Karlsruhe Institute of Technology

Knickminimierung in Orthogonalen Zeichnungen fast-planarer Graphen mit fester Topologie

Diplomarbeit
von

Robert Jungblut

An der Fakultät für Informatik
Institut für theoretische Informatik

Erstgutachter:	Prof. Dr. Dorothea Wagner
Zweitgutachter:	Prof. Dr. Peter Sanders
Betreuende Mitarbeiter:	Dr. Ignaz Rutter
	Dipl.-Inform. Thomas Bläsius

Bearbeitungszeit: 01. August 2012 – 31. Januar 2013

Selbstständigkeitserklärung

Hiermit erkläre ich, dass diese Arbeit selbstständig verfasst habe und dass alle wörtlich oder sinngemäß übernommene Stellen in der Arbeit als solche gekennzeichnet sind.

Karlsruhe, 14. März 2013

Zusammenfassung

Wir betrachten in dieser Arbeit das folgende Problem: Gegeben seien die Planarisierung eines nichtplanaren Graphen mit Maximalgrad 4 sowie eine Abbildung flex, die jeder Kante des Graphen eine Flexibilität zuweist. Gibt es eine Orthogonale Zeichnung dieser planaren Einbettung, so dass jede Kante e höchstens flex(e) Knicke aufweist? Wir untersuchen zunächst den Fall mit einer einzelnen Kreuzung und zeigen, dass es eine solche Zeichnung selbst dann nicht ohne weiteres gibt, wenn auf jeder Kante zwei Knicke erlaubt sind, anders als im planaren Fall. anschließend zeigen wird, dass dieses Problem im allgemeinen NP-schwer ist. Schließlich stellen wir ein ganzzahliges Lineares Programm zum Finden einer solchen Zeichnung vor, evaluieren dieses Programm anhand unterschiedlicher Beispielgraphen und zeigen damit, dass nur in extremen Fällen die erhöhte Komplexität zum Tragen kommt.

Inhaltsverzeichnis

1. Einleitung

In der Graphentheorie ist es oftmals wünschenswert, einen Graph derart darstellen zu können, dass er für Menschen leicht lesbar ist. Der Bereich des Graphzeichnens beschäftigt sich daher mit unterschiedlichen Kriterien für die Lesbarkeit. So ist es wichtig, dass die darzustellenden Elemente voneinander genügend Abstand haben, dass die Linienzüge der Kanten klar verlaufen und nicht miteinander verwechselt werden können, dass Symmetrien existieren um ein leichtes Erfassen der Darstellung zu ermöglichen, und ähnliches. Eine probate Vorgehensweise für Graphen mit Maximalgrad 4 ist hierbei das *orthogonale Zeichnen*, bei dem Kanten als vertikale beziehungsweise horizontale Linienzüge dargestellt werden, die je nach Verlauf der Kante eine gewisse Anzahl rechtwinkliger Knicke aufweisen. Dies sorgt für eine gute Übersichtlichkeit der Zeichnung, für klare Strukturen und somit für eine erhöhte Lesbarkeit. Abbildung 1.1 zeigt ein Beispiel.

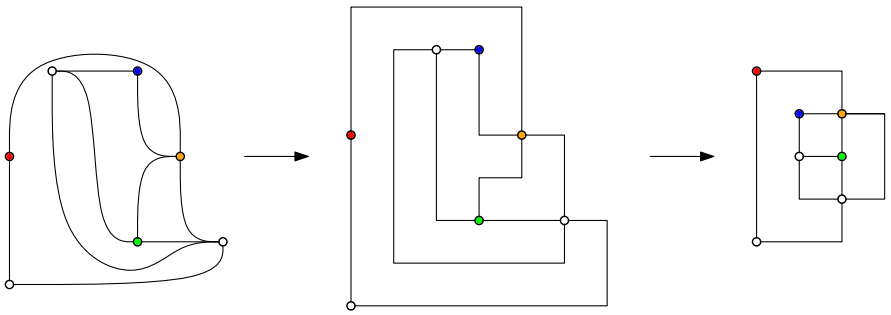


Abbildung 1.1: Der linke Zeichnung des Graphs ist aufgrund der fehlenden Struktur nur schwer lesbar. Die Orthogonale Zeichnung in der Mitte ist übersichtlicher, aber wegen der hohen Anzahl von Knicken ebenfalls nicht optimal. Die rechte Zeichnung mit minimaler Anzahl an Knicken hingegen ist klar und präzise.

Gleichzeitig gibt es weitere Anwendungsgebiete wie den Entwurf integrierter Schaltkreise (VLSI-Design) oder das sogenannte Floorplanning beim Chipentwurf, in denen Orthogonale Zeichnungen der zu realisierenden Schaltungen von Vorteil sind.

In beiden Fällen ist es dabei wünschenswert, eine Orthogonale Zeichnung zu finden, die gewisse Regeln bezüglich der Anzahl der Knicke ihrer Kanten einhalten. So ist ein Graph umso leichter lesbar, je weniger Knicke er insgesamt aufweist. Andererseits ist es aber auch nicht wünschenswert, dass eine einzelne Kante sehr viele Knicke hat. Entsprechend

konzentrieren sich die Optimierungsbemühungen im Allgemeinen entweder auf eine Minimierung der Gesamtzahl an Knicken oder auf das Finden einer Zeichnung mit einer gewissen Höchstzahl an Knicken pro Kante. Der Begriff Knickminimierung umfasst dabei beide Problemstellungen.

Soll ein nichtplanarer Graph in der Ebene gezeichnet werden (man spricht von *Planarisierung* des Graphen) sind ähnliche Qualitätskriterien anzuwenden: je geringer die Anzahl der benötigten Kreuzungen pro Kante oder insgesamt ist, desto besser. Bei der Orthogonalen Zeichnung nichtplanarer Graphen gilt es folglich, verschiedene Kriterien gegeneinander abzuwägen. Bisherige Arbeiten konzentrieren sich im Allgemeinen aber entweder auf die Knickminimierung, was auf Kosten der Anzahl der Kreuzungen geht, oder aber auf die Kreuzungsminimierung, ohne Berücksichtigung der Anzahl der Knicke.

1.1 Formulierung des Problems

Wir betrachten nun nichtplanare Graphen mit einer festen Planarisierung, gegeben sei also eine planare Zeichnung des Graphen, in der festgelegt ist, wie sich die einzelnen Kreuzungen genau schneiden. Desweiteren sei für jede Kante eine *Flexibilität* gegeben, die festlegt, wie viele Knicke diese Kante maximal haben darf.

Wir wollen nun entscheiden, ob es eine Orthogonale Zeichnung dieser Planarisierung gibt, in der jede Kante höchstens so viele Knicke hat, wie ihre Flexibilität zulässt. Dabei betrachten wir unterschliedliche Fälle bezüglich der Anzahl der sich kreuzenden Kanten, der Anzahl der Kanten, die eine einzelne Kante kreuzen, sowie insbesondere der Anzahl maximal zulässiger Knicke pro Kante. Zum Abschluss stellen wir ein ganzzahliges Lineares Programm vor, das die Problemstellung erfasst, und evaluieren die Laufzeit dieses ILPs anhand unterschiedlicher Beispielgraphen.

1.2 Motivation

Während planare Graphen einige nützliche Eigenschaften haben, die es erlauben, auf ihnen effiziente und einfache Algorithmen zu entwerfen, sind sie in der Praxis von eher geringerer Bedeutung. Es wäre daher wünschenswert, die auf planare Graphen zugeschnittenen Werkzeuge zur Knickminimierung auch auf nichtplanaren Graphen verwenden zu können.

Die wenigen vorhandenen Mechanismen zur Knickminimierung auf nichtplanaren Graphen setzen aber voraus, dass die Einbettung des Graphen frei gewählt werden kann. Dies hat zwei Nachteile: zum einen wird in der Folge nicht darauf geachtet, eine qualtitativ hochwertige Einbettung bezüglich der Anzahl der Kreuzungen zu wählen. Und zum anderen ist es so nicht möglich, gewisse Anforderungen an die Art der Einbettung zu stellen, zum Beispiel welche Kanten sich kreuzen sollen oder welche Kanten kreuzungsfrei bleiben müssen.

Es ist also von großem Interesse, Knickminimierung auf einer fest vorgegebenen Planarisierung zu untersuchen. Dabei soll es genau wie im planaren Fall möglich sein, für jede Kante die Anzahl der erlaubten Knicke festzulegen. Doch die vorhandenen Verfahrensweisen zur Knickminimierung auf planaren Graphen sind nicht in der Lage, diese Bedingungen auch für jene Kanten zu überprüfen, die die Planarität des Graphen verletzen. Dass wir im Laufe dieser Arbeit Verfahrensweisen vorstellen, die diese Leistung erbringen, schließt also eine wichtige Lücke im Bereich der Knickminimierung.

1.3 Stand der Forschung

Ein Großteil der vorhandenen Forschung beschäftigt sich wie erwähnt mit der Knickminimierung auf planaren Graphen, wobei oftmals die Einbettung frei gewählt werden kann.

	Freie Einbettung	Feste Einbettung
Planar	NPC (flex = 0), $O(n^{5/2})$ (flex $\geq$ 1), O(n) (flex = 2)	P
Nichtplanar	$O(n)$ (flex = 2)	?

Tabelle 1.1: Komplextität der unterschiedlichen Variationen der Knickminimierung

1.3.1 Knickminimierung mit freier Einbettung

In planaren Graphen ist es laut Garg und Tamassia [GT95] NP-schwer, zu entscheiden, ob ein Graph eine Einbettung mit 0 Knicken zulässt. Folglich ist es ebenso NP-schwer, die Anzahl der benötigten Knicke über alle möglichen Einbettungen des Graphen zu minimieren. Di Battista et al. [BLV98] zeigen dass dies nicht mehr gilt, falls die Knoten des Graphen höchstens den Grad 3 haben oder der Graph serien-parallel ist. Für diese Fälle stellen sie einen Polynomialzeitalgorithmus vor.

Eine Einbettung, die sich mit höchstens zwei Knicken pro Kante zeichnen lässt, gibt es laut Biedl und Kant [BK94] in jedem planaren Graphen ausser dem Oktaeder. Der dort vorgestellte Linearzeitalgorithmus kann eine derartige Zeichnung auch für nichtplanare Graphen finden, stellt dabei aber keine Einschränkungen an die Anzahl der Kreuzungen. Das Ergebnis sind Graphen mit sehr hohen Kreuzungszahlen, was besonders in Bezug auf die Lesbarkeit wenig zufriedenstellend ist.

Bläsius et al. [BKRW11] schließlich verallgemeinern das Problem der Knickminimierung zum sogenannten flex-draw-Problem, indem sie jeder Kante eine individuelle Höchstzahl an Knicken erlauben, die sogenannte Flexibilität. Für den Fall dass jede Flexibilität positiv ist geben sie einen Polynomialzeitalgorithmus an. Damit lässt sich also unter anderem auch entscheiden, ob ein Graph mit höchstens einem Knick pro Kante eingebettet werden kann.

1.3.2 Knickminimierung mit fester Einbettung

Anstatt alle möglichen Einbettungen eines planaren Graphen zu betrachten kann man auch eine feste Einbettung des Graphen wählen und ausschließlich auf dieser arbeiten. Tamassia [Tam87] stellt für diesen Fall ein Flussnetzwerk vor, mit dessen Hilfe sich die Anzahl der in einer Orthogonalen Zeichnung dieser Einbettung benötigten Knicke in Polynomialzeit minimieren lässt.

Dieses Flussnetzwerk lässt sich auch derart modifizieren, dass es genau dann einen gültigen Fluss enthält, wenn es eine Zeichnung gibt, welche die von Bläsius et al. [BKRW11] vorgestellten Flexibilitäten berücksichtigt. Morgana et al. [MdMS04] beschreiben, welche Kriterien eine planare Einbettung erfüllen muss, damit eine Zeichnung mit höchstens einem Knick pro Kante möglich ist.

Tabelle 1.1 gibt eine Übersicht über die vorliegenden Ergebnisse.

Es gibt also offensichtlich keine Ergebnisse darüber, wie die Komplexität der Knickminimierung sich im nichtplanaren Fall mit fester Einbettung des Graphen verhält. Dies ist wie gesagt jedoch von großem Interesse, nicht nur weil derartige Probleme von praktischer Bedeutung sind sondern vor allem, damit die Erkenntnisse aus der Knickminimierung mit den Erkenntnissen aus der Kreuzungsminimierung kombiniert werden können.

1.3.3 Kreuzungsminimierung

Laut Garey und Johnson [GJ83] ist es NP-schwer zu entscheiden, ob es zu einem gegebenen nichtplanaren Graphen eine Zeichnung gibt, deren Anzahl an Kreuzungen einen gewissen

Wert nicht überschreitet. Da das Minimieren der Anzahl an Kreuzungen aber von großem praktischen Interesse ist, gibt es zahlreiche Arbeiten, die in Abhängigkeit von der Beschaffenheit des Graphen möglichst optimale Ergebnisse zu erzielen versuchen. Eine ausführliche Übersicht über die gängigsten Verfahrensweisen bieten Gutwenger und Mutzel [GM04]. Die Vorgehensweise dabei ist im Allgemeinen das Finden eines planaren Subgraphen gefolgt vom Einfügen der fehlenden Kanten unter Beachtung verschiedener Kriterien. Arbeiten, die dabei auch die Anzahl der Knicke in einer Orthogonalen Zeichnung betrachten, sind uns nicht bekannt.

1.4 Struktur dieser Arbeit

In Kapitel 2 werden wir die für unsere Arbeit wichtigsten Begriffe und Notationen formal vorstellen, die zentrale Problemstellung der Arbeit klar formulieren und erste Verfahrensweisen einführen, die wir im Laufe des öfteren benötigen werden.

Kapitel 3 behandelt den Fall, der laut Biedl und Kant [BK94] in planaren Graphen stets lösbar ist: das Zeichnen mit zwei Knicken pro Kante. Wir untersuchen, ob dies auch auf nichtplanaren Graphen möglich ist, mit der einfachst möglichen Konfiguration: einem Graph mit nur einer Kreuzung. Dazu untersuchen wir die Beschaffenheit der sich kreuzenden Kanten und stellen so fest, dass es durchaus Situationen geben kann, in der eine dieser Kanten mindestens drei Knicke benötigt.

In Kapitel 4 verallgemeinern wir schließlich die Problemstellung auf den aus flex-draw [BKRW11] bekannten Fall, einen beliebigen Graphen mit Kanten unterschiedlicher Flexibilität. Wir zeigen, dass dieses auf planaren Graphen in Polynomialzeit lösbare Problem auf festen Einbettungen nichtplanarer Graphen NP-schwer ist. Dabei stellen wir in Anlehnung an Tamassia [Tam87] ein modifiziertes Flussnetzwerk vor, das genau dann einen gültigen Fluss enthält, wenn es eine entsprechende Orthogonale Zeichnung gibt. Dafür müssen wir dem Flussnetzwerk einige zusätzliche Bedingungen hinzufügen, die das Finden eines Flusses NP-schwer machen. Wir stellen ein ganzzahliges Lineares Programm vor, zum Finden eines Flusses verwendet werden kann.

Die Laufzeit dieses Linearen Programms evaluieren wir schließlich in Kapitel 5 anhand von Beispielgraphen, um zu untersuchen, inwiefern die Erhöhung der Komplexität in praxisnahen Fällen eine Rolle spielt. Wir werden dabei zeigen, dass lediglich in sehr dichten Graphen, die folglich eine hohe Anzahl sich kreuzender Kanten aufweisen, ein relevanter Unterschied zwischen der Laufzeit auf planaren Graphen und auf nichtplanaren Graphen besteht.

2. Grundlagen

Wir stellen die benötigten Grundlagen und Notationen vor, die wir im Verlauf der Arbeit verwenden werden.

2.1 Orthogonale Zeichnung und Orthogonale Repräsentation

Sei $G = (V, E)$ ein *einfacher planarer Graph mit Maximalgrad* 4, also ein planarer Graph ohne Mehrfachkanten und Schlingen, dessen Knoten höchstens 4 inzidente Kanten haben. Sei Γ eine *Orthogonale Zeichnung* des Graphen G, also eine Zeichnung, bei der die Kanten aus E dargestellt sind als Linienzüge, die horizontal oder vertikal verlaufen und als Knicke ausschließlich 90°-Winkel (bzw. 270°-Winkel) aufweisen. Dabei jede Kante ausschließlich in eine Richtung geknickt, es kann also nicht vorkommen, dass zwei Knicke einer Kante in derselben Facette unterschiedliche Winkel bilden. Dies ist keine Einschränkung der Allgemeinheit, da derartige Linienzüge stets ersetzt werden können durch eine gerade Linie [Tam87]. Seien $\mathcal{E}$ die von der Zeichnung Γ induzierte planare Einbettung des Graphen G und F die Facetten in dieser Einbettung. Zur Beschreibung der Orthogonalen Zeichnung Γ stellen wir die Orthogonale Repräsentation $\mathcal{R}$ nach Tamassia vor [Tam87].

Sei $e \in E$ eine Kante im Graph G. Dann definieren wir die *Rotation* $|\operatorname{rot}(e)|$ als die Anzahl der Knicke, die auf der Kante e liegen, also die Anzahl der Knicke, die der Linienzug zur Kante e in der Orthogonalen Zeichnung Γ aufweist. Sei desweiteren f eine in der Einbettung $\mathcal{E}$ zur Kante e inzidente Facette. Dann definieren wir $\operatorname{rot}(e_f)$ als Indikator dafür, in welche Richtung die Knicke auf der Kante e bezüglich der Facette f geknickt sind. Falls die Knicke in der Facette f 90°-Winkel bilden, sei $\operatorname{rot}(e_f) = |\operatorname{rot}(e)|$. Falls sie hingegen 270°-Winkel bilden, sei $\operatorname{rot}(e_f) = -|\operatorname{rot}(e)|$. Ist die Kante e zu einer weiteren Facette $g \neq f$ inzident, so folgt automatisch $\operatorname{rot}(e_f) = -\operatorname{rot}(e_g)$. Ist sie hingegen ausschließlich zur Facette f inzident (weil es sich um eine Brücke handelt), so nimmt $\operatorname{rot}(e_f)$ zwei unterschiedliche Werte an. Die entsprechende Interpretation folge im Zweifelsfall aus dem Zusammenhang. Abbildung 2.1 veranschaulicht diese Rotationen.

Die Kanten, die eine Facette begrenzen, bilden auch dort einen Winkel, wo sie in einem Knoten aufeinandertreffen. Wir benötigen also auch für die Knoten ein Maß für die Rotation, das analog zu den Knicken in Kanten dieselben Werte für dieselben Winkel annimmt. Seien dazu $v \in V$ ein Knoten im Graph G und f eine in der Einbettung $\mathcal{E}$ zum Knoten v inzidente Facette. Sei $\alpha(v_f)$ der Winkel, den die zum Knoten v inzidenten Kanten in der Facette f bilden. Dann erhalten wir $\operatorname{rot}(v_f) = 2 - \alpha(v_f)/90°$. Die Summe der Rotationen

um einen Knoten $v \in V$ ist somit gleich $2 \cdot \text{grad}(v) - 4$, wobei $\text{grad}(v)$ die Anzahl der zum Knoten v inzidenten Kanten ist.

Insgesamt ist für jede Facette f die Summe der Rotationen aller Knoten und aller Kanten -4, falls f die äußere Facette ist, und $+4$, falls f eine innere Facette ist. Abbildung 2.1 zeigt ein Beispiel für eine Orthogonale Zeichnung. Die Werte der Orthogonalen Repräsentation sind an den Kanten beziehungsweise an den Knoten in der entsprechenden Facette notiert. Aus Gründen der Übersichtlichkeit wurden Kanten, an denen die Rotation Null ist, nicht beschriftet.

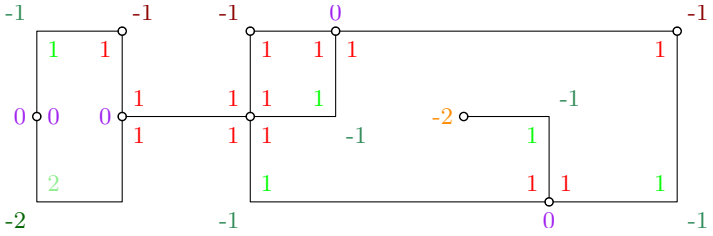


Abbildung 2.1: Orthogonale Zeichnung mit Orthogonaler Repräsentation.

Diese Eigenschaften der Orthogonalen Repräsentation sind nicht nur notwendig sondern hinreichend zur Beschreibung einer Orthogonalen Zeichnung. Folglich lässt sich durch Angabe der Orthogonalen Repräsentation die Orthogonale Zeichnung (abgesehen von Abständen und Längen der Linien) eindeutig festlegen [Tam87]. Der Vorteil daran ist, dass derartige Eigenschaften der Geometrie damit vernachlässigt werden können. Zudem erlaubt die abstrakte Form der Orthogonalen Repräsentation die effiziente Durchführung von Operationen wie zum Beispiel dem Ändern der Anzahl der Knicke einer Kante.

2.2 Planarisierung nichtplanarer Graphen

Sei nun zu einem nichtplanaren Graphen eine Zeichnung in der Ebene gegeben. In dieser Zeichnung schneidet also eine gewisse Anzahl der Kanten andere Kanten in einem oder mehreren Punkten. Derartige Kanten, die andere Kanten kreuzen, nennen wir *Kreuzungskanten*. Ersetzt man die Schnittpunkte der Kreuzungskanten durch Knoten, welche man zu den Knoten des Graphen hinzunimmt, und die Kreuzungskanten durch jene Kanten, die den Pfad über die hinzugenommenen Knoten bilden, so erhält man einen planaren Graphen. Diesen planaren Graphen nennen wir die *Planarisierung* des Graphen. Die hinzugenommenen Knoten nennen wir *Kreuzungsknoten*; die zu den Kreuzungsknoten inzidenten Kanten nennen wir *Kantensegmente* jener Kreuzungskanten, zu denen sie gehören. Da sich die Planarisierung aus einer festen Zeichnung ergibt, ist mit ihr automatisch auch eine Einbettung gegeben. Da wir die Topologie des Graphen festhalten wollen, werden wir diese Einbettung als feste Einbettung beibehalten.

Natürlich müssen wir nicht von einer vorhandenen Zeichnung eines nichtplanaren Graphen ausgehen. Stattdessen können wir auch zu einem nichtplanaren Graphen selbst eine Einbettung in der Ebene erarbeiten, beispielsweise mit den von Gutwenger und Mutzel [GM04] vorgestellten Verfahrensweisen. Dabei lassen sich auch Qualitätskriterien bezüglich der Beschaffenheit der Einbettung beachten. So oder so gehen wir aber von einer festen Einbettung aus.

Da wir fortan ausschließlich mit Planarisierungen von Graphen arbeiten werden, führen wir hier die entsprechende Notation formal ein: Sei $G = (V, E)$ die Planarisierung eines einfachen nichtplanaren Graphen $\tilde{G} = (\tilde{V}, \tilde{E})$. Die Knotenmenge V enthält als echte Teilmenge die Menge der Kreuzungsknoten, also jener Knoten, die die Kreuzung zweier Kanten in der Planarisierung verkörpern. Da diese Kreuzungsknoten nicht Teil des nichtplanaren

Graphen sind, ist ihre Menge disjunkt zur Knotenmenge $\tilde{V}$ des nichtplanaren Graphen. Genauer gesagt setzt sich die Knotenmenge V der Planarisierung aus der Knotenmenge $\tilde{V}$ des nichtplanaren Graphen und der Menge der Kreuzungsknoten zusammen.

Analog enthält die Kantenmenge $\tilde{E}$ des nichtplanaren Graphen als Teilmenge die Menge der Kreuzungskanten, also jener Kanten, die in der Planarisierung durch einen Pfad von Kanten ersetzt wurde, die jeweils zu mindestens einem Kreuzungsknoten inzident sind. Jene Kanten bilden die Menge der *Kantensegmente* der zugehörigen Kreuzungskante; die Menge der Kantensegmente aller Kreuzungskanten ist eine Teilmenge der Kantenmenge E der Planarisierung. Kanten der Planarisierung, die keine Kantensegmente sind, nennen wir *kreuzungsfreie Kanten*. Ihre Menge ist Teilmenge sowohl der Kantenmenge $\tilde{E}$ des nichtplanaren Graphen als auch der Kantenmenge E der Planarisierung. Somit setzt sich die Kantenmenge $\tilde{E}$ zusammen aus den kreuzungsfreien Kanten und den Kreuzungskanten. Diese Kanten meinen wir künftig, wenn wir ohne weitere Einschränkung von *Kanten* sprechen. Wollen wir hingegen Bezug nehmen auf die Kantenmenge E der Planarisierung, welche sich zusammensetzt aus den kreuzungsfreien Kanten und den Kantensegmenten, so sprechen wir von den *Planarisierungskanten*. Natürlich verwenden wir aber auch weiterhin den Begriff Kante für einzelne Kanten unabhängig davon, ob es sich um ein Kantensegment, eine kreuzungsfreie Kante oder eine Kreuzungskante handelt, wenn die genaue Art der Kante keine Rolle spielt.

Man beachte, dass sich die Menge der Kreuzungskanten anhand einer gegebenen Planarisierung ablesen lassen, ohne dass der ursprüngliche nichtplanare Graph bekannt ist. Die Anordnung der Kantensegmente um den Kreuzungsknoten ermöglicht eine eindeutige Zuordnung zweier benachbarter zueinander gehöriger Kantensegmente, so dass die Menge aller zueinander gehöriger Kantensegmente ermittelt werden kann.

2.3 Nichtplanares flex-draw

Je nach Anwendung kann es von Vorteil sein, unterschiedlichen Kanten eines Graphen eine unterschiedliche Anzahl an Knicken zu erlauben. Wir verwenden daher die Abbildung $\text{flex} : \tilde{E} \to \mathbb{N}_0$ um in einer gegebenen Planarisierung sämtlichen Kanten eine Obergrenze für die Anzahl der Knicke zuzuweisen. Wir sprechen hierbei von der *Flexibilität* der entsprechenden Kante. Gilt beispielsweise für eine Kreuzungskante e mit zwei Kantensegmenten $\text{flex}(e) = 2$, so darf diese Kreuzungskante höchstens zwei Knicke haben; wir sagen die Kante e hat Flexibilität 2. Dabei dürfen diese zwei Knicke beliebig auf die beiden Kantensegmente verteilt werden; die Kantensegmente haben keine separate Flexibilität, sie können beliebig viele Knicke haben, solange die Summe der Kantensegmente einer Kreuzungskante nicht deren Flexibilität übersteigt. Durch das Festlegen einer Flexibilität der Kreuzungskanten erhalten wir also einen echten Mehrwehrt gegenüber einer festen Flexibilität der Planarisierungskanten: würde man die Flexibilität beider Kantensegmente auf 2 setzen, so dürfte die Kreuzungskante bis zu vier Knicke haben. Würde man sie hingegen jeweils auf 1 setzen, wäre eine Verteilung beider Knicke auf ein einzelnes Segment nicht mehr zulässig.

Dies führt uns direkt zur zentralen Fragestellung dieser Arbeit.

Definition 2.1 (Nichtplanares flex-draw)**.** *Gegeben seien eine Planarisierung eines nichtplanaren Graphen sowie die zugehörige Flexibilität der einzelnen Kanten des nichtplanaren Graphen. Es gilt zu entscheiden, ob es eine Orthogonale Zeichnung der Planarisierung gibt, in der keine Kante mehr Knicke hat, als ihre Flexibilität zulässt. Wir nennen dieses Entscheidungsproblem nichtplanares flex-draw, kurz np-flex. Die Planarisierung samt Flexibilität nennen wir np-flex-Instanz.*

Hat in einer Orthogonalen Zeichnung einer Planarisierung eine Kante mehr Knicke, als ihre Flexibilität zulässt, sagen wir die Flexibilität der Kante ist verletzt. Falls es in jeder möglichen Orthogonalen Zeichnung einer np-flex-Instanz mindestens eine Kante gibt, deren Flexibilität verletzt ist, sagen wir diese np-flex-Instanz ist nicht lösbar. Gibt es hingegen eine Orthogonale Zeichnung, in der jede Kante höchstens so viele Knicke hat, wie ihre Flexibilität zulässt, sagen wir die np-flex-Instanz ist lösbar. Dabei ist Lösbarkeit nicht mit Entscheidbarkeit zu verwechseln: es ist immer entscheidbar, ob eine np-flex-Instanz lösbar ist oder nicht.

Zum einen wollen wir np-flex nun für einezlne Instanzen lösen, also für konkret vorgegebene Planarisierungen mit genau definierter Flexibilität. Darüber hinaus interessieren wir uns aber auch für allgemeinere Fragestellungen, bei denen die Planarisierung, die Flexibilität oder beides nicht genau vorgegeben sind. Dabei lassen sich diverse Charakteristika beliebig beschreiben. Wird eine Eigenschaft nicht genauer festgelegt, so ist das np-flex-Problem für alle möglichen Ausprägungen dieser Eigenschaft zu betrachten. Eine derartige Eigenschaft wäre zum Beispiel, dass alle Kanten dieselbe Flexibilität haben. Wir sprechen in diesem Fall von *globaler Flexibilität*.

Wenn die Problemstellung beispielsweise lautet „löse np-flex auf Instanzen, deren Kanten mindestens Flexibilität 1 haben“, so gilt es zu untersuchen, ob in jeder beliebigen Planarisierung eines nicht näher definierten nichtplanaren Graphen eine Orthogonale Zeichnung gefunden werden kann, in der keine Kante ihre Flexibilität verletzt, unabhängig davon, wie die Flexibilitäten verteilt sind, solange keine davon kleiner ist als 1.

2.4 Kanten glätten und knicken

Beim Lösen von np-flex werden wir immer wieder auf Orthogonale Zeichnungen stoßen, in denen die Flexibilität einer Kante verletzt ist. Um eine Zeichnung zu finden, die diese np-flex-Instanz löst, müssen wir dann die Anzahl der Knicke auf dieser Kante reduzieren. Wir sagen, die Kante beziehungsweise ihr Knick soll *geglättet* werden.

Seien $e \in E$ eine Kante im Graphen G, f und g die zu e in der Einbettung $\mathcal{E}$ inzidenten Facetten und sei $\text{rot}(e_f) < 0$, d.h. e hat mindestens einen Knick, und der 90°-Winkel dieses Knickes liegt in der Facette f, der 270°-Winkel in der Facette g. Soll die Kante e nun geglättet werden, muss $\text{rot}(e_f)$ erhöht werden und $\text{rot}(e_g)$ reduziert werden. Abbildung 2.2 veranschaulicht diese Änderung der beiden Rotationen mit Hilfe eines *Pfeiles*: auf der Seite, auf der er auftritt, erhöht er den Wert der Rotation der Kante, auf der anderen Seite senkt er den Wert. Wir nennen einen solchen Pfeil einen *reduzierenden Pfeil*.

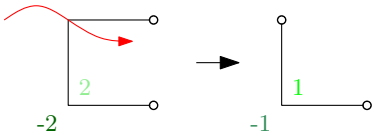


Abbildung 2.2: Glätten einer Kante über einen reduzierenden Pfeil.

Wir definieren die Begriffe Pfeil und reduzierender Pfeil formal. Sei dazu G^* der Dualgraph des Graphen G und $\mathcal{E}^*$ die zur Einbettung $\mathcal{E}$ duale Einbettung von G^*.

Definition 2.2 (Pfeil)**.** *Ein Pfeil einer Kante e eines Graphen G ist eine zur Kante e duale Kante im Dualgraph G^* des Graphen G, die gerichtet ist. Ist die Kante in der Einbettung $\mathcal{E}^*$ gerichtet von dem zur Facette f dualen Knoten zu dem zur Facette g dualen Knoten (wobei f und g die in der Einbettung $\mathcal{E}$ zur Kante e inzidenten Facetten sind) sprechen wir von einem Pfeil einer Kante e von f nach g.*

Definition 2.3 (Reduzierender Pfeil)**.** *Ein reduzierender Pfeil ist ein Pfeil einer Kante e von f nach g, für den* $\mathrm{rot}(e_f) < 0$ *gilt.*

Würde nun nur eine einzelne Kante geglättet werden, so würde sich die Summe der Rotationen in den Facetten f bzw. g erhöhen bzw. verringern. Da die Summe der Rotationen aber stets gleich bleiben muss, muss dieser Unterschied wie in einem Flussnetzwerk ausgeglichen werden. So muss also für die Benutzung eines reduzierenden Pfeiles einer Kante e von f nach g in der Facette g (in der sich nun eine überschüssige Flusseinheit befindet) die Rotation einer von e verschiedenen Kante erhöht werden. Hatte diese Kante zuvor einen 270°-Winkel in der Facette g, so wird sie also ebenfalls über einen reduzierenden Pfeil geglättet. Hatte sie hingegen keinen Knick oder gar einen Knick in der anderen Richtung, erhält sie durch den entsprechenden Pfeil einen Knick hinzu. Anstatt die Kante zu glätten, *knicken* wir sie also.

Auf diese Weise wird der überschüssige Fluss zwischen den Facetten über Kanten weitergereicht, bis die Facette f erreicht wird, von der der ursprüngliche reduzierende Pfeil ausgeht. Diese Vorgehensweise entspricht also der Suche nach einem Kreis im Dualgraph des Graphen. Wir nennen einen solchen Kreis einen *reduzierenden Kreis.*

Definition 2.4 (Reduzierender Kreis)**.** *Ein reduzierender Kreis einer Kante e eines Graphen G ist ein einfacher gerichteter Kreis im Dualgraph G^* des Graphen G, der einen reduzierenden Pfeil der Kante e enthält.*

Abbildung 2.3 zeigt ein Beispiel für einen reduzierenden Kreis sowie die entsprechende Änderung in der Orthogonalen Zeichnung.

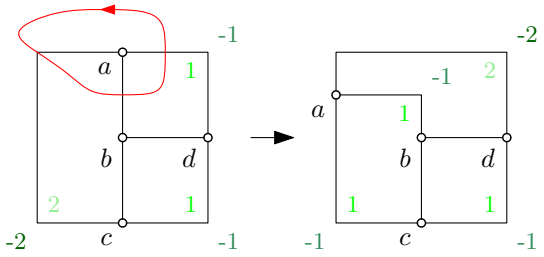


Abbildung 2.3: Glätten und Knicken mehrerer Kanten über einen reduzierenden Kreis der Kante $\{a, c\}$.

Nun wollen wir bei dieser Vorgehensweise aber nicht dafür sorgen, dass eine Kante mehr Knicke erhält, als ihre Flexibilität zulässt. Wir wollen also keine Pfeile verwenden, die diese Bedingung verletzen.

Definition 2.5 (Ungültiger Pfeil)**.** *Ein ungültiger Pfeil ist ein Pfeil einer Kante e von f nach g, für den* $\mathrm{rot}(e_f) \geq \mathrm{flex}(e)$ *gilt.*

Definition 2.6 (Ungültiger Kreis und gültiger Kreis)**.** *Ein ungültiger Kreis ist ein Kreis, der mindestens einen ungültigen Pfeil enthält. Ein gültiger Kreis ist ein Kreis, der keinen ungültigen Pfeil enthält.*

Im Allgemeinen kann der überschüssige Fluss einer Facette auch über einen inzidenten Knoten abfließen, falls dieser in der Facette einen Winkel von mindestens 180° hat. Da im Allgemeinen aber nicht garantiert werden kann, dass ein solcher Knoten in dieser Facette existiert, beschränken wir uns bei unserer Untersuchung darauf, Kanten zu glätten und zu knicken, um ein möglichst allgemeines Ergebnis zu erhalten.

3. Graphen mit einem einzelnen Kreuzungsknoten

Wir betrachten zunächst np-flex mit globaler Flexibilität 2. Es soll also untersucht werden, ob sich eine beliebige Planarisierung derart zeichnen lässt, dass jede Kante höchstens zwei Knicke hat. Laut Biedl und Kant [BK94] kann ein Graph immer derart gezeichnet werden, wenn die Einbettung frei gewählt werden kann, es sei denn es handelt sich um den Oktaeder. Für planare Graphen findet der dort angegebene Algorithmus auch bei fester Einbettung eine entsprechende Zeichnung. Wir können also voraussetzen, dass es für jede Planarisierung eine Orthogonale Zeichnung gibt, bei der alle kreuzungsfreien Kanten höchstens zwei Knicke haben. Es bleibt zu untersuchen, ob dies auch für die Kreuzungskanten erreicht werden kann.

3.1 Beschreibung des zugrundeliegenden Graphen

Wir betrachten also eine Planarisierung G mit genau einem Kreuzungsknoten x. Seien $a, b, u, v \in V$ diejenigen Knoten in V, die adjazent sind zum Kreuzungsknoten x, wobei sie in der Orthogonalen Zeichnung im Uhrzeigersinn um denselben in der Reihenfolge u, a, v, b angeordnet seien. Somit ergibt sich die Menge der Kreuzungskanten als $\{(a, b), (u, v)\}$. Sei die Benennenung der zu x inzidenten Facetten im Uhrzeigersinn um den Kreuzungksnoten f_1, g_1, g_2, f_2. Abbildung 3.1 zeigt die entsprechende Benennung anhand eines Beispiels.

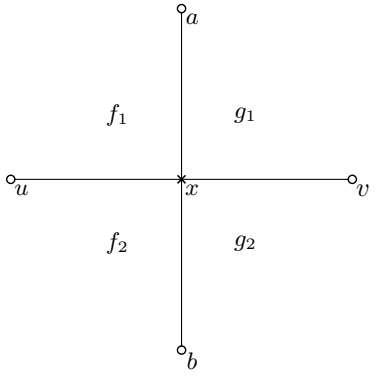


Abbildung 3.1: Anordnung der zum Kreuzungsknoten x inzidenten Kanten und Facetten.

Betrachte nun den von $V \setminus x$ induzierten Subgraphen von G. Dieser Graph ist planar und verfügt über eine Facette, auf deren Rand die Knoten u, v, a und b liegen. Wir fügen die

Kante $\{u, v\}$ hinzu. Nun können wir ohne Beschränkung der Allgemeinheit davon ausgehen, dass jede Kante in diesem Graph höchstens zwei Knicke hat. Schließlich fügen wir die Kante $\{a, b\}$ hinzu, die die Kante $\{u, v\}$ kreuzt, und erhalten so den nichtplanaren Graphen $\tilde{G}$, den wir mit dem Graphen G modelliert haben, indem wir die Stelle, in der sich die sogenannten Kreuzungskanten $\{u, v\}$ und $\{a, b\}$ als Kreuzungsknoten x modellierten. Die Kreuzungskante $\{u, v\}$ wird also in der Planarisierung ersetzt durch die Kantensegmente $\{u, x\}$ und $\{v, x\}$, die Kreuzungskante $\{a, b\}$ durch die Kantensegmente $\{a, x\}$ und $\{b, x\}$. Da die Kreuzungskante $\{u, v\}$ höchstens zwei Knicke in derselben Richtung hat, gilt $|\operatorname{rot}(\{u, x\})| + |\operatorname{rot}(\{v, x\})| \leq 2$ und $\operatorname{rot}(\{v, x\}_{g_1}) \cdot \operatorname{rot}(\{u, v\}_{f_1}) \geq 0$. Da wir bei der Modellierung keine Einschränkungen an die Beschaffenheit der Kreuzungskante $\{a, b\}$ gestellt haben, müssen diese Bedingungen für sie nicht gelten. Insbesondere können auch die Kanten $\{a, x\}$ und $\{b, x\}$ drei oder mehr Knicke aufweisen. Allerdings seien auch diese Kanten nur in eine Richtung geknickt.

3.2 Untersuchung der beiden Kreuzungskanten

Wir nehmen nun an, dass die Kreuzungskante $\{a, b\}$ mindestens drei Knicke aufweist, es gelte also $|\operatorname{rot}(\{a, x\})| + |\operatorname{rot}(\{b, x\})| \geq 3$. Wir wollen nun untersuchen, unter welchen Umständen sich die Anzahl dieser Knicke reduzieren lässt. Dazu führen wir eine Fallunterscheidung über die Beschaffenheit der zum Kreuzungsknoten x inzidenten Kanten durch und untersuchen, welche der Kanten geglättet und welche geknickt werden dürfen. Wir gehen dabei ohne Beschränkung der Allgemeinheit davon aus, dass die Kante $\{b, x\}$ höchstens so viele Knicke aufweist wie die Kante $\{a, x\}$, es gelte also $|\operatorname{rot}(\{a, x\})| \geq |\operatorname{rot}(\{b, x\})|$. Desweiteren sei die Kante $\{a, x\}$ derart geknickt, dass sie die 90°-Winkel in der Facette g_1 hat, es ergibt sich also $\operatorname{rot}(\{a, x\}_{f_1}) \leq -2$. Es ist leicht zu sehen, dass beide Eigenschaften immer garantiert werden können, sei es durch Spiegelung der Orthogonalen Zeichnung an einer der beiden Kreuzungkanten, durch Drehung der Orthogonalen Zeichnung um den Kreuzungsknoten oder durch Umbenennung der Knoten und Facetten. Abbildung 3.2 zeigt ein Beispiel für die entsprechende Vorgehensweise.

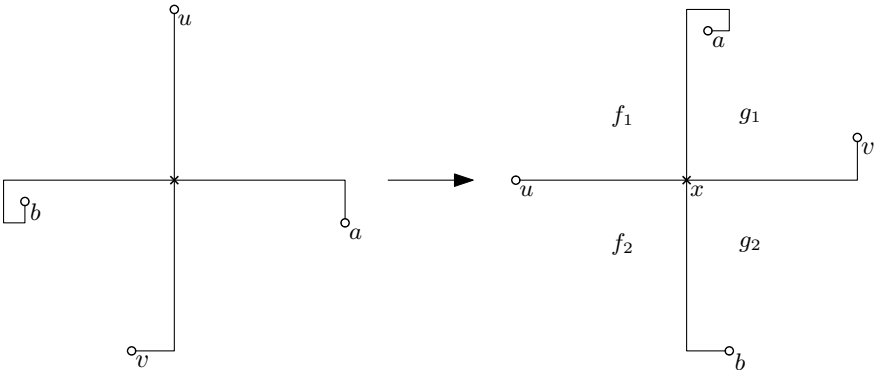


Abbildung 3.2: Normierung durch Umbenennung, Drehung und Spiegelung.

3.2.1 Beschaffenheit der Kreuzungskante $\{a, b\}$

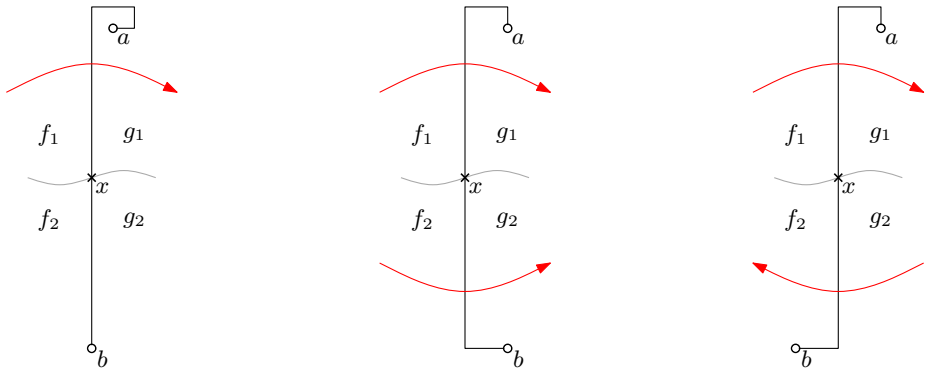
Die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ muss reduziert werden, es muss also für mindestens eine der beiden Kanten $\{a, x\}$ und $\{b, x\}$ ein reduzierender Pfeil verwendet werden, geknickt werden darf keine davon. Die Beschaffenheit der Kreuzungskante $\{a, b\}$ ergibt sich direkt aus der Beschaffenheit der Kante $\{b, x\}$. Daher ist es für eine eindeutige Kennzeichnung der unterschiedlichen Fälle ausreichend, ein Zeichen für die Beschaffenheit dieser Kante zu verwenden. Dieses Zeichen wird gleichzeitig die Kennzeichen für die Beschaffenheit der Kanten $\{u, x\}$ und $\{v, x\}$ trennen: links davon stehe $\operatorname{rot}(\{u, x\}_{f_2})$, rechts davon $\operatorname{rot}(\{v, x\}_{g_2})$. Es gibt drei Möglichkeiten:

Fall 1: $|\operatorname{rot}(\{b,x\})| = 0$ – Sämtliche Knicke der Kreuzungskante $\{a,b\}$ liegen auf der Kante $\{a,x\}$, es folgt also $\operatorname{rot}(\{a,x\}_{f_1}) \leq -3$. Somit muss die Kante $\{a,x\}$ geglättet werden. Wir kennzeichnen diesen Fall durch das Zeichen "|".

Fall 2: $\operatorname{rot}(\{b,x\}_{g_2}) \geq 1$ – Die Kreuzungskante $\{a,b\}$ ist auf beiden Seiten des Kreuzungsknoten x in dieselbe Richtung geknickt. Somit muss mindestens eine der beiden Kanten $\{a,x\}$ und $\{b,x\}$ geglättet werden. Wir kennzeichnen diesen Fall durch das Zeichen "⌊".

Fall 3: $\operatorname{rot}(\{b,x\}_{g_2}) \leq -1$ – Die Kreuzungskante $\{a,b\}$ weist Knicke in unterschiedliche Richtungen auf. Es muss mindestens eine der beiden Kanten $\{a,x\}$ und $\{b,x\}$ geglättet werden. Wir kennzeichnen diesen Fall durch das Zeichen "⌋."

Abbildung 3.3 zeigt die drei unterschiedlichen Fälle anhand eines Beispiels mit drei Knicken auf der Kreuzungskante $\{a,b\}$. Da wir zunächst über die Kreuzungskante $\{u,v\}$ keine Aussage treffen, wird diese in der Abbildung und der Kennzeichnung nur angedeutet.



(a) (1): ..|.. (b) (2): ..⌊.. (c) (3): ..⌋..

Abbildung 3.3: Die drei Fälle der Kreuzungskante $\{a,b\}$ – einer der Pfeile muss verwendet werden.

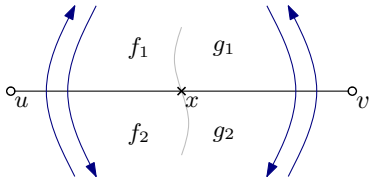
3.2.2 Beschaffenheit der Kreuzungskante $\{u,v\}$

Die Anzahl der Knicke auf der Kreuzungskante $\{u,v\}$ muss nicht reduziert werden, sie darf im Gegenteil sogar erhöht werden, solange dadurch nicht ihre Flexibilität verletzt wird. Die Kanten $\{u,x\}$ und $\{v,x\}$ dürfen also geknickt werden, falls danach weiterhin $|\operatorname{rot}(\{u,x\})| + |\operatorname{rot}(\{v,x\})| \leq 2$ gilt. Wir verzichten dabei bewusst auf weitere Einschränkungen, insbesondere darf $\{u,v\}$ auch in zwei unterschiedliche Richtungen geknickt werden. Wir dürfen die Kanten also auch derart knicken, dass die Bedingung $\operatorname{rot}(\{v,x\}_{g_1}) \cdot \operatorname{rot}(\{u,x\}_{f_1}) \geq 0$ verletzt wird. In Abhängigkeit von der Beschaffenheit der Kanten $\{u,x\}$ und $\{v,x\}$ ergeben sich also folgende Fälle:

Fall a: $|\operatorname{rot}(\{v,x\})| = |\operatorname{rot}(\{u,x\})| = 0$, d.h. keine der beiden Kanten hat einen Knick. Beide Kanten dürfen ohne Einschränkung in beide Richtungen geknickt werden. Abbildung 3.4 zeigt die gültigen Operationen. Da wir über die Kreuzungskante $\{a,b\}$ hier keine Aussage treffen, wird diese nur angedeutet.

Fall b: $|\operatorname{rot}(\{v,x\})| + |\operatorname{rot}(\{u,x\})| = 1$, d.h. eine der beiden Kanten hat einen Knick, die andere keinen. Es dürfen nicht beide Kanten gleichzeitig geknickt werden. Ansonsten sind alle Operationen erlaubt. Abbildung 3.5 zeigt die gültigen Operationen in den jeweiligen Varianten, die Art der Pfeile stellt die Abhängigkeiten dar.

Fall c: $\max(|\operatorname{rot}(\{v,x\})|, |\operatorname{rot}(\{u,x\})|) = 2$, d.h. eine der beiden Kanten hat zwei Knicke, die andere keinen. Die geknickte Kante darf geglättet werden. Ist dies der Fall, darf die andere Kante geknickt werden. Abbildung 3.6 zeigt die gültigen Operationen in den jeweiligen Varianten, die Art der Pfeile stellt die Abhängigkeiten dar.



(a) 0..0

Abbildung 3.4: Fall a - Sämtliche Pfeile dürfen verwendet werden.

(a) 1..0 (b) 0..1

(c) $-1..0$ (d) $0..-1$

Abbildung 3.5: Fall b - Nur einer der gestrichelten Pfeile darf verwendet werden.

Fall d: $|\operatorname{rot}(\{v,x\})| = |\operatorname{rot}(\{u,x\})| = 1$, d.h. jede der beiden Kanten hat genau einen Knick. Jede der beiden Kanten darf geglättet werden. Wird eine der Kanten geglättet, darf die andere Kante geknickt werden. Abbildung 3.7 zeigt die gültigen Operationen in den jeweiligen Varianten, die Art der Pfeile stellt die Abhängigkeiten dar. Man beachte, dass die beiden Varianten rechts nur dann auftreten können, wenn zuvor bereits eine Kante geknickt wurde, durch eine entsprechende Operation in einem der Fälle a, b oder c.

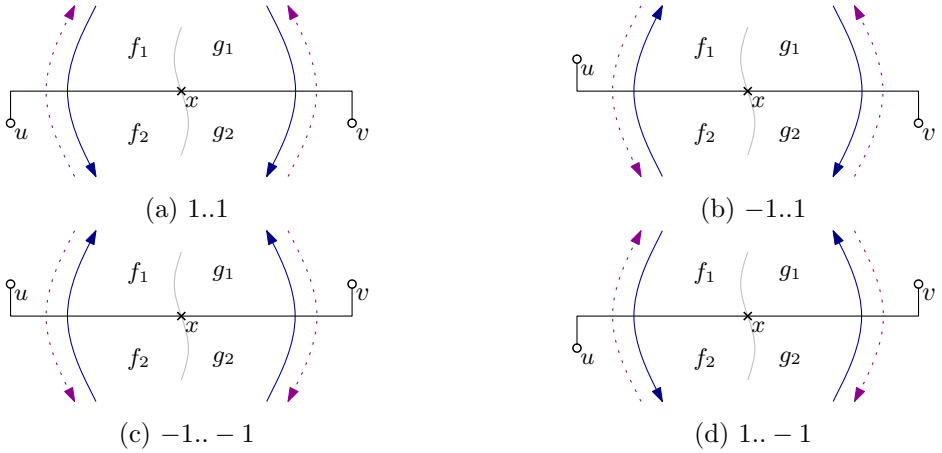
3.2.3 Gültige reduzierende Kreise

Wir wollen nun die Kreuzungskante $\{a,b\}$ glätten, ohne dabei die Kreuzungskante $\{u,v\}$ zu sehr zu knicken. Wir suchen also einen gültigen Kreis im Dualgraph zum Graph G, der mindestens einen der in den Fällen 1 bis 3 vorgestellten reduzierenden Pfeilen enthält, und

(a) 2..0 (b) 0..2

(c) $-2..0$ (d) $0..-2$

Abbildung 3.6: Fall c - Die gepunkteten Pfeile dürfen nur verwendet werden, wenn der durchgezogene Pfeil der anderen Kante ebenfalls verwendet wird.



(a) 1..1 (b) −1..1

(c) −1.. − 1 (d) 1.. − 1

Abbildung 3.7: Fall d - Die gepunkteten Pfeile dürfen nur verwendet werden, wenn der durchgezogene Pfeil der anderen Kante ebenfalls verwendet wird.

höchstens jene Pfeile aus den Fällen a bis d, die verwendet werden dürfen. Im Allgemeinen wird ein solcher Kreis nicht ausschließlich aus diesen Pfeilen bestehen, und somit nicht ausschließlich die zum Kreuzungsknoten x inzidenten Facetten umfassen. Der Verlauf des Kreises ausserhalb dieser vier Facetten wird zunächst aber nicht betrachtet. Innerhalb der vier Facetten nennen wir die erste zum Kreis zugehörige Facette den Eingang, die letzte zugehörige Facette den Ausgang. Für einen Kreis mit Eingang f und Ausgang g schreiben wir $C(f, g)$. Abbildung 3.8 zeigt ein Beispiel für den Eingang und Ausgang eines reduzierenden Kreises. Man beachte, dass es vorkommen kann, dass ein reduzierender Kreis zwei Eingänge und zwei Ausgänge hat. In diesem Fall behandeln wir den Kreis aber wie zwei getrennte Kreise.

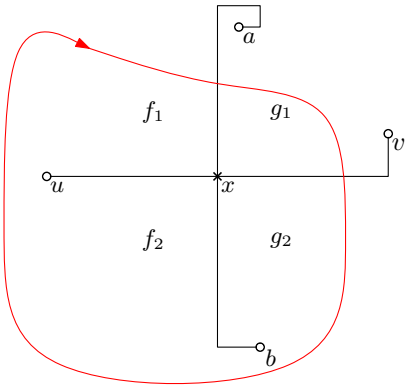


Abbildung 3.8: Ein reduzierender Kreis $C(f_1, g_2)$.

Um die Anzahl der Kreise, die wir untersuchen müssen, möglichst gering zu halten, untersuchen wir zunächst, aus welchen Pfeilen ein gültiger reduzierender Kreis bestehen kann. Die Kanten des Dualgraphen G^* können im Allgemeinen in jede der beiden Richtungen gerichtet werden. Für jede der zum Kreuzungsknoten x inzidenten Kanten gibt es also zwei mögliche Pfeile dieser Kante, die wir im Folgenden *Rechtspfeil* und *Linkspfeil* nennen.

Definition 3.1 (Rechtspfeil und Linkspfeil)**.** *Ein Rechtspfeil bzw. Linkspfeil ist ein Pfeil, für den der Kreuzungsknoten x in der Orthogonalen Zeichnung rechts bzw. links der Pfeilrichtung liegt.*

Diese Unterscheidung ist inbesondere deshalb nützlich, weil ein reduzierender Kreis nur entweder Rechtspfeile oder Linkspfeile enthalten kann, wie das folgende Lemma zeigt.

Lemma 3.2. *Ein reduzierender Kreis kann nicht sowohl Rechtspfeile als auch Linkspfeile enthalten.*

Beweis. Sei C ein reduzierender Kreis, der sowohl einen Rechtspfeil als auch einen Linkspfeil enthält. Sei der Rechtspfeil ohne Beschränkung der Allgemeinheit der Pfeil der Kante $\{a, x\}$ von f_1 nach g_1. Da ein reduzierender Kreis notwendigerweise einfach ist, kann C in jeder Facette nur einen eingehenden Pfeil und einen ausgehenden Pfeil haben. Daher kann der enthaltene Linkspfeil nicht zu den Facetten f_1 oder g_1 inzident sein, es muss sich also um den Linkspfeil der Kante $\{b, x\}$ von f_2 nach g_2 handeln. Dann muss es also im Kreis C einen gerichteten Pfad von der Facette g_1 zur Facette f_2 geben, und einen gerichteten Pfad von der Facette g_2 zur Facette f_1. Abbildung 3.9 veranschaulicht dies.

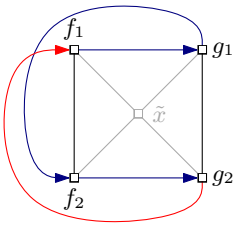


Abbildung 3.9: Ein reduzierender Kreis kann nicht sowohl Rechtspfeile als auch Linkspfeile enthalten.

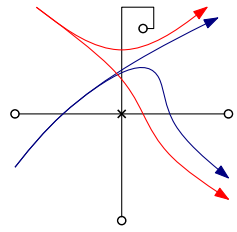
Nun ist aber bekannt, dass im Dualgraph G^* die zu den Facetten f_1, f_2, g_1 und g_2 dualen Knoten auf dem Rand derselben Facette liegen. In diese Facette lässt sich also auch ein Knoten $\tilde{x}$ einfügen, der zu den vier genannten Knoten adjazent ist. Dann bilden diese fünf Knoten aber mit den genannten Pfaden und Kanten den vollständigen Graph K_5, der bekanntlich nicht planar ist. Die beiden Pfade von g_1 nach f_2 und von g_2 nach f_1 müssen sich also kreuzen, woraus folgt dass ein solcher einfacher Kreis C nicht exisitieren kann. □

Somit können wir die reduzierenden Kreise danach kategorisieren, ob sie Rechtspfeile oder Linkspfeile enthalten.

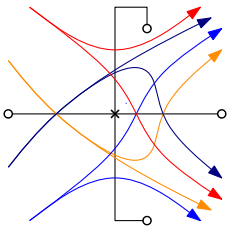
Definition 3.3 (Rechtskreis und Linkskreis)**.** *Ein reduzierender Kreis heisst Rechtskreis bzw. Linkskreis, falls er einen Rechtspfeil bzw. Linkspfeil enthält.*

3.2.4 Fallaufzählung

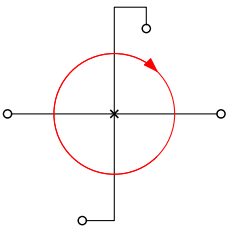
Die Kombination der möglichen Fälle für die Kreuzungskanten $\{a, b\}$ und $\{u, v\}$ ergibt die in den Abbildungen 3.10, 3.11 und 3.12 aufgeführten unterschiedlichen Varianten. Die Einschränkungen an die zu verwendenden Pfeile und reduzierenden Kreise ergeben die ebenfalls dort aufgeführten möglichen Kreise. Anschließend werden wir untersuchen, unter welchen Umständen die gefundenen Kreise existieren. Dabei lassen sich mehrere Fällen mit derselben Vorgehensweise lösen, wie sich im Verlauf des Kapitels zeigen wird. Dementsprechend ergibt sich eine Gruppierung der unterschiedlichen Fälle nach den Lösungsansätzen. Zur Erhöhung der Übersichtlichkeit ist die entsprechende Nummer der Gruppe, zu der ein Fall gehört, unter den einzelnen Abbildungen angegeben, gemeinsam mit der Kennzeichnung des Falls.



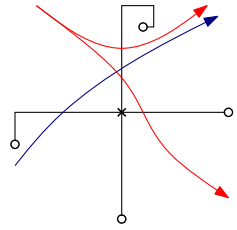
(a) (1a): $0|0$, Gruppe B



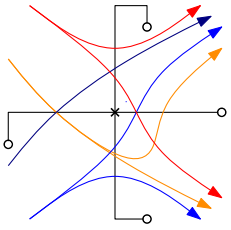
(b) (2a): $0\lfloor 0$, Gruppe B



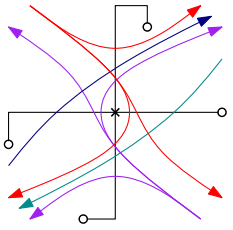
(c) (3a): $0\rfloor 0$, Gruppe A



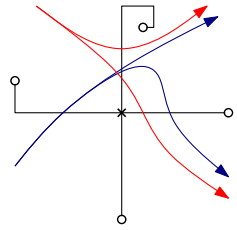
(d) (1b): $1|0$, Gruppe D



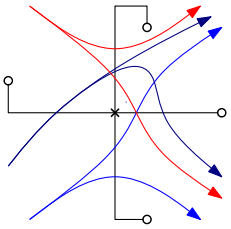
(e) (2b): $1\lfloor 0$, Gruppe B



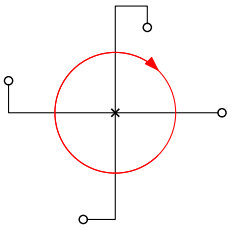
(f) (3b): $1\rfloor 0$, Gruppe E



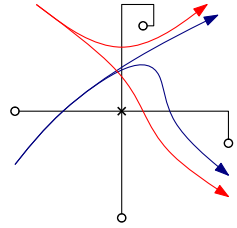
(g) (1b): $-1|0$, Gruppe B



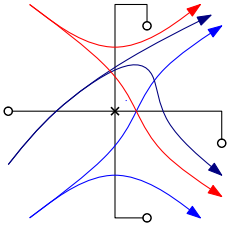
(h) (2b): $-1\lfloor 0$, Gruppe B



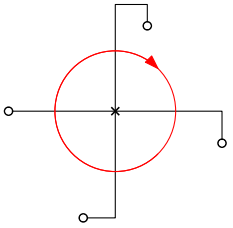
(i) (3b): $-1\rfloor 0$, Gruppe A



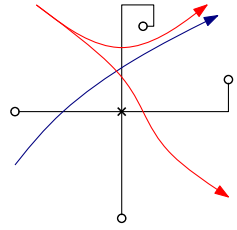
(j) (1b): $0|1$, Gruppe B



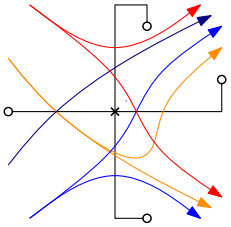
(k) (2b): $0\lfloor 1$, Gruppe B



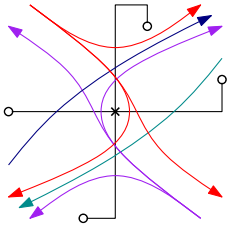
(l) (3b): $0\rfloor 1$, Gruppe A



(m) (1b): $0|-1$, Gruppe D

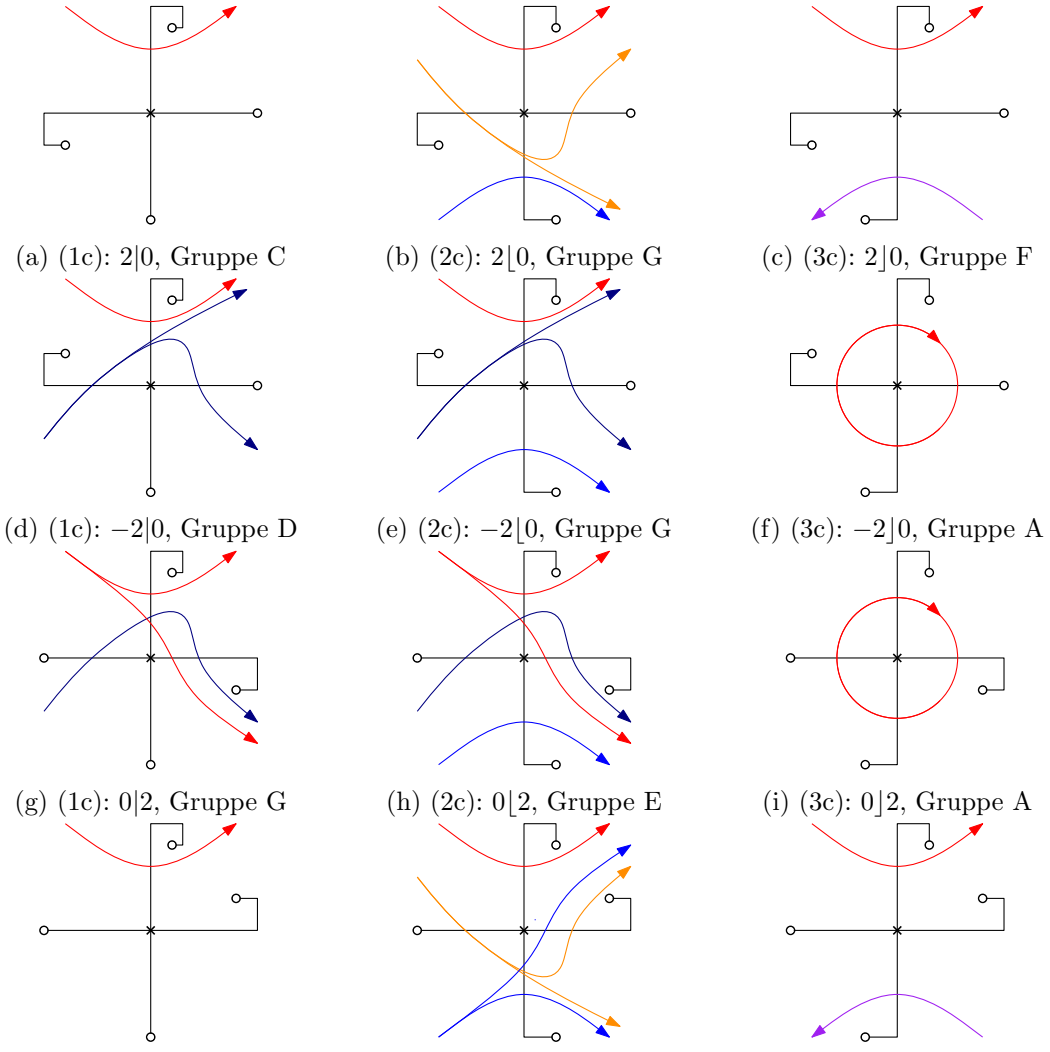


(n) (2b): $0\lfloor -1$, Gruppe B



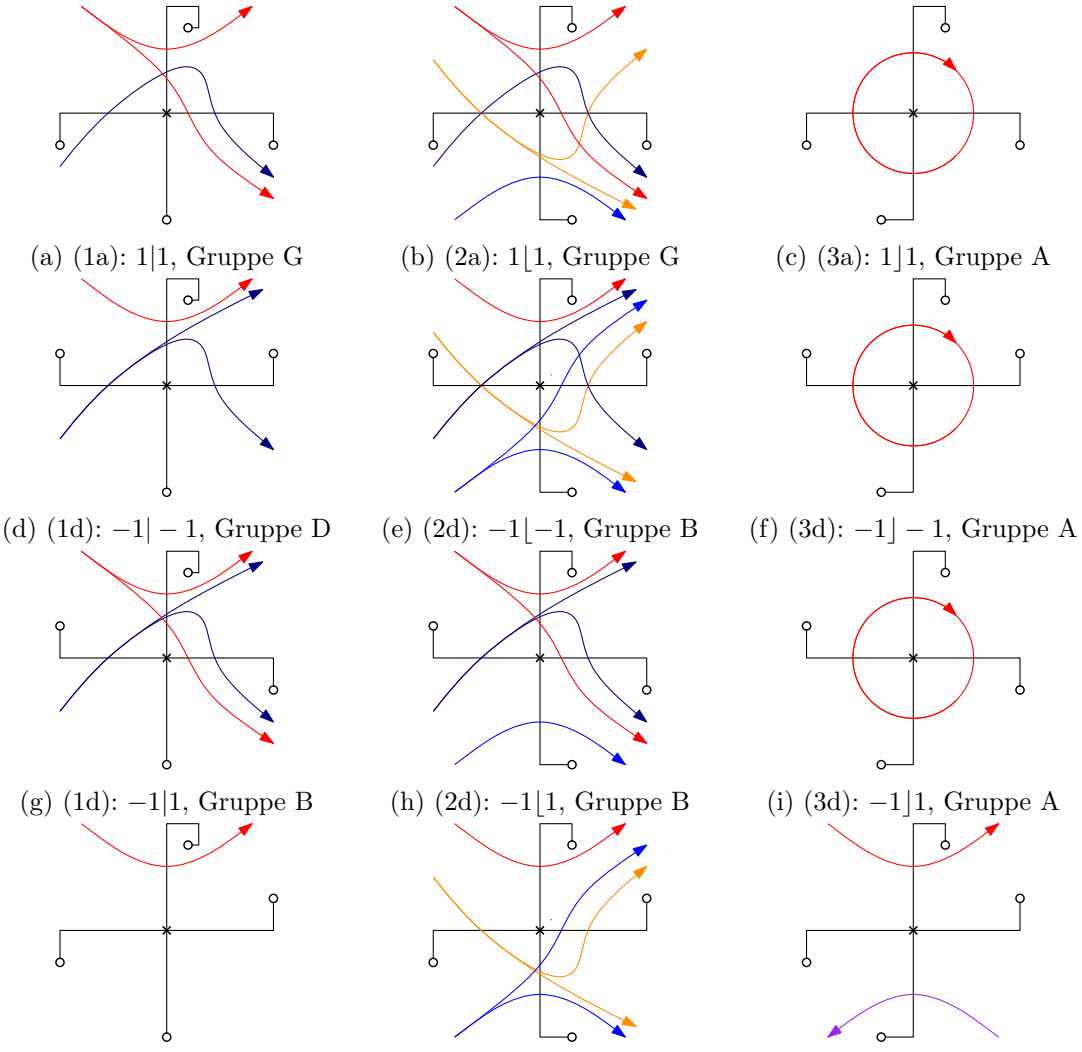
(o) (3b): $0\rfloor -1$, Gruppe E

Abbildung 3.10: Aufzählung der unterschiedlichen Varianten der Fälle a und b.



(a) (1c): $2|0$, Gruppe C
(b) (2c): $2\lfloor 0$, Gruppe G
(c) (3c): $2\rfloor 0$, Gruppe F
(d) (1c): $-2|0$, Gruppe D
(e) (2c): $-2\lfloor 0$, Gruppe G
(f) (3c): $-2\rfloor 0$, Gruppe A
(g) (1c): $0|2$, Gruppe G
(h) (2c): $0\lfloor 2$, Gruppe E
(i) (3c): $0\rfloor 2$, Gruppe A
(j) (1c): $0|-2$, Gruppe F
(k) (2c): $0\lfloor -2$, Gruppe B
(l) (3c): $0\rfloor -2$, Gruppe F

Abbildung 3.11: Aufzählung der unterschiedlichen Varianten des Falls c.



(a) (1a): $1|1$, Gruppe G
(b) (2a): $1\lfloor 1$, Gruppe G
(c) (3a): $1\rfloor 1$, Gruppe A
(d) (1d): $-1|-1$, Gruppe D
(e) (2d): $-1\lfloor -1$, Gruppe B
(f) (3d): $-1\rfloor -1$, Gruppe A
(g) (1d): $-1|1$, Gruppe B
(h) (2d): $-1\lfloor 1$, Gruppe B
(i) (3d): $-1\rfloor 1$, Gruppe A
(j) (1d): $1|-1$, Gruppe F
(k) (2d): $1\lfloor -1$, Gruppe B
(l) (3d): $1\rfloor -1$, Gruppe F

Abbildung 3.12: Aufzählung der unterschiedlichen Varianten des Falls d.

3.3 Lösung der einzelnen Gruppen

3.3.1 Gruppe A

Varianten: $0\rfloor 0$, $-1\rfloor 0$, $0\rfloor 1$, $-2\rfloor 0$, $0\rfloor -2$, $1\rfloor 1$, $-1\rfloor -1$, $-1\rfloor 1$
Der (hier eindeutige) gültige reduzierende Kreis enthält ausschließlich Pfeile der zum Kreuzungsknoten x inzidenten Kanten. Somit weist er weder Eingänge noch Ausgänge auf. Da dieser Kreis unabhängig von der Beschaffenheit der anderen Kanten des Graphen G stets genutzt werden kann, um die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren, ist dies in diesen Varianten stets möglich.

3.3.2 Gruppe B

Varianten: $0|0$, $-1|0$, $0|1$, $-1|1$, $0\lfloor 0$, $1\lfloor 0$, $-1\lfloor 0$, $0\lfloor -1$, $0\lfloor -2$, $-1\lfloor -1$, $-1\lfloor 1$, $1\lfloor -1$
Laut Biedl und Kant [BK94] lässt sich in jeder Orthogonalen Zeichnung eines planaren Graphen die Anzahl der Knicke auf jeder Kante, die nicht auf der äußeren Facette liegt, auf zwei reduzieren. Daher gibt es in jedem planaren Graphen zu jeder Kante mit mehr als zwei Knicken stets einen gültigen reduzierenden Kreis. Wir betrachten den Graphen, den man durch Entfernen der Kreuzungskante $\{u, v\}$ erhält. Wir entfernen also den Kreuzungsknoten x sowie alle dazu inzidenten Kanten und fügen dafür die Kreuzungskante $\{a, b\}$ (entsprechend der Beschaffenheit der Kanten $\{a, x\}$ und $\{b, x\}$) ein. Dadurch werden die beiden Facetten f_1 und f_2 zusammengefasst zu einer Facette f, die beiden Facetten g_1 und g_2 zu einer Facette g. Abbildung 3.13 veranschaulicht die Vorgehensweise am Beispiel der Variante $-1|0$.

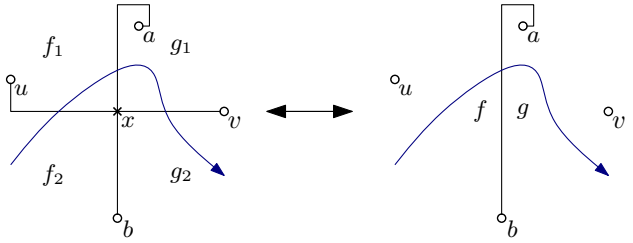


Abbildung 3.13: Ein gültiger reduzierender Kreis sowie der entsprechende Kreis im Graphen ohne die Kreuzungskante $\{u, v\}$.

Da dieser Graph planar ist, muss es in ihm einen gültigen reduzierenden Kreis der Kreuzungskante $\{a, b\}$ geben. Dieser Kreis muss als Eingang die Facette f haben und als Ausgang die Facette g. Nun erhält man aus diesem Kreis einen geeigneten gültigen reduzierenden Kreis im ursprünglichen Graphen, indem man den Pfeil der Kreuzungskante $\{a, b\}$ ersetzt durch ein bis drei geeignete Pfeile der zum Kreuzungsknoten x inzidenten Kanten. Dieser Kreis hat folglich als Eingang eine der beiden Facetten f_1, f_2 und als Ausgang eine der beiden Facetten g_1, g_2. Alle vier Kreise, die dieser Eigenschaft genügen, sind in sämtlichen Varianten der Gruppe B erlaubt. Somit ist es in diesen Varianten stets möglich, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren.

3.3.3 Gruppe C

Variante: $2|0$
Da die Kante $\{a, x\}$ mehr als zwei Knicke hat, gibt es stets einen gültigen reduzierenden Kreis dieser Kante. Dabei ist jedoch nicht garantiert, dass dieser Kreis nicht auch Pfeile der anderen zum Kreuzungsknoten x inzidenten Kanten enthält. Wir suchen also stattdessen einen gültigen reduzierenden Kreis der Kante $\{a, x\}$, der keinen dieser Pfeile enthält, also einen Kreis $C(f_1, g_1)$. Der gesuchte Kreis im Dualgraph G^* entspricht dann einem Schnitt im Primalgraph G, der den Knoten a von den drei Knoten u, v und b trennt. Genauer

gesagt soll a (sprich die zum Knoten a duale Facette) links des Kreises liegen, während die anderen drei zum Kreuzungsknoten x adjazenten Knoten (sprich die zu ihnen dualen Facetten) rechts davon liegen. Abbildung 3.14 zeigt zwei mögliche Kreise am Beispiel der Variante 2|0.

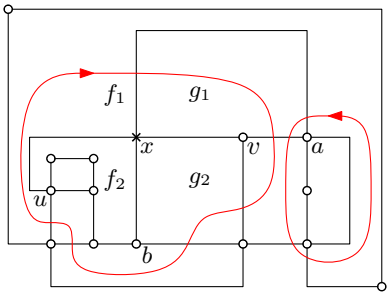


Abbildung 3.14: Zwei mögliche gültige reduzierende Rechtskreise $C(f_1, g_1)$. In beiden Fällen liegt a links, u, v und b liegen rechts des Kreises.

Wir untersuchen also, unter welchen Voraussetzungen ein solcher reduzierender Kreis ein gültiger Kreis ist. Da wir diese Vorgehensweise öfter anwenden werden, verallgemeinern wir zunächst. Sei $z \in \{a, b, u, v\}$ ein zum Kreuzungsknoten x adjazenter Knoten mit $|\operatorname{rot}(\{z, x\})| > 0$. Sei die Menge $\{a, b, u, v\}$ aufgeteilt in zwei nichtleere echte Teilmengen R und L, wobei $z \in R$ falls der reduzierende Pfeil der Kante $\{z, x\}$ ein Linkspfeil ist und $z \in L$, falls er ein Rechtspfeil ist. Betrachte nun die Menge $\mathcal{C}$ aller reduzierender Kreise der Kante $\{z, x\}$, für die sämtliche Knoten der Teilmenge R rechts des Kreises liegen und sämtliche Knoten der Teilmenge L links des Kreises. Abbildung 3.15 veranschaulicht die Aufteilung am Beispiel der Variante 0⌊1.

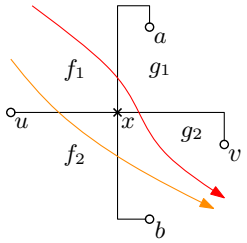


Abbildung 3.15: Sowohl der Rechtskreis als auch der Linkskreis $C(f_1, g_2)$ haben $R = \{b, u\}$ und $L = \{a, v\}$.

Betrachte nun für jedes Paar $\{r, l\}$ mit $r \in R$ und $l \in L$ die Menge der Pfade von l nach r im Graph G. Jeder Kreis in $\mathcal{C}$ muss mindestens einen Pfeil einer Kante auf jedem dieser Pfade enthalten. Falls es nun also einen Pfad in G gibt, auf dem die Pfeile aller Kanten ungültige Pfeile sind, so kann es keinen gültigen Kreis in $\mathcal{C}$ geben. Wir sprechen von einem *unknickbaren Pfad*.

Definition 3.4 (Unknickbarer Pfad)**.** *Ein unknickbarer Pfad ist ein gerichteter Pfad, auf dem jede Kante zwei Knicke hat, wobei die* 90°*-Winkel der Knicke auf der rechten Seite des Pfades liegen.*

Wir zeigen, dass auch die Umkehrung gilt.

Lemma 3.5. *Ist jeder Kreis in $\mathcal{C}$ ein ungültiger Kreis, so gibt es einen unknickbaren Pfad von einem Knoten $l \in L$ zu einem Knoten $r \in R$.*

Beweis. Betrachte die Menge $\tilde{L}$ derjenigen Knoten, die von den Knoten der Menge L aus durch unknickbare Pfade zu erreichen sind. Falls nun keiner der Knoten der Menge R in

der Menge $\tilde{L}$ enthalten ist, gibt es offenbar einen Schnitt, der die Mengen R und $\tilde{L}$ (und somit R und L) voneinander trennt. Betrachte nun die Menge jener Kanten, für die genau einer der beiden inzidenten Knoten in der Menge $\tilde{L}$ liegt. Die Pfeile dieser Kanten bilden offensichtlich einen gültigen Kreis in $\mathcal{C}$, was der Voraussetzung widerspricht. □

Dies führt direkt zu einer hinreichenden Bedingung für die Existenz des gesuchten gültigen reduzierenden Kreises:

Korollar 3.6. *Falls für jedes Paar $\{r, l\}$ mit $r \in R$ und $l \in L$ die Menge der Pfade von l nach r keinen unknickbaren Pfad enthält, gibt es einen gültigen Kreis in $\mathcal{C}$.*

Ob es einen gültigen reduzierenden Kreis in $\mathcal{C}$ gibt oder nicht, hängt also davon ab, ob es einen unknickbaren Pfad von L nach R gibt. Wir werden im folgenden zeigen, dass die Existenz eines solchen unknickbaren Pfades wiederum von den Rotationen der zum Kreuzungsknoten x inzidenten Kanten abhängig ist. Da diese uns in den einzelnen Fällen bekannt sind, können wir damit in jedem der Fälle eine entsprechende Aussage treffen.

Wir untersuchen nun, welche konkrete Werte die Rotationen der Kanten $\{l, x\}$ und $\{r, x\}$ (mit $l \in L$ und $r \in R$) haben müssen, damit es einen unknickbaren Pfad $\pi_{l,r}$ geben kann. Wir nehmen dazu an, es gäbe diesen unknickbaren Pfad $\pi_{l,r}$. Betrachte den Subgraph des Graphen G, der von diesem Pfad $\pi_{l,r}$ sowie den zum Kreuzungsknoten x inzidenten Kanten induziert wird. In diesem Graph gibt es nur eine innere Facette sowie die äußere Facette, wobei der Pfad $\pi_{l,r}$ sowie der Pfad $\pi^x_{r,l}$ über den Kreuzungsknoten x auf dem Rand beider Facetten liegen. Abbildung 3.16 zeigt ein entsprechendes Beispiel.

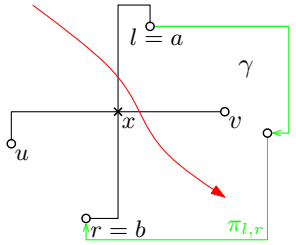


Abbildung 3.16: Der vom unknickbaren Pfad $\pi_{l,r}$ sowie den zum Kreuzungsknoten x inzidenten Kanten induzierte Subgraph anhand der Variante $1\rfloor 0$ mit $l = a$ und $r = b$.

Im Folgenden werden wir die Rotationen der Pfade $\pi_{l,r}$ und $\pi^x_{r,l}$ untersuchen. Wir geben dabei die Rotationen in jener Facette an, die rechts der beiden Pfade liegt. Diese Facette nennen wir im Folgenden γ. Abhängig von der Lage des Kreuzungsknotens x in Relation zum Pfad $\pi_{l,r}$ ist γ die innere oder die äußere Facette. Zur Unterscheidung führen wir analog zu den Begriffen Rechtspfeil und Rechtskreis den Begriff *Rechtspfad* ein.

Definition 3.7 (Rechtspfad und Linkspfad)**.** *Ein Rechtspfad bzw. Linkspfad ist ein unknickbarer Pfad, für den der Kreuzungsknoten x in der Orthogonalen Zeichnung rechts bzw. links des Pfades liegt.*

Ist $\pi_{l,r}$ ein Rechtspfad, so ist γ die innere Facette. Ist $\pi_{l,r}$ hingegen ein Linkspfad, so ist γ die äußere Facette. Abbildung 3.16 zeigte einen Rechtspfad und somit γ als innere Facette. Abbildung 3.17 hingegen zeigt ein Beispiel für einen Linkspfad, so dass γ die äußere Facette ist.

Wir berechnen nun, welche Rotation der Pfad $\pi^x_{r,l}$ haben muss, unter der Voraussetzung dass es den unknickbaren Pfad $\pi_{l,r}$ gibt. Dazu bestimmen wir die Rotation der Facette γ.

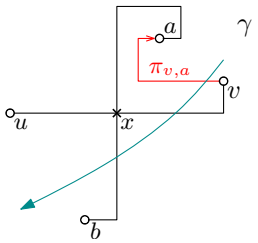


Abbildung 3.17: Rechts des Linkspfades $\pi_{v,a}$ befindet sich im entsprechenden induzierten Subgraph der Variante $0\rfloor - 1$ die äußere Facette γ.

Diese setzt sich zusammen aus den Rotationen der Pfade $\pi_{l,r}$ und $\pi^x_{r,l}$ sowie den Rotationen der Knoten r und l, an denen die beiden Pfade miteinander verknüpft sind.

Sei m die Anzahl der Kanten uaf dem Pfad $\pi_{l,r}$. Da dieser Pfad unknickbar ist, ist die Rotation jeder dieser m Kanten 2. Die tragen somit $2m$ zur Rotation bei. Auf dem Rand der Facette γ liegen dann $m+2$ Knoten, wobei die Rotation des Knoten x vorgegeben ist.

Wir berechnen die Rotationen der restlichen $m+1$ Knoten. Seien z_1 und z_2 die von l und r verschiedenen zum Kreuzungsknoten x inzidenten Knoten, also $\{a, b, u, v\} = \{l, r, z_1, z_2\}$. Liegt innerhalb der Facette γ einer der Knoten z_1, z_2, so muss es in G einen Pfad von $\pi_{l,r}$ zu diesem Knoten geben, damit die zum Kreuzungsknoten x inzidenten Facetten paarweise verschieden sind. Folglich muss einer der $m+1$ Knoten in der Facette γ einen Winkel von mindestens 180° haben. Umgekehrt muss dieser Winkel auch dann eingehalten werden, wenn γ die äußere Facette ist und keiner der beiden Knoten z_1, z_2 innerhalb von γ liegt, denn andernfalls wäre die Facette γ auch in G die äußere Facette. In beiden Fällen muss also einer der $m+1$ Knoten in γ die Rotation 0 oder -1 haben. Abbildung 3.18 veranschaulicht die beiden Fälle.

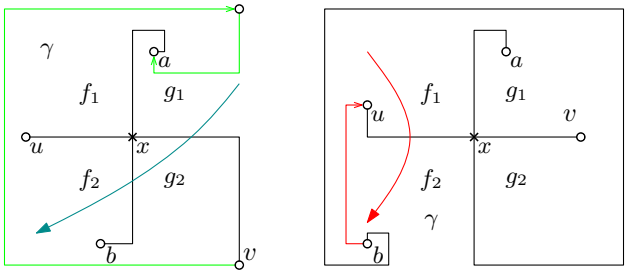


Abbildung 3.18: Die Knoten auf dem Rand der Facette γ haben ausschließlich 90°-Winkel. Daher können links die Facetten f_1 und g_2 nicht voneinander getrennt sein, während rechts die Facette f_2 die äußere Facette ist.

Analog darf in einem der $m+1$ Knoten der Winkel höchstens 180° betragen, falls mindestens einer der beiden Knoten z_1, z_2 ausserhalb der Facette γ liegt. Dasselbe gilt, falls es sich bei γ um eine innere Facette handelt und keiner der beiden Knoten z_1, z_2 ausserhalb liegt. In diesen beiden Fällen ergibt sich für einen der Knoten also eine Rotation von 0 oder 1. Abbildung 3.19 veranschaulicht beide Fälle.

Wir fassen diese vier Bedingungen zusammen. Vereinfacht ausgedrückt muss es mindestens einen Knoten geben, der zum Inneren bzw. zum Äußeren der Facette eine freie Stelle hat, falls dort einer der Knoten z_1, z_2 liegt oder falls dort die äußere Facette ist. Zur Formalisierung sei α die Rotation dieses dedizierten Knotens. Ist γ die innere Facette und liegt innerhalb dieser Facette weder z_1 noch z_2, so folgt $\alpha \in [0, 1]$. Ist γ die äußere Facette und liegt ausserhalb weder z_1 noch z_2, so folgt $\alpha \in [-1, 0]$. In allen anderen Fällen muss

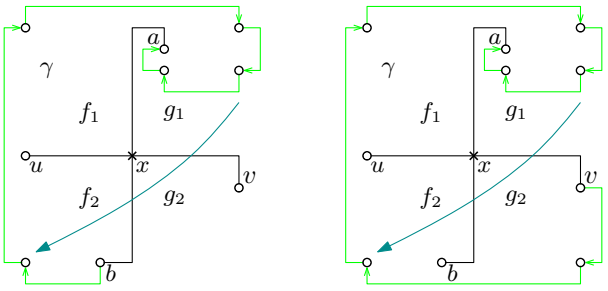


Abbildung 3.19: Die Knoten auf dem Rand der Facette γ haben ausschließlich 270°-Winkel. Daher können links die Facetten g_1 und g_2 nicht voneinander getrennt sein, während rechts die Facette g_1 die äußere Facette ist.

sowohl innerhalb der Facette γ als auch ausserhalb ein 180°-Winkel liegen, so dass $\alpha = 0$ folgt.

Alle anderen Knoten, die auf dem Rand der Facette γ liegen, dürfen einen beliebigen Winkel haben, so dass die Rotation für jeden dieser Knoten $\in [-1, 1]$ liegt. Insgesamt tragen diese m Knoten also $[-m, m]$ zur Rotation bei. Gemeinsam mit dem oben ermittelten Wert α sowie den Rotationen der Kanten auf dem Pfad $\pi_{l,r}$ ergibt sich also: $\text{rot}(\pi^x_{r,l}) + 2m + [-m, m] + \alpha = \pm 4$, mit $+4$ falls γ die innere Facette ist und -4 falls γ die äußere Facette ist. Wir erhalten das folgende Korollar:

Korollar 3.8 (Rotation des Pfades $\pi^x_{r,l}$)**.** *Sei $\pi_{l,r}$ ein unknickbarer Pfad und m die Anzahl der Kanten auf diesem Pfad. Dann gilt:*

$$\text{rot}(\pi^x_{r,l}) \in \begin{cases} [3-3m, 4-m], \textit{ falls } \pi_{l,r} \textit{ Rechtspfad und weder } z_1 \textit{ noch } z_2 \textit{ rechts davon.} \\ [4-3m, 4-m], \textit{ falls } \pi_{l,r} \textit{ Rechtspfad und } z_1 \textit{ und/oder } z_2 \textit{ rechts davon.} \\ [-4-3m, -4-m], \textit{ falls } \pi_{l,r} \textit{ Linkspfad und } z_1 \textit{ und/oder } z_2 \textit{ rechts davon.} \\ [-4-3m, -3-m], \textit{ falls } \pi_{l,r} \textit{ Linkspfad und weder } z_1 \textit{ noch } z_2 \textit{ rechts davon.} \end{cases}$$

Wir erhalten direkt eine notwendige Bedingung für die Existenz eines unknickbaren Pfades $\pi_{l,r}$ der Länge m.

Korollar 3.9 (Rotations-Korollar)**.** *Sei $m = 4 - \pi^x_{r,l}$. Dann kann es keinen unknickbaren Pfad $\pi_{l,r}$ geben, der mehr als m Kanten hat.*

Es gibt also keinen unknickbaren Pfad $\pi_{l,r}$, falls $m \leq 0$ ist. Ebenso gibt es diesen Pfad nicht, falls $m \leq 1$ ist und $\{l, r\} = \{a, b\}$ oder $\{l, r\} = \{u, v\}$, denn in diesem Fall muss der Pfad $\pi_{l,r}$ aus mindestens 2 Kanten bestehen.

Man beachte, dass sich das Rotations-Korollar aus der oberen Schranke für die Rechtspfade ergibt. Die notwendige Bedingung für die Existenz eines Linkspfades ist entsprechend schärfer. Abbildung 3.20 gibt eine Übersicht darüber, bei welchen Rotationen des Pfades $\pi^x_{r,l}$ es einen unknickbaren Pfad $\pi_{l,r}$ geben kann, nach der relativen Lage der Knoten l und r zueinander.

Mit Hilfe von Abbildung 3.20 können wir nun untersuchen, ob es in Variante 2|0 immer einen gültigen reduzierenden Kreis geben muss. Es gilt $L = \{a\}$ und $R = \{u, v, b\}$. Damit der gesuchte reduzierende Kreis gültig ist, darf also keiner der drei Pfade $\pi_{a,u}$, $\pi_{a,v}$ und $\pi_{a,b}$ unknickbar sein. Dies ist dann garantiert, wenn gilt $\text{rot}(\pi^x_{v,a}) > 3$, $\text{rot}(\pi^x_{b,a}) > 2$ und $\text{rot}(\pi^x_{u,a}) > 3$. Abbildung 3.21 zeigt, dass alle drei Bedingungen stets erfüllt sind. Es gibt also in der Variante 2|0 immer einen gültigen Kreis $C(f_1, g_1)$. Somit ist es in dieser Variante stets möglich, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren.

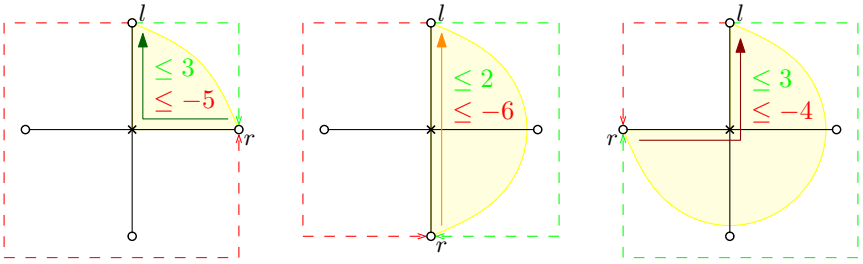


Abbildung 3.20: Ist die Rotation rechts des Pfades $\pi^x_{r,l}$ höchstens die in grün bzw. rot angegebene Zahl, so ist die notwendige Bedingung für die Existenz eines Rechtspfades bzw. Linkspfades $\pi_{l,r}$ erfüllt.

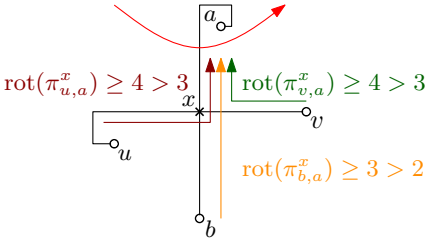


Abbildung 3.21: In der Variante $2|0$ kann es keine unknickbaren Pfade geben, da die Pfade über den Kreuzungsknoten x den entsprechenden Bedingungen genügen.

Leider ist Variante $2|0$ die einzige Variante, in der $\text{rot}(\pi^x_{u,a}) > 3$ gilt. In allen anderen Varianten kann es also stets einen unknickbaren Pfad $\pi_{a,u}$ geben, weswegen dort die Existenz eines gültigen reduzierenden Kreises $C(f_1, g_1)$ nicht garantiert werden kann. Wir gehen daher im Folgenden davon aus, dass es diesen gültigen reduzierenden Kreis tatsächlich nicht gibt und untersuchen in Abhängigkeit davon, ob es einen anderen gültigen reduzierenden Kreis gibt. Die Varianten $0|-2$ und $1|-1$ stellen wir dabei zunächst zurück, da in jenen Fällen der Kreis $C(f_1, g_1)$ der einzige mögliche reduzierende Kreis ist. Wir werden sie in Gruppe F betrachten.

3.3.4 Gruppe D

Varianten: $1|0$, $0|-1$, $-2|0$, $-1|-1$
Wir setzen nun wie gesagt voraus, dass es keinen gültigen reduzierenden Kreis $C(f_1, g_1)$ gibt. Dies bedeutet aber nicht automatisch, dass es auch tatsächlich den unknickbaren Pfad $\pi_{a,u}$ gibt, da ja alternativ auch die unknickbaren Pfade $\pi_{a,b}$ oder $\pi_{a,v}$ existieren könnten. Damit wir also die Existenz des Pfades $\pi_{a,u}$ voraussetzen können, müssen wir zusätzlich fordern, dass es die Pfade $\pi_{a,b}$ und $\pi_{a,v}$ nicht geben kann. Laut Rotations-Korollar ist dies dann der Fall, wenn $\text{rot}(\pi^x_{b,a}) > 2$ und $\text{rot}(\pi^x_{v,a}) > 3$ gelten. Um die Varianten der Gruppe D zu erhalten überprüfen wir, welche Varianten diese beiden Bedingungen erfüllen. Die erste Bedingung wird von sämtlichen Varianten $x|y$ und $x\lfloor y$ erfüllt, die zweite hingegen nur von den Varianten $x|y$ mit $y \leq 0$ sowie $x\lfloor y$ und $x\rfloor y$ mit $y \leq -1$. Lässt man dabei jene Varianten aussen vor, die bereits in den Gruppen A bis C gelöst wurden sowie die Varianten, die in Gruppe F zurückgestellt wurden, so erhält man die vier Varianten $1|0$, $0|-1$, $-2|0$ und $-1|-1$.

In diesen vier Varianten kann also vorausgesetzt werden, dass (falls es keinen gültigen Kreis $C(f_1, g_1)$ geben kann) der unknickbare Pfad $\pi_{a,u}$ existiert. In allen vier dieser Varianten ist ein reduzierender Kreis $C(f_2, g_1)$, mit $L = \{a, u\}$ und $R = \{b, v\}$ möglich. Wir untersuchen also, ob es die unknickbaren Pfade $\pi_{a,b}$, $\pi_{a,v}$, $\pi_{u,b}$ und $\pi_{u,v}$ geben kann. Wir wissen bereits, dass es die unknickbaren Pfade $\pi_{a,v}$ und $\pi_{a,b}$ nicht gibt. Die Pfade $\pi_{u,b}$ und $\pi_{u,v}$ würden aber gemeinsam mit dem vorhandenen Pfad $\pi_{a,u}$ wiederum die Pfade $\pi_{a,b}$ bzw. $\pi_{a,v}$ bilden,

so dass es auch sie nicht geben kann. Abbildung 3.22 veranschaulicht das Ganze anhand der Variante $0|-1$.

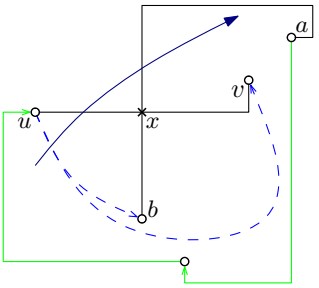


Abbildung 3.22: Gibt es in der Variante $0|-1$ den unknickbaren Pfad $\pi_{a,u}$, so kann es die unknickbaren Pfade $\pi_{u,b}$ und $\pi_{u,v}$ nicht geben, da es die unknickbaren Pfade $\pi_{a,b}$ und $\pi_{a,v}$ nicht gibt.

Es gibt also in diesen vier Varianten falls es keinen gültigen Kreis $C(f_1, g_1)$ gibt stattdessen einen gültigen Kreis $C(f_2, g_1)$. Somit ist es in diesen Varianten stets möglich, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren.

Um die Untersuchung der einzelnen Varianten zu vereinfachen, führen wir ein Schema ein: Wir untersuchen einen Pfad nach dem anderen, und halten fest, ob es diesen unknickbaren Pfad gibt ($\exists\pi$) oder ob seine Existenz ausgeschlossen werden kann ($\nexists\pi$). Sobald wir für einen reduzierenden Kreis gezeigt haben, dass es keinen entsprechenden unknickbaren Pfad geben kann, können wir folgern, dass es diesen reduzierenden Kreis stets gibt. Zusätzlich zum bereits verwendeten Rotations-Korollar führen wir dazu ein Korollar ein, das die Vorgehensweise in Gruppe D wiederspiegelt.

Korollar 3.10 (Verknüpfungs-Korollar)**.** $\exists\pi_{l,r_1} \wedge \nexists\pi_{l,r_2} \rightarrow \nexists\pi_{r_1,r_2}$.

3.3.5 Gruppe E

Variante: $0\lfloor 2$
In dieser Variante kann es neben dem Pfad $\pi_{a,u}$ auch den Pfad $\pi_{a,v}$ geben. Wenn wir also voraussetzen wollen, dass es keinen Kreis $C(f_1, g_1)$ gibt, müssen wir eine Fallunterscheidung machen, welcher dieser beiden Pfade existiert.

Es gilt also:
(a) $\nexists\pi_{a,b}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,a}) \geq 3 \rightarrow m \leq 1$
(b) $\nexists\pi_{v,b}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,v}) \geq 4 \rightarrow m \leq 0$

Fall 1: (ζ_1) $\nexists\pi_{a,u}$, (ζ_2) $\nexists\pi_{a,v}$
$(a), (\zeta_1), (\zeta_2) \Rightarrow \exists C(f_1, g_1)$

Fall 2: (α) $\exists\pi_{a,u}$ (Abbildung 3.23)
(c) $\nexists\pi_{u,b}$ - Verknüpfungs-Korollar mit $(\alpha), (a)$
$(a), (b), (c) \Rightarrow \exists C(f_2, g_2)$

Fall 3: (γ_1) $\nexists\pi_{a,u}$, (γ_2) $\exists\pi_{a,v}$
(d) $\nexists\pi_{v,u}$ - Verknüpfungs-Korollar mit $(\gamma_2), (\gamma_1)$
$(a), (b), (\gamma_1), (d) \Rightarrow \exists C(f_1, g_2)$

Es gibt also in der Variante $0\lfloor 2$ immer einen gültigen Kreis $C(f_2, g_2)$, $C(f_1, g_2)$ oder $C(f_1, g_1)$. Somit ist es in diesen Varianten stets möglich, die Anzahl der Knicke auf der

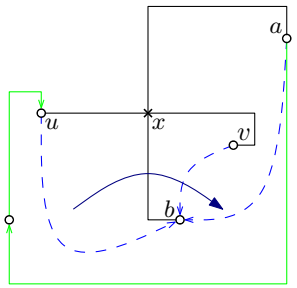


Abbildung 3.23: Gibt es in der Variante $0\rfloor 2$ den Pfad $\pi_{a,u}$, so kann es den Pfad $\pi_{u,b}$ nicht geben, da es den Pfad $\pi_{a,b}$ auf Grund der Rotation nicht gibt.

Kreuzungskante $\{a, b\}$ zu reduzieren.

Variante: $0\rfloor -1$
In dieser Variante kann es zwar nicht den Pfad $\pi_{a,v}$ geben, wohl aber den Pfad $\pi_{a,b}$.
(a) $\nexists\pi_{a,v}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,a}) \geq 4 \rightarrow m \leq 0$

Fall 1: $(\zeta_1) : \nexists\pi_{a,u}$, $(\zeta_2) : \nexists\pi_{a,b}$
$(a), (\zeta_1), (\zeta_2) \Rightarrow \exists C(f_1, g_1)$

Fall 2: (α) $\exists\pi_{a,u}$ (Abbildung 3.24)
(b) $\nexists\pi_{u,v}$ - Verknüpfungs-Korollar mit $(\alpha), (a)$
(c) $\nexists\pi_{b,v}$ - Es gilt $\text{rot}(\pi^x_{u,a}) \geq 1$ und $\text{rot}(\pi^x_{v,b}) = 1$. Somit können die Pfade $\pi_{a,u}$ und $\pi_{b,v}$ (falls es letzteren gibt) keine Linkspfade sein, sondern lediglich Rechtspfade (vergleiche Abbildung 3.20). Daher können die Knoten u und a nicht links des Pfades $\pi_{b,v}$ liegen, die Knoten b und v nicht links des Pfades $\pi_{a,u}$. Folglich müssen die beiden Pfade sich schneiden, in einem Knoten z. Dann bilden aber die Pfade $\pi_{a,z}$ und $\pi_{z,v}$ einen Pfad $\pi_{a,v}$, was laut (a) nicht möglich ist. Abbildung 3.24 veranschaulicht die Situation.
$(a), (b), (c) \Rightarrow \exists C(g_2, g_1)$

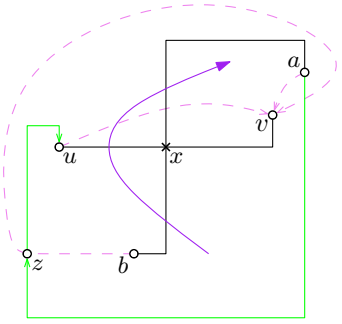


Abbildung 3.24: Gibt es in der Variante $0\rfloor -1$ den Pfad $\pi_{a,u}$, so kann es keine Pfade nach v geben, da es den Pfad $\pi_{a,v}$ nicht gibt.

Wir fassen die Beobachtung aus (c) in einem Korollar zusammen:

Korollar 3.11 (Schnitt-Korollar)**.** $\exists\pi_{l_1,r_1} \wedge (\nexists\pi_{l_1,r_2} \vee \nexists\pi_{l_2,r_1}) \rightarrow \nexists\pi_{l_2,r_2}$ *falls die Pfade* $\pi_{l_1,r_1}, \pi_{l_2,r_2}$ *sich schneidende Rechtspfade wären.*

Voraussetzung für die Nutzung des Schnitt-Korollars ist, dass die beiden sich schneidenden Pfade keine Linkspfade sein können. Zur Vereinfachung betrachten wir, bei welchen Pfaden es sich überhaupt um Linkspfade handeln kann. Laut Abbildung 3.20 muss die Rotation

des Pfades über den Kreuzungsknoten x dafür stark negativ werden können. Die Fälle, in denen dies der Fall ist, lassen sich aufzählen.

Korollar 3.12 (Linkspfad-Korollar)**.** *Die folgenden Pfade (und nur diese) können Linkspfade sein:*
Die Pfade mit $r = a$ (also $\pi_{v,a}$, $\pi_{b,a}$ und $\pi_{u,a}$) in allen Varianten.
Die Pfade mit $r = b$ und $l \neq a$ (also $\pi_{u,b}$ und $\pi_{v,b}$) in allen Varianten der Form $x\rfloor y$.
Die Pfade mit $l = b$ und $r \neq a$ (also $\pi_{b,u}$ und $\pi_{b,v}$) in allen Varianten der Form $x\lfloor y$.
Folglich müssen alle anderen unknickbaren Pfade Rechtspfade sein.

Fall 3: (β_1) $\nexists\pi_{a,u}, (\beta_2)$ $\exists\pi_{a,b}$
(d) $\nexists\pi_{u,v}$ - Schnitt-Korollar mit $(\beta_2), (a)$
(e) $\nexists\pi_{b,v}$ - Verknüpfungs-Korollar mit $(\beta_2), (a)$
$(a), (d), (e) \Rightarrow \exists C(g_2, g_1)$

Es gibt also in der Variante $0\rfloor -1$ immer einen gültigen Kreis $C(f_1, g_1)$ oder $C(g_2, g_1)$. Somit ist es in dieser Variante stets möglich, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren.

Variante: $1\rfloor 0$
In dieser Variante kann es alle drei Pfade von a geben, zudem verhindern die Rotationen keine relevanten Pfade. Daher fällt die folgende Fallunterscheidung etwas komplexer aus.

Fall 1: (ζ_1) $\nexists\pi_{a,u}, (\zeta_2)$ $\nexists\pi_{a,b}, (\zeta_3)$ $\nexists\pi_{a,v}$
$(\zeta_1), (\zeta_2), (\zeta_3) \Rightarrow \exists C(f_1, g_1)$

Fall 2: (ν_1) $\exists\pi_{a,u}, (\nu_2)$ $\exists\pi_{a,v}$
$(a_1)-(a_4)$ $\nexists\pi_{b,a}$, $\nexists\pi_{u,a}$, $\nexists\pi_{b,v}$, $\nexists\pi_{u,v}$ - Laut Rotations-Korollar mit $\text{rot}(\pi^x_{u,a}) \geq 2 \rightarrow m \leq 2$ kann der Pfad $\pi_{a,u}$ höchstens zwei Kanten haben. Da der Pfad $\pi_{v,u}$ mindestens zwei Kanten haben muss, kann der Pfad $\pi_{a,v}$ nicht Teil des Pfades $\pi_{a,u}$ sein. Entsprechend hat der Knoten a mindestens Grad 3. Abbildung 3.25 veranschaulicht die Situation. Auf Grund der Rotation des Pfades $\pi^x_{v,a}$ über den Kreuzungsknoten x müssen die beiden Knoten a und v ihre 180°- bzw. 270°-Winkel in der Facette g_1 haben. Dann kann aber keine weitere Kante von ausserhalb der Facette g_1 auf einen dieser beiden Knoten treffen, weswegen es die vier genannten Pfade nicht geben kann.
$(a_1) - (a_4) \Rightarrow \exists C(g_2, f_1)$

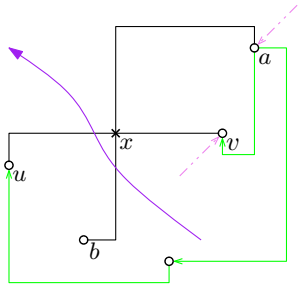


Abbildung 3.25: Gibt es in der Variante $1\rfloor 0$ die Pfade $\pi_{a,u}$ und $\pi_{a,v}$, so kann es keine Pfade nach a oder nach v geben.

Fall 3: (μ_1) $\exists\pi_{a,u}, (\mu_2)$ $\nexists\pi_{a,v}, (\mu_3)$ $\exists\pi_{a,b}$
(b) $\nexists\pi_{u,v}$ - Verknüpfungs-Korollar mit $(\mu_1), (\mu_2)$
(c) $\nexists\pi_{b,v}$ - Schnitt-Korollar mit $(\mu_1), (\mu_2)$

$(\mu_2), (b), (c) \Rightarrow \exists C(g_2, g_1)$

Fall 4: (α_1) $\exists \pi_{a,u}, (\alpha_2)$ $\nexists \pi_{a,v}, (\alpha_3)$ $\nexists \pi_{a,b}$
(d) $\nexists \pi_{u,v}$ - Verknüpfungs-Korollar mit $(\alpha_1), (\alpha_2)$
(e) $\nexists \pi_{u,b}$ - Verknüpfungs-Korollar mit $(\alpha_1), (\alpha_3)$
$(\alpha_2), (\alpha_3), (d), (e) \Rightarrow \exists C(f_2, g_1)$

Fall 5: (β_1) $\nexists \pi_{a,u}, (\beta_2) \exists \pi_{a,b}$
(f) $\nexists \pi_{b,u}$ - Verknüpfungs-Korollar mit $(\beta_2), (\beta_1)$
(g) $\nexists \pi_{v,u}$ - Schnitt-Korollar mit $(\beta_2), (\beta_1)$
$(\beta_1), (f), (g) \Rightarrow \exists C(f_1, f_2)$

Fall 6: (γ_1) $\nexists \pi_{a,u}, (\gamma_2)$ $\nexists \pi_{a,b}, (\gamma_3)$ $\exists \pi_{a,v}$
(h) $\nexists \pi_{v,u}$ - Verknüpfungs-Korollar mit $(\gamma_3), (\gamma_1)$
(j) $\nexists \pi_{v,b}$ - Verknüpfungs-Korollar mit $(\gamma_3), (\gamma_2)$
$(\gamma_1), (\gamma_2), (d), (e) \Rightarrow \exists C(f_1, g_2)$

Es gibt also in der Variante $1\rfloor 0$ immer einen der sechs oben aufgeführten gültigen Kreise. Somit ist es in dieser Variante stets möglich, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren.

3.3.6 Gruppe F

Varianten: $0| - 2$ und $1| - 1$
In diesen beiden Varianten gibt es ausser dem Kreis mit Eingang f_1 und Ausgang g_1 keinen weiteren gültigen reduzierenden Kreis. Dessen Existenz kann aber nicht garantiert werden. Stattdessen werden wir in diesen beiden Varianten die Anzahl der Knicke auf der Kante $\{v, x\}$ reduzieren, ohne dabei die Anzahl der Knicke auf den anderen zum Kreuzungsknoten x inzidenten Kanten zu erhöhen. Dadurch reduzieren wir zwar nicht die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$, wir erhalten aber die Varianten $0| - 1$ bzw. $1|0$, bei denen dies bekanntlich stets erreicht werden kann, und lösen das Problem auf diesem Umweg.

Es gilt wie in Gruppe D:
(a) $\nexists \pi_{a,v}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,a}) \geq 5 \rightarrow m \leq -1$
(b) $\nexists \pi_{a,b}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,a}) \geq 3 \rightarrow m \leq 1$

Fall 1: (ζ) $\nexists \pi_{a,u}$
$(a), (b), (\zeta) \Rightarrow \exists C(f_1, g_1)$

Fall 2: (α) $\exists \pi_{a,u}$, Abbildung 3.26
(d) $\nexists \pi_{b,v}$ - Schnitt-Korollar mit $(\alpha), (a)$
(e) $\nexists \pi_{u,v}$ - Verknüpfungs-Korollar mit $(\alpha), (a)$
$(a), (d), (e) \Rightarrow \exists C(g_2, g_1)$

Falls es also in den Varianten $0| - 2$ und $1| - 1$ keinen gültigen Kreis $C(f_1, g_1)$ gibt, so gibt es zumindest einen Kreis $C(g_2, g_1)$. Dieser reduziert in dieser Variante zwar nicht die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$, wir erhalten über ihn aber die Varianten $0| - 1$ bzw. $1|0$. Da es in diesen Varianten stets möglich ist, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren, gilt dies auch für die Varianten $0| - 2$ und $1| - 1$.

Varianten: $0\rfloor - 2$ und $1\rfloor - 1$
Diese beiden Varianten haben im Gegensatz zu den Varianten $0| - 2$ und $1| - 1$ zwar einen

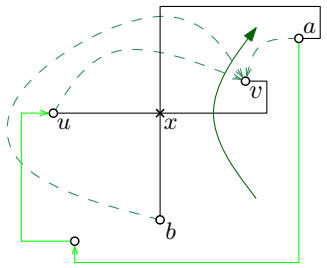


Abbildung 3.26: Gibt es in der Variante $0|-2$ den Pfad $\pi_{a,u}$, so kann es keine Pfade nach v geben. Damit existiert in diesem Fall der reduzierende Kreis, der die Variante $0|-2$ umwandelt in die stets lösbare Variante $0|-1$.

weiteren möglichen reduzierenden Kreis, doch auch dessen Existenz kann im Allgemeinen nicht garantiert werden. Daher wenden wir hier das selbe Verfahren an wie oben: wir wandeln die beiden Varianten um in die stets lösbaren Varianten $0\rfloor-1$ und $1\rfloor 0$.

Es gilt lediglich:
(a) $\nexists\pi_{a,v}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,a}) \geq 5 \rightarrow m \leq -1$

Fall 1: (ζ_1) $\nexists\pi_{a,u}, (\zeta_2)$ $\nexists\pi_{a,b}$
$(a), (\zeta_1), (\zeta_2) \Rightarrow \exists C(f_1, g_1)$

Fall 2: (α) $\exists\pi_{a,u}$
(b) $\nexists\pi_{b,v}$ - Schnitt-Korollar mit $(\alpha), (a)$
(c) $\nexists\pi_{u,v}$ - Verknüpfungs-Korollar mit $(\alpha), (a)$
$(a), (b), (c) \Rightarrow \exists C(g_2, g_1)$

Fall 3: (β_1) $\nexists\pi_{a,u}, (\beta_2)$ $\exists\pi_{a,b}$
(d) $\nexists\pi_{b,v}$ - Verknüpfungs-Korollar mit $(\beta_2), (a)$
(e) $\nexists\pi_{u,v}$ - Schnitt-Korollar mit $(\beta_2), (a)$
$(a), (d), (e) \Rightarrow \exists C(g_2, g_1)$

Falls es also in den Varianten $0\rfloor-2$ und $1\rfloor-1$ keinen gültigen Kreis $C(f_1, g_1)$ gibt, so erhalten wir zumindest die Varianten $0\lfloor-1$ bzw. $1\lfloor 0$. Da es in diesen Varianten stets möglich ist, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren, gilt dies auch für die Varianten $0\rfloor-2$ und $1\rfloor 0$.

Variante: $2\rfloor 0$
Obwohl diese Variante grundsätzlich anders aufgebaut ist als die vier anderen Varianten der Gruppe F, verwenden wir hier eine ähnliche Vorgehensweise: Wir reduzieren die Anzahl der Knicke auf der Kreuzungskante $\{u, v\}$ und erhalten so eine andere Variante, die stets lösbar ist. Diesmal verwenden wir aber die Kante $\{u, x\}$ und erhalten so die Variante $1\rfloor 0$.

Es gilt:
(a) $\nexists\pi_{b,u}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{u,b}) \geq 4 \rightarrow m \leq 0$

Fall 1: (ζ_1) $\nexists\pi_{a,u}, (\zeta_2)$ $\nexists\pi_{a,b}$
$(a), (\zeta_1), (\zeta_2) \Rightarrow \exists C(f_1, g_1)$

Fall 2: (α) $\exists\pi_{a,u}$
(b) $\nexists\pi_{b,a}$ - Schnitt-Korollar mit $(\alpha), (a)$

(c) $\not\exists \pi_{b,v}$ - Schnitt-Korollar mit $(\alpha), (a)$
$(a), (b), (c) \Rightarrow \exists C(g_2, f_2)$

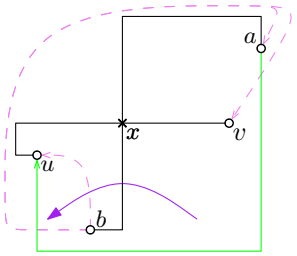


Abbildung 3.27: Gibt es in der Variante $2\rfloor 0$ den Pfad $\pi_{a,u}$, so kann es keine Pfade von b geben.

Fall 3: (β_1) $\not\exists \pi_{a,u}, (\beta_2)$ $\exists \pi_{a,b}$
(d) $\not\exists \pi_{b,u}$ - Verknüpfungs-Korollar mit $(\beta_2), (\beta_1)$
(e) $\not\exists \pi_{v,u}$ - Schnitt-Korollar mit $(\beta_2), (\beta_1)$
$(\alpha_1), (d), (e) \Rightarrow \exists C(f_1, f_2)$

Falls es also in Variante $2\rfloor 0$ weder einen gültigen Kreis $C(f_1, g_1)$ noch einen gültigen Kreis $C(g_2, f_2)$ gibt, so erhalten wir zumindest die Variante $1\rfloor 0$. Da es in dieser Variante stets möglich ist, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren, gilt dies auch für die Variante $2\rfloor 0$.

3.3.7 Gruppe G

Die fünf Varianten der Gruppe G sind mit unseren bisherigen Voraussetzungen nicht lösbar, weil es Pfade gibt, die sämtliche möglichen reduzierenden Kreise und sämtliche Kreise, die eine Umwandlung in eine andere Variante (wie in Gruppe F) verhindern. Abbildung 3.28 zeigt für jede Variante ein entsprechendes Beispiel.

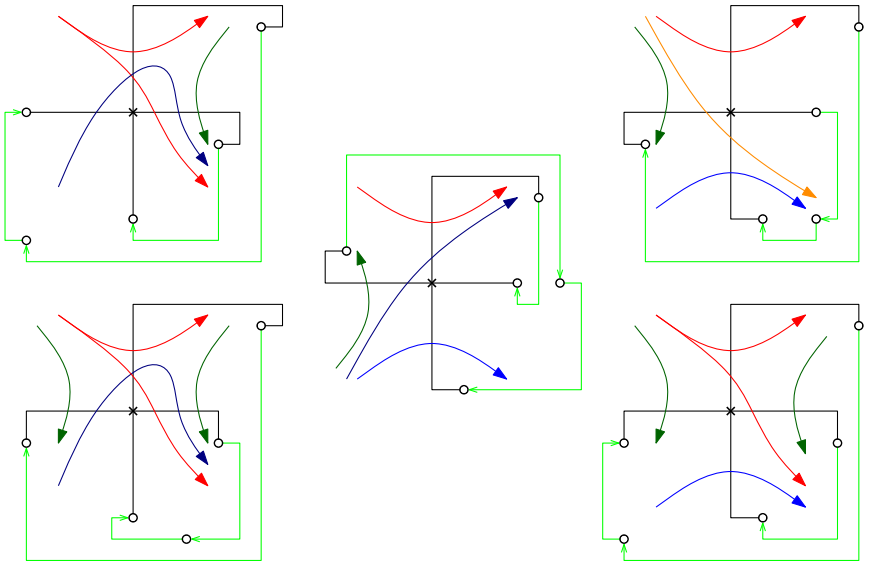


Abbildung 3.28: Die fünf Varianten der Gruppe G mit sämtlichen reduzierenden Kreisen und den unknickbaren Pfaden, die diese Kreise verhindern.

Man beachte, dass jeweils einer der in Abbildung 3.28 dargestellten unknickbaren Pfade aus nur einer einzelnen Kante besteht. Tatsächlich werden wir zeigen, dass jede der Varianten lösbar wird, wenn als zusätzliche Bedingung gefordert wird, dass jeder unknickbare

Pfad aus mindestens zwei Kanten besteht. Wir fordern also im folgenden, dass die zum Kreuzungsknoten x adjazenten Knoten untereinander paarweise nicht adjazent sind. Dies ist eine naheliegende Forderung, da in vielen Anwendungen eine Proximität dieser Knoten nicht vorkommen kann.

Varianten: $1|1$ und $2\lfloor 0$
(a) $\nexists\pi_{a,u}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{u,a}) \geq 3 \rightarrow m \leq 1$
(b) $\nexists\pi_{a,b}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,a}) \geq 3 \rightarrow m \leq 1$
(c) $\nexists\pi_{a,v}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{v,a}) \geq 3 \rightarrow m \leq 1$
$(a), (b), (c) \Rightarrow \exists C(f_1, g_1)$

Es gibt also in den Varianten $1|1$ und $2\lfloor 0$ immer einen gültigen Kreis $C(f_1, g_1)$, vorausgesetzt der Knoten a ist nicht adjazent zu den Knoten v und u. Somit ist es in diesen Varianten unter dieser Voraussetzung stets möglich, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren.

Variante: $-2\lfloor 0$
(a) $\nexists\pi_{a,b}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,a}) \geq 3 \rightarrow m \leq 1$
(b) $\nexists\pi_{a,v}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{v,a}) \geq 3 \rightarrow m \leq 1$

Fall 1: (ζ) $\nexists\pi_{a,u}$
$(a), (b), (\zeta) \Rightarrow \exists C(f_1, g_1)$

Fall 2: (α) $\exists\pi_{a,u}$
(c) $\nexists\pi_{u,b}$ - Verknüpfungs-Korollar mit $(\alpha), (a)$
(d) $\nexists\pi_{u,v}$ - Verknüpfungs-Korollar mit $(\alpha), (b)$
$(a), (b), (c), (d) \Rightarrow \exists C(f_2, g_1)$

Es gibt also in der Varianten $-2\lfloor 0$ immer einen gültigen Kreis $C(f_1, g_1)$ oder $C(f_2, g_1)$, vorausgesetzt die Knoten a und v sind nicht adjazent zueinander. Somit ist es in dieser Variante unter dieser Voraussetzung stets möglich, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren.

Varianten: $0|2$ und $1\lfloor 1$
(a) $\nexists\pi_{a,b}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,a}) \geq 3 \rightarrow m \leq 1$
(b) $\nexists\pi_{v,b}$ - Rotations-Korollar mit $\text{rot}(\pi^x_{b,v}) \geq 3 \rightarrow m \leq 1$

Fall 1: (ζ_1) $\nexists\pi_{a,u}, (\zeta_2)$ $\nexists\pi_{a,v}$
$(a), (\zeta_1), (\zeta_2) \Rightarrow \exists C(f_1, g_1)$

Fall 2: (α) $\exists\pi_{a,u}$
(c) $\nexists\pi_{u,b}$ - Verknüpfungs-Korollar mit $(\alpha), (a)$
$(a), (b), (c) \Rightarrow \exists C(f_2, g_2)$

Fall 3: (γ_1) $\nexists\pi_{a,u}$, (γ_2) $\exists\pi_{a,v}$
(d) $\nexists\pi_{v,u}$ - Verknüpfungs-Korollar mit $(\gamma_2), (\gamma_1)$
$(a), (b), (\gamma_1), (d) \Rightarrow \exists C(f_1, g_2)$

Es gibt also in den Varianten $0|2$ und $1\lfloor 1$ immer einen gültigen Kreis $C(f_1, g_1)$, $C(f_2, g_2)$ oder $C(f_1, g_2)$, vorausgesetzt die Knoten v und b sind nicht adjazent zueinander. Somit ist es in diesen Varianten unter dieser Voraussetzung stets möglich, die Anzahl der Knicke auf der Kreuzungskante $\{a, b\}$ zu reduzieren.

3.4 Zusammenfassung

Wir haben gezeigt, dass es np-flex-Instanzen mit genau einem Kreuzungsknoten und globaler Flexibilität 2 gibt, die nicht lösbar sind. Gleichzeitig haben wir aber gezeigt, dass alle Instanzen lösbar sind, wenn man als zusätzliche Einschränkung fordert, dass die zum Kreuzungsknoten x adjazenten Knoten untereinander paarweise nicht adjazent sind.

Während also im planaren Fall jede feste Einbettung eines Graphen so gezeichnet werden kann, dass jede Kante höchstens zwei Knicke hat, ist dies im nichtplanaren Fall nicht ohne weiteres möglich. Zudem haben wir demonstriert, dass die Betrachtung der Flexibilitäten von Kreuzungskanten einen echten Mehraufwand gegenüber dem planaren Fall erfordern. In Kapitel 4 werden wir untersuchen, wie sich dieser Mehraufwand im Allgemeinen auswirkt.

4. Graphen mit multiplen Kreuzungsknoten

Wir betrachten nun Planarisierungen, die mehr als einen Kreuzungsknoten besitzen. Zunächst untersuchen wir, in welchen Fällen keine Oberschranke für die benötigte Flexibilität einer Kante angegeben werden kann. Anschließend zeigen wir, dass np-flex im Allgemeinen NP-schwer ist, selbst in jenen Fällen, die in planaren Graph in Linearzeit lösbar sind. Dazu stellen wir zunächst ein modifiziertes Flussnetzwerk vor, in dem es genau dann einen Fluss gibt, wenn die zugehörige np-flex-Instanz lösbar ist.

4.1 Kanten mit unbeschränkter Anzahl an Knicken

Zunächst betrachten wir einige Fälle, in denen keine obere Schranke für die Anzahl der benötigten Knicke auf einer Kante angegeben werden kann. Abbildung 4.1 zeigt den Fall, dass zwei Kanten sich gegenseitig fünfmal kreuzen.

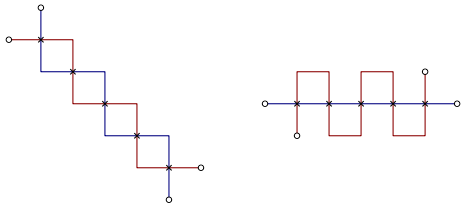


Abbildung 4.1: Kreuzen zwei Kanten sich fünfmal, so muss eine der beiden mindestens vier Knicke haben, beide zusammen müssen mindestens acht Knicke haben.

Offensichtlich erzwingt die Tatsache, dass sich zwei Kanten mehrfach kreuzen, dass diese Kanten eine gewisse Anzahl von Knicken benötigen. Wir erhalten das folgende Lemma.

Lemma 4.1. *Kreuzen sich zwei verschiedene Kanten n mal, so müssen sie zusammen mindestens $2n - 2$ Knicke haben.*

Beweis. Gegeben sei eine Planarisierung, in der sich zwei Kanten mindestens zweimal treffen. Betrachte den Subgraphen dieser Planarisierung, der lediglich aus den beiden Kreuzungskanten und ihren Kreuzungsknoten besteht. Betrachte nun eine innere Facette. Auf ihrem Rand liegen genau zwei Kantensegmente und genau zwei Kreuzungsknoten. Jeder

Kreuzungsknoten trägt zu der Facette die Rotation 1 bei, so dass auf den beiden Kantensegmenten zusammen genau zwei Knicke liegen müssen, entweder zwei auf einer der beiden und keiner auf der anderen, oder je ein Knick auf jedem der Kantensegemente. Hat der Graph n Kreuzungsknoten, so gibt es $n-1$ derartige innere Facetten. Auf den angrenzenden Kantensegmenten müssen also genau $2 \cdot (n-1)$ Knicke liegen. □

Somit ist es nicht möglich, eine obere Schranke für die benötigte Flexibilität aller Kanten anzugeben, wenn sich zwei Kanten beliebig oft schneiden dürfen. Wir erhalten somit das folgende Korollar.

Korollar 4.2. *Dürfen sich zwei Kanten beliebig oft schneiden, gibt es für jedes $k \in \mathbb{N}$ eine np-flex-Instanz mit einer Kante, deren Flexibilität mindestens k sein muss, damit diese Instanz lösbar ist.*

Der Wert des obigen Korollars wird in jeder np-flex-Instanz erreicht, in der zwei Kanten sich $k+1$ mal schneiden. Unabhängig von den Flexibilitäten der Instanz gibt es hier also Kanten, die zwangsweise eine bestimmte Anzahl an Knicken haben müssen. Somit erfordern diese Kanten eine gewisse Flexibilität, damit die Instanz lösbar ist. Da wir aber derartige Einschränkungen an die benötigte Flexibilität vermeiden wollen, gehen wir von nun an davon aus, dass sich zwei Kanten in unseren Instanzen nur einmal schneiden können.

Doch auch dies ist nicht ausreichend, um zu verhindern, dass einige wenige Kanten dafür sorgen, dass eine bestimmte Kante eine gewisse Anzahl an Knicken haben muss, wie der folgende Satz zeigt.

Satz 4.3. *Für jedes $k \in \mathbb{N}$ gibt es eine np-flex-Instanz mit einer Kante, deren Flexibilität mindestens k sein muss, damit diese Instanz lösbar ist.*

Beweis. Die schwarzen Kanten in Abbildung 4.2 haben Flexibilität 0. Daher kann der Knick der violetten Kreuzungskante nicht geglättet werden. Die schwarzen Kanten bilden also ein sogenanntes Gadget, das erzwingt, dass die Kreuzungskante, die dieses Gadget kreuzt, einen Knick erhält. Liegen nun mehrere derartige Gadgets auf derselben Kreuzungskante, so muss diese Kante folglich ebenso viele Knicke haben. Da es keine obere Schranke für die Anzahl dieser Gadgets gibt, folgt die Behauptung.

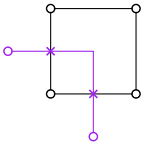


Abbildung 4.2: Der Knick auf der violetten Kreuzungskante kann nicht geglättet werden, da die schwarzen Kanten Flexibilität 0 haben.

□

Man beachte, dass das Gadget nicht nur erzwingt, dass es einen Knick gibt, es bestimmt auch die Richtung, in die dieser Knick gehen muss. Im vorliegenden Fall muss die Kreuzungskante (vom Betrachter aus gesehen) einen Linksknick machen, da sie das Gadget über jene Kante verlassen muss, die links derjenigen Kante liegt, über die sie es betreten hat.

Es sei erwähnt, dass auch im planaren Fall mit Kanten der Flexibilität 0 eine Kante dazu gezwungen werden kann, einen Knick zu haben, man betrachte beispielsweise ein einfaches Dreieck. Im Gegensatz zu hier kann im planaren Fall aber dieser Fall nicht mehrmals für

dieselbe Kante auftreten, so dass dort für jede Kante eine obere Schranke für die benötigte Flexibilität angegeben werden kann. Im nichtplanaren Fall ist dies aber nicht möglich, da eine Kreuzungskante beliebig viele der vorgestellten Gadgets durchlaufen kann.

Wir werden nun zeigen, dass es ein ähnliches Gadget auch dann geben kann, wenn es keine Kanten mit Flexibilität 0 gibt.

Satz 4.4. *Unter den np-flex-Instanzen, deren Kanten mindestens Flexibilität* 1 *haben, gibt es für jedes* $k \in \mathbb{N}$ *eine Instanz mit einer Kante, deren Flexibilität mindestens* k *sein muss, damit diese Instanz lösbar ist.*

Beweis. Man betrachte Abbildung 4.3. Die linke Abbildung zeigt die grundlegende Situation, die violette Kreuzungskante hat zwei Knicke. Die Flexibilität aller schwarzen Kanten sei 1. Die vier äußeren Kanten müssen zwangsweise wie in den drei Fällen angegeben gezeichnet werden: soll eine davon geglättet werden, muss dafür eine der drei anderen Kanten geknickt werden, was nicht möglich ist. Sollen nun die beiden Knicke auf der violette Kreuzungskante geglättet werden, so müssen zwei überschüssige Flusseinheiten verarbeitet werden. Dabei kann jede der inneren Kanten nur einmal geknickt werden, die äußeren Kanten können nur dann geknickt werden, wenn sie gleichzeitig (auf der anderen Seite der Kreuzungskante) geglättet werden. Somit verbleibt für jeden Knick nur genau ein reduzierender Kreis.

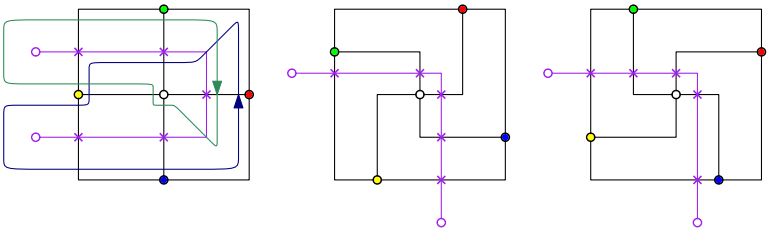


Abbildung 4.3: Nur einer der beiden Knicke der violette Kreuzungskante kann geglättet werden. Die beiden Graphen rechts zeigen den jeweils verbleibenden Knick.

Nun können aber offensichtlich nicht beide reduzierenden Kreise gleichzeitig angewendet werden. Somit muss einer der beiden Knicke verbleiben. Wir erhalten also wie zuvor ein Gadget, das einen Knick auf dieser Kreuzungskante erzwingt. □

Man beachte dass auch hier die Richtung des Knicks vorgegeben ist. Wir werden nun zeigen, dass es selbst dann noch ein derartiges Gadget geben kann, wenn man Flexibilität 1 verbietet.

Satz 4.5. *Unter den np-flex-Instanzen, deren Kanten mindestens Flexibilität* 2 *haben, gibt es für jedes* $k \in \mathbb{N}$ *eine Instanz mit einer Kante, deren Flexibilität mindestens* k *sein muss, damit diese Instanz lösbar ist.*

Beweis. Da Flexibilität 2 mehr Freiheitsgrade erlaubt als Flexibilität 1, ist das benötigte Gadget deutlich komplexer als zuvor. Gleichzeitig gibt es mehrere Möglichkeiten, ein derartiges Gadget zu realisieren. Abbildung 4.4 zeigt ein Beispiel.

Wir erläutern dieses Gadget genauer. Die linke Seite zeigt wie zuvor das Gadget im Grundzustand, keine der inneren Kanten weist einen Knick auf. Die Flexibilität sämtlicher schwarzen Kanten sei 2. Die äußeren Kanten des Gadgets verfügen über je zwei Knicke, so dass sie nicht erneut geknickt werden können, es sei denn sie werden gleichzeitig geglättet. Soll nun einer der Knicke auf der violette Kreuzungskante geglättet werden, so erhält dadurch jene Facette, in der dieser Knick den 90°-Winkel hat, eine überschüssige

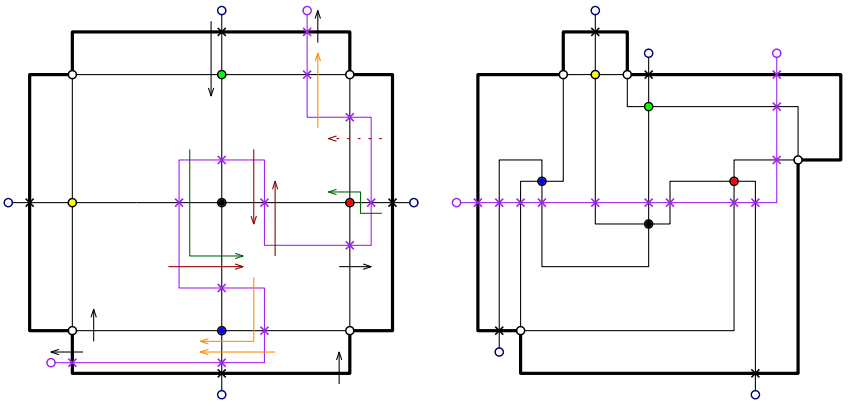


Abbildung 4.4: Der gestrichelte reduzierende Pfeil kann nicht existieren, weil die Facette die er verlässt keine Flusseinheit erhalten kann. Daher muss die violette Kreuzungskante mindestens einen Fluss haben.

Flusseinheit. Anders als zuvor, wo die überschüssige Flusseinheit das Gadget verlassen musste, damit sich ein reduzierender Kreis ergibt, kann die Flusseinheit nun auch erneut die Kreuzungskante überqueren, wenn sie dadurch einen weiteren Knick glättet.

Die in Abbildung 4.4 dargestellten reduzierenden Pfeile geben an, über welche Facetten der überschüssige Fluss dabei geleitet werden muss. Die vier roten Pfeile sind dabei eindeutig, denn den überschüssigen Fluss in die andere zur Auswahl stehende Facette zu geben ergibt keinen Sinn. Man beachte dabei, dass eine der Kanten dadurch bereits zwei Knicke erhält, so dass über diese Kante kein weiterer reduzierender Pfeil gehen kann. Dies hat zur Folge, dass die beiden grünen Pfeile festgelegt werden, was wiederum dazu führt dass zwei weitere Kanten ihren zweiten Knick erhalten. Dadurch ergibt sich wiederum automatisch die Lage der drei orangefarbenen Pfeile.

Dadurch wird aber nun offensichtlich, dass die beiden Facetten rechts aussen nicht in der Lage sind, zwei Flusseinheiten für den roten gepunkteten Pfeil und den grünen Pfeil bereitzustellen. Folglich kann einer der beiden Knicke nicht geglättet werden, in unserem Beispiel der gepunktete rote Pfeil. Die schwarzen Pfeile geben an, wie die Flusserhaltungsbedingung für die restlichen Flüsse sichergestellt werden kann, die rechte Abbildung zeigt den Graphen, wie er nach Anwendung der entsprechenden Glättungen aussieht. Auch dort lässt sich leicht nachvollziehen, dass der verbleibende Knick nicht geglättet werden kann. □

Wir erhalten ein Korollar, das deutlich macht, dass sich np-flex von herkömmlichem flexdraw unterscheidet.

Korollar 4.6. *Unter den np-flex-Instanzen, deren Kanten mindestens Flexibilität* 2 *haben, gibt es Instanzen, die nicht lösbar sind.*

4.2 Das nplex-Flussnetzwerk

In Kapitel 3 haben wir die Anzahl von Knicken auf den Kreuzungskanten untersucht, indem wir ausgenutzt haben, dass für alle anderen Kanten des Graphen zwei Knicke ausreichend sind. Dies ist leider nicht möglich, sobald es mehrere Kreuzungsknoten gibt, da nicht garantiert werden kann, dass für das Glätten einer Kreuzungskante nicht eine andere Kreuzungskante geknickt werden muss. Daher verwenden wir stattdessen das bereits angesprochene Flussnetzwerk nach Tamassia [Tam87], um np-flex in Planarisierungen mit mehreren Kreuzungsknoten zu lösen.

Wir betrachten zunächst ein Flussnetzwerk für den herkömmlichen Fall eines planaren Graphen G. Der Fluss soll dabei jenen Rotationen entsprechen, die die Facetten untereinander durch die Knicke der Kanten austauschen und die von den Knoten zu den Facetten beigetragen werden. Entsprechend enthält der Graph, auf dem das Flussnetzwerk operiert, die Knoten des Graphen G sowie einen Knoten für jede Facette des Graphen G. Die Kanten entsprechen den zu den ursprünglichen Kanten dualen Kanten sowie Kanten zwischen Knoten und Facetten, die im Graph G benachbart sind. Anstelle ungerichteter Kanten verwenden wir dabei jeweils zwei entgegengesetzt gerichtete Kanten.

Nun werden mit den Flusseinheiten also Rotationen in die einzelnen Facetten getragen. Wie wir in Kapitel 2 gezeigt haben, muss die Summe der Rotationen einer Facette $+4$ oder -4 betragen, je nachdem ob es sich um eine innere oder die äußere Facette handelt (vergleiche mit Abbildung 2.1). Um dies auch hier zu gewähren, müssen also die inneren Facetten einen Bedarf von $+4$ erfüllen, die äußere Facette einen Bedarf von -4.

Mit diesen Einschränkungen lässt sich also aus jedem gültigen Fluss dieses Flussnetzwerks eine entsprechende Orthogonale Zeichnung konstruieren. Momentan gibt es dabei aber noch keine Begrenzung für die Anzahl der Knicke auf einer Kante, da die Kanten des Flussnetzwerks keine Kapazitätsbeschränkung haben. Wir setzen nun daher für jede Kante des ursprünglichen Graphen die Kapazität der zu ihr dualen Kante des Flussnetzwerks auf den Wert der Flexibilität der Kante. Somit sichern wir, dass jede Kante des Flussnetzwerks nur so viele Flusseinheiten über die zu ihr duale Kante trägt, wie deren Flexibilität an Knicken gestattet. Abbildung 4.5 zeigt ein Beispiel.

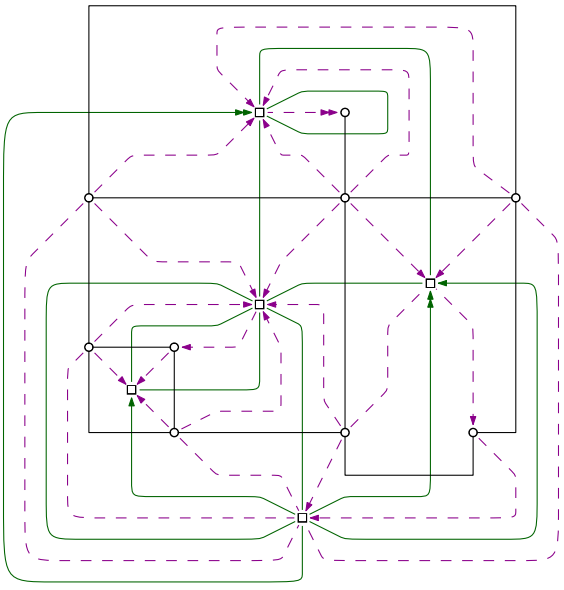


Abbildung 4.5: Ein planarer Graph mit Flussnetzwerk und gültigem Fluss: die Anzahl der Pfeile entspricht der Anzahl der Flusseinheiten auf dieser Kante.

Wir wollen dieses Flussnetzwerk nun derart erweitern, dass auch die Flexibilitäten der Kreuzungskanten berücksichtigt werden. Sei dazu G die Planarisierung eines nichtplanaren Graphen. Wir führen nun sogenannte *Bündelkapazitäten* ein, also zusätzliche gemeinsame Kapazitätsbedigungen für jene Kantenbündel, die die Kantensegmente der einzelnen Kreuzungskanten darstellen. Dies lässt sich nicht mit einem herkömmlichen Flussnetzwerk realisieren, sondern stellt eine zusätzliche Bedingung dar.

Wir fassen formal zusammen: sei $N = (G', c, d)$ das Flussnetzwerk der Planarisierung G, mit $G' = (V', E')$ als jenem Graph, auf dem das Flussnetzwerk operiert. Dabei sei die Knotenmenge $V' = V \cup F$ zusammengesetzt aus den Knoten und Facetten des Graphen G,

die Kantenmenge $E' = E_v \cup E_f$ zusammengesetzt aus den Kanten des Dualgraphen: $E_f = \{(f,g)_e, (g,f)_e \mid f,g \in F \textit{ inzident zu } e \in E\}$ sowie den Kanten, die jeden Knoten mit den Facetten verbindet, auf dessen Rand er liegt: $E_v = \{(f,v),(v,f) \mid v \in V \textit{ inzident zu } f \in F\}$.

Die Kapazitätsbedingung $c : E' \to \mathbb{N}_0$ ist eine Abbildung der Kanten des Graphen G' auf die natürlichen Zahlen, die für die Dualkanten den Wert der entsprechenden Flexibilität annimmt (also $c(e_f) = \text{flex}(e)\, \forall e_f \in E_f \textit{ dual zu } e \in E$), für die Kanten von Knoten zu Facetten den Wert 1 (also $c(e_v) = 1\, \forall e_v = (v,f) \in E_v : f \in F, v \in V$) und für die Kanten von Facetten zu Knoten den Wert 2 (also $c(e_v) = 2\, \forall e_v = (f,v) \in E_v : f \in F, v \in V$).

Man beachte dabei, dass die Kapazitätsbedingung für die Kanten in E_f definiert ist über die Flexibilität der entsprechenden dualen Kante der Planarisierung. Folglich ist sie genau dort undefiniert, wo auf den Planarisierungskanten auch die Flexibilität undefiniert ist: auf den Kantensegmenten. Denn die Kantensegmente haben keine eigene Flexibilität, sie dürfen beliebig viele Knicke haben, solange dadurch nicht die Flexibilität der Kreuzungskante verletzt wird, zu der sie gehören. Daher haben auch im Flussnetzwerk die zu den Kantensegmenten dualen Kanten keine eigene Kapazitätsbedingung. Stattdessen haben sie gemeinsam mit den zu den anderen Kantensegmenten derselben Kreuzungskante dualen Kanten eine Bündelkapazität, die der Flexibilität der Kreuzungskante entspricht.

Der Bedarf $d : V' \to \mathbb{N}_0$ ist eine Abbildung der Knoten des Graphen G' auf die natürlichen Zahlen, die für innere Facetten den Wert 4 annimmt (also $d(f) = 4\, \forall f \in F : f \textit{ ist innere Facette in } G$), für die äußere Facetten den Wert -4 (also $d(f) = -4\, \forall f \in F : f \textit{ ist äußere Facette in } G$) und für Knoten einen Wert in Abhängigkeit vom Grad des Knotens im Graph G ($d(v) = 4 - 2 \cdot \text{grad}(v)\ \forall v \in V$).

Ein gültiger Fluss im Flussnetzwerk N ist eine Abbildung $f : E' \to \mathbb{N}_0$ von den Kanten des Graphen G' auf die natürlichen Zahlen, die folgende Bedingungen erfüllt:
1.: für alle $e \in E' : f(e) \leq c(e)$. (Kantenkapazität)
2.: für alle $v \in V' : \sum_{u \in V'} f(u,v) - \sum_{u \in V'} f(v,u) = d(v)$. (Flusserhaltung)
Damit auch die Kreuzungskanten höchstens zwei Knicke haben, fordern wir zusätzlich:
3.: für alle Kreuzungskanten $e_x \in \tilde{E} : \sum_{(f,g) \in \Phi(e_x)} f(f,g) \leq \text{flex}(e_x)$. (Bündelkapazität)
Dabei ist $\Phi(e_x)$ die Menge jener Kanten in E', die dual sind zu den Kantensegmenten der Kreuzungskante e_x.

Man beachte, dass es dabei möglich ist, dass sowohl eine Kante als auch ihre Gegenkante einen Fluss tragen. Dies ist jedoch nicht zielführend, da es im Fall einer Kante aus E_f bedeuten würde, dass die Kante Knicke in entgegengesetzter Richtung hat, und im Fall einer Kante aus E_v, dass der Knoten zwei unterschiedliche Winkel in der angrenzenden Facette hat. Stattdessen betrachten wir daher den Nettofluss über diese beiden Kanten, also jenen Fluss, der tatsächlich zwischen diesen beiden Knoten des Flussnetzwerks ausgetauscht wird. Wir erhalten ihn, indem wir den niedrigeren Fluss vom höheren Fluss abziehen und den niedrigeren Fluss anschließend auf 0 setzen. Danach gilt also für alle $(u,v) \in E' : \min\{f(u,v), f(v,u)\} = 0$.

In jedem herkömmlichen Flussnetzwerk mit ausschließlich ganzzahligen Kapazitäten gibt es einen maximalen Fluss, der ebenfalls ausschließlich ganzzahlige Werte hat. Leider können wir diese Eigenschaft nicht mehr garantieren, wenn wir die Bündelkapazitäten hinzunehmen. Da wir jedoch auch weiterhin lediglich an ganzzahligen Flüssen interessiert sind, weil die Rotationen stets ganzzahlig sein müssen, stellen wir diese Forderung hier manuell: deswegen ist die Flussfunktion auf den natürlichen Zahlen definiert. Es liegt also ein ganzzahliges Flussnetzwerk vor.

Auf diese Weise können wir also zu jeder np-flex-Instanz ein Flussnetzwerk konstruieren. Wir nennen dieses Flussnetzwerk folglich das zu dieser np-flex-Instanz zugehörige np-flex-Flussnetzwerk.

Satz 4.7. *In dem zu einer np-flex-Instanz zugehörigen np-flex-Flussnetzwerk gibt es genau dann einen gültigen Fluss, wenn die np-flex-Instanz lösbar ist.*

Beweis. Der Beweis folgt direkt aus der Konstruktion des Flussnetzwerks: eine Flusseinheit entspricht der Rotation, die die Kanten bzw. Knoten zu den ihnen angrenzenden Facetten beitragen. Die Kapazitäten wurden so gewählt, dass sie den Flexibilitäten der kreuzungsfreien Kanten entsprechen, die Bündelkapazitäten so, dass sie den Flexibilitäten der Kreuzungskanten entsprechen. Abbildung 4.6 veranschaulicht den Satz.

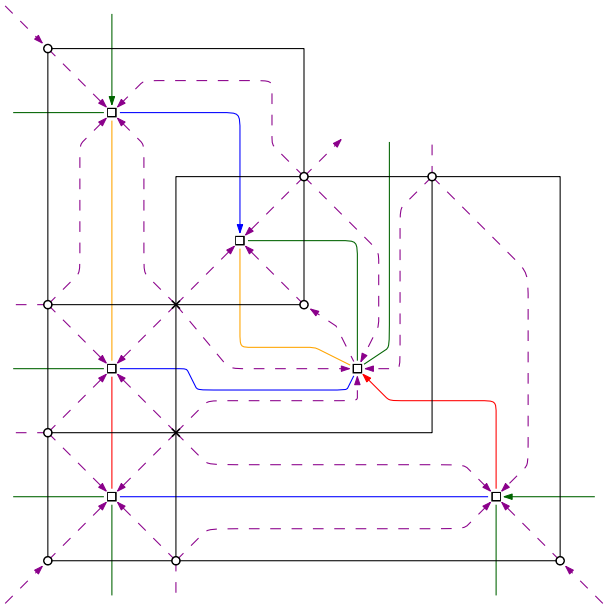


Abbildung 4.6: Eine Planarisierung mit Flussnetzwerk und gültigem Fluss: die Anzahl der Pfeile entspricht der Anzahl der Flusseinheiten auf dieser Kante. Die äußere Facette wird nur angedeutet. Die blauen, roten und orangefarbenen Kanten haben jeweils gemeinsame Bündelkapazität.

□

Natürlich lässt sich auch ein Flussnetzwerk mit Bündelkapazitäten konstruieren, die nicht zugehörig sind zu einer np-flex-Instanz, zum Beispiel indem man ein herkömmliches Flussnetzwerk nimmt und für zwei beliebige Kanten eine beliebige Bündelkapazität angibt. Ein solches Flussnetzwerk ohne zugehörige np-flex-Instanz nennen wir verwaistes np-flex-Flussnetzwerk.

Jedes herkömmliche Flussnetzwerk lässt sich als Lineares Programm beschreiben, was das Finden eines gültigen Flusses auf einfache Weise ermöglicht, allerdings in suboptimaler Zeit. Dazu werden die Kanten des Flussnetzwerks zu Variablen des LPs, die Kantenkapazität und die Flusserhaltung werden zu Bedingungen des LPs. Als Optimierungsziel verwendet man eine einfache Dummyfunktion.

Auch die von uns hinzugefügten Bündelkapazitäten lassen sich problemlos als Bedingungen des LPs formulieren. Allerdings müssen wir auch hier fordern, dass die Variablen ausschließlich ganzzahlige Werte annehmen, es handelt sich also um ein ganzzahliges lineares Problem oder ILP. Die np-flex-Flussnetzwerke lassen sich also nicht wie ein herkömmliches

LP in polynomieller Zeit lösen, das Lösen von ILPs ist grundsätzlich zunächst NP-schwer. Tatsächlich werden wir im nächsten Kapitel anhand von Beispielgraphen das Laufzeitverhalten der ILPs von np-flex-Flussnetzwerken untersuchen.

4.3 Komplexitätsbeweis

Mit Hilfe des np-flex-Flussnetzwerks können wir nun die Komplexität von np-flex untersuchen. Wir führen zunächst eine Reduktion des 3SAT-Problems auf das Finden eines gültigen Flusses im np-flex-Flussnetzwerk durch, um zu zeigen, dass dies NP-schwer ist. Wir verwenden dabei aber zunächst ein verwaistes np-flex-Flussnetzwerk, und stellen somit zunächst keine direkte Verbindung zu np-flex her. Um dies zu beheben, betrachten wir anschließend eine Variante des 3SAT-Problems, die weiterhin NP-schwer ist. Wir führen eine Reduktion jenes Problems auf ein np-flex-Flussnetzwerk durch und stellen schließlich die zu diesem Flussnetzwerk zugehörige np-flex-Instanz vor, und zeigen somit, dass np-flex NP-schwer ist.

4.3.1 3SAT als np-flex-Flussnetzwerk

Wir stellen ein np-flex-Flussnetzwerk vor, in dem es genau dann einen gültigen Fluss gibt, wenn es eine gültige Variablenbelegung in einer zugehörigen Instanz des 3SAT-Problems gibt.

Satz 4.8. *Das Finden eines Flusses im np-flex-Flussnetzwerk ist NP-schwer.*

Beweis. Das 3SAT-Problem ist ein Entscheidungsproblem der boolschen Algebra. Es gilt zu entscheiden, ob es zu einem gegebenen boolschen Ausdruck eine Belegung der Variablen gibt, welche den Ausdruck wahr werden lässt. Der boolsche Ausdruck ist dabei die Konjunktion einer Reihe von Klauseln, welche wiederum die Disjunktion von je drei paarweise verschiedenen Literalen sind. Seien $x_1, ..., x_n$ die n Variablen und $y_1, ..., y_k$ die k Klauseln. Dann ist jede Klausel y_i die Disjunktion von drei Literalen paarweise verschiedener Variablen, wobei ein Literal einer Variable entweder identisch mit dieser Variable ist oder deren Negation. Ein Beispiel wäre der Ausdruck $(x_1 \vee \bar{x_3} \vee x_4) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_5}) \wedge (x_3 \vee \bar{x_4} \vee x_5)$.

Wir stellen nun zu einer beliebigen 3SAT-Instanz ein verwaistes np-flex-Flussnetzwerk vor, in dem es genau dann einen gültigen Fluss gibt, wenn es eine Belegung der Variablen mit den Werten 0 und 1 gibt, für welche die Konjunktion der Klauseln (und somit der Ausdruck) den Wert 1 annimmt. Dazu bilden wir für jede Variable x_i zwei Pfade x_i und $\bar{x_i}$ der Länge k, über welche genau dann ein Fluss fließen soll, falls das entsprechende Literal den Wert 1 bzw. den Wert 0 annimmt. Damit nur über einen der beiden Pfade ein Fluss fliessen kann, verbinden wir die beiden Pfade mit einem Knoten für die Variable x_i, welche einen Bedarf von -1 hat, so dass sie genau eine Flusseinheit abzugeben hat. Das Ende der beiden Pfade verbinden wir ebenso mit einer gemeinsamen Senke mit Bedarf 1. So lässt sich nun jeder gültige Fluss des Flussnetzwerkes interpretieren als Zuweisung der n Literale zu den Werten 0 und 1.

Nun fügen wir die Klauseln $y_1, ..., y_k$ der Reihe nach in Form von Bündelkapazitäten hinzu wie folgt: ist das Literal x_i in der Klausel y_j enthalten, so füge die j-te Kante des Pfades $\bar{x_i}$ dem Kantenbündel E_j hinzu. Ist hingegen das Literal $\bar{x_i}$ enthalten, füge die j-te Kante des Pfades x_i hinzu. Auf diese Weise besteht also jedes Kantenbündel E_j aus genau drei Kanten, wobei durch jede dieser Kanten genau dann ein Fluss fliesst, wenn das entsprechende Literal die Klausel y_j nicht erfüllt. Nun setzen wir die Kapazität c_j jedes Kantenbündels E_j auf den Wert 2. Durch die Bündelkapazität wird also gefordert, dass höchstens zwei dieser Kanten einen Fluss führen, so dass mindestens eines der Literale, welches die Klausel y_j erfüllt, belegt sein muss. Abbildung 4.7 illustriert ein Beispiel mit $n = 4$ und $k = 4$.

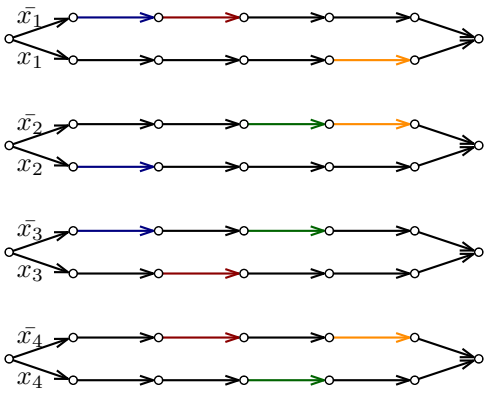


Abbildung 4.7: Die 3SAT-Instanz $(x_1 \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee \bar{x_3} \vee x_4) \wedge (x_2 \vee x_3 \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_2 \vee x_4)$ als Flussnetzwerk. Die Kanten gleicher Farbe haben eine gemeinsame Bündelkapazität von 2. Ein gültiger Fluss würde beispielsweise über die Pfade x_1^1, x_2^1, x_3^0 und x_4^1 fließen. Dies entspricht der gültigen Belegung $x_1 \wedge x_2 \wedge \bar{x_3} \wedge x_4$.

Somit gibt es in diesem np-flex-Flussnetzwerk genau dann einen gültigen Fluss, wenn es eine Belegung der Variablen gibt, für die die zugehörige 3SAT-Instanz wahr wird. Da das Finden einer solchen Belegung aber NP-schwer ist, folgt die Behauptung. □

Das in diesem Beweis verwendete np-flex-Flussnetzwerk ist wie erwähnt verwaist, denn es kann keine zugehörige np-flex-Instanz geben: jene Kanten, die hier einer gemeinsamen Bündelkapazität gehorchen müssen, können nicht tatsächlich einer gemeinsamen Kreuzungskante angehören. Dafür müssten die Kanten paarweise auf dem Rand einer gemeinsamen Facette liegen. Die Bündelkapazitäten sind so gesehen willkürlich gewählt.

4.3.2 Planares monotones 3SAT als np-flex-Flussnetzwerk

Ein gegebener boolscher Ausdruck lässt sich als Graph darstellen, in dem jedes Literal und jede Klausel ein Knoten ist. Jede Klausel ist dabei verbunden mit ihren drei Literalen, und die beiden Literale, die zur selben Variable gehören sind ebenfalls verbunden. Ist dieser Graph planar, so spricht man von einem planaren Ausdruck. Einen Ausdruck, in dessen Klauseln stets alle drei Literale entweder positiv oder negativ sind, nie jedoch gemischt, nennt man monotonen Ausdruck, wenn zusätzlich alle Klauseln mit positiven Literalen auf einer Seite der Variablen liegen und alle Klauseln mit negativen Literalen auf der anderen Seite. Abbildung 4.8 zeigt ein Beispiel für die Klauseln eines boolschen Ausdrucks, der sowohl planar als auch monoton ist.

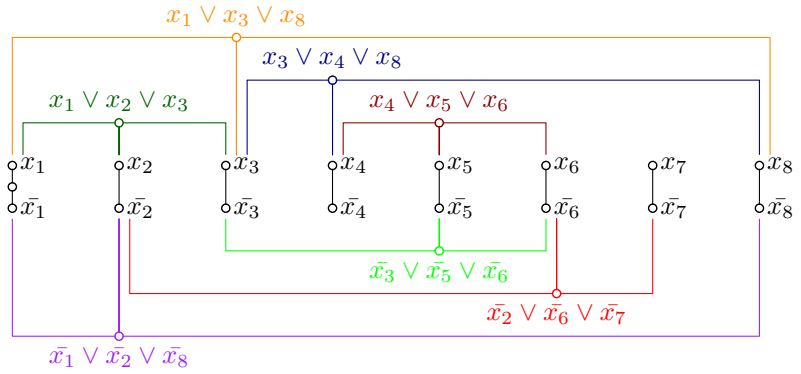


Abbildung 4.8: Die Klauseln eines planaren monotonen boolschen Ausdrucks des 3SAT-Problems.

Laut deBerg und Khosravi [dBK09] ist das Finden einer gültigen Belegung im 3SAT-Problem auch dann noch NP-schwer, wenn alle zugelassenen boolschen Ausdrücke monoton und planar sind. Man spricht in diesem Fall von planarem monotonen 3SAT.

Analog zur Vorgehensweise beim regulären 3SAT können wir nun zu einem gegebenen planaren monotonen Ausdruck ein zugehöriges np-flex-Flussnetzwerk erstellen, wobei durch die Planarität und die Monotonie jene Kanten, die zu einem Kantenbündel gehören, nahe beieinander liegen. Das einzige Problem hierbei ist jene Kante, die das Ende der Pfade der einzelnen Literale mit der gemeinsamen Senke der beiden Literale derselben Variable verbindet. Diese Kante kann nicht in jedem Fall so gelegt werden, dass sie nicht zwischen zwei Kanten eines Kantenbündels verläuft. Wir nehmen daher zu jedem Kantenbündel die drei Senkenkanten der drei enthaltenen Literale hinzu, und setzen die Bündelkapazität auf 4 anstatt auf 2. Da über die Kantensenke genau dann ein Fluss läuft, wenn auch über die entsprechende Kante des Literals ein Fluss läuft, erfüllt dies weiterhin die Bedingung, dass die Bündelkapazität genau dann verletzt wird, wenn keine der drei Variablen die Klausel erfüllt. Abbildung 4.9 zeigt als Beispiel das zu Abbildung 4.8 gehörige np-flex-Flussnetzwerk.

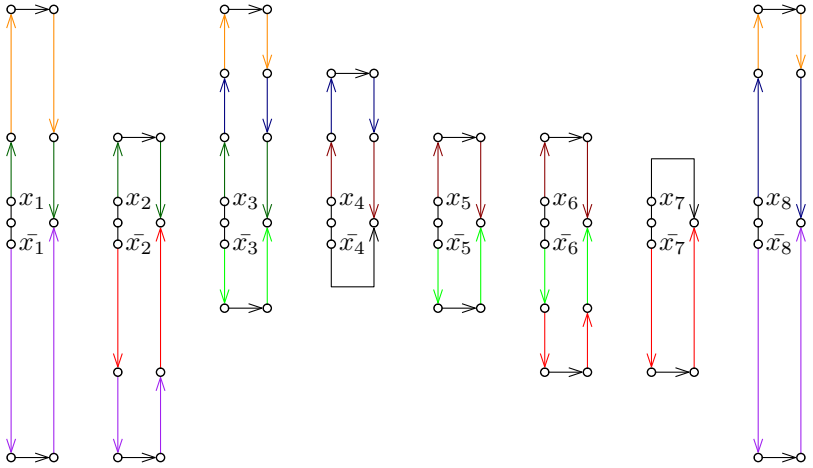


Abbildung 4.9: Das np-flex-Flussnetzwerk der obigen planaren monotonen 3SAT-Instanz. Die Kanten gleicher Farbe haben eine gemeinsame Bündelkapazität von 4.

4.3.3 Die zugehörige np-flex-Instanz

Wir zeigen nun, welche Form eine Planarisierung haben muss, damit sie zum soeben vorgestellten np-flex-Flussnetzwerk zugehörig ist und zeigen so, dass np-flex NP-schwer ist.

Satz 4.9. *np-flex ist NP-schwer.*

Beweis. Zunächst benötigen wir ein weiteres Gadget, das die Zuweisung der Variablen übernimmt: Aus diesem Gadget soll genau eine einzelne Flusseinheit abgegeben werden, entweder nach unten, oder nach oben. Dies erreichen wir mit einer Facette, die zu fünf Knoten inzident ist, wobei jeder Knoten in dieser Facette einen 90°-Winkel haben muss. Die einfachste Möglichkeit, dies sicherzustellen, ist allen Knoten den Grad 4 zu geben, durch Anhängen von einfachen Brücken. Damit die dadurch enstehende überschüssige Flusseinheit über eine der beiden Kanten abgegeben wird, die dual sind zum Beginn des Literal-Pfades, setzen wir die Flexibilität der drei anderen Kanten der Facette auf 0. Ebenso benutzen wir 0-Knick-Kanten, um die Literalpfade voneinander abzutrennen sowie um die Senken-Kanten von den Hinfluss-Kanten zu trennen. Die Senke wiederum realisieren wir als Facette mit genau 3 Knoten und einer 0-Knick-Kante, so dass eine Flusseinheit über

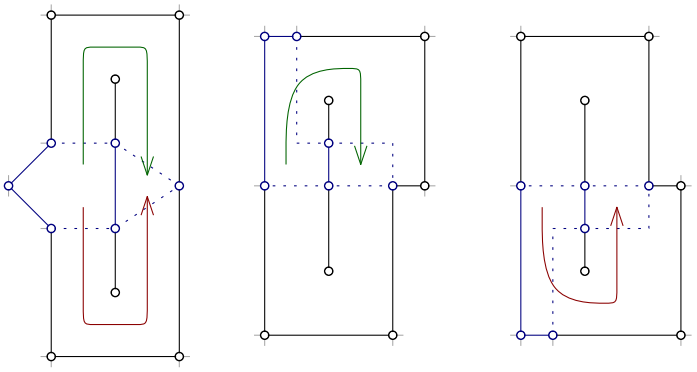


Abbildung 4.10: Das Gadget zur Realisierung einer Variablen mitsamt den dazugehörigen Literal-Pfade, sowie die Orthogonalen Zeichnungen der beiden Schaltzustände. Die durchgezogenen Kanten haben Flexibilität 0. Die überschüssige Flusseinheit der 5-Knoten-Facette muss auf dem grünen oder dem roten Pfad in die 3-Knoten-Facette fliessen.

eine der beiden anderen Kanten hineinfliessen muss. Abbildung 4.10 zeigt das gesamte Gadget einer Variablen.

Nun müssen lediglich die Gadgets derart passend zusammengefügt werden, dass die zu den geforderten Kantenbündeln dualen Kreuzungskanten korrekt gezeichnet werden können. Man betrachte dazu in der linken Abbildung die beiden äußeren Knoten des Gadgets: der linke Knoten hat zwei inzidente Kanten mit Flexibilität 0, die Teil des Gadgets sind, sowie zwei nicht näher definierte Kanten, die außerhalb des Gadgets liegen. Der rechte Knoten hat zwei Kanten mit Flexibilität 1, die Teil des Gadgets sind, und zwei Kanten mit Flexibilität 1, die Teil der Begrenzung des Pfades sind. Wir können nun zwei Gadgets nebeneinander platzieren, indem wir den linken Knoten des rechten Gadgets als den rechten Knoten des linken Gadgets verwenden: die beiden Kanten des rechten Gadgets mit Flexibilität 0 werden so zu einem Teil der Begrenzung des Pfades des linken Gadgets, entsprechend kommt auf dieser Begrenzung jeweils ein Knoten hinzu. Gleichzeitig wird für die linke Begrenzung des rechten Pfades dieselbe Kante verwendet wie für die rechte Begrenzung des linken Pfades. Endet einer der beiden Pfade, so liegt auch auf der Begrenzung des anderen Pfades ein entsprechender Knoten zur Abtrennung. Aus Gründen der Übersichtlichkeit zeigt Abbildung 4.11 ein vereinfachtes Beispiel.

Wir können nun also zu jeder beliebigen planaren monotonen 3SAT-Instanz eine np-flex-Instanz angeben, die genau dann lösbar ist, wenn die 3SAT-Instanz lösbar ist. Da für diese Reudktion lediglich eine konstante Anzahl an Operationen für jede Variable und für jede Klausel nötig ist, läuft sie in Polynomialzeit ab. Folglich ist np-flex NP-schwer, wenn es Kanten mit Flexibilität 0 gibt. □

4.3.4 Verfeinerungen des Komplexitätsbeweises

Eine der Forderungen im vorangegangenen Beweis war, dass der Graph über Kanten mit Flexibilität 0 verfügt. Wir haben jedoch in Satz 4.5 gezeigt, dass wir in np-flex-Instanzen, deren Kanten mindestens Flexibilität 2 haben, Kanten konstruieren können, die gezwungenermaßen über Knicke in eine gewisse Richtung verfügen. Erzwingt man auf diese Weise auf einer Kreuzungskante mit Flexibilität 2 direkt hintereinander zwei Knicke in entgegengesetzter Richtung, erhält man eine gerade Kante, auf der es keinen weiteren Knick geben darf. Diese Kante verhält sich also in allen Belangen wie eine Kante mit Flexibilität 0. Abbildung 4.12 veranschaulicht dies anhand des Gadgets zur Knickerzwingung unter Flexibilität 1 aus Satz 4.4.

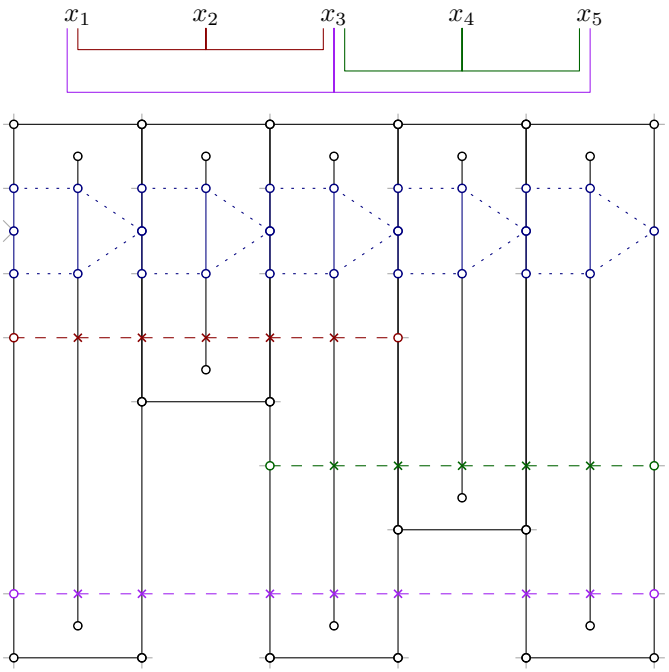


Abbildung 4.11: Der Graph, dessen zugehöriges np-flex-Flussnetzwerk genau dann einen gültigen Fluss hat, wenn die dargestellte planare monotone 3SAT-Instanz eine Lösung hat. Die Kreuzungskanten haben Flexibilität 4.

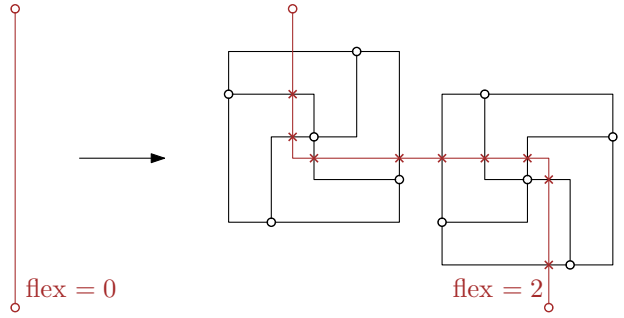


Abbildung 4.12: Anstelle einer Kante mit Flexibilität 0 verwenden wird eine Kreuzungskante mit Flexibilität 2, die durch zwei entgegengesetzt knickende Knickerzwingungs-Gadgets führt.

Wiur können also auch dann die für den Schwere-Beweis benötigten Gadgets konstruieren, wenn alle Kanten der np-flex-Instanz mindestens Flexibilität 2 haben, indem wir die in den Gadgets benötigten Kanten mit Flexibilität 0 durch die soeben konstruierten Kanten virtueller Flexibilität 0 ersetzen. Wir erhalten das folgende Korollar.

Korollar 4.10. *np-flex ist auf Instanzen, deren Kanten mindestens Flexibilität* 2 *haben, NP-schwer.*

Die Kreuzungskanten, die wir zur Realisierung der Klauseln der 3SAT-Instaz verwendet haben, hatten 6 oder mehr Segmente und Flexibilität 4. Auch hier können wir die Flexibilität auf 2 senken, und zwar indem wir jede Kreuzungskante durch zwei einzelne Kreuzungskanten ersetzten. Anstatt alle drei Literale der Klausel gleichzeitig miteinander zu vergleichen, vergleichen wir das innere der drei Literale jeweils mit den beiden äußeren. Wir fassen also das linke Literal mit dem mittleren Literal zu einer Kreuzungskante mit Flexibilität 2 zusammen und das mittlere Literal mit dem rechten Literal ebenso. Damit es nun weiterhin einen gültigen Fluss gibt, wenn sowohl das mittlere Literal als auch eines der äußeren Li-

terale erfüllt sind, geben wir für das mittlere Literal eine Ausweichmöglichkeit vor. Diese Ausweichmöglichkeit darf jedoch nicht sowohl auf dem Hinfluss als auch auf dem Rückfluss verwendet werden, sondern nur auf einem der beiden Wege, da andernfalls ein Fluss durch alle drei Literale möglich wäre. Abbildung 4.13 veranschaulicht diese Anforderung.

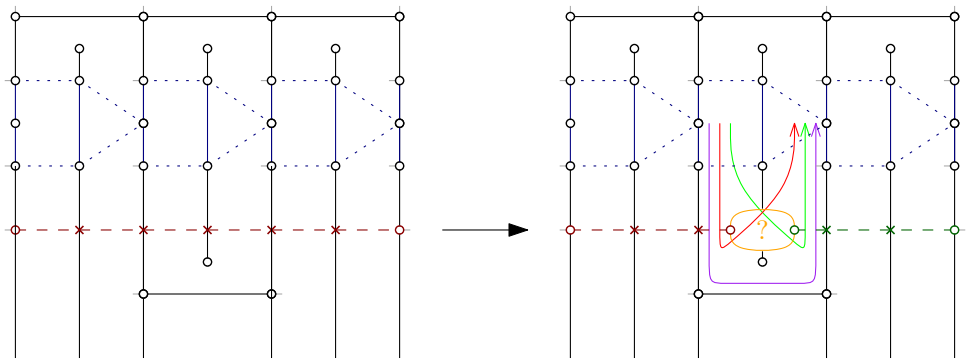


Abbildung 4.13: Anstelle einer Kreuzungskante mit Flexibilität 4 verwenden wir zwei Kreuzungskanten mit Flexibilität 2. Das mittlere Literal muss dabei die Möglichkeit haben, entweder auf dem Hinfluss oder auf dem Rückfluss der entsprechenden Kreuzungskante auszuweichen.

Die Lösung ist naheliegenderweise eine dritte Kreuzungskante, die die beiden anderen Kreuzungskanten miteinander verbindet und die Flexibilität 1 hat. Auf diese Weise kann nur entweder der Hinfluss oder der Rückfluss diese dritte Kante knicken, auf dem anderen Weg muss dementsprechend die zugehörige ursprüngliche Kreuzungskante verwendet werden. Damit wir auch hier die Bedingung einhalten, dass alle Kanten mindestens Flexibilität 2 haben, verwenden wir anstelle einer herkömmlichen Kante mit Flexibilität 1 eine geknickte Kante mit Flexibilität 2, wobei wir den Knick mit Hilfe des Gadgets aus Satz 4.5 erzwingen. Dazu müssen wir natürlich den Winkel an einem der zu dieser 1-Knick-Kante inzidenten Knoten verändern, was hier aber problemlos möglich ist. Abbildung 4.14 veranschaulicht die vier möglichen resultierenden Situationen.

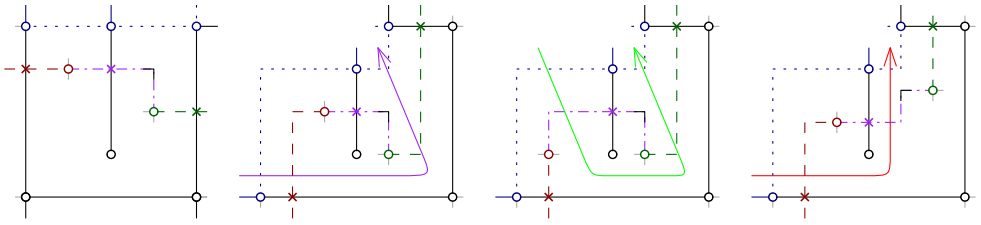


Abbildung 4.14: Ist das Literal nicht aktiv, fliesst über keine der drei Kreuzungskanten ein Fluss. Andernfalls ergeben sich genau die drei zuvor geforderten Möglichkeiten für den Fluss, da die mittlere Kreuzungskante zusätzlich zu dem erzwungenen Knick höchstens einen weiteren Knick haben kann.

Wir erhalten direkt ein weiteres Korollar.

Korollar 4.11. *np-flex mit globaler Flexibilität* 2 *ist NP-schwer.*

4.4 Zusammenfassung und Ausblick

Wir haben nun also gezeigt, dass np-flex selbst dann NP-schwer ist, wenn alle Kanten mindestens Flexibilität 2 haben. Dies ist ein großer Unterschied zum planaren Fall mit freier Einbettung, wo nicht nur in fast allen Fällen eine Zeichnung mit höchstens 2 Knicken exisitert, sondern diese sogar leicht gefunden werden kann.

Man beachte dabei aber, dass die für die Konstruktion benötigten Gadgets aus Satz 4.4 Kreuzungskanten mit einer hohen Anzahl an Kreuzungsknoten erfordern. Bei zwei benötigten Gadgets und 12 Kreuzungsknoten pro Gadget ergeben sich Kreuzungskanten mit jeweils 24 Kreuzungsknoten. Da derartige Gadgets in Graphen aus der Praxis im Allgemeinen nicht vorkommen dürften, ist die von uns vorgestellte Vorgehensweise trotz der theoretischen Worst-Case-Laufzeit sicherlich praxistauglich. Um dies zu überprüfen, werden wir das ILP in Kapitel 5 anhand von Beispielgraphen evaluieren.

Es bleibt zu untersuchen, ob eine Reduktion von np-flex auf ein NP-schweres Problem auch dann noch möglich ist, wenn man die Benutzung der von uns vorgestellten Gadgets untersagt, indem man die Anzahl der Kreuzungsknoten pro Kante stärker einschränkt als wir dies getan haben. Eine zukünftige Untersuchung der Laufzeit sollte also neben der Flexibilitäten der Kanten auch deren Anzahl an Kreuzungsknoten berücksichtigen.

5. Evaluation des Linearen Programms

Wir wollen nun die durchschnittliche Laufzeit des in Abschnitt 4.2 vorgestellten ILPs zur Lösung des np-flex-Problems anhand von Beispielgraphen mit minimaler Flexibilität 1 evaluieren. Dabei wollen wir insbesondere untersuchen, wie viel schneller dieses lineare Programm ist, wenn wir die Bündelkapazitäten des npflex-Flussnetzwerks ignorieren und somit eine Lösung für eine planare Instanz suchen; denn dies ist bekanntlich in Polynomialzeit möglich und kann somit als Referenz herangezogen werden dafür, wie viel schwerer np-flex in der Praxis ist als das herkömmliches flex-draw-Problem.

5.1 Erwartungshaltung

Wie wir in Abschnitt 4.3 gezeigt haben, ist np-flex mit minimaler Flexibilität 2 NP-schwer. Folglich ist im worst-case damit zu rechnen, dass extrem lange Laufzeiten auftreten. Wie wir aber ebenfalls in Abschnitt 4.3 betont haben, setzte der Komplexitätsbeweis die Existenz von Gadgets voraus, deren Auftreten in nicht speziell dafür konstruierten Graphen eher unwahrscheinlich ist. Somit ist zu erwarten, dass ein derartiger worst-case eher selten auftritt, weswegen die Laufzeit unseres ILPs im Allgemeinen durchaus in für die Praxis erträglichem Rahmen liegen sollte.

Wir wollen nun also untersuchen, welche Laufzeit unser ILP auf verschiedenen Graphen mit unterschiedlichen Flexibilitäten hat, wollen dabei feststellen, von welchen Parametern die Laufzeit abhängt und klären, in wie vielen Fällen denn nun tatsächlich Laufzeiten auftreten, die erheblich viel höher sind als im planaren Fall.

Dabei ist zu erwarten, dass die Laufzeit stark von der Flexibilität der Kanten abhängt: Sind die Flexibilitäten zu hoch, so ist das Finden einer passenden Zeichnung trivial. Sind sie hingegen deutlich zu niedrig, kann schnell festgestellt werden, dass eine Lösung nicht existieren kann. Daher betrachten wir im Laufe der Evaluation unterschiedliche Flexibilitäten in Abhängigkeit von der Anzahl der Kreuzungsknoten auf den Kanten.

5.2 Gestaltung der Versuchsreihen

Die Komplexität von np-flex rührt aus der Notwendigkeit, bei der Verteilung der für die Zeichnung notwendigen Knicke nicht nur die Flexibilität herkömmlicher planarer Kanten betrachten zu müssen, sondern zusätzlich auch die möglichen Verteilungen der Knicke auf die Kantensegmente der einzelnen Kreuzungskanten. Folglich wird die benötigte Laufzeit

mit der Anzahl der Kreuzungskanten steigen, insbesondere mit der durchschnittlichen Anzahl der Kreuzungsknoten auf den Kreuzungskanten.

Wir wollen daher zwei unterschiedliche Arten von Graphen betrachten: Graphen, auf denen nur wenige der Kanten Kreuzungskanten sind auf der einen Seite und Graphen, bei denen die meisten Kanten Kreuzungskanten sind auf der anderen Seite.

Um eine Verfälschung der Ergebnisse zu vermeiden wollen wir die Planarisierungen der Graphen aber nicht manuell konstruieren. Stattdessen verwenden wir zufällig generierte Graphen mit Maximalgrad 4 und lassen diese vom Open Graph Drawing Framework (OGDF) [CGJ+12], einer Bibliothek zur Zeichnung von Graphen, mittels der von Gutwenger und Mutzel [GM04] erstellten Heuristiken planarisieren. Auf diese Weise erhalten wir natürliche, praxisrelevante Einbettungen zufälliger nichtplanarer Graphen.

Dabei hängt die Anzahl der Kreuzungskanten direkt von der Dichte des Graphen ab: je mehr Kanten es in Relation zur Anzahl der Knoten gibt, desto höher ist naheliegenderweise die Anzahl der Kreuzungsknoten, die für eine Planarisierung notwendig sind. Wir verwenden daher einen dünn besiedelten Graphen zur Erstellung der Planarisierungen mit wenigen Kreuzungskanten und einen dicht besiedelten Graphen zur Erstellung der Planarisierungen mit vielen Kreuzungskanten.

Für jede dieser Klassen von Graphen lassen wir 20 unterschiedliche nichtplanare einfache Graphen mit Maximalgrad 4 generieren, wobei wir die Anzahl der Knoten und Kanten des Graphen vorgeben. Andere mögliche Parameter wie die Anzahl der Knoten mit Grad 4 oder das Vorhandensein von Cliquen verwenden wir nicht. Anschließend lassen wir jeden dieser Graphen wie beschrieben planarisieren und erhalten so eine feste Einbettung der 20 Zufallsgraphen.

Nun wollen wir unser ILP für jede dieser Planarisierungen mit unterschiedlichen Flexibilitäten laufen lassen. Um eine Referenz zu den planaren Graphen zu erhalten, behandeln wir die Planarisierung zunächst, als wäre sie ein herkömmlicher planarer Graph. Wir ignorieren also die Bündelkapazitäten und geben stattdessen jedem Kantensegment eine eigene Kapazitätsbedingung.

Da wir nicht ausschließlich an globalen Flexibilitäten interessiert sind, geben wir jeder Planarisierungskante zufällig eine Flexibilität von 1 oder 2. Die Wahrscheinlichkeit, dass eine Kante die Flexibilität 2 erhält, lassen wir dabei von Versuch zu Versuch in 5%-Schritten ansteigen. So hat zunächst jede Kante Flexibilität 1, anschließend gibt es ein paar wenige Kanten mit Flexibilität 2, bis zum Schluss zunächst fast jede und schließlich jede Kante Flexibilität 2 hat. Dabei lassen wir für jede Wahrscheinlichkeit 20 unterschiedliche Verteilungen der Flexibilität untersuchen. Ausgenommen sind die Wahrscheinlichkeiten 0% und 100%, da für diese die Flexibilität global 1 bzw. 2 ist. Pro Graph führen wir also 382 Versuche mit dieser Konfiguration der Flexibilität durch, insgesamt also 7.640 Versuche.

Anschließend entfernen wir in jeder der Planarisierungen die Kapazitätsbedingungen der Kantensegmente und verwenden stattdessen die Bündelkapazitäten der Kreuzungskanten. Sukzessive verwenden wir dabei geringere Flexibilitäten und machen es so stetig unwahrscheindlicher, dass die np-flex-Instanz lösbar ist. Dabei legen wir die Flexibilität einer Kreuzungskante in Abhängigkeit von der Anzahl ihrer Kreuzungsknoten fest: zunächst steigt die Anzahl der erlaubten Knicke linear mit der Anzahl der Kreuzungsknoten, dann nur noch als Wurzel der Anzahl und schließlich sogar nur noch logarithmisch. Zum Abschluß erhält jede Kante unabhängig von ihrer Kreuzungsknotenanzahl die selbe Flexibilität. Bei alldem geben wir wie zuvor zufällig manchen der Kreuzungskanten eine um eins erhöhte Flexibilität, und lassen die Wahrscheinlichkeit dafür sukzessive ansteigen. So erhalten wir für jeden der vier Fälle weitere 7.640 Versuche, für eine Gesamtzahl von 38.200 Durchläufen für jede der beiden Klassen pro Graph.

Flex-Modus	Flexibilität	flex $\in [2..3]$	flex $\in [3..4]$	flex $\in [4..5]$
Planar	Plan.kanten $S+1$	*	*	*
Linear	$S+X$	$X=2$	$X=3$	$X=4$
Wurzel	$S+\lfloor\sqrt{X}\rfloor$	$X \in [4..8]$	$X \in [9..15]$	$X \in [16..24]$
Log	$S+\max(1,\lfloor\ln(X)\rfloor)$	$X \in [8..20]$	$X \in [21..54]$	$X \in [55..148]$
Konstant	$S+1$	-	-	-

Tabelle 5.1: Die unterschiedlichen Flexibilitäten der Kreuzungskanten mistamt der Anzahl an Kreuzungsknoten, für die die jeweilige Flexibilität die Werte $2+S$, $3+S$ und $4+S$ annimmt. S ist je nach Wahrscheinlichkeit 0 oder 1, X die Anzahl der Kreuzungsknoten

Tabelle 5.1 zeigt die genaue Berechnung der verschiedenen Flexibilitäten tabellarisch. Dabei sei X die Anzahl der Kreuzungsknoten der jeweiligen Kreuzungskante und S zufällig 0 oder 1, wobei die Wahrscheinlichkeit für $S = 1$ im Verlaufe der Versuche wie gesagt ansteigt.

5.3 Technische Durchführung

Zur Implementierung des ILPs verwenden wir C++ mit OGDF (in der Version 2012.07) zur Verwaltung der Graphen und für die Erstellung der Planarisierungen. Zwar enthält OGDF auch Graphgeneratoren, doch diese sind nicht in der Lage, Graphen mit Maximalgrad 4 zu erstellen.

Wir verwenden daher einen eigenen Graphgenerator, dem wir als Eingabeparameter die Anzahl der Knoten n und die Anzahl der Kanten m übergeben. Dieser Graphgenerator erstellt zunächst aus den n Knoten einen Baum, indem er jeden Knoten beginnend mit dem ersten zufällig mit einem der bereits verbundenen Knoten verbindet. Dabei wird darauf geachtet, dass nur Knoten ausgewählt werden, die nicht bereits Grad 4 haben. Auf diese Weise erhalten wir einen zusammenhängenden einfachen Graph mit n Knoten und $n-1$ Kanten sowie Maximalgrad 4.

Anschließend bestimmen wir zufällig Paare von Knoten, die nicht bereits miteinander verbunden sind und die beide noch nicht Grad 4 haben, und verbinden diese miteinander. Dies wiederholen wir solange, bis insgesamt m Kanten eingefügt wurden. So erhalten wir den gewünschten zufälligen einfachen zusammenhängenden Graph mit Maximalgrad 4.

Diesen Graph übergeben wir an den *SubgraphPlanarizer* der OGDF-Bibliothek. Dieser erstellt mittels der Funktion *FastPlanarSubgraph* zunächst einen planaren Subgraphen unseres Zufallsgraphen und merkt sich, welche Kanten dafür aus dem Graphen entfernt werden mussten. Anschließend verwendet sie die Funktion *FixedEmbeddingInserter*, um diesem planaren Subgraphen nach und nach die zuvor entfernten Kanten wieder hinzuzufügen, wobei darauf geachtet wird, dass dadurch möglichst wenige Kreuzungen entstehen. Diese Funktion liefert keineswegs optimale Ergebnisse, ist für unsere Zwecke aber völlig aussreichend; schließlich interessiert uns primär die Knickminimierung, nicht die Kreuzungsminimierung. Auf diese Weise erhalten wir also eine feste Planarisierung unseres Zufallsgraphen.

Tabelle 5.2 zeigt die von uns verwendeten Parameter für die Erstellung der unterschiedlichen Graphen mitsamt dem Bereich der Anzahl an Kreuzungskanten und Kreuzungsknoten, die die Planarisierungen dieser Graphen hatten, sowie dem Bereich der Anzahl an Kreuzungsknoten, die jene Kante im Durchschnitt hatte, die in der jeweiligen Planarisierung die meisten Kreuzungsknoten hatte.

	Dünner Graph	Dichter Graph
Anzahl Knoten	1.000	500
Anzahl Kanten	1.100	900
Anzahl Kreuzungskanten	[143..168]	[621..668]
Anzahl Kreuzungsknoten	[353..557]	[8.078..9.351]
Kreuzungsreichste Kante	[17..28]	[85..126]

Tabelle 5.2: Die Werte der beiden von uns betrachteten Klassen von Graphen.

Nun wollen wir die Variablen und Bedingungen festlegen, die zur Lösung jenes ILPs benötigt werden, das zum Finden eines gültigen Flusses in dem zur np-flex-Instanz dieser Planarisierung zugehörigen np-flex-Flussnetzwerk verwendet werden kann. Zur Lösung des ILPs verwenden wir *Gurobi Optimizer* [GO12], ein Programm zur Lösung von Optimierungsproblemen (in der Version 5.0.2). In diesem erstellen wir das Modell zur Lösung unseres ILPs wie in Abschnitt 4.2 beschrieben, nur dass wir zunächst keine konkreten Werte für die Kantenkapazitäten und Bündelkapazitäten festlegen: zwar erstellen wir hier bereits die entsprechenden Bedingungen, da die Werte der Flexibilitäten aber noch nicht bekannt sind, fügen wir sie erst später dem zu lösenden Modell hinzu.

Nun folgen drei ineinander geschachtelte Schleifen. Die äußerste Schleife iteriert über die 20 Durchläufe, die zu jeder Wahrscheinlichkeit (ausser den Wahrscheinlichkeiten 0% und 1%) durchgeführt werden soll. Die nächste Schleife iteriert über eben jene Wahrscheinlichkeiten. Die innerste Schleife wiederum iteriert über die fünf verschiedenen Flex-Modi, beginnend mit dem Flex-Modus Planar.

Hierfür wird für jede Variable des ILPs, die für eine der Planarisierungskanten steht, eine Kapazitätsbedingung von 1 oder 2 zugefügt, in Abhängigkeit von der aktuellen Wahrscheinlichkeit für $S = 1$. Mit diesen Flexibilitäten wird die Optimierung von gurobi gestartet. Anschließend werden die Kapazitätsbedingungen für die Kantensegmente entfernt. An ihre Stelle sollen nun die Bündelkapazitäten der Kreuzungskanten im Flex-Modus Linear treten. Dafür wird zunächst für jede Kreuzungskante zufällig ein S bestimmt, wiederum der Wahrscheinlichkeit entsprechend. Dann wird für jede Kreuzungskante die entsprechende Bündelkapazität in Höhe von $S + X$ hinzugefügt, in Abhängigkeit von der Anzahl X der Kreuzungsknoten auf dieser Kreuzungskante. Die Optimierung wird erneut gestartet (mit Verwerfung der zuvor gesammelten Ergebnisse). Anschließend werden die Bündelkapazitäten entfernt und durch Bündelkapazitäten des Flex-Modus Wurzel in Höhe von $S + \lfloor\sqrt{(X)}\rfloor$ ersetzt, mit demselben S wie im Flex-Modus Linear. Das Ganze wiederholt sich für die Flex-Modi Log und Konstant. Daraufhin werden sämtliche Bündelkapazitäten sowie sämtliche Kantenkapazitäten entfernt, und erneut mit Flex-Modus Planar begonnen, nun mit erhöhter Wahrscheinlichkeit für $S = 1$.

Da np-flex grundsätzlich NP-schwer ist, ist zu erwarten, dass einige der Instanzen eine sehr hohe Laufzeit benötigen werden. Da es uns ausreicht, derartige Instanzen als solche zu erkennen, brechen wir die gurobi-Optimierung nach einer gewissen Zeit ab und markieren die Instanz als *schwer*, um unnötig lange Laufzeiten zu vermeiden. Bei der Berechnung der durchschnittlichen Laufzeit stellen wir diese Instanzen aussen vor. Als Wert für die maximale Laufzeit wählen wir dabei eine Sekunde pro Kreuzungskante des Graphen, also (je nach Graph) etwa zweieinhalb Minuten auf den dünnen Graphen und etwa 11 Minuten auf den dichten Graphen.

Wir kompilieren unser Programm unter openSUSE 11.3 (kernel version 2.6.34) mit g++ version 4.5.0 und Optimierungslevel -O3. Der Rechner, den wir verwenden, verfügt als Prozessor über einen 8-core Intel Xeon E5430 mit 2,66 GHz Taktfrequenz und je 6 MB L2

Cache sowie über 32 GB Arbeitsspeicher. Um aussagekräftigere Ergebnisse zu erhalten, beschränken wir gurobi dabei aber auf die Verwendung eines einzelnen Prozessorkerns.

5.4 Ergebnisse

Nachdem sämtliche Versuche abgeschlossen sind, tragen wir die Laufzeiten der unterschiedlichen Instanzen zusammen und bilden sie gemeinsam mit dem Anteil der lösbaren und der schweren Instanzen in Schaubilder ein, in Abhängigkeit von der Wahrscheinlichkeit für $S = 1$ und getrennt nach dünnen und dichten Graphen. Wir besprechen die Erkenntnisse, die wir aus diesen Schaubildern ziehen können, detailliert.

5.4.1 Durchschnittliche Laufzeiten auf der Klasse der dünnen Graphen

Wir betrachten zunächst die Ergebnisse der Graphen-Klasse dünne Graphen, also jene Graphen, bei denen der Anteil der Kreuzungskanten bei etwa 15% liegt. Abbildung 5.1 zeigt den Anteil der schweren und den Anteil der lösbaren Instanzen im Vergleich mit der durchschnittlichen Laufzeit des Programms, aufgeschlüsselt nach lösbaren Instanzen und nicht lösbaren Instanzen. Jeder Graph enthält dabei eine eigene Kurve für jeden der fünf Flex-Modi, jedoch nur dort, wo es auch tatsächlich entsprechende Instanzen gibt.

Wir erläutern die rote Kurve, die für den Flex-Modus Planar steht. Dem untersten Schaubild, das die Aufschlüsselung der Instanzen nach Schwere und Lösbarkeit enthält, können wir anhand der Punkte entnehmen, dass keine der Instanzen schwer war. Dies ist nicht weiter überraschend, da der Flex-Modus Planar den planaren Fall wiedergibt, der bekanntlich nicht NP-schwer ist. Die rote Kurve hingegen zeigt, dass die meisten der Instanzen lösbar waren. Lediglich nahe der y-Achse gab es einige Instanzen, für die keine Zeichnung unter Einhaltung der Flexibilität gefunden werden konnte. Dies ist jener Bereich, in dem die Wahrscheinlichkeit gering war, dass eine Kante Flexibilität 2 anstatt Flexibilität 1 hat. Das obere und das mittlere Schaubild zeigen, dass unabhängig von der Lösbarkeit die Laufzeit des ILPs in diesem Modus stets im Millisekunden-Bereich lag.

Sehr ähnlich verhielt es sich im Flex-Modus Linear, erkennbar an der dunkelblauen Kurve. Dort war jede einzelne Instanz lösbar, und auch hier war die Laufzeit, die benötigt wurde, um dies herauszufinden, sehr gering. Offenbar war es also mehr als ausreichend, auf jeder Kreuzungskante einen Knick pro Kreuzungsknoten zu erlauben. Durch diese großzügige Wahl der Flexibilitäten war es wohl ein leichtes, eine entsprechende Zeichnung zu finden. Dies zeigt deutlich, dass die Hinzunahme der Bündelkapazitäten zunächst keine bedeutsame Auswirkung auf die Laufzeit des Programms hat: solange die Flexibilität der Kreuzungskanten hoch genug ist, ist es in kurzer Laufzeit möglich, sie zu überprüfen.

Ein wenig anders sieht es bei der grünen Kurve für Flex-Modus Wurzel aus. Offensichtlich gab im geringen Wahrscheinlichkeitsbereich einige Instanzen, in denen es nicht ausreichend war, auf einer Kante mit X Kreuzungsknoten eine Flexibilität von $\lfloor\sqrt{(X)}\rfloor + S$ zur Verfügung zu stellen: Hier gab es einige Instanzen, die nicht lösbar waren. Im Gegensatz zu Flex-Modus Planar, wo ein ähnlicher Effekt auftrat, ohne sich dabei auf die Laufzeit auszuwirken, ist es hier in jenen Instanzen, die nicht lösbar sind, deutlich zeitaufwändiger als zuvor, dies herauszufinden. Das Überprüfen der Bündelkapazitäten ist also dann nicht mehr ohne weiteren Aufwand möglich, wenn sie Schwierigkeiten bezüglich der Lösbarkeit machen, weil hier viele verschiedene Möglichkeiten zur Verteilung der Knicke betrachtet werden müssen.

Noch deutlicher ist dieser Effekt anhand der orangefarbenen Kurven für Flex-Modus Log zu beobachten. Der Anteil der lösbaren Instanzen steigt hier in etwa linear mit der Wahrscheinlichkeit für eine Erhöhung der Flexibilität: Anfangs ist keine der Instanzen lösbar,

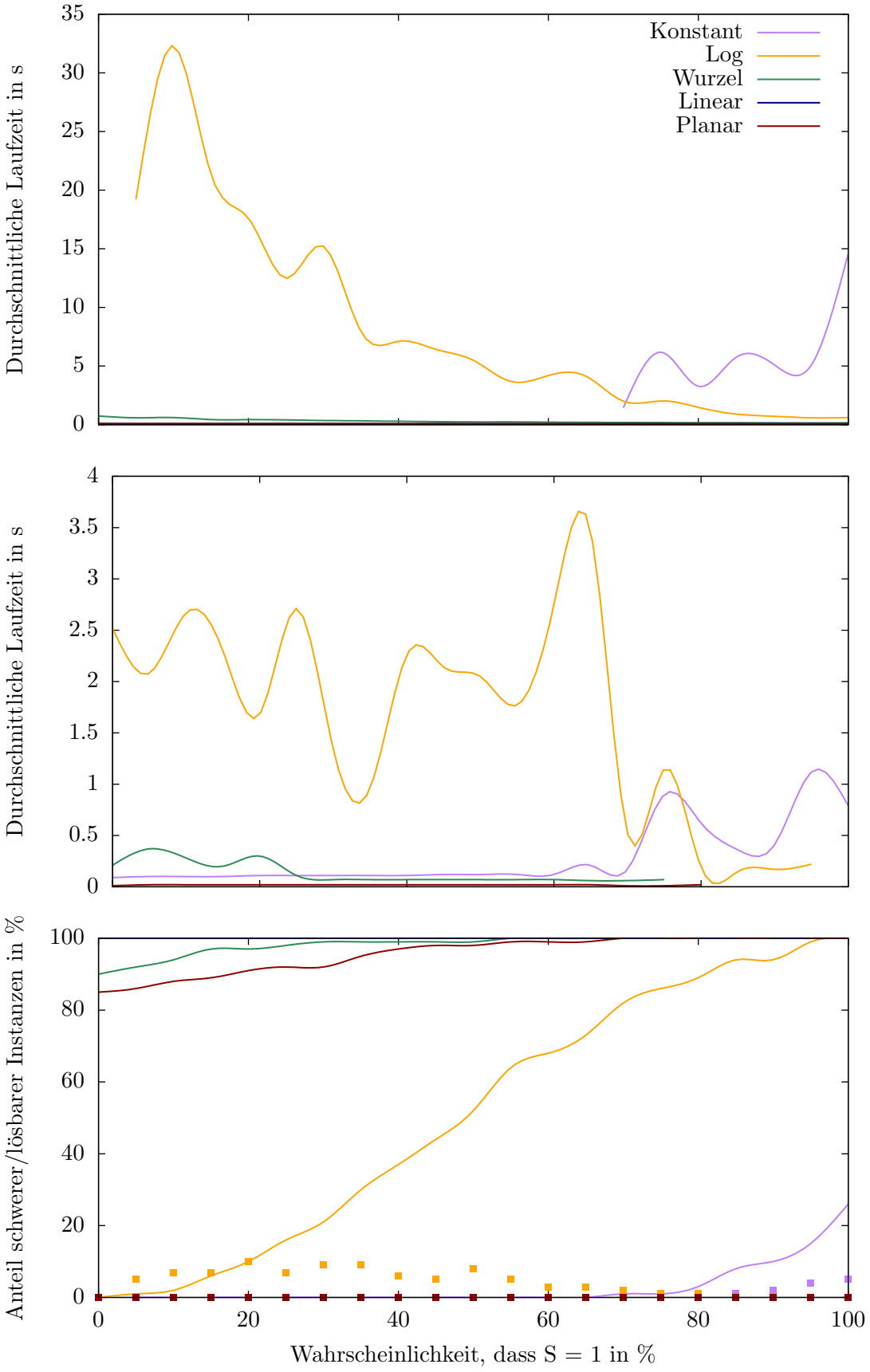


Abbildung 5.1: Die Schaubilder der dünnen Graphen; Oben bzw. Mitte: Durchschnittliche Laufzeit der lösbaren bzw. nicht lösbaren Instanzen; Unten: Anteil der lösbaren Instanzen (Linien) und schweren Instanzen (Punkte)

am Ende alle. Gleichzeitig sehen wir, dass in jenem Bereich, in dem sich viele nicht lösbare Instanzen befinden, gleichzeitig einige schwere Instanzen auftauchen. Hier benötigte das Programm also mehr als eine Sekunde pro Kreuzungskante des Graphen und brach deshalb ab. Ähnlich wie im Flex-Modus Wurzel korreliert diese Erhöhung der Laufzeit mit dem Absinken der Lösbarkeit, weil es schwierig ist, herauszufinden, ob es unter den vielen verschiedenen Möglichkeiten nicht doch eine gibt, die das Problem löst. Anders als zuvor finden wir hier aber selbst dort, wo die Instanzen sich als lösbar herausstellten, eine Erhöhung der Laufzeiten. Diese fiel sogar noch deutlicher aus als bei den nicht lösbaren Instanzen. Hier war es also offensichtlich schwierig, jene Verteilung zu finden, die die np-flex-Instanz löst, weil dafür zunächst viele Verteilungen probiert werden mussten, mit denen die die Instanz nicht gelöst wurde. Beide Arten der Laufzeit nehmen dort ab, wo der Anteil der lösbaren Instanzen ansteigt. Weil in diesem Bereich die Freiheiten bei der Verteilung der Knicke größer sind, müssen nicht so viele Varianten durchprobiert werden, weswegen es leichter ist, eine lösende Belegung zu finden oder zu zeigen, dass es eine solche nicht geben kann. Wir werden diese Laufzeiten im Folgenden noch näher betrachten.

Zunächst interpretieren wir aber die violette Kurve, die den Flex-Modus Konstant repräsentiert, in dem auch die Kreuzungskanten nur 1 oder 2 Knicke haben dürfen, unabhängig davon, wie viele Kreuzungsknoten auf ihnen liegen. Es ist wenig überraschend, dass die allermeisten dieser Instanzen nicht lösbar waren. Erst dort, wo die Wahrscheinlichkeit für 2 Knicke pro Kante groß genug wird, gibt es vereinzelte Instanzen, die lösbar sind. Genau wie im Flex-Modus Wurzel steigt mit dem Anteil der lösbaren Instanzen gleichzeitig auch die durchschnittliche Laufzeit und die Anzahl der schweren Instanzen. Davor erfordert das ILP hier nur sehr wenig Laufzeit, weil die geringen Flexibilitäten schnell deutlich machen, dass keine lösende Belegung gefunden werden kann.

Wir konnten also deutlich beobachten, dass die Überprüfung der Bündelkapazitäten nur dann zu einer Erhöhung der Laufzeit führt, wenn die Flexibilitäten so gewählt sind, dass nicht ohne weiteres geklärt werden kann, ob es eine lösende Belegung gibt oder nicht. Solange dies aufgrund zu hoher oder zu niedriger Flexibilitäten leicht erkennbar ist, sind die Bündelkapazitäten ohne bedeutsamen Mehraufwand überprüfbar.

5.4.2 Aufschlüsselung der Laufzeiten des Flex-Modus der dünnen Graphen

Um die großen Schwankungen der Kurve im Flex-Modus Wurzel zu untersuchen, betrachten wir nun in Abbildung 5.2 eine Aufschlüsselung der einzelnen Laufzeiten nur dieses einen Modus. Das unterste Schaubild zeigt erneut die Aufteilung in lösbare und schwere Instanzen, diesmal als Balkendiagramm. Hier wird deutlich, dass der Anteil der schweren Instanzen dort abnimmt, wo der Anteil der lösbaren Instanzen zunimmt.

Die beiden anderen Schaubilder zeigen sogenannte Kerzendiagramme: die orangefarbene Markierung zeigt den Median, also jenen Wert, über und unter dem jeweils 50% der Werte liegen. Die oberen und unteren Kanten der blauen Balken zeigen das obere und untere Quartil, also jene Werte, über bzw. unter denen jeweils 75% bzw. 25% der Werte liegen. Das obere und untere Ende der Striche schließlich geben das Maximum und das Minimum der Laufzeiten an. Diese Darstellungsweise gibt im Gegensatz zum zuvor abgebildeten Durchschnittswert Aufschlüsse über die Verteilung der Laufzeiten.

So lässt sich am oberen Schaubild erkennen, dass es im Bereich der niedrigen Wahrscheinlichkeit noch ein großer Teil der lösbaren Instanzen ähnliche Laufzeiten hatte. Mit steigender Anzahl der lösbaren Instanzen hingegen ließen sich die meisten Instanzen in recht kurzer Zeit lösen: je näher die Quartile beieinander sind, desto größer ist der Anteil der Instanzen, deren Laufzeit in diesem kleinen Bereich liegt. Lediglich einige wenige Instanzen

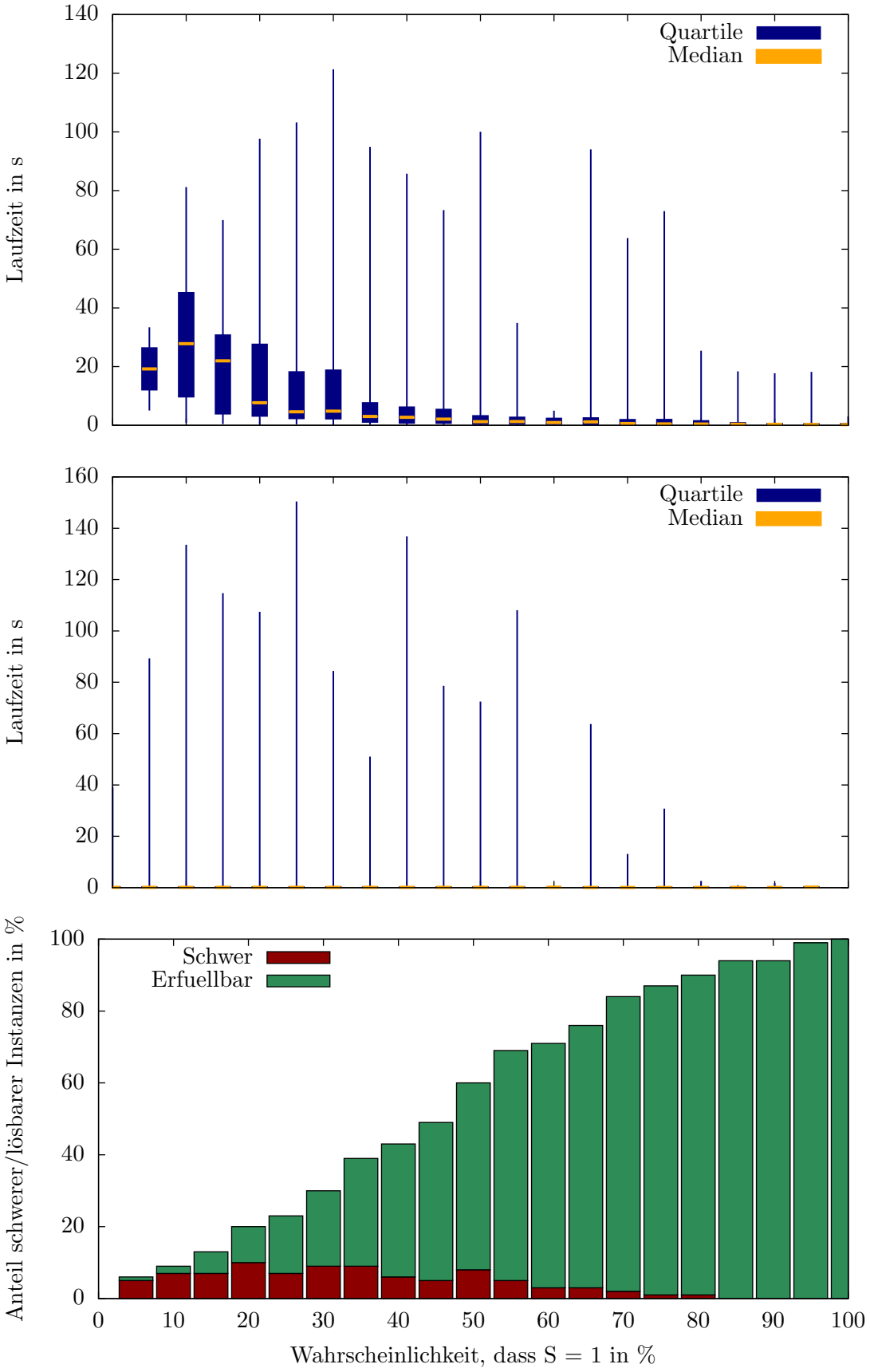


Abbildung 5.2: Die Schaubilder für Flex-Modus Log der dünne Graphen; Oben bzw. Mitte: Maximum, Oberes Quartil, Median, Unteres Quartil und Minimum unter den Laufzeiten der lösbaren bzw. nicht lösbaren Instanzen; Unten: Anteil der lösbaren Instanzen und schweren Instanzen

benötigten eine Laufzeit, die sehr viel größer als die der restlichen Instanzen war. Diese wenigen Instanzen haben aber eine deutliche Auswirkung auf die durchschnittliche Laufzeit, was die starken Schwankungen in der orangefarbenen Kurve des anfänglich betrachteten Schaubilds erklärt.

Das mittlere Schaubild verdeutlicht diesen Effekt noch mehr: hier sind sowohl der Median als auch das obere Quartil im Millisekundenbereich, und das durchweg über alle Wahrscheinlichkeiten. Die allermeisten der nicht lösbaren Instanzen stellten sich also binnen kurzer Zeit als solche heraus, lediglich bei einigen wenigen wurde mehr Laufzeit benötigt. Diese Ausreißer haben teilweise sogar so hohe Laufzeiten, dass sie als schwer klassifiziert worden wären, wenn sie noch ein wenig länger gebraucht hätten.

Es bleibt noch zu erklären, warum diese Ausreißer sowohl im lösbaren als auch im nicht lösbaren Bereich derart verteilt sind, anstatt exakt mit der Zunahme der Lösbarkeit zu korrelieren. Dies liegt daran, dass die unterschiedlichen Ausreißern von unterschliedlichen Planarisierungen hervorgerufen werden. Während bei einer Planarisierung die schwierigen Instanzen beispielsweise im Bereich 55% liegen, hat eine andere Planarisierung im Bereich 25% erhöhte Laufzeiten. Im Bereich 60% beispielsweise gab es offenbar keine Planarisierung, die Schwierigkeiten hatte, was den Einbruch der Laufzeiten an dieser Stelle erklärt. Diese Effekte würden sich natürlich mit einer Erhöhung der Anzahl untersuchter Graphen abmildern, das Schaubild würde ebenmäßiger werden. Der Effekt, dass die Laufzeiten und die Anzahl der schweren Instanzen in jenem Bereich ansteigen, in dem die Instanzen von nicht lösbar zu lösbar umschwenken, liesse sich dann auch noch deutlicher beobachten.

5.4.3 Durchschnittliche Laufzeiten auf der Klasse der dichten Graphen

Schließlich betrachten wir noch die durchschnittlichen Laufzeiten der Klasse der dichten Graphen, in Abbildung 5.3. Hier ist die Aufteilung sehr viel deutlicher als bei den dünnen Graphen: Der Flex-Modus Planar ist der einzige Modus, in dem es sowohl lösbare als auch nicht lösbare Instanzen gibt. Die Laufzeit, diese Unterscheidung zu treffen, ist wie zuvor unabhängig vom Ergebnis gering.

Nimmt man hingegen die Bündelkapazitäten hinzu, sind die Verhältnisse klar aufgeteilt: Die Flex-Modi Linear und Wurzel sind durchweg lösbar. Im Flex-Modus Linear ist dies festzustellen auf Grund der großzügigen Flexibilität in kürzester Zeit möglich: erneut bringt die Hinzunahme der Bündelkapazitäten keine Erhöhung der Laufzeit.

Im Fall des Flex-Modus Wurzel ist dies hingegen anders: Hier muss aufgrund der hohen Anzahl an Kreuzungskanten offensichtlich weitaus mehr Rechenaufwand betrieben werden, um eine lösende Belegung zu finden, als dies in den dünn besetzten Graphen im Allgemeinen der Fall war. Dies gilt insbesondere im Bereich der niedrigen Wahrscheinlichkeiten, da hier noch mehr Belegungen probiert werden müssen, bevor schließlich eine lösende gefunden wird. Die Laufzeiten steigen hier also nicht zuvor mit dem Anteil der nicht lösbaren Instanzen, der Effekt ist aber derselbe: je strikter die Bündelkapazitäten sind, desto zeitaufwändiger ist es, sie zu überprüfen. In diesem Bereich ist das Problem sogar so schwer, dass der Anteil derjenigen Instanzen, die mehr als eine Sekunde pro Kreuzungskante benötigt haben, bei 40% liegt.

Das mittlere Schaubild bekräftigt diese Aussage erneut: obwohl sich sämtliche Instanzen der Flex-Modi Log und Konstant im Endeffekt als nicht lösbar herausstellen, ist die Laufzeit nicht flach: je höher die Anzahl der Kanten mit zusätzlicher Flexibilität wird, desto mehr Rechenaufwand muss betrieben werden, um ausschließen zu können, dass es nicht doch eine lösende Belegung gibt. Selbst im Flex-Modus konstant steigt die Laufzeit gegen Ende leicht an, weil die Frage nicht mehr ohne weiteres geklärt werden kann.

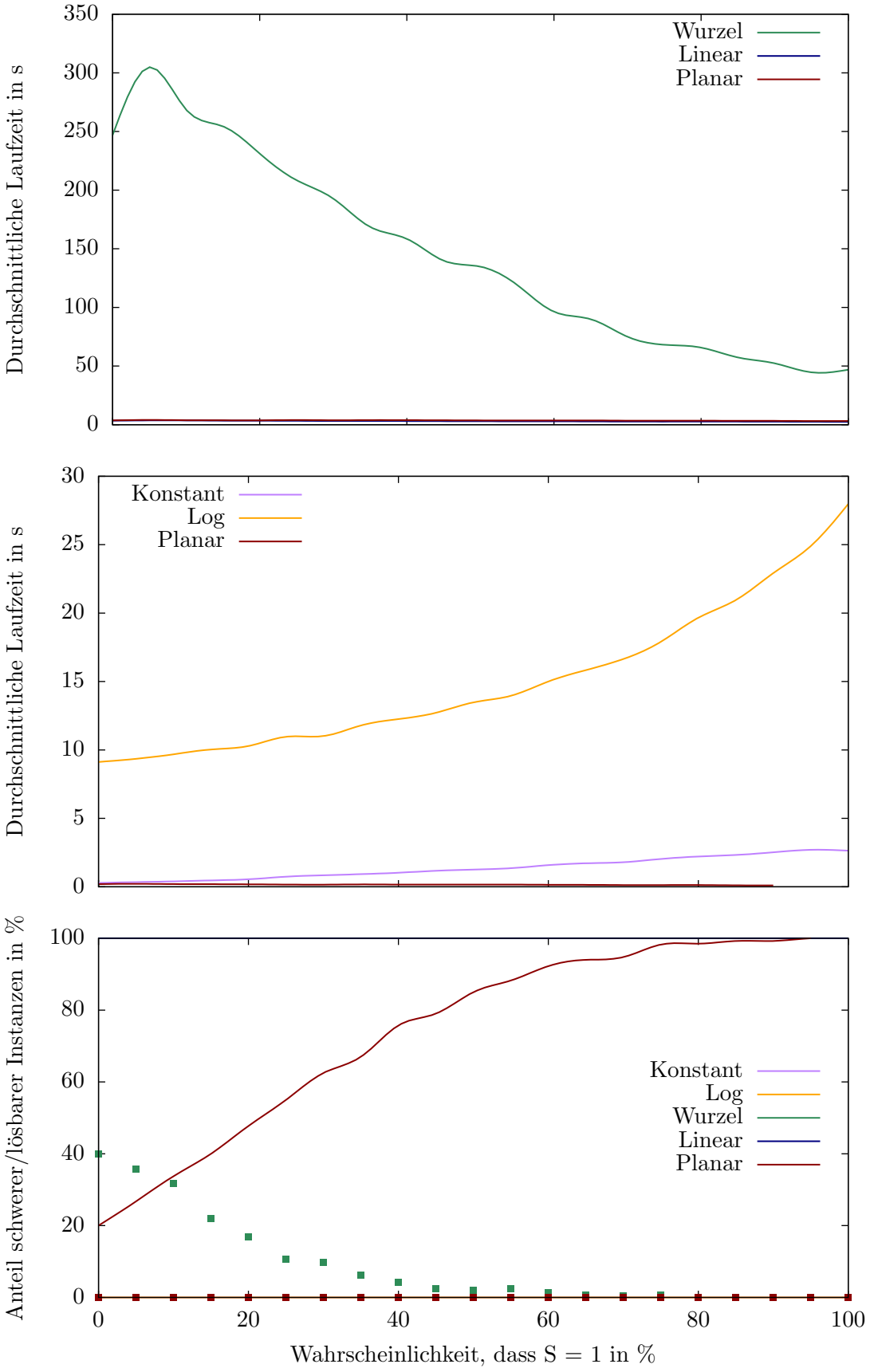


Abbildung 5.3: Die Schaubilder der dichten Graphen; Oben bzw. Mitte: Durchschnittliche Laufzeit der lösbaren bzw. nicht lösbaren Instanzen; Unten: Anteil der lösbaren Instanzen (Linien) und schweren Instanzen (Punkte)

5.5 Zusammenfassung

Wir konnten beobachten, dass das Überprüfen der Bündelkapazitäten, das unser ILP von der Untersuchung planarer Graphen unterscheidet, nicht von Natur aus zu einer Erhöhung der Laufzeit führt. Lediglich dort, wo die Flexibilitäten so gewählt sind, dass sie im Bereich der für die Lösbarkeit notwendigen Flexibilitäten liegen, sorgen sie für eine Erhöhung der Laufzeit. In diesem Bereich kann es sogar vereinzelte Instanzen geben, für die eine Überprüfung in einem praktikablen Rahmen nicht möglich ist. Dabei handelt es sich wie wir gesehen haben aber tatsächlich um vereinzelte Ausreißer, die meisten Instanzen sind auch hier in Zeiten überprüfbar, die nicht allzu sehr von denen im planaren Fall abweichen.

Die Höhe der benötigten Flexibilität lässt sich dabei nicht verallgemeinern. Sie hängt vom Anteil der Kreuzungskanten an den Kanten des Graphen ab, und somit von der Dichte des nichtplanaren Graphen. Das Entwickeln zuverlässiger Heuristiken für das Finden dieser Grenzen wäre ein naheliegender nächster Schritt.

6. Zusammenfassung und Ausblick

Wir fassen die von uns erarbeiteten Ergebnisse zusammen und geben einen kurzen Ausblick darüber, welche Fragestellungen wir für zukünftige Arbeiten eröffnet haben.

6.1 Zusammenfassung

Wir haben uns als erste wissenschaftliche Arbeit damit beschäftigt, wie die Techniken der Knickminimierung auf feste Einbettungen nichtplanarer Graphen übertragen werden können. Feste Einbettungen waren für uns dabei aus zweierlei Gründen interessant: zum einen weil ein gegebener Anwendungsfall möglicherweise eine feste Einbettung erfordert, zum anderen aber vor allem auch, weil auf diese Weise mit Hilfe der Kreuzungsminimierung eine Einbettung gewählt werden kann, die gewissen Anforderungen genügt. Vorhandene Knickminimierungsverfahren für nichtplanare Graphen hingegen beachteten derartige Anforderungen bei der Einbettung des Graphen nicht und lieferten deswegen diesbezüglich keine zufriedenstellende Ergebnisse.

Wir betrachteten also feste Planarisierungen nichtplanarer Graphen, also planare Graphen, in denen die Kreuzungskanten des nichtplanaren Graphen ersetzt wurden durch Pfade von einzelnen Kanten, den Kantensegmenten. Zusätzlich gaben wir für jede Kante des nichtplanaren Graphen eine Flexibilität an. Die zentrale Fragestellung war dabei, ob sich diese np-flex-Instanz derart zeichnen lässt, dass die Kanten des nichtplanaren Graphen höchstens so viele Knicke haben, wie ihre Flexibilität zulässt.

Zunächst zeigten wir, dass dies selbst in Planarisierungen, die nur einen einzelnen Kreuzungsknoten besitzen, nicht ohne weiteres möglich ist, wenn alle Flexibilitäten 2 sind. Dies ist deswegen bemerkenswert, weil eine Einbettung eines planaren Graphen stets mit höchstens zwei Knicken pro Kante gezeichnet werden kann. Zur Lösung dieser eigentlich recht grundlegenden Fragestellung mussten wir einige Verfahrensweisen entwickeln, die in planaren Graphen nicht nötig sind, und stellten so einige der Schwierigkeiten heraus, die das np-flex-Problem aufweist.

Anschließend demonstrierten wir, dass es unter gewissen Voraussetzungen nicht vermeidbar ist, dass eine Kreuzungskante eine gewisse Anzahl an Knicken benötigt, und zeigten so, dass viele np-flex-Instanzen unabhängig vom restlichen Aufbau der Planarisierung nicht lösbar sein können.

Wir entwickelten eine Verfahrensweise, wie zu einer gegebenen np-flex-Instanz ein modifiziertes Flussnetzwerk konstruiert werden kann, das genau dann einen gültigen Fluss

enthält, wenn die Instanz lösbar ist. Zudem erläuterten wir, dass dieser Fluss nicht wie in einem üblichen Flussnetzwerk durch Lösung eines Linearen Programms ermittelt werden kann, sondern dass stattdessen ein ganzzahliges Lineares Programm benötigt wird, welches wir in der Folge ebenfalls vorstellten.

Schließlich bewiesen wir, dass das Lösen einer np-flex-Instanz im Allgemeinen NP-schwer ist. Dies gilt selbst dann, wenn alle Flexibilitäten der Instanz mindestens 2 sind. Da dieses Entscheidungsproblem im planaren Fall sehr einfach gelöst werden kann, stellt diese Erkenntnis einen sehr wichtigen Beitrag dar.

Sowohl das modifizierte Flussnetzwerk als auch die Gadgets, die wir im Laufe der Arbeit entwickelten, stellen dabei Werkzeuge dar, die für derartige Problemstellungen so oder in ähnlicher Form nützliche Dienste erbringen. Auch die Entwicklung dieser Verfahrensweisen ist somit von Bedeutung.

Abschließend wollten wir untersuchen, wie sich die theoretische Komplexität in praxisnahen Anwendungen auswirken würde. Dazu erstellten wir Zufallsgraphen, auf denen wir das zuvor vorgestellte ILP eine Lösung der entsprechenden np-flex-Instanzen finden ließen. Als Ergebnis stellten wir fest, dass eine Erhöhung der Laufzeit nur dort auftrat, wo die verwendeten Flexibilitäten nahe an den minimal benötigten Flexibilitäten lag. Waren die Flexibilitäten hingegen so niedrig, dass das ILP leicht erkönnen konnte, dass die Instanz nicht lösbar ist oder so hoch, dass eine Vielzahl unterschiedlicher Lösungen existierte, war die Laufzeit vergleichbar mit Verfahrensweisen zur Lösung desselben Problems auf planaren Graphen. Somit konnten wir zeigen, dass unser ILP in der Praxis akzeptable Laufzeiten hat.

6.2 Ausblick

Wir haben gezeigt, dass np-flex selbst mit starken Einschränkungen an die Flexibilität NP-schwer ist. Nur wenig Bezug genommen haben wir dabei auf andere Merkmale der zu betrachtenden Instanzen. So wäre ein logischer nächster Schritt, zu untersuchen, wie sich die Komplexität verhält, wenn eine obere Grenze für die Anzahl der Kreuzungskanten oder für die Anzahl der Kreuzungsknoten auf einer Kreuzungskante gesetzt wird, oder wie eine gewisse Struktur der zugrundeliegenden Graphen genutzt werden kann.

Bei der Untersuchung des ILPs haben wir festgestellt, dass die Laufzeit desselben stark von den Flexibilitäten des Graphen abhängt. Als nächstes könnte man versuchen, Heuristiken zu entwickeln, um festzustellen, wie die Flexibilitäten am besten gewählt werden sollen. Alternativ könnte das ILP leicht derart angepasst werden, dass es die Anzahl der Knicke minimiert unter Berücksichtigung der Flexibilitäten. Zusätzlich könnte man einen Spielraum für die möglichen Flexibilitäten vorgeben und so zahlreiche Optimierungsziele gegeneinander abwägen.

Zudem sind wir in unserer Arbeit stets von einer festen Planarisierung ausgegangen, mit dem Gedanken, dass ein nichtplanarer Graph zuvor mit den Methoden der Kreuzungsminimierung eingebettet werden kann. Eine Möglichkeit wäre vielleicht, diese beiden Schritte zusammenzuführen, indem beispielsweise mehrere unterschiedliche Verfahrensweisen zur Kreuzungsminimierung angewendet werden, bevor auf jeder Einbettung die von uns vorgestellte Knickminimierung durchgeführt wird. Auf diese Weise könnte man die Anzahl der Knicke abwägen gegen die Anzahl der Kreuzungen und so Ergebnisse finden, die einen guten Kompromiss zwischen den beiden Minimierungszielen erbringen.

Literaturverzeichnis

[BK94] Therese Biedl und Goos Kant: *A Better Heuristic for Orthogonal Graph Drawings.* In: *Proceedings of the 2nd Annual European Symposium on Algorithms (ESA'94)*, Band 855 der Reihe *Lecture Notes in Computer Science*, Seiten 24–35, September 1994. http://www.springerlink.com/content/h775q2054151x16m/.

[BKRW11] Thomas Bläsius, Marcus Krug, Ignaz Rutter und Dorothea Wagner: *Orthogonal Graph Drawing with Flexibility Constraints.* In: Ulrik Brandes und Sabine Cornelsen (Herausgeber): *Proceedings of the 18th International Symposium on Graph Drawing (GD'10)*, Band 6502 der Reihe *Lecture Notes in Computer Science*, Seiten 92–104. Springer, 2011. http://dx.doi.org/10.1007/978-3-642-18469-7_9.

[BLV98] Giuseppe Di Battista, Giuseppe Liotta und Francesco Vargiu: *Spirality and Optimal Orthogonal Drawings.* SIAM J. Comput., 27(6):1764–1811, December 1998, ISSN 0097-5397. http://dx.doi.org/10.1137/S0097539794262847.

[CGJ+12] Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein und Petra Mutzel: *The Open Graph Drawing Framework (OGDF).* CRC Press, 2012. to appear.

[dBK09] Mark de Berg und Amirali Khosravi: *Finding perfect auto-partitions is np-hard.* In: *Proceedings of the 24th European Workshop on Computational Geometry (EuroCG'09)*, Seiten 255–258, 2009. http://www.win.tue.nl/~Akhosrav/papers/.

[GJ83] Michael R. Garey und David S. Johnson: *Crossing Number is NP-Complete.* SIAM Journal on Algebraic and Discrete Methods, 4:312–316, 1983.

[GM04] Carsten Gutwenger und Petra Mutzel: *An Experimental Study of Crossing Minimization Heuristics.* In: Giuseppe Liotta (Herausgeber): *Graph Drawing*, Band 2912 der Reihe *Lecture Notes in Computer Science*, Seiten 13–24. Springer Berlin Heidelberg, 2004, ISBN 978-3-540-20831-0. http://dx.doi.org/10.1007/978-3-540-24595-7_2.

[GO12] Inc. Gurobi Optimization: *Gurobi Optimizer Reference Manual*, 2012. http://www.gurobi.com.

[GT95] Ashim Garg und Roberto Tamassia: *On the Computational Complexity of Upward and Rectilinear Planarity Testing.* In: *DIAMCS International Workshop*, Band 894 der Reihe *Lecture Notes in Computer Science*, Seiten 286–297. Springer, January 1995. http://www.springerlink.com/content/p04802154821p278/.

[MdMS04] Aurora Morgana, Célia Picinin de Mello und Giovanna Sontacchi: *An algorithm for 1-bend embeddings of plane graphs in the two-dimensional grid.* Discrete

Appl. Math., 141(1-3):225–241, May 2004, ISSN 0166-218X. http://dx.doi.org/10.1016/S0166-218X(03)00373-1.

[Tam87] Roberto Tamassia: *On Embedding a Graph in the Grid with the Minimum Number of Bends*. SIAM Journal on Computing, 16(3):421–444, 1987. http://dx.doi.org/10.1137/0216030.