3-Reguläres Planares Editieren von Graphen

Bachelorarbeit
von

Julian Beckenbauer

An der Fakultät für Informatik und Mathematik
Lehrstuhl für Informatik mit Schwerpunkt Theoretische Informatik

Gutachter: Prof. Dr. Ignaz Rutter

Bearbeitungszeit: 30. April 2021 – 03. August 2021

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Passau, 30. Juli 2021

Abstract

In this thesis we consider the problem of editing a graph by inserting and deleting edges so that it becomes 3-regular and planar. If only the insertion of edges is allowed, there already exists an article by Hartmann, Rollin and Rutter [HRR14], in which a parity- and a matching-condition are introduced. If a graph fulfils these conditions, an augmentation is possible. Our goal was therefore to develop methods that establish the parity- and the matching-condition in a graph by deleting edges, so that the graph can then be augmented.
In each case, there is the restriction that reassignments are not allowed, since we show at the end of the thesis by a reduction of Monotone Planar 3-SAT that editing a graph with allowed reassignments of valences is NP-complete. Without reassignments, for connected graphs in which no degree-1 and degree-2 vertices are adjacent, with fixed embedding and maximum degree 3, we develop a polynomial-time method to satisfy the matching-condition.
Also without reassignments, we provide a polynomial-time procedure that establishes the parity-condition in connected graphs with fixed embeddings, where the graphs contain only degree-3 and single degree-2 vertices.

Deutsche Zusammenfassung

In dieser Arbeit betrachten wir das Problem, einen Graphen durch Einfügen und Löschen von Kanten so zu editieren, dass dieser 3-regulär und planar wird. Falls nur das Einfügen von Kanten erlaubt ist, existiert bereits eine Arbeit von Hartmann, Rollin und Rutter [HRR14], in der eine Parity- und eine Matching-Bedingung eingeführt werden. Erfüllt ein Graph diese, so ist eine Augmentierung möglich. Unser Ziel war es daher Verfahren zu entwickeln, welche durch Löschen von Kanten die Parity- und die Matching-Bedingung in einem Graphen herstellen, sodass dieser dann augmentierbar ist.
Bei den Verfahren existiert jeweils die Einschränkung, dass Neuzuweisungen nicht gestattet sind, da wir dafür am Ende der Arbeit durch eine Reduktion von Monotone Planar 3-SAT zeigen, dass das Editieren eines Graphen mit erlaubten Neuzuweisungen der Valenzen NP-vollständig ist.
Ohne Neuzuweisungen entwickeln wir für zusammenhängende Graphen, in denen keine Grad-1 und Grad-2 Knoten benachbart sind, mit fester Einbettung und Maximalgrad 3 ein Polynomialzeit-Verfahren, um die Matching-Bedingung zu erfüllen.
Ebenfalls ohne Neuzuweisungen liefern wir ein Polynomialzeit-Verfahren, welches die Parity-Bedingung in zusammenhängenden Graphen mit fester Einbettung herstellt, wobei die Graphen nur Grad-3 und einzelne Grad-2 Knoten enthalten.

Inhaltsverzeichnis

1. Einleitung

Das Editieren eines Graphen $G = (V, E)$ besteht darin, in G Kanten und Knoten einzufügen und aus diesem auch falls nötig zu löschen. Daraus resultiert ein neuer Graph $G' = (V', E')$. In dieser Arbeit wird betrachtet, wie G zu dem resultierende Graph $G' = (V', E')$ editiert werden kann, sodass dieser mit möglichst wenig Operationen im Anschluss die Eigenschaften planar und 3-regulär erfüllt.
Damit für jeden Knoten $v \in V$ $\deg(v) = 3$ gilt, muss $|E| = \frac{|V|*3}{2}$ vorliegen. Die Anzahl an Kanten des resultierenden Graphen ist somit genau festgelegt.
Besitzt ein Graph nur maximal Grad-3 Knoten und erfüllt bereits die Parity- und Matching-Bedingung, welche in der Arbeit von Hartmann, Rollin und Rutter [HRR14] eingeführt werden, so reicht es aus, den Graphen zu augmentieren, also nur Kanten einzufügen (vgl. Abbildung 1.1). Hierbei besagt die Parity-Bedingung, dass in jeder Facette eine gerade Anzahl an freien Graden vorhanden sein muss und die Matching-Bedingung fordert, dass die freien Grade immer paarweise verbunden werden können. Diese Augmentierung entspricht auch gleich der optimalen Lösung, da man die oben definierte Anzahl an Kanten erreichen muss. Dies ist dadurch sichergestellt, dass man nur die nötigen Kanten einfügt. Dieses Augmentieren funktioniert in $O(n^{1.5})$, aber nur falls für den Graphen eine feste Einbettung existiert. Ist die Einbettung variabel, so ist das Problem einen solchen Graphen zu editieren bereits NP-vollständig. Es ist deshalb auch nicht sinnvoll unser Problem, das Editieren von Graphen mit Löschen und Einfügen von Kanten, für eine variable Einbettung zu betrachten. Da es das Ziel ist durch das Löschen von möglichst wenig Kanten für den Graphen die Parity- und die Matching-Bedingung zu erfüllen, sodass der resultierende

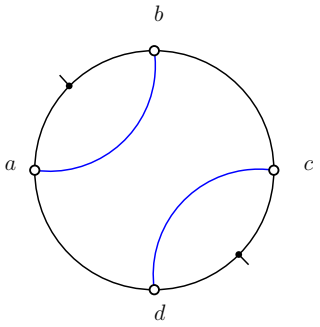


Abbildung 1.1: Graph (schwarz) mit vier Grad-2 Knoten a, b, c, d und einer Augmentierung durch die neueingefügten blauen Kanten.

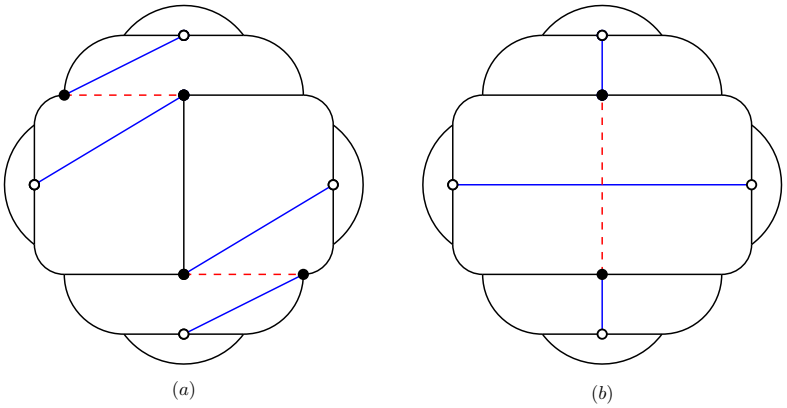


Abbildung 1.2: Graphen mit gelöschten Kanten (rot gestrichelt) und neu eingefügten Kanten (blau)
(a) Editieren ohne Neuzuweisungen (b) Editieren mit Neuzuweisungen

Graph augmentierbar ist. Dadurch entsteht die Ordnung, dass zuerst die nötigen Kanten gelöscht werden und dann der Graph augmentiert wird. Somit folgt für das Editieren auf Graphen mit variabler Einbettung, dass dies NP-vollständig ist, da schon das Augmentieren, was den zweiten Teil des Editierens darstellt, NP-vollständig ist.
Hieraus resultiert die eigentliche Fragestellung der Arbeit, welche Kanten aus einem noch nicht augmentierbaren Graphen mit fester Einbettung zu löschen sind, sodass dieser augmentierbar wird. Dabei kommt sofort die Frage auf, welche Kanten hierfür die effizientesten sind und wie mit den freiwerdenden Graden umgegangen werden soll. Dafür ist grundlegend zu entscheiden, ob man es erlaubt, dass die freiwerdenden Grade, welche beim Löschen einer Kante entstehen, neu zugewiesen werden dürfen. Abbildung 1.2 zeigt beispielhaft den Gewinn der Neuzuweisungen. In (a) sind 2 Kanten zu löschen und 4 einzufügen, also insgesamt sind 6 Operationen nötig. In b mit Neuzuweisungen reicht es 1 Kante zu löschen und 3 einzufügen. Somit kann der Graph mit $6 - 4 = 2$ Operationen weniger editiert werden.

Bevor man einen Graphen verändert, ist es oft sinnvoll, in diesem zuerst nach Mustern zu suchen, die dann beim Editieren verwendet werden können.
Dazu kann es interessant sein Teilgraphen von G zu suchen, in denen für jeden Knoten v gilt, dass dessen Grad einer Funktion $f(v)$ entspricht. Dies sind sogenannte f-factors in einem Graphen. Falls ein solcher f-factor existiert, lässt er sich in Polynomialzeit finden. Verschärft man die Fragestellung nun so, dass man einen zusammenhängenden f-factor in einem Graphen sucht, so wird das Problem NP-vollständig [DGv'H$^+$17]. Wie sich unschwer erkennen lässt, verschärft sich durch diese kleinen Änderungen der Schwierigkeitsgrad der Problemstellung erheblich, denn die Problemstellungen sind dann der Klasse der NP-vollständigen Probleme zuzuordnen und wir wissen bisher nicht, ob diese deterministisch in Polynomialzeit lösbar sind.
Genauso kann man für einen gegebene Graphen $G = (V, E)$ eine Überdeckung $H = (V, E')$ suchen, sodass jeder Knoten $v \in V$ in maximal einer Kante $e \in E'$ vorkommt, also ein Matching auf G. Ein solches Matching kann in $O(\sqrt{|V|} \cdot |E|)$ gefunden werden [MV80]. Dieses Matching-Problem soll nun auf ein B-Matching-Problem erweitert werden. Dort existiert eine Funktion $b(v)$ für $v \in V$, welche jedem Knoten einen Grad zuweist, den v im B-Matching haben soll. Auch dieses Problem ist nach Gabow [Gab18] in Polynomialzeit lösbar. Hierbei wird der Unterschied zwischen einem f-factor und einem B-Matching deutlich. Beim f-factor-Problem wählt man Knoten aus, welche bereits einem bestimmten Grad haben und beim B-Matching werden Kanten ausgewählt, sodass in der resultierenden Überdeckung an jedem Knoten ein gewisser Grad anliegt.

Nun kann man fordern, dass durch das Editieren jeder Knoten des Graphen G einen gewissen Grad hat. Dies kann wie bei unserer Problemstellung der Grad 3 sein oder man betrachtet allgemein einen Grad k. Beim Degree Constraint Editing-Probelm ist ein Graph G gegeben, zwei positive ganze Zahlen d, k und eine Funktion δ, die jedem Knoten einen Grad $\leq d$ zuweist. Die Frage ist nun, ob sich G in einen Graphen $G' = (V', E')$ mit k Operationen so umbauen lässt, dass für jeden Knoten $v \in V'$ $\deg_{G'}(v) = \delta(v)$ gilt.
Dabei ist zu unterscheiden, welche Arten von Operationen zugelassen sind. Erlaubt man nur das Einfügen und Löschen von Kanten, so ist das Problem in Polynomialzeit lösbar. Erlaubt man zusätzlich das Löschen von Knoten, so ist das Problem Degree Constraint Editing NP-vollständig [MS12, DGv'H$^+$17]. Zusätzlich kann auch noch eine Abänderung auf eine planare Variante nur mit Kanten- und Knotenlöschungen erfolgen, also auf das Problem Planar Degree Constraint Deletion. Dabei ist ein planarer Graph $G = (V, E)$ gegeben, zwei nicht negative ganze Zahlen k_v, k_e und wieder eine Funktion $\delta : V \to \mathbb{N}_0$. Gesucht ist nun ein modifizierter Graph $G' = (V', E')$, sodass jeder Knoten $v \in V'$ $\deg_{G'}(v) = \delta(v)$ erfüllt. Bei dieser Modifizierung dürfen maximal k_v Knoten und k_e Kanten gelöscht werden. Auch hier gilt wiederum, dass das Problem Planar Degree Constraint Deletion NP-vollständig ist. Dies trifft sogar für die 3-reguläre Variante zu, also für $\delta \equiv 3$ [DGv'H$^+$17].
Dies führt uns wieder zu unserer Problemstellung, dem Editieren eines Graphen mit fester Einbettung in 3-reguläre planare Form.
Dazu werden in Kapitel 2 zunächst einige Grundlagen genauer betrachtet und definiert, sodass die im Weiteren verwendeten Ausdrücke verständlich werden.
Im Anschluss beschäftigen wir uns mit einigen eingeschränkten Varianten unserer Problemstellung. Dazu werden wir im Kapitel 3 zusammenhängende Graphen mit fester Einbettung und Maximalgrad 3 betrachten, in denen keine Grad-1 und Grad-2 Knoten benachbart sind. Für diese Einschränkung entwickeln wir ein Verfahren um eine minimale Anzahl an Kanten zu löschen, sodass die Matching-Bedingung erfüllt ist. Das Neuzuweisen von Valenzen ist dabei nicht erlaubt.
Kapitel 4 liefert dann ein Verfahren, welches die Parity-Bedingung in zusammenhängenden Graphen mit fester Einbettung herstellt, wobei die Graphen nur Grad-3 und einzelne Grad-2 Knoten enthalten. Auch hier sind Neuzuweisungen nicht gestattet.
Im anschließenden Kapitel 5 begründen wir dann die in den vorherigen Kapiteln vorgenommene Einschränkung, Neuzuweisungen nicht zu erlauben. Denn bei der Zulassung von Neuzuweisungen und Grad-5 Knoten entsteht ein Editierproblem, welches Np-vollständig ist.

2. Grundlagen

Zur Vorbereitung auf die eigentliche Thematik ist es notwendig einige Begriffe zu definieren und deren Bedeutung zu erklären.
Ein Graph $G = (V, E)$ ist genau dann planar, wenn dieser in die Ebene eingebettet werden kann, ohne dass sich Kanten dieses Graphen kreuzen. Weiterhin ist G genau dann 3-regulär, wenn der Grad jedes Knoten $v \in V$ genau 3 ist. Somit folgt, dass $|E| = \frac{3 \cdot |V|}{2}$ gelten muss. Da aber $|E|$ eine ganze Zahl ist, gilt das $|V|$ eine gerade Zahl sein muss, sodass ein Editieren des Graphen nur durch Einfügen und Löschen von Kanten umgesetzt werden kann. Ansonsten müsste noch zusätzlich die Anzahl der Knoten verändert werden. In vorliegender Arbeit wird aber nur das Editieren eines Graphen in 3-reguläre und planare Form mit gerader Knotenanzahl betrachtet.

Liegt bereits ein planarer Graph vor, so kann man diesen in Facetten unterteilen. Eine Facette ist dabei ein Bereich im Graphen, welcher komplett von Kanten umschlossen ist. Abbildung 2.1 zeigt beispielhaft die für einen Graphen existierenden Facetten. Dabei stellt die grau hinterlegte Fläche explizit die Facette f_1 dar. Insgesamt existieren 5 Facetten in diesem Beispiel, wobei f_1 bis f_4 Facetten im Inneren des Graphen sind und f_5 ist die offenen Facette außerhalb des Graphen. Weiterhin zeigt die Abbildung für f_1 den Rand dieser Facette. Die grünen Kanten stellen den Rand von f_1 dar, also die Kanten, welche

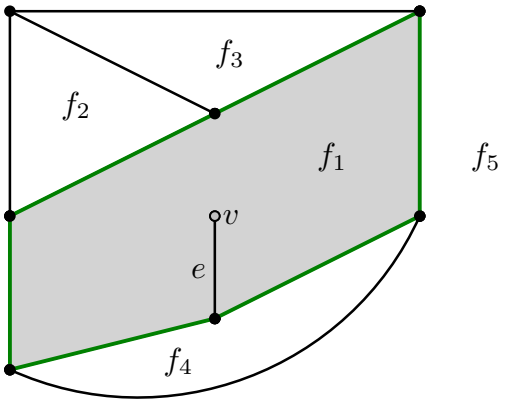


Abbildung 2.1: Beispiel eines Graphen mit eingezeichneten Facetten. Die grünen Kanten stellen den Rand der Facette f_1 dar. e ist eine innere Kante der Facette f_1 und nicht Teil des Randes.

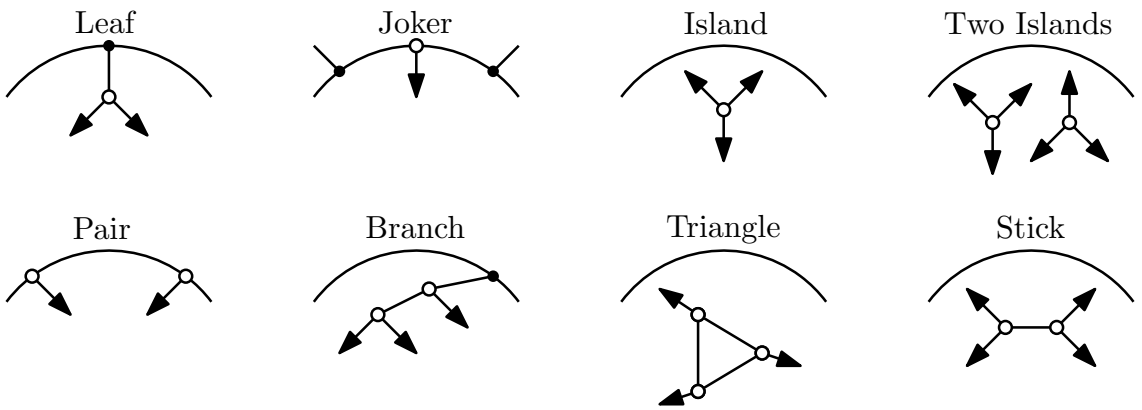


Abbildung 2.2: Indikator-Mengen

die Facette begrenzen. Für e gilt, dass diese Kante nicht im Rand von f_1 liegt, sondern im Inneren der Facette. Für die offene Facette f_5 sei der Rand definiert durch die Kanten, die die anderen Facetten von f_5 abgrenzen. Für eine Kante gilt also, dass diese entweder Teil das Randes ist oder im Inneren einer Facette liegt.

Die Abbildung 2.1 zeigt ein weiteres grundlegendes Element. Der Knoten v, dargestellt durch einen nicht ausgefüllten Kreis, hat nicht den für 3-Regularität gewünschten Grad 3. Jeder Knoten $v \in V$ mit Grad ≤ 3 wird als Valenz bezeichnet. Eine solche Valenz fordert dann, je nachdem welchen aktuellen Grad sie hat, $1 - 3$ Kanten, bietet aber auch genau so viele Anschlussstellen für neue Kanten an. Im Beispiel von v aus der Abbildung gilt, dass der aktuelle Grad von v 1 ist. Somit fordert v 2 Kanten, bietet aber auch 2 Anschlusspunkte.
Nachdem jetzt Facetten und Valenzen definiert sind, kann man die einzelnen Valenzen den einzelnen Facetten zuweisen.
Liegt eine Valenz innerhalb einer Facette, so ist die Zuweisung zu der Facette, in welcher die Valenz liegt, klar definiert. Ist die Valenz teil des Randes der Facette, so kann diese einer der beiden anliegenden Facetten zugewiesen werden.
Sind nun alle Valenzen den einzelnen Facetten zugewiesen, so entstehen Indikator-Mengen. Die Abbildung 2.2 zeigt die acht verschiedenen Indikator-Mengen.
Für Leaf und Pair gilt, dass diese Sets jeweils einen Demand von 2 haben, also 2 Kanten fordern und auch jeweils 2 Anschlusspunkte für die Kanten bieten. Beim Leaf müssen die geforderten Kanten von 2 verschiedenen anderen Valenzen kommen, wohingegen beim Pair eine andere Valenz mit 2 Anschlusspunkte ausreichen würde.
Für den Joker gilt, dass dies eine einzelnen Valenz ist, welche eine Kante, von einer anderen Valenz fordert, also einen Demand von 1 hat. Der Joker bietet auch nur genau eine Anschlussmöglichkeit.
Für Branch, Island und Triangle gilt, dass diese einen Demand von 3 haben und auch wieder jeweils 3 Anschlussmöglichkeiten stellen. Die Anzahl an Valenzen, mit welchen die nötigen Kanten gebildet werden, ist jedoch bei den drei Versionen verschieden. Dem Triangle würde eine Valenz mit 3 Anschlusspunkten genügen. Um einen Branch zufriedenzustellen, sind mindestens eine Valenz mit einer Anschlussmöglichkeit und eine mit zwei Anschlussmöglichkeiten nötig. Die Island fordert sogar 3 verschiedene Valenzen, um die Kanten bilden zu können. Dies liegt an der Tatsache, dass zwischen zwei Knoten immer maximal eine Kante existieren darf.
Der Stick hat einen Demand von 4 und stellt dafür 4 Anschlussmöglichkeiten. Die Kanten müssen von mindestens zwei verschiedenen Valenzen mit 2 Anschlusspunkten kommen.
Für Two Islands gilt, dass diese 6 Anschlussmöglichkeiten haben aber nur einen Demand

von 4, da zwischen den beiden Islands eine Kante entstehen kann, wofür keine externe Valenz nötig ist. Aber um den Demand zu erfüllen sind, wie beim Stick, 2 Valenzen mit jeweils 2 Anschlussstellen nötig.

Eine Facette kann somit verschiedene Indikator-Mengen in beliebiger Anzahl enthalten. Damit jetzt aber die Facette nur durch Einfügen von Kanten zufriedengestellt werden kann, also danach 3-regulär und planar ist, hat diese Facette einige Bedingungen zu erfüllen.
Die erste Bedingung ist die Parity-Bedingung. Diese besagt, dass der Facette eine gerade Anzahl an freien Valenzen zugeordnet sein muss, denn durch Einfügen einer Kante werden immer genau 2 Anschlussmöglichkeiten zufriedengestellt.
Die zweite Bedingung ist die Matching-Bedingung, welche besagt, dass der Demand für jede Indikator-Menge in der Facette erfüllt sein muss. Die einzelnen Valenzen können sich also gegenseitig so zugewiesen werden, dass für jede Indikator-Menge der Demand erfüllt ist.
Die dritte Bedingung ist die Planarity-Bedingung. Nach dieser müssen die Demands jedes Pfades mit $k > 2$ und jedes Kreises mit $k > 3$ Grad-2 Valenzen erfüllt sein. Dabei liegen die Grad-2 Valenzen auf dem Rand der Facette f und sind dieser zugewiesen. Ein Pfad ist eine Aneinanderreihung von Grad-2 Valenzen, wobei die erste und die letzte Valenz nicht die gleiche sein dürfen. Für den Kreis gilt, dass die erste und die letzte Valenz gleich sind.
Falls für einen planaren Graphen G eine Valenzenzuweisung existiert, sodass jede Facette die Parity- und die Matching-Bedingung erfüllt, so kann in $O(n)$ eine Zuweisung berechnet werden, womit dann auch noch die Planarity-Bedingung erfüllt ist [HRR14].
Das Editieren eines Graphen, dessen Resultat dann 3-regulär und planar ist, kann man auf einem allgemeinen Graphen betrachten, an welchen keine Einschränkungen gestellt sind. Das heißt, dass Knoten mit beliebigem Grad vorkommen können und der Graph auch in keinster Weise planar sein muss. Da dies allerdings ein sehr allgemeines Problem ist, kann man einige Einschränkungen treffen, also Eigenschaften definieren, welche der zu editierende Graph aufweisen soll.
Zuerst fordern wir, dass der zu editierende Graph bereits planar ist, denn nur dann ist es sinnvoll von Facetten zu sprechen. Ohne diese Voraussetzungen existieren in dem Graphen bereits Kantenkreuzungen, die dann gelöscht werden müssten, damit der Graph kreuzungsfrei wird. Dies führt dazu, dass für diese Kreuzung mindestens so viele Kanten gelöscht werden müssen, dass sich die Kreuzung auflöst. Dabei ist dann zu entscheiden, welche Kanten hier den meisten Mehrwert bringen.
Als weitere Einschränkung, die unabhängig von der Planaritätseinschränkung getroffen werden kann, begrenzen wir den Grad der Knoten. Da der Graph so editiert werden soll, dass dieser dann 3-regulär ist, kann man unterscheiden, ob der maximale Grad der Knoten ≤ 3 oder > 3 ist. Falls der maximale Grad der Knoten > 3 ist, existieren Knoten, welche es erfordern, dass an diesen Kanten gelöscht werden. Sei v ein Knoten mit $\deg_v = i > 3$. Dann gilt für v, dass an diesem mindestens $i - 3$ Kanten entfernt werden müssen. Welche der i Kanten aber zu entfernen sind, orientiert sich wieder daran, womit der meiste Mehrwert generiert wird. Dadurch kann sichergestellt werden, dass insgesamt eine minimale Anzahl an Kanten entfernt wird. Ist der maximale Grad ≤ 3 so fordert keiner der Knoten, dass an diesem Kanten gelöscht werden.
Betrachten wir nun die Einschränkungen, dass der Ausgangsgraph bereits planar ist und der maximale Grad ≤ 3 ist. Außerdem soll ein festes Assignment, also eine feste Zuweisung der Valenzen zu den einzelnen Facetten, vorliegen. Dadurch können wir die oben beschriebenen Indikator-Mengen sowie die Bedingungen sinnvoll betrachten. Jetzt ist es aber möglich an den Ausgangsgraphen weitere Einschränkungen zu stellen. Es kann beispielsweise gefordert werden, dass der Ausgangsgraph zusammenhängend ist, oder dass nur bestimmte Indikator-Mengen in diesem vorkommen.
Wird nun eine Kante gelöscht, so entstehen ja wieder neue Valenzen, für diese auch

Einschränkungen getroffen werden können. Einerseits kann gefordert werden, dass diese Valenzen wieder in die Facette zugewiesen werden müssen, aus welcher die Kante gelöscht wurde. Andererseits kann man auch erlauben, dass diese Valenzen beim Editieren neu zugewiesen werden, also in den benachbarten Facette verwendet werden.
Hat man wiederum die Einschränkung, dass der Ausgangsgraph bereits planar ist, der maximale Grad der Knoten ≤ 3 ist und man über ein festes Assignment verfügt, so kann man auch noch entscheiden, welche Bedingung man anstrebt zu erfüllen. Versucht man also beispielsweise nur die Parity-Bedingung oder nur die Matching-Bedingung zu lösen.
All dies sind Ansätze zur Vereinfachung des ursprünglichen Problems einen allgemeine Graphen mit möglichst wenig Löschungen in 3-regulär, planare Form zu editieren. So besteht die Möglichkeit zumindest für Teilprobleme einen Mechanismus zu finden, der in Polynomialzeit das eingeschränkte Problem löst.
Im späteren Teil der Arbeit finden sich auch für zwei dieser eingeschränkten Probleme Lösungsschemata. Für ein weiteres Problem mit weniger Einschränkungen wird dann auch gezeigt, dass dieses zur Klasse der NP-vollständig Probleme gehört.

Ist der Ausgangsgraph $G = (V, E)$ bereits planar und der maximale Grad der Knoten ist ≤ 3, dann kann es ausreichend sein, den Graphen nur durch Einfügen von Kanten zu editieren. Dies ist genau dann der Fall, wenn jede Facette die Parity- und Matching-Bedingung erfüllt und eine feste Einbettung des Graphens vorliegt. Ist die Einbettung nämlich variabel, so ist das Augmentieren schon NP-vollständig. Unter Augmentieren versteht man dabei das Überführen eines Graphen in 3-reguläre, planare Form nur durch Einfügen von Kanten. Damit ist eine weitere Einschränkung an die Ausgangsgraphen definiert. Denn wenn die Einbettung des Graphen variabel ist, brauchen wir das Editieren eines solchen Graphen gar nicht betrachten, da die Augmentierung teil der Editierung ist, womit das gesamte Editierproblem NP-vollständig ist.
Liegt nun aber eine feste Einbettung vor, so hat das Augmentieren nach gewissen Regeln zu erfolgen, sodass der resultierende Graph 3-regulär und weiterhin planar ist.
Durch die oben beschriebenen Voraussetzungen können die verschiedenen Indikator-Mengen in den einzelnen Facetten betrachtet werden, wodurch jeder Facette ein k_{max} zugewiesen werden kann. Dieses entspricht dem maximalen Demand, welcher in der Facette vorliegt. Zudem ist klar definiert, welche Knoten einer Facette zugewiesen sind. Damit kann die erste Regel definiert werden.

Regel 1.
1. *Falls $k_{max} \geq 3$, setze S als eine maximale Indikator-Menge der Facette. Wähle einen Knoten u mit minimalem Grad in S und verbinde u mit einem beliebigen zugeordneten Knoten $v \notin S$.*
2. *Falls $k_{max} = 2$ und es existiert ein Leaf u, setze $S = u$ und verbinde u mit einem zugeordneten Knoten v.*
3. *Falls $k_{max} = 2$ und es existiert kein Leaf, sondern ein Pfad xuy von Grad-2 Knoten, so wähle $S = x, u, y$. Dann verbinde u mit einem beliebigen zugeordneten Knoten $v \notin S$.*
4. *Falls $k_{max} = 2$ und es existiert weder Leaf noch ein Pfad xuy von Grad-2 Knoten, so setze als S ein Pair u, w. Verbinde u mit einem beliebigen zugeordneten Knoten $v \notin S$.*
5. *Falls $k_{max} = 1$, wähle $S = u$, wobei u ein Joker ist. Verbinde u mit einem anderen Joker v.*

Diese Regel 1 definiert das Einfügen einer Kanten in eine Facette f. Führt man diese Regel iterativ aus, so entsteht ein Ergebnis, welches aber nicht unbedingt planar ist. Um dem entgegen zu wirken, gibt es eine zweite Regel, welche aus zwei Phasen besteht.

Regel 2. *Phase 1: Falls verschiedene zusammenhängende Komponenten der Facette f Valenzen zuweisen, betrachte alle Pfade und Kreise, wie oben definiert, von Grad-2 Knoten. Dabei ist es egal, ob diese Grad-2 Knoten nur zu f oder zu zwei verschiedenen Facetten zugeordnet werden können.*

1. *Falls irgendein Pfad oder Kreis von Grad-2 Knoten mit einer Länge von mehr als k_{max} existiert, setze u als den Mittelknoten $v_{\lceil k/2 \rceil}$ eines längsten solchen Pfades oder Kreises $\pi = v_1, \dots v_k$. Verbinde u mit einem beliebigen zugeordneten Knoten v aus einer anderen Komponente.*
2. *Falls für alle Pfade oder Kreise von Grad-2 Knoten gilt, dass deren Länge maximal k_{max} ist, so wende Regel 1 an und wähle den Knoten v aus einer anderen Komponente.*

Phase 2: Alle zugewiesenen Valenzen liegen in der selben zusammenhängenden Komponente. Betrachte nur die Pfade von zugewiesenen Grad-2 Knoten, welche zu zwei verschieden Facetten zugehörig sind, also auf dem Rand der Facette liegen.

1. *Falls ein Pfad existert, der länger als k_{max} ist, wähle $u = v_k$ als den rechten Endknoten des längsten Pfades $\pi = v_1, \dots v_k$. Wähle v so, dass es die erste Valenz ist, welche durch eine Suche im Uhrzeigersinn entlang der Grenzen von f ausgehend von u gefunden wird.*
2. *Falls alle Pfade höchstens Länge k_{max} haben, wende Regel 1 an. Wähle dabei v wie folgt: Seien v_1, v_2 die ersten zu f zugewiesenen Valenzen, welche nicht zu u benachbart sind und durch eine Suche gegen beziehungsweise im Uhrzeigersinn entlang der Grenzen von f ausgehend von u gefunden werden. Falls S aus Regel 1 ein Branch ist und v_1 oder v_2 ein Grad-2 Knoten, so wähle diesen Grad-2 Knoten als v. Ansonsten setzt $v = v_1$.*

Damit ist die zweite Regel definiert, welche zum Einfügen der Kanten ausgeführt werden soll. Dabei liefert das iterative Ausführen der Regel 2 eine planare, 3-reguläre Augmentierung der Facette f. Durch Anwenden dieser Regel auf alle Facetten des Graphen G entsteht eine Augmentierung für den gesamten Graph. Die Richtigkeit dieser Regeln haben Hartmann, Rollin und Rutter [HRR14] gezeigt.
Sind nun aber in den Facetten die Parity- oder Matching-Bedingung nicht zufriedengestellt, so ist es notwendig, aus dem Graphen auch Kanten zu entfernen. Eine solche Kante ist durch ihre beiden Endpunkte charakterisiert, also die beiden Knoten, die über die Kante verbunden sind. Wird eine Kante entfernt, so wird der Grad der beiden Knoten um eins verringert. Dies definiert, wie sich das Löschen einer Kante auf den Grad eines an dieser Kante anliegenden Knotens auswirkt. Unter der Voraussetzung, dass der maximale Grad ≤ 3 ist, entsteht an beiden Knoten eine Valenz, da der Grad des Knoten dann sicher < 3 ist. Ist der Grad eines Knotens > 3 so entsteht an diesem beim Löschen keine Valenz.
Zurück zu dem Problem, dass der gegebene Graph zwar maximalen Grad 3 hat und bereits planar ist, jedoch die Parity- und Matching-Bedingung nicht erfüllt sind, wodurch das Löschen von Kanten unvermeidbar ist. Doch wie wirkt sich das Löschen einer Kante auf die in einer Facette vorkommenden Indikator-Mengen aus?
Durch das Löschen einer Kante $\{x, y\}$ sinkt nicht nur der Grad an den Knoten x, y um eins, sondern gleichzeitig erhöht sich die Forderung der Knoten x, y jeweils um eins. Beide Knoten x, y stellen eine Möglichkeit mehr dar, an welche wieder Kanten angeknüpft werden können, um damit alle Demands der Facette zu erfüllen. Darüber hinaus werden durch das Entfernen einer Kante potentiell zwei benachbarte Facetten zu einer Facette verschmolzen. Dies führt dazu, dass die offenen Demands der beiden ursprünglichen Facetten zusammenfallen und dadurch teilweise alle Demands erfüllt werden können.
Durch ein festes Assignment können Indikator-Mengen betrachtet werden und wie sich das Löschen einer Kante auf die Bedarfe der verschiedenen Indikator-Mengen auswirkt.

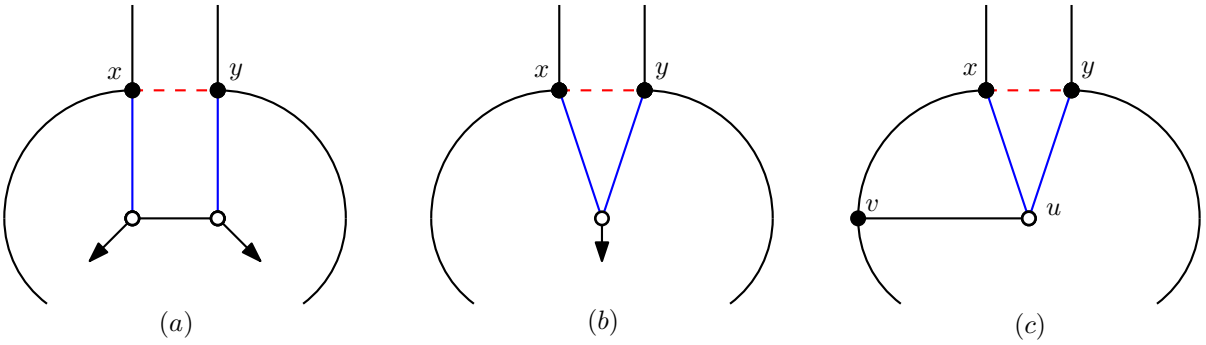


Abbildung 2.3: Beispielhaft die Interaktion einer Indikator-Menge mit einer gelöschten Kante (rot gestrichelt) und den eingefügten Kanten (blau). (a) Stick, (b) Island, (c) Leaf

Betrachten wir nun die einzelnen Indikator-Mengen im Zusammenhang mit einer gelöschten Kante. Dabei wird die Interaktion mit anderen freien Valenzen außer Acht gelassen, ebenso wie, dass durch das Löschen einer Kante eine zweite Facette mit potentiell zusätzlichen Valenzen mit eingebunden werden kann.
Sei $\{x, y\}$ die gelöschte Kante, die nicht Teil einer Indikator-Menge ist, dann stellen die beiden Knoten x, y jeweils einen zusätzlichen freien Grad bereit.
Ein Stick oder Two Islands haben jeweils einen Demand von vier. Mit den beiden freien Valenzen von x und y wird der nicht zufriedengestellte Demand auf $6 - 2 \cdot 2 = 2$ reduziert. (vgl. Abbildung 2.3 (a))
Für Branch, Island und Triangle gilt, dass diese einen Demand von 3 aufweisen. Aber diese stellen auch jeweils drei freie Grade zur Verfügung. Zusammen mit den Knoten x und y ist es also möglich den nicht zufriedengestellten Demand auf $5 - 2 \cdot 2 = 1$ zu reduzieren. Dies gelingt, da mindestens drei verschiedenen Knoten Grade zur Verfügung stellen. (vgl. Abbildung 2.3 (b))
Das Zusammenspiel von x und y mit einem Pair ergibt, dass die Demands vollständig zufriedengestellt werden können. Das Pair hat nämlich einen Demand von 2, dieser wird von den Knoten x und y genau zufriedengestellt. Ähnliches gilt für das Leaf, wobei hier noch zu beachten ist, dass das Leaf aus genau einem Knoten u besteht und genau einen benachbarten Knoten v hat. Sollte nun $x = v$ gelten, so sind nicht alle Demands erfüllt, denn dann können u und x nicht verbunden werden. Dadurch kann nur die Kante $\{u, y\}$ eingefügt werden und es liegt weiterhin ein Demand von 1 jeweils bei u und y vor. Analog gilt dies, falls $y = v$ gilt. (vgl. Abbildung 2.3 (c))
Die Situation eines Jokers v kann durch das Löschen einer Kante und die reine Betrachtung von x, y und v allerdings nicht verbessert werden. Nach dem Löschen der Kante $\{x, y\}$ stehen zusammen drei freie Grade zur Verfügung. Nach dem Einfügen einer Kante, werden allerdings die freien Grade nur auf $3 - 2 = 1$ reduziert. Somit bleibt weiterhin ein Joker übrig.
Dies verdeutlicht, dass die freien Valenzen, welche durch das Löschen einer Kante entstehen, die Demands der Indikator-Mengen nicht vergrößern, sondern je nach Indikator-Menge verringern oder sogar zufriedenstellen können. Da nun die grundlegenden Begrifflichkeiten definiert sind, können wir das erste eingeschränkte Problem betrachten, um für das Editierien von Graphen Lösungsansätze zu finden.

3. Leaf/Pair-Graphen

Um das Problem einen Graphen so zu editieren, dass dieser dann 3-regulär und planar ist, in den Griff zu bekommen, betrachten wir zunächst einmal ein eingeschränktes Problem. Für die zu editierenden Graphen soll gelten, dass diese bereits planar sind und die Knoten einen maximalen Grad von 3 aufweisen. Weist ein Knoten nicht Grad 3 auf, soll gelten, dass diese genau Teil eines Leafs oder Pairs ist, sodass nicht ein Knoten zu zwei verschiedenen Pairs zugewiesen werden kann. Außerdem existiert eine feste Einbettung. Nachfolgend bezeichnen wir diese Art von Graphen als Leaf/Pair-Graphen.
Zusätzlich soll in jedem Graphen ein festes Assignment gelten, also eine feste Zuweisung der freien Valenzen zu einer bestimmten Facette. Dies bedeutet, dass ein Leaf der Facette zugeordnet wird, in welcher es liegt, ein Pair einer der beiden angrenzenden Facetten. Außerdem dürfen in dem Graphen keine Valenzen neu zugewiesen werden. Somit müssen die beiden Valenzen, die durch das Löschen einer Kante frei werden, wieder der Facette, aus der die Kante entfernt wurde, beziehungsweise der verschmolzenen Facette zugewiesen werden.
Das folgende Kapitel liefert ein Verfahren, wie man in solchen Graphen die Matching-Bedingung erfüllen kann. Da die Valenzen nur in Leafs und Pairs vorkommen, wird einer Facette stets eine gerade Anzahl an Valenzen zugewiesen, weshalb die Parity-Bedingung bereits erfüllt ist. Dies wird auch durch das Löschen einer Kante nicht verändert.

3.1 Passende Kante für nicht zufriedengestellte Facetten

In den betrachteten Graphen tauchen Valenzen nur mehr in Form von Pairs und Leafs auf und sind jeweils einer festen Facette zugeordnet. Dies führt uns zur Frage, welche Kombinationen von Pairs und Leafs innerhalb einer Facette existieren können, sodass die Matching-Bedingung in dieser Facette nicht erfüllt ist. Betrachten wir beispielsweise eine Facette, die drei Leafs u, v und w enthält, dann existiere insgesamt $3 \cdot 2 = 6$ Anschlusspunkte. Dadurch können drei Kanten $\{u, v\}, \{v, w\}$ und $\{w, u\}$ eingefügt und die einzelnen Demands erfüllt werden. Die eingefügten Kanten bilden mit den Knoten einen Kreis, da jedes Leaf sich mit zwei anderen Leafs verbindet. Dies gilt auch, wenn mehr Leafs einer Facette zugeordnet sind, der entsprechende Kreis wird dann größer.
Analog ist es möglich in dieses Konstrukt ein beziehungsweise mehr Pairs einzubinden. Sei das Pair, bestehend aus den Knoten (x, y), ebenfalls der Facette zugewiesen. Dann sind $\{u, v\}, \{v, w\}, \{w, x\}$ und $\{y, u\}$ beispielsweise die eingefügten Kanten, durch welche der

gesamte Demand von acht erfüllt ist. Die selbe Konstruktion lässt sich mit einer Facette durchführen, die zwei oder mehr Pairs enthält. Diese Erkenntnis resultiert nun in folgendem Lemma.

Lemma 3.1. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden.*
Der gesamte Demand einer Facette f von G kann genau dann ohne das Löschen einer Kante aus f nicht zufriedengestellt werden, wenn die Facette genau ein oder zwei Leafs oder genau ein Pair enthält.

Beweis. „⇒“ Falls die Facette f weder ein Leaf noch ein Pair enthält, so ist auch kein offener Demand vorhanden, der in der Facette erfüllt werden muss. Falls in f mindestens drei Leafs enthalten sind, so gilt, dass der Demand $D(f) \geq 6 = 3 \cdot 2$ und zusätzlich gerade ist. Somit ist es möglich $D(f)/2$ Kanten in f einzufügen, da auch $D(f)/2$ Knoten existieren, die Grad kleiner drei haben. Falls in f mindestens zwei Pairs enthalten sind, so gilt $D(f) \geq 4 = 2 \cdot 2$ und $D(f)$ ist gerade. Deshalb können $D(f)/2$ Kanten in f eingefügt werden, weil genug verschiedene Valenzen, nämlich $D(f)$, vorhanden sind. Zuletzt bleibt noch der Fall, dass sowohl mindestens ein Leaf und mindestens ein Pair in der Facette f enthalten sind. Auch hier gilt wieder, dass $D(f) \geq 4 = 2 \cdot 2$ gilt und $D(f)$ gerade ist. Somit folgt mit der selben Argumentation, dass der Demand innerhalb von f komplett gelöst werden kann.
Dies beweist, mit Ausnahme der drei spezialisierten Fälle, dass die Demands von f direkt innerhalb der Facette f gelöst werden kann, ohne dass eine Kante aus f gelöscht werden muss.
„⇐“ Für die Rückrichtung betrachten wir drei verschiedenen Fälle.
Fall I (1 Leaf): Ein Leaf hat einen Demand von zwei. Dieser Demand kann aber ohne weitere freie Valenzen nicht erfüllt werden.
Fall II (2 Leafs): Zwei Leaf haben zusammen einen Demand von vier. Es kann aber zwischen den beiden Leafs nur eine Kante eingefügt werden, da sonst zwei Kanten zwischen den gleichen Knoten existiern müssten. Somit kann der Demand maximal auf $4-2 = 2$ reduziert werden. Dieser Demand von zwei kann ohne weitere Valenzen, welche in der Facette nicht vorhanden sind, nicht weiter verringert werden.
Fall III (1 Pair): Das Pair hat auch einen Demand von zwei. Da aber dieser Demand ebenfalls nicht ohne weitere freie Valenzen zufriedenzustellen ist, bleibt dieser auch hier übrig. □

In jedem der drei Fälle können zwei Anschlusspunkte nicht innerhalb der Facette zufriedengestellt werden, womit die Matching-Bedingung nicht erfüllt ist.
Betrachten wir nun die drei Fälle noch einmal detaillierter:
Fall I (genau 1 Leaf kommt in der Facette vor): Vergleiche hierzu das Beispiel in Abbildung 3.1 (a). Um den gesamten Demand des Leafs u zu erfüllen, ist es nötig, aus der Facette f, in welcher u liegt, eine passende Kante zu löschen. Das nachfolgende Korollar beschreibt, dass das Löschen einer nicht am Nachbarknoten des Leafs anliegenden Kante in der gleichen Facette den Demand des Leafs erfüllt. Dabei wird außer Acht gelassen, dass das Löschen einer Kante potentiell eine Verbindung zu einer benachbarten Facette schafft, welche wiederum freie Valenzen mitbringt, die den Demand des Leafs u erfüllen.
Würde man nämlich eine Kante der Form $\{v, w\}$ entfernen, also eine Kante, die am Nachbarknoten von u, also an v anliegt, kann der Demand des Leafs nicht zufriedengestellt werden, da v und u bereits verbunden sind. Somit kann nur mit Hilfe von w der Demand des Leafs auf $2 - 1 = 1$ reduziert werden. Durch das Entfernen einer anderen Kante aus der Facette kann der Demand auf $2 - 2 \cdot 1 = 0$ verringert werden.

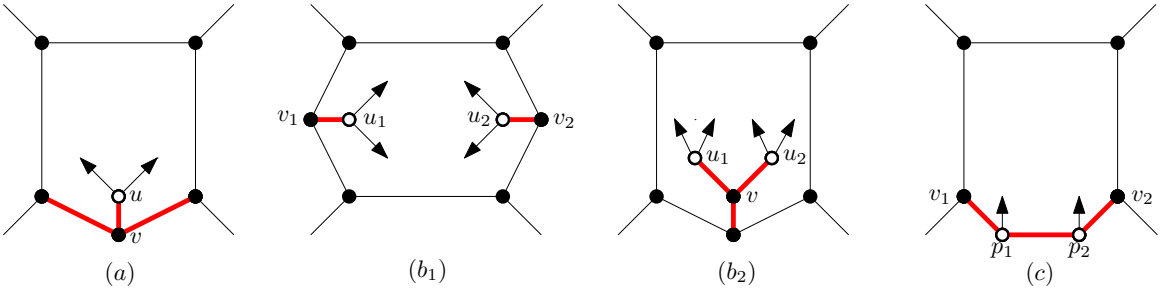


Abbildung 3.1: Die vier Graphen zeigen die drei verschiedenen Fälle und den Spezialfall. Die roten Kanten sind jeweils die Kanten, welche nicht zum Entfernen geeignet sind. (a) ein Leaf, (b_1) zwei Leafs, (b_2) zwei Leafs Spezialfall, (c) ein Pair

Korollar 3.2. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden. Sei $u \in V$ ein Leaf mit Nachbarknoten v in einer Facette f. Sei $e = \{x, y\} \in E$ Teil der Facette f mit $x \neq v \wedge y \neq v$.*
Dann gilt, dass das Löschen von e aus dem Graphen den Demand des Leafs u zufriedenstellt.

Das Korollar zeigt, dass der erste Fall durch das Löschen einer Kante aus der Facette zufriedengestellt werden kann.
Fall II (genau 2 Leafs kommen in der Facette vor): Die beiden Leafs haben, wie bereits festgestellt, miteinander einen noch offene Demand von zwei, der innerhalb der Facette f nicht zufriedengestellt werden kann. Wiederum wird nur die eine Facette für sich betrachtet und keine globale Effekte miteinbezogen. Weiterhin betrachten wir wie in Abbildung 3.1 (b_1) die beiden Leafs u_1, u_2 mit Nachbarknoten v_1, v_2. Dabei soll aber $v_1 \neq v_2$ gelten, den Spezialfall $v_1 = v_2$ betrachten wir im Anschluss gesondert. Wird nun eine passende Kante e gelöscht, so werden zwei Valenzen erzeugt. Da nicht beide Valenzen gleichzeitig eine Verbindung zu einem der Leafs aufweisen können oder nach Voraussetzung eine der Valenzen sowohl mit u_1 als auch mit u_2 verbunden sein kann, ist es immer möglich, jedem der zwei Leafs eine der Valenzen zuzuweisen. Daraus resultiert folgendes Korollar.

Korollar 3.3. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden. Seien $u_1, u_2 \in V$ zwei Leafs mit entsprechenden Nachbarknoten $v_1, v_2 \in V$ mit $v_1 \neq v_2$ in einer Facette f. Seien die Leafs über die Kanten $\{u_1, v_1\}, \{u_2, v_2\} \in E$ angebunden. Sei $e \in E \setminus \{\{u_1, v_1\}, \{u_2, v_2\}\}$ Teil der Facette f.*
Dann gilt, dass das Löschen von e aus dem Graphen den Demand der Leafs u_1, u_2 zufriedenstellt.

Dieses Korollar zeigt also, dass der offene Demand bei zwei Leafs bis auf den Spezialfall erfüllt werden kann, indem eine passende Kante aus der Facette f entfernt wird.
Betrachten wir nun noch den Spezialfall, wie in Abbildung 3.1 (b_2) dargestellt. Dabei haben sowohl u_1 als auch u_2 als benachbarten Knoten v, das heißt die Leafs sind über die Kanten $\{u_1, v\}$ und $\{u_2, v\}$ in den Graphen eingebunden. Dies erfordert eine weitere Einschränkung der möglichen zu entfernenden Kanten aus der Facette. Nun gilt für den Knoten v, dass dieser noch über eine weitere Kante $\{v, w\}$ mit $w \neq u_1 \wedge w \neq u_2$ verfügt. Um den gesamten Demand der Facette f ohne globale Interaktion mit anderen Facetten zu erfüllen, dürfen keine der Kanten $\{u_1, v\}, \{u_2, v\}, \{v, w\}$ die gelöschte Kante e darstellen. Daraus lässt sich folgern, dass e eine Kante sein muss, die v nicht als Knoten enthält. Denn würde v durch das Löschen einer Kante eine freie Valenz werden, kann die Matching-Bedingung nicht erfüllt werden. Denn entweder kann v keinem der Leafs zuweisen werden, da v bereits mit Beiden verbunden ist, oder andererseits, falls $\{u_1, v\}$ oder $\{u_2, v\}$ e darstellen, dann müsste genau die gelöschte Kante wieder eingefügt werden, was aber keinen Mehrwert bringt.

Korollar 3.4. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden. Seien $u_1, u_2 \in V$ zwei Leafs mit gleichem Nachbarknoten $v \in V$ in einer Facette f. Seien die Leafs über die Kanten $\{u_1, v\}, \{u_2, v\} \in E$ angebunden. Sei $e = (x, y) \in E$ mit $x \neq v \wedge y \neq v$ Teil der Facette f.*
Dann gilt, dass das Löschen von e aus dem Graphen den Demand der Leafs u_1, u_2 zufriedenstellt.

Insgesamt ist es also auch für den zweiten der drei Fälle durch das Entfernen einer passende Kante aus der Facette möglich, den Demand der Facette zufriedenzustellen und die Matching-Bedingung zu erfüllen.
Fall III (genau 1 Pair kommt in der Facette vor): Als Beispiel kann hier der Graph (c) der Abbildung 3.1 betrachtet werden. Auch hier erfüllen wieder einige Kanten der Facette nicht die Anforderung, dass ihr Entfernen ohne globale Interaktion den Demand der Facette erfüllt. Denn betrachten wir ein Pair (p_1, p_2) mit den jeweils dazu gehörenden benachbarten Knoten v_1 beziehungsweise v_2, dann sind diese über die Kanten $\{p_1, v_1\}$ und $\{p_2, v_2\}$ verbunden. Entfernt man nun eine der Kanten $\{p_1, v_1\}$, $\{p_2, v_2\}$ oder $\{p_1, p_2\}$, so ergibt sich, dass der Demand innerhalb der betrachteten Facette nicht erfüllt werden kann. Somit ist die Matching-Bedingung nicht erfüllt, da sonst nur eine Kante entfernt werden würde, die bereits in Verbindung zu mindestens einem der Pair-Knoten gestanden hat. Beim Entfernen der Kante e werden dann nicht genug Valenzen erzeugt, die nicht bereits Teil des Pairs sind, sodass der Demand nicht innerhalb der Facette gelöst werden kann. Um die Matching-Bedingung zu erfüllen, ist also eine andere Kante der Facette zu wählen.

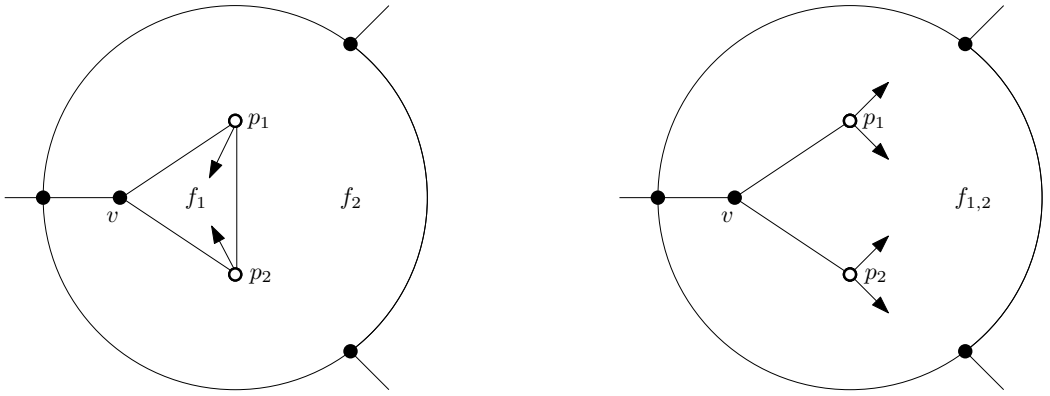
Korollar 3.5. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden. Seien $p_1, p_2 \in V$ ein Pair mit Nachbarknoten $v_1, v_2 \in V$ in einer Facette f. Sei das Pair durch die Kante $\{p_1, p_2\} \in E$ definiert und die Nachbarknoten über $\{p_1, v_1\}, \{p_2, v_2\} \in E$ angebunden. Sei $e = (x, y) \in E \setminus \{\{p_1, v_1\}, \{p_2, v_2\}, \{p_1, p_2\}\}$ Teil der Facette f.*
Dann gilt, dass das Löschen von e aus dem Graphen den Demand des Pairs (p_1, p_2) zufriedenstellt.

Die vorangegangenen vier Korollare zeigen, dass das Löschen von maximal einer passenden Kante aus der Facette f alle Demands in f zufriedenstellt. Eine passende Kante ist dabei jeweils eine Kante, die je nach Fall, wie in den obigen Korollaren beschrieben, richtig gewählt ist. Somit kann die Matching-Bedingung pro Facette erfüllt werden.

Korollar 3.6. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden.*
Weist eine Facette f genau einen der drei Fälle (genau ein Leaf, genau zwei Leafs oder genau ein Pair) auf, so kann der Demand $D(f)$ der Facette f ohne globale Interaktion mit anderen Facetten zufriedengestellt werden, indem eine passende Kante $e = \{x, y\}$ aus der Facette f gelöscht wird.

Beweis. Folgt sofort aus den Korollaren 3.2, 3.3, 3.4 und 3.5. □

Dieses Korollar liefert, dass es für eine Facette f ausreicht maximal eine passende Kante aus dieser zu entfernen, sodass der Demand der Facette erfüllt ist, ohne dass das Zusammenspiel zwischen einzelnen Facetten betrachtet wird.



(a) (b)

Abbildung 3.2: Der Graph (a) zeigt die Situation, dass die Facette f_1 nur aus drei Knoten besteht und dabei f_1 ein Pair zugeordnet ist. Graph (b) zeigt den Umgang mit solch einer Facette und das Entfernen der Kante p_1, p_2.

3.2 Existenz einer passenden Kante

Da nun gezeigt ist, dass eine passende Kante für eine Facette ausreicht, stellt sich die Frage, ob für jeden der Fälle eine passenden Kante e in der jeweiligen Facette f existiert, die entfernt werden kann. Betrachten wir hierzu das Konstrukt, welches beispielhaft in Abbildung 3.2 (a) dargestellt ist. Dabei besteht eine Facette f_1 genau aus drei Knoten $\{p_1, p_2, v\}$. f_1 ist das Pair (p_1, p_2) zugewiesen. Dies bedeutet, dass aus f_1 eine Kante entfernt werden muss, um den Demand $D(f_1) = 2$ zu erfüllen. Allerdings tritt das Problem auf, dass keine der drei Kanten von f_1 nach den obigen Regeln eine passende Kante ist. Dieser Sonderfall soll im Folgenden betrachtet werden.
Das Löschen einer der drei Kanten kann den Demand nach Korollar 3.5 nicht zufriedenstellen. Dies liefert, dass mit dem Löschen einer Kante die Matching-Bedingung für die Facette f_1 nicht erfüllt ist. Somit muss eine solche Facette mit einer anderen Facette interagieren. Da der Graph zusammenhängend ist, muss die Facette über den Knoten v in den Graphen eingebunden sein. Dies liefert aber zugleich, dass f_1 in genau einer anderen Facette f_2 liegt. Entfernt man nun die Kante $\{p_1, p_2\}$, so entstehen aus dem Pair zwei Leafs p_1, p_2, wie in Teil (b) der Abbildung 3.2 gezeigt. Diese beiden Pairs hängen an dem Nachbarknoten v. Dies entspricht genau dem Sonderfall, dass zwei Leafs an einem gemeinsamen Nachbarknoten hängen, welcher im Korollar 3.4 behandelt wird. Somit bleibt zu zeigen, dass für den Graphen G genau dann die Matching-Bedingung mit k gelöschten Kanten erfüllt werden kann, wenn für G', also den Graphen, der die Kante $\{p_1, p_2\}$ nicht enthält, die Matching-Bedingung mit $k - 1$ gelöschten Kanten erfüllt werden kann .

Lemma 3.7. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden. Sei (p_1, p_2) ein Pair, welches einer Facette f_1, bestehend aus den drei Knoten $\{v, p_1, p_2\}$ zugewiesen ist. Sei zudem f_2 die Facette, in welcher f_1 liegt. Sei $G' = (V, E \setminus \{p_1, p_2\})$.*
Für den Graphen G existiert genau dann eine Lösung mit k zu entfernenden Kanten, wenn für G' eine Lösung mit $k - 1$ zu entfernenden Kanten existiert.

Beweis. „⇒“ Angenommen es existiert für $G = (V, E)$ eine Lösung mit k zu entfernenden Kanten, sodass die Matching-Bedingung für alle Facetten von G erfüllt ist. Für die Facette f_1 gilt, dass diese den Demand $D(f_1) = 2$, welcher durch das Pair (p_1, p_2) entsteht, nicht innerhalb der Facette zufriedenstellen kann. Daraus folgt, dass aus der Facette f_1 mindestens eine Kante gelöscht werden muss, um diese in die Facette f_2 mit einzubinden.

Dabei existieren die drei möglichen Kanten $\{\{p_1, p_2\}, \{p_1, v\}, \{p_2, v\}\}$, die aus f_1 entfernt werden können. Durch das Entfernen der Kante $\{p_1, p_2\}$ entstehen aus dem Pair zwei Leafs p_1, p_2. Dies passt wieder in die Struktur, dass im Graphen nur Valenzen in Form von Pairs und Leafs enthalten sind. Das Entfernen einer der anderen beiden Kante erzeugt einen Joker und einen Branch. Dies liefert insgesamt eine Demand von zwei, was nicht besser ist als der von zwei Leafs. Daraus ergibt sich kein Vorteil in der Anzahl an zu löschenden Kanten insgesamt. Damit kann ohne Beschränkung der Allgemeinheit die Kante $\{p_1, p_2\}$ entfernt werden, wodurch der Graph G' entsteht. Da gilt, dass für eine Lösung für G k Kanten entfernt werden müssen, folgt, dass für G' eine Lösung mit $k-1$ Kanten existiert. Denn die Kante $\{p_1, p_2\}$, welche aus G entfernt wird, existiert in G' nicht.
„⇐“ Angenommen es existiert für $G' = (V, E \backslash \{p_1, p_2\})$ eine Lösung mit $k-1$ zu entfernenden Kanten, sodass die Matching-Bedingung für alle Facetten von G' erfüllt ist. Somit gilt für die Facette f_2, welche mindestens die zwei Leafs p_1, p_2 enthält, dass die Matching-Bedingung erfüllt ist, wenn aus G' $k-1$ Kanten entfernt sind. G' entsteht aus G durch das Entfernen der Kante $\{p_1, p_2\}$. Dies liefert, dass für G eine Lösung mit der entfernten Kante $\{p_1, p_2\}$ und den $k-1$ Kanten, die für G' nötig sind, existiert. Also insgesamt sind es $k-1+1 = k$ Kanten. □

Da nun gezeigt ist, dass der Spezialfall aufgelöst werden kann, bleibt der allgemeine Fall zu beweisen. Dieser besagt, dass in allen Facetten, in denen der Demand nicht innerhalb der Facette zufriedengestellt werden kann, eine passenden Kante existiert. Betrachten wir hierzu die kleinstmögliche Größe einer Facette. Hierfür gilt aufgrund der 3-Regularität, dass der Rand einer Facette aus mindestens drei verschiedenen Knoten bestehen muss und somit auch aus mindestens drei Kanten.
Da der Spezialfall für diese Größe bereits betrachtet ist und dieser den Fall des Pairs abdeckt, bleiben noch die Fälle, die nur Leafs enthalten.
Betrachten wir zuerst den Fall, dass nur genau ein Leaf in der Facette enthalten ist. Dieses Leaf hängt an einem der Randknoten der Facette. Nun gilt, dass zwei der drei Kanten nicht passend sind, was aber umgekehrt bedeutet, dass die dritte Kante eine passende ist. In dem Fall, dass genau zwei Leafs in der Facette enthalten sind, wird keine der Randkanten der Facette als unpassend deklariert. Es existiert also auch hier in jedem Fall eine passende Kante. Daraus resultiert nun folgendes Lemma.

Lemma 3.8. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden. Sei f eine Facette, deren Matching-Bedingung nicht erfüllt ist. Sei e eine Kante der Facette f.*
Für f existiert mindestens eine passende Kante e, sodass das Entfernen von e die Matching-Bedingung für f erfüllt, ohne freien Valenzen anderer Facetten zu verwenden.

Beweis. Betrachten wir hierzu wieder die drei Fälle, in denen eine Facette f ihren Demand nicht innerhalb zufriedenstellen kann.
Fall I (1 Leaf): Annahme der Rand von f besteht aus nur drei Kanten $\{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}\}$ (kleinstmögliche Facette) und die Facette enthält genau ein Leaf u. Ohne Beschränkung der Allgemeinheit kann angenommen werden, dass u als benachbarten Knoten v_1 hat, da der benachbarte Knoten aufgrund der Einschränkungen des Graphen ein Randknoten der Facette sein muss. Nach Korollar 3.2 ist keine Kante passend, die v_1 enthält, also nicht die Kante$\{u, v_1\}$ und die beiden anderen Kanten an v_1, welche Teil des Randes sind. Da eine Facette aber mindestens drei Randkanten besitzt, existiert mindestens eine Kante welche passend ist. In diesem speziellen Fall wäre dies $e = \{v_2, v_3\}$. Sei r die Anzahl an Randkanten, so existieren $r-2$ passende Kanten.
Fall II (2 Leafs): Nach den Korollaren 3.3 und 3.4 folgt, dass alle Kanten des Randes der Facette passend sind. Somit existiert mindestens eine passende Kante e.

Fall III (1 Pair): Lemma 3.7 deckt den Spezialfall ab, dass f genau drei Knoten und darunter ein Pair zugeordnet sind. Sei nun die Anzahl r der Randkanten von f mindestens vier. Nach Korollar 3.5 folgt, dass genau drei Kanten des Randes von f nicht passend für e sind. Damit gilt, dass $r - 3 \geq 4 - 3 = 1$ passende Kanten in f existieren.
Somit ist für alle Fälle gezeigt, dass eine passende Kante e in der Facette f existiert. □

3.3 Matching auf dem Dualgraphen

Um nun die Anzahl an zu entfernenden Kanten zu minimieren, betrachten wir die Auswirkungen von benachbarten Facetten f_1, f_2, deren Matching-Bedingung jeweils nicht erfüllt ist und die über mindestens eine gemeinsame Kante verfügen.
Wir betrachten nun also benachbarte Facetten, in denen jeweils einer der obigen drei Fälle vorliegt. Nach dem Entfernen einer Kante e, die beiden Facetten als Randkante dient, enthält die neue Facette $f_{1,2}$ die Demands, welche in f_1 und in f_2 vorlagen, sowie die beiden Valenzen, welche beim Entfernen von e entstehen. Also folgt insgesamt $D(f_{1,2}) = D(f_1) + D(f_2) + 2$. Nachdem der nicht zufriedengestellte Demand von f_1 und f_2 jeweils 2 ist und das Entfernen von e zwei weitere Valenzen liefert, folgt, dass insgesamt 6 nicht zufriedengestellt Anschlussstellen existiert, von mindestens drei verschiedenen Knoten. So kann die Matching-Bedingung insgesamt durch das Entfernen von e für die beiden Facette f_1, f_2 erfüllt werden.

Lemma 3.9. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden. Seien f_1, f_2 Facetten mit $f_1 \neq f_2$ und nicht erfüllter Matching-Bedingung. Weiterhin sollen f_1, f_2 über mindestens eine gemeinsame Kante $e = \{x, y\}$ verfügen. Sei $f_{1,2}$ die Facette, welche aus f_1, f_2 und dem Entfernen von e entsteht.*
Dann gilt, dass die Matching-Bedingung innerhalb der Facette $f_{1,2}$ erfüllt ist.

Beweis. Da für die Facetten f_1, f_2 jeweils nach Voraussetzung die Matching-Bedingung nicht erfüllt ist, liegt bei beiden jeweils einer der obigen drei Fälle (1 Leaf, 2 Leafs, 1 Pair) vor. Weiterhin fügt das Entfernen von e aus G zwei freie Valenzen hinzu. Somit gilt insgesamt, dass für den Demand $D(f_{1,2}) = D(f_1) + D(f_2) + 2 \geq 2 + 2 + 2 = 6$ gilt. Nach den Voraussetzungen folgt $D(f_1), D(f_2) \in \{2, 4\}$ Damit ist $D(f_{1,2})$ immer gerade.
Falls nun jede der Facetten f_1, f_2 genau ein Leaf u_1 beziehungsweise u_2 enthält, so folgt, dass $f_{1,2}$ dann zwei Leafs u_1, u_2 enthält und zudem zwei freie Valenzen an x und y. Dabei gilt $u_1 \neq x$, $u_1 \neq y$, $u_2 \neq x$ und $u_2 \neq y$, da e eine Randkante der Facetten ist und ein Leaf nach Definition nicht auf dem Rand liegen kann. Somit existiert in $f_{1,2}$ 6 Anschlusspunkte von 4 vier verschiedenen Valenzen. Dies erfüllt die Matching-Bedingung.
In den anderen Fällen entsteht eine Facette, die mindestens ein Leaf und ein Pair oder mindestens drei Leafs oder mindestens zwei Pairs enthält. Zudem sind die beiden Valenzen x und y vorhanden. Da nun gilt, dass die Anzahl an freien Anschlussstellen von $f_{1,2}$ gerade ist und da für die Anzahl a an verschiedenen Valenzen gilt, dass $a \geq D(f_{1,2}) \div 2$ ist, folgt, dass die Matching-Bedingung in $f_{1,2}$ erfüllt ist. □

Dieses Lemma zeigt, dass durch das Verbinden von zwei Facetten, bei denen jeweils der Demand nicht innerhalb zufriedengestellt werden kann, die eine entfernte Kante ausreicht um die Matching-Bedingung zu erfüllen. Im Vergleich zu einer einzeln betrachteten Facette spart man hier eine Kante ein, welche sonst gelöscht werden würde. Weiterhin gilt, dass das paarweise Verbinden von Facette die effektivste Methode darstellt, um Kanten zu sparen. Denn sollte man drei Facetten miteinander verbinden wollen, muss man mindestens zwei Kanten löschen. Dies entspricht aber der Anzahl an Kanten, die man erhält, wenn man zwei Facetten verbindet und die dritte Facette einzeln betrachtet.

Für das folgende Lemma untersuchen wir Graphen, welche über keine Facetten verfügen, die nur aus drei Knoten bestehen und dabei zwei der drei Knoten ein Pair darstellen, dessen freie Valenzen der Facette zugeordnet sind. Diese Annahme kann nach Lemma 3.7 getroffen werden, da diese Facetten als Sonderfall extra behandelt wurden.
Nun enthält G keine Facetten mehr dieser Art. Weiterhin kann jeder Facette, welche ihren Demand nicht innerhalb zufriedenstellen kann, einem der drei Fälle aus Lemma 3.1 zugeordnet werden. Um nur eine minimale Anzahl an Kanten entfernen zu müssen, gilt nach Lemma 3.9, dass möglichst viele benachbarte Facetten mit offenem Demand paarweise zu verbinden sind. Hierfür betrachten wir einen Teil des Dual Graph $G^* = (V^*, E^*)$ von G. Dabei besteht V^* nur aus den Knoten, die eine Facette von G repräsentieren, welche über einen nicht zufriedengestellten Demand verfügt. Dementsprechend existieren auch nur Kanten $\{u^*, w^*\} \in E^*$, die benachbarte Facetten aus G mit offenem Demand verbinden. Der Dual Graph ist dabei nicht unbedingt zusammenhängend. Gesucht ist ein Teilgraph $H = (V^*, S^* \subseteq E^*)$, sodass jeder Knoten $v^* \in V^*$ maximal Teil einer Kante $e \in E^*$ ist. Dies entspricht genau einem maximalen Matching auf dem Dual Graphen.

Lemma 3.10. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden. Sei weiterhin $G^* = (V^*, E^*)$ der Dual Graph zu G, der wie oben beschrieben nur die Facetten von G repräsentiert, deren Demand nicht zufriedengestellt ist. Sei $H = (V^*, S^* \subseteq E^*)$ ein Teilgraph von G^*, der durch ein maximales Matching auf G^* entsteht.*
Eine maximale Anzahl an paarweisen Verbindungen von benachbarten, nicht zufriedengestellten Facetten in G, also das Entfernen einer Menge von Kanten $S \subseteq E$, entspricht genau einem maximalen Matching H auf G^*. Dabei gilt zudem $|S| = |S^*|$.

Beweis. Nach Voraussetzung gilt, dass G^* die Facetten von G repräsentiert, die ihren Demand nicht innerhalb zufriedenstellen können. Zudem stellen die Kanten von G^* Kanten aus G dar, die zu zwei dieser Facetten zugehörig sind. Weiterhin bedeutet paarweises Verbinden von zwei Facetten, dass diese durch Löschen einer Kante verbunden werden.
Seien nun f_1, f_2 benachbarte Facetten in G mit nicht zufriedengestellten Demands. Dann gilt, dass eine Kante $e \in E$ existiert, die sowohl f_1 als auch f_2 zugehörig ist. Durch die Konstruktion von G^* existiert eine Kante $e^* \in E^*$, welche die Kante e kreuzt. Dies kann auf alle Facetten mit nicht zufriedengestellten Demands von G angewendet werden. Verbindet man nun paarweise eine maximale Anzahl von solchen Facetten, so wählt man eine Menge $S \subseteq E$ an Kanten aus, die aus G entfernt werden. Nach Konstruktion von G^* existiert auch eine Menge $S^* \subseteq E^*$. Eine Kante s^* ist genau dann Teil von S^*, wenn eine Kante $s \in S$ existiert, sodass sich s und s^* kreuzen. Daraus resultiert, dass eine maximale Anzahl an paarweisen Verbindungen von benachbarten Facetten genau einem maximalen Matching $H = (V^*, S^*)$ auf dem Dual Graphen G^* entspricht.
Für die Gegenrichtung gilt, dass ein maximales Matching auf dem so konstruierten Dual Graphen jedem Knoten $v^* \in V^*$ maximal eine Kante $s^* \in S^*$ zuordnet. Dies entspricht genau dem Entfernen der Kanten $s \in S \subseteq E$ aus G, also dem paarweisen Verbinden von Facetten mit nicht zufriedengestellter Matching-Bedingung. Dabei gilt für die Kante s, dass eine Kante s^* existiert, die s kreuzt. Somit gilt auch, dass $|S| = |S^*|$, da zwischen den gekreuzten Kanten eine $1:1$ -Beziehung gilt. Aufgrund des maximalen Matchings ist auch die Anzahl an paarweisen Verbindungen von Facetten maximal. □

Das maximale Matching liefert auf dem so konstruierten Dual Graphen eine maximale Anzahl an paarweisen Verbindungen von Facetten, wodurch eine maximale Anzahl an Kanten gespart werden kann, die, wenn man jede Facette einzeln betrachtet, entfernt werden müssten.
Nachdem dann eine maximale Anzahl an Facetten paarweise verbunden sind, ist die

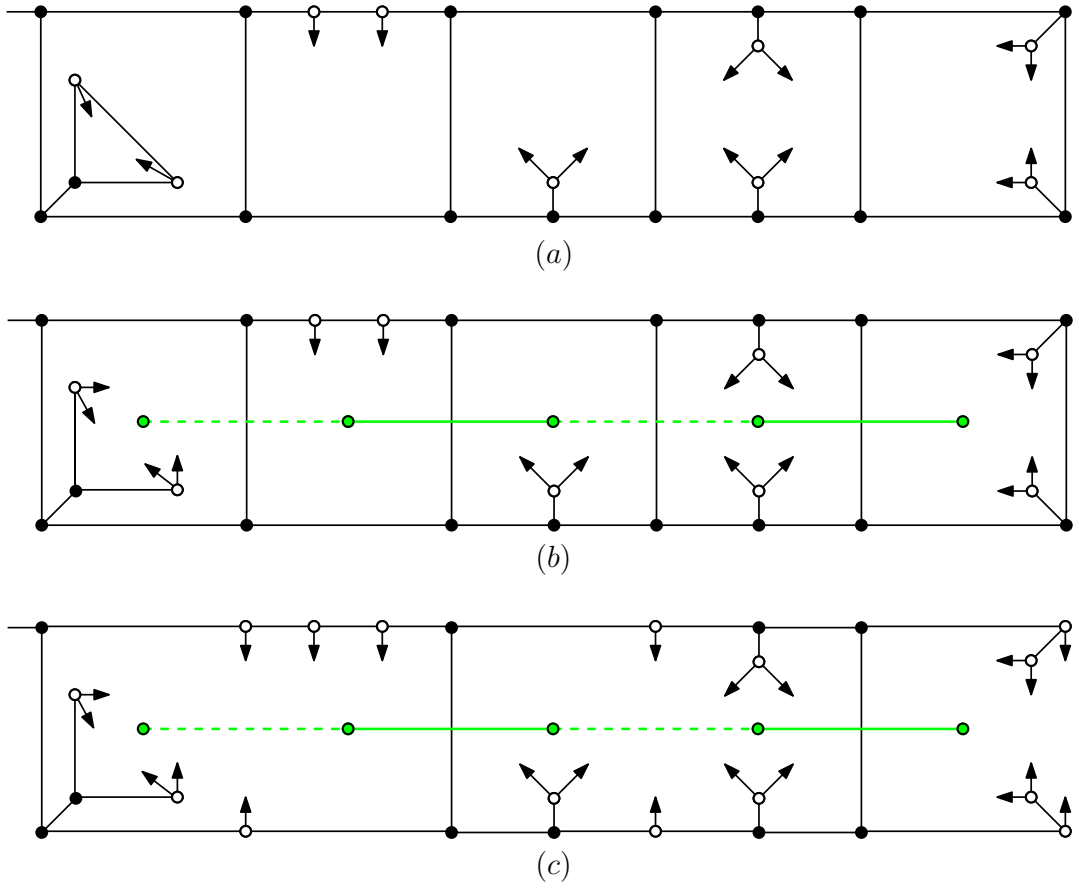


Abbildung 3.3: Graph G (schwarz) mit Leafs und Pairs. Die nicht ausgefüllten Punkte repräsentieren Knoten mit nicht vollem Grad. Der grüne Graph stellt den Dual Graphen dar. **(a)** zeigt den Graphen G in seiner ursprünglichen Form. **(b)** In der linken Facette ist die Kante beim Pair nach der Regel im Sonderfall. Die gestrichelten Kanten sind das maximale Matching. **(c)** Hier sind die zu löschenden Kanten entfernt.

Matching-Bedingung in diesen nach Lemma 3.9 zufriedengestellt. Für die übrigen Facetten, die noch nicht nicht zufriedengestellt sind, gilt nun, dass diese keine benachbarte Facette mehr besitzen, deren Matching-Bedingung ebenfalls nicht erfüllt ist. Diese Facetten können nach Lemma 3.8 durch das Löschen einer passenden Kante zufriedengestellt werden.

Fassen wir die Erkenntnisse nun zu einem Verfahren zusammen. Betrachten wir einen Graphen für den ein festes Assignment gilt. Zudem dürfen die durch das Entfernen einer Kante entstehenden Valenzen nicht reassigned werden. Weiterhin sollen in dem Graphen Knoten mit einem Grad kleiner als drei nur in Leafs und Pairs vorkommen. Dann können wir folgendes Verfahren anwenden, um eine minimale Anzahl an Kanten zu löschen, damit alle Demands im Graphen zufriedengestellt sind. So ist also die Matching-Bedingung für den Graphen erfüllt.
Zuerst werden alle Facetten durchlaufen. Falls eine Facette wie in Lemma 3.7 existiert, so entferne die Kante beim Pair. Erstelle im Anschluss den Dual Graphen, eingeschränkt auf die Facetten, die einen nicht zufriedengestellten Demand aufweisen. Um eine maximale Anzahl an Facetten paarweise zu verbinden, berechne nach Lemma 3.10 ein maximales Matching auf dem Dual Graph. Teil (b) der Abbildung 3.3 stellt diese Schritte in einem Beispielgraphen dar. Die Kanten des Graphen, die sich mit den Kanten des maximalen Matchings kreuzen, sind ebenfalls aus dem Graphen zu entfernen. Nach Lemma 3.9 sind

diese gelöschten Kanten jeweils ausreichend, um beide Facetten zufriedenzustellen. Für die restlichen Facetten gilt, wie oben beschrieben, dass diese keine unzufriedenen benachbarten Facetten mehr besitzen. Somit kann für diese eine passende Kante nach Korollar 3.6 entfernt werden, welche nach Lemma 3.8 auch existiert. Den Umbau zeigt die Abbildung 3.3 in Teil (c) für den Beispielgraphen. Hier ist eine minimale Anzahl an Kanten entfernt, sodass die Matching-Bedingung für alle verbleibenden Facetten erfüllt ist. Dieses Verfahren erstellt also aus einem Graphen $G = (V, E)$ eine neuen Graphen $G' = (V, E' \subseteq E)$.

Lemma 3.11. *Sei $G = (V, E)$ ein Leaf/Pair-Graph mit festem Assignment und Valenzen dürfen nicht neu zugewiesen werden.*
Für G lässt sich eine minimale Menge an zu entfernende Kanten in Polynomialzeit im Bezug auf die Eingabegröße berechnen, sodass für G dann die Matching-Bedingung erfüllt ist.

Beweis. Betrachten wir das beschriebene Verfahren. Falls der Graph Facetten, wie in Lemma 3.7 beschrieben, enthält, so existiert nach diesem Lemma eine Lösung, sodass die Matching-Bedingung erfüllt ist, wenn die jeweilige Kante beim Pair entfernt wird. Alle Facetten bezüglich dieser speziellen Form zu überprüfen funktioniert in linearer Laufzeit. Somit sind im folgenden keine Facetten mit diesem Spezialfall mehr enthalten. Das dieses Entfernen der Kanten die Anzahl an zu entfernenden Kanten nur in einem minimal nötigen Maß erhöht, geht ebenfalls aus dem Lemma 3.7 hervor.
Im nächsten Schritt existieren k Facetten im Graphen G, deren Demand unzufrieden ist. Nach Lemma 3.8 existiert in diesen Facetten eine passende Kante, wodurch maximal noch k Kanten entfernt werden müssen. Durch die Konstruktion des Dual Graphen, was auch in linearer Zeit funktioniert und dem Maximalen Matching, was nach dem Algorithmus von Micali und Vazirani [MV80, Gab83] in Polynomialzeit läuft, können Kanten gespart werden. Das Lemma 3.10 bestimmt ein maximales Matching, sodass eine maximale Anzahl an Facetten paarweise Verbunden werden. Es werden s Kanten aus G entfernt, wodurch man $2 \cdot s$ Facetten zufriedengestellt. Jetzt müssen nur $k - s$ Kanten entfernt werden.
Für die verbleibenden $k - 2 \cdot s$ Facetten gilt, dass diese nicht mehr mit benachbarten unzufriedenen Facetten zu einem Vorteil verbunden werden können. Aus diese übrigen Facetten wird eine nach Korollar 3.6 passende Kante entfernt, welche nach Lemma 3.8 immer existiert. Dies läuft in linearer Zeit, wobei es nicht möglich ist weitere Kanten einzusparen.
Somit entsteht in Polynomialzeit ein Graph in dem die Matching-Bedingung erfüllt ist. □

Dieser Algorithmus funktioniert nun also für Leaf/Pair-Graphen mit festem Assignment und ohne dass Valenzen neu zugeordnet werden.

4. Joker-Graphen

Auch im nachfolgenden Kapitel werden planare Graphen betrachtet, die nur Knoten mit einem Grad ≤ 3 enthalten. Weiterhin existiert ein festes Assignment und eine feste Einbettung. Darüber hinaus sollen alle Knoten, welche Grad < 3 haben, als Joker vorkommen, also als einzelne Grad-2 Knoten. Zusätzlich soll gelten, dass die Anzahl an Jokern im gesamten Graphen gerade ist, da nur so eine Gesamtlösung gefunden werden kann. Außerdem dürfen Valenzen wieder nicht neu zugewiesen werden.
In diesem Fall ist die Parity-Bedingung die interessante Bedingung, da es aufgrund der Joker vorkommen kann, dass einer Facette eine ungerade Anzahl an Valenzen zugewiesen ist. Die Matching-Bedingung ist hier zu vernachlässigen, da diese wegen der Einschränkungen bereits erfüllt ist, sobald die Parity-Bedingung erfüllt ist. Es können nämlich dann die Joker paarweise verbunden werden. Das bedeutet, dass durch das Erfüllen der Parity-Bedingung auch die Matching-Bedingung erfüllt ist.

Sind einer Facette f im Graphen eine gerade Anzahl an Jokern zugewiesen, so ist die Parity-Bedingung für die Facette f erfüllt. Falls einer Facette aber eine ungerade Anzahl an Jokern zugewiesen ist, so kann die Parity-Bedingung in der Facette nicht erfüllt werden, da eine ungerade Anzahl an Valenzen nicht allein durch Einfügen von Kanten zu lösen ist. Dies bedeutet, dass eine globale Interaktion zwischen Facetten nötig ist, um die Parity-Bedingung für alle Facetten zu lösen.
Es existieren also in dem Graphen potentiell Facetten mit und ohne erfüllter Parity-Bedingung. Erstellt man nun zum gegeben Graph $G = (V, E)$ den entsprechenden Dual-Graph $G^* = (V^*, E^*)$ so kann man in diesem eine Menge $T \subseteq V^*$ an Terminalen und eine Menge $N = V^* \setminus T$ an Nicht-Terminalen definieren. Ein Terminal stellt dabei eine Facette f von G dar, in welcher die Parity-Bedingung nicht erfüllt ist. Analog stellt ein Nicht-Terminal eine Facette dar, in welcher die Parity-Bedingung erfüllt ist. So kann das ursprüngliche Problem auf den Dual-Graphen G^* übertragen werden.
Um Facetten f_T mit nicht erfüllter Parity-Bedingung zufrieden zu stellen, muss aus dem Rand der Facette mindestens eine Kante entfernt werden. Dies entspricht der Auswahl einer Kante $e^* \in E^*$ aus dem Dual-Graphen, die an dem Terminal anliegt, welches f_T repräsentiert. Dies liefert, dass jedes Terminal mindestens in einer ausgewählten Kante vorkommen muss.
Das Ziel ist eine minimale Auswahl an Kanten $S \subseteq E^*$ zu finden, sodass jedes Terminal in mindestens einer Kante dieser Menge vorkommt. Weiterhin soll für jeden zusammenhängenden Teilgraphen in H gelten, dass dieser jeweils eine gerade Anzahl an Terminalen

enthält. Dadurch wird sichergestellt, dass im ursprünglichen Problem jeweils eine gerade Anzahl an Jokern verbunden werden, wodurch die Parity-Bedingung erfüllt ist.
Nun stellt sich die Frage, welche Eigenschaft für eine Kante $s \in S$ gelten muss. Betrachtet man den Überdeckungsgraphen $H = (V^*, S)$ so gilt für jede Kante $s \in S$, dass an beiden Seiten der Kante s über H eine ungerade Anzahl an Terminalen angebunden ist.

Lemma 4.1. *Sei $T \subseteq V^*$ die Menge an Terminalen. Sei $H = (V^*, S \subseteq E^*)$ eine minimale Überdeckung für G^*, sodass jedes Terminal teil einer Kante $s \in S$ ist und in jedem unabhängigen Teilgraphen von H eine gerade Anzahl an Terminalen vorliegt.*
Dann gilt, dass an jeder Seite einer Kante $s \in S$ eine ungerade Anzahl an Terminalen anliegt.

Beweis. Nimmt man an, dass an beiden Seiten einer Kante $s \in S$ eine gerade Anzahl an Terminalen anliegen, dann gilt, dass in beiden Teilgraphen links und rechts von der Kante eine gerade Anzahl an Terminalen existiert. Dies widerspricht aber der ursprünglichen Bedingung, dass die Überdeckung H minimal ist, da für eine Überdeckung $H' = (V^*, S \setminus \{s\})$ die Bedingungen immer noch erfüllt sind, da weiterhin alle Terminale eingebunden sind und auch in jedem Teilgraphen eine gerade Anzahl an Terminalen vorliegt.
Geht man von der Annahme aus, dass an einer Seite von $s \in S$ eine gerade Anzahl an Terminalen anliegt und an der anderen eine ungerade Anzahl, dann gilt für den Teilgraphen, der s enthält, dass dieser insgesamt eine ungerade Anzahl an Terminalen enthält. Dies widerspricht aber den Voraussetzungen.
Insgesamt folgt daraus für eine Kante in H, dass an beiden Seiten eine ungerade Anzahl an Terminalen anliegen muss. □

Dadurch folgen auch einige Eigenschaften für die Terminale und die Nicht-Terminale bezüglich H.

Korollar 4.2. *Sei $T \subseteq V^*$ die Menge an Terminalen. Sei $H = (V^*, S \subseteq E^*)$ eine minimale Überdeckung für G^*, sodass jedes Terminal teil einer Kante $s \in S$ ist und in jedem unabhängigen Teilgraphen von H eine gerade Anzahl an Terminalen vorliegt.*
Dann gilt für jedes Terminal, dass an diesem eine ungerade Anzahl an Kanten bezüglich der Überdeckung H anliegt.

Beweis. Betrachten wir ein Terminal $t \in T$. Angenommen an t liegen in der Überdeckung H eine gerade Anzahl an Kanten an. Nach Lemma 4.1 gilt für jede dieser Kanten, dass an der Seite, an welcher nicht t anliegt, eine ungerade Anzahl an Terminalen anliegen muss. Somit sind in dem jeweiligen zusammenhängenden Teilgraphen, der t beinhaltet, insgesamt eine ungerade Anzahl an Terminalen. Dies ergibt sich aus der Tatsache, dass t in diesem Teilgraphen liegt und gerade oft (Anzahl Kanten an t) eine ungerade Anzahl an Terminalen pro Kante eingebunden sind. Deshalb ist die Anzahl der Terminale dann in dem Teilgraphen ungerade. Das widerspricht aber der Annahme, das in jedem Teilgraphen eine gerade Anzahl an Terminalen enthalten ist. □

Ähnliches gilt für die Nicht-Terminale in dem Graphen G^* bezüglich der Überdeckung H. Dabei liegen aber an den Nicht-Terminalen eine gerade Anzahl an Kanten an.

Korollar 4.3. *Sei $T \subseteq V^*$ die Menge an Terminalen und $NT = V^* \setminus T$ die Menge an Nicht-Terminalen. Sei $H = (V^*, S \subseteq E^*)$ eine minimale Überdeckung für G^*, sodass jedes Terminal teil einer Kante $s \in S$ ist und in jedem unabhängigen Teilgraphen von H eine gerade Anzahl an Terminalen vorliegt.*
Dann gilt für jedes Nicht-Terminal, dass an diesem eine gerade Anzahl an Kanten bezüglich der Überdeckung H hängt.

Beweis. Betrachten wir ein Nicht-Terminal $n \in NT$. Angenommen an n liegen in der Überdeckung H eine ungerade Anzahl an Kanten an. Nach Lemma 4.1 gilt für jede der Kanten, dass an der Seite, an welcher nicht n liegt, eine ungerade Anzahl an Terminalen anliegt. Somit existiert in dem Teilgraphen, in welchem n liegt, insgesamt eine ungerade Anzahl an Terminalen, da an n ungerade oft (Anzahl Kanten an n) eine ungerade Anzahl an Terminalen anliegt. Das widerspricht aber der Annahme, dass in jedem Teilgraphen eine gerade Anzahl an Terminalen enthalten ist. □

Betrachten wir nun noch die andere Richtung, dass man über eine Teilgraphen in der Überdeckung behaupten kann, dass dieser eine gerade Anzahl an Terminalen enthält, wenn in der Überdeckung an jedem Terminal eine ungerade Anzahl und an jedem Nicht-Terminal eine gerade Anzahl an Kanten anliegt.

Korollar 4.4. *Sei $G = (V, E)$ ein Graph. Sei $T \subseteq V$ die Menge an Terminalen und $NT = V \setminus T$ die Menge an Nicht-Terminalen. Sei $H = (V, S)$ eine minimale Überdeckung von G, sodass an jedem Terminal eine ungerade Anzahl an Kanten und an jedem Nicht-Terminal eine gerade Anzahl an Kanten in der Überdeckung anliegt.*
Dann gilt, dass in jedem Teilgraphen der Überdeckung eine gerade Anzahl an Terminalen vorliegt.

Beweis. Angenommen in der Überdeckung $H = (V, S)$ existiert ein Teilgraph, der eine ungerade Anzahl an Terminalen enthält. Da an jedem der Terminale ungerade viele Kanten anliegen sollen, liegt dann an allen Terminalen insgesamt eine ungerade Anzahl an Kanten an. Weiterhin sollen an allen Nicht-Terminalen eine gerade Anzahl an Kanten anliegen. Damit hat man insgesamt eine gerade und eine ungerade Anzahl, also insgesamt eine ungerade Anzahl. Bei dieser Zählweise ist jede Kante doppelt gezählt worden. Da aber eine ungerade Anzahl nicht durch 2 teilbar ist, ist die Annahme, dass ein Teilgraph eine ungerade Anzahl an Terminalen enthält, falsch. □

Somit ist definiert, wie viele Kanten mindestens an den Terminalen und Nicht-Terminalen bezüglich einer minimalen Überdeckung anliegen. Das Ziel ist nun mittels eines Maximum-Weighted-B-Matchings eine solche minimale Überdeckung zu finden. Dafür ist der Dual-Graph $G^* = (V^*, E^*)$ noch passend zu erweitern.
Abbildung 4.1 zeigt dabei die Umbaumaßnahmen beispielhaft für eine Graphen. Zuerst weist man jedem Knoten $v \in V^*$ zu, wie viele Kanten im späteren Matching an diesem anliegen sollen. Hierfür definiert man eine Grad-Funktion b. Jeder Knoten v erhält dabei $deg(v)$ beziehungsweise $deg(v) - 1$. Dies ist davon abhängig, ob v eine Terminal oder ein Nicht-Terminal ist. Wie in den Korollaren 4.2 und 4.3 beschrieben, benötigt ein Terminal eine ungerade Anzahl an Kanten in der Überdeckung und ein Nicht-Terminal eine gerade Anzahl. Dann gilt beispielsweise für einen Knoten $x \in T$ mit Grad 6, dass $b(x) = deg(x) - 1 = 6 - 1 = 5$ ist.
Zusätzlich wird der Graph um eine Gewichtsfunktion c erweitert. Diese Funktion weist dabei jeder Kante $e \in E^*$ ein Gewicht von -1 zu. Weiter werden „Dreiecke“ an die Knoten von G^* angefügt. Diese Dreiecke sind in der Abbildung 4.1 beispielhaft in blau zu sehen. Ein solches Dreieck besteht dabei aus zwei neuen Knoten, denen die Funktion b als Wert 1 zuweist und einem Knoten aus G^*. Zusätzlich kommen drei Kanten hinzu, denen c jeweils ein Gewicht von 0 zuweist. Dabei erhält jeder Knoten $v \in V$ $\lfloor \frac{b(v)}{2} \rfloor$ Dreiecke.
Dadurch ist ein erweiterter Graph gebaut, dessen Nutzen in nachfolgendem Lemma gezeigt wird.

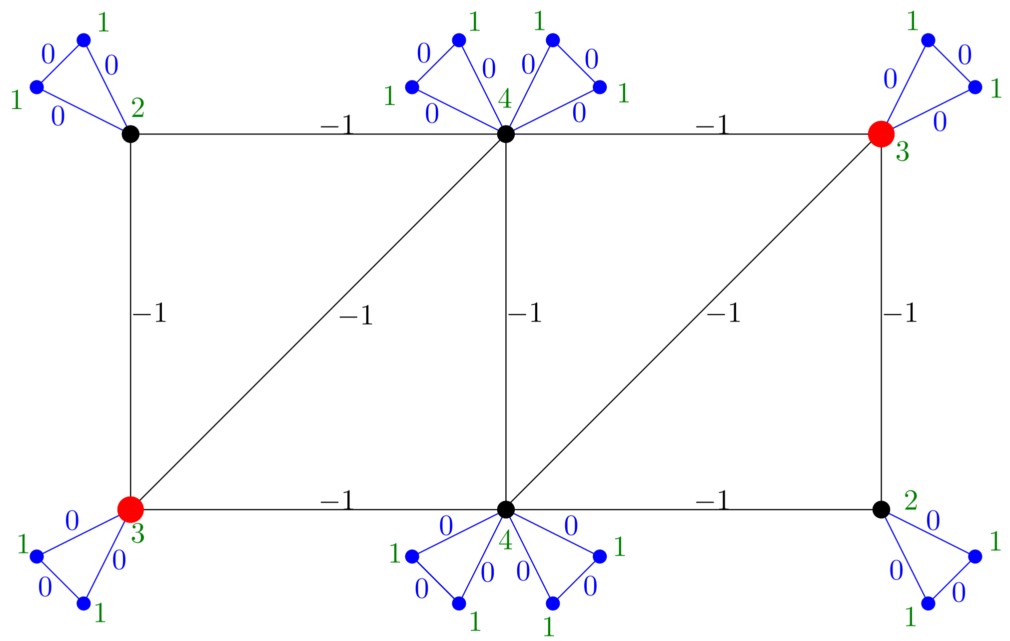


Abbildung 4.1: Umbau des Graphen G (schwarz) mit Terminalen (rot) und Nicht-Terminalen (schwarz). Hinzufügen von Kantengewichten für G von -1 und bei jedem Knoten die passende Anzahl an Kanten in der Überdeckung (grün). Hinzu kommen noch die Dreiecke (blau) mit Kantengewichten von 0.

Lemma 4.5. *Sei $T \subseteq V^*$ ein Menge an Terminalen. Sei $G' = (V', E')$ der nach obigen Regeln umgebaute Graph von $G^* = (V^*, E^*)$. Sei $H' = (V', S')$ ein Maximum-Weighted-B-Matching auf G'.*
Dann gilt, dass $H = (V*, S = (S' \wedge E^*))$ eine minimale Überdeckung von G^* ist, sodass jedes Terminal teil einer Kante $s \in S$ ist und in jedem unabhängig Teilgraph von H eine gerade Anzahl an Terminalen vorliegt.

Beweis. Aufgrund der Konstruktion der „Dreiecke“ wird entweder die Kante in das Matching genommen, die die zwei neuen Knoten des Dreiecks verbindet, oder es werden die zwei Kanten genommen, die in Verbindung zum Knoten des ursprünglichen Graphen stehen. Dies liegt daran, dass die beiden neuen Knoten im Matching jeweils genau eine Kante fordern.
Somit haben die Dreiecke auch eine Auswirkung auf die Knoten des ursprünglichen Graphen. Da die Kanten im Dreieck ein Gewicht von 0 haben, werden diese bevorzugt vor den Kanten von G^*, welche ein Gewicht von -1 haben, in das maximale Matching genommen. Dadurch werden möglichst viele neue Kanten an die einzelnen Knoten $v \in V^*$ gebunden. Da aber dann immer zwei neue Kanten durch eine Dreieck an v gebunden werden, kann $b(v)$ um genau 2 reduziert werden. Daher gilt für ein Terminal, dass mindestens eine Kante aus E^* in S' übernommen wird, da nur so die geforderte ungerade Anzahl zufriedengestellt werden kann. Es werden aber auch nicht mehr als unbedingt nötig übernommen, da sonst das Gesamtgewicht nicht maximal wird. Für ein Nicht-Terminal n gilt analog, dass am besten $b(n)$ komplett durch die Dreiecke abgedeckt wird. Dies funktioniert, da $b(n)$ gerade ist.
Das Maximum-Weighted-B-Matching liefert also eine Überdeckung $H' = (V', S')$ für die nun gilt, dass an jedem Terminal eine ungerade Anzahl an Kanten hängt und an jedem Nicht-Terminal eine gerade Anzahl. Da das Matching maximales Gewicht haben soll, werden möglichst wenig Kanten aus E^* verwendet. Dadurch bildet $H = (V^*, S = (S' \wedge E^*))$ eine minimale Überdeckung für G^*, denn in H werden nur die Kanten übernommen, welche auch

in E^* liegen. Da die Anzahl bei jedem Terminal ungerade und bei jedem Nicht-Terminal gerade ist, folgt nach Korollar 4.4, dass in der Überdeckung in den Teilgraphen eine gerade Anzahl an Terminalen vorliegt. □

Durch dieses Verfahren kann nun eine passende minimale Überdeckung gefunden werden. Ein Maximal-Weighted-B-Matching kann nach dem Artikel von Gabow [Gab18] in $O(\min\{b(V), n \cdot \log n\} \cdot (m + n \cdot \log n))$ gefunden werden. Dabei ist $b(V) = \sum_{v \in V'} b(v)$ und dies liegt nach Konstruktion in $O(E^*)$. Weiterhin sind $n = |V'|$ und $m = |E'|$. Aufgrund der Konstruktion von G' gilt für n, dass dieses in $O(V^* + E^*)$ liegt und für m, dass es in $O(E^*)$ liegt. Somit kann das Mamimum-Weighted-B-Matching in Polynomialzeit berechnet werden. Der Umbau von G zu G' funktioniert ebenfalls in Polynomialzeit. Damit kann das ursprüngliche Problem auf einem Graphen G in Polynomialzeit gelöst werden.

5. NP-Vollständigkeit vom Editieren mit Neuzuweisungen

Bei den bisherigen Ergebnissen existiert immer die Einschränkung, dass Valenzen nicht neu zugewiesen werden dürfen. Lässt man allerdings Neuzuweisungen zu, können potentiell weniger Kanten zu löschen sein. Abbildung 5.1 zeigt, dass es mit Neuzuweisen ausreicht, die eine rot gestrichelte Kante zu löschen. Dürfte man hier nicht Neuzuweisen, so müssten zwei Kanten gelöscht werden. Welche Probleme das Weglassen dieser Einschränkung mit sich bringt, soll nun in diesem Kapitel dargelegt werden.
Betrachten wir einen allgemeinen Graphen, in welchem Neuzuweisungen gestattet sind und in dem der Grad der Knoten maximal 5 ist. Das Problem einen solchen Graphen zu einem 3-regulären und planaren Graphen umzubauen, ist NP-vollständig.

Es stellt sich die Frage, ob es ausreicht, K Kanten aus dem Graphen zu löschen, sodass dieser dann in eine 3-reguläre und planare Form augmentiert werden kann. Dieses Problem liegt in NP. Rate dazu K Kanten, welche entfernt werden sollen, und verifiziere dann in Polynomialzeit, ob in dem resultierenden Graphen die Parity- und die Matching-Bedingung erfüllt sind.
Um jetzt nachzuweisen, dass das Problem NP-schwer ist, reduzieren wir das Problem Monotone Planar 3-SAT auf unser Problem.

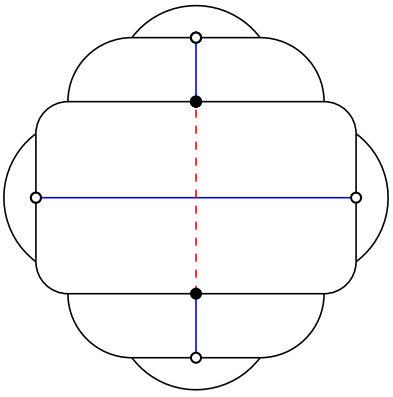


Abbildung 5.1: Das Löschen der roten gestrichelten Kante reicht aus, damit der Graph komplett durch Einfügen der blauen Kanten umgebaut werden kann.

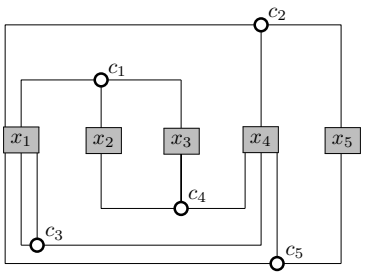


Abbildung 5.2: Aufbau eines Variable-Klausel Graphen mit 5 Variablen und 5 Klauseln für planares 3-SAT.

Die Reduktion ist inspiriert von der Reduktion von Hartmann, Rollin und Rutter [HRR14], welche zeigt, dass das Problem c-connected 3-PRA für $c = 0, 1, 2, 3$ NP-vollständig ist. Also dem Problem einen c-verbunden Graphen so zu augmentieren, dass dieser 3-regulär und planar wird.
Um zu verstehen, wie eine Instanz von Monotone Planar 3-SAT aufgebaut ist, betrachten wir zuerst Planar 3-SAT. Eine Instanz von Planar 3-SAT ist eine 3-SAT Formel φ, deren Variable-Klausel Graph planar ist. Die Abbildung 5.2 zeigt beispielhaft den Aufbau eines Variable-Klausel Graphen für planares 3-SAT. Dabei gilt, dass die Variablen in der Mitte liegen und die Klauseln oberhalb und unterhalb der Variablen angeordnet werden. Da es sich um eine spezifische Form von 3-SAT handelt, sind die Klauseln durch Konstrukte dargestellt, welche über genau drei Anschlüsse verfügen, mit welchen die Klausel-Konstrukte mit den Variablen verbunden sind.
Damit daraus nun ein Instanz von Monotone Planar 3-SAT wird, muss der Variable-Klausel Graph von φ eine weitere Bedingung erfüllen. Für die Klauseln, welche oberhalb der Variablen-Achse liegen, soll gelten, dass in diesen nur positive Literale vorkommen. Analog gilt für die Klauseln, welche unterhalb der Variablen-Achse angeordnet sind, dass in diesen nur negative Literale existieren. Für jede Klausel gilt, dass die Variablen in diesen entweder nur in negierter oder nur in nicht negierter Form vorkommen. Man kann direkt unterscheiden, ob es sich um eine positive Klausel handelt, die nur positive Literale enthält, oder um eine negative Klausel.

Jetzt konstruieren wir zu einer Instanz φ von Monotone Planar 3-SAT ein Tupel $(G_\varphi; K)$. Für dieses soll gelten, dass genau dann für φ eine erfüllende Belegung existiert, wenn G_φ mit maximal K Kantenlöschungen zu 3-regulärer, planarer Form editierbar ist. Existiert keine erfüllende Belegung für φ, so müssen in G_φ mindestens $K + 1$ Kanten gelöscht werden, sodass G_φ editierbar ist.
Das Tupel besteht aus zwei Bestandteilen. Betrachten wir zuerst den Graphen G_φ des Tupels. Der Graph soll eine Repräsentation für φ werden. Da sich φ aus Klauseln zusammensetzt, welche Variablen enthalten, soll G_φ eine Konstruktion für jede Variable, sowie für jede Klausel enthalten. Darüber hinaus ist φ eine Instanz von Monotone Planar 3-SAT. Wie oben bereits beschrieben, liegen für einen solche Instanz Einschränkungen vor. Diese Einschränkungen, dass ein planarer Variable-Klausel-Graph mit positiven Klauseln oben und negativen Klauseln unten vorliegt, werden in der nachfolgenden Konstruktion ausgenutzt. Deshalb betrachten wir die einzelnen Bestandteile zunächst für sich und führen diese in einem zweiten Schritt dann zu G_φ zusammen.

Das Konstrukt einer Variable ist in Abbildung 5.3 beispielhaft dargestellt. Eine solche Variable besteht grundsätzlich aus einer Aneinanderreihung von Grundmodulen, welche dann zu einem Kreis zusammengeschlossen werden. Ein solches Grundmodul besteht aus

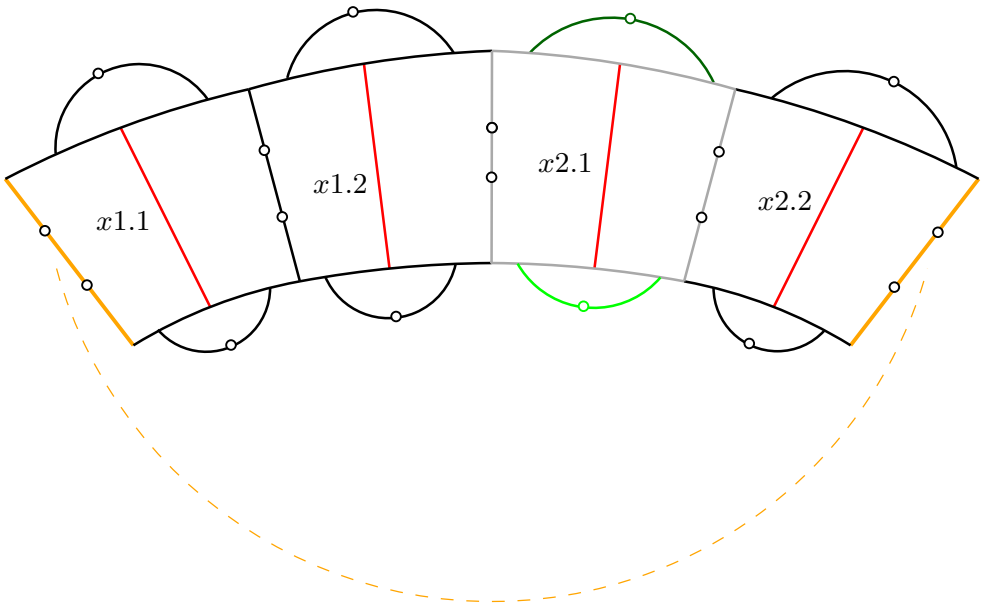


Abbildung 5.3: Aufbau einer Variable in G_φ. Hinter jeder Kreuzung von drei Kanten liegt ein Knoten, welche aufgrund der Übersichtlichkeit hier nicht explizit eingezeichnet ist. Drei Teile eines Grundmoduls: Äußere Kuppel (dunkelgrün), Innere Kuppel (grün) und Ring (grau) mit Trennkante (rot). Die Kante ganz links und die ganz rechts sind die gleiche Kante (orange).

drei Teilen, nämlich einem Ring, einer inneren Kuppel und einer äußeren Kuppel. Der Ring besteht dabei aus zwei gegenüberliegenden Begrenzungskanten, sowie zwei Verbindungskanten dieser Begrenzungskanten, auf welchen jeweils zwei freie Valenzen liegen. Zwischen den Verbindungskanten mit den Valenzen liegt jeweils eine Trennkante, welche ebenfalls an die Begrenzungskanten angeschlossen ist. Zwischen den Valenzen auf den Verbindungskanten besteht daher keine direkte Verbindung im Inneren des Rings. Die innere beziehungsweise äußere Kuppel liegen jeweils an den Begrenzungskanten an, sodass diese eine Kuppel über den jeweiligen Anschlusspunkt der Trennkante bilden. Des Weiteren liegt auf jeder Kuppel eine freie Valenz.
Die Aneinanderreihung der Grundmodule kommt dadurch zu Stande, dass sich zwei Grundmodule jeweils eine Verbindungskante mit den darauf liegenden Valenzen teilen. Wobei das erste Grundmodul sich die Kante mit dem letzten teilt und so den Kreisschluss herbeiführt. Dadurch liegen die inneren Kuppeln auch im Inneren der Konstruktion und die äußeren Kuppeln außen und sind somit klar zugewiesen.
Den Grundmodulen weisen wir einen Bezeichner zu, der sich aus dem Variablennamen, also beispielsweise x, und einer fortlaufenden Nummerierung zusammensetzt. Das erste Grundmodul wird mit $x1.1$ und das Zweite mit $x1.2$ bezeichnet, das Dritte mit $x2.1$ und das Vierte mit $x2.2$. Diese Nummerierung wird dann für die weiteren Grundmodule so fortgesetzt. Sie resultiert daraus, dass immer zwei Grundmodule zusammen einen Anschlusspunkt für die Klauseln bilden.
Auf den Verbindungskanten existieren immer genau zwei Valenzen, die aufgrund des Kreisschlusses jeweils zu zwei Ringen gehören. Diese beiden Valenzen erfordern das Löschen einer Kante, sodass deren Demand erfüllt werden kann. Betrachtet man nun einen einzelnen Ring, so reicht es aus, die Trennkante in diesem Ring zu löschen, sodass die Valenzen auf den beiden Verbindungskanten zufriedengestellt werden können. Die Valenzen, welche beim Löschen der Trennkante entstehen, werden aber nicht benötigt um die Valenzen auf den Verbindungskanten zufriedenstellen zu können. Dies resultiert daher, dass die $2 \cdot 2 = 4$ Valenzen der Verbindungskanten gegenseitig zufriedenstellen. Die frei werdenden Valenzen

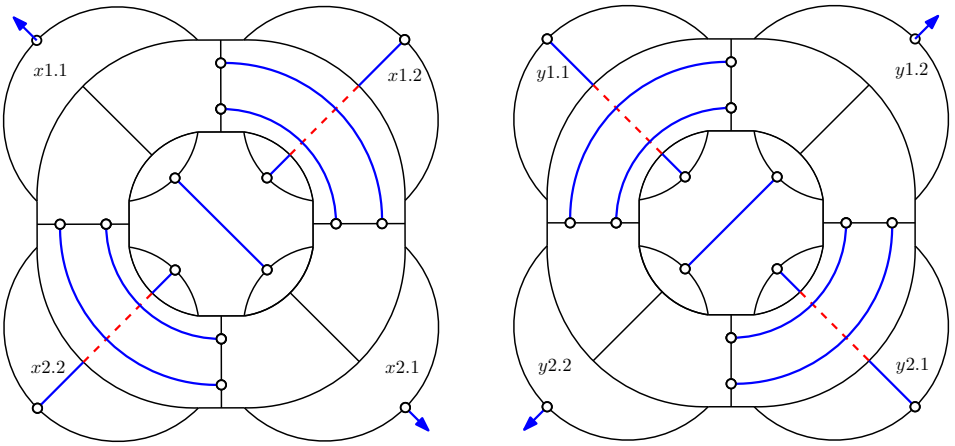


Abbildung 5.4: Editierte Variable-Konstruktion in G_φ mit gelöschten Kanten (rot-gestrichelt) und neu eingefügten Kanten (blau) (a) Variable $x = true$ (b) Variable $y = false$

können somit in die innere beziehungsweise äußere Kuppel zugewiesen werden, da diese aufgrund der Konstruktion genau über der Anschlussstelle der Trennkante liegt. Die neu zugewiesenen Valenzen in den Kuppeln stellen dann den Demand der einen Valenz der Kuppelkonstruktion zufrieden. Dies beweist, dass es ausreicht, genau die Trennkante zu löschen, sodass ein Grundmodul dann komplett augmentierbar ist.
Da aber in der gesamten Variable-Konstruktion eine Verbindungskante immer zu zwei Ringen gehört, folgt, dass die Valenzen in nur einem der Ringe zufriedengestellt werden müssen. Zum Beispiel reicht es für die Verbindungskanten aus, welche sich $x1.2$ und $x2.1$ beziehungsweise $x2.1$ und $x2.2$ teilen, die Trennkante in $x2.1$ zu löschen. Dadurch reicht es aus, abwechselnd in einem Ring die Trennkante zu löschen und im nächsten diese nicht zu löschen. Damit dieses abwechselnde Löschen ausreicht, fordern wir, dass die Anzahl der Grundmodule gerade ist. Somit reicht es aus, entweder die ungeraden oder die geraden Trennkanten zu löschen. Also entweder in den Grundmodulen, welche mit .1 enden, oder in denen, welche auf .2 enden.
Damit sind alle Valenzen der Ringe zufriedengestellt. Zudem sind in den Grundmodulen, in denen die Trennkanten entfernt wurden auch die Valenzen in den anliegenden inneren beziehungsweise äußeren Kuppeln zufriedengestellt. Es müssen also noch die jeweils anderen inneren und äußeren Kuppeln zufriedengestellt werden. Für die äußeren Kuppeln gilt, dass die dort verbleibenden Valenzen mit in die Klausel-Konstruktion eingebaut werden, welche wir im Anschluss genauer betrachten.
Aufgrund des Kreiskonstuktion der Variable wird im inneren der Konstruktion eine Facette eingeschlossen, in welche die inneren Kuppeln hineinragen. Da noch jede zweite innere Kuppel, also entweder die geraden oder ungeraden eine nicht zufriedengestellte Valenz besitzt, fordern wir für die Variablenkonstruktion, dass die Anzahl nicht nur ein Vielfaches von 2 sondern ein Vielfaches von 4 ist. Somit können immer zwei der nicht zufriedenen inneren Kuppeln über die innen eingeschlossene Facette verbunden werden.
Abbildung 5.4 zeigt beispielhaft eine editierte Variable-Konstruktion in der kleinsten Ausführung mit 4 Grundmodulen. Dabei sind in (a) die geraden Trennkanten entfernt und in (b) die ungeraden Trennkanten. Das Entfernen der geraden Kanten entspricht dem Setzen einer Variable in φ auf $true$. Analog entspricht das Löschen der ungeraden Kanten dem Setzen der Variable in φ auf $false$.

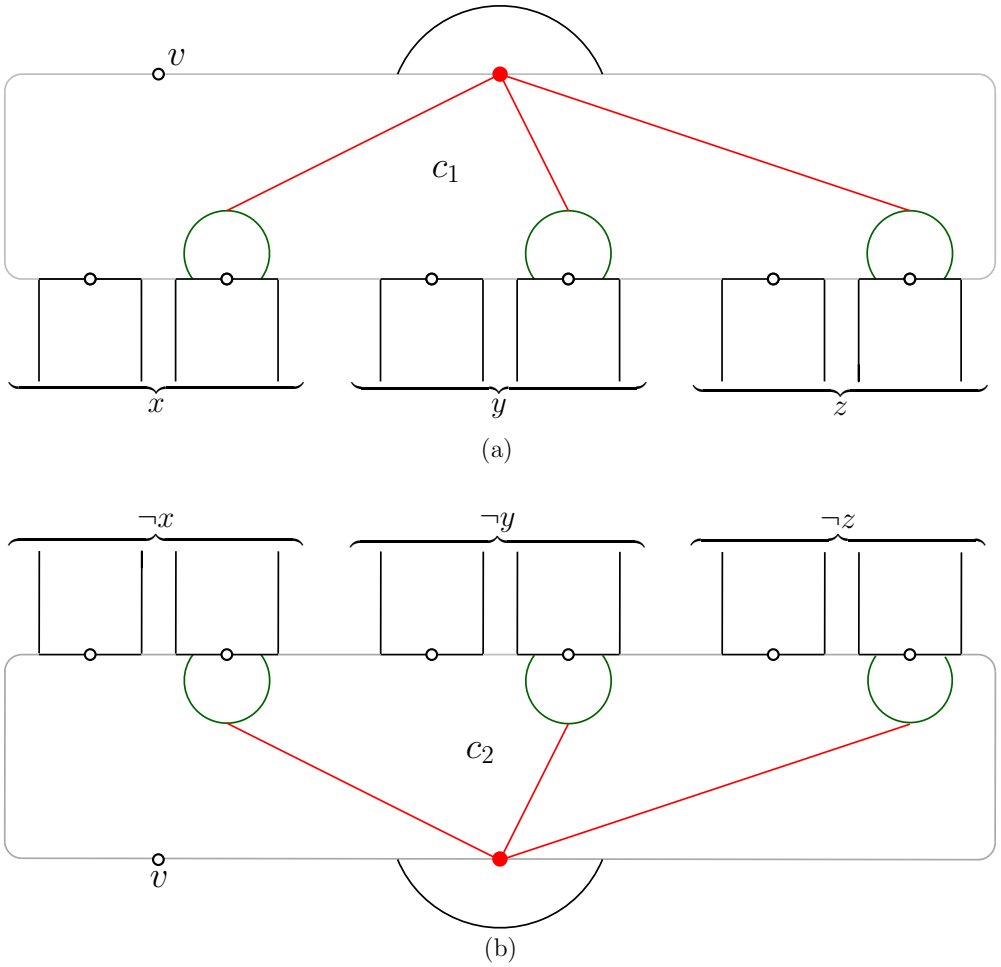


Abbildung 5.5: Aufbau der Klausel-Konstruktion mit Kuppeln (grün), Trennkanten (rot) und Rahmen (grau). (a) positive Klausel: $c_1 = x \vee y \vee z$
(b) negative Klausel: $c_2 = \neg x \vee \neg y \vee \neg z$

Wie bereits oben angerissen, bilden die äußeren Kuppel von zwei benachbarten Grundmodulen eine Anschlussstelle für die Klausel-Konstruktion. Also ist die Anschlussstelle xi durch die Grundmodule $xi.1$ und $xi.2$ definiert. Braucht man für die Variable eine weitere Anschlussstelle, so muss die Variable um zwei Anschlussstellen erweitert werden, damit die Anzahl der Grundmodule weiterhin durch 4 teilbar ist. Insgesamt gilt für das Editieren, dass in der Variable-Konstruktion für jede Anschlussstelle eine Kante gelöscht werden muss. Dadurch folgt folgende Eigenschaft für die Variable-Konstruktion.

Eigenschaft 1. *Um eine Variable-Konstruktion mit einer durch 4 teilbaren Anzahl an Grundmodulen zu editieren, sind mindestens der Anzahl an Anschlussstellen entsprechend viele Kanten zu löschen, sodass die Konstruktion augmentierbar wird. Dazu sind entweder die geraden oder die ungeraden Trennkanten zu löschen.*

Da wir eine Reduktion mit Monotone Planar 3-SAT anstreben, können wir zwischen positiven und negativen Klauseln unterscheiden. Wir konstruieren zuerst die Repräsentation eine positiven Klausel von φ in G_φ. Diese Konstruktion ist beispielhaft in Abbildung 5.5 (a) dargestellt. Die Klausel-Konstruktion ist von einem Rahmen eingefasst, auf welchen im oberen Teil ein Grad-5 Knoten enthalten ist, welcher unter einer Kuppel liegt. Links neben dem Grad-5 Knoten und dessen Kuppel liegt eine einzelne freie Valenz v. Im unteren Teil liegen die Anschlusspunkte, welche in den Rahmen eingebunden sind. Über dem

jeweiligen rechte Teil des Anschlusspunktes liegt wiederum eine Kuppel, welche die Valenz des Anschlusspunktes gegenüber dem Inneren des Rahmens abschirmt. Diese Kuppeln sind über Trennkanten mit dem Grad-5 Knoten verbunden. Somit ist die Valenz im linken Teil des Anschlusspunktes jeweils vom Inneren des Rahmens frei zugänglich und die Valenz im rechte Teil liegt unter einer Kuppel.
Aufgrund der Variable-Konstruktion und der Eigenschaft 1 gilt für einen Anschlusspunkt, dass entweder bereits die linke Valenz oder die rechte Valenz durch die Variable-Konstruktion zufriedengestellt ist. Deshalb bleibt auf jeden Fall pro Anschlussstelle eine Valenz übrig, welche in der Klausel-Konstruktion zufriedengestellt werden muss. Falls die linke Valenz übrig bleibt, so ist die zugrunde liegende Variable mit $true$ belegt und falls die rechte Valenz übrig bleibt, so ist die Variable mit $false$ belegt.
Im Folgenden werden wir die Eigenschaft des Grad-5 Knotens ausnutzen, dass an diesem zwei Kanten gelöscht werden müssen.
Betrachten wir nun die vier verschiedenen Fälle, welche bezüglich der Anzahl an Valenzen, die im linken Teil der Anschlusspunkte übrig bleiben, auftreten können.
Im ersten Fall liegen an allen drei Anschlusspunkten die nicht zufriedengestellten Valenzen auf der linken Seite. In diesem Fall kann die linke freie Valenz direkt mit v in Verbindung gebracht werden. Für die anderen beiden Valenzen gilt, dass die Trennkanten gelöscht werden, die zu der mittleren und der rechten Anschlussstelle gehören. Durch das Löschen der Trennkanten entsteht auf der jeweiligen Kuppel eine Valenz, welche direkt die freie Valenz der jeweiligen Anschlussstelle zufriedenstellt. Berücksichtigt man die Forderung des Grad-5 Knotens, dass zwei Kanten gelöscht werden müssen, so ist dies auch eine beste Lösung.
Liegen nur zwei der drei Valenzen frei zugänglich, so gilt, dass diejenigen Trennkante der Anschlussstelle gelöscht wird, bei der die freie Valenz unter der Kuppel liegt. Welche weitere Trennkanten gelöscht wird, ist dann egal. Die Valenzen in der Klausel-Konstruktion können in jedem Fall zufriedengestellt werden.
Auch wenn nur noch eine der von den Anschlussstellen unzufriedenen Valenzen nicht unter einer Kuppel liegt, reichen zwei zu löschende Kanten. In diesem Fall müssen die Trennkanten gelöscht werden, die zu den Anschlussstellen gehören, bei denen die Valenz unter der Kuppel liegt. Dadurch werden die Valenzen unterhalb der Kuppel mit der auf der Kuppel entstehenden Valenz zufriedengestellt. Weiterhin kann dann die bereits frei zugängliche Valenz im Inneren des Rahmens mit v in Verbindung gebracht werden, da die Trennkanten, welche potentiell eine Verbindung von v mit der frei zugänglichen Valenz verhindert hätten, gelöscht werden.
In den ersten drei Fällen reicht es aus, die durch den Grad-5 Knoten geforderten zwei Kanten zu löschen, sodass die Klausel-Konstruktion augmentierbar wird. Betrachtet man nun aber den vierten Fall, bei dem alle nicht zufriedengestellten Valenzen der Variable-Konstruktionen unter den Kuppeln liegen, reichen zwei Kanten nicht mehr aus. Durch die zwei zu löschenden Kanten am Grad-5 Knoten könne zwei der drei Valenzen zufriedengestellt werden. Es bleiben aber noch v und die Valenz unter der dritten Kuppel übrig. Diese sind allerdings durch eine Kuppel getrennt, sodass diese nicht verbunden werden können, ohne eine weitere Kante zu löschen.
Dieses Verhalten spiegelt das Verhalten einer Klausel in φ wieder. Kann die Klausel erfüllt werden, sind also nicht alle drei Variablen $false$, so liegen nicht alle nicht zufriedengestellten Valenzen unter den Kuppeln. Deshalb reichen dann 2 Kanten die gelöscht werden müssen, um die Konstruktion zufriedenzustellen.
Analog funktioniert die Konstruktion für eine negative Klausel. Dabei ist der Aufbau der Konstruktion nur gespiegelt. So liegt die Kuppel immer über dem linken Teil des Anschlusspunktes, wodurch der Wert der Variable negiert wird. Das Argument, dass aber nur dann 2 zu löschende Kanten ausreichen, wenn die Klausel mit den Variablenbelegungen

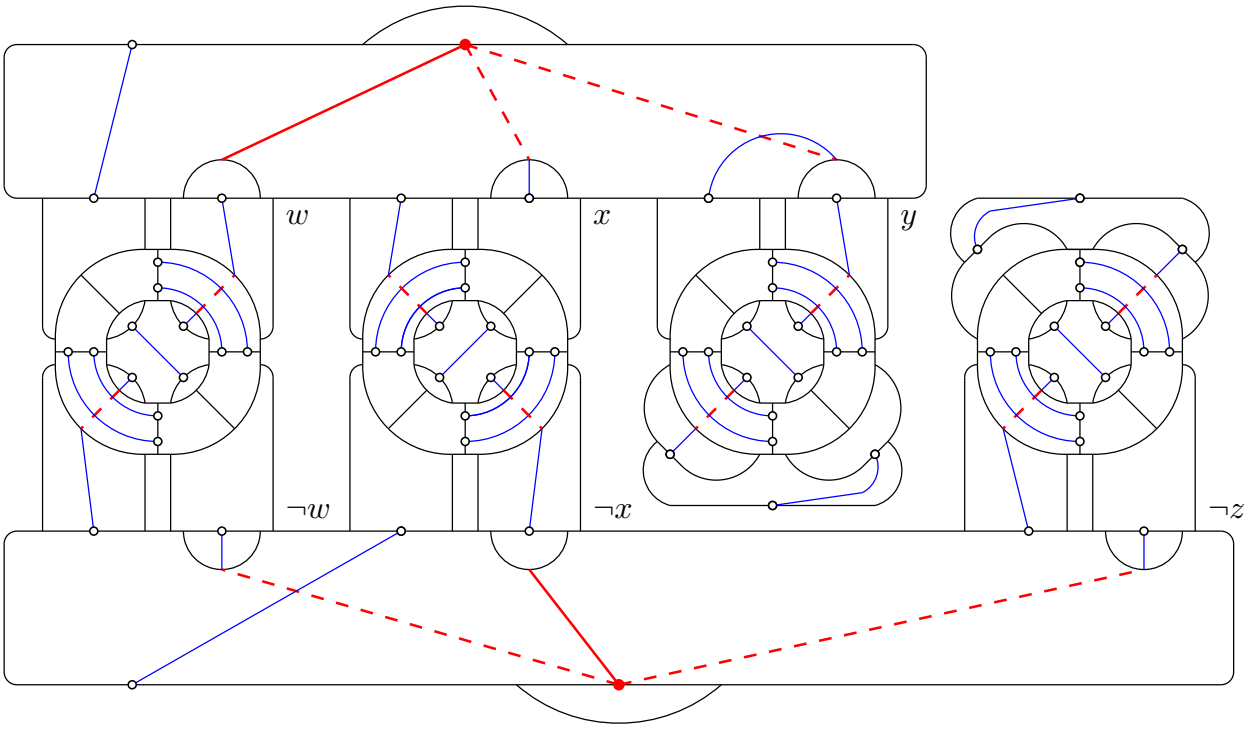


Abbildung 5.6: Editierter Graph G_φ mit $K = 4*2+2*2 = 12$ für die Monotone Planar 3-SAT-Instanz $\varphi = (w \vee x \vee y) \wedge (\neg w \vee \neg x \vee \neg z)$ mit den Variablen w, x, y, z mit $w = y = z = true$ und $x = false$.

insgesamt *true* ist, gilt weiterhin. Dies liefert die folgende Eigenschaft für die Klausel-Konstruktion.

Eigenschaft 2. *Die Klausel-Konstruktion kann genau dann mit* 2 *zu löschenden Kanten editiert werden, wenn nicht alle nicht zufriedengestellten Valenzen unter den Kuppeln liegen.*

Nachdem jetzt die zwei Hauptbestandteile von G_φ definiert sind, können wir G_φ komplett zusammenfügen. Abbildung 5.6 zeigt beispielhaft die Umsetzung von $\varphi = (w \vee x \vee y) \wedge (\neg w \vee \neg x \vee \neg z)$ als G_φ. Als Variablenbelegungen ist in dem Beispiel $w = y = z = true$ und $x = false$ gewählt.
Weiterhin zeigt die Abbildung auch, wie mit Variablen-Anschlusspunkten umgegangen wird, die nicht genutzt sind. Dies kann vorkommen, da eine Variable-Konstruktion immer um 2 Anschlussstellen erweitert werden muss. Hierfür wird über die äußeren Kuppeln der Anschlussstelle eine weitere Kuppel gespannt, welche ebenfalls eine Valenz enthält. Somit kann, egal ob die geraden oder die ungeraden Trennkanten gelöscht werden, die nicht zufriedengestellte Valenz der Variable-Konstruktion in Verbindung gebracht werden.
Bleibt noch der zweite Teil des Tupels $(G_\varphi; K)$ zu definieren. Die Zahl K ist abhängig von der Anzahl an Klauseln und der Anzahl an Anschlussstellen in G_φ. Wir definieren K nämlich als die Anzahl an Anschlussstellen plus 2-mal die Anzahl an Klausel-Konstruktionen.
Diese Definition liefert, dass für eine Instanz φ von Monotone Planar 3-SAT das Tupel $(G_\varphi; K)$ in Polynomialzeit konstruiert werden kann. Somit folgt folgendes Lemma.

Lemma 5.1. *Das Editieren eines Graphen mit maximal Grad-*5*-Knoten und Neuzuweisungen mit maximal K Löschungen, sodass dieser dann 3-regulär und planar ist, ist NP-schwer.*

Beweis. Wir zeigen, dass das Editieren in 3-reguläre planare Form mit maximal K Löschungen auf einem Graphen mit Neuzuweisungen NP-schwer ist, indem wir Monotone

Planar 3-SAT darauf reduzieren.
Nach de Berg und Khosravi ist Monotone Planar 3-SAT NP-vollständig [dBK10].
Eine Monotone Planar 3-SAT-Instanz φ kann nach obiger Konstruktion in Polynomialzeit in das Tupel $(G_\varphi; K)$ überführt werden. Existiert nun für φ eine erfüllende Belegung, so können die Variable-Konstruktionen in G_φ entsprechend editiert werden. Also falls die Variablenbelegung *true* ist, so werden die geraden Trennkanten gelöscht und falls die Variablenbelegung $false$ ist, die ungeraden. Dies entspricht für jede Variable nach Eigenschaft 1 genau der Anzahl an Anschlusspunkten dieser Variable. Da für φ eine erfüllende Variablenbelegung existiert, liegt in jeder Klausel-Konstruktion mindestens eine nicht zufriedengestellte Valenz der Anschlussstelle nicht unter einer Kuppel. Nach Eigenschaft 2 reicht es also für jede Klausel-Konstruktion aus, genau 2 Kanten zu löschen. Dies entspricht nach Definition von K genau K.
Betrachtet man nun das Tupel $(G_\varphi; K)$, so folgt mit den Eigenschaften 1 und 2, falls man entscheiden kann, ob G_φ mit K Löschungen editiert werden kann, dass in jeder Variablen-Konstruktion genau die geraden oder ungeraden Trennkanten gelöscht werden und in den Klausel-Konstruktionen genau 2 der 3 Trennkanten. Damit hat man eine erfüllenden Belegung für φ gefunden. Sollte es nicht möglich sein G_φ mit K Löschungen zu editieren, so kann die Eigenschaft 2 nicht erfüllt werden und in mindestens einer Klausel-Konstruktion müssen 3 Kanten entfernt werden. Damit ist gezeigt, dass für φ keine erfüllende Belegung existiert.
Also wenn man in Polynomialzeit entscheiden könnte, ob es für G_φ ausreicht, K Kanten zu löschen, sodass G_φ komplett in 3-reguläre planare Form gebracht werden kann, dann könnte man auch für das Problem, ob für eine Instanz φ von Monotone Planar 3-SAT eine erfüllende Belegung existiert, eine Lösung finden. □

Somit folgt insgesamt für das Problem, ob das Editieren mit maximal K Löschungen mit Neuzuweisungen eines Graphen, dessen Knoten maximal Grad 5 haben, ein NP-vollständiges Problem ist. Denn das Problem liegt in NP und ist nach Lemma 5.1 zusätzlich NP-schwer.

6. Zusammenfassung

In dieser Arbeit betrachteten wir das Editieren eines Graphen in 3-reguläre, planare Form mit den Operatoren Einfügen und Löschen von Kanten genauer. Dabei beschränkten wir uns zuerst darauf, für einen bereits planaren Graphen mit Maximalgrad 3 mit fester Zuweisung der Valenzen zu den Facetten ein Verfahren zur Lösung der Matching-Bedingung zu entwickeln. Die Valenzen kamen dabei nur in Form von Leafs und Pairs vor und die freiwerdenden Valenzen durften nicht neu zugewiesen werden. Das Verfahren läuft in Polynomialzeit.
Im anschließenden Kapitel 4 konnten wir ein Verfahren entwickeln, welches einen planaren Graphen mit Maximalgrad 3 mit fester Zuweisung der Valenzen zu den Facetten so umbaut, dass die Parity-Bedingung erfüllt ist. Die Valenzen kamen dabei nur als Joker vor und auch hier waren Neuzuweisungen nicht gestattet. Dieses Verfahren kann ebenfalls in Polynomialzeit bezüglich der Eingabegröße durchgeführt werden.
Im Kapitel 5 konnten wir zeigen, dass wenn Neuzuweisungen und Grad-5 Knoten erlaubt sind, das Editieren eines bereits planaren Graphen mit maximal K Löschungen ein NP-vollständiges Problem darstellt. Dies bewiesen wir durch eine Reduktion von Monotone Planar 3-SAT auf unser Problem.

Nach den bereits entwickelten Ergebnissen bleiben noch einige ungeklärte Fragen.
Wir konnten zeigen, dass für das Editieren mit einigen Einschränkungen Polynomialzeit-Verfahren existieren und auch dass bei teilweiser Veränderung der Einschränkungen das Problem NP-vollständig wird. Doch wo genau liegt die Grenze dazwischen? Mit welchen Einschränkungen können wir das Editieren noch in Polynomialzeit ausführen und für welche Einschränkungen wissen wir es nicht, da diese NP-vollständige Probleme darstellen?
Dafür ist ein möglicher Ansatz eine Reduktion zu finden, welche ohne Grad-5 Knoten auskommt, also mit Maximalgrad 3, um eindeutig nachzuweisen, dass das Neuzuweisen die Problemstellung so erschwert. Dann könnte man sich auch damit befassen, für dieses NP-vollständige Problem zumindest einen approximativen Lösungsansatz zu finden.
Auf der anderen Seite wäre es auch möglich die Polynomialzeit-Verfahren weiter zu entwickeln, also zum Beispiel die Parity-Bedingung nicht nur auf Joker-Graphen herzustellen, sondern auch Branches, Islands und Triangles zuzulassen. Dieses Verfahren könnte dann noch so erweitert werden, dass zusätzlich die Matching-Bedingung erfüllt wird, was mit Branches, Islands und Triangles nicht mehr automatisch der Fall ist.
Somit könnte die Grenze zwischen lösbar und NP-vollständig weiter ausgelotet werden.

Literaturverzeichnis

[dBK10] Mark de Berg und Amirali Khosravi: *Optimal Binary Space Partitions in the Plane*. In: *Lecture Notes in Computer Science*, Seiten 216–225. Springer Berlin Heidelberg, 2010.

[DGv'H+17] Konrad K. Dabrowski, Petr A. Golovach, Pim van 't Hof, Daniël Paulusma und Dimitrios M. Thilikos: *Editing to a planar graph of given degrees*. Journal of Computer and System Sciences, 85:168–182, may 2017.

[Gab83] Harold N. Gabow: *An Efficient Reduction Technique for Degree-Constrained Subgraph and Bidirected Network Flow Problems*. In: *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, Seite 448–456, New York, NY, USA, 1983. Association for Computing Machinery, ISBN 0897910990. https://doi.org/10.1145/800061.808776.

[Gab18] Harold N. Gabow: *Data Structures for Weighted Matching and Extensions to b -matching and f -factors*. ACM Transactions on Algorithms, 14(3):1–80, jul 2018.

[HRR14] Tanja Hartmann, Jonathan Rollin und Ignaz Rutter: *Regular Augmentation of Planar Graphs*. Algorithmica, 73(2):306–370, aug 2014.

[MS12] Luke Mathieson und Stefan Szeider: *Editing graphs to satisfy degree constraints: A parameterized approach*. Journal of Computer and System Sciences, 78(1):179–191, jan 2012.

[MV80] S. Micali und V. V. Vazirani: *An O(v|v| c |E|) algoithm for finding maximum matching in general graphs*. In: *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, Seiten 17–27, Oct 1980.