technische universität
dortmund

Masterarbeit

Entwicklung eines Max-Cut-Algorithmus für fast-planare Graphen

Christine Dahn
25. September 2017

Betreuer:
Prof. Dr. Petra Mutzel
Dr. Nils Kriege

Fakultät für Informatik
Algorithm Engineering (LS 11)
Technische Universität Dortmund
http://ls11-www.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Graphen können zur Beschreibung einer Vielzahl von Optimierungsproblemen eingesetzt werden. Graphalgorithmen nutzen die spezifischen Eigenschaften der Graphen, um diese zu lösen. Im Folgenden werden Cut-Algorithmen auf Graphen genauer betrachtet. Diese finden in verschiedensten Bereichen der Optimierung Anwendung. So werden sie in der Festkörperphysik zur Bestimmung des Grundzustands von Ising-Spingläsern eingesetzt [4,6]. In der VLSI (*engl. very-large-scale-integrated*) Schaltkreis-Entwicklung werden sie unter anderem zur Minimierung der Anzahl an VIAs eingesetzt [4]. (Ein VIA, *engl. vertical interconnect access*, ist eine elektrische Verbindung zwischen verschiedenen Ebenen in einem elektrischen Schaltkreis.) In der Informatik finden Cut-Algorithmen zum Beispiel beim Lösen von quadratischen Funktionen über binären Vektoren, der 0/1-Programmierung, Anwendung [3].
Optimierungsprobleme können in zwei Kategorien geteilt werden: Maximierungsprobleme und Minimierungsprobleme. Das Max-Cut-Problem berechnet für einen gegebenen Graphen einen Schnitt mit maximaler Größe. Das Min-Cut-Problem minimiert hingegen die Größe eines Schnitts. Dabei verbinden die Kanten eines Schnitts F in $G = (V, E)$ zwei Knotenmengen V_1 und $V \setminus V_1$, sodass $\delta(V_1) = F$ gilt. Die Größe eines Schnitts ist die Summe seiner Kantengewichte. Die beiden Probleme können ineinander überführt werden, indem alle Kantengewichte des Graphen invertiert werden.
Das Min-Cut-Problem für Graphen mit ausschließlich positiven Kantengewichten, und damit auch das Max-Cut-Problem für Graphen mit ausschließlich negativen Kantengewichten ist in polynomieller Zeit berechenbar [18,28,31]. Für gewichtete Graphen mit beliebigen Kantengewichten ist das Max-Cut-Problem hingegen NP-schwer [19]. Es existieren jedoch Verfahren, die den maximalen Schnitt für Instanzen bestimmter Teilklassen der allgemeinen Graphen in polynomieller Zeit berechnen [24]. Ein bekanntes Beispiel hierfür ist die Klasse der planaren Graphen [15, 23, 25, 30].

Ziele der Arbeit

In dieser Arbeit wird die Klasse der k-fast-planaren Graphen eingeführt, die eine Oberklasse der planaren Graphen und eine Unterklasse der 1-planaren Graphen ist. Ein Graph heißt 1-planar, wenn es eine 2-dimensionale Zeichnung des Graphen gibt, sodass jede Kante höchstens eine andere Kante des Graphen schneidet. Ein Graph heißt k-fast-planar, wenn er eine 1-planare Zeichnung mit insgesamt höchstens k Kantenkreuzungen besitzt. In ähnlicher Form wurden fast-planare Graphen bereits von Hochstein definiert [16]. Allerdings lässt er offen, wie oft eine Kante geschnitten werden darf und die Anzahl der Kreuzungen wird nur mit „wenig“ beschrieben. Ziel dieser Arbeit ist es einen Max-Cut-Algorithmus für k-fast-planare Graphen auf Basis eines Max-Cut-Algorithmus für planare Graphen zu entwickeln.

Aufbau der Arbeit

Diese Arbeit ist wie folgt strukturiert: In Kapitel 2 werden grundlegende Begriffe und Notationen definiert, die in den darauf folgenden Kapiteln genutzt werden. Dazu gehören grundlegende Begriffe der Graphentheorie sowie die Graphklassen der planaren, 1-planaren und k-fast-planaren Graphen. Auch werden das Max-Cut-Problem sowie zwei Max-Cut-Algorithmen für planare Graphen vorgestellt. Zuletzt wird noch die Parametrisierbarkeit von Algorithmen und Problemen eingeführt. In Kapitel 3 werden die Eigenschaften der im vorherigen Kapitel vorgestellten Graphklassen sowie die Eigenschaften von Schnitten in k-fast-planaren Graphen untersucht. In Kapitel 4 wird zunächst ein Max-Cut-Algorithmus für 1-fast-planare Graphen entwickelt. Im Anschluss wird dieser zu einem Max-Cut-Algorithmus für k-fast-planare Graphen erweitert. Die Resultate aus Kapitel 3 werden in Kapitel 4 aufgegriffen, um die Korrektheit und Optimalität der entwickelten Max-Cut-Algorithmen für 1- bzw. k-fast-planare Graphen zu beweisen. In Abschnitt 4.3 wird ein parametrisierbarer Max-Cut-Algorithmus für 1-planare Graphen mit Parameter k vorgestellt. Kapitel 4 schließt mit der Vorstellung eines weiteren Ansatzes für einen Max-Cut-Algorithmus für 1-fast-planare Graphen und den Gründen warum dieser nicht weiter verfolgt wurde. In Kapitel 5 folgt eine Zusammenfassung der Resultate sowie ein Ausblick auf die offen gebliebenen Fragen. Im Anhang sind Beweis-Struktur-Graphen zu einigen längeren Beweisen der Arbeit gegeben. Diese stellen den Zusammenhang der Lemmata, auf denen ein Beweis basiert, graphisch dar.

Kapitel 2

Grundlagen

In diesem Kapitel werden grundlegende Begriffe und Notationen eingeführt, die im weiteren Verlauf der Arbeit benötigt werden. In Abschnitt 2.1 werden grundlegende Begriffe der Graphentheorie sowie die Graphklassen der planaren, 1-planaren und k-fast-planaren Graphen vorgestellt. In den folgenden beiden Abschnitten 2.2 und 2.3 werden das Max-Cut-Problem und zwei Max-Cut-Algorithmen für planare Graphen vorgestellt. Zuletzt wird in Abschnitt 2.5 noch auf die Parametrisierbarkeit von Algorithmen und Problemen eingegangen.

2.1 Graphen

Einige Definitionen dieses Abschnitts wurden dem Standardwerk „Graphentheorie“ von R. Diestel [7] entnommen.

2.1.1 Definition (ungerichteter Graph). Ein *Graph* G ist ein Tupel (V, E) mit $E \subseteq V \times V$. G heißt *ungerichtet*, wenn für alle $u, v \in V$ gilt: $(u, v) \in E$ gdw. $(v, u) \in E$. G hat keine *Schleifen*, wenn für alle $v \in V$ gilt: $(v, v) \notin E$.

In dieser Arbeit ist jeder Graph ungerichtet und hat keine Schleifen. Im weiteren Verlauf der Arbeit bezeichnet n die Anzahl der Knoten und m die Anzahl der Kanten in einem gegebenen Graphen.
Die Kanten (u, v) und (v, u) bezeichnen in einem ungerichteten Graphen die selbe Kante; sie wird zu uv zusammengefasst. Wenn die Knotenmenge V und die Kantenmenge E nicht explizit angegeben sind, so bezeichnet V_G die Knotenmenge des Graphen G und E_G seine Kantenmenge.

2.1.2 Definition (gewichteter Graph). Ein *gewichteter* Graph ist ein Tripel (V, E, c) mit $c : E \to \mathbb{Q}$, wobei (V, E) ein Graph ist.

Die zusätzliche Gewichtsfunktion $c : E \to \mathbb{Q}$ weist jeder Kante ein Gewicht zu. Für die Kurzschreibweise gilt: $c_e = c(e)$ für $e \in E$. Um Graphen und gewichtete Graphen

zu unterscheiden, spricht man manchmal von *ungewichteten* Graphen. Oftmals wird ein ungewichteter Graph (V, E) mit einen gewichteten Graphen (V, E, c) identifiziert, wobei in diesem Fall die Funktion c jeder Kante den Wert 1 zuweist. Wenn die Gewichtsfunktion eines gewichteten Graphen G nicht explizit angegeben ist, so bezeichnet c_G die Gewichtsfunktion von G.
In Multigraphen sind – im Gegensatz zu Graphen – Mehrfachkanten erlaubt. Daher werden Kanten nicht mehr durch ihre Endknoten beschrieben, sondern erhalten eindeutige Namen. Eine seperate Funktion f ordnet jeder Kante ihre beiden Endknoten zu.

2.1.3 Definition (Multigraph). Ein *Multigraph* ist ein Tripel $M = (V, E, f)$ mit $f : E \to V \times V$. M heißt *ungerichtet*, wenn für alle Kanten $e_1 \in E$ eine Kante $e_2 \in E \setminus \{e_1\}$ existiert, sodass gilt: $f(e_1) = (u, v)$ und $f(e_2) = (v, u)$ mit $u, v \in V$. Ein *gewichteter Multigraph* ist ein Quadrupel $M = (V, E, f, c)$ mit $c : E \to \mathbb{Q}$, wobei (V, E, f) ein Multigraph ist.

Ein Multigraph kann auch *Schleifen* enthalten; Dies sind Kanten $e \in E$, deren Endknoten identisch sind ($f(e) = vv$, $v \in V$). In einem gewichteten Multigraphen können zwei Kanten mit dem selben Endknotenpaar verschiedene Gewichte haben. In dieser Arbeit werden ausschließlich ungerichtete Multigraphen verwendet.

2.1.1 Zeichnung und Einbettung

Für die folgenden Definitionen werden zunächst einige mathematische Begriffe wiederholt. Eine *Strecke* zwischen zwei Punkten $x, y \in \mathbb{R}^2$ in der euklidischen Ebene $\mathbb{R}^2$ ist die Menge $s_{xy} = \{(1 - \lambda)x + \lambda y \mid \lambda \in [0, 1]\}$. Ein *Polygonzug* zwischen zwei Punkten $x, y \in \mathbb{R}^2$ ist eine endliche Vereinigung von aneinander anknüpfenden Strecken

$$P_{xy} = (s_{xp_0}, s_{p_0p_1}, \ldots, s_{p_\theta y})$$

mit Zwischenpunkten $p_i \in \mathbb{R}^2$. Abkürzend werden nur die Endpunkte zwischen den einzelnen Strecken notiert: $P_{xy} = (x, p_0, p_1, \ldots, p_\theta, y)$. Das *Innere eines Polygonzuges* wird mit $\mathring{P}_{xy} = P_{xy} \setminus \{x, y\}$ bezeichnet.

2.1.4 Definition (Zeichnung). Eine *Zeichnung* Γ_G des Graphen G ist eine Funktion, die jedem Knoten $v \in V_G$ eindeutige Koordinaten $(x, y) \in \mathbb{R}^2$ und jeder Kante $vw \in E_G$ einen eindeutigen Polygonzug $P_{vw}^{\Gamma_G}$ zwischen den Koordinaten seiner Endknoten zuweist.

Durch eine Zeichnung wird ein Graph eindeutig in die Ebene eingebettet. Ist durch den Kontext klar, zu welchem Graphen die Zeichnung Γ_G bzw. zu welcher Zeichnung der Polygonzug $P_{vw}^{\Gamma_G}$ gehört, so werden diese in Kurzform mit Γ bzw. P_{vw} bezeichnet. Für eine Kante $e = xy$ in einer Zeichnung Γ bezeichnen P_e und P_{xy} den selben Polygonzug; es gilt $\Gamma(e) = P_e = P_{xy} = \Gamma(xy)$. Jeder Graph hat unendlich viele verschiedene Zeichnungen.

2.1.5 Definition (Kreuzung). Die Menge $z = \{e, e'\}$ mit $e, e' \in E_G$ heißt *Kreuzung* in der Zeichnung Γ des Graphen G, wenn gilt:

$$\mathring{P}_e^\Gamma \cap \mathring{P}_{e'}^\Gamma \neq \emptyset$$

2.1.6 Definition (einfache Kreuzung). Eine Kreuzung $z = \{e, e'\}$ in Γ heißt *einfach*, wenn gilt:

$$\left|\mathring{P}_e^\Gamma \cap \mathring{P}_{e'}^\Gamma\right| = 1$$

2.1.7 Definition (Eckknoten einer Kreuzung). Die vier *Eckknoten einer Kreuzung* $z = \{ac, bd\}$ sind die Endknoten der Kreuzungskanten: a, b, c, d. Die Eckknoten a und c bzw. b und d heißen *gegenüberliegend*, da sie jeweils Endknoten einer Kreuzungskante sind. Zwei Eckknoten heißen *nebeneinanderliegend*, wenn sie nicht gegenüberliegend sind.

2.1.8 Definition (Kreuzungszahl). Die *Anzahl an Kantenkreuzungen* in der Zeichnung Γ_G wird mit $\sigma(\Gamma_G)$ bezeichnet. Es gilt:

$$\sigma(\Gamma_G) = \left| \bigcup_{e,e' \in E_G,\ e \neq e'} \mathring{P}_e^{\Gamma_G} \cap \mathring{P}_{e'}^{\Gamma_G} \right|$$

2.1.9 Definition (Knoten-Nachbarschaft). Zwei Knoten in einem Graphen G heißen *benachbart*, wenn sie durch eine Kante in E_G verbunden sind. Die *Nachbarn* eines Knoten $v \in V_G$ sind alle Knoten, die zu v benachbart sind.

Sei $\pi : X \to X$ eine bijektive Abbildung mit $\pi(x) \neq x\ \forall x \in X$. Dann heißt $(x_{i_1}, \ldots, x_{i_\theta})$ *Zyklus* auf X, wenn gilt: $\pi : x_{i_1} \mapsto x_{i_2},\ x_{i_2} \mapsto x_{i_3},\ \ldots\ ,\ x_{i_{\theta-1}} \mapsto x_{i_\theta},\ x_{i_\theta} \mapsto x_{i_1}$ und $\{x_{i_1}, \ldots, x_{i_\theta}\} = X$.

2.1.10 Definition (sortierte Knoten-Nachbarschaft). Seien $v_1, \ldots, v_\theta \in V_G$ die Nachbarn des Knoten $v \in V_G$. Die *sortierten Nachbarn* von v in Γ_G sind ein Zyklus $(v_{i_1}, \ldots, v_{i_\theta})$ auf $\{v_1, \ldots, v_\theta\}$, der die Nachbarn von v in der Reihenfolge der von v ausgehenden Kanten in Γ_G gegen den Uhrzeigersinn sortiert.

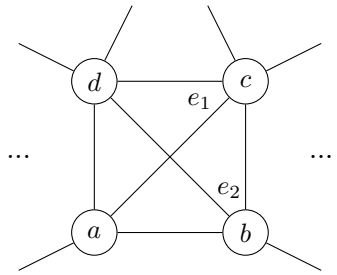
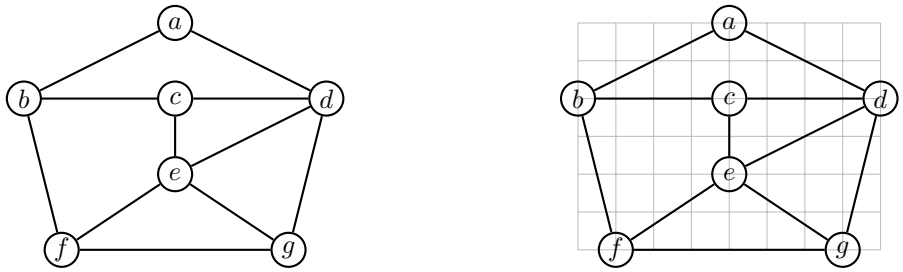


Abbildung 2.1: Eine einfache Kreuzung $z = \{e_1, e_2\}$.



(a) Graph H **(b)** Zeichnung Γ_H

Abbildung 2.2: Der Graph H und seine Zeichnung Γ_H.

2.1.11 Definition (Einbettung). Die *Einbettung* Π_Γ des Graphen G zur Zeichnung Γ weist jedem Knoten aus $v \in V_G$ seine sortierten Nachbarn $(v_{i_1}, \ldots, v_{i_\theta})$ in Γ zu.

Anstatt Π_Γ wird die Kurzform Π verwendet, wenn sowohl der Graph als auch die Zeichnung aus dem Kontext ersichtlich sind. Jede Zeichnung definiert eine eindeutige Einbettung. Zu einer Einbettung gibt es jedoch viele verschiedene Zeichnungen.

2.1.12 Beispiel (Zeichnung, Einbettung). Wir betrachten den Graphen H mit Knotenmenge $V_H = \{a, b, c, d, e, f, g\}$ und Kantenmenge $E_H = \{ab, ad, bc, bf, cd, ce, de, dg, ef, eg, fg\}$ sowie dessen Zeichnung Γ_H (s. Abb. 2.2). Die Zeichnung Γ_H ist gegeben durch:

$$\begin{aligned}\Gamma_H : a \mapsto (4,6),\ b \mapsto (0,4),\ c \mapsto (4,4),\ d \mapsto (8,4),\\ e \mapsto (4,2),\ f \mapsto (1,0),\ g \mapsto (7,0),\\ uv \mapsto (\Gamma_H(u), \Gamma_H(v)) \quad \forall uv \in E_H\end{aligned}$$

Die Zeichnung Γ_H definiert die Einbettung Π_H. Diese ist gegeben durch:

$$\begin{aligned}\Pi_H : a \mapsto (b,d),\ b \mapsto (f,c,a),\ c \mapsto (b,e,d),\\ d \mapsto (a,c,e,g),\ e \mapsto (f,g,d,c),\\ f \mapsto (b,g,e),\ g \mapsto (d,e,f)\end{aligned}$$

2.1.2 Kontraktion und Minoren

Eine Kontraktion auf einem Graphen G entsteht durch die Verschmelzung von zwei Knoten in G. Alle Kanten, die zuvor zu einem der beiden Knoten führten, führen nun zu dem neuen Knoten. Die Kantengewichte werden übertragen. Falls von einem Knoten jeweils eine Kante zu den beiden zu verschmelzenden Knoten führt, so werden deren Gewichte für die neue Kante aufaddiert.

2.1.13 Definition (Knotenkontraktion). Gegeben sei ein ungerichteter gewichteter Graph $G = (V, E, c)$ und zwei Knoten $x, y \in V$. Der gewichtete Graph $G/xy = (V', E', c')$ heißt *Kontraktion von x und y* auf G, wenn folgendes gilt:

$$V' = (V \backslash \{x, y\}) \cup \{v_{xy}\} \qquad \text{, wobei } v_{xy} \notin V \cup E$$
$$E' = \{vw \in E \mid x, y \notin \{v, w\}\} \cup \{uv_{xy} \mid ux \in E \backslash \{xy\} \ \vee \ uy \in E \backslash \{xy\}\}$$
$$c'_{vw} = \begin{cases} c_{vw} & \text{, wenn } vw \in E' \cap E \\ c_{wx} & \text{, wenn } v = v_{xy} \ \wedge \ wx \in E \ \wedge \ wy \notin E \\ c_{wy} & \text{, wenn } v = v_{xy} \ \wedge \ wx \notin E \ \wedge \ wy \in E \\ c_{wx} + c_{wy} & \text{, wenn } v = v_{xy} \ \wedge \ wx \in E \ \wedge \ wy \in E \end{cases}$$

Der *kontrahierte Knoten* von G/xy gegenüber G, der durch die Kontraktion von x und y auf G entstanden ist, wird mit v_{xy} bezeichnet. Um mehrere Knoten $U = x_1 x_2 \dots x_\theta$ mit $x_i \neq x_j$ für $i \neq j$ zu einem Knoten zu kontrahieren schreiben wir:

$$G/U = G/x_1 x_2 \dots x_\theta = (\ \dots((G/x_1 x_2)/v_{x_1 x_2} x_3)/\dots\ )/v_{x_1 \dots x_{\theta-1}} x_\theta$$

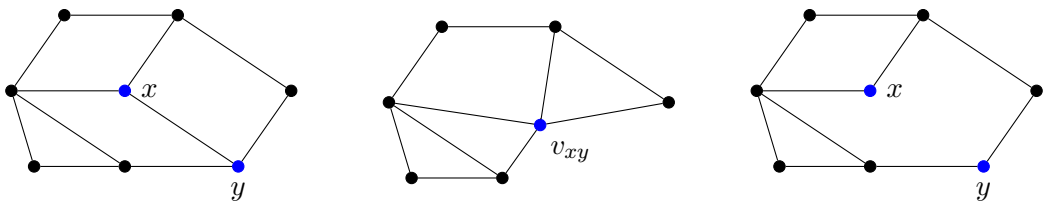
Der *kontrahierte Knoten* von G/U gegenüber G, der durch die *Kontraktion von U* auf G entstanden ist, wird mit v_U bezeichnet. Um mehrere Kontraktionen hintereinander durchzuführen schreiben wir:

$$G/X = (\ \dots((G/U_1)/U_2)/\dots\ )/U_\theta$$

mit $X = \{U_i \mid U_i \subseteq V \ \wedge \ U_i \cap U_j = \emptyset \text{ für } i, j = 1, \dots, \theta, i \neq j\}$.
Die Knotenkontraktion ist *rechtskommutativ*, da die kontrahierten Knotenmengen disjunkt sind und sich untereinander nicht beeinflussen; d. h. es gilt: $(G/U_1)/U_2 = (G/U_2)/U_1$. Im Folgenden wird die inverse Operation zur Knotenkontraktion definiert.

2.1.14 Definition (Knotenexpansion). Ein Graph G heißt *Knotenexpansion von $v \in V_K$* auf K, wenn gilt:

$$\exists x, y \in V_G : K = G/xy \ \wedge \ v = v_{xy}$$



(a) G **(b)** G/xy **(c)** $G - xy$

Abbildung 2.3: Ein Beispiel für die Knotenkontraktion von x und y in G und die Löschung der Kante xy aus G. (Idee aus [7])

Um die Teilmengen $W \subseteq V_K$ und $F \subseteq E_K$ zurück auf G abzubilden, schreiben wir:

$$\kappa_{xy}^{-1}(W) = \begin{cases} (W \setminus \{v_{xy}\}) \cup \{x, y\} & \text{, wenn } v_{xy} \in W \\ W & \text{, sonst} \end{cases}$$

$$\kappa_{xy}^{-1}(F, E_G) = \{uw \in F \mid v_{xy} \notin \{u, w\}\} \cup (\{ux, uy \mid uv_{xy} \in F\} \cap E_G) \subseteq E_G$$

2.1.15 Definition (Kantenlöschung). Gegeben sei ein gewichteter Graph $G = (V, E, c)$ und eine Kante $e \in E$. Der durch *Löschung der Kante e* aus G entstehende Graph wird mit $G - e$ bezeichnet. Es gilt $G - e = (V, E \setminus \{e\}, w)$ mit $w_{e'} = c_{e'}$ für alle $e' \in E \setminus \{e\}$.

Für Kantenmengen $X \subseteq E$ gilt entsprechend: $G - X$ ist der durch *Löschung aller Kanten* $e \in X$ aus G entstehende Graph. Die Kantenlöschung ist *rechtskommutativ*, da die Löschung einer Kante keine anderen Kanten oder Knoten verändert; d. h. es gilt: $(G - e_1) - e_2 = (G - e_2) - e_1$. In Abbildung 2.3 ist jeweils ein Beispiel für eine Knotenkontraktion und eine Kantenlöschung angegeben. Die Graphen sind von Beispielen aus dem Standardwerk „Graphentheorie“ von R. Diestel [7] inspiriert.

2.1.16 Definition (*xy*-Weg). Ein xy-Weg in einem Graphen G mit $x, y \in V_G$ ist eine Folge von Kanten $e_1, \ldots, e_\rho$, für die gilt:

1. $e_1 = xv_2$ und $e_\rho = v_\rho y$ mit $v_2, v_\rho \in V_G$
2. $e_i = v_i v_{i+1}$ und $e_{i+1} = v_{i+1} v_{i+2}$ mit $v_i, v_{i+1}, v_{i+2} \in V_G$ für alle $i \in \{2, \rho - 2\}$

Die Knoten $v_2, \ldots v_\rho$ heißen *innere Knoten* des xy-Weges.

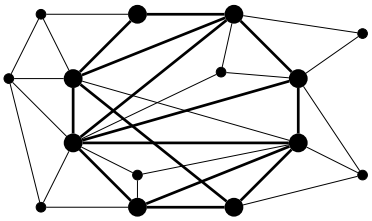
2.1.17 Definition (zusammenhängend). Ein Graph G heißt zusammenhängend, wenn für alle Knoten $x, y \in V_G$ ein xy-Weg in G existiert.

2.1.18 Definition (k-zusammenhängend). Ein Graph G heißt k-zusammenhängend, wenn $|E_G| > k$ gilt und $G - X$ für alle $X \subseteq E_G$ mit $|X| < k$ zusammenhängend ist.

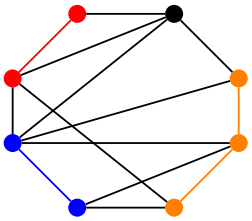
2.1.19 Definition (Teilgraph). Ein Graph G ist ein *Teilgraph* des Graphen Y, wenn G durch das Entfernen von Knoten und/oder Kanten aus Y entsteht oder $G = Y$ gilt.

2.1.20 Definition (Minor). Ist G ein Teilgraph von Y und entsteht X aus G durch eine Folge von Knotenkontraktionen, wobei die kontrahierten Knoten durch eine Kante in G verbunden sind, so ist X ein *Minor* von Y.

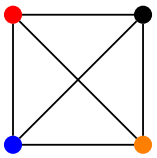
2.1.21 Definition (Unterteilung). Ein Graph G ist eine *Unterteilung* des Graphen X, wenn jede Kante xy in E_X einem xy-Weg in G entspricht und die inneren Knoten dieses Weges jeweils genau Knotengrad 2 haben.



(a) Graph Y_1

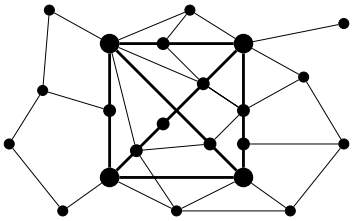


(b) G_1 ist Teilgraph von Y_1

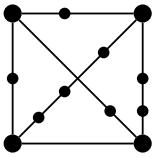


(c) X ist Minor von Y_1

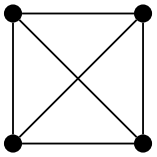
Abbildung 2.4: Der Graph Y_1, sein Teilgraph G_1 und sein Minor X. Die Farben in G_1 und X zeigen an, welche Knoten kontrahiert werden. (Idee aus [7])



(a) Graph Y_2



(b) G_2 ist Teilgraph von Y_2



(c) X top. Minor von Y_2

Abbildung 2.5: Der Graph Y_2, sein Teilgraph G_2 und sein topologischer Minor X. (Idee aus [7])

Ein innerer Knoten zeichnet sich dadurch aus, dass er keine Kanten zu Knoten hat, die nicht auf seinem xy-Weg liegen.

2.1.22 Definition (topologischer Minor). Ist G ein Teilgraph von Y und eine Unterteilung von X, so ist X ein *topologischer Minor* von Y.

2.1.23 Korollar. *Ein topologischer Minor von Y ist ebenfalls ein Minor von Y.*

In den Abbildungen 2.4 und 2.5 ist jeweils ein Beispiel für einen Minor und einen topologischen Minor angegeben. Die Graphen sind von Beispielen aus dem Standardwerk „Graphentheorie“ von R. Diestel [7] inspiriert.

2.1.3 Planare Graphen

Eine Unterklasse der allgemeinen Graphen sind die planaren Graphen. Ein Graph heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann. Kreuzungsfrei heißt, dass die Polygonzüge verschiedener Kanten sich nicht schneiden.

2.1.24 Definition (planare Zeichnung). Die *Zeichnung* Γ_G eines Graphen G heißt *planar*, wenn gilt:

$$\sigma(\Gamma_G) = 0$$

Die *Flächen* f_i einer planaren Zeichnung Γ_G sind die einzelnen Gebiete, in die die Kanten von G die Ebene unterteilen. Jede Fläche ist eindeutig durch ihre begrenzenden Kanten (bzw. die Reihenfolge ihrer Eckknoten) bestimmt. Eine Fläche ist unbeschränkt: die *Außenfläche* $\varphi(\Gamma_G)$. Die *Flächenmenge* der planaren Zeichnung Γ_G wird mit $\mathrm{F}(\Gamma_G)$ bezeichnet. Abkürzend wird die Außenfläche mit φ_Γ und die Flächenmenge mit F_Γ oder F_G bezeichnet.

2.1.25 Definition (planare Einbettung). Das Tupel $(\Pi_\Gamma, \varphi_\Gamma)$ heißt *planare Einbettung* zu Γ, wenn Γ eine planare Zeichnung und Π_Γ deren Einbettung ist.

2.1.26 Definition (planarer Graph). Ein Graph heißt *planar*, wenn für ihn eine planare Zeichnung existiert.

Wird zu einer planaren Einbettung keine explizite Außenfläche angegeben, so heißt die Einbettung *kombinatorisch*. Als abkürzende Notation werden planare Zeichnungen mit Γ^0 und planare (kombinatorische) Einbettungen mit Π^0 bezeichnet.

2.1.27 Beispiel (planare Zeichnung/ Einbettung, Fläche). Wir betrachten wieder den Graphen H aus Beispiel 2.1.12. Dieses Mal betrachten wir zwei verschiedene Zeichnungen von H, zu sehen in Abbildung 2.6. Die Zeichnung Γ_1 ist nicht planar, da sie vier Kantenkreuzungen enthält. Sie induziert die Einbettung Π_1.

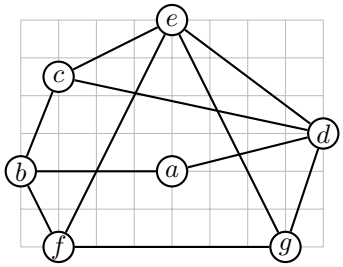
$$\begin{aligned}\Pi_1 : a &\mapsto (b,d),\ b \mapsto (f,a,c),\ c \mapsto (b,d,e),\ d \mapsto (a,g,e,c),\\ e &\mapsto (f,g,d,c),\ f \mapsto (g,e,b),\ g \mapsto (d,e,f)\end{aligned}$$

Die Zeichnung Γ_2 ist planar, da sie keine Kantenkreuzungen enthält. Sie induziert die kombinatorische Einbettung Π_2.

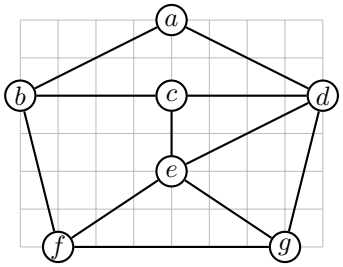
$$\begin{aligned}\Pi_2 : a &\mapsto (b,d),\ b \mapsto (f,c,a),\ c \mapsto (b,e,d),\ d \mapsto (a,c,e,g),\\ e &\mapsto (f,g,d,c),\ f \mapsto (g,e,b),\ g \mapsto (d,e,f)\end{aligned}$$

Die Flächen der planaren Zeichnung Γ_2 sind in Abbildung 2.6c zu sehen. Es gilt:

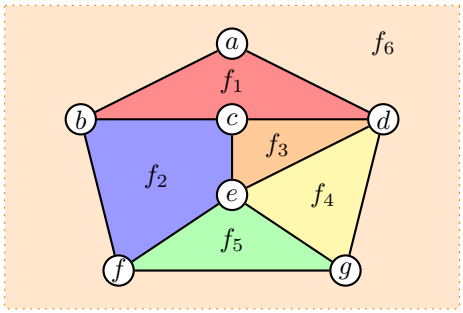
$$\begin{aligned}f_1 &= (a,b,c,d),\ f_3 = (c,e,d),\ f_5 = (e,f,g),\\ f_2 &= (b,f,e,c),\ f_4 = (d,e,g),\ f_6 = (a,d,g,f,b) = \varphi(\Gamma_2)\end{aligned}$$



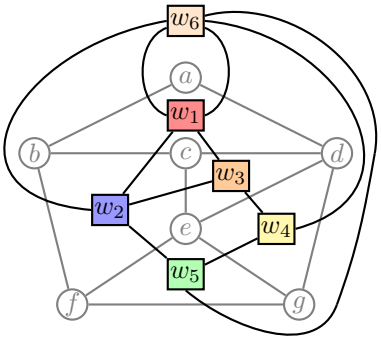
(a) Nicht planare Zeichnung Γ_1 von H.



(b) Planare Zeichnung Γ_2 von H.



(c) Flächen der planaren Zeichnung Γ_2 von H.



(d) Der duale Graph zur Zeichnung Γ_2.

Abbildung 2.6: Zwei verschiedene Zeichnungen des planaren Graphen H sowie die Flächen seiner planaren Zeichnung.

Das Tupel (Π_2, f_6) ist die planare Einbettung zur planaren Zeichnung Γ_2 von H. Somit ist H planar.

2.1.28 Definition (dualer Graph). Der *duale Graph* zu einer planaren Zeichnnung Γ_G ist ein Multigraph D, für den gilt:

i) Für jede Fläche $f_i \in \mathrm{F}(\Gamma_G)$ existiert genau ein korrespondierender Knoten $w_i \in V_D$.

ii) Für jede Kante $e' \in E_G$ mit angrenzenden Flächen f_i und f_j existiert eine Kante $e \in E_D$, so dass e die beiden Knoten w_i und w_j, die den Flächen f_i und f_j entsprechen, verbindet. Die Kante $e \in E_D$ ist die duale Kante zu $e' \in E_G$.

iii) Falls G ein gewichteter Graph ist, so werden die Kantengewichte in G auf die korrespondierenden dualen Kanten in D übertragen und D wird ein gewichteter Multigraph.

In Abbildung 2.6d ist der duale Graph zur planaren Zeichnung Γ_2 (Abb. 2.6c) zum Graphen H aus Beispiel 2.1.12 zu sehen.

2.1.4 1-planare Graphen

Eine Oberklasse der planaren Graphen sind die 1-planaren Graphen nach Ringel [29]. Ein Graph heißt 1-planar, wenn er in die Ebene eingebettet werden kann, sodass jede Kante höchstens einmal gekreuzt wird; d. h. für jede Kante gilt: der ihr zugehörige Polygonzug schneidet höchstens einen Polygonzug einer anderen Kante (gemeinsame Endpunkte zählen nicht als Schnitt). Solche Kantenkreuzungen heißen einfache Kreuzungen (Def. 2.1.6).

2.1.29 Definition (1-planare Zeichnung). Die *Zeichnung* Γ_G eines Graphen G heißt *1-planar*, falls gilt:

$$\forall e \in E_G : \left| \bigcup_{e' \in E_G \setminus \{e\}} \mathring{P}_e \cap \mathring{P}_{e'} \right| \leq 1$$

Die *Menge der sich kreuzenden Kanten* in der Zeichnung Γ_G wird mit $\mathrm{I}(\Gamma_G) = \{\{e, e'\} \mid e, e' \in E_G \ \wedge \ \mathring{P}_e \cap \mathring{P}_{e'} \neq \emptyset\}$ bezeichnet.

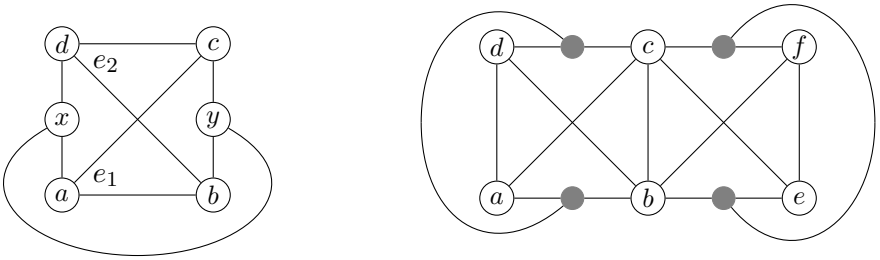
Die *Flächen* f_i einer 1-planaren Zeichnung Γ_G sind die einzelnen Gebiete, in die die Kanten von G die Ebene unterteilen. Jede Fläche ist eindeutig durch die Reihenfolge ihrer Eckknoten und ggf. angrenzenden Kreuzung $z \in \mathrm{I}(\Gamma_G)$ bestimmt. Eine Fläche ist unbeschränkt: die *Außenfläche* $\varphi(\Gamma_G)$ (kurz: φ_Γ). Die *Flächenmenge* der 1-planaren Zeichnung Γ_G wird mit $\mathrm{F}(\Gamma_G)$ (kurz: F_Γ oder F_G) bezeichnet.

2.1.30 Definition (1-planare Einbettung). Das Tupel $(\Pi_\Gamma, \varphi_\Gamma)$ heißt *1-planare Einbettung* zu Γ, wenn Γ eine 1-planaren Zeichnung und Π_Γ deren Einbettung ist.

2.1.31 Definition (1-planarer Graph). Ein Graph heißt *1-planar*, wenn für ihn eine 1-planare Zeichnung existiert.

In Abbildung 2.7 sind zwei 1-planare Graphen zu sehen.

2.1.32 Satz (Korzhik und Mohar 2013 [21]). *Einen Graphen auf 1-Planarität zu testen ist NP-schwer.*



(a) Ein 1-fast-planarer Graph. **(b)** Ein 2-fast-planarer Graph.

Abbildung 2.7: Zwei 1-planare Graphen.

2.1.5 k-fast-planare Graphen

Als Unterklasse der 1-planaren Graphen werden in diesem Abschnitt die k-fast-planaren Graphen definiert. Ein Graph heißt k-fast-planar, wenn er in die Ebene eingebettet werden kann, sodass er höchstens k einfache Kreuzungen enthält. D. h. es gibt höchstens k (innere) Schnittpunkte von jeweils zwei Polygonzügen verschiedener Kanten und jeder Polygonzug schneidet höchstens einen anderen Polygonzug.

2.1.33 Definition (k-fast-planare Zeichnung). Die *Zeichnung* Γ_G eines Graphen G heißt *k-fast-planar*, wenn sie 1-planar ist und $\sigma(\Gamma_G) \leq k$ gilt. Die *Menge der sich kreuzenden Kanten* in Γ_G wird mit $\mathrm{I}(\Gamma_G) = \{\{e, e'\} \mid e, e' \in E_G \ \wedge \ \mathring{P}_e \cap \mathring{P}_{e'} \neq \emptyset\}$ bezeichnet.

Die *Flächen* f_i einer k-fast-planaren Zeichnung Γ_G sind die einzelnen Gebiete, in die die Kanten von G die Ebene unterteilen. Jede Fläche ist eindeutig durch die Reihenfolge ihrer Eckknoten und ggf. angrenzenden Kreuzungen $z_i \in \mathrm{I}(\Gamma_G)$ bestimmt. Eine Fläche ist unbeschränkt: die *Außenfläche* $\varphi(\Gamma_G)$. Die *Flächenmenge* der k-fast-planaren Zeichnung Γ_G wird mit $\mathrm{F}(\Gamma_G)$ bezeichnet. Abkürzend wird die Außenfläche mit φ_Γ und die Flächenmenge mit F_Γ oder F_G bezeichnet.

2.1.34 Definition (k-fast-planare Einbettung). Das Tupel $(\Pi_\Gamma, \varphi_\Gamma)$ heißt *k-fast-planare Einbettung* zu Γ, wenn Γ eine k-fast-planaren Zeichnung und Π_Γ deren Einbettung ist.

2.1.35 Definition (k-fast-planarer Graph). Ein Graph heißt *k-fast-planar*, wenn für ihn eine k-fast-planare Zeichnung existiert.

In Abbildung 2.7 sind ein 1-fast-planarer und ein 2-fast-planarer Graph zu sehen.

2.1.36 Definition (echt k-fast-planarer Graph). Ein Graph G heißt *echt k-fast-planar*, wenn für alle seine 1-planaren Zeichnungen Γ_G gilt: $\sigma(\Gamma_G) \geq k$.

Aus den Definitionen 2.1.33 und 2.1.36 folgt eine weitere Charakterisierung eines echt k-fast-planaren Graphen.

2.1.37 Korollar. *Ein Graph ist echt k-fast-planar, wenn er k-fast-planar und nicht $(k-1)$-fast-planar ist.*

Die k-fast-planare Zeichnung zu einem echt k-fast-planaren Graphen ist kreuzungsminimal. An die Notation der planaren Zeichnungen und Einbettungen angelehnt, werden k-fast-planare Zeichnungen mit Γ^k und k-fast-planare (kombinatorische) Einbettungen mit Π^k bezeichnet.

2.2 Max-Cut-Problem

Zunächst werden einige Grundbegriffe definiert, die für das MAX-CUT-Problem benötigt werden. Diese Definitionen orientieren sich an den Definitionen in [26].

2.2.1 Definition (Schnitt). Gegeben sei ein ungerichteter Graph $G = (V, E)$. Eine Kantenmenge $F \subseteq E$ heißt *Schnitt* (*engl. cut*) in G, wenn eine Knotenmenge $V_1 \subseteq V$ existiert, so dass gilt:

$$F = \delta(V_1) = \{uv \in E \mid u \in V_1, v \in \overline{V}_1 = V \backslash V_1\}$$

Der Schnitt F enthält somit alle Kanten in G zwischen V_1 und $\overline{V}_1$. Die Knotenmenge, die den Schnitt F definiert, wird mit $V_1(F)$ bezeichnet.

2.2.2 Definition (Wert eines Schnitts). Für einen Schnitt $F \subseteq E_G$ in einem gewichteten Graphen $G = (V, E, c)$ bezeichnet

$$\chi(F) = \sum_{e \in F} c_e$$

den *Wert des Schnitts* F. Für einen ungerichteten Graphen G gilt: $\chi(F) = |F|$.

Der Wert des Schnitts F in einem gewichteten Graphen ist somit die Summe der Kantengewichte aller Kanten in F. Da jede Kante in einem ungewichteten Graphen G das Gewicht 1 hat, entspricht der Wert eines Schnitts in G der Anzahl der Kanten im Schnitt.

2.2.3 Definition (größerer Schnitt). Gegeben sei ein ungerichteter Graph G und zwei Schnitte $F_1, F_2 \subseteq E_G$ in G. Der Schnitt F_1 heißt *größer als* F_2, wenn $\chi(F_1) > \chi(F_2)$ gilt. F_2 heißt *kleiner als* F_1.

Das MAX-CUT-Problem kann als Entscheidungsproblem oder als Optimierungsproblem definiert werden. Im MAX-CUT-Entscheidungsproblem (2.2.4) wird nur gefragt, ob es einen Schnitt im gegebenen Graphen gibt, dessen Wert größer oder gleich einer vorgegebenen natürlichen Zahl s ist. Das MAX-CUT-Optimierungsproblem (2.2.5) berechnet hingegen einen konkreten Schnitt im gegebenen Graphen, dessen Wert maximal ist. Man kann das MAX-CUT-Problem für gewichtete und ungewichtete Graphen betrachten.

2.2.4 Problem (Max-Cut).
Gegeben: Ungerichteter Graph G, $s \in \mathbb{N}$.
Gesucht: Gibt es einen Schnitt F in G mit $\chi(F) \geq s$?

2.2.5 Problem (Max-Cut-O).
Gegeben: Ungerichteter Graph G.
Gesucht: $\underset{F \subseteq E_G}{\operatorname{argmax}} \chi(F)$, wobei F ein Schnitt in G ist.

Das Max-Cut-Problem lässt sich analog für planare, 1-planare und k-fast-planare Graphen definieren. Im folgenden sind die Max-Cut-Optimierungsprobleme für die drei Graphklassen gegeben. Die Max-Cut-Entscheidungsprobleme lassen sich daraus herleiten, indem der maximale Schnitt mit Max-Cut-O bestimmt wird und im Anschluss überprüft wird, ob dessen Wert größer oder gleich s ist.

2.2.6 Problem (Max-Cut-O für planare Graphen).

Gegeben: Ungerichteter planarer Graph G.

Gesucht: $\underset{F \subseteq E_G}{\operatorname{argmax}} \chi(F)$, wobei F ein Schnitt in G ist.

2.2.7 Problem (Max-Cut-O für 1-planare Graphen).

Gegeben: Ungerichteter 1-planarer Graph G.

Gesucht: $\underset{F \subseteq E_G}{\operatorname{argmax}} \chi(F)$, wobei F ein Schnitt in G ist.

2.2.8 Problem (Max-Cut-O für k-fast-planare Graphen).

Gegeben: Ungerichteter k-fast-planarer Graph G, $k \in \mathbb{N}$.

Gesucht: $\underset{F \subseteq E_G}{\operatorname{argmax}} \chi(F)$, wobei F ein Schnitt in G ist.

2.3 Max-Cut-Algorithmus für planare Graphen

Karp zeigte 1972, dass das Max-Cut-Problem für allgemeine ungewichtete Graphen NP-schwierig ist [19]. Für gewisse Teilklassen der Graphen gibt es jedoch Polynomialzeitalgorithmen. Ein bekanntes Beispiel ist die Klasse der planaren Graphen [15].

Planare Graphen

In den 1970er Jahren entwickelten Orlova und Dorfman (1972) und Hadlock (1975) unabhängig voneinander einen Max-Cut-Algorithmus für planare Graphen mit nicht-negativen Gewichten [15, 27]. Beide Ansätze nutzen die Dualität des Max-Cut-Problems in planaren Graphen zum Max-Eulerkreis-Problem. Denn ein (maximaler) Schnitt im planaren Graph entspricht einem (maximalen) Eulerkreis in dessen dualem Graphen [15]. Das Max-Eulerkreis-Problem kann auf das Chinese-Postman-Problem übertragen werden. Letzteres wurde von Edmonds und Johnson 1973 auf das Min-T-Join-Problem reduziert, welches durch eine Kombination des Kürzeste-Wege- und Min-Matching-Problems gelöst werden kann [8]. Da die Berechnung des Matchings die Laufzeit dominiert, beträgt die Laufzeit des Max-Cut-Algorithmus von Hadlock $\mathcal{O}(n^3)$. Der Max-Cut-Algorithmus für ungewichtete planare Graphen von Hadlock ist in Algorithmus 2.1 zu sehen [15].
Mutzel stellte 1990 einen Polynomialzeitalgorithmus für das Max-Cut-Problem für planare Graphen mit beliebigen Gewichten vor [25]. Um die negativen Kantengewichte zu berücksichtigen, löst sie das Min-T-Join-Problem mit den Beträgen der Kantengewichte. Die Laufzeit ihres Algorithmus beträgt $\mathcal{O}(n^3)$. Ihr angepasster Max-Cut-Algorithmus für

planare Graphen ist in Algorithmus 2.2 abgebildet [25, 26]. Im selben Jahr stellten Shih, Wu und Kuo ebenfalls einen MAX-CUT-Algorithmus für planare Graphen mit beliebigen Gewichten vor [30]. Die Laufzeit ihres Algorithmus beträgt $\mathcal{O}(n^{3/2} \log n)$. Dies ist die bisher beste obere Schranke für die Laufzeit eines MAX-CUT-Algorithmus für planare Graphen. Des Weiteren existieren Polynomialzeitalgorithmen für das MAX-CUT-Problem auf schwach bipartiten Graphen [14], auf Graphen mit positiven Gewichten, die nicht zum K_5 kontrahiert werden können [2], auf Graphen, die keine langen ungeraden Kreise enthalten [13], und auf Graphen, die kreuzungsfrei auf einem Torus eingebettet werden können [12]. Die ersten beiden Graphklassen sind jeweils Oberklassen der planaren Graphen [14]. Liers und Pardella stellten 2009 einen neuen Ansatz für einen MAX-CUT-Algorithmus für planare Graphen vor. Dieser hat ebenfalls eine Laufzeit von $\mathcal{O}(n^{3/2} \log n)$, soll jedoch in der Praxis schneller laufen als der bis dahin schnellste MAX-CUT-Algorithmus für planare Graphen [23].
Für die Notation im Pseudocode der Algorithmen von Hadlock und Mutzel wird der Begriff der symmetrischen Differenz eingeführt.

2.3.1 Definition (symmetrische Differenz $\triangle$). Für eine Menge von Teilmengen $\{T_1, \ldots, T_l\} \subseteq \mathcal{P}(M)$ einer Menge M ist die *symmetrische Differenz* der Mengen $T_i \subseteq M$ mit $i \in I = \{1, \ldots, l\}$ wie folgt definiert:

$$\triangle_{i \in I} T_i = \bigcup_{i \in I} T_i \setminus \bigcap_{i \in I} T_i$$

2.3.2 Satz (Optimalität und Korrektheit [25, 26]). *Algorithmus 2.2 berechnet einen maximalen Schnitt für einen gewichteten planaren Graphen G.*

2.3.3 Satz (Laufzeit [25, 26]). *Die Laufzeit des Algorithmus 2.2 beträgt $\mathcal{O}(n^3)$ bei Eingabe eines planaren Graphen G mit n Knoten.*

$\underline{\texttt{MaxCut}_0(G, \Gamma_G^0)}$

Eingabe: ungerichteter planarer Graph G mit Zeichnung Γ_G^0

Ausgabe: maximaler Schnitt $F \subseteq E_G$

1: $D \leftarrow$ dualer Graph zu Γ_G^0
2: $T \leftarrow \{v \in V_D \mid v \text{ hat ungeraden Grad}\}$
3: **for each** $vw \in T^2$ **do**
4: $\quad \Phi_{vw} \leftarrow$ kürzester vw-Weg in D
5: $\quad c_{vw} \leftarrow |\Phi_{vw}|$
6: **end for**
7: $H \leftarrow (T, T^2, c)$ // berechne vollständigen Graphen
8: $M \leftarrow$ minimales perfektes Matching in H
9: $J \leftarrow \triangle_{vw \in M} \Phi_{vw}$
10: $F_D \leftarrow E_D \setminus J$ // berechne Eulermenge
11: $F \leftarrow$ zu F_D dualen Kanten aus E_G

Algorithmus 2.1: MAX-CUT-Algorithmus für planare Graphen [15, 26]

$\underline{\texttt{GewMaxCut}_0(G, \Gamma_G^0)}$

Eingabe: ungerichteter gewichteter planarer Graph $G = (V_G, E_G, c)$ mit $c : E_G \to \mathbb{Q}$ und Zeichnung Γ_G^0

Ausgabe: maximaler Schnitt $F \subseteq E_G$

1: $D = (V_D, E_D, w) \leftarrow$ dualer gewichteter Graph zu Γ_G^0
2: $E_D^+ \leftarrow \{e \in E_D \mid w_e \geq 0\}$
3: $E_D^- \leftarrow \{e \in E_D \mid w_e < 0\}$
4: $T \leftarrow \{v \in V_D \mid v \text{ hat ungeraden Grad in } (V_D, E_D^+)\}$
5: **for each** $vw \in T^2$ **do**
6: $\quad \Phi_{vw} \leftarrow$ kürzester vw-Weg in D bezüglich $|w|$
7: $\quad t_{vw} \leftarrow \sum_{e \in \Phi_{vw}} |w_e|$
8: **end for**
9: $H \leftarrow (T, T^2, t)$ // berechne vollständigen Graphen
10: $M \leftarrow$ minimales perfektes Matching in H
11: $J \leftarrow \triangle_{vw \in M} \Phi_{vw}$
12: $F_D \leftarrow (E_D^+ \setminus J) \cup (E_D^- \cap J)$ // berechne Eulermenge
13: $F \leftarrow$ zu F_D dualen Kanten aus E_G

Algorithmus 2.2: MAX-CUT-Algorithmus für gewichtete planare Graphen [25, 26]

2.4 Zeichenalgorithmen

Fárys Theorem besagt, dass für jeden planaren Graphen eine planare Zeichnung mit ausschließlich geraden Kanten existiert [9, 11]. Fraysseix, Pach und Pollack stellten 1988 ein Verfahren vor, welches eine solche Zeichnung für planare Graphen mit n Knoten mit $\mathcal{O}(n)$ Speicherplatz in $\mathcal{O}(n \log n)$ Zeit berechnet [11]. Somit kann aus einem gegebenen planaren Graphen eine planare Zeichnung des Graphen in polynomieller Zeit berechnet werden.
Einen Graphen auf 1-Planarität zu testen ist nach Korzhik und Mohar (2013) NP-schwer [21]. Daher muss zum Zeichnen eines 1-planaren Graphen in polynomieller Zeit eine 1-planare Einbettung des Graphen gegeben sein [17, 20]. Für 1-planare Graphen ist die Berechnung einer Zeichnung mit ausschließlich geraden Kanten bisher nur für Teilklassen gelöst. Hong et. al. [17] stellten 2012 einen Polynomialzeitalgorithmus vor, der eine 1-planare Zeichnung mit ausschließlich geraden Kanten für eine Teilklasse der 1-planaren Graphen berechnet. Diese Teilklasse schließt alle Einbettungen aus, deren Zeichnung einen *Bulgari*- oder einen *Gucci*-Graph nach Hong et. al. enthalten. Bekos et. al. [5] stellten 2017 einen Linearzeitalgorithmus vor, der für jede 1-planare Zeichnung bzw. dessen 1-planare Einbettung eine 1-planare Zeichnung mit folgenden Eigenschaften erzeugt: Alle Kanten der Zeichnung bestehen aus höchstens zwei geraden Teilstecken und Kanten schneiden sich ausschließlich im rechten Winkel. Somit kann aus einer gegebenen 1-planaren Einbettung eines Graphen eine 1-planare Zeichnung in polynomieller Zeit berechnet werden.

2.5 Parametrisierbarkeit

Die hohe Komplexität legt nahe das MAX-CUT-Problem unter dem Aspekt der Parametrisierbarkeit zu betrachten. Ein Algorithmus heißt parametrisierbar mit Parameter k, wenn seine Laufzeit in eine berechenbaren Funktion, die vom Parameter k abhängt, und in ein Polynom, das von der Größe der Eingabe abhängt, zerlegt werden kann. Dabei muss der Parameter k in polynomieller Zeit aus der Eingabe berechnet werden können. Die folgenden Definitionen orientieren sich an Flum [10].

2.5.1 Definition (FPT-Algorithmus). Ein Algorithmus **A** mit Eingabe X heißt *parametrisierbar* mit Parameter k (*engl. fixed parameter tractable*, kurz: FPT), wenn es einen in polynomieller Zeit aus X berechenbaren Parameter k gibt, sodass bei Eingabe X der Algorithmus **A** eine Laufzeit von $\mathcal{O}(f(k) \cdot p(|X|))$ hat, wobei f eine berechenbare Funktion und p ein Polynom ist.

Des Weiteren sind Probleme, die sich durch einen parametrisierbaren Algorithmus lösen lassen, ebenfalls parametrisierbar.

2.5.2 Definition (FPT-Problem). Ein Entscheidungsproblem heißt *parametrisierbar* mit Parameter k (*engl. fixed parameter tractable*, kurz: FPT), wenn es einen FPT-Algorithmus mit Parameter k gibt, der es löst.

Die Klasse aller parametrisierbaren Entscheidungsprobleme heißt **FPT**.

Kapitel 3

Eigenschaften

In diesem Kapitel werden die Eigenschaften der planaren, 1-fast-planaren und k-fast-planaren Graphen untersucht. Es gilt folgender hierarchischer Zusammenhang zwischen den in Kapitel 2 definierten Graphklassen:

$$\text{planar} \subseteq \text{1-fast-planar} \subseteq k\text{-fast-planar} \subseteq \text{1-planar}$$

Im Anschluss werden in Abschnitt 3.4 die Eigenschaften von Schnitten in k-fast-planaren Graphen untersucht.

3.1 Planare Graphen

Folgende Eigenschaften der planaren Graphen wurden aus „Graphentheorie“ von R. Diestel [7] übernommen.

3.1.1 Satz (Whitney 1932 [7]). *Die planare Einbettung eines 3-zusammenhängenden planaren Graphen ist (bis auf Symmetrie) eindeutig.*

3.1.2 Satz (Eulersche Polyederformel für die Ebene [7]). *Für jede planare Zeichnung eines zusammmenhängenden, planaren Graphen G mit $n \geq 1$ Knoten, m Kanten und l Flächen gilt: $n - m + l = 2$*

3.1.3 Lemma (zur Eulersche Polyederformel für die Ebene [7]). *Jeder planare Graph mit $n \geq 3$ Knoten hat höchstens $3n - 6$ Kanten.*

3.1.4 Satz (Kuratowski 1930, Wagner 1937 [7, 22, 32, 33]). *Ein Graph G ist genau dann planar, wenn G weder K_5 noch $K_{3,3}$ als topologischen Minor enthält.*

In Abbildung 3.1 sind die nicht-planaren Graphen K_5 und $K_{3,3}$ abgebildet.

(a) K_5 **(b)** $K_{3,3}$

Abbildung 3.1: Die Clique K_5 und der vollständige bipartite Graph $K_{3,3}$ sind nicht planar.

3.2 1-fast-planare Graphen

Da in diesem Abschnitt ausschließlich 1-fast-planare Graphen betrachtet werden, wird – insbesondere für die Beweise – Folgendes als Konvention festgelegt:

> Sei G ein echt 1-fast-planarer Graph mit 1-fast-planarer Zeichnung Γ und Kantenkreuzungsmenge $\mathrm{I}(\Gamma) = \{\{e_1, e_2\}\}$. Die sich kreuzenden Kanten seien $e_1 = ac$ und $e_2 = bd$.

Zur vereinfachten Darstellung sei die in G enthaltene Kreuzung o. B. d. A. von der Form wie in Abbildung 3.2a zu sehen. Die Kanten zwischen den Eckknoten der Kreuzung – außer e_1 und e_2 – dürfen ggf. überlappende Kantenzüge sein (s. Abb. 3.2b). Warum diese Annahme getroffen werden kann, wird in Satz 3.2.17 und Korollar 3.2.18 gezeigt. Bevor diese jedoch in Abschnitt 3.2.4 bewiesen werden können, werden zunächst die Eigenschaften des K_5 und des $K_{3,3}$ in Abschnitt 3.2.2 und 3.2.3 betrachtet. Davor wird in Abschnitt 3.2.1 untersucht wie die Kreuzung aus einem 1-fast-planaren Graphen entfernt werden kann.

3.2.1 Entfernen einer Kreuzung

In diesem Abschnitt werden sechs Möglichkeiten zum Entfernen der Kreuzung aus einem 1-fast-planaren Graphen vorgestellt. Dabei kann die Kreuzung entweder durch das Entfernen einer Kreuzungskante (vgl. Abb. 3.2c) oder durch die Kontraktion von zwei nebeneinanderliegenden Eckknoten (vgl. Abb. 3.2d) planarisiert werden. Dies wird in den folgenden Lemmata bewiesen.

3.2.1 Lemma. *Die Graphen $G - e_1$ und $G - e_2$ sind planar, wenn G ein 1-fast-planarer Graph mit 1-fast-planarer Zeichnung Γ und $\mathrm{I}(\Gamma) = \{\{e_1, e_2\}\}$ ist.*

Beweis. In den Zeichnungen $\Gamma - e_1$ und $\Gamma - e_2$ wurde jeweils eine der beiden sich kreuzenden Kanten gelöscht. Daher wurde die einzige Kreuzung in Γ eliminiert und die resultierenden Zeichnungen sind planar. Nach Definition 2.1.26 sind die Graphen $G - e_1$ und $G - e_2$ demnach planar. □

3.2.2 Lemma. *Die Graphen G/ab, G/ad, G/bc und G/cd sind planar, wenn G ein 1-fast-planarer Graph mit 1-fast-planarer Zeichnung Γ und $\mathrm{I}(\Gamma) = \{\{ac, bd\}\}$ ist.*

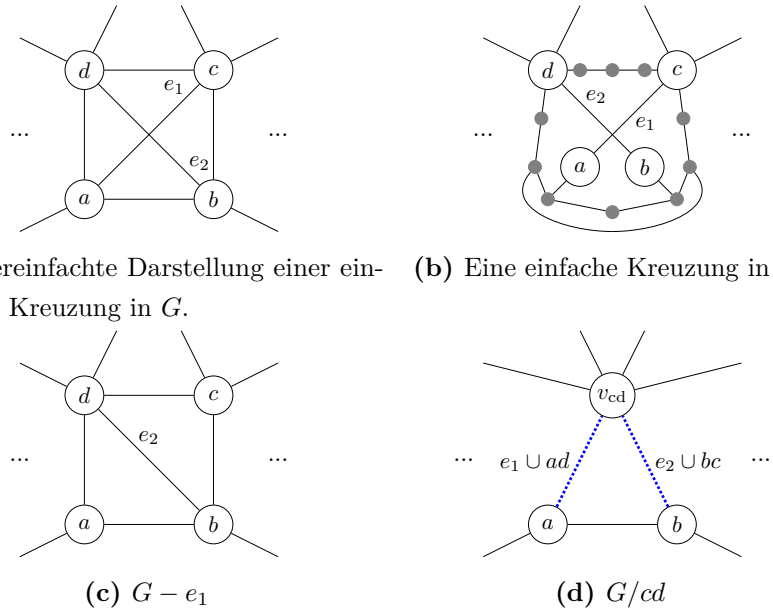
Beweis. In den Graphen G/ab, G/ad, G/bc und G/cd wurden jeweils zwei nebeneinanderliegende Eckknoten der einzigen in G enthaltenen Kreuzung verschmolzen. Die zwei zu verschmelzenden Knoten waren jeweils Endknoten einer der beiden sich kreuzenden Kanten $e_1 = ac$ bzw. $e_2 = bd$. Daher ist der neue verschmolzene Knoten Endknoten beider Kanten e_1 und e_2. Da die zwei Kanten nun einen gemeinsamen Endknoten haben, können sie so gezeichnet werden, dass sie sich nicht mehr kreuzen. Somit wurde die einzige Kreuzung in G entfernt und die resultierenden Graphen sind planar. □

Die folgende Aussage folgt direkt aus der Definition einer Kantenlöschung (2.1.15). Sie gilt insbesondere für $e \in \{e_1, e_2\}$.

3.2.3 Korollar. *Die Summe aller Kantengewichte in einem 1-fast-planaren Graphen G ist um $c_G(e)$ größer als die Summe aller Kantengewichte in $G - e$ mit $e \in E_G$.*

3.2.4 Lemma. *Die Summe aller Kantengewichte in einem 1-fast-planaren Graphen G ist um höchstens $c_G(xy)$ größer als die Summe aller Kantengewichte des planaren Graphen G/xy mit $xy \in \{ab, ad, bc, cd\}$.*

Beweis. G/ab, G/ad, G/bc und G/cd entstehen aus G, indem zwei nebeneinanderliegende Eckknoten verschmolzen werden. Alle so entstandenen Doppelkanten werden zu einer Kante verschmolzen (Def. 2.1.13) und die Gewichte werden addiert. Somit ändert sich die



(a) Vereinfachte Darstellung einer einfachen Kreuzung in G.

(b) Eine einfache Kreuzung in G.

(c) $G - e_1$

(d) G/cd

Abbildung 3.2: Zwei einfache Kreuzungen in G und zwei Möglichkeiten diese zu entfernen (blaue, gepunktete Kanten sind zusammengelegte Kanten).

Summe der Kantengewichte nur, wenn es in G eine direkte Kante zwischen den beiden verschmolzenen Knoten gibt ($xy \in E_G$). Diese verschwindet nach der Knotenkontraktion ersatzlos. Daher reduziert sich der Summe der Kantengewichte um höchstens c_{xy}. Falls xy nicht in E_G liegt, gilt: $c_{xy} = 0$. □

3.2.2 Eigenschaften des K_5

In diesem Abschnitt werden die Eigenschaften des K_5 genauer betrachtet.

3.2.5 Lemma. *Der Graph K_5 ist echt 1-fast-planar.*

Beweis. Gegeben sei der Graph $K_5 = (V, E)$ mit $V = \{a, b, c, d, e\}$ und $E = \{ab, ac, ad, ae, bc, bd, be, cd, ce, de\}$ aus Abbildung 3.3a. Wähle folgende Zeichnung des K_5 (s. Abb. 3.3b):

$$\begin{aligned}\Gamma_{K_5} : a \mapsto (2,4), b \mapsto (1,2), c \mapsto (0,0), d \mapsto (4,0), e \mapsto (3,2),\\ ab \mapsto ((2,4),(1,2)), ac \mapsto ((2,4),(0,4),(0,0)),\\ ad \mapsto ((2,4),(4,4),(4,0)), ae \mapsto ((2,4),(3,2)),\\ bc \mapsto ((1,2),(0,0)), bd \mapsto ((1,2),(4,0)), be \mapsto ((1,2),(3,2)),\\ cd \mapsto ((0,0),(4,0)), ce \mapsto ((0,0),(3,2)), de \mapsto ((4,0),(3,2))\end{aligned}$$

Da sich nur die Kanten-Polygonzüge $P_{ce} = ((0,0),(3,2))$ und $P_{bd} = ((1,2),(4,0))$ im Punkt $(2, \frac{4}{3})$ schneiden (vgl. Abb. 3.3b), ist Γ_{K_5} eine 1-fast-planare Zeichnung und K_5 ist demnach 1-fast-planar. Laut Satz 3.1.4 ist K_5 nicht planar, also ist K_5 nach Korollar 2.1.37 echt 1-fast-planar. □

Aus Lemma 3.2.5 und Definition 2.1.21 ergibt sich folgendes Korollar.

3.2.6 Korollar. *Jede Unterteilung des K_5 ist echt 1-fast-planar.*

3.2.7 Lemma. *Die kombinatorische 1-fast-planare Einbettung Π_{K_5} des K_5 ist (bis auf Symmetrie) eindeutig.*

$$\Pi_{K_5} : a \mapsto (b,e,d,c),\ b \mapsto (a,c,d,e),\ c \mapsto (a,d,e,b),\ d \mapsto (a,e,b,c),\ e \mapsto (a,b,c,d)$$

Beweis. Da der K_5 ein vollständiger Graph mit fünf Knoten ist, können vier beliebige Knoten ausgewählt werden, um eine Kreuzung in einer Zeichnung des K_5 zu erzeugen. Diese vier Knoten bilden einen vollständigen Graphen mit vier Knoten. Die Kanten werden so angeordnet, dass genau eine Kreuzung entsteht (vgl. b, c, d, e in Abb. 3.3). Da diese vier Knoten alle an die aktuelle Außenfläche angrenzen, können sie alle durch eine Kante mit dem fünften Knoten (a) verbunden werden ohne eine weitere Kantenkreuzung zu erzeugen. Da die Knoten zum Erzeugen dieser 1-fast-planaren Zeichnung beliebig gewählt wurden, ist dies – bis auf Symmetrie und konkrete Koordinatenwahl – die einzige Möglichkeit den K_5 einzubetten. □

Aus Definition 2.1.22 und Abbildung 3.3b ergibt sich folgendes Korollar.

3.2.8 Korollar. *Der K_4 ist ein topologischer Minor des K_5.*

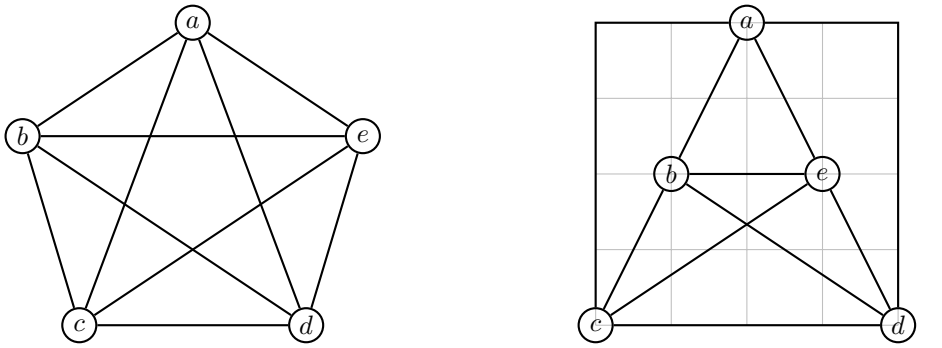
3.2.9 Lemma. *In einer 1-fast-planaren Zeichnung des K_5 können höchstens zwei Eckknoten der Kreuzung an dieselbe Fläche angrenzen. Diese Eckknoten müssen nebeneinanderliegend sein.*

Beweis. Zwei gegenüberliegende Eckknoten (z. B. b, d in Abb. 3.3b) in einer Zeichnung des K_5 können nicht an dieselbe Fläche angrenzen, da sie auf den inneren Flächen durch die andere Kreuzungskante (ce) und auf den äußeren Flächen durch die Kanten (ac, ae) zwischen den anderen beiden Eckknoten (c, e) und dem fünften Knoten (a) getrennt werden. Insbesondere können auch keine drei Eckknoten der Kreuzung an eine Fläche angrenzen, da drei Eckknoten immer zwei gegenüberliegende Eckknoten beinhalten. Daher bleibt nur noch die Möglichkeit, dass zwei nebeneinanderliegende Eckknoten an dieselbe Fläche angrenzen. Dies ist in Abbildung 3.3b zu sehen. □

Aus Lemma 3.2.9 und Definition 2.1.21 ergibt sich folgendes Lemma.

3.2.10 Lemma. *In einer 1-fast-planaren Zeichnung einer Unterteilung des K_5 können höchstens zwei Eckknoten der Kreuzung an dieselbe Fläche angrenzen. Diese Eckknoten müssen nebeneinanderliegend sein.*

Beweis. Sei U eine Unterteilung des K_5. Da jede Kante im K_5 einem Weg in U entspricht, gibt es zu jeder 1-fast-planaren Zeichnung von U eine 1-fast-planare Zeichnung des K_5 mit identischen Flächen, wobei die Begrenzung einer Fläche in Γ_U gegebenenfalls mehr Knoten enthält als die Begrenzung der selben Fläche in Γ_{K_5}. Daher folgt die Aussage analog zum Beweis von Lemma 3.2.9 und mit Lemma 3.3.7. □



(a) 1-fast-planarer Graph K_5 **(b)** 1-fast-planare Zeichnung Γ_{K_5}

Abbildung 3.3: Der 1-fast-planare Graph K_5 und seine Zeichnung Γ_{K_5} mit $\Gamma_{K_5} : a \mapsto (2, 4)$, $b \mapsto (1, 2), c \mapsto (0, 0), d \mapsto (4, 0), e \mapsto (3, 2)$.

3.2.3 Eigenschaften des $K_{3,3}$

In diesem Abschnitt werden die Eigenschaften des $K_{3,3}$ genauer betrachtet.

3.2.11 Lemma. *Der Graph $K_{3,3}$ ist echt 1-fast-planar.*

Beweis. Gegeben sei der Graph $K_{3,3} = (V, E)$ mit $V = \{a, b, c, d, e, f\}$ und $E = \{ad, ae, af, bd, be, bf, cd, ce, cf\}$ aus Abbildung 3.4a. Wähle folgende Zeichnung des $K_{3,3}$ (s. Abb. 3.4b):

$$\begin{aligned}\Gamma_{K_{3,3}} : a \mapsto (1,2), b \mapsto (3,6), c \mapsto (5,2), d \mapsto (5,4), e \mapsto (3,0), f \mapsto (1,4),\\ ad \mapsto ((1,2),(5,4)), ae \mapsto ((1,2),(3,0)), af \mapsto ((1,2),(1,4)),\\ bd \mapsto ((3,6),(5,4)), be \mapsto ((3,6),(6,6),(6,0),(3,0)), bf \mapsto ((3,6),(1,4)),\\ cd \mapsto ((5,2),(5,4)), ce \mapsto ((5,2),(3,0)), cf \mapsto ((5,2),(1,4)),\end{aligned}$$

Da sich nur die Kanten-Polygonzüge $P_{ad} = ((1,2),(5,4))$ und $P_{cf} = ((5,2),(1,4))$ im Punkt $(3,3)$ schneiden (vgl. Abb. 3.4b), ist $\Gamma_{K_{3,3}}$ eine 1-fast-planare Zeichnung und $K_{3,3}$ ist demnach 1-fast-planar. Laut Satz 3.1.4 ist $K_{3,3}$ nicht planar, also ist $K_{3,3}$ nach Korollar 2.1.37 echt 1-fast-planar. □

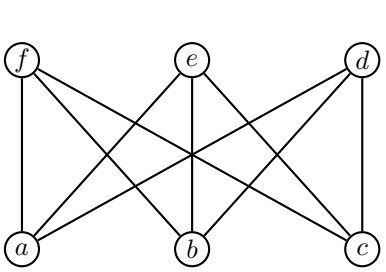
Aus Lemma 3.2.11 und Definition 2.1.21 ergibt sich folgendes Korollar.

3.2.12 Korollar. *Jede Unterteilung des $K_{3,3}$ ist echt 1-fast-planar.*

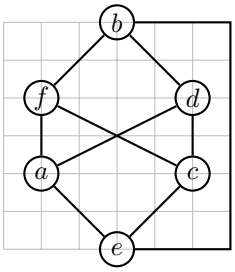
3.2.13 Lemma. *Die kombinatorische 1-fast-planare Einbettung $\Pi_{K_{3,3}}$ des $K_{3,3}$ ist (bis auf Symmetrie) eindeutig.*

$$\begin{aligned}\Pi_{K_{3,3}} : a \mapsto (d,f,e),\ b \mapsto (d,e,f),\ c \mapsto (d,f,e),\ d \mapsto (a,c,b),\ e \mapsto (a,b,c),\\ f \mapsto (a,c,b)\end{aligned}$$

Beweis. Um genau eine Kreuzung in einer Zeichnung des $K_{3,3}$ zu erzeugen, müssen jeweils zwei der vier Eckknoten aus einer der beiden Knotenmengen (V_1 und V_2) des bipartiten



(a) 1-fast-planarer Graph $K_{3,3}$



(b) 1-fast-planare Zeichnung $\Gamma_{K_{3,3}}$

Abbildung 3.4: Der 1-fast-planare Graph $K_{3,3}$ und seine Zeichnung $\Gamma_{K_{3,3}}$ mit $\Gamma_{K_{3,3}} : a \mapsto (1,2)$, $b \mapsto (3,6), c \mapsto (5,2), d \mapsto (5,4), e \mapsto (3,0), f \mapsto (1,4)$.

Graphen $K_{3,3}$ gewählt werden (z. B. $a, c \in V_1$ und $f, d \in V_2$ in Abb. 3.4). Die vier Knoten bilden den bipartiten Graphen $K_{2,2}$ mit einer Kreuzung. Die verbleibenden zwei Knoten und ihre ausgehenden Kanten müssen so angeordnet werden, dass keine weitere Kreuzung entsteht. Der dritte Knoten aus den beiden Knotenmengen ($b \in V_1$ bzw. $e \in V_2$) muss demnach zwischen den beiden Knoten aus der jeweils anderen Knotenmenge ($f, d \in V_2$ bzw. $a, c \in V_1$) angeordnet werden. Die Kante zwischen diesen beiden dritten Knoten (be) kann außen herum gelegt werden, da die beiden Knoten beide die Außenfläche berühren. Da die Knoten zum Erzeugen dieser 1-fast-planaren Zeichnung beliebig gewählt wurden, ist dies – bis auf Symmetrie und konkrete Koordinatenwahl – die einzige Möglichkeit den $K_{3,3}$ einzubetten. □

Aus Definition 2.1.22 und Abbildung 3.4b ergibt sich folgendes Korollar.

3.2.14 Korollar. *Der K_4 ist ein topologischer Minor des $K_{3,3}$.*

3.2.15 Lemma. *In einer 1-fast-planaren Zeichnung des $K_{3,3}$ können höchstens zwei Eckknoten der Kreuzung an die selbe Fläche grenzen. Diese Eckknoten müssen nebeneinanderliegend sein.*

Beweis. Seien V_1 und V_2 die beiden Knotenmengen des bipartiten Graphen $K_{3,3}$. Da die Kanten im $K_{3,3}$ nur zwischen diesen beiden Knotenmengen verlaufen und zwei gegenüberliegende Eckknoten der Kreuzung in einer Zeichnung des $K_{3,3}$ durch eine Kante verbunden sind, gilt: Einer der beiden Eckknoten liegt in V_1 und der andere liegt in V_2. Also sind jeweils zwei der vier Eckknoten aus einer der beiden Knotenmengen. Im Folgenden wird gezeigt, dass zwei gegenüberliegende Eckknoten nicht an die selbe Fläche angrenzen können. Dazu werden zwei gegenüberliegende Eckknoten der Kreuzung in einer Zeichnung des $K_{3,3}$ betrachtet. Zur Veranschaulichung werden die Überlegungen auf die 1-fast-planare Zeichnung des $K_{3,3}$ in Abbildung 3.4b übertragen. Seien a und d die betrachteten Knoten aus Abbildung 3.4b. Auf den inneren Flächen werden die beiden betrachteten Eckknoten durch die andere Kreuzungskante getrennt. Diese entspricht der Kante cf in Abbildung 3.4b. Auf den äußeren Flächen trennen sie die Kante zwischen den beiden Knoten des $K_{3,3}$, die keine Eckknoten der Kreuzung sind, – be in Abbildung 3.4b – und die vier Kanten zwischen jeweils einem dieser beiden Nichteckknoten und den zwei Eckknoten, die jeweils in der anderen Knotenmenge des $K_{3,3}$ liegen – bd, bf und ec, ea in Abbildung 3.4b. Daher können zwei gegenüberliegende Eckknoten nicht an die selbe Fläche angrenzen. Insbesondere können auch keine drei Eckknoten der Kreuzung an eine Fläche angrenzen, da drei Eckknoten immer zwei gegenüberliegende Eckknoten beinhalten. Es bleibt nur noch die Möglichkeit, dass zwei nebeneinanderliegende Eckknoten an die selbe Fläche angrenzen. Dies ist in Abbildung 3.4b zu sehen. □

Aus Lemma 3.2.15 und Definition 2.1.21 ergibt sich folgendes Lemma.

3.2.16 Lemma. *In einer 1-fast-planaren Zeichnung einer Unterteilung des $K_{3,3}$ können höchstens zwei Eckknoten der Kreuzung an die selbe Fläche angrenzen. Diese Eckknoten müssen nebeneinanderliegend sein.*

Beweis. Sei U eine Unterteilung des $K_{3,3}$. Da jede Kante im $K_{3,3}$ einem Weg in U entspricht, gibt es zu jeder 1-fast-planaren Zeichnung von U eine 1-fast-planare Zeichnung des $K_{3,3}$ mit identischen Flächen, wobei die Begrenzung einer Fläche in Γ_U gegebenenfalls mehr Knoten enthält als die Begrenzung der selben Fläche in $\Gamma_{K_{3,3}}$. Daher folgt die Aussage analog zum Beweis von Lemma 3.2.15 und mit Lemma 3.3.7. □

3.2.4 Anwendung vom Satz von Kuratowski und Wagner

Mit dem Satz von Kuratowski und Wagner (3.1.4) und den Eigenschaften des K_5 und des $K_{3,3}$ kann nun folgende Aussage über 1-fast-planare Graphen bewiesen werden.

3.2.17 Satz. *Jeder echt 1-fast-planare Graph G enthält den K_5 oder den $K_{3,3}$ als topologischen Minor.*

Beweis. Sei G ein beliebiger echt 1-fast-planarer Graph. Da jede Zeichnung von G mindestens eine Kreuzung enthält, ist G nicht planar. Laut dem Satz von Kuratowski und Wagner (3.1.4) ist ein Graph genau dann nicht planar, wenn er den K_5 oder den $K_{3,3}$ als topologischen Minor enthält. Demnach enthält G den K_5 oder den $K_{3,3}$ als topologischen Minor. □

Folgendes Korollar kann leicht aus Satz 3.2.17 gefolgert werden, da K_4 ein topologischer Minor des K_5 und des $K_{3,3}$ ist (Korollare 3.2.8 und 3.2.14).

3.2.18 Korollar. *Jeder echt 1-fast-planare Graph G enthält den K_4 als topologischen Minor.*

3.3 k-fast-planare Graphen

Da in diesem Abschnitt ausschließlich k-fast-planare Graphen betrachtet werden, wird – insbesondere für die Beweise – Folgendes als Konvention festgelegt:

> Sei G ein echt k-fast-planarer Graph mit k-fast-planarer Zeichnung Γ und Kantenkreuzungsmenge $\mathrm{I}(\Gamma) = \{z_1, \ldots, z_k\}$, wobei $z_i = \{e_{i1}, e_{i2}\}$ für $i = 1, \ldots, k$ gilt. Die sich kreuzenden Kanten der Kreuzung z_i seien $e_{i1} = a_i c_i$ und $e_{i2} = b_i d_i$. Die vier Eckknoten der Kreuzung z_i sind somit a_i, b_i, c_i und d_i.

Dabei sind die Bezeichnungen für die vier Eckknoten a_i, b_i, c_i und d_i einer Kreuzung z_i symbolisch (als Zeiger) zu verstehen, d. h. auch wenn ein Knoten v Teil mehrerer Kreuzungen ist (z. B. $v = a_i = b_j$) und in einer anderen Kreuzung z_j verschmolzen wird (z. B. zu $v_{b_j c_j}$), dann wird für die Kreuzung z_i der Zeiger des betroffenen Knoten auf den entsprechenden neuen Knoten aktualisiert ($a_i = v_{b_j c_j}$).

3.3.1 Lemma. *Jeder k-fast-planare Graph mit m Kanten hat eine k-fast-planare Zeichnung Γ mit höchstens $\frac{m}{2}$ Kreuzungen.*

Beweis. Da jede Kante in Γ höchstens eine andere Kante schneidet, kann es höchstens halb so viele Kreuzungen geben wie Kanten. Wenn jede Kante genau eine andere Kante schneidet, gilt $k = \frac{m}{2}$. □

3.3.2 Korollar. *Jeder 1-planare Graph mit m Kanten hat eine 1-planare Zeichnung mit höchstens $\frac{m}{2}$ Kreuzungen.*

3.3.1 Entfernen einer Kreuzung

Wie im 1-fast-planaren Fall (Abschnitt 3.2.1) kann eine Kreuzung in einem k-fast-planaren Graphen auf sechs verschiedene Möglichkeiten entfernt werden: Entweder durch das Entfernen einer Kreuzungskante (vgl. Abb. 3.2c) oder durch die Kontraktion von zwei nebeneinanderliegenden Eckknoten (vgl. Abb. 3.2d). Dies wird in den folgenden Lemmata bewiesen.

3.3.3 Lemma. *Die Graphen $G - e_{i1}$ und $G - e_{i2}$ sind echt $(k-1)$-fast-planar für $i = 1, \ldots, k$, wenn G ein echt k-fast-planarer Graph mit k-fast-planarer Zeichnung Γ und $I(\Gamma) = \{z_1, \ldots, z_k\}$ mit $z_i = \{e_{i1}, e_{i2}\}$ ist.*

Beweis. Da jede Kante in Γ höchstens einmal gekreuzt wird (s. Def. 2.1.33), kann durch das Entfernen einer Kante höchstens eine Kreuzung entfernt werden. War die entfernte Kante eine Kreuzungskante, so enthält die entstandene Zeichnung eine Kantenkreuzung weniger. Also sind $\Gamma - e_{i1}$ und $\Gamma - e_{i2}$ $(k-1)$-fast-planar, da jeweils eine Kreuzungskante

der Kreuzung z_i entfernt wurde. Demnach sind $G - e_{i1}$ und $G - e_{i2}$ echt $(k-1)$-fast-planar, da Γ eine Zeichnung von G mit minimaler Kreuzungszahl war und durch das Entfernen der Kante nur eine Kreuzung eleminiert wurde. □

3.3.4 Lemma. *Die Graphen G/a_ib_i, G/a_id_i, G/b_ic_i und G/c_id_i sind $(k-1)$-fast-planar für $i = 1, \ldots, k$, wenn G ein echt k-fast-planarer Graph mit k-fast-planarer Zeichnung Γ und $\mathrm{I}(\Gamma) = \{z_1, \ldots, z_k\}$ mit $z_i = \{a_ic_i, b_id_i\}$ ist. Wenn die verschmolzenen Knoten nur Eckknoten von einer einzigen Kreuzung in Γ waren, so ist die Knotenkontraktion echt $(k-1)$-fast-planar.*

Beweis. In den Graphen G/a_ib_i, G/a_id_i, G/b_ic_i und G/c_id_i wurden jeweils zwei nebeneinanderliegende Eckknoten einer in G enthaltenen Kreuzung z_i verschmolzen. Die zwei zu verschmelzenden Knoten sind jeweils Endknoten einer der beiden sich kreuzenden Kanten $e_{i1} = a_ic_i$ und $e_{i2} = b_id_i$. Daher ist der neue verschmolzene Knoten Endknoten beider Kanten e_{i1} und e_{i2}. Da die zwei Kanten nun einen gemeinsamen Endknoten haben, können sie so gezeichnet werden, dass sie sich nicht mehr kreuzen. Somit wurde die Kreuzung z_i entfernt und der resultierende Graph ist $(k-1)$-fast-planar. Wenn die verschmolzenen Knoten nicht Eckknoten einer weiteren Kreuzung in $\mathrm{I}(\Gamma)$ waren, sind die Graphen echt $(k-1)$-fast-planar, da durch das Verschmelzen der Knoten nur eine einzige Kreuzung in G eliminiert wurde und Γ eine Zeichnung von G mit minimaler Kreuzungszahl war. □

3.3.2 Eigenschaften einer Kreuzung

In den Abbildungen A.1a und A.1c im Anhang sind die Beweis-Struktur-Graphen der folgenden beiden Aussagen dargestellt.

3.3.5 Lemma. *Jede Kreuzung eines k-fast-planaren Graphen G ist in einem echt 1-fast-planaren Teilgraphen von G enthalten, der den K_5 oder den $K_{3,3}$ als topologischen Minor enthält.*

Beweis. Seien G_i die Graphen, die nur noch die i-te Kreuzung z_i enthalten. Jede Kante in Γ wird höchstens einmal geschnitten und ist somit höchstens Teil einer Kreuzung. Daher kann durch die folgende Wahl von G_i die Kreuzung z_i nicht entfernt werden:

$$G_i = G - \bigcup_{j \neq i} e_{j1}$$

Zunächst wird per *Induktion nach k* gezeigt, dass die Graphen G_i echt 1-fast-planar sind.
IA: Für $k = 1$ enthält G nur eine Kreuzung und $G_1 = G$ ist echt 1-fast-planar.
IV: Für jeden echt $(k-1)$-fast-planaren Graphen G sind die $k-1$ Teilgraphen $G_1, \ldots, G_{k-1}$ (wie oben beschrieben) echt 1-fast-planar.
IS: Sei G ein echt k-fast-planarer Graph mit $\mathrm{I}(\Gamma_G) = \{z_1, \ldots, z_k\}$. Nach Lemma 3.3.3 sind die Graphen $H^j = G - e_{j1}$ für $j = 1, \ldots, k$ echt $(k-1)$-fast-planar. Nach IV hat jeder von ihnen $k-1$ Teilgraphen H_i^j $(i = 1, \ldots, k-1)$, die jeweils echt 1-fast-planar sind. Wähle $G_i = H_i^k$ für $i = 1, \ldots, k-1$ und $G_k = H_{k-1}^1$. Diese Teilgraphen von G sind somit ebenfalls echt 1-fast-planar und sie entsprechen den oben beschriebenen G_i.
Laut Satz 3.2.17 enthalten demnach alle G_i den K_5 oder den $K_{3,3}$ als topologischen Minor. Da der K_5 und der $K_{3,3}$ echt 1-fast-planar sind (Lemmata 3.2.5 und 3.2.11), sind die Kreuzungskanten der i-ten Kreuzung dieselben Kanten, die sich auch in der Unterteilung des K_5 bzw. des $K_{3,3}$ kreuzen. □

Folgendes Korollar kann leicht gefolgert werden, da der K_4 ein topologischer Minor des K_5 und des $K_{3,3}$ ist (Korollare 3.2.8 und 3.2.14).

3.3.6 Korollar. *Jede Kreuzung eines k-fast-planaren Graphen G ist in einem echt 1-fast-planaren Teilgraphen von G enthalten, der den K_4 als topologischen Minor enthält.*

Die folgenden Lemmata beziehen sich auf die Definition einer *Fläche* in einer k-fast-planaren Zeichnung (Abschnitt 2.1.5) und die Eckknoten eine Kreuzung (Def. 2.1.7).

3.3.7 Lemma. *Zwei nebeneinanderliegende Eckknoten einer einfachen Kreuzung in einer Zeichnung Γ liegen mindestens an einer gemeinsamen Fläche.*

Beweis. Da die vier Eckknoten der Kreuzung und der Schnittpunkt der zwei Kreuzungskanten jeweils durch eine Teilkante verbunden sind, grenzen jeweils zwei nebeneinanderliegende Eckknoten und der Schnittpunkt der zwei Kanten an dieselbe Fläche. □

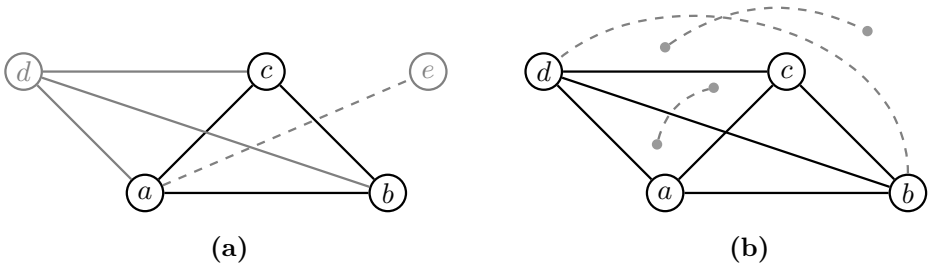


Abbildung 3.5: Beispiele zur Veranschaulichung des Beweises von Lemma 3.3.10

In Abbildung A.1b im Anhang ist der Beweis-Struktur-Graph des folgenden Lemmas dargestellt.

3.3.8 Lemma. *In einer k-fast-planaren Zeichnung Γ können höchstens zwei Eckknoten einer Kreuzung z an dieselbe Fläche angrenzen. Diese Eckknoten müssen nebeneinanderliegend sein.*

Beweis. Sei $z \in \mathrm{I}(\Gamma)$ eine beliebige Kreuzung in Γ. Nach Lemma 3.3.5 ist z in einem Teilgraphen von G enthalten, der den K_5 oder den $K_{3,3}$ als topologischen Minor enthält. Aus diesem Teilgraphen von G werden nun minimal viele Kanten und Knoten entfernt, sodass er eine Unterteilung des K_5 bzw. des $K_{3,3}$ wird. Durch das Entfernen von Kanten (und Knoten) aus Γ können Flächen höchstens zusammengelegt werden, aber nicht getrennt werden, d. h. die Anzahl an Eckknoten, die an dieselbe Fläche angrenzen, kann nicht kleiner werden. Nach Lemma 3.3.7 grenzen schon vor dem Entfernen der Kanten und Knoten jeweils zwei nebeneinanderliegende Eckknoten an eine gemeinsame Fläche. Mit Lemmata 3.2.10 und 3.2.16 folgt die Aussage. □

3.3.9 Lemma. *Zu jeder Kreuzung in einer k-fast-planaren Zeichnung gibt es vier angrenzende Flächen, die jeweils die Kreuzung selbst und zwei nebeneinanderliegende Eckknoten der Kreuzung beinhalten.*

Beweis. Wenn zwei Kanten sich schneiden, wird die Fläche in vier Teilflächen geteilt. Die Kreuzung kann nur an genau diese vier Teilflächen angrenzen. Da nach Korollar 3.3.6 jeweils zwei nebeneinanderliegende Eckknoten durch einen Weg verbunden sind und nach Lemma 3.3.7 jeweils zwei nebeneinanderliegende Eckknoten und die Kreuzung, zu der sie gehören, an eine gemeinsame Fläche angrenzen, sind die vier Flächen, die an die Kreuzung angrenzen, nicht miteinander verbunden. □

In Abbildung A.1d im Anhang ist der Beweis-Struktur-Graph des folgenden Lemmas dargestellt.

3.3.10 Lemma. *Zwei verschiedene Kreuzungen $z_1, z_2 \in I(\Gamma)$ eines echt k-fast-planaren Graphen G mit k-fast-planarer Zeichnung Γ können höchstens zwei Eckknoten gemein haben; diese müssen in beiden Kreuzungen nebeneinanderliegend sein.*

Beweis. Angenommen, es gäbe zwei verschiedene Kreuzungen $z_1, z_2 \in \mathrm{I}(\Gamma)$, die sich drei Eckknoten $a, b, c \in V_G$ teilen (s. Abb. 3.5a). Demnach gibt es drei mögliche Kreuzungskanten zur Auswahl: ab, ac, bc. Da jeweils zwischen zwei Eckknoten eine Schnittkante verläuft, muss von dem dritten Knoten (z.B. b) eine Kante durch die gewählte Kreuzungskante (ac) zum vierten Eckknoten der ersten Kreuzung (d) verlaufen. Dann gibt es jedoch keine Möglichkeit mehr, eine der anderen beiden Kanten (ab bzw. bc) von dem der Kante gegenüberliegenden Knoten aus (c bzw. a) zu schneiden, ohne die zuvor eingezogene Kante (bd) erneut zu schneiden. Daher können sich zwei verschiedene Kreuzungen höchstens zwei Eckknoten teilen.
Teilen zwei Kreuzungen $z_1, z_2 \in \mathrm{I}(\Gamma)$ sich zwei Eckknoten, so gibt es jeweils zwei Möglichkeiten pro Kreuzung, diese zu wählen: Entweder sind die Eckknoten gegenüberliegend oder nebeneinanderliegend. Zwei gegenüberliegende Eckknoten in z_1 schließen die Möglichkeit von zwei gegenüberliegenden Eckknoten in z_2 aus, da sonst eine Kante zweimal geschnitten oder eine Doppelkante eingezogen werden müsste (s. Abb. 3.5b); was jeweils Definition 2.1.33 bzw. 2.1.1 widerspricht. Wenn zwei Eckknoten nebeneinanderliegend sind, so grenzen sie nach Lemma 3.3.9 an eine gemeinsame Fläche. Da nach Lemma 3.3.8 gegenüberliegenden Eckknoten nicht an dieselbe Fläche angrenzen können, sind die Fälle, in denen die zwei Eckknoten in einer Kreuzung nebeneinanderliegend und in der anderen Kreuzung gegenüberliegend sind, unmöglich. Es verbleibt nur die Möglichkeit, dass die zwei Eckknoten in beiden Kreuzungen nebeneinanderliegend sind. □

Die folgende Definition folgt aus Lemma 3.3.10.

3.3.11 Definition. Zwei Kreuzungen $z_i, z_j \in \mathrm{I}(\Gamma)$ heißen *horizontal benachbart*, wenn sie sich die Knoten $b_i = a_j$ und $c_i = d_j$ teilen, und sie heißen *vertikal benachbart*, wenn sie sich die Knoten $c_i = b_j$ und $d_i = a_j$ teilen.

Das folgende Korollar folgt aus Lemma 3.3.5 und der Definition einer *Fläche* in einer k-fast-planaren Zeichnung.

3.3.12 Korollar. *Jede Fläche einer k-fast-planaren Zeichnung hat höchstens eine angrenzende Kreuzung.*

3.3.3 Entfernen aller Kreuzungen

Enthält ein Graph bzw. dessen Zeichnung mehrere einfache Kreuzungen, so können diese durch die im vorherigen Abschnitt vorgestellten Operationen sukzessive entfernt werden, um einen planaren Graphen zu erhalten. Um diese Graphen benennen zu können, wird die folgende Notation eingeführt.

3.3.13 Definition. Sei G ein echt k-fast-planarer Graph wie global gegeben. Für eine feste Sortierung der Kreuzungen $(z_1, \ldots, z_k)$ kodiert der String $i_1 \ldots i_k$, welche Kreuzung bereits auf welche Art im Graphen $G_{i_1 \ldots i_k}$ entfernt wurde. Dabei gilt für alle $j = 1, \ldots, k$: $i_j \in \{0, 1, 2, 3, 4, 5, 6\}$. Die Bedeutung des j-ten Zeichens ist wie folgt:

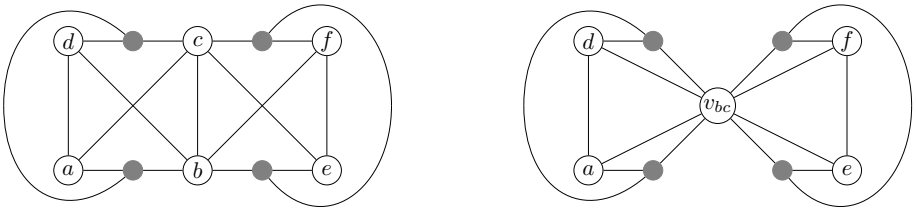
$$i_j = \begin{cases} 0 & \text{, Kreuzung } z_j \text{ existiert noch} \\ 1 & \text{, } e_{j1} \text{ wurde entfernt} \\ 2 & \text{, } e_{j2} \text{ wurde entfernt} \\ 3 & \text{, } c_j \text{ und } d_j \text{ wurden verschmolzen} \\ 4 & \text{, } b_j \text{ und } c_j \text{ wurden verschmolzen} \\ 5 & \text{, } a_j \text{ und } b_j \text{ wurden verschmolzen} \\ 6 & \text{, } a_j \text{ und } d_j \text{ wurden verschmolzen} \end{cases}$$

Aus den Lemmata 3.3.3 und 3.3.4 folgt: Wenn $i_j \neq 0$ gilt, so wurde die Kreuzung z_j in $G_{i_1 \ldots i_k}$ entfernt. Im Folgenden sind ein paar Beispiele für verschiedene Graphen $G_{i_1 \ldots i_k}$ angegeben. Dabei kodiert i^l einen String, der aus l Wiederholungen des Zeichens $i \in \{0, 1, 2, 3, 4, 5, 6\}$ besteht.

3.3.14 Beispiel (zu Definition 3.3.13).

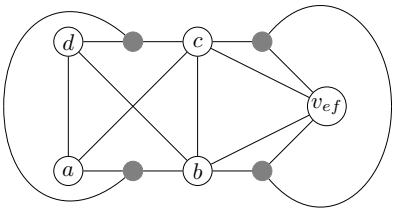
$G_{0^k} = G$	
$G_{1^k} = G - \bigcup_{j=1,\ldots k} e_{j1}$	$G_{10^{k-1}} = G - e_{11}$
$G_{2^k} = G - \bigcup_{j=1,\ldots k} e_{j2}$	$G_{20^{k-1}} = G - e_{12}$
$G_{32^{k-1}} = (G/c_1d_1) - \bigcup_{j=2,\ldots k} e_{j2}$	$G_{30^{k-1}} = G/c_1d_1$
$G_{340^{k-2}} = (G/c_1d_1)/b_2c_2$	$G_{40^{k-1}} = G/b_1c_1$
$G_{342^{k-2}} = ((G/c_1d_1)/b_2c_2) - \bigcup_{j=3,\ldots k} e_{j2}$	$G_{0^{i-1}40^{k-i}} = G/b_ic_i$

Wenn zwei Kreuzungen sich zwei Knoten teilen, kann es passieren, dass nachdem die erste Kreuzung durch Knotenkontraktion entfernt wurde, die zweite auch verschwindet, weil die beiden kontrahierten Knoten Eckknoten beider Kreuzungen waren (vgl. Lemma 3.3.10). Wenn die Kreuzungen in G nur durch die Fälle $1 - 4$ aus Definition 3.3.13 *aktiv* aus G *entfernt* werden, geben die Fälle 5 und 6 an, dass eine Kreuzung passiv entfernt wurde. Im Fall $i_j = 5$ (bzw. $i_j = 6$) wurde die Kreuzung z_j *passiv entfernt*, indem eine vertikal (bzw. horitontal) benachbarte Kreuzung z_i, mit der sich z_j zwei Eckknoten teilt, entfernt wurde. Dies geschieht, wenn z_i durch den Fall 3 (bzw. 4) entfernt wird, d. h. die zwei Knoten c_id_i (bzw. b_ic_i) verschmolzen werden. Hierzu kann ein Beispiel betrachtet werden.

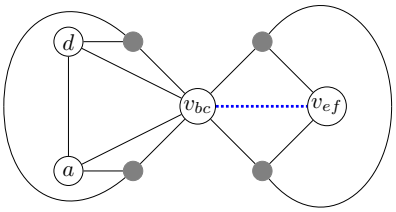


(a) $H_{00} = H$

(b) $H_{46} = H/bc$



(c) $H_{04} = H/ef$



(d) $H_{44} = (H/ef)/cd$

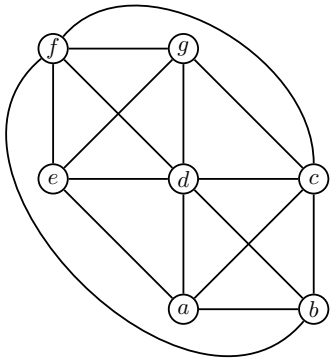
Abbildung 3.6: Zwei Reihenfolgen zum Entfernen beider Kreuzungen aus H. Abb. 3.6b zeigt das Resultat der Reihenfolge z_1, z_2 und Abb. 3.6d zeigt das Resultat der Reihenfolge z_2, z_1 (blaue, gepunktete Kanten sind zusammengelegte Kanten).

3.3.15 Beispiel. Betrachtet wird der Graph H aus Abbildung 3.6a mit den horizontal benachbarten Kreuzungen $z_1 = \{ac, bd\}$ und $z_2 = \{bf, ce\}$. Wird zuerst z_1 durch Knotenkontraktion von bc entfernt, so entsteht der Graph H_{46} (Abb. 3.6b). In diesem Fall impliziert die Art wie die erste Kreuzung z_1 entfernt wurde ($i_1 = 4$) die Art wie die zweite Kreuzung z_2 verschwindet ($i_2 = 6$). Die Kreuzung z_2 wird passiv entfernt. Die restlichen Fälle ($i_2 \in \{0, 1, 2, 3, 4, 5\}$ für $i_1 = 4$) können nicht mehr auftreten, was in dem Baum aus Abbildung 3.9a dazu führt, dass der entsprechende Teilbaum (mit $i_1 = 4$) kürzer ist als der Rest des Baums. Wird allerdings zuerst z_2 durch Knotenkontraktion von ef (Abb. 3.6c) und im Anschluss z_1 durch Knotenkontraktion von bc entfernt, so entsteht der Graph H_{44} (Abb. 3.6d). Die resultierenden Graphen H_{46} und H_{44} sind nicht identisch und hängen von der Reihenfolge ab, in der die Kreuzungen aus H entfernt werden. Die resultierenden Bäume für beide Reihenfolgen sind in Abbildung 3.9 abgebildet.

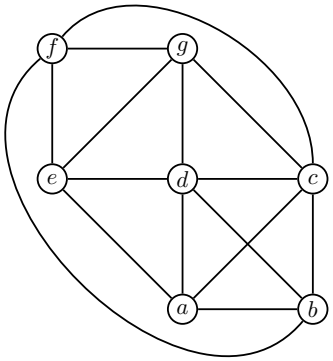
3.3.16 Korollar. *Wenn sich zwei Kreuzungen in* $\mathrm{I}(\Gamma)$ *zwei Eckknoten teilen, kann die Reihenfolge, in der die beiden Kreuzungen entfernt werden, die resultierenden Graphen verändern.*

3.3.17 Korollar. *Wenn sich zwei Kreuzungen in* $\mathrm{I}(\Gamma)$ *zwei Eckknoten teilen, werden, im Fall einer Kontraktion der beiden gemeinsamen Eckknoten, beide Kreuzungen gleichzeitig entfernt – eine Kreuzung aktiv und die andere Kreuzung passiv.*

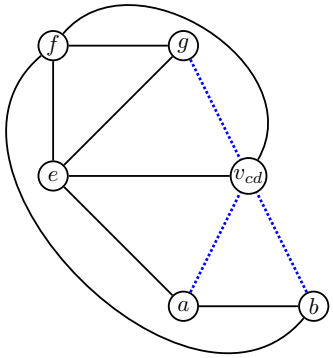
Es gibt einen weiteren Fall, in dem eine Kreuzung passiv entfernt werden kann. In diesem Fall müssen sich zwei Kreuzungen nur einen Eckknoten teilen. Wird dieser gemeinsame



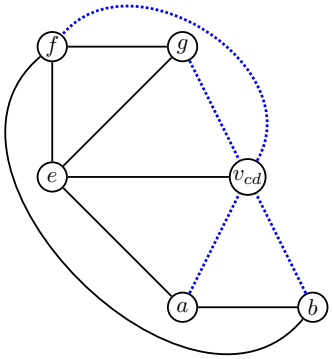
(a) $K_{00} = K$



(b) $K_{20} = K - df$



(c) $K_{23} = (K - df)/cd$



(d) $K_{*3} = K/cd$

Abbildung 3.7: Die aus K entstehenden Graphen für die Reihenfolge z_1, z_2 und z_2, z_1 (blaue, gepunktete Kanten sind zusammengelegte Kanten).

Eckknoten in der ersten Kreuzung mit einem anderen Knoten verschmolzen, sodass die Kreuzungskanten der zweiten Kreuzung so gezeichnet werden können, dass sie sich nicht mehr schneiden, wurde die zweite Kreuzung *passiv entfernt*. Dieser Fall ist allerdings schwieriger zu erkennen und lässt sich in der eingeführten Notation nicht abbilden. Auch hierzu kann ein Beispiel betrachtet werden.

3.3.18 Beispiel. Betrachtet wird der Graph K mit Zeichnung Γ_K aus Abbildung 3.7a und Kantenkreuzungsmenge $\mathrm{I}(\Gamma_K) = \{z_1, z_2\}$, wobei $z_1 = \{eg, df\}$ und $z_2 = \{ac, bd\}$ gilt. Wenn zuerst z_1 durch Löschung der Kante $e_{12} = df$ entfernt wird (Abb. 3.7b) und im Anschluss z_2 durch Knotenkontraktion von cd entfernt wird, so entsteht der Graph K_{23} (Abb. 3.7c). Wird hingegen zuerst z_2 durch Knotenkontraktion von cd entfernt, so werden die Kanten $e_{12} = df$ und cf zu einer Kante verschmolzen. Die neue Kante fv_{cd} kann nun (wie zuvor cf) so gezeichnet werden, dass sie $e_{11} = eg$ nicht mehr schneidet (Abb. 3.7d). Dadurch wurden in diesem Spezialfall durch eine Knotenkontraktion zwei Kreuzungen gleichzeitig planarisiert. Die Kreuzung z_1 wurde passiv entfernt. Da die Planarisierung von z_1 in keine der sechs vordefinierten Fälle passt, wird der entstehende Graph mit K_{*3}

bezeichnet (Abb. 3.7d). Die resultierenden Graphen K_{23} und K_{*3} unterscheiden sich im Gewicht der Kante fv_{cd},welches von der Reihenfolge abhängt, in der die Kreuzungen aus K entfernt werden.

3.3.19 Korollar. *Wenn sich zwei Kreuzungen in* $\mathrm{I}(\Gamma)$ *einen Eckknoten teilen und die Eckknoten der beiden Kreuzungen durch direkte Kanten verbunden sind, kann die Reihenfolge, in der die beiden Kreuzungen entfernt werden, die resultierenden Graphen verändern.*

3.3.20 Korollar. *Wenn sich zwei Kreuzungen in* $\mathrm{I}(\Gamma)$ *einen Eckknoten teilen und die Eckknoten der beiden Kreuzungen durch direkte Kanten verbunden sind, können beide Kreuzungen gleichzeitig entfernt werden – eine Kreuzung aktiv, durch Kontraktion des gemeinsamen Eckknotens mit einem anderen Eckknoten und die andere Kreuzung passiv, durch Neuzeichnen der Kreuzungskanten.*

Für den Fall, dass G keine zwei Kreuzungen enthält, die sich zwei Eckknoten teilen oder deren Eckknoten durch direkte Kanten verbunden sind, wenn sie sich einen Eckknoten teilen, wird nun bewiesen, dass die Reihenfolge, in der die Kreuzungen aus G entfernt werden, nichts an den Graphen $G_{i_1 \dots i_k}$ mit $i_j = 1, 2, 3, 4$ für $j = 1, \dots, k$ ändert.

3.3.21 Lemma. *Die Reihenfolge, in der die Kreuzungen* $z_1, \dots, z_k$ *aus* G *entfernt werden, ändert nichts an den Graphen* $G_{i_1 \dots i_k}$ *mit* $i_j = 1, 2, 3, 4$ *für* $j = 1, \dots, k$*, wenn* G *keine zwei Kreuzungen enthält, die sich zwei Eckknoten teilen oder deren Eckknoten durch direkte Kanten verbunden sind, wenn sie sich einen Eckknoten teilen.*

Beweis. Sei $z_{l_1}, z_{l_2}, \dots z_{l_k}$ eine beliebige Reihenfolge zur Entfernung der Kreuzungen aus G, wobei $l_1, l_2, \dots l_k$ eine Permutation von $1, 2, \dots k$ ist. Im Folgenden werden die Auswirkungen auf die entstehenden Graphen $G_{i_1 \dots i_k}$ mit $i_j \in \{1, 2, 3, 4\}$ für $j = 1, \dots, k$ untersucht, wenn zwei beliebige Kreuzungen z_{l_p} und z_{l_q} mit $p < q$ in der Reihenfolge vertauscht werden. Durch das Entfernen der Kreuzungen $z_{l_{p+1}} \dots z_{l_{q-1}}$ werden o. B. d. A. die Kreuzungen z_{l_p} und z_{l_q} nicht verändert. Falls doch, so werden die $2q - 2p - 1$ Vertauschungen von direkten Nachbarn betrachtet, indem zuerst z_{l_p} jeweils 1 Position nach rechts bis an Position $q - 1$ und im Anschluss z_{l_q} jeweils 1 Position nach links bis an Position p getauscht wird.
Es gibt drei mögliche Fälle, wie die zwei Kreuzungen planarisiert werden können. Diese werden nun genauer betrachtet.
Fall 1: *Beide Kreuzungen werden durch Kantenlöschung planarisiert*
Für die Löschung von zwei Kanten aus G ist die Reihenfolge, in der diese gelöscht werden, irrelevant, da die Löschung einer Kante keine anderen Kanten oder Knoten verändert.
Fall 2: *Eine Kreuzung wird durch Kantenlöschung, die andere durch Knotenkontraktion planarisiert*
Die Knotenkontraktion entfernt höchstens die Kante zwischen den kontrahierten Knoten (Def. 2.1.13). Da nach Lemma 3.3.10 die Endknoten einer Schnittkante nicht Eckknoten

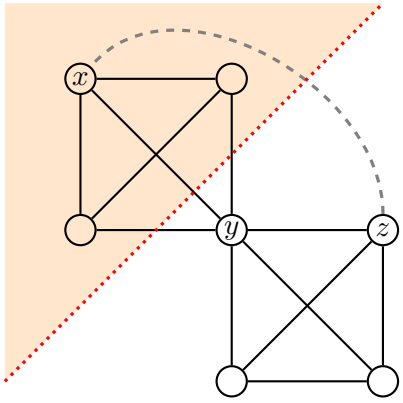


Abbildung 3.8: Beispiel zu Fall 2 im Beweis von Lemma 3.3.21

einer anderen Kreuzung sein können, kann die Kante zwischen den kontrahierten Knoten nicht die Schnittkante einer anderen Kreuzung sein. Die Kantenlöschung entfernt nur die Kante zwischen zwei gegenüberliegenden Eckknoten x und y einer Kreuzung. Falls zwei Kreuzungen sich jedoch einen Eckknoten y teilen und dieser in der Knotenkontraktion von y und z verschmolzen wurde, könnte nach der Knotenkontraktion eine Kante zwischen x und z entfernt werden (graue, gestrichelte Kante in Abb. 3.8). Diese kann jedoch nicht existieren, da nach Voraussetzung die Eckknoten zweier verschiedener Kreuzungen aus $\mathrm{I}(\Gamma)$ nicht durch direkte Kanten verbunden sind, wenn sie sich einen Eckknoten teilen.
Demnach können auch eine Kantenlöschung und eine Knotenkontraktion beliebig vertauscht werden.

Fall 3: *Beide Kreuzungen werden durch Knotenkontraktion planarisiert*
Laut Voraussetzung können sich zwei Kreuzungen höchstens einen Eckknoten teilen. Daher werden für die Knotenkontraktion zwei Fälle betrachtet:

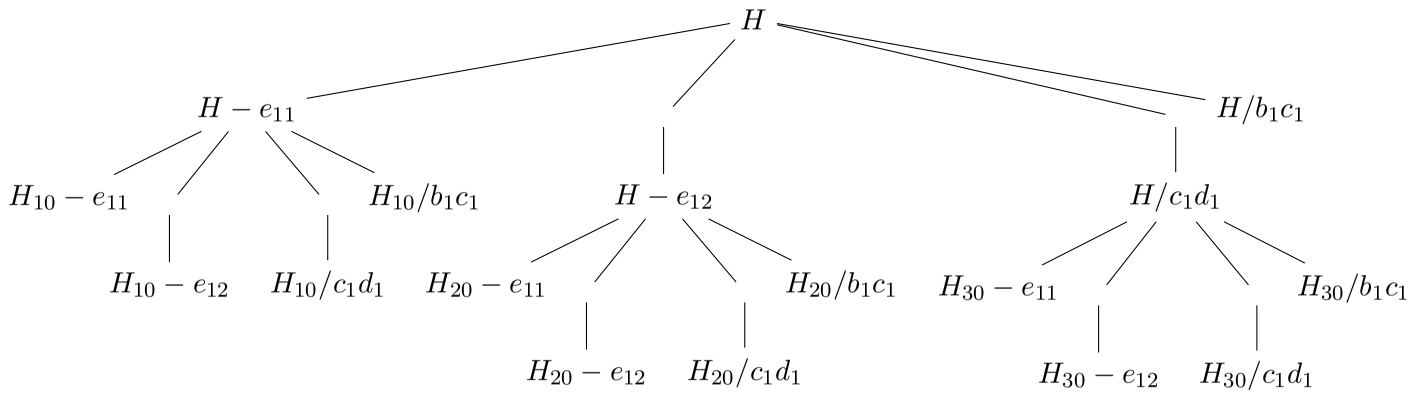
Fall 3a: *Die Eckknotenmengen sind disjunkt*
In diesem Fall, ist die Reihenfolge, in der die beiden Eckknotenpaaren kontrahiert werden, irrelevant, da die Kontraktion der zwei Eckknoten der einen Kreuzung keinen Einfluss auf die Eckknoten oder Kanten der anderen Kreuzungen hat.

Fall 3b: *Die zwei Kreuzungen teilen sich einen Eckknoten*
Falls der gemeinsame Eckknoten nicht kontrahiert wird, gilt die Argumentation von Fall 3a. Falls der gemeinsame Eckknoten Teil einer bzw. beider Knotenkontraktionsmengen ist, so wird durch die eine bzw. die erste Kontraktion zwar der Name des gemeinsamen Knoten geändert (er wird zum kontrahierten Knoten), aber die andere Kreuzung bleibt danach immer noch bestehen. Falls zwei Knotenkontraktion $b_{l_p}c_{l_p}$, $c_{l_q}d_{l_q}$ ausgeführt werden, die den gemeinsamen Knoten $c_{l_p} = c_{l_q}$ enthalten, so entspricht dies der Kontraktion der Menge $U = b_{l_p}c_{l_p}d_{l_q}$. Da am Ende alle drei Knoten zu einem verschmolzen sind, ist die Reihenfolge, in der sie verschmolzen werden, irrelevant. □

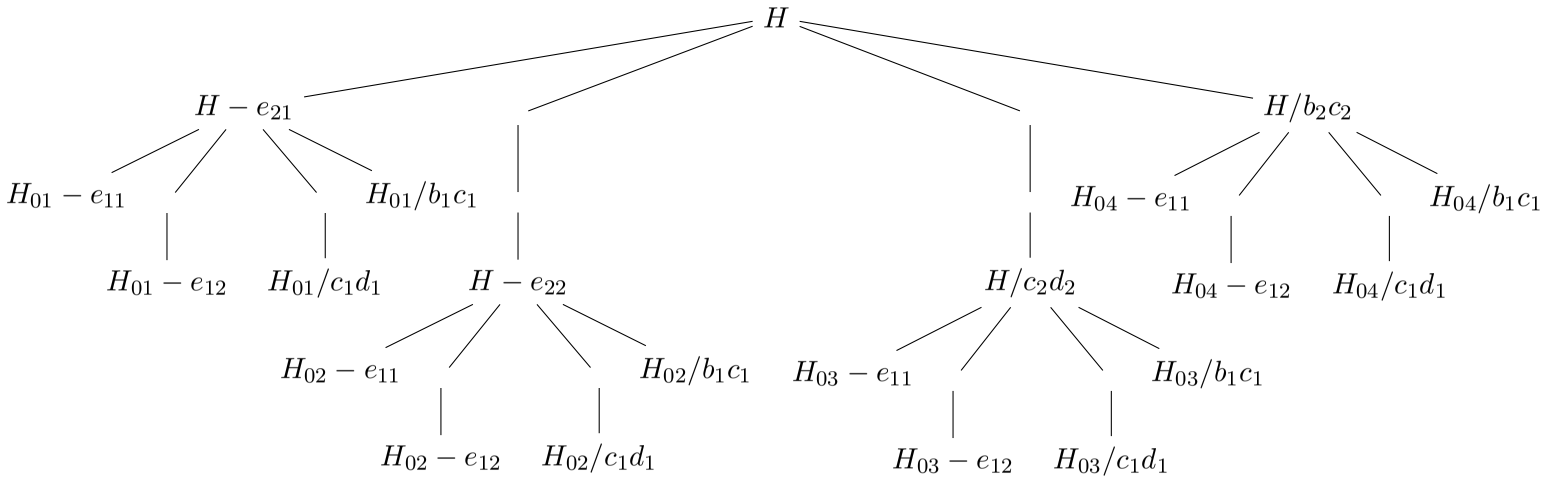
3.3.22 Lemma. *Die Graphen $G_{i_1 \dots i_k}$ mit $i_j = 1, 2, 3, 4, 5, 6$ für $j = 1, \dots, k$ sind – so sie existieren – planar. Wenn sich zwei Kreuzungen in G jeweils höchstens einen Eckknoten teilen und die Eckknoten zwei verschiedener Kreuzungen aus* $\mathrm{I}(\Gamma)$ *nicht durch direkte Kanten verbunden sind, sind die Graphen $G_{i_1 \dots i_k}$ mit $i_j = 1, 2, 3, 4$ für $j = 1, \dots, k$ eindeutig.*

Beweis. Die Planarität der Graphen folgt durch rekursive Anwendung der Lemmata 3.3.3 und 3.3.4, da in jedem Schritt mindestens eine Kreuzung entfernt wird. Die Eindeutigkeit der Graphen $G_{i_1 \dots i_k}$ mit $i_j = 1, 2, 3, 4$ für $j = 1, \dots, k$ folgt aus Lemma 3.3.21. Sie entsprechen den 4^k Blättern des Baums in Abbildung 3.10. □

In Abbildung 3.10 ist das sukzessive Entfernen der Kreuzungen aus einem k-fast-planaren Graphen G durch jeweils einen der vier Fälle $1 - 4$ aus Definition 3.3.13 graphisch in einer Baumstruktur mit Verzweigungsgrad 4 dargestellt. In der ersten Ebene ($\sigma = k - 1$) wurde die Kreuzung z_1 entfernt und die Graphen der Ebene sind $(k - 1)$-fast-planar (Lemmata 3.3.3 und 3.3.4). In der zweiten Ebene ($\sigma = k - 2$) wurden die Kreuzungen z_1 und z_2 entfernt und die Graphen der Ebene sind $(k - 2)$-fast-planar. In der untersten Ebene ($\sigma = 0$) wurden alle Kreuzungen entfernt. Der Baum hat Verzweigungsgrad 4 und Tiefe k. Die Blätter des Baums entsprechen den 4^k planarisierten Versionen von G: $G_{i_1 \dots i_k}$ mit $i_j = 1, 2, 3, 4$ für $j = 1, \dots, k$.



(a) Reihenfolge z_1, z_2



(b) Reihenfolge z_2, z_1

Abbildung 3.9: Zwei Reihenfolgen zum Entfernen der Kreuzungen aus H.

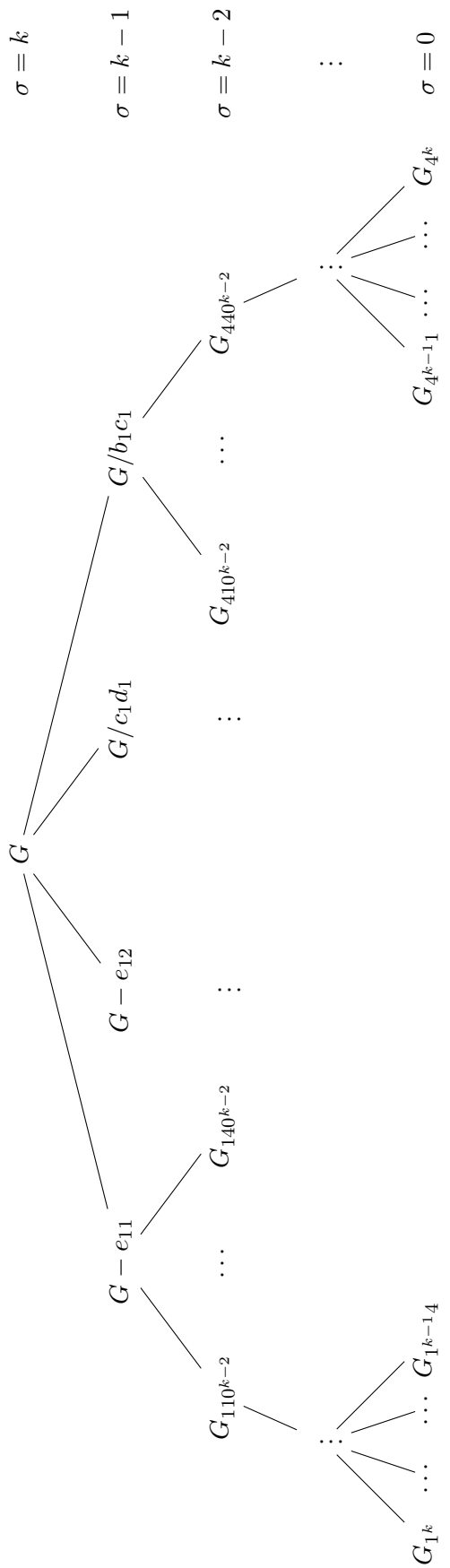


Abbildung 3.10: Das sukzessive Entfernen der k Kreuzungen aus G durch jeweils einen der vier Fälle $1-4$ aus Definition 3.3.13.

3.3.4 Anwendung der eulerschen Polyederformel für die Ebene

Mit dem Lemma zur eulerschen Polyederformel für die Ebene (3.1.3) kann nun eine obere Schranke für die Kantenanzahl in k-fast-planaren Graphen bewiesen werden. Mit Lemma 3.3.1 folgt eine Verallgemeinerung der Aussage auf 1-planare Graphen. In Abbildung A.1e im Anhang ist der Beweis-Struktur-Graph der beiden folgenden Lemmata dargestellt.

3.3.23 Lemma. *Jeder k-fast-planare Graph mit $n \geq 3$ Knoten hat höchstens $3n - 6 + k$ Kanten.*

Beweis. Nach Lemma 3.1.3 hat jeder planare Graph mit $n \geq 3$ Knoten höchstens $3n - 6$ Kanten. Da jeder k-fast-planare Graph G durch die Löschung jeweils einer Kreuzungskante aller k Kreuzungen in Γ_G^k planar wird (rekursive Anwendung von Lemma 3.3.3), enthält G höchstens $3n - 6 + k$ Kanten. □

3.3.24 Lemma. *Jeder 1-planare Graph mit $n \geq 3$ Knoten hat höchstens $6n - 12$ Kanten.*

Beweis. Wähle ein k, dass G k-fast-planar ist. Sei m die Anzahl an Kanten in G. Nach Lemma 3.3.23 gilt $m \leq 3n - 6 + k$. Mit Lemma 3.3.1 folgt $m \leq 3n - 6 + \frac{m}{2}$. Wird die Ungleichung mit 2 multipliziert und wird m subtrahiert, so ergibt sich die Aussage. □

Aus Lemma 3.3.1 und 3.3.24 folgt eine verallgemeinerte obere Schranke für die Anzahl an einfachen Kreuzungen in einem 1-planaren Graphen.

3.3.25 Korollar. *Jeder 1-planare Graph mit $n \geq 3$ Knoten hat eine 1-planare Zeichnung mit höchstens $3n - 6$ Kreuzungen.*

3.4 Schnitte in k-fast-planaren Graphen

In diesem Abschnitt werden zunächst alle Möglichkeiten für einen Schnitt durch eine Kreuzung betrachtet. Im Anschluss werden die Anforderungen an eine Schnittübertragung zwischen einer Kantenlöschung bzw. Knotenkontraktion und dem Originalgraphen betrachtet.

3.4.1 Mögliche Schnitte durch eine Kreuzung

Um alle Möglichkeiten zur Aufteilung der vier Eckknoten einer Kreuzung auf zwei Mengen zu berechnen, wird zunächst die Symmetrie von Knotenfärbungen auf Graphen definiert.

3.4.1 Definition (c-Knotenfärbung). Eine *c-Knotenfärbung* $f : V \to \{1, 2, \ldots, c\}$ weist jedem Knoten in V eine von c verschiedenen Farben zu.

3.4.2 Definition (Symmetrie). Für einen Graphen G heißen zwei 2-Knotenfärbungen *farbsymmetrisch*, wenn die eine Färbung durch Invertierung der Farben aus der anderen Färbung hervorgeht.

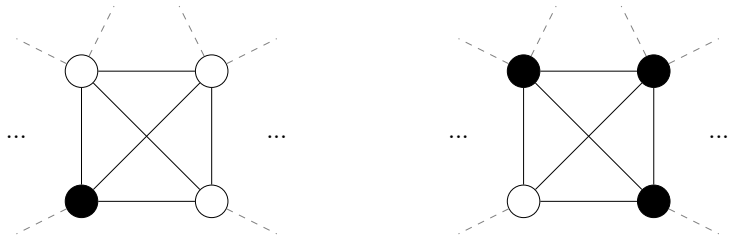


Abbildung 3.11: Ein Beispiel für zwei farbsymmetrische 2-Knotenfärbungen.

3.4.3 Lemma. *Es gibt 8 Möglichkeiten die vier Eckknoten einer Kreuzung durch einen Schnitt in zwei Mengen aufzuteilen.*

Beweis. Ein Schnitt teilt die Knotenmenge eines Graphen in zwei Teilmengen. Um darzustellen zu welcher Teilmenge ein Knoten gehört, wird jedem Knoten eine von zwei Farben zugewiesen, sodass die Endknoten einer Schnittkante unterschiedlich eingefärbt sind. Um die Anzahl der Möglichkeiten eines Schnitts durch eine Kreuzung zu berechnen, werden die möglichen Farbbelegungen für die vier Eckknoten abgezählt. Dafür wird 4 Mal – für jeden Knoten einmal – mit Zurücklegen aus einem Topf mit zwei Farben gezogen. Das entspricht $2^4 = 16$ Möglichkeiten für die Farbverteilung auf die vier Knoten. Diese 16 Fälle lassen sich in $8 \cdot 2$ farbsymmetrische Fälle aufteilen. Da nicht die spezifische Mengenzugehörigkeit relevant ist, sondern die Verbindung zweier Knoten mit unterschiedlichen Farben, werden die farbsymmetrischen Fälle nicht weiter betrachtet und es verbleiben 8 relevante Fälle. Diese lassen sich in drei Gruppen aufteilen:

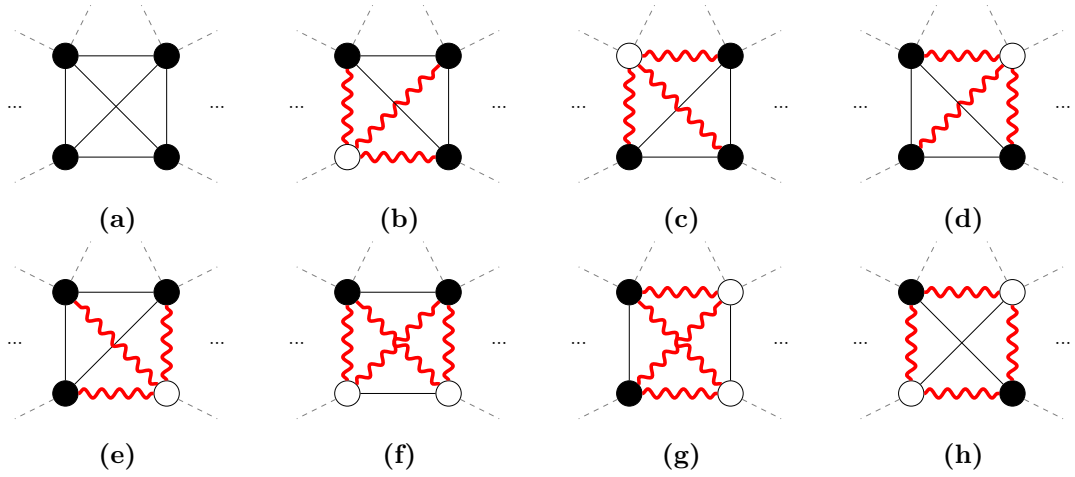


Abbildung 3.12: Die 8 Fälle zur Aufteilung der Eckknoten einer Kreuzung auf zwei durch einen Schnitt separierte Mengen. (Schnittkanten sind rot und geschwungen)

1. Alle Knoten in einer Menge und kein Knoten in der anderen Menge
2. Drei Knoten in einer Menge und ein Knoten in der anderen Menge
3. Zwei Knoten in einer Menge und zwei Knoten in der anderen Menge

Die erste Gruppe enthält 1, die zweite 4 und die dritte 3 der 8 Fälle. Somit decken diese drei Gruppen alle 8 Fälle ab. Die 8 Möglichkeiten die vier Eckknoten einer Kreuzung durch einen Schnitt in zwei Mengen aufzuteilen sind:

(a) Alle Knoten in einer Menge

(b) Drei Knoten in einer Menge ohne a

(c) Drei Knoten in einer Menge ohne d

(d) Drei Knoten in einer Menge ohne c

(e) Drei Knoten in einer Menge ohne b

(f) Jeweils zwei horizontal nebeneinanderliegende Eckknoten in einer Menge

(g) Jeweils zwei vertikal nebeneinanderliegende Eckknoten in einer Menge

(h) Jeweils zwei gegenüberliegende Eckknoten in einer Menge

In Abbildung 3.12 sind diese acht Fälle graphisch dargestellt. □

3.4.4 Lemma. *Wenn sich zwei Kreuzungen zwei Eckknoten teilen, dann gibt es zwei Möglichkeiten die beiden gemeinsamen Eckknoten durch einen Schnitt auf zwei Mengen aufzuteilen. Für jeden der beiden Fälle gibt es jeweils 16 Möglichkeiten die sechs Eckknoten beider Kreuzungen durch einen Schnitt in zwei Mengen aufzuteilen.*

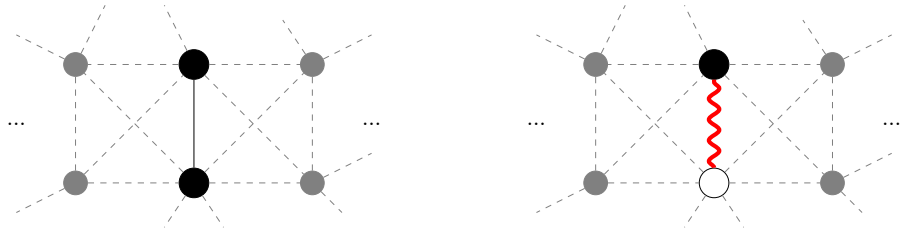


Abbildung 3.13: Die 2 Fälle zur Aufteilung der zwei gemeinsamen Eckknoten zweier Kreuzungen auf zwei durch einen Schnitt separierte Mengen. (Schnittkanten sind rot und geschwungen)

Beweis. Wenn sich zwei Kreuzungen zwei Eckknoten teilen, so gibt es für diese beiden Eckknoten folgende Möglichkeiten durch einen Schnitt auf zwei Mengen aufgeteilt zu werden: Entweder sind die beiden Eckknoten in derselben Menge (s. Abb. 3.13a) oder sie werden durch den Schnitt getrennt (s. Abb. 3.13b). Es wird o. B. d. A. angenommen, dass die Kreuzungen horizontal benachbart sind. Die folgenden Aussagen lassen sich leicht auf zwei vertikal benachbarte Kreuzungen übertragen, indem alle betrachteten (Teil-)Graphen um 90° gedreht werden. Nun wird untersucht, welche Fälle aus Lemma 3.4.3 für die beiden Kreuzungen kombiniert werden können. Die Farbbelegung kann vertauscht werden, damit die gemeinsamen Eckknoten in den kombinierten Fällen die gleichen Farben haben. Wenn die beiden gemeinsamen Eckknoten in derselben Menge sind, so sind nur die Fälle (a),(b),(c),(g) für die linke Kreuzung und (a),(d),(e),(g) für die rechte Kreuzung möglich. Werden die beiden gemeinsamen Eckknoten durch den Schnitt getrennt, so sind nur die Fälle (d),(e),(f),(h) für die linke Kreuzung und (b),(c),(f),(h) für die rechte Kreuzung möglich. Daraus ergeben sich jeweils $4 \cdot 4 = 16$ Möglichkeiten zur Aufteilung der sechs Eckknoten der beiden Kreuzungen auf zwei Mengen. □

3.4.2 Schnittübertragung

Die folgenden Aussagen nennen die Anforderungen an eine Schnittübertragung zwischen einer Kantenlöschung bzw. Knotenkontraktion und dem Originalgraphen. In Abbildung A.3c im Anhang sind die Zusammenhänge der Lemmata dieses Abschnitts graphisch dargestellt.

3.4.5 Lemma (Schnittübertrag von $G - e$ nach G).
Für die Graphen G und $G - e$ mit $e \in E_G$ gilt:
Wenn F ein Schnitt in $G - e$ ist, dann ist F ebenfalls ein Schnitt in G.

Beweis. Sei F ein Schnitt in $L = G - e$ mit $e \in E_G$. Nach Definition 2.1.15 gilt $V_L = V_G$ und $E_L = E_G \setminus \{e\}$. Sei $V_1 \subseteq V_L$ die Knotenmenge, die F definiert (Def. 2.2.1). Da $F \subseteq E_L \subseteq E_G$ und $V_1 \subseteq V_L = V_G$ gilt und F ein Schnitt in L ist, ist F ebenfalls ein Schnitt in G. □

3.4.6 Lemma (Schnittübertrag von G/xy nach G).
Für die Graphen G und $K = G/xy$ mit $x, y \in V_G$ gilt:

1. *Ist F ein Schnitt in K, dann ist $\kappa_{xy}^{-1}(F, E_G)$ ein Schnitt in G.*
2. *Es gilt $\chi(F) = \chi(\kappa_{xy}^{-1}(F, E_G))$ für alle Schnitte $F \subseteq E_K$ in K.*

Beweis. Sei G ein Graph mit $x, y \in V_G$ und $K = G/xy$.

1. Sei F ein Schnitt in K. Also existiert nach Definition 2.2.1 eine Knotenmenge, die diesen definiert: diese Knotenmenge sei V_1. Wird diese Knotenmenge durch Knotenexpansion von v_{xy} auf G übertragen, so entsteht die Menge $V' = \kappa_{xy}^{-1}(V_1, E_G)$ (s. Def. 2.1.14). Die Knotenmenge V' definiert den Schnitt $F' = \delta(V')$ in G. Diese Kantenmenge entsteht, indem F ebenfalls durch Knotenexpansion von v_{xy} auf G projiziert wird: $F' = \kappa_{xy}^{-1}(F, E_G)$. Laut Definition 2.2.1 ist F' ein Schnitt in G.

2. Sei $F \subseteq E_K$ ein Schnitt in K. Es gilt $\chi(F) = \sum_{e \in F} c_K(e)$. Die Gewichte in $F' = \kappa_{xy}^{-1}(F, E_G)$ entsprechen entweder einem Kantengewicht in F oder einem Anteil eines Kantengewichtes in F. Da alle Kanten in F auf eine oder mehrere Kanten in F' übertragen werden und die Gewichte sich dabei nicht verändern, sondern nur aufgeteilt werden, gilt:

$$\chi(F) = \sum_{e \in F} c_K(e) = \sum_{e' \in F'} c_G(e') = \chi(F') = \chi(\kappa_{xy}^{-1}(F, E_G))$$

□

3.4.7 Lemma (Schnittübertrag von G nach $G - e$).
Für die Graphen G und $G - e$ mit $e \in E_G$ gilt:
Wenn F ein Schnitt in G ist und $e \notin F$ gilt, dann ist F ebenfalls ein Schnitt in $G - e$.

Beweis. Sei $e \in E_G$, F ein Schnitt in G mit $e \notin F$ und $L = G - e$. Nach Definition 2.2.1 existiert eine Knotenmenge, die F definiert: diese Knotenmenge sei V_1. Da e nicht in F liegt und nach Definition 2.1.15 $V_L = V_G$ und $E_L = E_G \setminus \{e\}$ gilt, definiert V_1 ebenfalls einen Schnitt in L. Demnach ist $F = \delta(V_1)$ ebenfalls ein Schnitt in $G - e$. □

3.4.8 Lemma (Schnittübertrag von G nach G/xy).
Für die Graphen G und $K = G/xy$ mit $x, y \in V_G$ gilt:

1. *Ist F ein Schnitt in G und gilt $xy \notin F$, dann ist $\kappa_{xy}(F)$ ein Schnitt in K.*
2. *Es gilt $\chi(F) = \chi(\kappa_{xy}(F))$ für alle Schnitte $F \subseteq E_G \setminus \{xy\}$ in G.*

Beweis. Sei G ein Graph mit $x, y \in V_G$ und $K = G/xy$.

1. Sei F ein Schnitt in G mit $xy \notin F$. Also existiert nach Definition 2.2.1 eine Knotenmenge, die F definiert: diese Knotenmenge sei V_1. Wird diese Knotenmenge durch Knotenkontraktion von xy auf K übertragen, so entsteht die Menge $V' = \kappa_{xy}(V_1)$ (s. Def. 2.1.13). Da xy nicht in F liegt, gilt entweder $x, y \in V_1$ oder $x, y \notin V_1$. Dementsprechend liegt v_{xy} entweder in V' oder nicht. Die Knotenmenge V' definiert den Schnitt $F' = \delta(V')$ in K. Diese Kantenmenge entsteht, indem F ebenfalls durch Knotenkontraktion von xy auf K projiziert wird: $F' = \kappa_{xy}(F)$. Laut Definition 2.2.1 ist F' ein Schnitt in K.

2. Sei $F \subseteq E_G \setminus \{xy\}$ ein Schnitt in G. Es gilt $\chi(F) = \sum_{e \in F} c_G(e)$. Die Gewichte in $F' = \kappa_{xy}(F)$ entsprechen entweder einem Kantengewicht in F oder der Summe von zwei Kantengewichten in F (vgl. Def. 2.1.13). Da xy nicht in F liegt, ist das Gewicht dieser Kante nicht relevant. Dies ist die einzige Kante, deren Gewicht nicht auf eine Kante in K übertragen wird. Daher gilt:

$$\chi(F) = \sum_{e \in F} c_G(e) \overset{xy \notin F}{=} \sum_{e' \in F'} c_K(e') = \chi(F') = \chi(\kappa_{xy}(F))$$

□

Im Folgenden werden die 8 Fälle (a)–(h) zur Aufteilung der Eckknoten einer Kreuzung auf zwei durch einen Schnitt separierten Mengen aus Lemma 3.4.3 weiter betrachtet (vgl. Abb. 3.12, S. 44).

3.4.9 Lemma. *Wenn der Schnitt F eines k-fast-planaren Graphen G für die vier Eckknoten einer Kreuzung z_i in G den Fällen (a), (c), (e) oder (h) entspricht, dann ist F ebenfalls ein Schnitt in $G_1 = G - e_{i1}$. Ist F ein maximaler Schnitt in G, so ist F auch in G_1 maximal.*

Beweis. Sei F ein Schnitt in G, der für die vier Eckknoten der Kreuzung z_i in G einem der Fälle (a), (c), (e) oder (h) entspricht; demnach gilt $e_{i1} \notin F$. Sei $G_1 = G - e_{i1}$. Nach Lemma 3.4.7 ist F ebenfalls ein Schnitt in G_1.
Sei F nun ein maximaler Schnitt in G mit $e_{i1} \notin F$. Angenommen es gäbe einen Schnitt F' in G_1, der größer als F ist. Nach Lemma 3.4.5 wäre F' ebenfalls ein Schnitt in G. Dann wäre F' in G ebenfalls größer als F, was der Voraussetzung widerspricht, dass F ein maximaler Schnitt in G ist. Also ist F ebenfalls ein maximaler Schnitt in G_1. □

3.4.10 Lemma. *Wenn der Schnitt F eines k-fast-planaren Graphen G für die vier Eckknoten einer Kreuzung z_i in G den Fällen (a), (b), (d) oder (h) entspricht, dann ist F ebenfalls ein Schnitt in $G_2 = G - e_{i2}$. Ist F ein maximaler Schnitt in G, so ist F auch in G_2 maximal.*

Beweis. Analog zu dem Beweis von Lemma 3.4.9, mit den folgenden Ersetzungen:
Fälle (a), (c), (e) oder (h) $\mapsto$ Fälle (a), (b), (d) oder (h); $G_1 \mapsto G_2$; $e_{i1} \mapsto e_{i2}$. □

3.4.11 Lemma. *Wenn der Schnitt F eines k-fast-planaren Graphen G für die vier Eckknoten einer Kreuzung z_i in G den Fällen (a), (b), (e) oder (f) entspricht, dann ist $F_3 = \kappa_{c_i d_i}(F)$ ein Schnitt in $G_3 = G/c_i d_i$. Es gilt $\chi(F) = \chi(F_3)$. Ist F ein maximaler Schnitt in G, so ist F_3 auch in G_3 maximal.*

Beweis. Sei F ein Schnitt in G, der für die vier Eckknoten der Kreuzung z_i in G einem der Fällen (a), (b), (e) oder (f) entspricht, und sei V_1 die ihn definierende Knotenmenge. Da F einem der Fällen (a), (b), (e) oder (f) entspricht, gilt $c_i, d_i \in V_1$ oder $c_i, d_i \in V_G \setminus V_1$. Daraus folgt: $c_i d_i \notin F$. Sei $G_3 = G/c_i d_i$. Nach Lemma 3.4.8 ist $F_3 = \kappa_{c_i d_i}(F)$ ein Schnitt in G_3 und es gilt $\chi(F) = \chi(F_3)$.
Sei F nun ein maximaler Schnitt in G. Angenommen es gäbe einen Schnitt F' in G_3, der größer als F ist. Nach Lemma 3.4.6 wäre $\kappa^{-1}_{c_i d_i}(F', E_G)$ ebenfalls ein Schnitt in G und dieser wäre wertgleich mit F'. Dann wäre $\kappa^{-1}_{c_i d_i}(F', E_G)$ in G ebenfalls größer als F, was der Voraussetzung widerspricht, dass F ein maximaler Schnitt in G ist. Also ist $F_3 = \kappa_{c_i d_i}(F)$ ebenfalls ein maximaler Schnitt in G_3. □

3.4.12 Lemma. *Wenn der Schnitt F eines k-fast-planaren Graphen G für die vier Eckknoten einer Kreuzung z_i in G den Fällen (a), (c), (d) oder (f) entspricht, dann ist $F' = \kappa_{a_i b_i}(F)$ ein Schnitt in $G' = G/a_i b_i$. Es gilt $\chi(F) = \chi(F')$. Ist F ein maximaler Schnitt in G, so ist F' auch in G' maximal.*

Beweis. Analog zu dem Beweis von Lemma 3.4.11, mit den folgenden Ersetzungen:
Fälle (a), (b), (e) oder (f) $\mapsto$ Fälle (a), (c), (d) oder (f); $G_3 \mapsto G'$; $c_i \mapsto a_i$; $d_i \mapsto b_i$; $F_3 \mapsto F'$. □

3.4.13 Lemma. *Wenn der Schnitt F eines k-fast-planaren Graphen G für die vier Eckknoten einer Kreuzung z_i in G den Fällen (a), (b), (c) oder (g) entspricht, dann ist $F_4 = \kappa_{b_i c_i}(F)$ ein Schnitt in $G_4 = G/b_i c_i$. Es gilt $\chi(F) = \chi(F_4)$. Ist F ein maximaler Schnitt in G, so ist F_4 auch in G_4 maximal.*

Beweis. Analog zu dem Beweis von Lemma 3.4.11, mit den folgenden Ersetzungen:
Fälle (a), (b), (e) oder (f) $\mapsto$ Fälle (a), (b), (c) oder (g); $G_3 \mapsto G_4$; $d_i \mapsto b_i$; $F_3 \mapsto F_4$. □

3.4.14 Lemma. *Wenn der Schnitt F eines k-fast-planaren Graphen G für die vier Eckknoten einer Kreuzung z_i in G den Fällen (a), (d), (e) oder (g) entspricht, dann ist $F' = \kappa_{a_i d_i}(F)$ ein Schnitt in $G' = G/a_i d_i$. Es gilt $\chi(F) = \chi(F')$. Ist F ein maximaler Schnitt in G, so ist F' auch in G' maximal.*

Beweis. Analog zu dem Beweis von Lemma 3.4.11, mit den folgenden Ersetzungen:
Fälle (a), (b), (e) oder (f) $\mapsto$ Fälle (a), (d), (e) oder (g); $G_3 \mapsto G'$; $c_i \mapsto a_i$; $F_3 \mapsto F'$. □

Kapitel 4

Max-Cut-Algorithmen

Die hier vorgestellte Idee für einen Max-Cut-Algorithmus für k-fast-planare Graphen besteht darin, den Graphen so abzuwandeln, dass er planar wird. Beim betrachteten Ansatz wird der Graph für jede Kreuzung auf vier verschiedene Weisen variiert, sodass der resultierende Graph planar wird und der Max-Cut-Algorithmus für planare Graphen angewendet werden kann. Im ersten Abschnitt 4.1 wird der Ansatz zunächst auf 1-fast-planare Graphen angewendet. Im Anschluss wird er in Abschnitt 4.2 auf k-fast-planare Graphen verallgemeinert. In Abschnitt 4.3 wird auf Basis des entwickelten Algorithmus ein FPT-Algorithmus mit Parameter k für das Max-Cut-Problem auf 1-planaren Graphen vorgestellt. Das Kapitel schließt mit der Vorstellung eines weiteren Ansatzes für einen Max-Cut-Algorithmus für 1-fast-planare Graphen und den Gründen warum dieser nicht weiter verfolgt wurde.

4.1 Gewichtete 1-fast-planare Graphen

Da in diesem Abschnitt ausschließlich 1-fast-planare Graphen betrachtet werden, wird – insbesondere für die Beweise – Folgendes als Konvention festgelegt:

> Sei G ein gewichteter echt 1-fast-planarer Graph mit 1-fast-planarer Zeichnung Γ und Kantenkreuzungsmenge $\mathrm{I}(\Gamma) = \{z\}$ mit $z = \{ac, bd\}$. Die sich kreuzenden Kanten seien $e_1 = ac$ und $e_2 = bd$.

Nach Lemma 3.3.5 sind alle Eckknoten einer Kreuzung paarweise entweder durch eine Kante (z. B. e_1 und e_2) oder durch einen Kantenzug verbunden. Zur vereinfachten Darstellung sei die in G enthaltene Kreuzung o. B. d. A. von der Form wie in Abbildung 4.1b zu sehen. Die Kanten zwischen den Eckknoten der Kreuzung – außer e_1 und e_2 – dürfen ggf. überlappende Kantenzüge sein, wie in Abbildung 4.1a zu sehen. Diese Annahmen können laut Satz 3.2.17 und Korollar 3.2.18 getroffen werden. Da in den folgenden Aussagen und ihren Beweisen

(a) Eine einfache Kreuzung in G.

(b) Vereinfachte Darstellung einer einfachen Kreuzung in G.

Abbildung 4.1: Zwei Möglichkeiten einer einfachen Kreuzung in G.

keine Kantengewichte erwähnt oder ausgenutzt wurden, gelten sie sowohl für ungewichtete also auch für gewichtete 1-fast-planare Graphen.
Bei dem nun betrachteten Ansatz zur Berechnung eines maximalen Schnitts in einem 1-fast-planaren Graphen G wird G auf vier verschiedene Weisen planarisiert. Diese Veränderungen beziehen sich jeweils auf die einzige im Graphen enthaltene Kreuzung der Kanten e_1 und e_2 (s. Abb. 4.1). Die vier Variationen von G sehen wie folgt aus: In den ersten beiden Variationen wird jeweils eine der sich kreuzenden Kanten gelöscht (vgl. Def. 2.1.15). In den weiteren beiden Variationen werden die vier Eckknoten der Kreuzung betrachtet und es wird jeweils ein horizontal und ein vertikal nebeneinanderliegendes Eckknotenpaar zu einem Knoten kontrahiert (vgl. Def. 2.1.13).

1. Entferne e_1: $G - e_1$ (s. Abb. 4.2a)
2. Entferne e_2: $G - e_2$ (s. Abb. 4.2b)
3. Verschmelze zwei Knoten horizontal: G/cd (s. Abb. 4.2c)
4. Verschmelze zwei Knoten vertikal: G/bc (s. Abb. 4.2d)

Die Lemmata 3.2.1 und 3.2.2 zeigen, dass die so entstandenen neuen Graphen planar sind. Daher kann ein Max-Cut-Algorithmus für gewichtete planare Graphen auf die vier neuen Graphen angewendet werden.

Algorithmus

Im Folgenden wird der Max-Cut-Algorithmus `GewMaxCut`$_1$ für 1-fast-planare Graphen vorgestellt (Alg. 4.1). Im Algorithmus werden zunächst die vier neuen Graphen aus Abbildung 4.2 erzeugt (Z. 1–4). Im Anschluss wird für jeden neuen Graphen ein Max-Cut-Algorithmus für gewichtete planare Graphen aufgerufen (Z. 6). Zum Schluss wird der Schnitt mit maximalem Gewicht $\chi(F_i)$ bestimmt (Z. 8) und zurück gegeben (Z. 16). Falls der Schnitt F_3 oder F_4 das maximale Gewicht hat, so werden die Schnittkanten durch Knotenexpansion wieder zurück auf G übertragen (Z. 9–15), bevor sie als Lösung zurück gegeben werden.

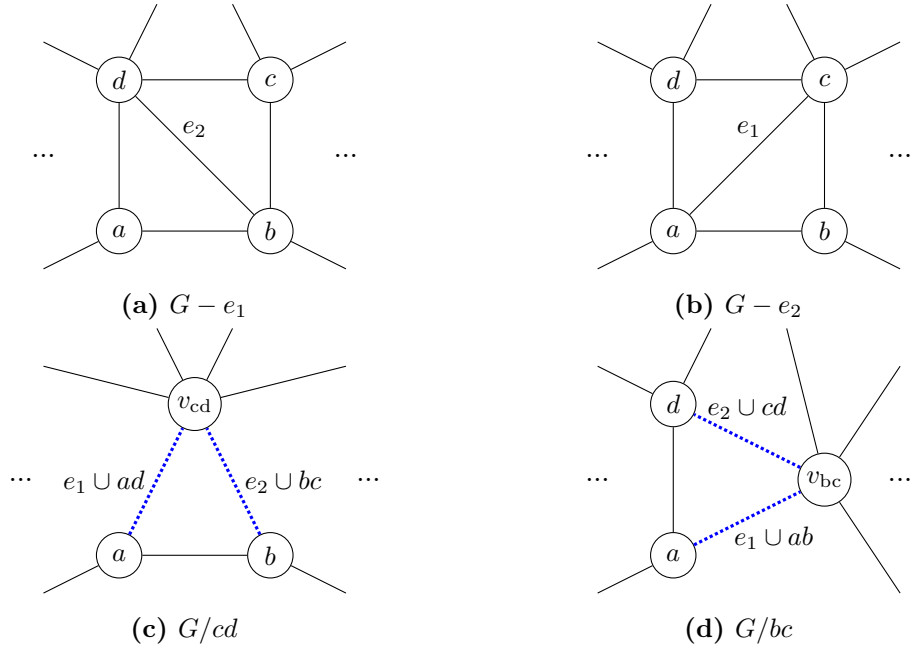
$\underline{\textsc{GewMaxCut}_1(G, \Gamma, \mathrm{I}(\Gamma))}$

Eingabe: Gewichteter ungerichteter 1-fast-planarer Graph G mit 1-fast-planarer Zeichnung Γ und Kantenkreuzungsmenge $\mathrm{I}(\Gamma) = \{\{e_1 = ac, e_2 = bd\}\}$.

Ausgabe: Maximaler Schnitt $F \subseteq E_G$

1: $G_1 = G - e_1$, $\Gamma_1^0 = \Gamma - e_1$ // s. Abb. 4.2a
2: $G_2 = G - e_2$, $\Gamma_2^0 = \Gamma - e_2$ // s. Abb. 4.2b
3: $G_3 = G/cd, \quad \Gamma_3^0 = \Gamma/cd$ // s. Abb. 4.2c
4: $G_4 = G/bc, \quad \Gamma_4^0 = \Gamma/bc$ // s. Abb. 4.2d
5: **for** $i = 1, 2, 3, 4$ **do**
6: $\quad F_i = \textsc{GewMaxCut}_0(G_i, \Gamma_i^0)$
7: **end for**
8: $j = \underset{1 \le i \le 4}{\operatorname{argmax}} \chi(F_i)$ // Def. 2.2.2
9: **if** $j \in \{1, 2\}$ **then**
10: $\quad F = F_j$
11: **else if** $j == 3$ **then**
12: $\quad F = \kappa_{cd}^{-1}(F_3, E_G)$
13: **else**
14: $\quad F = \kappa_{bc}^{-1}(F_4, E_G)$
15: **end if**
16: **return** F

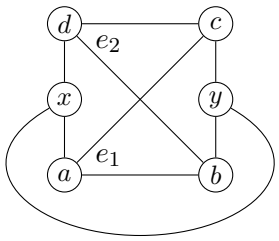
Algorithmus 4.1: Gewichteter Max-Cut-Algorithmus für 1-fast-planare Graphen



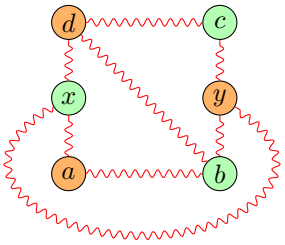
(a) $G - e_1$ **(b)** $G - e_2$

(c) G/cd **(d)** G/bc

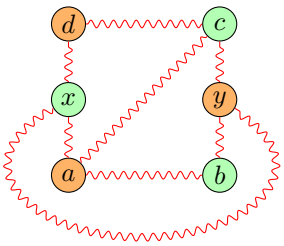
Abbildung 4.2: Mögliche Fälle eine Kreuzung zu entfernen (blau-gepunktete Kanten sind zusammengelegte Kanten).



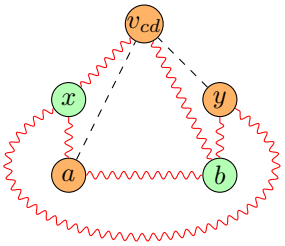
(a) Zeichnung Γ^1 des $K_{3,3}$



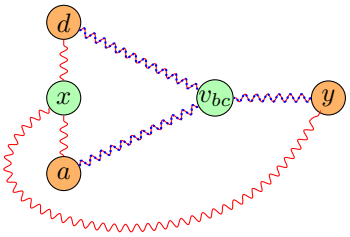
(b) MAX-CUT in $K_{3,3} - e_1$



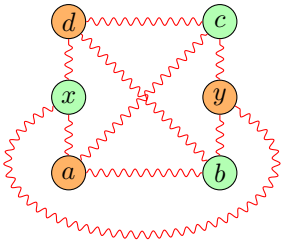
(c) MAX-CUT in $K_{3,3} - e_2$



(d) MAX-CUT in $K_{3,3}/cd$



(e) MAX-CUT in $K_{3,3}/bc$



(f) MAX-CUT in $K_{3,3}$

Abbildung 4.3: Beispiel zur MAX-CUT-Berechnung in dem ungewichteten 1-fast-planaren Graphen $K_{3,3}$ (blau-gepunktete Kanten sind zusammengelegte Kanten mit Gewicht 2; rot-geschwungene und rot-blau-geschwungene Kanten sind Schnittkanten; schwarz-gestrichelte Kanten sind nicht im Schnitt enthalten).

Bevor die Korrektheit und Optimalität des vorgestellten Algorithmus bewiesen werden, wird dessen Vorgehensweise an einem Beispiel verdeutlicht.

4.1.1 Beispiel. Wir betrachten den ungewichteten 1-fast-planaren Graphen $K_{3,3}$. Als Eingabe erhält Algorithmus 4.1 den Graphen $K_{3,3}$, dessen 1-fast-planare Zeichnung Γ^1 aus Abbildung 4.3a und die Kantenkreuzungsmenge $\mathrm{I}(\Gamma^1) = \{\{ac, bd\}\}$. Die in Zeile 1–4 berechneten planaren Graphen und ihre in Zeile 6 berechneten Schnitte sind in den Abbildungen 4.3b–4.3e zu sehen, dabei sind die rot- bzw. rot-blau-geschwungenen Kanten im jeweiligen Schnitt enthalten. Die maximalen Schnitte in $K_{3,3} - e_1$ (Abb. 4.3b) und $K_{3,3} - e_2$ (Abb. 4.3c) haben beide den Wert 8 und der maximale Schnitt in $K_{3,3}/cd$ (Abb. 4.3d) hat einen Wert von 6. Die blau-gepunkteten Kanten in $K_{3,3}/bc$ (dv_{bc}, av_{bc} und $v_{bc}y$) wurden bei der Knotenkontraktion von b und c zusammengelegt und haben daher ein höheres Gewicht ($= 2$), als die restlichen Kanten ($= 1$). Somit hat der maximale Schnitt in $K_{3,3}/bc$ (Abb. 4.3e) einen Wert von 9 und ist damit der größte berechnete Schnitt. Dieser

wird im Anschluss in Zeile 14 zurück auf $K_{3,3}$ übertragen. Der maximale Schnitt in $K_{3,3}$ ist in Abbildung 4.3f zu sehen.

Korrektheit und Optimalität

Nun muss noch gezeigt werden, dass Algorithmus 4.1 einen gültigen und optimalen Schnitt berechnet. Dafür werden die in Abschnitt 3.2 und 3.4 bewiesenen Eigenschaften genutzt. In den Abbildungen A.2a und A.2b im Anhang sind die Beweis-Struktur-Graphen der folgenden beiden Sätze dargestellt.

4.1.2 Satz (Korrektheit). *Der Algorithmus 4.1 berechnet einen gültigen Schnitt für einen* 1*-fast-planaren Graphen G.*

Beweis. Sei G ein 1-fast-planarer Graph, wie global gegeben (s. S. 49). Seien G_i die vom Algorithmus 4.1 berechneten Graphen (Z. 1–4). Nach Lemma 3.2.1 und 3.2.2 sind die Graphen G_i planar. Daher berechnet ein Max-Cut-Algorithmus für gewichtete planare Graphen jeweils einen gültigen Schnitt F_i für G_i (Alg. 4.1, Z. 6). Laut Zeile 8 im Algorithmus 4.1 wird der Größte dieser vier Schnitte F_i auswählt. Laut Lemmata 3.4.5 und 3.4.6 werden die Schnitte F_i korrekt auf G übertragen und erhalten ihre Werte (vgl. Alg. 4.1, Z. 9–15), d. h. $F = \texttt{GewMaxCut}_1(G, \Gamma)$ ist ein gültiger Schnitt in G. □

4.1.3 Satz (Optimalität). *Der maximale Schnitt eines 1-fast-planaren Graphen G ist nicht größer als der vom Algorithmus 4.1 berechnete Schnitt.*

Beweis. Sei G ein 1-fast-planarer Graph, wie global gegeben (s. S. 49), F^* ein maximaler Schnitt in G und $F = \texttt{GewMaxCut}_1(G, \Gamma)$ der vom Algorithmus 4.1 berechnete Schnitt. Seien G_i die vom Algorithmus 4.1 berechneten Graphen (Z. 1–4). Nach Lemma 3.4.3 muss F^* die Eckknoten der Kreuzung z in G entsprechend einem der Fälle (a)–(h) in zwei Mengen teilen.

Fall (a),(c),(e) oder (h): Nach Lemma 3.4.9 ist F^* wertgleich mit einem maximalen Schnitt in G_1. Da G_1 nach Lemma 3.2.1 planar ist, berechnet der Algorithmus $\texttt{GewMaxCut}_0$ einen maximalen Schnitt F_1 in G_1 (Alg. 4.1, Z. 6). Da F wertgleich mit dem größten der vier Schnitte F_i ist (Alg. 4.1, Z. 8; Lemma 3.4.5), folgt:

$$\chi(F^*) \overset{3.4.9}{=} \chi(\text{Max-Cut}(G_1)) \overset{3.2.1}{=} \chi(\texttt{GewMaxCut}_0(G_1)) \overset{4.1(6)}{=} \chi(F_1) \overset{4.1(8)}{\leq} \chi(F)$$

Fall (b) oder (d): Nach Lemma 3.4.10 ist F^* wertgleich mit einem maximalen Schnitt in G_2. Da G_2 nach Lemma 3.2.1 planar ist, berechnet der Algorithmus $\texttt{GewMaxCut}_0$ einen maximalen Schnitt F_2 in G_2 (Alg. 4.1, Z. 6). Da F wertgleich mit dem größten der vier Schnitte F_i ist (Alg. 4.1, Z. 8; Lemma 3.4.5), folgt:

$$\chi(F^*) \overset{3.4.10}{=} \chi(\text{Max-Cut}(G_2)) \overset{3.2.1}{=} \chi(\texttt{GewMaxCut}_0(G_2)) \overset{4.1(6)}{=} \chi(F_2) \overset{4.1(8)}{\leq} \chi(F)$$

Fall (f): Nach Lemma 3.4.11 ist F^* wertgleich mit einem maximalen Schnitt in G_3. Da G_3 nach Lemma 3.2.2 planar ist, berechnet der Algorithmus $\texttt{GewMaxCut}_0$ einen maximalen Schnitt F_3 in G_3 (Alg. 4.1, Z. 6). Da F wertgleich mit dem größten der vier Schnitte F_i ist (Alg. 4.1, Z. 8; Lemma 3.4.6), folgt:

$$\chi(F^*) \overset{3.4.11}{=} \chi(\text{Max-Cut}(G_3)) \overset{3.2.2}{=} \chi(\texttt{GewMaxCut}_0(G_3)) \overset{4.1(6)}{=} \chi(F_3) \overset{4.1(8)}{\leq} \chi(F)$$

Fall (g): Nach Lemma 3.4.13 ist F^* wertgleich mit einem maximalen Schnitt in G_4. Da G_4 nach Lemma 3.2.2 planar ist, berechnet der Algorithmus $\texttt{GewMaxCut}_0$ einen maximalen Schnitt F_4 in G_4 (Alg. 4.1, Z. 6). Da F wertgleich mit dem größten der vier Schnitte F_i ist (Alg. 4.1, Z. 8; Lemma 3.4.6), folgt:

$$\chi(F^*) \overset{3.4.13}{=} \chi(\text{Max-Cut}(G_4)) \overset{3.2.2}{=} \chi(\texttt{GewMaxCut}_0(G_4)) \overset{4.1(6)}{=} \chi(F_4) \overset{4.1(8)}{\leq} \chi(F)$$

In allen Fällen wurde gezeigt, dass F^* nicht größer als F ist. □

4.1.4 Theorem. *Algorithmus 4.1 berechnet einen maximalen Schnitt für 1-fast-planare Graphen bei gegebener 1-fast-planarer Zeichnung.*

Beweis. Durch die Sätze 4.1.2 und 4.1.3 ergibt sich die Korrektheit und Optimalität von Algorithmus 4.1. □

Laufzeit

Für die Laufzeitbeweise wird o. B. d. A. angenommen, dass als Datenstruktur Listen verwendet werden. Effizientere Datenstrukturen können die Laufzeit in der Praxis beschleunigen. Bevor die Laufzeit von Algorithmus 4.1 untersucht wird, werden zunächst die Laufzeiten zur Berechnung einer Kantenlöschung, einer Knotenkontraktion und einer Knotenexpansion betrachtet. Die Laufzeit einer Kantenlöschung folgt direkt aus Definition 2.1.15.

4.1.5 Korollar. *Die Berechnung des Graphen $G - e$ mit $e \in E_G$ benötigt höchstens $|E_G|$ Schritte.*

Die Laufzeiten einer Knotenkontraktion und einer Knotenexpansion folgen ebenfalls aus ihren Definitionen, sind jedoch nicht so offensichtlich und werden daher bewiesen.

4.1.6 Lemma. *Die Berechnung des Graphen G/xy mit $x, y \in V_G$ benötigt höchstens $4|E_G| + 2|V_G| + 1$ Schritte.*

Beweis. Die Erzeugung einer Kontraktion von x und y auf G (Def. 2.1.13) setzt sich aus folgenden Teilberechungen zusammen, die jeweils die folgenden Laufzeiten haben. Das Entfernen der beiden zu verschmelzenden Knoten x und y aus V_G benötigt höchstens $2|V_G|$ Schritte und das Hinzufügen des verschmolzenen Knoten v_{xy} benötigt einen Schritt. Beim

Verschmelzen der Kanten muss für jede Kante e überprüft werden, ob nur x oder nur y oder keiner der beiden Knoten Endknoten der Kante e ist. Im Anschluss muss eine neue bzw. die alte Kante zur neuen Kantenmenge E' hinzugefügt werden. Das Verschmelzen der Kanten benötigt demnach höchstens $3|E_G|$ Schritte. Beim Berechnen der neuen Kantengewichte muss jedes Kantengewicht aus $E_G \setminus \{xy\}$ auf eine Kante in E' übertragen werden. Dafür werden höchstens $|E_G|$ Schritte benötigt. Zusammengefasst ergibt sich eine Laufzeit von höchstens $4|E_G| + 2|V_G| + 1$ Schritten für die Berechnung einer Knotenkontraktion. □

4.1.7 Lemma. *Die Berechnung einer Knotenexpansion $\kappa_{xy}^{-1}(F, E)$ der Kantenmenge $F \subseteq E$ benötigt höchstens $2|F| \cdot |E|$ Schritte.*

Beweis. Die Erzeugung einer Knotenexpansion von v auf F (Def. 2.1.14) setzt sich aus folgenden Teilberechungen zusammen, die jeweils die folgenden Laufzeiten haben. Sei c die Anzahl an Kanten in F, die v als Endknoten haben. Alle Kanten in F, die den kontrahierten Knoten v nicht als Endknoten haben, werden in die neue Kantenmenge übernommen. Diese Berechnung benötigt höchstens $2(|F| - c)$ Schritte. Für jede Kante in F, die den kontrahierten Knoten v als Endknoten hat, wird überprüft, ob sie einer oder zwei Kanten in E entspricht. Diese Berechnung benötigt höchstens $2c \cdot |E|$ Schritte. Wird c durch $|F|$ nach oben abgeschätzt, so ergibt sich eine Gesamtlaufzeit von höchstens $2|F| \cdot |E|$ Schritten. □

In Abbildung A.2c im Anhang ist der Beweis-Struktur-Graph des folgenden Satzes und des daraus resultierenden Theorems dargestellt.

4.1.8 Satz. *Die Laufzeit des Algorithmus 4.1 beträgt $\mathcal{O}(T(n) + n^2)$ bei Eingabe eines 1-fast-planaren Graphen G mit n Knoten und m Kanten, wobei $T(n)$ die Laufzeit des verwendeten* Max-Cut*-Algorithmus für planare Graphen ist.*

Beweis. Die Kantenlöschung in Zeile 1 und 2 benötigen nach Korollar 4.1.5 jeweils höchstens m Schritte. Die Knotenkontraktion in Zeile 3 und 4 benötigen nach Lemma 4.1.6 jeweils höchstens $4m + 2n + 1$ Schritte. Die Knotenexpansion in Zeile 12 und 14 benötigt nach Lemma 4.1.7 jeweils höchstens $2m|F_3|$ bzw. $2m|F_4|$ Schritte. Da diese Größen jedoch unbekannt sind, schätzen wir sie durch $2m^2$ nach oben ab. Nach Lemma 3.3.24 gilt $2m^2 \leq 2(6n - 12)^2$. Der längste Verzweigungsweg in Zeile 9–15 (zu Zeile 14) hat somit eine Länge von höchstens $2(6n - 12)^2 + 3$ Schritten. Die vier Aufrufe des Max-Cut-Algorithmus für gewichtete planare Graphen in Zeile 6 haben jeweils eine Laufzeit von $T(n)$. Somit beträgt die Gesamtlaufzeit höchstens $\mathcal{O}(T(n) + n^2)$. □

Betrachten wir nun die konkrete Laufzeit des Algorithmus 4.1 für zwei verschiedene Max-Cut-Algorithmen für gewichtete planare Graphen: Wird der Algorithmus 2.2 von Mutzel [25,26] verwendet, so beträgt die Laufzeit des Algorithmus 4.1 $\mathcal{O}(n^3)$. Wird hingegen

einer der Algorithmen von Shih, Wu und Kuo [30] oder von Liers und Pardella [23] verwendet, so beträgt die Laufzeit des Algorithmus 4.1 $\mathcal{O}(n^2)$. Dabei hat der Algorithmus von Mutzel eine Laufzeit von $\mathcal{O}(n^3)$ (Satz 2.3.3) und die Algorithmen von Shih, Wu und Kuo und von Liers und Pardella haben eine Laufzeit von $\mathcal{O}(n^{3/2} \cdot \log n)$ [23, 30]. Somit erzielt der Algorithmus 4.1 die beste Laufzeit, wenn einer der MAX-CUT-Algorithmen für gewichtete planare Graphen von Shih, Wu und Kuo oder von Liers und Pardella genutzt wird.

4.1.9 Theorem. *Die bestmögliche Laufzeit des Algorithmus 4.1 beträgt $\mathcal{O}(n^2)$ bei Eingabe eines 1-fast-planaren Graphen G mit n Knoten.*

Beweis. Wird einer der MAX-CUT-Algorithmen für gewichtete planare Graphen von Shih, Wu und Kuo oder von Liers und Pardella in Algorithmus 4.1 verwendet, so gilt $T(n) = \mathcal{O}(n^{3/2} \cdot \log n)$ [23, 30]. Da $\mathcal{O}(n^{3/2} \cdot \log n + n^2) = \mathcal{O}(n^2)$ gilt, folgt die Aussage aus Satz 4.1.8. Eine Verbesserung der Laufzeit des MAX-CUT-Algorithmus für planare Graphen hätte keine Auswirkungen auf die asymptotische Laufzeit des Algorithmus 4.1, da n^2 die Laufzeit dominiert. □

4.2 Gewichtete k-fast-planare Graphen

Da in diesem Abschnitt ausschließlich k-fast-planare Graphen betrachtet werden, wird – insbesondere für die Beweise – Folgendes als Konvention festgelegt:

> Sei G ein echt k-fast-planarer Graph mit k-fast-planarer Zeichnung Γ und Kantenkreuzungsmenge $\mathrm{I}(\Gamma) = \{z_1, \ldots, z_k\}$, wobei $z_i = \{e_{i1}, e_{i2}\}$ für $i = 1, \ldots, k$ gilt. Die sich kreuzenden Kanten der Kreuzung z_i seien $e_{i1} = a_i c_i$ und $e_{i2} = b_i d_i$. Die vier Eckknoten der Kreuzung z_i sind somit a_i, b_i, c_i und d_i.

Nach Lemma 3.3.5 sind alle Eckknoten einer Kreuzung paarweise entweder durch eine Kante (z. B. e_{i1} und e_{i2}) oder durch einen Kantenzug verbunden. In Abbildung 4.1 (S. 50) sind zwei Möglichkeiten einer einfachen Kreuzung in G abgebildet. Zur vereinfachten Darstellung betrachten wir o. B. d. A. Kreuzungen von der Form wie in Abbildung 4.1b zu sehen (Korollar 3.2.18). Dabei sind die Bezeichnungen für die vier Eckknoten a_i, b_i, c_i und d_i einer Kreuzung z_i symbolisch (als Zeiger) zu verstehen, d. h. wenn ein Knoten v Teil mehrerer Kreuzungen ist (z. B. $a_i = b_j = v$) und in einer anderen Kreuzung z_j verschmolzen wird (z. B. zu $v_{b_j c_j}$), so wird für die Kreuzung z_i der Zeiger des betroffenen Knoten auf den entsprechenden neuen Knoten aktualisiert ($a_i = v_{b_j c_j}$).
In Kapitel 3.3 wurden sechs Möglichkeiten zur Planarisierung einer Kreuzung in einem k-fast-planaren Graphen vorgestellt. Wie im Algorithmus 4.1 für 1-fast-planare Graphen werden für jede Kreuzung vier der sechs Möglichkeiten genutzt, um diese zu entfernen: Entweder durch das Entfernen jeweils einer Kreuzungskante (Abb. 4.2a, 4.2b; S. 51) oder durch die Kontraktion von jeweils zwei horizontal bzw. vertikal nebeneinanderliegenden Eckknoten (Abb. 4.2c, 4.2d; S. 51). Nach Lemma 3.3.22 sind die so entstehenden Graphen planar. Daher kann ein Max-Cut-Algorithmus für gewichtete planare Graphen auf die (bis zu 4^k verschiedenen) neuen planaren Graphen angewendet werden.

Algorithmus

Wir wollen nun den rekursiven Max-Cut-Algorithmus `GewMaxCut`$_k$ für k-fast-planare Graphen betrachten (Alg. 4.2). Falls der übergebene Graph G – bzw. dessen Zeichnung Γ – noch Kreuzungen enthält, so erzeugt Algorithmus 4.2 für eine beliebige Kreuzung z aus $\mathrm{I}(\Gamma)$ die vier oben beschriebenen Teilgraphen und ruft sich im Anschluss vier Mal (jeweils mit einem der erzeugten Graphen als Eingabe) rekursiv auf (Z. 6–11). Ein Aufruf des Algorithmus `Contract` (Alg. 4.3) überprüft im Fall einer Knotenkontraktion zuvor, ob die zu verschmelzenden Knoten Eckknoten weiterer Kreuzungen in $\mathrm{I}(\Gamma)$ sind (Alg. 4.3: Z. 5, 11) und löscht diese Kreuzungen aus der Kantenkreuzungsmenge (Alg. 4.3: Z. 21), da sie durch die Kontraktion der gemeinsamen Knoten passiv entfernt werden (Korollar 3.3.17). `Contract` überprüft ebenfalls, ob einer der zu verschmelzenden Knoten Teil einer

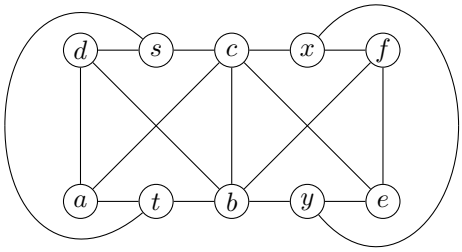
weiteren Kreuzung ist, und benennt diesen in der Kantenkreuzungsmenge um (Alg. 4.3: Z. 7, 13, 16, 18). In Algorithmus 4.2 wird in jedem Rekursionsschritt der Schnitt mit dem maximalen Gewicht bestimmt (Z. 12) und zurück gegeben (Z. 21). Falls der Schnitt F_3 oder F_4 das maximale Gewicht hat, werden die Schnittkanten durch eine Knotenexpansion wieder zurück auf den vorherigen Graphen übertragen (Z. 13–20), bevor sie als Lösung weitergegeben werden. Ist der übergebene Graph planar ($\mathrm{I}(\Gamma) == \emptyset$), so wird für ihn ein Max-Cut-Algorithmus für gewichtete planare Graphen aufgerufen (Z. 2) und der berechnete Schnitt zurückgegeben (Z. 21).
Bevor die Korrektheit und Optimalität des vorgestellten Algorithmus bewiesen werden, wird dessen Vorgehensweise an einem Beispiel verdeutlicht.

$\texttt{GewMaxCut}_{\mathtt{k}}(G, \Gamma, \mathrm{I}(\Gamma))$

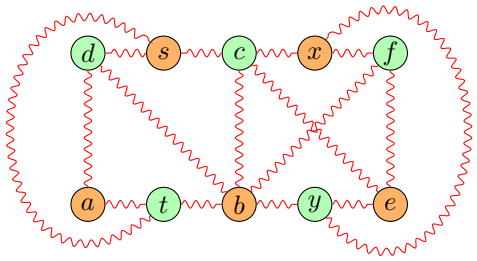
Eingabe: Gewichteter ungerichteter k-fast-planarer Graph G mit k-fast-planarer Zeichnung Γ und Kantenkreuzungsmenge $\mathrm{I}(\Gamma)$ mit $\mathtt{k} = |\mathrm{I}(\Gamma)|$.
Ausgabe: Maximaler Schnitt $F \subseteq E_G$

1: **if** $\mathrm{I}(\Gamma) == \emptyset$ **then**
2: $\quad F = \texttt{GewMaxCut}_{\mathtt{0}}(G, \Gamma)$
3: **else**
4: $\quad$ wähle ein Element $z = \{e_1 = ac, e_2 = bd\} \in \mathrm{I}(\Gamma)$
5: $\quad \mathrm{I}' = \mathrm{I}(\Gamma) \setminus \{z\}$
6: $\quad F_1 = \texttt{GewMaxCut}_{\mathtt{k-1}}(G - e_1, \Gamma - e_1, \mathrm{I}')$
7: $\quad F_2 = \texttt{GewMaxCut}_{\mathtt{k-1}}(G - e_2, \Gamma - e_2, \mathrm{I}')$
8: $\quad \mathrm{I}_3 = \textsc{Contract}(\mathrm{I}', c, d, v_{cd})$
9: $\quad F_3 = \texttt{GewMaxCut}_{|\mathrm{I}_3|}(G/cd, \Gamma/cd, \mathrm{I}_3)$
10: $\quad \mathrm{I}_4 = \textsc{Contract}(\mathrm{I}', b, c, v_{bc})$
11: $\quad F_4 = \texttt{GewMaxCut}_{|\mathrm{I}_4|}(G/bc, \Gamma/bc, \mathrm{I}_4)$
12: $\quad j = \underset{1 \le i \le 4}{\operatorname{argmax}} \chi(F_i)$ // Def. 2.2.2
13: $\quad$ **if** $j \in \{1, 2\}$ **then**
14: $\quad\quad F = F_j$
15: $\quad$ **else if** $j == 3$ **then**
16: $\quad\quad F = \kappa_{cd}^{-1}(F_3, E_G)$
17: $\quad$ **else**
18: $\quad\quad F = \kappa_{bc}^{-1}(F_4, E_G)$
19: $\quad$ **end if**
20: **end if**
21: **return** F

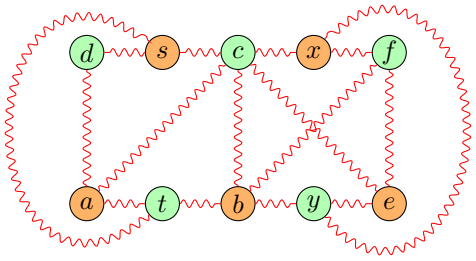
Algorithmus 4.2: Gewichteter Max-Cut-Algorithmus für k-fast-planare Graphen



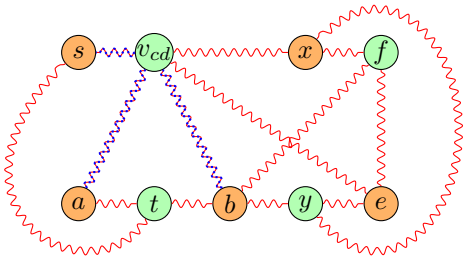
(a) Zeichnung Γ^2 von H



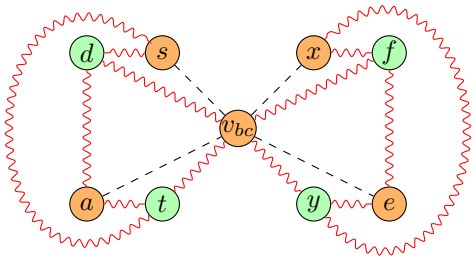
(b) Max-Cut in $H - e_{11}$



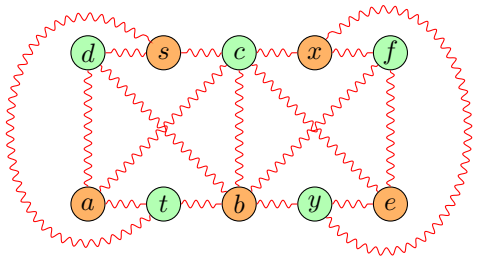
(c) Max-Cut in $H - e_{12}$



(d) Max-Cut in H/cd



(e) Max-Cut in H/bc



(f) Max-Cut in H

Abbildung 4.4: Beispiel zur Max-Cut-Berechnung in einem ungewichteten 2-fast-planaren Graphen H (blau-gepunktete Kanten sind zusammengelegte Kanten mit Gewicht 2; rot-geschwungene und rot-blau-geschwungene Kanten sind Schnittkanten; schwarz-gestrichelte Kanten sind nicht im Schnitt enthalten).

4.2.1 Beispiel. Wir betrachten den ungewichteten 2-fast-planaren Graphen H aus Abbildung 4.4a mit den horizontal benachbarten Kreuzungen $z_1 = \{ac, bd\}$ und $z_2 = \{bf, ce\}$. Als Eingabe erhält Algorithmus 4.2 den Graphen H, dessen 2-fast-planare Zeichnung Γ^2 aus Abbildung 4.4a und die Kantenkreuzungsmenge $\mathrm{I}(\Gamma^2) = \{z_1, z_2\}$. Die in den Zeilen 6, 7, 9 und 11 berechneten 1-fast-planaren Graphen und ihre rekursiv berechneten Schnitte (vgl. Bsp. 4.1.1) sind in den Abbildungen 4.4b–4.4e zu sehen, dabei sind die rot- bzw. rot-blau-geschwungenen Kanten im jeweiligen Schnitt enthalten. Die maximalen Schnitte in $H - e_{11}$ (Abb. 4.4b) und $H - e_{12}$ (Abb. 4.4c) haben beide einen Wert von 16 und der maximale Schnitt in H/bc (Abb. 4.4e) hat einen Wert von 12. Die blau-gepunkteten

Kanten in H/cd (av_{cd}, bv_{cd} und sv_{cd}) wurden bei der Knotenkontraktion von c und d zusammengelegt und haben daher ein höheres Gewicht (= 2), als die restlichen Kanten (= 1). Somit hat der maximale Schnitt in H/cd (Abb. 4.4d) einen Wert von 17 und ist damit der größte berechnete Schnitt. Dieser wird im Anschluss in Zeile 16 zurück auf H übertragen. Der maximale Schnitt in H ist in Abbildung 4.4f zu sehen.

Korrektheit und Optimalität

Nun muss noch gezeigt werden, dass der Algorithmus 4.2 einen gültigen und optimalen Schnitt berechnet. Für den Beweis der Korrektheit muss gezeigt werden, dass in jedem Rekursionsschritt des Algorithmus 4.2 die Schnitte der betrachteten Teilgraphen korrekt auf G zurück übertragen werden. Die Reihenfolge, in der die Kreuzungen aus G entfernt werden, ist dabei irrelevant. In Abbildung A.3a im Anhang ist der Beweis-Struktur-Graph des folgenden Satzes dargestellt.

4.2.2 Satz (Korrektheit). *Der Algorithmus 4.2 berechnet einen gültigen Schnitt für einen k-fast-planaren Graphen G.*

Beweis. Wir beweisen die Aussage per Induktion nach k:
IA: Für $k = 0$ ruft der Algorithmus 4.2 einen Max-Cut-Algorithmus für gewichtete planare Graphen auf. Dieser berechnet einen gültigen Schnitt für G, da G planar ist.
IV: Die Aussage gilt für einen $(k-1)$-fast-planaren Graphen.
IS: Sei G ein k-fast-planarer Graph wie global gegeben (s. S. 57) und z_i eine beliebige Kreuzung in G. Seien $G_1 = G - e_{i1}$, $G_2 = G - e_{i2}$, $G_3 = G/c_id_i$ und $G_4 = G/b_ic_i$ die vom Algorithmus 4.2 berechneten Graphen (Z. 6–11). Nach Lemma 3.3.3 und 3.3.4 sind die Graphen G_i $(k-1)$-fast-planar. Daher kann die Induktionsvoraussetzung (IV) angewendet werden und ihre rekursiv durch Algorithmus 4.2 berechneten Schnitte F_i sind gültig (vgl. Z. 6–11). Der Algorithmus 4.2 wählt den größten dieser vier Schnitte aus (Z. 12) und überträgt ihn auf G (Z. 13–20). Laut Lemmata 3.4.5 und 3.4.6 werden die Schnitte F_i korrekt auf G übertragen und erhalten ihre Werte, d. h. $F = \texttt{GewMaxCut}_\texttt{k}(G, \Gamma)$ ist ein gültiger Schnitt in G. □

Für den Beweis der Optimalität muss hingegen gezeigt werden, dass der Algorithmus – unabhängig von der Reihenfolge, in der die Kreuzungen aus G entfernt werden – für jede Kreuzung alle 8 Möglichkeiten zur Aufteilung der vier Eckknoten auf zwei Mengen berücksichtigt (vgl. Lemma 3.4.3) und somit eine optimale Lösung findet.
Einen Spezialfall bilden die passiv entfernten Kreuzungen. Dieser wird in den beiden folgenden Lemmata betrachtet. In den Abbildungen A.3b und A.3c im Anhang sind die Beweis-Struktur-Graphen der beiden Lemmata dargestellt.

CONTRACT(I, x, y, v_{xy})

Eingabe: Kantenkreuzungsmenge I, die zu verschmelzenden Knoten x, y und der verschmolzene Knoten v_{xy}.

Ausgabe: Kantenkreuzungsmenge, in der jedes Vorkommen von x oder y durch v_{xy} ersetzt wurde und in der alle Kantenpaare entfernt wurden, die sich nicht mehr kreuzen.

```
1:  H = ∅                                   // Menge der Kanten, die sich nicht mehr kreuzen
2:  for z = {e1, e2} ∈ I do
3:    if e1 == xv then
4:      if e2 == yw then
5:        H = H ∪ z
6:      else
7:        e1 = v_xy v
8:      end if
9:    else if e1 == yv then
10:     if e2 == xw then
11:       H = H ∪ z
12:     else
13:       e1 = v_xy v
14:     end if
15:   else if e2 == xv then
16:     e2 = v_xy v
17:   else if e2 == yv then
18:     e2 = v_xy v
19:   end if
20: end for
21: return I \ H
```

Algorithmus 4.3: Algorithmus zum Verschmelzen der Knoten x, y in I, der das Ergebnis von sich nicht mehr kreuzenden Kantenpaaren bereinigt.

4.2.3 Lemma. *Falls zwei Kreuzungen sich zwei Eckknoten teilen und eine von beiden im Algorithmus 4.2 passiv – durch Kontraktion der gemeinsamen Knoten in der anderen Kreuzung – entfernt wird, so werden für die beiden Kreuzungen trotzdem alle 32 Möglichkeiten zur Aufteilung der sechs Eckknoten auf zwei Mengen berücksichtigt.*

Beweis. Seien z_i und z_j zwei beliebige Kreuzungen in G, die sich zwei Eckknoten teilen. Sei F der von Algorithmus 4.2 berechnete Schnitt. Laut Lemma 3.3.10 müssen die gemeinsamen Eckknoten nebeneinanderliegend sein. Wir nehmen o. B. d. A. an, dass die Kreuzungen horizontal benachbart sind. Die folgenden Aussagen lassen sich leicht auf zwei vertikal

benachbarte Kreuzungen übertragen, indem alle betrachteten (Teil-)Graphen um 90° gedreht werden.
Sei z_i die linke und z_j die rechte Kreuzung (vgl. Abb. 3.13, S. 45). Dann gilt: $b_i = a_j$ und $c_i = d_j$. Laut Lemma 3.4.4 gibt es 2 Möglichkeiten zur Aufteilung der zwei gemeinsamen Eckknoten auf zwei Mengen.
Fall 1: *Die gemeinsamen Knoten werden nicht vom Schnitt F getrennt*
Laut Lemma 3.4.4 entsprechen in diesem Fall die Eckknoten von z_i in F einem der Fälle (a), (b), (c) oder (g) aus Lemma 3.4.3. Nach Lemma 3.4.13 ist F wertgleich mit dem maximalen Schnitt in G/b_ic_i. Da durch das Verschmelzen der gemeinsamen Knoten $b_i = a_j$ und $c_i = d_j$ beide Kreuzungen gleichzeitig entfernt werden, muss noch überprüft werden, ob alle verbleibenden Möglichkeiten zur Aufteilung der vier Eckknoten von z_j in G/b_ic_i berücksichtigt werden; Diese sind nach Lemma 3.4.4: (a), (d), (e) oder (g). Falls F für die Eckknoten von z_j einem dieser Fällen entspricht, so ist F nach Lemma 3.4.14 wertgleich mit dem maximalen Schnitt in G/a_jd_j. Da $b_ic_i = a_jd_j$ gilt, ist der Graph identisch zu G/b_ic_i. Somit werden alle 16 Möglichkeiten zur Aufteilung der sechs Eckknoten in Fall 1 im Graphen G/b_ic_i berücksichtigt.
Fall 2: *Die gemeinsamen Knoten werden vom Schnitt F getrennt*
Laut Lemma 3.4.4 entsprechen in diesem Fall die Eckknoten von z_i in F einem der Fälle (d), (e), (f) oder (h). Falls F für die Eckknoten von z_i dem Fall (e) oder (h) / (d) / (f) entspricht, so ist dieser wertgleich mit dem maximalen Schnitt in $G - e_{i1}$ / $G - e_{i2}$ / G/c_id_i (Lemma 3.4.9 / 3.4.10 / 3.4.11). In diesem Fall werden die gemeinsamen Knoten nicht verschmolzen und die Kreuzung z_j bleibt nach Entfernen der Kreuzung z_i noch bestehen. Es verbleiben nach Lemma 3.4.4 noch vier Möglichkeiten zur Aufteilung der vier Eckknoten von z_j in F: (b), (c), (f) oder (h). Falls F für die Eckknoten von z_j dem Fall (c) oder (h) / (b) / (f) entspricht, so ist dieser wertgleich mit dem maximalen Schnitt in $G - e_{j1}$ / $G - e_{j2}$ / G/c_jd_j (Lemma 3.4.9 / 3.4.10 / 3.4.11). Mit Lemma 3.4.9, 3.4.10 und 3.4.11 folgt: alle 16 Möglichkeiten zur Aufteilung der sechs Eckknoten in Fall 2 werden in den neun Graphen $G' - e_{j1}$, $G' - e_{j2}$ und G'/c_jd_j mit $G' \in \{G - e_{i1}, G - e_{i2}, G/c_id_i\}$ berücksichtigt. □

4.2.4 Lemma. *Falls eine Kreuzung im Algorithmus 4.2 passiv entfernt wird, werden für sie trotzdem alle 8 Möglichkeiten zur Aufteilung der vier Eckknoten auf zwei Mengen berücksichtigt.*

Beweis. Es gibt zwei Möglichkeiten für eine Kreuzung passiv entfernt zu werden. Entweder durch die Kontraktion zweier Eckknoten, die sie sich mit einer weiteren Kreuzung teilt, (vgl. Korollar 3.3.17 und Beispiel 3.3.15) oder durch die Kontraktion einer ihrer Eckknoten, den sie sich mit einer anderen Kreuzung teilt, mit einem Eckknoten der anderen Kreuzung, wodurch eine Kreuzungskante so gezeichnet werden kann, dass sie die andere Kreuzungskante nicht mehr schneidet (vgl. Korollar 3.3.20 und Beispiel 3.3.18). Der zweite Fall wird von dem betrachteten Algorithmus nicht berücksichtigt, d. h. die Kreuzung wird im Algorithmus

nicht passiv entfernt, sondern verbleibt in der Kantenkreuzungsmenge, um aktiv entfernt zu werden. Der erste Fall wird durch den Aufruf des Algorithmus 4.3 im Algorithmus 4.2 berücksichtigt. Für diesen Fall zeigt Lemma 4.2.3 die gewünschte Aussage. □

Für den folgenden Beweis der Optimalität des Algorithmus 4.2 werden die Erkenntnisse aus Abschnitt 3.3 und 3.4 genutzt. In Abbildung A.3d im Anhang ist der Beweis-Struktur-Graph des folgenden Satzes dargestellt.

4.2.5 Satz (Optimalität). *Der maximale Schnitt eines k-fast-planaren Graphen G ist nicht größer als der vom Algorithmus 4.2 berechnete Schnitt.*

Beweis. Wir beweisen die Aussage per Induktion nach k:
IA: Für $k = 0$ ruft der Algorithmus 4.2 einen Max-Cut-Algorithmus für gewichtete planare Graphen auf. Dieser berechnet einen maximalen Schnitt für G, da G planar ist.
IV: Die Aussage gilt für einen $(k-1)$-fast-planaren Graphen.
IS: Sei G ein k-fast-planarer Graph wie global gegeben (s. S. 57), z_i eine beliebige Kreuzung in G, F^* ein maximaler Schnitt in G und F der vom Algorithmus 4.2 berechnete Schnitt. Nach Lemma 3.4.3 muss F^* die Eckknoten der Kreuzung z_i in G entsprechend einem der Fälle (a)–(h) (s. S. 44) in zwei Mengen teilen.

Fall (a),(c),(e) oder (h): Nach Lemma 3.4.9 ist F^* wertgleich mit einem maximalen Schnitt in $G - e_{i1}$. Da $G - e_{i1}$ nach Lemma 3.3.3 $(k-1)$-fast-planar ist, kann die Induktionsvoraussetzung (IV) angewendet werden. Sei F_1 ein maximaler Schnitt in $G - e_{i1}$ nach Algorithmus 4.2, Zeile 6. Da F wertgleich mit dem größten der vier Schnitte F_i ist (Alg. 4.2, Z. 12; Lemma 3.4.5), folgt:

$$\chi(F^*) \overset{3.4.9}{=} \chi(\text{Max-Cut}(G - e_{i1})) \overset{(IV)}{\leq} \chi(\texttt{GewMaxCut}_{\texttt{k-1}}(G - e_{i1})) \overset{4.2(6)}{=} \chi(F_1) \overset{4.2(12)}{\leq} \chi(F)$$

Fall (b) oder (d): Nach Lemma 3.4.10 ist F^* wertgleich mit einem maximalen Schnitt in $G - e_{i2}$. Da $G - e_{i2}$ nach Lemma 3.3.3 $(k-1)$-fast-planar ist, kann die Induktionsvoraussetzung (IV) angewendet werden. Sei F_2 ein maximaler Schnitt in $G - e_{i2}$ nach Algorithmus 4.2, Zeile 7. Da F wertgleich mit dem größten der vier Schnitte F_i ist (Alg. 4.2, Z. 12; Lemma 3.4.5), folgt:

$$\chi(F^*) \overset{3.4.10}{=} \chi(\text{Max-Cut}(G - e_{i2})) \overset{(IV)}{\leq} \chi(\texttt{GewMaxCut}_{\texttt{k-1}}(G - e_{i2})) \overset{4.2(7)}{=} \chi(F_2) \overset{4.2(12)}{\leq} \chi(F)$$

Fall (f): Nach Lemma 3.4.11 ist F^* wertgleich mit einem maximalen Schnitt in $G/c_i d_i$. Da $G/c_i d_i$ nach Lemma 3.3.4 $(k-1)$-fast-planar ist, kann die Induktionsvoraussetzung

(IV) angewendet werden. Sei F_3 ein maximaler Schnitt in G/c_id_i nach Algorithmus 4.2, Zeile 9. Da F wertgleich mit dem größten der vier Schnitte F_i ist (Alg. 4.2, Z. 12; Lemma 3.4.6), folgt:

$$\chi(F^*) \overset{3.4.11}{=} \chi(\textsc{Max-Cut}(G/c_id_i)) \overset{(IV)}{\leq} \chi(\texttt{GewMaxCut}_{\texttt{k-1}}(G/c_id_i)) \overset{4.2(9)}{=} \chi(F_3) \overset{4.2(12)}{\leq} \chi(F)$$

Fall (g): Nach Lemma 3.4.13 ist F^* wertgleich mit einem maximalen Schnitt in G/b_ic_i. Da G/b_ic_i nach Lemma 3.3.4 $(k-1)$-fast-planar ist, kann die Induktionsvoraussetzung (IV) angewendet werden. Sei F_4 ein maximaler Schnitt in G/b_ic_i nach Algorithmus 4.2, Zeile 11. Da F wertgleich mit dem größten der vier Schnitte F_i ist (Alg. 4.2, Z. 12; Lemma 3.4.6), folgt:

$$\chi(F^*) \overset{3.4.13}{=} \chi(\textsc{Max-Cut}(G/b_ic_i)) \overset{(IV)}{\leq} \chi(\texttt{GewMaxCut}_{\texttt{k-1}}(G/b_ic_i)) \overset{4.2(11)}{=} \chi(F_4) \overset{4.2(12)}{\leq} \chi(F)$$

Nun muss noch der Fall betrachtet werden, dass eine Kreuzung passiv entfernt wird. Falls es mindestens eine Kreuzung z_j gibt, die sich zwei Eckknoten mit z_i teilt, und diese gemeinsamen Eckknoten in einem der Fälle (f) oder (g) verschmolzen werden, so werden alle Kreuzungen planarisiert, die die verschmolzenen Knoten als Eckknoten hatten (Lemma 3.3.4). Aus Lemmata 4.2.3 und 4.2.4 folgt, dass in diesem Fall alle Möglichkeiten zur Aufteilung der Eckknoten von z_j berücksichtigt werden, d. h. auch für die passiv entfernte Kreuzung z_j wird ein optimaler Schnitt gefunden.
In allen Fällen wurde gezeigt, dass F^* nicht größer als F ist. Da die Kreuzung z_i beliebig gewählt wurde, ist die Aussage unabhängig von der Reihenfolge, in der die Kreuzungen aus G entfernt werden. □

4.2.6 Theorem. *Algorithmus 4.2 berechnet einen maximalen Schnitt für k-fast-planare Graphen bei gegebener k-fast-planarer Zeichnung.*

Beweis. Durch die Sätze 4.2.2 und 4.2.5 ergibt sich die Korrektheit und Optimalität von Algorithmus 4.2. □

Laufzeit

Für die Laufzeitbeweise wird o. B. d. A. angenommen, dass als Datenstruktur Listen verwendet werden. Effizientere Datenstrukturen können die Laufzeit in der Praxis beschleunigen. Bevor die Laufzeit von Algorithmus 4.2 untersucht wird, betrachten wir zunächst die Laufzeit von Algorithmus 4.3.

4.2.7 Lemma. *Der Algorithmus 4.3 benötigt höchstens* $t^2 + 5t + 1$ *Schritte bei Eingabe (*I*, x, y,* v_{xy}*) mit* $t = |\mathrm{I}|$.

Beweis. Sei c die maximale Anzahl an Kreuzungen in I, die sowohl x als auch y als Eckknoten enthalten. Es gilt $c = |H|$ und $c \leq t = |\mathrm{I}|$. Ein Aufruf des Algorithmus 4.3 benötigt höchstens $1 + t \cdot 5 + c \cdot t$ Schritte. Diese Laufzeit ergibt sich aus dem Anlegen einer leeren Liste in Zeile 1, der Anzahl von Wiederholungen der Schleife in Zeile 2 ($|\mathrm{I}| = t$) multipliziert mit der Länge des längsten Verzweigungswegs innerhalb der Schleife (zu Zeile 18) und dem Entfernen der c Elemente in H aus I in Zeile 21. Schätzen wir c durch t nach oben ab, ergibt sich eine Gesamtlaufzeit von höchstens $t^2 + 5t + 1$ Schritten. □

In Abbildung A.3e im Anhang ist der Beweis-Struktur-Graph des folgenden Satzes und des daraus resultierenden Theorems dargestellt.

4.2.8 Satz. *Die Laufzeit des Algorithmus 4.2 beträgt* $\mathcal{O}(4^k \cdot (T(n) + n^2))$ *bei Eingabe eines k-fast-planaren Graphen G mit n Knoten, wobei* $T(n)$ *die Laufzeit des verwendeten* Max-Cut*-Algorithmus für planare Graphen ist.*

Beweis. Auf eine Reduktion der Größe von G nach einem rekursiven Aufruf wird in diesem Beweis verzichtet, da es an der asymptotischen Laufzeit nichts ändert. Seien daher $n = |V_G|$ und $m = |E_G|$ feste Konstanten.
Sei I die Kantenkreuzungsmenge in Γ. Sei $L(k)$ die Laufzeit des Algorithmus 4.2 mit $k = |\mathrm{I}|$. Um die Laufzeit eines Rekursionsschritts zu berechnen, betrachten wir zunächst die Laufzeiten der einzelnen Schritte im Algorithmus:
Tritt die Abbruchbedingung in Zeile 1 in Kraft ($k = 0$), so wird ein Max-Cut-Algorithmus für gewichtete planare Graphen aufgerufen. Dieser hat eine Laufzeit von $T(n)$. Zusammen mit den Operationen in Zeile 1 und 21 ergibt sich für $k = 0$ die folgende rekursive Laufzeit: $L(0) = T(n) + 2$
Für den Fall $k > 0$ wird die Bedingung in Zeile 1 trotzdem überprüft. Dies benötigt einen Schritt. Da als Datenstruktur eine Liste vorausgesetzt wurde, benötigt die Operation in Zeile 4 einen Schritt und die Löschen-Operation in Zeile 5 benötigt höchstens k Schritte. Die beiden Kantenlöschungen in Zeile 6 und 7 benötigen nach Korollar 4.1.5 höchstens $2m$ Schritte. Da ein Element aus I entfernt wurde (Z. 5), gilt $|\mathrm{I}'| = k - 1$. Somit und nach Lemma 4.2.7 benötigen die beiden Aufrufe von Contract (Alg. 4.3) in Zeile 8 und 10 höchstens $2 \cdot ((k-1)^2 + 5 \cdot (k-1) + 1)$ Schritte. Die beiden Knotenkontraktionen in Zeile 9

und 11 benötigen nach Lemma 4.1.6 höchstens $2 \cdot (4m+2n+1)$ Schritte. Die vier rekursiven Aufrufe des Algorithmus 4.2 in den Zeilen 6, 7, 9 und 11 haben jeweils eine Laufzeit von höchstens $L(k-1)$, da in jedem Schritt mindestens eine Kreuzung entfernt wird (Lemma 3.3.3 und 3.3.4). Die Knotenexpansion in Zeile 16 bzw. 18 benötigt nach Lemma 4.1.7 jeweils höchstens $2m|F_3|$ bzw. $2m|F_4|$ Schritte. Da diese Größen jedoch unbekannt sind, schätzen wir sie durch $2m^2$ nach oben ab. Der längste Verzweigungsweg in Zeile 13 – 19 (zu Zeile 18) hat somit eine Länge von höchstens $2m^2+3$ Schritten. Mit den $4+1$ Schritten der Operationen in Zeile 12 und 21 ergibt sich folgende rekursive Laufzeit für $k > 0$:

$$\begin{aligned} L(k) &\leq 1+1+k+2m+2\cdot((k-1)^2+5\cdot(k-1)+1)+2\cdot(4m+2n+1) \\ &\quad +4\cdot L(k-1)+4+(2m^2+3)+1 \\ &= 4\cdot L(k-1)+2k^2+7k+2m^2+10m+4n+6 \end{aligned}$$

Da n und m Konstanten sind, wird zur besseren Lesbarkeit des Beweises die Variable $c = 2m^2+10m+4n+6$ eingeführt. Im Folgenden wird zunächst die Intuition zur Abschätzung der Rekursionsformel durch eine konkrete Laufzeit gegeben.

$$\begin{aligned} L(k) &\leq 4\cdot L(k-1)+2k^2+7k+c \\ &= 4\cdot[4\cdot[4\cdot[\ldots[4\cdot[4\cdot(T(n)+2)+2\cdot 1^2+7\cdot 1+c] \\ &\qquad +2\cdot 2^2+7\cdot 2+c] \\ &\qquad \ldots] \\ &\qquad +2\cdot(k-2)^2+7\cdot(k-2)+c] \\ &\quad +2\cdot(k-1)^2+7\cdot(k-1)+c] \\ &\quad +2k^2+7k+c] \\ &= 4^k\cdot[T(n)+2]+4^{k-1}\cdot(2\cdot 1^2+7\cdot 1+c) \\ &\qquad +4^{k-2}\cdot(2\cdot 2^2+7\cdot 2+c) \\ &\qquad +\ldots \\ &\qquad +4^2\cdot(2\cdot(k-2)^2+7\cdot(k-2)+c) \\ &\qquad +4^1\cdot(2\cdot(k-1)^2+7\cdot(k-1)+c) \\ &\qquad +4^0\cdot(2k^2+7k+c) \\ &= 4^k\cdot[T(n)+2]+\sum_{i=1}^{k}4^{k-i}\cdot(2i^2+7i+c) \end{aligned}$$

Im Folgenden wir diese Abschätzung der Rekursionsformel per Induktion nach k bewiesen.
IA: Für $k = 0$ gilt:

$$L(0) = T(n)+2 = 1\cdot[T(n)+2]+0 = 4^0\cdot[T(n)+2]+\sum_{i=1}^{0}4^{0-i}\cdot(2i^2+7i+c)$$

IV: Es gilt: $L(k-1) \leq 4^{k-1} \cdot [T(n)+2] + \sum_{i=1}^{k-1} 4^{(k-1)-i} \cdot (2i^2+7i+c)$

IS: Für $k > 0$ gilt:

$$\begin{aligned}
L(k) &\leq 4 \cdot L(k-1) + 2k^2 + 7k + c \\
&\overset{\text{IV}}{\leq} 4 \cdot \left(4^{k-1} \cdot [T(n)+2] + \sum_{i=1}^{k-1} 4^{(k-1)-i} \cdot (2i^2+7i+c) \right) + 2k^2 + 7k + c \\
&= 4^k \cdot [T(n)+2] + \sum_{i=1}^{k-1} 4^{k-i} \cdot (2i^2+7i+c) + 4^{k-k} \cdot (2k^2+7k+c) \\
&= 4^k \cdot [T(n)+2] + \sum_{i=1}^{k} 4^{k-i} \cdot (2i^2+7i+c)
\end{aligned}$$

Somit gilt die Abschätzung der Rekursionsformel durch eine konkrete Laufzeit. Diese wird nun weiter vereinfacht.

$$\begin{aligned}
L(k) &\leq 4^k \cdot [T(n)+2] + \sum_{i=1}^{k} 4^{k-i} \cdot (2i^2+7i+c) \\
&= 4^k \cdot [T(n)+2 + \sum_{i=1}^{k} 4^{-i} \cdot (2i^2+7i+c)] \\
&\leq 4^k \cdot [T(n)+2 + \sum_{i=1}^{k} 4^{-i} \cdot (2k^2+7k+c)] \\
&\overset{*_1}{=} 4^k \cdot [T(n)+2 + \frac{1}{3} \cdot \left(1 - \left(\frac{1}{4}\right)^k\right) \cdot (2k^2+7k+c)] \\
&= 4^k \cdot [T(n)+2] + 4^k \cdot \frac{1}{3} \cdot \left(1 - \left(\frac{1}{4}\right)^k\right) \cdot (2k^2+7k+c) \\
&\overset{*_c}{=} 4^k \cdot [T(n)+2] + \frac{1}{3} \cdot (4^k - 1) \cdot (2k^2+7k+2m^2+10m+4n+6) \\
&\leq 4^k \cdot [T(n)+2] + \frac{1}{3} \cdot 4^k \cdot (3k^2+9k+3m^2+12m+6n+6) \\
&= 4^k \cdot [T(n)+2+k^2+3k+m^2+4m+2n+2] \\
&\overset{*_2}{\leq} 4^k \cdot [T(n)+2+\left(\frac{m}{2}\right)^2 + 3 \cdot \left(\frac{m}{2}\right) + m^2 + 4m + 2n + 2] \\
&= 4^k \cdot [T(n)+2+\frac{1}{4} \cdot m^2 + \frac{3}{2} \cdot m + m^2 + 4m + 2n + 2] \\
&\in \mathcal{O}(4^k \cdot [T(n) + m^2 + n])
\end{aligned}$$

$*_1$ Durch Anwendung der geometrischen Summe folgt:

$$\begin{aligned}
\sum_{i=1}^{k} 4^{-i} &= \sum_{i=0}^{k} \left(\frac{1}{4}\right)^i - 1 = \frac{1 - \left(\frac{1}{4}\right)^{k+1}}{1 - \frac{1}{4}} - 1 = \frac{4}{3} \cdot \left(1 - \left(\frac{1}{4}\right)^{k+1}\right) - 1 \\
&= \frac{1}{3} - \frac{1}{3} \cdot \left(\frac{1}{4}\right)^k = \frac{1}{3} \cdot \left(1 - \left(\frac{1}{4}\right)^k\right)
\end{aligned}$$

*c Zur besseren Lesbarkeit wurde die Variable $c = 2m^2 + 10m + 4n + 6$ eingeführt.
*2 Da jede Kante höchstens eine andere Kante schneidet, folgt $k \leq \frac{m}{2}$ (Lemma 3.3.1). Schätzen wir m durch $6n-12$ nach oben ab (Lemma 3.3.24), so ergibt sich die Gesamtlaufzeit von $\mathcal{O}(4^k \cdot (T(n) + n^2))$. □

Betrachten wir nun die konkrete Laufzeit des Algorithmus 4.2 für zwei verschiedene Max-Cut-Algorithmen für gewichtete planare Graphen: Wird der Algorithmus 2.2 von Mutzel [25, 26] verwendet, so beträgt die Laufzeit des Algorithmus 4.2 $\mathcal{O}(4^k \cdot n^3)$. Wird hingegen einer der Algorithmen von Shih, Wu und Kuo [30] oder von Liers und Pardella [23] verwendet, so beträgt die Laufzeit des Algorithmus 4.2 $\mathcal{O}(4^k \cdot n^2)$. Dabei hat der Algorithmus von Mutzel eine Laufzeit von $\mathcal{O}(n^3)$ (Satz 2.3.3) und die Algorithmen von Shih, Wu und Kuo und von Liers und Pardella haben eine Laufzeit von $\mathcal{O}(n^{3/2} \cdot \log n)$ [23, 30]. Somit erzielt der Algorithmus 4.2 die beste Laufzeit, wenn einer der Max-Cut-Algorithmen für gewichtete planare Graphen von Shih, Wu und Kuo oder von Liers und Pardella genutzt wird.

4.2.9 Theorem. *Die bestmögliche Laufzeit des Algorithmus 4.2 beträgt $\mathcal{O}(4^k \cdot n^2)$ bei Eingabe eines k-fast-planaren Graphen G mit n Knoten.*

Beweis. Wird einer der Max-Cut-Algorithmen für gewichtete planare Graphen von Shih, Wu und Kuo oder von Liers und Pardella in Algorithmus 4.2 verwendet, so gilt $T(n) = \mathcal{O}(n^{3/2} \cdot \log n)$ [23, 30]. Da $\mathcal{O}(n^{3/2} \cdot \log n + n^2) = \mathcal{O}(n^2)$ gilt, folgt die Aussage aus Satz 4.2.8. Eine Verbesserung der Laufzeit des Max-Cut-Algorithmus für planare Graphen hätte keine Auswirkungen auf die asymptotische Laufzeit des Algorithmus 4.2, da n^2 die Laufzeit dominiert. □

4.3 Gewichtete 1-planare Graphen

Da in diesem Abschnitt ausschließlich 1-planare Graphen betrachtet werden, wird – insbesondere für die Beweise – Folgendes als Konvention festgelegt:

> Sei G ein 1-planarer Graph mit 1-planarer Zeichnung Γ.

Der Max-Cut-Algorithmus für k-fast-planare Graphen lässt sich zu einem parametrisierbaren Max-Cut-Algorithmus für 1-planare Graphen mit Parameter k erweitern. Der Parameter k gibt dabei die Anzahl der einfachen Kantenkreuzungen in einer gegebenen 1-planaren Zeichnung an. Der Algorithmus `KANTENKREUZUNGSMENGE` (Alg. 4.4) berechnet für eine gegebene 1-planare Zeichnung Γ_G den Parameter k sowie die Kantenkreuzungsmenge der Zeichnung. Dazu überprüft er für jedes Kantenpaar aus E_G, ob sich die Polygonzüge der Kanten in Γ_G schneiden.

4.3.1 Lemma. *Algorithmus 4.4 berechnet die Menge und Anzahl aller sich kreuzenden Kantenpaare in der Zeichnung Γ des Graphen G.*

Beweis. In Zeile 3 des Algorithmus wird für alle Kantenpaare aus E_G (Z. 2) überprüft, ob sie sich schneiden. Da in Zeile 4 alle Kantenpaare, die sich in Γ schneiden, zur Menge K hinzugefügt werden, enthält K in Zeile 7 alle Kreuzungen in Γ. Zudem wird die aktuelle Anzahl an Kreuzungen in K in der Variablen k gespeichert (Z. 1, 4). Diese ist zusammen mit K Teil der Ausgabe in Zeile 7. Somit berechnet `KANTENKREUZUNGSMENGE` die Menge K und Anzahl k aller sich kreuzenden Kantenpaare in der Zeichnung Γ. □

Wir werden nun sehen, dass sich die Menge der Kantenkreuzungen in polynomieller Zeit berechnen lässt.

`KANTENKREUZUNGSMENGE`(G, Γ)

Eingabe: Gewichteter ungerichteter Graph G mit Zeichnung Γ
Ausgabe: Kantenkreuzungsmenge $\mathrm{I}(\Gamma)$ und $k = |K|$

1: $K = \emptyset$; $k = 0$;
2: **for** $e_1, e_2 \in E_G$ **do**
3: **if** $z = \{e_1, e_2\}$ ist eine Kreuzung in Γ **then**
4: $K = K \cup \{z\}$; $k = k + 1$;
5: **end if**
6: **end for**
7: **return** (K, k)

Algorithmus 4.4: Berechnet die Menge und Anzahl aller Kreuzungen $\mathrm{I}(\Gamma)$ in einer 1-planaren Zeichnung Γ.

4.3.2 Lemma. *Algorithmus 4.4 hat eine Laufzeit von $\mathcal{O}(l^2 \cdot m^2)$ bei Eingabe des Graphen G mit Zeichnung Γ, wobei l die maximale Anzahl an Teilstrecken, aus denen ein Polygonzug P_e^Γ besteht, und m die Anzahl an Kanten in G bezeichnet.*

Beweis. Um zu überprüfen, ob ein Kantenpaar $\{e_1, e_2\} \subseteq E_G$ eine Kreuzung in Γ ist (Z. 3), muss überprüft werden, ob $\mathring{P}_{e_1}^\Gamma \cap \mathring{P}_{e_2}^\Gamma \neq \emptyset$ gilt. Bezeichne $\#(P_e^\Gamma)$ die Anzahl der Teilstrecken aus denen der Polygonzug P_e^Γ besteht. Dann kann in höchstens $\#(P_{e_1}^\Gamma) \cdot \#(P_{e_2}^\Gamma)$ Schritten überprüft werden, ob sich zwei Teilstrecken von $\mathring{P}_{e_1}^\Gamma$ und $\mathring{P}_{e_2}^\Gamma$ schneiden. Sei l der maximale Wert den $\#(P_e^\Gamma)$ für alle Polygonzüge P_e^Γ mit $e \in E_G$ annimmt. Da die Schleife in Zeile 2 genau m^2 Mal ausgeführt wird und die Überprüfung der Bedingung in Zeile 3 höchstens l^2 Schritte benötigt, liegt die Laufzeit des Algorithmus insgesamt bei $\mathcal{O}(l^2 \cdot m^2)$. □

Der MAX-CUT-Algorithmus für 1-planare Graphen nutzt den von KANTENKREUZUNGSMENGE berechneten Parameter k, um mit dem MAX-CUT-Algorithmus für k-fast-planare Graphen den maximalen Schnitt durch die gegebene 1-planare Zeichnung zu berechnen. Da der Algorithmus 4.5 das MAX-CUT-Entscheidungsproblem löst, wird zum Schluss überprüft, ob er berechnete Schnitt mindestens eine vorgegebene Größe s hat.

4.3.3 Satz (Korrektheit). *Algorithmus 4.5 berechnet, ob es in einem 1-planaren Graphen G mit 1-planarer Zeichnung Γ einen Schnitt der Größe s oder größer gibt.*

Beweis. Nach Lemma 4.3.1 wird die Kantenkreuzungsmenge und die Anzahl an Kreuzungen in Γ korrekt berechnet (Z.1). Da G nun nach Definition 2.1.35 k-fast-planar ist, berechnet $\texttt{GEWMAXCUT}_\texttt{k}$ in Zeile 2 nach Theorem 4.2.6 einen maximalen Schnitt in G. In Zeile 3 wird überprüft, ob der Wert des berechneten Schnitts größer oder gleich dem vorgegebenen Wert s ist. □

Die Optimalität des in Algorithmus 4.5 berechneten Schnitts folgt aus Satz 4.2.5 und Lemma 4.3.1. Betrachten wir nun die Laufzeit von Algorithmus 4.5.

4.3.4 Satz (Laufzeit). *Algorithmus 4.5 hat eine Laufzeit von $\mathcal{O}(4^k \cdot n^2 + l^2 \cdot n^2)$ bei Eingabe eines 1-planaren Graphen G mit n Knoten und 1-planarer Zeichnung Γ mit k einfachen Kreuzungen, wobei l die maximale Anzahl an Teilstrecken bezeichnet, aus denen ein Polygonzug P_e^Γ mit $e \in E_G$ besteht.*

Beweis. Nach Lemma 4.3.2 benötigt der Aufruf von KANTENKREUZUNGSMENGE in Zeile 1 $\mathcal{O}(l^2 \cdot m^2)$ Schritte. Ein Aufruf von $\texttt{GEWMAXCUT}_\texttt{k}$ in Zeile 2 benötigt nach Theorem 4.2.9 $\mathcal{O}(4^k \cdot n^2)$ Schritte. Die Überprüfung der Bedingung in Zeile 3 benötigt höchstens $|F| \leq m$ Schritte. Somit hat der längste Verzweigungsweg in Zeile 3 – 7 (zu Zeile 6) eine Länge von höchstens $m + 1$ Schritten. Wird m durch $6n - 12$ nach oben abgeschätzt (Lemma 3.3.24), so ergibt sich eine Gesamtlaufzeit von $\mathcal{O}(4^k \cdot n^2 + l^2 \cdot n^2)$ für Algorithmus 4.5. □

$\underline{\textsc{GewMaxCut}_{\texttt{1-pl}}(G, \Gamma)}$

Eingabe: Gewichteter ungerichteter 1-planarer Graph G mit 1-planarer Zeichnung Γ und $s \in \mathbb{N}$.

Ausgabe: Gibt es einen Schnitt F in G mit $\chi(F) \geq s$?

1: $(K, \mathtt{k}) = \textsc{Kantenkreuzungsmenge}(G, \Gamma)$
2: $F = \textsc{GewMaxCut}_{\mathtt{k}}(G, \Gamma, K)$
3: **if** $\chi(F) \geq s$ **then**
4: **return** true
5: **else**
6: **return** false
7: **end if**

Algorithmus 4.5: Max-Cut-Algorithmus für gewichtete 1-planare Graphen

Betrachten wir nun die konkrete Laufzeit des Algorithmus 4.5 für bestimmte Werte des Parameters l. Bekos et. al. stellten 2017 einen Polynomialzeitalgorithmus vor, der für jede 1-planare Einbettung eines Graphen eine 1-planare Zeichnung erzeugt, in der alle Kanten aus höchstens zwei geraden Teilstecken bestehen und Kanten sich ausschließlich im rechten Winkel schneiden [5]. Daher existiert für jeden 1-planaren Graphen eine 1-planare Zeichnung mit $l \leq 2$. Wird solch eine Zeichnung als Eingabe für Algorithmus 4.5 verwendet, so gilt:

4.3.5 Theorem (Laufzeit). *Algorithmus 4.5 hat eine Laufzeit von $\mathcal{O}(4^k \cdot n^2)$ bei Eingabe eines 1-planaren Graphen G mit n Knoten und 1-planarer Zeichnung Γ mit k Kreuzungen, wenn Γ mit dem Algorithmus von Bekos et. al. [5] erzeugt wurde.*

Beweis. Da die 1-planare Zeichnung Γ von G mit dem Algorithmus von Bekos et. al. erzeugt wurde, bestehen alle Kanten in Γ aus höchstens zwei geraden Teilstecken [5]. Es gilt $l \leq 2$. Somit folgt die Aussage aus Satz 4.3.4. □

Betrachten wir den Max-Cut-Algorithmus für 1-planare Graphen nun unter dem Aspekt der Parametrisierbarkeit.

4.3.6 Satz (FPT). *Der Max-Cut-Algorithmus für 1-planare Graphen (4.5) ist parametrisierbar mit Parameter k bei gegebener 1-planarer Zeichnung Γ, wobei k die Anzahl der Kreuzungen in Γ bezeichnet.*

Beweis. Sei der betrachtete Parameter k die Anzahl an Kantenkreuzungen in Γ. Diese kann nach Lemmata 4.3.1 und 4.3.2 in polynomieller Zeit mit Algorithmus 4.4 berechnet werden (Zeile 1). Nach Satz 4.3.4 beträgt die Laufzeit von Algorithmus 4.5 $\mathcal{O}(4^k \cdot n^2 + l^2 \cdot n^2)$, wobei l die maximale Anzahl an Teilstrecken bezeichnet, aus denen ein Polygonzug P_e^Γ mit $e \in E_G$ besteht. Die Laufzeit kann durch $\mathcal{O}(4^k \cdot (l^2 \cdot n^2))$ nach oben abgeschätzt werden.

Für $f(k) = 4^k$ und $p(|G|, |\Gamma|) = l^2 \cdot n^2$ folgt mit Definition 2.5.1, dass Algorithmus 4.5 ein FPT-Algorithmus mit Parameter k ist. □

Da soeben ein FPT-Algorithmus für das MAX-CUT-Problem für 1-planare Graphen bei gegebener 1-planarer Zeichnung vorgestellt wurde, kann folgende Aussage getroffen werden.

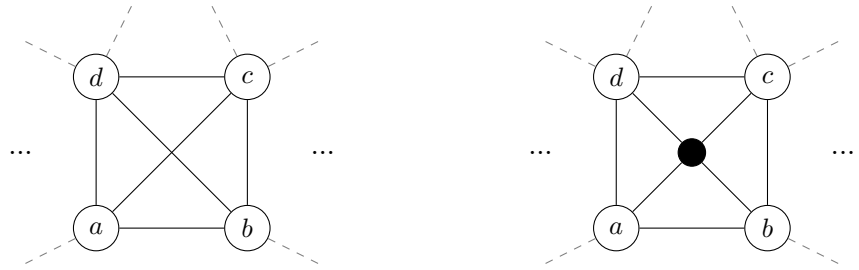
4.3.7 Theorem (FPT). *Das* MAX-CUT*-Problem für 1-planare Graphen ist parametrisierbar mit Parameter k bei gegebener 1-planarer Zeichnung Γ, wobei k die Anzahl der Kreuzungen in Γ bezeichnet.*

Beweis. Da in Satz 4.3.6 bewiesen wurde, dass ein FPT-Algorithmus mit Parameter k existiert, der das MAX-CUT-Problem für 1-planare Graphen bei gegebener 1-planarer Zeichnung löst, folgt die Aussage aus Definition 2.5.2. □

Das parametrisierbare MAX-CUT-Problem für 1-planare Graphen mit Parameter k liegt bei gegebener 1-planarer Zeichnung in der Komplexitätsklasse **FPT**, wobei k die Anzahl der Kreuzungen in der gegebenen Zeichnung ist.

4.4 Ein anderer Ansatz

Der hier vorgestellte Ansatz wurde als erste Idee zur Entwicklung eines Max-Cut-Algorithmus für ungewichtete 1-fast-planare Graphen verfolgt. Dessen Entwicklung sowie die Gründe, warum er nicht weiter verfolgt wurde, werden in diesem Abschnitt kurz dargestellt.



(a) Einfache Kreuzung **(b)** Einfache Kreuzung mit Hilfsknoten

Abbildung 4.5: Einfügen eines Hilfsknotens auf einer einfachen Kreuzung.

Die grundlegende Idee zur Entwicklung eines Max-Cut-Algorithmus für ungewichtete 1-fast-planare Graphen besteht auch hier darin, den Graphen so abzuwandeln, dass er planar wird, um den Max-Cut-Algorithmus für planare Graphen anwenden zu können. Dazu wird bei diesem Ansatz auf jede Kreuzung ein Knoten gesetzt, wie in Abbildung 4.5 zu sehen. Im Anschluss kann der Max-Cut-Algorithmus für planare Graphen (Alg. 2.1) angewendet werden und der duale Graph wird gebildet (s. Abb. 4.6). Nachdem im dualen Graphen ein maximaler Eulerkreis gefunden wurde, muss bei der Übertragung der Eulerkanten (im dualen Graphen) auf die Schnittkanten (im ursprünglichen Graphen) jedoch das vorherige Einfügen des zusätzlichen Hilfsknoten berücksichtigt werden.

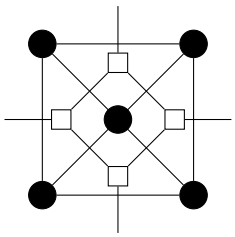
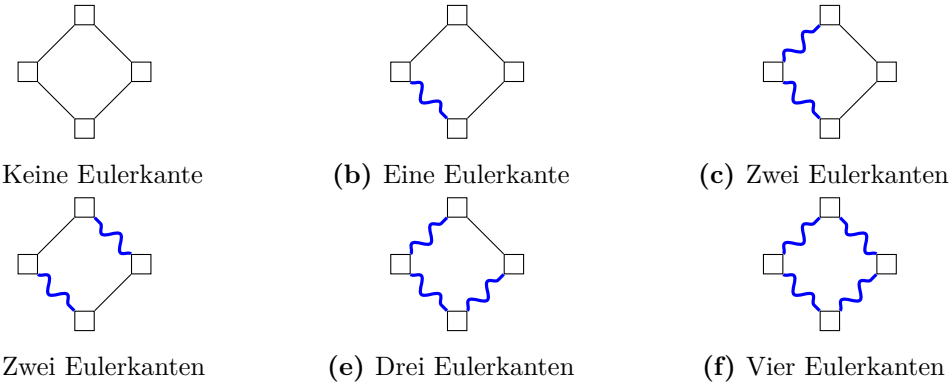


Abbildung 4.6: Planarisierte Kreuzung mit dualem Graphen D

Die Notation in diesem Abschnitt richtet sich nach den in Algorithmus 2.1 auf Seite 17 verwendeten Variablennamen. Sei G ein planarer Graph mit planarer Zeichnung Γ_G^0. Sei D der duale Graph zur Zeichnung Γ_G^0. Die Kantenmenge J bezeichnet einen minimalen T-Join in D, wobei T alle Knoten aus D mit ungeradem Grad beinhaltet. Die Kantenmenge $F_D = E_D \setminus J$ ist ein maximaler Eulerkreis in D.
Betrachten wir zunächst welche Möglichkeiten zur Wahl der Eulerkanten es für die vier künstlich erzeugten Kanten im dualen Graphen gibt. Für den Beweis wird folgendes Lemma benötigt.



(a) Keine Eulerkante **(b)** Eine Eulerkante **(c)** Zwei Eulerkanten

(d) Zwei Eulerkanten **(e)** Drei Eulerkanten **(f)** Vier Eulerkanten

Abbildung 4.7: Mögliche Wahl der Eulerkanten F_D im dualen Graphen D.
(F_D: blau und geschwungen, J: schwarz und gerade)

4.4.1 Lemma. *Alle Pfade aus J haben minimale Länge in D.*

Beweis. Die Kantenmenge $J \subseteq E_D$ ergibt sich aus einem minimalen Matching M in dem gewichteten vollständigen Graphen H. Dieser setzt sich aus allen Knoten des dualen Graphen D mit ungeradem Grad und Kanten mit der Länge des kürzesten Weges zwischen diesen Knoten in D als Kantengewichten zusammen (s. Alg. 2.1). Das Matching M in H induziert durch die symmetrische Differenz der in D verwendeten Kanten die Kantenmenge J in D. Da das Matching nicht in D, sondern in H berechnet wurde, sind angrenzende Kanten in J möglich. Da das Matching M jedoch minimal (in Bezug auf die Kantengewichte) in H ist, müssen alle Pfade in J auch minimal (in Bezug auf die Länge eines Pfades zwischen zwei Knoten) in D sein. □

4.4.2 Lemma. *Für die vier künstlich erzeugten Kanten im dualen Graphen D gibt es 11 Möglichkeiten zur Wahl der Eulerkanten. Lässt man die punktsymmetrischen Fälle weg, verbleiben vier mögliche Optionen zur Kantenauswahl (s. Abb. 4.7c-4.7f).*

Beweis. Wir färben die Eulerkanten blau und die Nicht-Eulerkanten schwarz. Um die Anzahl der Möglichkeiten zur Wahl der Eulerkanten in dem gegebenen Teilgraph zu berechnen, zählen wir die möglichen Farbbelegungen für die vier Kanten ab. Dafür ziehen wir 4 Mal – für jede Kante einmal – mit Zurücklegen aus einem Topf mit zwei Farben. Das entspricht $2^4 = 16$ Möglichkeiten für die Farbverteilung auf die vier Kanten. Diese lassen sich in drei Gruppen aufteilen:

1. Alle Kanten in einer Menge und keine Kante in der anderen Menge
2. Drei Kanten in einer Menge und ein Kanten in der anderen Menge
3. Zwei Kanten in einer Menge und zwei Kanten in der anderen Menge

Die erste Gruppe enthält 2, die zweite 8 und die dritte 6 der 16 Fälle. Für die erste Gruppe sind die beiden Fälle in den Abbildungen 4.7a und 4.7f zu sehen. Für die zweite Gruppe

sind die ersten vier Fälle in Abbildung 4.7b (rotiert um $0°, 90°, 180°$ und $270°$) und die weiteren vier Fälle in der Abbildung 4.7e (rotiert um $0°, 90°, 180°$ und $270°$) zu sehen. Für die dritte Gruppe sind die ersten vier Fälle in Abbildung 4.7c (rotiert um $0°, 90°, 180°$ und $270°$) und die weiteren zwei Fälle in der Abbildung 4.7d (rotiert um $0°, 90°$) zu sehen.
Die beiden Fälle mit keiner bzw. nur einer Eulerkante aus Abbildung 4.7a und 4.7b können allerdings nicht eintreten. Betrachten wir die beiden Fälle separat. (Die Notation orientiert sich an den in Algorithmus 2.1 verwendeten Bezeichnungen. Die Eulerkantenmenge F_D entspricht dem Komplement der Kantenmenge $J \subseteq E_D$.)

Fall (a) Da jeder Pfad, der einen Kreis enthält, kürzer wird, wenn man den Kreis entfernt, kann die Kantenmenge J (schwarze Kanten) nach Lemma 4.4.1 keine Kreise enthalten. Somit kann Fall (a) in $F_D = E_D \setminus J$ nicht vorkommen.

Fall (b) Die schwarzen geraden Kanten in Abbildung 4.7b entsprechen der Kantenmenge J. Der Weg über die blaue geschwungene Kante ist allerdings kürzer als der Umweg über zwei zusätzliche Kanten. Somit kann nach Lemma 4.4.1 Fall (b) in $F_D = E_D \setminus J$ nicht vorkommen.
Also müssen wir von den 16 (rechnerisch möglichen) Fällen die 5 Fälle aus den Abbildungen 4.7a und 4.7b (rotiert um $0°, 90°, 180°$ und $270°$) abziehen. Es verbleiben 11 Möglichkeiten zur Wahl der Eulerkanten. Diese werden durch die vier Abbildungen 4.7c-4.7f repräsentiert.□

Umwandlung der Eulerkanten in Schnittkanten

Nun müssen die Eulerkanten im dualen Graphen D wieder zurück auf G übertragen werden. Dabei muss jedoch das vorherige Einfügen des zusätzlichen Hilfsknotens berücksichtigt werden. Betrachten wir nun für die vier Möglichkeiten zur Wahl der Eulerkanten im Teilgraph der Kreuzung des dualen Graphen D (s. Abb. 4.7c-4.7f) die vier Übertragungen auf einen Schnitt in G.

Fall (c) Falls jeweils eine duale Teilkante beider Kreuzungskanten in der Eulermenge enthalten ist, sind beide Kreuzungskanten im Schnitt enthalten.

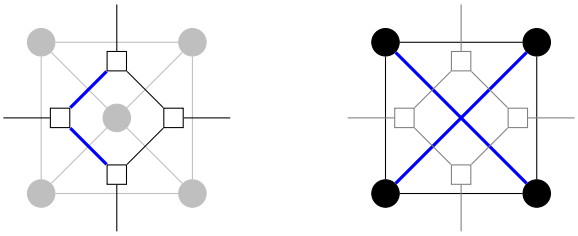


Abbildung 4.8: Umwandlung Eulerkanten zu Schnittkanten (Fall c)

Fall (d) Falls beide dualen Teilkanten einer Kreuzungskanten in der Eulermenge enthalten sind, ist diese Kreuzungskante nicht im Schnitt enthalten.

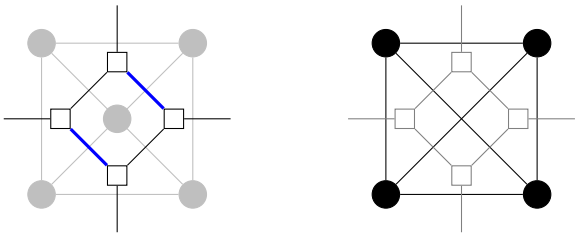


Abbildung 4.9: Umwandlung Eulerkanten zu Schnittkanten (Fall d)

Fall (e) Falls nur eine duale Teilkante der einen Kreuzungskanten und beide dualen Teilkanten der anderen Kreuzungskante in der Eulermenge enthalten sind, ist die erste Kreuzungskanten im Schnitt und die zweite Kreuzungskante nicht im Schnitt enthalten.

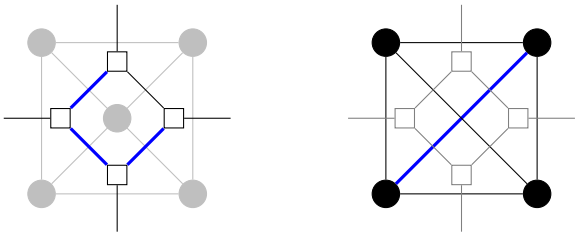


Abbildung 4.10: Umwandlung Eulerkanten zu Schnittkanten (Fall e)

Fall (f) Falls beide dualen Teilkanten beider Kreuzungskanten in der Eulermenge enthalten sind, ist keine der beiden Kreuzungskanten im Schnitt enthalten.

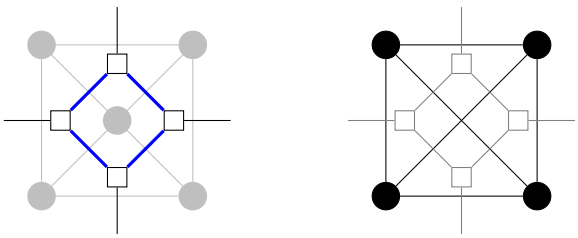


Abbildung 4.11: Umwandlung Eulerkanten zu Schnittkanten (Fall f)

In den Fällen (d), (e) und (f) stellt sich die Frage, ob der berechnete Schnitt maximal sein kann, da die Größe des berechneten Eulerkreises im dualen Graphen um 2 bzw. 4 größer ist, als der übertragene Schnitt. Da durch diese Frage der Optimalitätsbeweis immer komplexer wurde, wurde dieser Ansatz verworfen, um einen anderen Ansatz zu verfolgen. Ob es sich lohnt diesen Ansatz weiter zu verfolgen oder ob es ein Gegenbeispiel gibt, bleibt offen.

Kapitel 5

Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war, einen Max-Cut-Algorithmus für k-fast-planare Graphen auf Basis eines Max-Cut-Algorithmus für planare Graphen zu entwickeln. Dafür wurde in Kapitel 4 zunächst ein Max-Cut-Algorithmus für 1-fast-planare Graphen mit der Laufzeit $\mathcal{O}(n^2)$ entworfen. Dieser wurde im Anschluss zu einem rekursiven Max-Cut-Algorithmus für k-fast-planare Graphen mit Laufzeit $\mathcal{O}(4^k \cdot n^2)$ erweitert. Da in Abschnitt 4.3 gezeigt werden konnte, dass die Anzahl der Kreuzungen in einer 1-planaren Zeichnung in polynomieller Zeit berechenbar ist, konnte der Max-Cut-Algorithmus für k-fast-planare Graphen außerdem zu einem parametrisierbaren Max-Cut-Algorithmus für 1-planare Graphen mit Parameter k bei gegebener 1-planarer Zeichnung mit k Kreuzungen erweitert werden. Dieser hat eine Laufzeit von $\mathcal{O}(4^k \cdot n^2)$, wenn die verwendete Zeichnung mit dem Algorithmus von Bekos et. al. [5] erzeugt wurde. Theorem 4.3.7 zeigt, dass das Max-Cut-Problem für 1-planare Graphen bei gegebener 1-planarer Zeichnung Γ parametrisierbar mit Parameter k ist, wobei k die Anzahl der Kantenkreuzungen in Γ angibt, und somit in der Komplexitätsklasse **FPT** liegt.
Weitere Resultate dieser Arbeit sind eine obere Schranke von $6n - 12$ für die Anzahl der Kanten in einem 1-planaren Graphen (Lemma 3.3.24) und eine obere Schranke von $3n - 6$ für die Anzahl der Kreuzungen in einer 1-planaren Zeichnung (Lemma 3.3.25). Des Weiteren wurden in Abschnitt 3.3 die Eigenschaften von Kreuzungen untersucht, die sich Eckknoten teilen. Lemma 3.3.8 zeigt, dass zwei Kreuzungen in einem k-fast-planaren Graphen sich höchstens zwei Eckknoten teilen können und diese nebeneinanderliegend sein müssen. Zudem wurden die Probleme aufgezeigt, die entstehen können, wenn zwei Kreuzungen sich Eckknoten teilen. So wurde in Abschnitt 3.3 der Begriff einer passiv entfernten Kreuzung eingeführt. In Bezug auf Schnitteigenschaften wurde in Abschnitt 3.4 gezeigt, dass es acht Möglichkeiten gibt einen Schnitt durch eine Kreuzung zu legen. Außerdem wurden in Abschnitt 3.3 sechs Möglichkeiten aufgezeigt eine einfache Kreuzung aus einem Graphen zu entfernen. Satz 4.2.5 zeigt, dass es ausreicht für jede Kreuzung

vier dieser Möglichkeiten zu betrachten, um einen maximalen Schnitt in dem betrachten Graphen zu finden.
Es bleibt offen, ob das Max-Cut-Problem für 1-planare Graphen ohne gegebene Zeichnung ebenfalls parametrisierbar ist. Ein möglicher Ansatz dazu könnte den 2013 von Bannister et. al. [1] vorgestellten FPT-Algorithmus für den 1-Planaritätstest auf Graphen mit Parameter v nutzen, wobei v die Größe einer minimalen Knotenüberdeckung in G bezeichnet. Die zu überprüfende Hypothese wäre, ob der FPT-Algorithmus zum 1-Planaritätstest von Bannister et. al. [1] zusammen mit dem Zeichenalgorithmus für 1-planare Graphen von Bekos et. al. [5] und dem vorgestellten FPT-Max-Cut-Algorithmus für 1-planare Graphen (bei gegebener Zeichnung) zu einem parametrisierbaren Max-Cut-Algorithmus für 1-planare Graphen (ohne gegebene Zeichnung) zusammengefügt werden können.
Zudem verbleibt die Aufgabe der Implementierung der Algorithmen aus Kapitel 4 sowie die Untersuchung der praktischen Laufzeiten durch experimentelle Laufzeittests. Des Weiteren bleibt die Frage offen, ob der in Abschnitt 4.4 vorgestellte Ansatz zur Entwicklung eines Max-Cut-Algorithmus für 1-fast-planare Graphen möglicherweise doch funktioniert oder ob es ein Gegenbeispiel gibt.

Literaturverzeichnis

[1] Bannister, Michael J., Sergio Cabello und David Eppstein: *Parameterized Complexity of 1-Planarity.* In: Dehne, Frank, Roberto Solis-Oba und Jörg-Rüdiger Sack (Herausgeber): *Algorithms and Data Structures - 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, Band 8037 der Reihe *Lecture Notes in Computer Science*, Seiten 97–108. Springer, 2013.

[2] Barahona, Francisco: *The max-cut problem on graphs not contractible to K5.* Operations Research Letters, 2(3):107–111, 1983.

[3] Barahona, Francisco: *A solvable case of quadratic 0-1 programming.* Discrete Applied Mathematics, 13(1):23–26, 1986.

[4] Barahona, Francisco, Martin Grötschel, Michael Jünger und Gerhard Reinelt: *An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design.* Operations Research, 36(3):493–513, 1988.

[5] Bekos, Michael A., Walter Didimo, Giuseppe Liotta, Saeed Mehrabi und Fabrizio Montecchiani: *On RAC drawings of 1-planar graphs.* Theor. Comput. Sci., 689:48–57, 2017.

[6] De Simone, Caterina, Martin Diehl, Michael Jünger, Petra Mutzel, Gerhard Reinelt und Giovanni Rinaldi: *Exact ground states of Ising spin glasses: New experimental results with a branch-and-cut algorithm.* Journal of Statistical Physics, 80(1-2):487–496, 1995.

[7] Diestel, Reinhard: *Graphentheorie (3. Aufl.).* Springer, Berlin, 2006.

[8] Edmonds, Jack und Ellis L. Johnson: *Matching, Euler tours and the Chinese postman.* Mathematical Programming, 5(1):88–124, 1973.

[9] Fáry, Istvan: *On straight line representation of planar graphs.* Acta Sci. Math., 11:229–233, 1948.

[10] Flum, Jörg: *Parameterized Complexity and Logic.* In: Cooper, S. Barry, Benedikt Löwe und Andrea Sorbi (Herausgeber): *Computation and Logic in the Real World, Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 18-23, 2007, Proceedings*, Band 4497 der Reihe *Lecture Notes in Computer Science*, Seiten 278–289, Berlin, 2007. Springer.

[11] Fraysseix, Hubert de, János Pach und Richard Pollack: *Small Sets Supporting Fáry Embeddings of Planar Graphs.* In: Simon, Janos (Herausgeber): *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, Seiten 426–433. ACM, 1988.

[12] Galluccio, A. und M. Loebl: *Max cut in toroidal graphs.* Instituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche, Oktober 1998.

[13] Grötschel, Martin und George L. Nemhauser: *A polynomial algorithm for the max-cut problem on graphs without long odd cycles.* Mathematical Programming, 29(1):28–40, 1984.

[14] Grötschel, Martin und William R. Pulleyblank: *Weakly bipartite graphs and the max-cut problem.* Operations Research Letters, 1(1):23–27, 1981.

[15] Hadlock, F.: *Finding a Maximum Cut of a Planar Graph in Polynomial Time.* SIAM J. Comput., 4(3):221–225, 1975.

[16] Hochstein, Jan M.: *Maximale Flüsse in fast-planaren Graphen.* Doktorarbeit, Fachbereich Informatik, Technische Universität Darmstadt, Darmstadt, Juni 2007.

[17] Hong, Seok-Hee, Peter Eades, Giuseppe Liotta und Sheung-Hung Poon: *Fáry's Theorem for 1-Planar Graphs.* In: Gudmundsson, Joachim, Julián Mestre und Taso Viglas (Herausgeber): *Computing and Combinatorics - 18th Annual International Conference, COCOON 2012, Sydney, Australia, August 20-22, 2012. Proceedings*, Band 7434 der Reihe *Lecture Notes in Computer Science*, Seiten 335–346. Springer, 2012.

[18] Itai, Alon und Yossi Shiloach: *Maximum flow in planar networks.* SIAM Journal on Computing, 8(2):135–150, 1979.

[19] Karp, Richard M.: *Reducibility Among Combinatorial Problems.* In: Miller, Raymond E. und James W. Thatcher (Herausgeber): *Proceedings of a symposium on the Complexity of Computer Computations, New York*, The IBM Research Symposia Series, Seiten 85–103. Plenum Press, New York, 1972.

[20] Kobourov, Stephen G., Giuseppe Liotta und Fabrizio Montecchiani: *An annotated bibliography on 1-planarity.* CoRR, abs/1703.02261, 2017.

[21] Korzhik, Vladimir P. und Bojan Mohar: *Minimal Obstructions for 1-Immersions and Hardness of 1-Planarity Testing.* Journal of Graph Theory, 72(1):30–71, 2013.

[22] Kuratowski, Casimir: *Sur le probleme des courbes gauches en topologie.* Fundamenta mathematicae, 15(1):271–283, 1930.

[23] Liers, Frauke und Gregor Pardella: *A Simple MAX-CUT Algorithm for Planar Graphs.* In: Cafieri, Sonia, Antonio Mucherino, Giacomo Nannicini, Fabien Tarissan und Leo Liberti (Herausgeber): *Proceedings of the 8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, CTW 2009, Paris, France, June 2-4 2009*, Seiten 351–354, 2009.

[24] McCormick, S. Thomas, M. R. Rao und Giovanni Rinaldi: *Easy and difficult objective functions for max cut.* Mathematical Programming, 94(2-3, Ser. B):459–466, 2003.

[25] Mutzel, Petra: *Implementierung und Analyse eines Max-Cut Algorithmus für planare Graphen.* Diplomarbeit, Fakultät für Mathematik, Universität Augsburg, Augsburg, 1990.

[26] Mutzel, Petra: *Graphenalgorithmen, Master Vertiefungsvorlesung.* Fakultät für Informatik, TU Dortmund, 2016.

[27] Orlova, GI und Ya G Dorfman: *Finding maximum cut in a graph.* Engineering Cybernetics, 10(3):502–506, 1972.

[28] Reif, John H.: *Minimum s-t Cut of a Planar Undirected Network in O(n log^2(n)) Time.* SIAM J. Comput., 12(1):71–81, 1983.

[29] Ringel, Gerhard: *Ein Sechsfarbenproblem auf der Kugel.* Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, 29(1):107–117, 1965.

[30] Shih, Wei-Kuan, Sun Wu und Yue-Sun Kuo: *Unifying maximum cut and minimum cut of a planar graph.* IEEE Transactions on Computers, 39(5):694–697, 1990.

[31] Stoer, Mechthild und Frank Wagner: *A simple min-cut algorithm.* Journal of the ACM (JACM), 44(4):585–591, 1997.

[32] Thomassen, Carsten: *Kuratowski's theorem.* Journal of Graph Theory, 5(3):225–241, 1981.

[33] Wagner, K.: *Über eine Eigenschaft der ebenen Komplexe.* Mathematische Annalen, 114(1):570–590, Dec 1937.

Abbildungsverzeichnis

Algorithmenverzeichnis

Symbolverzeichnis

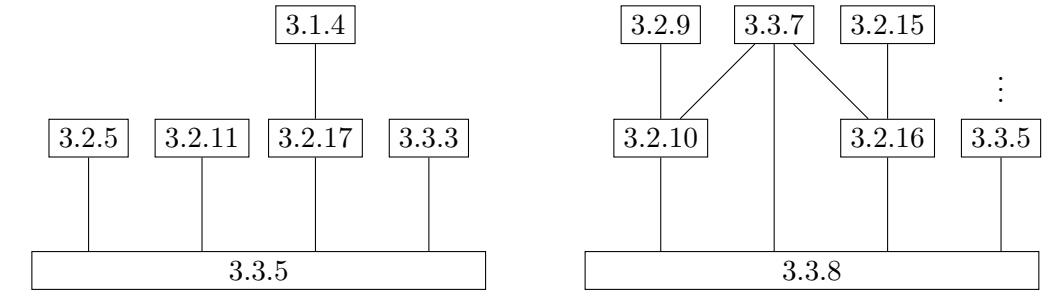
$\mathbb{N}$	Menge aller natürlichen Zahlen ohne die Null
$\mathbb{Q}$	Menge aller rationalen Zahlen
$\mathbb{R}^2$	euklidische Ebene
G	Graph
n	Anzahl der Knoten in einem gegebenen Graphen
m	Anzahl der Kanten in einem gegebenen Graphen
E_G	Kantenmenge des Graphen G
V_G	Knotenmenge des Graphen G
s_{xy}	Strecke zwischen x und y
P_e	Polygonzug zwischen den Endknoten von e
$\mathring{P}_e$	Innere eines Polygonzuges
Γ_G	Zeichnung des Graphen G
$\sigma(\Gamma_G)$	Anzahl an Kantenkreuzungen in der Zeichnung Γ_G
Π_Γ	Einbettung zur Zeichnung Γ
$\mathrm{I}(\Gamma)$	Menge der sich kreuzenden Kanten in der Zeichnung Γ
$G - e$	Kantenlöschung: entsteht durch Löschung der Kante e aus G
G/xy	Kontraktion: entsteht durch Verschmelzen der Knoten x und y in G
F_Γ	Flächenmenge der planaren oder 1-planaren Zeichnung Γ
φ_Γ	Außenfläch der Zeichnung Γ
$\delta(V_1)$	Kantenmenge, die einen Endknoten in der Knotenmenge V_1 hat
$V_1(F)$	Knotenmenge, die den Schnitt F definiert
$\triangle_{i\in I} T_i$	symmetrische Differenz der Mengen $T_i \subseteq M$

Index

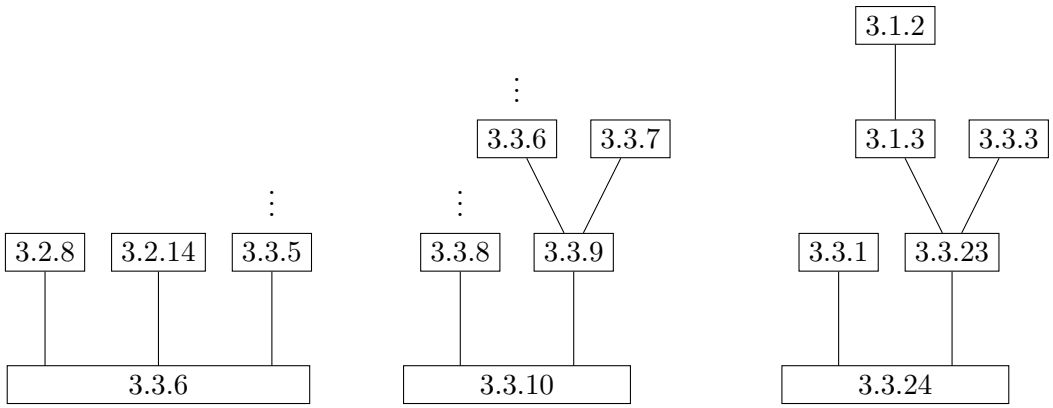
Anhang A

Beweis-Struktur-Graphen

In diesem Kapitel sind Beweis-Struktur-Graphen zu einigen längeren Beweisen der Arbeit gegeben. Diese stellen den Zusammenhang der Lemmata, auf denen ein Beweis basiert, graphisch dar.

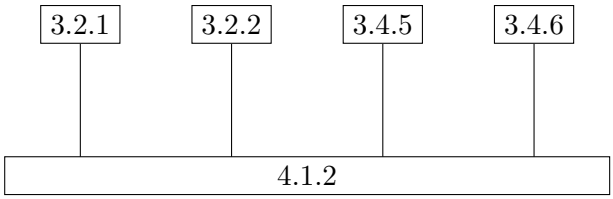


(a) Für Lemma 3.3.5 **(b)** Für Lemma 3.3.8

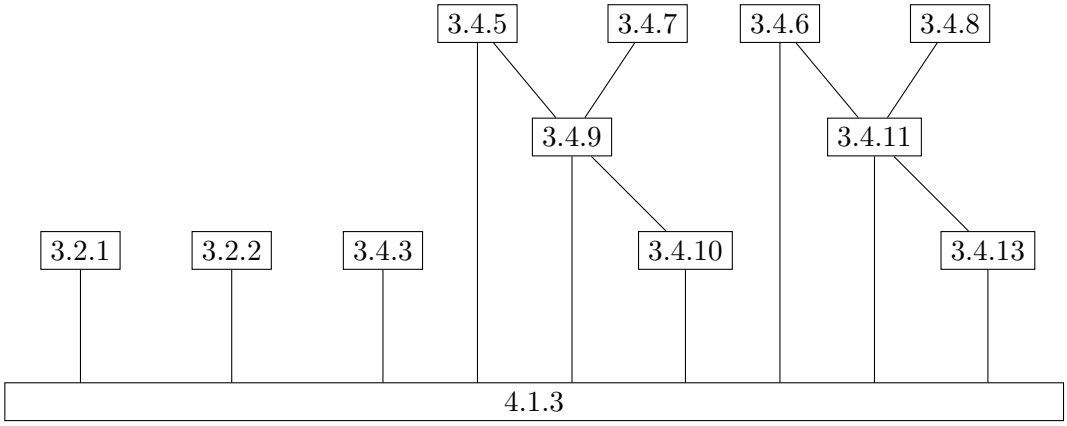


(c) Für Korollar 3.3.6 **(d)** Für Lemma 3.3.10 **(e)** Für Lemma 3.3.24

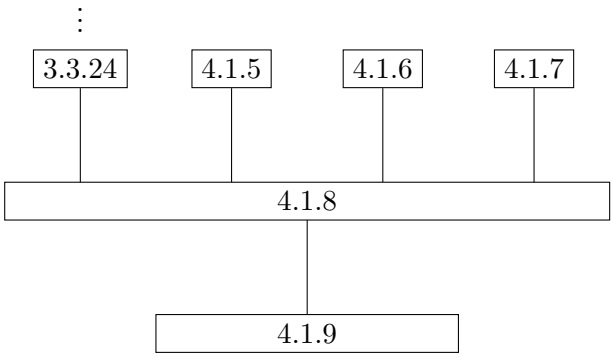
Abbildung A.1: Beweis-Struktur-Graphen für Abschnitt 3.3



(a) Für Satz 4.1.2

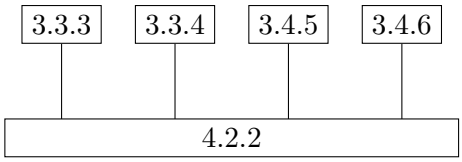


(b) Für Satz 4.1.3

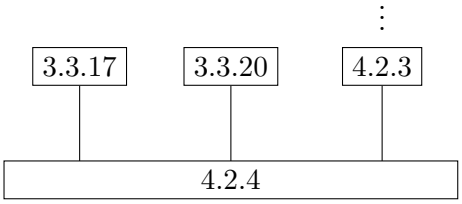


(c) Für Satz 4.1.8 und Theorem 4.1.9

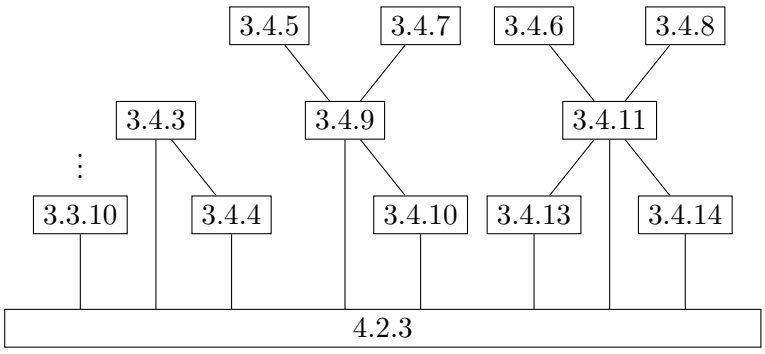
Abbildung A.2: Beweis-Struktur-Graphen für Abschnitt 4.1



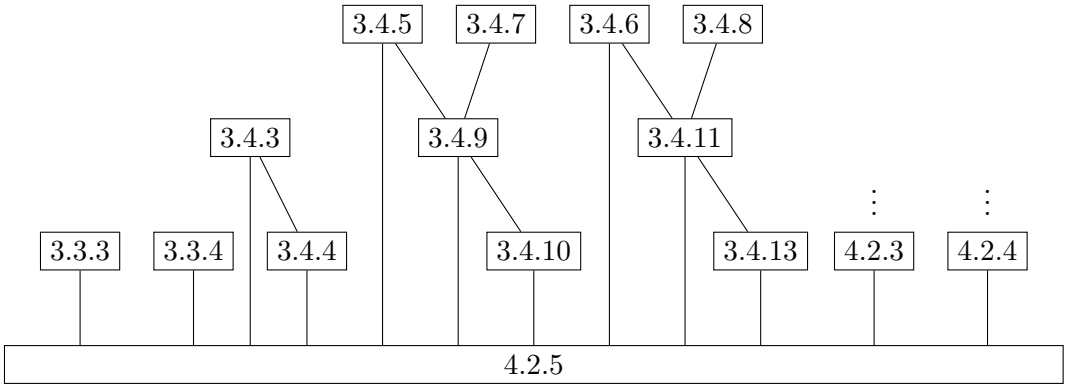
(a) Für Satz 4.2.2



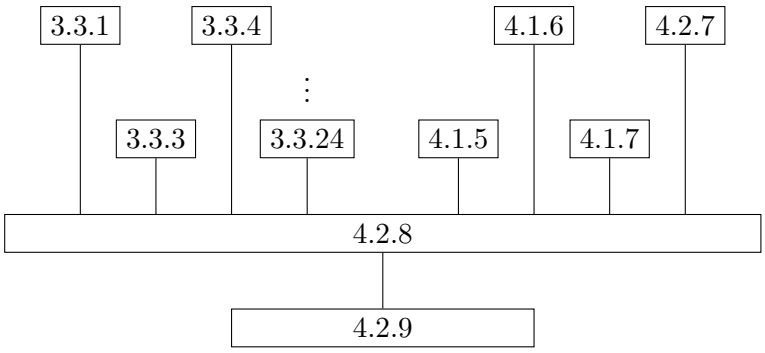
(b) Für Lemma 4.2.4



(c) Für Lemma 4.2.3



(d) Für Satz 4.2.5



(e) Für Satz 4.2.8 und Theorem 4.2.9

Abbildung A.3: Beweis-Struktur-Graphen für Abschnitt 4.2

Eidesstattliche Versicherung

Dahn, Christine	133841
Name, Vorname	Matr.-nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Masterarbeit mit dem Titel

Entwicklung eines Max-Cut-Algorithmus
für fast-planare Graphen

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dortmund, den 25. September 2017	
Ort, Datum	Unterschrift

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/ die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Dortmund, den 25. September 2017	
Ort, Datum	Unterschrift