x', y') + y' \rightsquigarrow y$. Es muss nun gezeigt werden, dass die Konkatenation der beiden Pfade aufwärtskonsistent ist. Da $y' \rightsquigarrow y$ ein Teilpfad von P ist, ist er sicherlich aufwärtskonsistent. Der Pfad $x \rightsquigarrow x' + (x', y')$ wurde vom Algorithmus berechnet, folglich ist er nach (a) aufwärtskonsistent. Da $y' \in adj(f, x')$ ist, erfüllt die Kante (x', y') die Beobachtung 5.2. Erfüllt ist auch die Eigenschaft (c) des Lemma 5.2, da zwischen x' und y' keine weiteren Dummy-Knoten sind. Es muss noch gezeigt werden, dass in f keine Kreuzungen zwischen der Kante (x', y') und dem Pfad $y' \rightsquigarrow y$ existieren, da $y' \rightsquigarrow y$ das Face f mehrfach traversieren kann. Das ist sicherlich der Fall, da y' der letzte Austrittsknoten von P in f ist und nach y' der Pfad P nach Annahme Face f nicht mehr betritt. Der Algorithmus berechnet folglich eine Pfad der kürzer ist als P.

(ii): $y' \notin f'$ und $y' \notin adj(f, x')$
Da P den Pfadabschnitt $w \rightsquigarrow y'$ enthält, ist $y' \in adj(f, w)$, sonst ist P nicht aufwärtskonsistent. Wegen dem Pfad P'_1, der vom Algorithmus berechnet wurde, gilt $w \in adj(f, x')$. Nach Beobachtung 5.3 sind alle zu x' benachbarten Dummy-Knoten auch zu w benachbart und umgekehrt. Also ist $y' \in adj(f, x')$. Widerspruch zur Voraussetzung! Dieser Fall kann nicht vorkommen.

(iii): $y' \in f'$ und $y' \in adj(f, x')$
Bei der Untersuchung des Knotens x' wird der Algorithmus alle Knoten aus $M = \text{AdjDummies}(\mathcal{A}, x', f)$ untersuchen (Zeile 19-24). Folglich wird auch die Kante (x', y') untersucht. Der Algorithmus berechnet dann einen Pfad $x \rightsquigarrow x' + (x', y') + y' \rightsquigarrow y$ ohne den Knoten w. Die Begründung warum die Konkatenation der Teilpfade zu einem aufwärtplanaren Pfad führt ist wie im Fall *(i)*.

(iv): $y' \in f'$ und $y' \notin adj(f, x')$
Sei G' der maximale Teilgraph, der von $P_l := s \rightsquigarrow x + P'_1$, $P_r := s \rightsquigarrow w$ und den Kanten von f' eingeschlossen wird (siehe Abbildung 5.10).

- $y \in G'$:
  Es existiert ein von der Prozedur Mark-G_o konstruierter Pfad P_B, der von y zu einer Senke t im äußeren Face führt. Dieser Pfad ist markiert und enthält keinen Dummy-Knoten (siehe Prozedur Init Zeile 3). Folglich kann P_B von keinem vom Algorithmus berechneten Pfad gekreuzt werden. Da y von P_l und P_r eingeschlossen wird, schneidet P_B entweder P_l oder P_r. Wenn P_B den Pfad P_l schneidet, dann kann es den Pfad P'_1 nicht geben. Widerspruch!
  Wenn P_B den Pfad P_r schneidet, dann wird der Knoten w von der Prozedur Mark-G_o markiert. w muss somit oberhalb von y gezeichnet werden. Der Pfad P, der w enthält, kann kein aufwärtskonsistenter Pfad sein. Wi-

derspruch!

- $y \notin G'$:
  Da y' der letzte Austrittsknoten von P aus f ist, muss der Pfad P durch G' verlaufen. Er kann dabei G' verlassen, indem er entweder den Pfad P_l oder den Pfad P_r kreuzt. (Da y nicht in G' ist, muss P den Teilgraphen G' verlassen.) Wenn P, z.B. an einem Knoten z, P_r kreuzt, so entsteht der Kreis $z \rightsquigarrow w \rightsquigarrow y' \rightsquigarrow w$ und P wäre nicht aufwärtskonsistent. Widerspruch!
  Folglich kann P nur den Pfad P_l kreuzen. Die Kreuzung muss in einem Face f^* stattfinden. Entweder findet die Kreuzung an einem Dummy-Knoten $d \in f^*$ statt oder die Kreuzung entsteht durch zwei gekreuzte Kanten in f^*. Der Beweisweg für beide Fälle ist fast identisch. Es wird deshalb nur für den letzten Fall bewiesen.
  Seien a der Eintrittsknoten und b der letzte Austrittsknoten des Pfades P und a' der erste Eintrittsknoten und b' der letzte Austrittsknoten des Pfades P'_1 in f^*. Falls $b \in adj(f^*, a')$ ist, so wird der Algorithmus den Pfad $s \rightsquigarrow a' + (a', b) + b \rightsquigarrow y$ berechnen. Dieser Pfad enthält nicht den Knoten w (siehe Abbildung 5.11 (a)). Die Konkatenation der Pfadabschnitte ist aufwärtskonsistent (Argumentation wie im Falle (i)). Falls $b \notin adj(f^*, a')$ ist, dann schneiden sich die Pfade P und P'_1 zwar in f^*, sie kreuzen sich jedoch noch einmal (siehe Abbildung 5.11 (b)). Sei dann f^{**} das Face, wo sich die beiden Pfade zum letztenmal kreuzen. Eine einfache Untersuchung zeigt, dass wenn sich die beiden Pfade in f^{**} zum letztenmal kreuzen, dann stets eine Kante vom ersten Eintrittsknoten c von P'_1 zum letzten Austrittsknoten d von P existiert, so dass die Konkatenation des Pfadabschnittes $x \rightsquigarrow c$ von P'_1 mit der Kante (c, d) und mit dem Pfadanschnitt $d \rightsquigarrow y$ zu $x \rightsquigarrow c + (c, d) + d \rightsquigarrow y$ aufwärtskonsistentent ist. Dieser Pfad wird vom Algortihmus auf Grund seiner Greedy Eigenschaft berechnet. Er ist kürzer als P und enthält w nicht.

Beweis zu (c): Da die Zeilen 15-25 zusammen mit der Initialisierung INIT Dijkstras Algorithmus zur Berechnung eines kürzesten Pfades realisieren, ist der berechnete Pfad ein kürzester Pfad. Es muss noch gezeigt werden, dass, wenn ein kürzester Pfad P_{min} existiert, kein Knoten von P_{min} vom Algorithmus gesperrt wird. Somit wird er auch vom Algorithmus berechnet. Angenommen v ist ein gesperrter Knoten von P_{min}. Aus dem Beweis von (a) und (b) kann geschlossen werden, dass die Sperrung nicht von der Prozedur ADJDUMMIES durchgeführt wird. Die Prozeduren MARK-G_u und MARK-G_o markieren nur Kanten die eindeutig unterhalb des Knoten x und oberhalb des Knoten y verlaufen. Für diese Kanten werden keine Dummy-Knoten konstruiert, was offensichtlich korrekt ist. Folglich ist die Annahme, dass ein Knoten v existiert, der vom Algorithmus

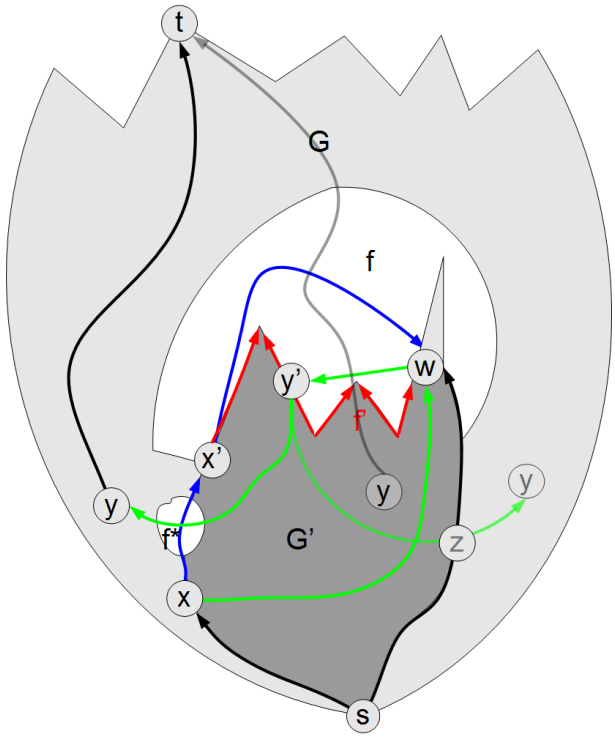


Abbildung 5.10: Illustration zum Beweis 5.2 Fall 2 *(iv)*: In der Abbildung sind die verschiedenen Positionen von y dargestellt. Falls $y \in G'$ ist, dann kreuzt der Pfad $P_B := y \rightsquigarrow t$ die Kante (x', w). Falls $y' \notin G'$ ist, dann kreuzt P_A den Pfad $s \rightsquigarrow w$ und es entsteht ein Kreis oder P_B kreuzt den Pfad $x \rightsquigarrow w$ in einem Face f^*.

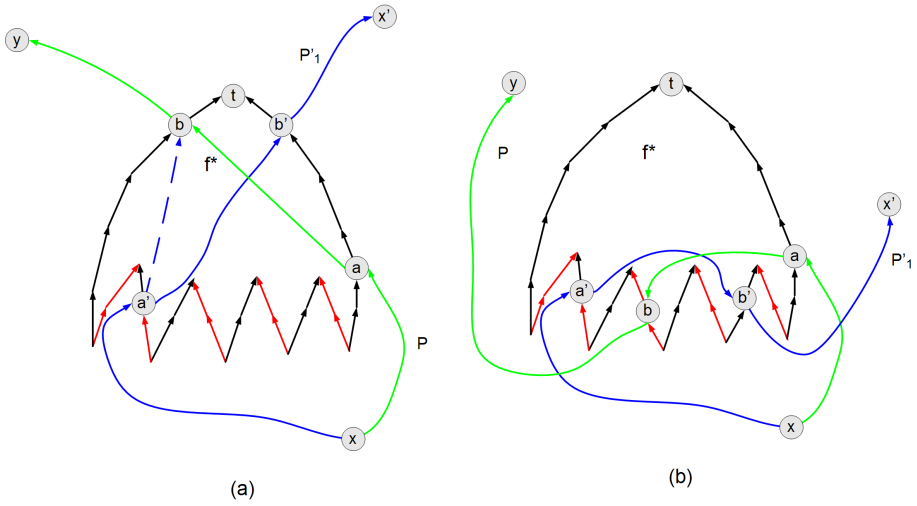


Abbildung 5.11: Illustration zum Beweis 5.2 Fall 2 *(iv)*: Der Pfad P kreuzt den Pfad P'_1 im Face f^*. Die roten Kanten sind Kanten aus $\overline{adj}(f^*, a')$. (a) Falls $a \in adj(f^*, a')$ ist, dann gibt es eine Kante von a' nach b. (b) Wenn $a \notin adj(f^*, a')$ ist, dann müssen sich die beiden Pfade noch einmal kreuzen.

gesperrt wird, falsch.

Aus (a), (b) und (c) folgt die Korrektheit. Wir betrachten nun die Laufzeit für eine Iteration der **for**-Schleife (Zeile 8-30) im Algorithmus.

1. Prozedur HasConsistentInsertionPath:
   Der Test, ob ein Pfad von x nach y existiert benötigt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.
   Die Konstruktion des Face-Senken-Graph F benötigt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.
   Konstruktion von $G^+ := G \cup F^* \cup (x, y)$ benötigt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.
   Test auf Kreisfreiheit von G^+ benötigt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.
   Gesamtlaufzeit: $\mathcal{O}(|V| + |A|)$.

2. Berechnung einer aufwärtskonsistenten Zuweisung $\mathcal{A}$ benötigt nach Lemma 5.1 eine Laufzeit von $\mathcal{O}(|V| + |A|)$.

3. Berechnung der möglichen äußeren Faces benötigt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.

4. Prozedur Mark-G_u benötigt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.

5. Prozedur MARK-G_o benötigte eine Laufzeit von $\mathcal{O}(|V| + |A|)$.

6. Prozedur INIT benötigt für die Konstruktion und Initialisierung der Dummy-Knoten eine Laufzeit von $\mathcal{O}(|A|)$.

7. Initialisierung des Priority-Queue Q (Fibonacci Heap) benötigt eine Laufzeit von $\mathcal{O}(|V| \cdot \log |V| + |A|)$ [10].

8. **While**-Schleife in Zeilen 15-25:
   Die **While**-Schleife wird $|A|$ mal durchlaufen.

   - Die Berechnung der Faces adjazent zu d in Zeile 17 benötigt im Worse-Case $\mathcal{O}(|A|)$.
   - Die Prozedur ADJDUMMIES benötigt im Worse Case für die Nummerierung eines Faces f eine Laufzeit $\mathcal{O}(|V|+|A|)$. Die Berechnung des Pfades bis zum Knoten v hat eine Laufzeit von $\mathcal{O}(|A|)$. Die Berechnung der nicht gesperrten Dummy-Knoten benötigt eine Laufzeit von $\mathcal{O}(|V| + |A|)$.
   - Für die Zeilen 19-24 (Relaxation) wird eine Zeit von $\mathcal{O}(|A|)$ benötigt.

   Somit beträgt die Gesamtlaufzeit der **While**-Schleife $\mathcal{O}(|A||V| + |A|^2)$.

9. Das Löschen der Markierungen und der hinzugefügten Kanten sowie die Ausgabe der Lösung benötigt zusammen eine Laufzeit von $\mathcal{O}(|V| + |A|)$.

Besitzt G k mögliche äußere Faces, so beträgt die Laufzeit $\mathcal{O}(k(|A||V| + |A|^2))$. □

Im Worse-Case ist k in der Größenordnung $\mathcal{O}(|A|)$. Somit ist die Laufzeit $\mathcal{O}((|A|^2|V|+|A|^3))$ und die Laufzeit für sT-Graphen ohne Multikanten ist $\mathcal{O}(|V|^3)$. Wir können jedoch davon ausgehen, dass in der Praxis aufwärtsplanare Einbettungen mit $\mathcal{O}(|A|)$ äußeren Faces selten vorkommen.

6 Berechnung eines Einfügepfades in einem 2-zusammenhängenden sT-Graphen

Der Algorithmus 12 InsertEdgeEmbedded aus Kapitel 5 berechnet für eine gegebene aufwärtsplanare Einbettung einen Einfügepfad mit minimaler Kreuzungsanzahl. Aus den Untersuchungen von Kapitel 3 wissen wir, dass die Spiegelung der Graphkomponenten, durch die der Einfügepfad verläuft, die Kreuzungsanzahl des Pfades beeinflussen kann. Diese Tatsache werden wir ausnutzen, um für einen 2-zusammenhängenden, aufwärtsplanaren sT-Graphen G einen gegenüber Algorithmus InsertEdgeEmbedded verbesserten Ansatz zu entwickeln. Der im folgenden beschriebene Algorithmus berechnet einen Einfügepfad von Knoten x nach y, wenn er eine Einbettung mit mindestens einem aufwärtskonsistenten Einfügepfad findet. Der Algorithmus setzt sich aus folgenden Phasen zusammen:

1. Berechnung einer zufälligen, aufwärtsplanaren Einbettung Γ_G von G:
   Hierfür werden die Ergebnisse aus Kapitel 4 benutzt. Zuerst wird ein Expansionsgraph G_{exp} von G und der SPQR-Baum $\mathcal{T}_{G_{exp}}$ von G_{exp} sowie die entsprechenden sT-Skeletons berechnet. Eine aufwärtsplanare Einbettung für G_{exp} erhält man, indem man z.B. zufällig ein R-Knoten oder P-Knoten von $\mathcal{T}_{G_{exp}}$ mit gleicher Wahrscheinlichkeit wählt. Wenn ein R-Knoten gewählt wird, dann wird die Einbettung des R-Skeletons gespiegelt. Wenn ein P-Knoten gewählt wird, dann werden zwei virtuelle Kanten unter Berücksichtigung von Lemma 4.1 zufällig (gleichverteilt) gewählt[1]. Die beiden Kanten vertauschen dann ihre Position in dem sT-Skeleton. $\mathcal{T}_{G_{exp}}$ legt dann eine aufwärtsplanare Einbettung von G_{exp} fest. Aus dieser Einbettung wird Γ_G konstruiert, indem die Expansion von G rückgängig gemacht wird.

2. Berechnung eines aufwärtskonsistenten Pfades P von G:
   Berechne mit InsertEdgeEmbedded(Γ_G, x, y) einen aufwärtskonsistenten Einfügepfad P. Falls die Einbettung keinen aufwärtskonsistenten Einfügepfad besitzt, dann wird die Berechnung abgebrochen.

[1] Das bedeutet, dass nur die Kanten gewählt werden, so dass durch deren Vertauschung die Einbettung des sT-Skeletons aufwärtsplanar bleibt. Lemma 4.1 gibt Auskunft welche Kantenkombinationen nicht gewählt werden dürfen.

3. Berechnung des Teilbaumes $\mathcal{T}_\mu$:
   Berechne einen Knoten μ von $\mathcal{T}$, so dass in dessen pertinenten Graphen der ganze Pfad P lokalisiert ist. Berechne dann den Teilbaum $\mathcal{T}_\mu$ von $\mathcal{T}$, der μ als Wurzel hat. Wenn der Pfad P die Komponenten $K_1, \ldots, K_k$ traversiert, dann sind alle diese Komponenten im pertinenten Graphen von μ enthalten.

4. Klassifizierung der Graphkomponenten nach den Ersetzungsregeln $\mathcal{R}_1$ bis $\mathcal{R}_4$:
   $\mathcal{T}_\mu$ wird von μ aus mit einer Breitensuche traversiert. Dabei werden die Knoten $\nu_1, \ldots, \nu_k$[2] sowie die zu ihnen korrespondierenden pertinenten Graphen $G_1, \ldots, G_k$, durch die der Pfad P verläuft, berechnet. Die pertinenten Graphen werden dann nach den Ersetzungsregeln $\mathcal{R}_1$ bis $\mathcal{R}_4$ klassifiziert.

5. Spiegelung der Graphkomponenten:
   Durch Γ_G wird auf jeden pertinenten Graphen $G_1, \ldots, G_k$ die Einbettung $\Gamma_1, \ldots, \Gamma_k$ induziert. Für alle $G_i \in \{G_1, \ldots, G_k\}$ werden folgende Schritte durchgeführt:

   (a) Falls alle pertinenten Graphen untersucht wurden, dann wird die Berechnung abgebrochen und der berechnete Einfügepfad P sowie die Einbettung Γ'_G als Lösung zurückgegeben. Ansonsten überprüfe, ob G_i den Punkten (a) und (b) von Satz 3.1 genügt. Wenn nicht, dann fahre mit G_{i+1} fort.

   (b) Spiegele die Einbettung Γ_i. Das Ergebnis der Spiegelung ist die Einbettung Γ'_G.

   (c) Der Pfad P kann in drei Abschnitte eingeteilt werden. Der Abschnitt P_2 ist der Abschnitt, der durch G_i führt, der Abschnitt P_3 ist der Abschnitt, der unmittelbar an P_2 anschließt und der Abschnitt P_1 ist der Abschnitt vor P_2. Dabei enthalten P_1 und P_3 keine Kante aus G_i.
   Sei $P_1 := x \rightsquigarrow x'$ und $P_2 := y' \rightsquigarrow y$. Markiere alle Kanten von $G - G_i$ als nicht passierbar und berechne dann einen Einfügepfad P'_2 für die Knoten x' und y' mit
   InsertEdgeEmbedded(Γ'_G, x', y') [3]. P'_2 führt somit durch G_i.

   (d) Wenn kein aufwärtskonsistenter Pfad existiert oder die Konkatenation von P_1, P'_2 und P_3 zu $P' = P_1 + P_2 + P_3$ nicht aufwärtskonsistent ist, dann wird die Spiegelung rückgängig gemacht. Es wird mit dem nächsten pertinenten Graphen G_{i+1} in Schritt (a) fortgefahren.

   (e) Wenn P' weniger Kreuzungen hat als P, dann gehe zu Schritt (a) und fahre mit Γ'_G, P' und dem nächsten pertineten Graphen G_{i+1} fort. An-

[2] Die Reihenfolge der Knoten, die durch die Breitensuche entsteht, muss beachtet werden.
[3] Hierfür muss die Prozedur modifiziert werden, damit der markierte Teilgraph nicht zur Einfügepfadberechnung benutzt wird.

sonsten wird die Spiegelung rückgängig gemacht und es wird mit Γ_G, P und G_{i+1} in Schritt (a) fortgefahren.

6. Das Ergebnis ist ein aufwärtskonsistenter Einfügepfad, falls einer gefunden wird, und eine Einbettung Γ'_G, die aus Spiegelungen von $\Gamma_1, \ldots, \Gamma_k$ resultiert.

Der Nachteil dieses Ansatzes ist, dass er keine Einbettung berechnet, die einen aufwärtskonsistenten Pfad besitzt. Er überprüft auch nicht, ob der Graph G überhaupt eine Einbettung mit einem aufwärtskonsistenten Pfad besitzt. Im schlechtesten Falle wird kein Einfügepfad berechnet, obwohl einer existiert. Die Wahrscheinlichkeit kann aber verringert werden, indem der oben beschriebene Algorithmus mehrmals gestartet wird.

Betrachten wir nun die Laufzeit. Schritt 1 kann in $\mathcal{O}(|V|+|A|)$ berechnet werden. Schritt 2 hat nach Satz 5.2 eine Laufzeit von $\mathcal{O}(k(|A||V|+|A|^2))$. Die Laufzeit von Schritt 3 und 4 ist $\mathcal{O}(|V|+|A|)$. Schritt 5 wird im Worse Case von (c) dominiert. Da ein SPQR-Baum maximal $\mathcal{O}(|A|)$ Knoten besitzen kann, ist die Laufzeit von Schritt 5 $\mathcal{O}(k(|A|^2|V|+|A|^3))$. Somit ist die Gesamtlaufzeit $\mathcal{O}(k(|A|^2|V|+|A|^3))$.

7 Fazit

Zum Abschluss dieser Arbeit werden die erzielten Ergebnisse in diesem Kapitel zusammengefasst. Anschließend werden noch offenen Probleme aufgezeigt, deren Lösung eine sinnvolle Ergänzung dieser Diplomarbeit darstellt.

7.1 Ergebnisse

In dieser Arbeit haben wir ein Lösungsansatz für das *Edge-Insertion-Problem* für sT-Graphen beschrieben, der nicht auf Schichtung basiert. Bisher gab es noch keinen Lösungsansatz dieser Art. Dabei sind wir bei der Analyse des Problems auf folgende Fakten gestoßen, die im Gegensatz zum ungerichteten Fall, eine wichtige Rolle spielen:

- Die Traversierungskosten einer Komponente K ist abhängig von der Wahl der aufwärtsplanaren Einbettung von K.
- Die Kreuzungsanzahl eines Einfügepfades ist abhängig von der Wahl des äußeren Faces einer aufwärtsplanaren Einbettung.
- Es gibt Digraphen, die für zwei gegebene Knoten x und y keine aufwärtsplanare Einbettung mit einem aufwärtskonsistenten Einfügepfad von x nach y besitzen.

Wir haben festgestellt, dass die Kosten eines Einfügepfades durch Spiegelung von Graphkomponenten, durch die der Pfad verläuft, verringert werden können. Aus diesem Grunde wurde untersucht, unter welchem Umstände eine Komponente mit einer festgelegten Einbettung gespielt werden kann, so dass die resultierende Einbettung wieder aufwärtsplanar ist. Das Ergebnis dieser Untersuchung wurde Satz 3.1 festgehalten.

Eine nützliche Datenstruktur für die implizite Aufzählung aller aufwärtsplanaren Einbettungen eines 2-zusammenhängenden sT-Graphen wurde in Kapitel 2.2 beschrieben. Ähnliche Datenstrukturen existierten bisher nur für planare, ungerichtete 2-zusammenhängende Graphen.

In Kapitel 5 wurde ein Ansatz zur Berechnung einer aufwärtskonsistenten Zuweisung für einen sT-Graphen G mit Laufzeit $\mathcal{O}(|V|+|A|)$ angegeben. Der bisherige Ansatz von Bertolazzi et al. [6] benötigt eine Laufzeit $\mathcal{O}(|V|^2+|A|)$. Dieser ist aber nicht auf sT-Graphen beschränkt. Mit der Charakterisierung eines aufwärtskonsistenten

Einfügepfades in Lemma 5.2 haben wir dann den Algorithmus 12 InsertEdgeEmbedded zur Berechnung eines Einfügepfades entworfen. Dieser Algorithmus berechnet für eine gegebene aufwärtsplanare Einbettung eines sT-Graphen einen kürzesten aufwärtskonsistenten Einfügepfad. Er besitzt eine Laufzeit von $\mathcal{O}(k(|A||V| + |A|^2))$, wobei k die Anzahl der möglichen äußeren Faces von G ist.

Basierend auf dem Algorithmus InsertEdgeEmbedded wurde in Kapitel 6 ein verbesserter Ansatz für das *Edge-Insertion-Problem* für 2-zusammenhängende sT-Graphen mit eine Laufzeit von $\mathcal{O}(k(|A|^2|V| + |A|^3))$ beschrieben. Dieser Ansatz benutzt dabei die von uns entwickelte Datenstruktur zur impliziten Aufzählung der aufwärtsplanaren Einbettungen sowie die Ergebnisse der Untersuchung der Spiegelung von Graphkomponenten.

7.2 Offenen Probleme

Die folgende Probleme wurden im Rahmen dieser Arbeit nicht untersucht.

(1) Das Problem für sT-Graphen ohne aufwärtskonsistenten Einfügepfad ist noch ungelöst. Wie lässt sich ein aufwärtskonsistenter Pfad hierfür berechnen ohne eine Schichtung des Graphen durchzuführen? Spielt dieses Problem eine entscheidende Rolle in der Aufwärtsplanarisierung von sT-Graphen?

(2) Falls ein sT-Graph eine Einbettung mit mindestens einem aufwärtskonsistenten Einfügepfad besitzt, wie kann man dann diese Einbettung berechnen?
Das Korollar 5.1 gibt einen Hinweis, wie man feststellen kann, ob ein sT-Graph G einen aufwärtskonsistenten Einfügepfad für zwei gegebenen Knoten x und y besitzt. Wir vermuten, dass G genau dann keine aufwärtskonsistenten Einpfügepfade besitzt, wenn für x und y folgendes gilt:
Sei $\mathcal{T}$ ein SPQR-Baum von G und S ein Skeleton eines R-Knoten von $\mathcal{T}$.

- x und y sind Knoten von S, so dass Korollar 5.1 (b) gilt.
- x und y sind in zwei verschiedene virtuelle Kanten e_x und e_y von S lokalisiert. Sei dann x' und y' Repräsentanten von x und y in S. D.h. es wird in e_x der Dummy-Knoten x' und in e_y der Dummy-Knoten y' eingefügt. Für die beiden Repräsentanten gilt Korollar 5.1 (b).
- x bzw. y ist ein Knoten von S und y' bzw. x' ist ein Repräsentant. Für x und y' bzw. x' und y gilt Korollar 5.1 (b).

(3) Mit einer Untersuchung der Praxistauglichkeit der entwickelten Ansätze im Aufwärtsplanarierungsprozess von sT-Graphen kann festgestellt werden, ob unsere Ansätze in die richtige Richtung weisen. Insbesondere ist der Vergleich mit dem klassischen Sugiyama-Ansatz im Bezug auf Kreuzungsanzahl interessant. Hierfür muss jedoch Problem (1) und (2) gelöst werden.

(4) Eine optimale Lösung des *Edge-Insertion-Problems* für sT-Graphen steht noch aus. Wie gut ist die von uns entwickelte Lösung im Vergleich zur optimalen Lösung?

Die genannten Probleme können in weitere Arbeiten untersucht werden.

Abbildungsverzeichnis

Literaturverzeichnis

[1] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom. Theory Appl.*, 7(5-6):303–325, 1997.

[2] G. Di Battista, W. P. Liu, and I. Rival. Bipartite graphs, upward drawings, and planarity. *Inf. Process. Lett.*, 36(6):317–322, 1990.

[3] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theor. Comput. Sci.*, 61:175–198, 1988.

[4] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5):956–997, 1996.

[5] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display in drawing graphs. In *Symposium on Computational Geometry*, pages 51–60, 1989.

[6] P. Bertolazzi and G. Di Battista. On upward drawing testing of triconnected digraphs (extended abstract). In *Symposium on Computational Geometry*, pages 272–280, 1991.

[7] P. Bertolazzi, G. Di Battista, C. Mannino, and R. Tamassia. Optimal upward planarity testing of single-source digraphs. *SIAM J. Comput.*, 27(1):132–169, 1998.

[8] P. Bertolazzi, R. F. Cohen, G. Di Battista, R. Tamassia, and I. G. Tollis. How to draw a series-parallel digraph (extended abstract). In *SWAT '92: Procedings of the Third Scandinavian Workshop on Algorithm Theory*, pages 272–283, London, UK, 1992. Springer-Verlag.

[9] E. Coffman and R. Graham. Optimal scheduling for two processor systems. *Acta Informatica*, 1:200–213, 1972.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

[11] P. Eades and D. Kelly. Heuristics for drawing 2-layered networks. *Ars Combinatoria*, 21A:89–98, 1986.

[12] P. Eades, X. Lin, and W. F. Smyth. Heuristics for the feedback arc set problem. Technical Report 1, Curtin University of Technology, Perth, Australia, 1989.

[13] P. Eades and K. Sugiyama. How to draw a directed graph. *J. Inf. Process.*, 13(4):424–437, 1990.

[14] P. Eades and N.C. Wormald. The median heuristic for drawing two-layered networks. Technical Report 69, University Queensland, 1986.

[15] P. Eades and N.C. Wormald. Edge crosssings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.

[16] M. Eiglsperger, M. Kaufmann, and F. Eppinger. An approach for mixed upward planarization. *J. Graph Algorithms Appl.*, 7(2):203–220, 2003.

[17] A. Frank. How to make a digraph strongly connected. *Combinatorica*, 1:148–153, 1981.

[18] E. R. Gansner, S. C. North, and K. P. Vo. DAG - a programm that draws directed graphs. *Software Practice and Experiments*, 18(11):1047–1062, 1988.

[19] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic and Discrete Methods*, pages 312–316, 1983.

[20] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[21] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing, Princeton, 1994*, pages 286–297. Springer, 1995.

[22] A. Garg and R. Tamassia. Upward planarity testing. *Order: A Journal on the Theory of Ordered Sets and Its Applications*, 12:109–133, 1995.

[23] J. L. Gross and J. Yellen. *Handbook of Graph Theory*. CRC Press, 2004.

[24] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90, 2000.

[25] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.

[26] J. Hopcroft and R. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[27] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

[28] M. D. Hutton and A. Lubiw. Upward planar drawing of single-source acyclic digraphs. *SIAM J. Comput.*, 25(2):291–311, 1996.

[29] M. Jünger and P. Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications*, 1:1–25, 1997.

[30] D. Kelly. Fundamentals of planar ordered sets. *Discrete Math.*, 63(2-3):197–216, 1987.

[31] S. Lam and R.Sethi. Worst case analysis of two scheduling algorithms. *SIAM J. Computing*, 6:518–536, 1977.

[32] E. Mäkinen. Experiments in drawing two-level hierarchical graphs. *Intern. J. Computer Math.*, 37:129–135, 1990.

[33] A. Papakostas. Upward planarity testing of outerplanar dags. In *Graph Drawing '94*, volume 986 of *Lecture Notes in Computer Science*, pages 298–306. Springer Verlag, 1995.

[34] K. Sugiyama, S. Tagawa, and M. Toda. Effective representations of hierarchical structures. Technical Report 8, IIAS-SIS, Fujitsu Limited, Numazu,Shizuoka, Japan, 1979.

[35] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Sys. Man. Cybern.*, 11(2):109–125, 1981.

[36] R. Weiskircher. *New Applications of SPQR-Trees in Graph Drawing.* PhD thesis, Universität des Saarlandes, 2002.