Algorithmen zur Visualisierung von Schnitten in Graphen

Diplomarbeit von Christian Fieß [1] [2]
Fachbereich für Mathematik und Statistik
Universität Konstanz

8. Januar 2001

[1]WWW: http://www.fmi.uni-konstanz.de/~fiess
[2]E-Mail: Christian@Fiess.com

Zusammenfassung

Minimale Schnitte stellen ein gut studiertes Gebiet der Graphentheorie dar. Wir beschränken uns in der Hauptsache auf planare Graphen und entwickeln ein Verfahren, mit dem die minimalen Schnitte visualisiert werden können. Wir werden sehen, dass sich dieses Problem auf das Zeichnen eines geclusterten Graphen zurückführen lässt. Um eine übersichtliche Darstellung zu gewährleisten, verwenden wir eine orthogonale Gittereinbettung des Graphen mit minimaler Anzahl von Knicken und geben eine Heuristik an, die die wiederholte Darstellung minimaler Schnittes verhindert.

Danksagung

Ich möchte mich bei Dorothea Wagner und besonders bei Sabine Cornelsen für die Betreuung und Unterstützung bedanken.

Inhaltsverzeichnis

Einleitung

Ein Graph besteht aus Knoten und Kanten, die Verbindungen zwischen den Knoten darstellen. Möchte man den Zusammenhang zwischen zwei Knoten im Vergleich zu anderen Knoten hervorheben, so gibt man der Kante, die sie verbindet, ein Gewicht. Den Begriff des Zusammenhangs kann man nun auf Teilmengen der Knotenmenge übertragen: der Zusammenhang zweier disjunkter Teilmengen der Knotenmenge ist gleich der Summe der Gewichte der Kanten, die die beiden Teilmengen verbinden. Somit ist der Zusammenhang zweier disjunkter Teilmengen durch die Definition der Gewichte bestimmt. Uns interessieren nun die Partitionen der Knotenmenge, also zwei nichtleere disjunkte Teilmengen, die zusammen alle Knoten enthalten, so dass deren Zusammenhang am kleinsten ist. Diese Partitionen nennen wir *minimale Schnitte*. Wir werden sehen, dass sich die minimalen Schnitte eines Graphen, in einem *Kaktus*, darstellen lassen. Das ist ein Graph, der als ein Baum von Kreisen aufgefasst werden kann. Wir sprechen dann von der *Kaktusdarstellung* der minimalen Schnitte eines Graphen.

Wir werden Graphen behandeln, die sich in der Ebene zeichnen lassen, d. h. Knoten werden durch Punkte dargestellt und Kanten durch Linien, die zwei Punkte verbinden, so dass sich keine zwei Linien überkreuzen. Die Frage die sich nun stellt, lautet:

> „Wenn man einen Graphen auf diese Art visualisieren kann, ist es dann auch möglich, seine minimalen Schnitte zu visualisieren?“

Die Antwort ist „ja“.

Wir werden sehen, wie man verschiedene Ergebnisse aus der Graphentheorie verwendet, um dies zu zeigen. Dazu gehören [12], [13], [18] und [9], die grundlegende Ergebnisse zur Erstellung einer Kaktusdarstellung geliefert haben. Wir werden in Kapitel 1 und Kapitel 2 darauf eingehen. Dort werden wir auch auf die Ergebnisse aus [8] und [20] eingehen, die für die Visualisierung unerlässlich sind. In Kapitel 3 zeigen wir, wie die minimalen Schnitte eines Graphen visualisiert, dessen Kaktusdarstellung eine spezielle Struktur hat. Wir werden dann in Kapitel 4 darauf aufbauend ein Verfahren beschreiben, das dieser Einschränkung nicht unterliegt.

In Kapitel 5 entwickeln wir eine Heuristik, die zum Ziel hat, die minimalen Schnitte leichter aus der Zeichnung zu erkennen. Wir untersuchen in Kapitel 6

Graphklassen daraufhin, ob sie die Voraussetzungen des in Kapitel 3 vorgestellten Algorithmus erfüllen. In Kapitel 7 ist eine Zusammenfassung gegeben und ein Ausblick, der mögliche Verallgemeinerungen betrachtet.

Kapitel 1

Notation

Im Folgenden verstehen wir unter einem *gewichteten ungerichteten Graphen* G das Tripel $G = (V, E, c)$. Dabei bezeichnen $V = \{v_1, \ldots, v_n\}$ die Knotenmenge, $E \subseteq \{\{u, w\} \mid u, w \in V\}$ die Kantenmenge und $c : E \to \mathbb{R}^+$ die Gewichtsfunktion. Desweiteren setzen wir voraus, dass G zusammenhängend ist und keine *Schleifen*, d. h. Kanten der Form $\{v, v\}$ für ein $v \in V$, enthält. Besitzt G keine Schleifen und werden zwei Knoten durch höchstens eine Kante verbunden, so heißt G *einfach*.

Ein Graph $G = (V, E)$ heißt *planar*, wenn es eine Einbettung in die Ebene gibt, so dass sich keine Linien kreuzen. Dabei verstehen wir unter einer Einbettung eine Abbildung $P : V \to \mathbb{R}^2$ der Knoten auf Punkte der Ebene und eine Abbildung $K : E \to C([0, 1], \mathbb{R}^2)$ der Kanten auf stetige doppelpunktfreie Kurven der Ebene, die die Anfangs- bzw. Endpunkte der Kante verbinden, also

$$\begin{array}{rcl} P:\ V & \to & \mathbb{R}^2 \\ v & \mapsto & P(v) = (x, y) \\ K:\ E & \to & C([0, 1], \mathbb{R}^2) \\ \{v, w\} & \mapsto & J \text{ mit } J(0) = P(v) \text{ und } J(1) = P(w) \end{array}$$

Eine Einbettung von G ist also planar, falls P injektiv ist und

$$\forall e_1, e_2 \in E:\ K_{e_1}(]0, 1[) \cap K_{e_2}(]0, 1[) = \emptyset$$

gilt, mit $K_{e_i} = K(e_i)$ für $i = 1, 2$.

Mit $\emptyset \neq X_i \subsetneq V$, für $i = 1, 2$, nennen wir das Paar $\{X_1, X_2\}$ einen *Schnitt* in G, und die Kanten aus

$$E(X_1, X_2; G) = \{\{u, v\} \in E \mid u \in X_1, v \in X_2\}$$

sind genau die Kanten, die X_1 und X_2 verbinden. Ist aus dem Zusammenhang klar, um welchen Graphen es sich handelt, so schreiben wir kurz $E(X_1, X_2)$. Unter dem

Wert c eines Schnittes $\{X_1, X_2\}$ verstehen wir die Summe aller Gewichte der Kanten, die X_1 und X_2 verbinden, also

$$c(X_1, X_2; G) = \sum_{e \in E(X_1, X_2; G)} c(e),$$

oder kurz $c(X_1, X_2)$. Wir werden uns hauptsächlich mit Schnitten beschäftigen, deren Wert minimal ist unter allen Schnitten von G. Den Wert eines *minimalen Schnittes* wollen wir mit

$$\lambda(G) = \min_{\{X, \overline{X}\} \in \mathcal{C}(G)} \{c(X, \overline{X}; G)\}$$

bezeichnen und die Menge der minimalen Schnitte mit

$$\mathcal{C}(G) = \{\{X, \overline{X}\} \mid c(X, \overline{X}; G) = \lambda(G)\}.$$

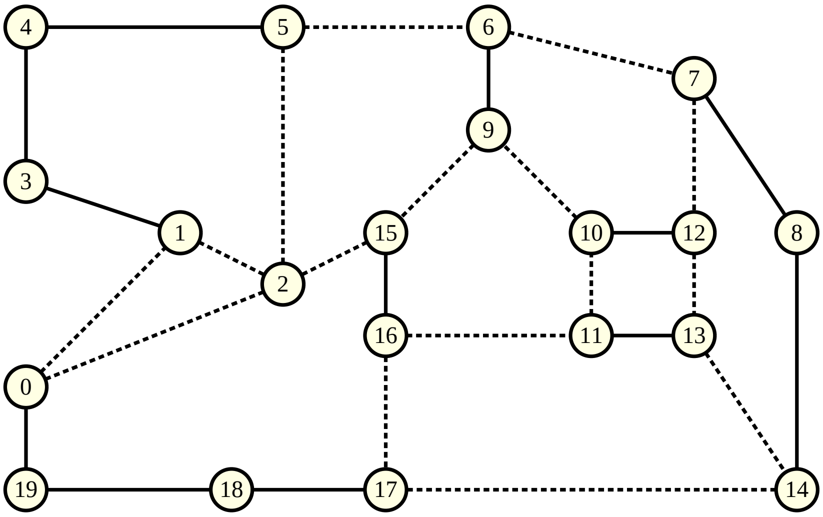


Abbildung 1.1: Graph G mit $\lambda(G) = 4$. Gestrichelte Kanten besitzen Gewicht 1, durchgehende Kanten Gewicht 2.

1.1 Planare Graphen

Planare Graphen besitzen eine Reihe von Eigenschaften, die wir im weiteren Verlauf verwenden werden. Eine planare Einbettung eines Graphen teilt die Ebene in topologisch zusammenhängende Gebiete, die wir *Facetten* nennen. Das unbeschränkte

Gebiet heißt *äußere Facette*. Unter einer Zusammenhangskomponente verstehen wir eine maximale Teilmenge der Knotenmenge eines Graphen, so dass es zu je zwei Knoten einen Pfad gibt, der sie verbindet. Das folgende Lemma stellt eine Verbindung zwischen der Anzahl der Knoten, Kanten, Facetten und Zusammenhangskomponenten dar.

Lemma 1.1 (Satz von Euler) *Für einen einfachen, planaren Graphen $G = (V, E)$ mit n Knoten, m Kanten, f Facetten und k Zusammenhangskomponenten gilt*

$$n - m + f = 1 + k \tag{1.1}$$

Beweis. Per Induktion nach $m = |E|$.

$m = 0$ Sind keine Kanten vorhanden, so gibt es nur eine Facette und jeder Knoten stellt eine Zusammenhangskomponente dar. Daraus folgt $f = 1$ und $k = n$.

Sei die Aussage für Graphen mit $|E| < m$ gezeigt.

$m > 0$ Sei $e \in E$ und $G' = (V, E - \{e\})$ der Graph, der entsteht indem man die Kante e aus G löscht. Weiter seien f', k', m' bzw. n' die Anzahl der Facetten, Zusammenhangskomponenten, Kanten bzw. Knoten von G'. Dabei gilt $m' = m - 1$ und $n' = n$. Wegen $m' < m$ gilt nach Induktionsvoraussetzung $n' - m' + f' = 1 + k'$. Es sind die folgenden zwei Fälle zu unterscheiden.

1. Fall e liegt auf einem einfachen Kreis. Dann gilt $k = k'$ und die beiden Facetten, die zu e benachbart sind, werden beim Löschen von e vereinigt. Daraus folgt $f' = f - 1$.
2. Fall e liegt nicht auf einem einfachen Kreis. Es entsteht beim Löschen von e eine zusätzliche Zusammenhangskomponente, also $k' = k + 1$ und $f' = f$.

□

Lemma 1.2 *Sei $G = (V, E)$ ein planarer Graph mit $V = \{v_1, \ldots, v_n\}$. Desweiteren sei $\Delta(G)$ der Maximalgrad in G und n_i die Anzahl der Knoten mit Grad genau i. Dann gilt*

$$6n_0 + 5n_1 + 4n_2 + 3n_3 + 2n_4 + n_5 \quad \geq \quad n_7 + \cdots + (\Delta(G) - 6)n_{\Delta(G)} + 12 \tag{1.2}$$

Beweis. Der Beweis von Lemma 1.2 verwendet die Beziehungen

$$n = \sum_{i=0}^{\Delta(G)} n_i \quad \text{und} \quad 2m = \sum_{i=0}^{\Delta(G)} i\ n_i.$$

Sei f die Anzahl der Facetten und k die Anzahl der Zusammenhangskomponenten von G, so folgt $n-m+f = k+1 \geq 2$ aus der Euler–Formel 1.1. Die Zahl der Facetten kann durch die Zahl der Kanten abgeschätzt werden. Und zwar wird jede Facette von mindestens drei Kanten begrenzt und jede Kante gehört zu zwei Facetten. Somit gilt $f \leq \frac{2}{3}\,m$. Mit Äquivalenzumformungen erhält man $6n = 6m-6f+12 \geq 2m+12$ und mit den oben genannten Beziehungen schließlich

$$6\sum_{i=0}^{\Delta(G)} n_i \;\geq\; \sum_{i=0}^{\Delta(G)} i\; n_i \;+12.$$

Formel (1.2) ergibt sich durch Subtraktion von $6\sum_{i=6}^{\Delta(G)} n_i$ und $\sum_{i=0}^{5} i\; n_i$ auf beiden Seiten. □

Daraus ergeben sich die beiden folgenden Korollare.

Korollar 1.3 *Ein planarer Graph besitzt mindestens einen Knoten von Grad höchstens fünf.*

Beweis. Nimmt man an, dass es nur Knoten mit Grad größer 6 gibt, d.h.

$$\sum\nolimits_{i=0}^{5}(6-i)\; n_i = 0,$$

so erhält man den Widerspruch $0 \geq 12$. □

Korollar 1.4 *Der Wert eines minimalen Schnittes eines planaren ungewichteten Graphen ist höchstens fünf.*

1.2 Darstellung minimaler Schnitte

In diesem Abschnitt werden wir eine Darstellung der minimalen Schnitte eines Graphen kennen lernen. Ziel ist es, die minimalen Schnitte eines Graphen $G = (V, E)$, von denen es $\mathcal{O}(|V|^2)$ viele gibt, in einem geeigneten Graphen $R = (W, F)$ darzustellen, dessen Größe linear in $|V|$ ist, und aus dem die minimalen Schnitte einfach zu rekonstruieren sind.

Das nächste Lemma zeigt, dass ein minimaler Schnitt eine Partition von V in zwei zusammenhängende Mengen darstellt.

Lemma 1.5 *Sei G ein Graph und $\mathcal{C} \subseteq \mathcal{C}(G)$ eine Menge von minimalen Schnitten in G und $\{X, \overline{X}\} \in \mathcal{C}$. Dann sind X und $\overline{X}$ zusammenhängend.*

Beweis. Angenommen, das Lemma gilt nicht. Das heißt, es gibt einen minimalen Schnitt $\{X, \overline{X}\} \in \mathcal{C}$ wobei (o.B.d.A.) X nicht zusammenhängend ist — d.h. $X = X_1 \dot{\cup} X_2$ und $c(X_1, X_2) = 0$. Seien nun $\alpha = c(X_1, \overline{X})$ und $\beta = c(X_2, \overline{X})$. Es gilt $\alpha + \beta = \lambda(G)$ und $\alpha, \beta \neq 0$, da sonst G schon unzusammenhängend wäre. Daraus folgt $\alpha, \beta < \lambda(G)$, also die Partitionen $\{X_1, \overline{X} \cup X_2\}$ und $\{X_2, X_1 \cup \overline{X}\}$ implizieren einen Schnitt mit kleinerem Wert als dem als minimal angenommenen. Widerspruch. □

Definition 1.1 Ein Graph heißt *Kaktus*, falls je zwei Kreise höchstens einen Knoten gemeinsam haben. Ein Graph der Form $(\{v\}, \emptyset, c)$ heißt *trivialer* Kaktus. Eine Kante, die zu einem Kreis gehört heißt Kreiskante, sonst Baumkante. Ein Kaktus heißt *uniform*, falls eine Konstante $\lambda > 0$ existiert mit $c(e) = \lambda/2$ für jede Kreiskante und $c(e) = \lambda$ für jede Baumkante.

Bemerkung Die folgenden Klassen von Graphen sind Kakteen: Pfade, Bäume, Kreise, Sterne.

Ist G ein zusammenhängender, nicht trivialer uniformer Kaktus, so gilt für den Wert eines minimalen Schnittes $\lambda(G) = \lambda$ und jeder minimale Schnitt besteht entweder aus einer Baumkante oder aus zwei Kreiskanten, die zu demselben Kreis gehören.

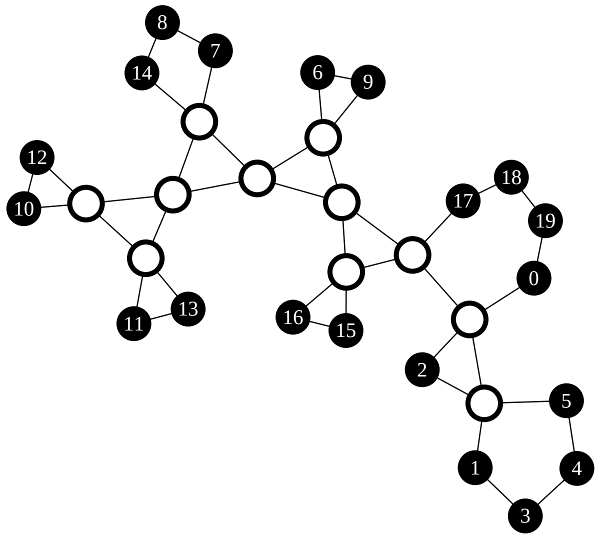


Abbildung 1.2: Eine normale Kaktusdarstellung für $\mathcal{C}(G)$. Leere Knoten sind weiß gezeichnet, nicht leere Knoten tragen die Nummer des Knotens, den sie darstellen. Dies ist ein spezieller Fall, in dem nicht leere Knoten im Kaktus genau einen Knoten des Graphen darstellen. Alle Kanten besitzen Gewicht 2.

Definition 1.2 Ein Schnitt $\{X, \overline{X}\}$ eines Graphen *kreuzt* einen anderen Schnitt $\{Y, \overline{Y}\}$ in G, falls die folgenden Schnittmengen nicht leer sind:

$$X \cap Y \neq \emptyset, \quad \overline{X} \cap Y \neq \emptyset, \quad X \cap \overline{Y} \neq \emptyset, \quad \overline{X} \cap \overline{Y} \neq \emptyset.$$

Zwei verschiedene Schnitte von G heißen *komplementär*, falls sie sich nicht kreuzen, also mindestens eine der Schnittmengen leer ist.

Nun wollen wir festlegen, was wir unter einer Darstellung minimaler Schnitte verstehen. Sei $G = (V, E, c)$ ein Graph, so wird G in einen anderen Graphen $R = (W, F, d)$ durch $\varphi : V \to W$ abgebildet.

Es seien

$$\begin{aligned} \varphi(X) &= \{\varphi(v) \in W \mid v \in X\} \quad \text{für } X \subseteq V, \\ \varphi^{-1}(S) &= \{v \in V \mid \varphi(v) \in S\} \quad \text{für } S \subseteq W \end{aligned}$$

und $\mathcal{C}(G)$ bzw. $\mathcal{C}(R)$ bezeichnen die Menge aller minimalen Schnitte in G bzw. R. Im Folgenden wird für $X \in V$ bzw. $Y \in W$ das Komplement mit $\overline{X} = V - X$ bzw. mit $\widetilde{Y} = W - Y$ bezeichnet. Sei $\mathcal{C} \subset \mathcal{C}(G)$ eine Menge von minimalen Schnitten von G.

Definition 1.3 $(\mathcal{R}, \varphi)$ heißt *Darstellung* von $\mathcal{C}$, falls sie folgende Eigenschaften erfüllt:

1. Für einen beliebigen minimalen Schnitt $\{S, \widetilde{S}\} \in \mathcal{C}(\mathcal{R})$ gilt

   $$\{\varphi^{-1}(S), \varphi^{-1}(\widetilde{S})\} \in \mathcal{C}.$$

2. Umgekehrt gibt es zu jedem minimalen Schnitt $\{X, \overline{X}\} \in \mathcal{C}$ einen minimalen Schnitt $\{S, \widetilde{S}\} \in \mathcal{C}(\mathcal{R})$ mit $S \supseteq \varphi(X)$ und $\widetilde{S} \supseteq \varphi(\overline{X})$.

Desweiteren *korrespondieren* zwei Schnitte $\{X, \overline{X}\} \in \mathcal{C}$ und $\{S, \widetilde{S}\} \in \mathcal{C}(\mathcal{R})$, falls $\varphi(X) \subseteq S$ und $\varphi(\overline{X}) \subseteq \widetilde{S}$. Ein Knoten $x \in W$ heißt *leer*, falls $\varphi^{-1}(x) = \emptyset$.

Bemerkung

(a) Mit der Definition

$$\varphi^{-1}(\mathcal{C}') \equiv \{\{\varphi^{-1}(S), \varphi^{-1}(\widetilde{S})\} \mid \{S, \widetilde{S}\} \in \mathcal{C}'\} \quad \text{für } \mathcal{C}' \subseteq \mathcal{C}(\mathcal{R})$$

kann man die Bedingungen (i) und (ii) zusammenfassen zu

$$\varphi^{-1}(\mathcal{C}(\mathcal{R})) = \mathcal{C}.$$

(b) Besitzt $\mathcal{R}$ einen leeren Knoten, so kann ein minimaler Schnitt in $\mathcal{C}$ mit mehr als einem minimalen Schnitt in $\mathcal{C}(\mathcal{R})$ korrespondieren. Umgekehrt korrespondiert ein minimaler Schnitt in $\mathcal{C}(\mathcal{R})$ immer mit genau einem minimalen Schnitt in $\mathcal{C}$. Daraus folgt, dass es keinen minimalen Schnitt $\{S, \widetilde{S}\}$ in $\mathcal{C}(\mathcal{R})$ geben kann, wenn S nur aus leeren Knoten besteht (denn der dazu korrespondierende Schnitt in $\mathcal{C}$ wäre $\{\emptyset, V\}$ — dies ist aber kein gültiger Schnitt).

Nun wollen wir eine Darstellung minimaler Schnitte konstruieren. Dieses Beispiel soll die eingeführten Begriffe verdeutlichen. In Kapitel 2 werden wir ein Verfahren kennen lernen, das eine Kaktusdarstellung für alle minimalen Schnitte eines Graphen erstellt. Es sei $\pi = \{V_1, \ldots, V_r\}$ eine Partition von V, so dass für $i = 1, 2, \ldots, r$ gilt

$$\lambda(u, u'; G) > \lambda(G) \quad \text{für alle } u, u' \in V_i.$$

Für einen minimalen Schnitt in $\{X, \overline{X}\} \in \mathcal{C}(R)$ bedeutet das, dass ein V_i entweder ganz in X oder ganz in $\overline{X}$ enthalten ist. Sei nun $R = (U, E', c')$ der gewichtete Graph, der aus G entsteht, indem man die Mengen V_i jeweils zu unterschiedlichen Knoten $u_i \in U$ verschmilzt (man lösche auftretende Schlaufen). Die Abbildung $\varphi : V \to U$ wird definiert durch $\varphi(x) = u_i$ für alle $x \in V_i$, $i = 1, \ldots, r$. Nun sieht man, dass alle minimalen Schnitte von G in R erhalten bleiben. Desweiteren gibt es keinen minimalen Schnitt in R, der nicht mit einem minimalen Schnitt in G korrespondiert. Somit ist $\mathcal{R} = (R, \varphi)$ eine Darstellung für $\mathcal{C}(G)$. Die Partition π kann maximal gewählt werden, also falls für $X \subset V$

$$\lambda(u, u'; G) > \lambda(G) \;\; \text{für alle } \; u, u' \in X$$

gilt, so gibt es ein $i \in \{1, \ldots, r\}$ mit $X \subseteq V_i$. Leider weiß man nicht viel über die Darstellung (R, φ) von $\mathcal{C}(G)$. Lediglich für den Fall, dass die Kanten von G Gewicht $c = 1$ haben und $\lambda(G) = 2$ ist, wurde in gezeigt, dass R ein uniformer Kaktus mit $\lambda(R) = 2$ ist [18]. Desweiteren ist bekannt, dass es für einen beliebigen gewichteten Graphen $G = (V, E, c)$ eine Darstellung (R, φ) für $\mathcal{C}(G)$ gibt, wobei $R = (W, F)$ ein uniformer Kaktus ist und $|W| + |F| = \mathcal{O}(|V|)$ [10, 15, 5]. Deshalb werden wir im Folgenden mit dem Ausdruck *Kaktusdarstellung* $(\mathcal{R}, \varphi)$ immer von einem uniformen Kaktus ausgehen. Zu bemerken ist, dass die Existenz einer Kaktusdarstellung für $\mathcal{C}(G)$ keine Garantie für eine Kaktusdarstellung einer Teilmenge $\mathcal{C} \subsetneq \mathcal{C}(G)$ darstellt.

Für das nächste Lemma benötigen wir den Begriff der Kuchenpartition:

Definition 1.4 Unter einer *Kuchenpartition* von V der *Größe r über $\mathcal{C}(G)$* verstehen wir eine geordnete Menge $\pi = (V_1, V_2, \ldots, V_r)$ mit

$$V_i \neq \emptyset, V_i \cap V_j = \emptyset \text{ für } 1 \leq i < j \leq r, \;\text{ und } \bigcup_{i=1}^{r} V_i = V,$$

und

$$\{\{X,\overline{X}\} \mid X = \bigcup_{i=k}^{l} V_i \text{ mit } 1 \leq k \leq l \leq r-1\} \subseteq \mathcal{C}(G).$$

π heißt *maximale Kuchenpartition*, falls für jedes $i = 1, \ldots, r$ und $X \subsetneq V_i$ die Partition

$$(V_1, \ldots, V_{i-1}, X, V_i - X, V_{i+1}, \ldots, V_r)$$

keine Kuchenpartition ist.

Es handelt sich also um disjunkte, nichtleere Mengen, so dass sukzessive Vereinigungen minimale Schnitte ergeben.

Definition 1.5 Ein Knoten v heißt *k–Kreuzungsknoten*, falls v genau zu r Kreisen und s Baumkanten gehört, und $k = r + s$ gilt. Ein Kaktus R heißt *normal*, falls keine leeren 2–Kreuzungsknoten hat, die zu einer Baumkante inzident sind. Eine Kaktusdarstellung $\mathcal{R} = (R, \varphi)$ heißt *normal*, falls R normal ist. (s. Abbildung 1.2)

Sei C ein Kreis in einem Kaktus $R = (W, F)$. $W[C]$ und $F[C]$ bezeichnen die Knoten- bzw. Kantenmenge von C. Der Graph $R - F[C]$, der aus R durch entfernen der Kanten von C entsteht, zerfällt in $r = |C|$ Komponenten (W_i, F_i). Mit Hilfe von Abbildung 1.3 sieht man leicht, dass $\pi_C = (V_1, V_2, \ldots, V_r)$ (mit $V_i = \varphi^{-1}(W_i)$ für

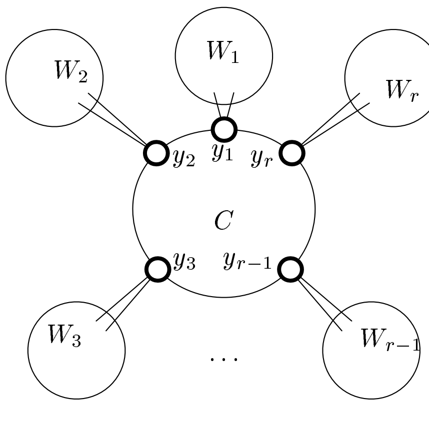


Abbildung 1.3: Ein Kreis in einer normalen Kaktusdarstellung

$i = 1, 2, \ldots, r$) eine Kuchenpartition von V über $\mathcal{C}$ darstellt. Wir sagen, $\mathcal{C}$ *erzeugt* π_C. Betrachten wir nun die beiden Situationen in Abbildung 1.4. Sowohl der Kreis der Länge drei wie auch der leere 3–Kreuzungsknoten erzeugt eine Kuchenpartition der Größe drei. Daran sieht man, dass ein Kaktus, der Kreise der Länge drei oder leere 3–Kreuzungsknoten enthält, nicht eindeutig ist. Die minimalen Schnitte $\{W_i \cup y_i, \widetilde{W_i \cup y_i}\}$, $i \in \{1, 2, 3\}$, sind komplementär zueinander.

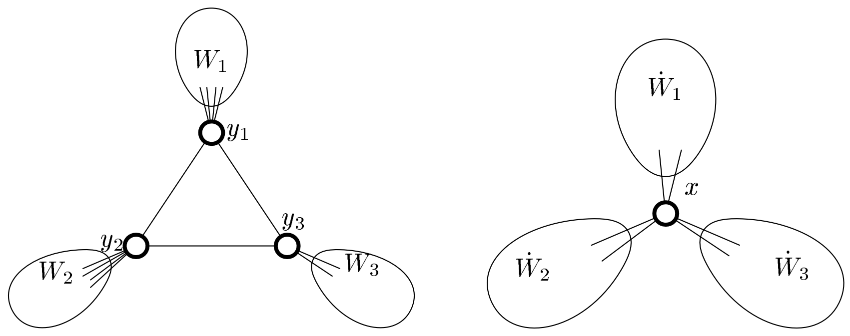


Abbildung 1.4: Beide Subgraphen erzeugen eine Kuchenpartition der Größe drei

Für den Fall sich kreuzender Schnitte können wir folgendes Lemma festhalten.

Lemma 1.6 *Sei $G = (V, E, c)$ ein gewichteter Graph, $\{X, \overline{X}\}, \{Y, \overline{Y}\} \in \mathcal{C}(G)$ sich kreuzende Schnitte, $V_1 = X \cap Y, V_2 = \overline{X} \cap Y, V_3 = X \cap \overline{Y}, V_4 = \overline{X} \cap \overline{Y}$, dann gilt:*

(i) $c(V_1, V_2) = c(V_2, V_3) = c(V_3, V_4) = c(V_4, V_1) = \lambda/2$

(ii) $c(V_1, V_3) = c(V_2, V_4) = 0$

(iii) (V_1, V_2, V_3, V_4) *ist eine Kuchenpartition von V über $C(G)$.*

Beweis. Es gilt

$$\lambda = c(X, \overline{X}) = c(V_1, V_2) + c(V_3, V_4) + \mu_1 + \mu_2,$$

mit $\mu_1 = c(V_1, V_3)$ und $\mu_2 = c(V_2, V_4)$. Sei z.B. $c(V_1, V_2) > c(V_2, V_3)$. Dann würde man mit $\{V_1 \cup V_2 \cup V_4, V_3\}$ einen Schnitt mit dem Gewicht

$$c(V_1 \cup V_2 \cup V_4, V_3) = c(V_2, V_3) + c(V_3, V_4) + \mu_1 =: \lambda^*$$

erhalten. Dieser Schnitt hätte aber wegen $\lambda - \lambda^* = c(V_1, V_2) - c(V_2, V_3) + \mu_2 > 0$ ein kleineres Gewicht als die als minimal vorausgesetzten Schnitte $\{X, \overline{X}\}$ und $\{Y, \overline{Y}\}$. Widerspruch! Analog führt man Annahmen, die weiteren Gleichungen würden nicht gelten, zum Widerspruch.

Mit derselben Argumentation führt die Annahmen $\mu_1 > 0$ und $\mu_2 > 0$ zum Widerspruch. (iii) folgt aus der Definition einer Kuchenpartition. □

Das nächste Lemma hält die Beziehung zwischen maximalen Kuchenpartitionen des Graphen und Kreisen im Kaktus fest.

Lemma 1.7 *Sei $G = (V, E, c)$ Graph, $\mathcal{R} = (R, \varphi)$ kreisartige normale Kaktusdarstellung für $\mathcal{C}(G)$ mit $R = (W, F)$.*

- *Ist $\pi = (V_1, \ldots, V_r)$ eine maximale Kuchenpartition über $\mathcal{C}(G)$ der Größe $r \geq 3$, dann besitzt R einen Kreis der Länge r, der π erzeugt.*
- *Für jeden Kreis C, der Länge $r \geq 3$ in R ist die durch C gegebene Partition $\pi_C = (V_1, \ldots, V_r)$ eine maximale Partition über $\mathcal{C}(G)$.*

Wir wollen nun die Menge der Kuchepartitionen der Größe 3 von G mit $\Pi^3(\mathcal{C}(G))$ bezeichnen. Der folgende Satz besagt, dass eine normale Kaktusdarstellung der minimalen Schnitte von G nach Festlegung der Darstellung von Kuchenpartitionen der Größe 3 eindeutig ist.

Satz 1.1 ([18]) *Sei G ein gewichteter Graph ohne Schleifen, sowie $\mathcal{R}_1$ und $\mathcal{R}_2$ normale Kaktusdarstellungen für $\mathcal{C}(G)$, so dass jedes $\pi \in \Pi^3(\mathcal{C}(G))$ genau dann von einem Kreis der Länge drei in $\mathcal{R}_1$ erzeugt wird, wenn π von einem Kreis der Länge drei in $\mathcal{R}_2$ erzeugt wird. Sei $\mathcal{R}_1 = (R_1, varphi_1)$ und $\mathcal{R}_2 = (R_2, \varphi_2)$ mit $R_1 = (W_1, F_1)$ und $R_1 = (W_1, F_1)$. Dann gibt es eine Bijektion $\alpha : W_1 \to W_2$ mit*

(i) $\varphi^{-1}(x) = \varphi^{-1}(\alpha(x))$

(ii) $(x, x') \in F_1 \Longleftrightarrow (\alpha(x), \alpha(x')) \in F_2$

Abschließend wollen wir noch zwei Kaktusdarstellungen auszeichnen. Ist $\mathcal{R} = (R, \varphi)$ eine Kaktusdarstellung, so dass R keine Kreise der Länge 3 enthält, so heißt sie *kreuzungsartige Kaktusdarstellung*. Enthält R keine leeren 3–Kreuzungsknoten, so ist $\mathcal{R}$ eine *kreisartige Kaktusdarstellung*. Aus Satz 1.1 folgt, dass kreuzungsartige sowie kreisartige Kaktusdarstellungen eindeutig sind.

1.3 Hierarchische Clustergraphen

Feng et al. haben 1995 den Begriff des hierarchisch geclusterten Graphen eingeführt und Graphen charakterisiert, die in Bezug auf die Clusterung eine planare Zeichnung zulassen. Hier wollen wir die Definitionen und Ergebnisse aus [8] zusammenfassen.

Definition 1.6 (Clustergraph) Seien $G = (V, E)$ ein Graph und $T = (W, F)$ ein Baum mit Wurzel dessen Blätter genau die Knoten in V sind. Wir bezeichnen das Paar $C = (G, T)$ als *Clustergraph*. Jeder Knoten $\nu \in W - V$ repräsentiert einen *Cluster*. Die Menge $V(\nu)$ der Knoten, die zu dem Cluster ν gehören, sind genau die Blätter im Unterbaum mit Wurzel ν in T. Der Baum T legt die so genannte *Clusterhierarchie* fest. Die Wurzel von T wird *Root-Cluster* genannt. Der Graph G

ist der dem Clustergraphen C *zugrundeliegende* Graph. Eine Kante $\{v, w\} \in E$ ist *inzident* zu einem Cluster $\nu \in T$, falls entweder $v \in V(\nu)$ oder $w \in V(\nu)$.

Ein Clustergraph heißt *zusammenhängend*, falls für jeden Cluster $\nu \in W - V$ der von $V(\nu)$ induzierte Subgraph von G zusammenhängend ist.

Definition 1.7 (Clusterzeichnung) Eine Zeichnung $\mathcal{Z}$ eines Clustergraphen $C = (G, T)$ besteht aus der Zeichnung des zugrundeliegenden Graphen $G = (V, E)$ und der Zeichnung der Clusterhierarchie $T = (W, F)$ in der Ebene. Jeder Knoten $v \in V$ wird durch einen Punkt $\mathcal{Z}(v)$ repräsentiert und jede Kante $\{v, w\} \in E$ wird durch eine einfache Kurve $\mathcal{Z}(e)$ zwischen $\mathcal{Z}(v)$ und $\mathcal{Z}(w)$ dargestellt. Jeder Cluster $\nu \in W - V$ wird durch ein einfaches abgeschlossenes Gebiet dargestellt, das von einer einfachen geschlossenen Kurve $\partial Z(\nu)$ begrenzt wird, so dass gilt

- $\mathcal{Z}(\mu) \subset \mathcal{Z}(\nu)$ für alle Nachfolger μ von ν,
- $\mathcal{Z}(\mu) \cap \mathcal{Z}(\nu) = \emptyset$ falls μ weder Vorgänger noch Nachfolger von ν ist,
- $\mathcal{Z}(e) \subset \mathcal{Z}(\nu)$ für alle $e = \{v, w\} \in E$ mit $v, w \in V(\nu)$ und
- $\mathcal{Z}(e) \cap \mathcal{Z}(\nu)$ enthält genau einen Punkt, genau dann, wenn e inzident zu $\mathcal{Z}(\nu)$ ist.

Eine Clusterzeichnung $\mathcal{Z}$ von $C = (G, T)$ besitzt eine *Überkreuzung*, falls es eine Kante $e = \{v, w\} \in E$ und einen Cluster $\nu \in T$ gibt, so dass $v, w \notin V(\nu)$ aber $\mathcal{Z}(e) \cap \mathcal{Z}(\nu) \neq \emptyset$. Eine Clusterzeichnung $\mathcal{Z}$ heißt *c–planar*, falls sich keine Kanten kreuzen und $\mathcal{Z}$ keine Überkreuzung besitzt. Weiter ist ein Clustergraph *c–planar*, falls er eine c–planare Zeichnung besitzt.

Der folgende Satz besagt, wann ein Clustergraph eine c–planare Zeichnung besitzt.

Satz 1.2 ([8]) *Ein zusammenhängender Clustergraph $C = (G, T)$ ist c–planar genau dann, wenn es eine planare Zeichnung von G gibt, so dass für jeden Cluster ν von T alle Knoten in $V - V(\nu)$ in der äußeren Facette der Zeichnung von $G(\nu)$ liegen.*

Im weiteren Verlauf interessieren wir uns für eine spezielle Art von Clusterzeichnungen. Dabei handelt es sich um eine *orthogonale Gittereinbettung mit rechteckigen Clustern (OGRC)*: die Knoten von G liegen auf Gitterpunkten und Kanten werden als eine Folge von senkrechten und waagerechten Segmenten dargestellt und jeder Cluster $\nu \in T$ wird durch ein achsenparalleles Rechteck dargestellt. Der nächste Satz beschäftigt sich mit der Machbarkeit einer OGRC–Zeichnung.

Satz 1.3 ([8]) *Für einen c–planaren Clustergraphen mit n Knoten und Maximalgrad 4, kann eine c–planare OGRC–Zeichnung mit $\mathcal{O}(n^2)$ Platzbedarf und höchstens 3 Knicke pro Kante in $\mathcal{O}(n)$ Zeit erstellt werden.*

Der Beweis von Satz 1.3 ist konstruktiv und verwendet eine Sichtbarkeitsrepräsentation, die auf einer speziellen Knotennummerierung beruht. Dabei wird der Graph zu einem gerichteten Graphen, so dass Kanten einen Cluster am oberen Rand verlassen oder am unteren Rand in einen Cluster eintreten. Wir werden in Kapitel 3 einen Algorithmus vorstellen, der ohne diese Einschränkung auskommt und zusätzlich die Anzahl der auftretenden Knicke minimiert. Dies muss allerdings mit einer schlechteren Laufzeit bezahlt werden.

1.4 Orthogonale Zeichnung

Hier möchte ich ein Verfahren beschreiben, das für einen planar eingebetteten Graphen mit Maximalgrad 4 eine Gittereinbettung berechnet, so dass die Gesamtzahl der Knicke auf Kanten minimal ist. Dabei sei $G = (V, E)$ der eingebettete Graph und $\mathcal{F}$ die Facettenmenge, wobei $f_0 \in \mathcal{F}$ die äußere Facette ist.

Unter dieser Voraussetzung definieren wir den Begriff der *orthogonalen Beschreibung*. Diese besteht für jede Facette $f_i \in \mathcal{F}$ aus einer Folge von Facettenbeschreibungen der Form (e, δ, x), wobei $e = \{v, w\}$ die Kanten durchläuft, die an f_i grenzen. Dabei ist δ eine Folge aus $\{0, 1\}$. In δ jedes mal eine 0, wenn die Kante e beim Umlaufen der Facette f_i im Uhrzeigersinn einen Linksknick besitzt, bei einem Rechtsknick hingegen eine 1. Der Wert $x \in \{0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi\}$ gibt an, welcher Winkel im Bogenmaß an, der in w zwischen e und der nächsten Kante anliegt. Aus einer orthogonalen Beschreibung kann in $\mathcal{O}(|V|)$ Zeit eine Gittereinbettung, berechnet werden.

Nun ist es die Aufgabe, eine orthogonale Beschreibung der Facetten zu erstellen. Dazu konstruieren wir ein sogenanntes *Flussnetzwerk mit Kostenfunktion*. Die Idee dabei ist, dass jeder Knoten Fluss „verursacht“, der von den Facetten absorbiert wird. Überquert der Fluss dabei eine Kante, so bedeutet das, dass sie einen Knick bekommt. Dazu definieren wir das Flussnetzwerk $N(G) = ((W, A), b, l, u, cost)$ wie folgt:

- $W = V \cup F$
- $A = \{(v, f) \in V \times F \mid v \text{ liegt auf } f\} \cup \{(f, g) \in F \times F \mid f \text{ und } g \text{ benachbart}\}$

Dabei sind die Kanten der Form $(f, g) \in A$ gerichtete Dualkanten zu Kanten von G.

Nun definieren wir den Flussbedarf bzw. -überschuß b der Facetten bzw. Knoten von G. Dabei bezeichnet $d(f)$ die Anzahl Kanten, von denen $f \in F$ begrenzt wird.

- $b(v) = 2\pi$ für alle $v \in V$

- $b(f) = -(d(f) + 2)\pi$ für alle $f \in F - f_0$
- $b(f) = -(d(f_0) - 2)\pi$

Dabei entspricht eine Flusseinheit einem Winkel bzw. Knick von $\frac{\pi}{2}$.

Nun modellieren wir die Tatsache, dass an einem Knoten zwischen zwei Kanten mindestens ein Winkel von $\frac{\pi}{2}$ anliegen muss. Desweiteren kann an einem Knoten höchstens der Winkel 2π anliegen. Eine Kante kann keine oder beliebig viele Knicke besitzen. Dies drücken wir durch die Funktionen l bzw. u („lower“ bzw. „upper“) auf den Kanten in A aus:

- $l(v, f) = 1$ und $u(v, f) = 4$ für alle $v \in V, f \in F$
- $l(v, f) = 1$ und $u(v, f) = \infty$ für alle $f, g \in F$

Um die Anzahl der Knicke auf Kanten in G zu minimieren, d. h. den Fluss auf deren Dualkanten zu minimieren, legen wir eine Kostenfunktion *cost* fest, die besagt, wieviel ein „Knick kostet“.

- $cost(v, f) = 0$ für alle $(v, f) \in A \cap V \times F$
- $cost(f, g) = 1$ für alle $(f, g) \in A \cap F \times F$

Nun haben wir das Flussnetzwerk mit Kostenfunktion aufgestellt. Unter einem Fluss X auf $N(G)$ verstehen wir eine Abbildung $X : A \to \mathbb{R}$, die die Kapazitätzbedingung

$$l(e) \leq X(e) \leq u(e) \quad \text{für alle } e \in A,$$

die Facettenbedingung

$$\begin{aligned} \sum_{v \in V,\text{ v liegt auf } f} X(v, f) &= d(f) - 2 \text{ für alle } f \in F - f_0 \\ \sum_{v \in V,\text{ v liegt auf } f_0} X(v, f_0) &= d(f_0) - 2 \end{aligned}$$

erfüllt. Die Kosten des Flusses X definieren wir mit

$$cost(X) = \sum_{(f,g) \in A} X(f, g).$$

Nun können wir den folgenden Satz zitieren.

Satz 1.4 (Tamassia [20]) *Sei $G = (V, E)$ ein planar eingebetteter Graph mit Facetten F und äußerer Facette $f_0 \in F$. Es existiert eine orthogonale Beschreibung von G mit k Knicken genau dann, wenn es in $N(G)$ einen Fluss X mit Kosten k gibt.*

Wendet man nun auf $N(G)$ einen Algorithmus an, der einen Fluss mit minimalen Kosten berechnet, so erhält man eine orthogonale Beschreibung mit minimaler Anzahl an Knicken. Ein Algorithmus der beachtet, dass $N(G)$ relativ wenig Kanten besitzt, ist in [11] gegeben. Er garantiert eine Laufzeit von $\mathcal{O}(n^{7/4} \log n$.

Kapitel 2

Erstellen einer Kaktusdarstellung

In diesem Abschnitt wollen wir uns mit dem Erstellen einer Kaktusdarstellung der minimalen Schnitte eines gewichteten Graphen $G = (V, E, c)$ beschäftigen. Die Menge der minimalen Schnitte von G seien wieder mit $\mathcal{C}(G)$ bezeichnet. Das hier vorgestellte Verfahren beruht auf mehreren Teilen. Als erstes müssen alle minimalen Schnitte gefunden werden. Dazu wird der Algorithmus von Goldberg und Tarjan [12] verwendet, der zu einem gegebenen Netzwerk einen maximalen Fluss berechnet. Er wird auf $|V|$ Netzwerke angewendet, deren Größe abnimmt. Hao und Orlin [13] nutzen diese Tatsache aus, und erreichen mit einigen Abänderungen, dass die Laufzeit zur Bestimmung aller minimalen Schnitte asymptotisch gleich dem Durchlauf eines Goldberg–Tarjan–Algorithmus ist, d. h. in $\mathcal{O}(|V||E| \log {}^{|V|^2}/_{|E|})$ liegt. Im zweiten Schritt wird aus den so ermittelten minimalen Schnitten eine Kaktusdarstellung konstruiert. In [9] stellt L. Fleischer einen Algorithmus vor, mit dem es möglich ist, die Kaktusdarstellung ebenfalls in $\mathcal{O}(|V||E| \log {}^{|V|^2}/_{|E|})$ anzufertigen. Karger und Stein [14] haben mittlerweile ein Verfahren entwickelt, mit dem es möglich ist, alle minimalen Schnitt in $\mathcal{O}(|V|^2 \log |V|)$ Zeit zu ermitteln. Es ist aber noch nicht bekannt, ob es einen effizienten Algorithmus gibt, mit dem daraus eine Kaktusdarstellung entwickelt werden kann.

Für den weiteren Verlauf sei $G = (V, E, c)$ ein gewichteter ungerichteter Graph mit $c : E \to \mathbb{N}$, $n = |V|$ und $m = |E|$.

2.1 Finden der minimalen Schnitte von G

Um den Goldberg–Tarjan–Algorithmus anwenden zu können, interpretieren wir die Kantengewichte als Kapazitäten. G wird zum *Flussnetzwerk* $N_1 = (V, A, cap)$, indem wir alle Kanten $v, w \in E$ durch zwei gerichtete Kanten $(v, w) \in A$ und $(w, v) \in A$ ersetzen, mit $cap(v, w) = cap(w, v) = c(\{v, w\})$. (w, v) heißt dann *Rückkante* zu (v, w), ebenso ist (v, w) Rückkante zu (w, v). Ein (s, t)–*Fluss* in N_1 ist eine Abbildung $X_1 : A \to \mathbb{N}$, die die *Kapazitätsbedingung* $X_1(e) \leq cap(e)$ und die *Flusserhaltungsbedingung*

$$\sum_{u \in V \ : \ (u,v) \in A} X_1(u,v) - \sum_{w \in V \ : \ (v,w) \in A} X_1(v,w) = 0 \quad \text{für alle } v \in V - \{s,t\}$$

erfüllt. Dabei heißt $s \in V$ *Quelle* und $t \in V$ *Senke*. Der Wert des Flusses X_1 wird mit $w(X_1) = \sum_{w \in V \ : \ (s,w) \in A} X_1(s,w)$, bezeichnet, wobei dies der Fluss ist, der die Quelle verlässt.

Nun wird ein Knoten $v_1 \in V$ als Quelle und ein Knoten $v_2 \neq v_1$ als Senke ausgewählt. Im ersten Durchlauf des Goldberg–Tarjan–Algorithmus wird jedem Knoten ein Wert zugewiesen. Es werden v_1 mit n und alle anderen Knoten mit 0 initialisiert. Nun wird ein maximaler (v_1, v_2)–Fluss berechnet. Dabei wird der Wert einiger Knoten erhöht. Dieser Vorgang wird nun $n-2$ mal wiederholt. Im j–ten Durchlauf erhalten wir N_{j+1} durch verschmelzen der aktuellen Senke v_j mit der Quelle (auftretende Schleifen werden gelöscht). Es wird eine neue Senke v_{j+1} aus den Knoten mit kleinstem Wert ausgewählt.

Am Ende des Algorithmus erhalten wir den Wert eines minimalen Schnittes als das Minimum von $\{w(X_1), \ldots, w(X_{n-1})\}$. Dies folgt aus Satz 2.1.

Satz 2.1 (Max–Flow Min–Cut Theorem) *Der Wert eines maximalen (s,t)–Flusses in einem Flussnetzwerk ist gleich dem Wert eines minimalen Schnitte, der s und t trennt.*

Wir können annehmen, dass für alle $i \in \{1, \ldots, n-1\}$ und $(v,w) \in A(N_i)$ das Produkt $X_i(v,w)X_i(w,v) = 0$ ist, d. h. dass nicht sowohl eine Kante wie auch ihre Rückkante Fluss trägt. Somit ist es sinnvoll von einer *saturierten* Kante (v,w) zu sprechen, wenn $X(v,w) = cap(c,w)$ gilt. Entfernt man aus einem Flussnetzwerk die saturierten Kanten, so spricht man von dem zugehörigen *Residualnetzwerk*. Der nächste Satz hilft uns, aus den zu $N_1, \ldots, N_{n-1}$ gehörenden Residualnetzwerken alle minimalen Schnitte von G zu erhalten. Dazu benötigen wir den Begriff der abgeschlossenen Menge.

Definition 2.1 Seien $A, S \subseteq V$ Knotenmengen. S heißt der *Abschluss* von A, falls $A \subseteq S$ und S keine ausgehenden Kanten besitzt. A heißt *abgeschlossene Menge*, falls A gleich seinem Abschluss ist.

Satz 2.2 *Es gibt eine Bijektion zwischen den minimalen Schnitten, die s und t trennen, und den abgeschlossenen Knotenmengen, die s enthalten, im Residualnetzwerk bezüglich eines maximalen (s,t)–Flusses.*

2.2 Die Kettendarstellung von $\mathcal{C}(G)$

Wir betrachten nun den Spezialfall, dass die durch den Algorithmus in Abschnitt 2.1 gegebene Reihenfolge der Knoten $v_1, \ldots, v_n$ eine *Adjazenzordnung* darstellt, d. h. für $i = 1, \ldots, n-1$ ist v_{i+1} adjazent zu einem Knoten aus $V_i = \{v_1, \ldots, v_i\}$. Einen minimalen Schnitt der Form $\{S, \overline{S}\}$ mit $V_i \subseteq S - v_{i+1}$ wollen wir als minimalen (V_i, v_{i+1})–Schnitt bezeichnen. Die Menge M_{i+1} enthalte alle minimalen (V_i, v_{i+1})–Schnitte. Das nächste Lemma besagt, dass je zwei minimale Schnitte in M_{i+1} komplementär sind.

Lemma 2.1 ([15]) *Falls die Knoten $v_1, \ldots, v_n$ eine Adjazenzordnung darstellen, so enthält M_i für $i = 2, \ldots, n$ keine sich kreuzenden Schnitte.*

Beweis. Sei $v_1, \ldots, v_n$ eine Adjazenzordnung. Angenommen, $\{X, \overline{X}\}, \{Y, \overline{Y}\} \in M_i$, für ein $2 \leq i \leq n$, seien sich kreuzende minimale (V_{i-1}, v_i)–Schnitte und o. E. sei $V_{i-1} \subseteq X \cap Y$. Da also V_{i-1} in X und in Y liegt, muss v_i in $\overline{X}$ und $\overline{Y}$ liegen, d.h. $v_i \in \overline{X} \cap \overline{Y}$. Aus Lemma 1.6 folgt $c(X \cap Y, \overline{X} \cap \overline{Y} = 0$. Dies steht im Widerspruch zur Tatsache, dass v_i adjazent ist zu Knoten in V_{i-1}. □

Sei $i \in \{2, \ldots, n\}$ und $M_i = \big\{\{S_j, \overline{S}_j\} \mid 1 \leq j \leq l\big\}$. Da die minimalen Schnitte in M_i komplementär sind, können wir $S_1 \subsetneq S_2 \subsetneq \ldots \subsetneq S_l$ annehmen. Wir wollen eine *Kette*, d. h. einen Pfad, P_i aus Mengen konstruieren, aus dem wir die minimalen Schnitte $\{S_j, \overline{S}_j\}$ rekonstruieren können. Wir beginnen mit $X_1 = S_1$ und definieren die folgenden Mengen rekursiv durch $X_j = S_j - S_{j-1}$ für $2 \leq j \leq l$ und $X_{l+1} = \overline{S}_l$. Jede Menge X_j, für $j = 3, \ldots, n-1$, ist genau zu den Mengen X_{j-1} und X_{j+1} verbunden. Somit ist $P : X_2, \ldots, X_n$ eine Kette aus Mengen.

Für $j = 1, \ldots, l$ können wir nun den minimalen Schnitt $\{S_j, \overline{S}_j\}$ rekonstruieren, indem wir die Kanten zwischen X_j und X_{j+1} löschen. Desweiteren ist i der kleinste Index, so dass $v_{i-1} \in S_j$ und $v_i \in \overline{S}_j$, d. h. jeder minimale Schnitt in G ist genau in einem M_i enthalten. Daher nennen wir $P_2, \ldots, P_n$ die *Kettendarstellung* von $\mathcal{C}(G)$.

In diesem Abschnitt sind wir davon ausgegangen, dass es für die Knoten von $v_1, \ldots, v_n$ eine Adjazenzordnung darstellt. Ist dies nicht der Fall, so kann es passieren, dass für ein $i \in \{2, \ldots, n\}$ die Menge M_i sich kreuzende minimale Schnitte enthält. Damit ist P_i aber keine Kette, sondern ein Kaktus. L. Fleischer hat in [9] ein Verfahren gezeigt, das aus der Darstellung der minimalen Schnitte durch die abgeschlossenen Mengen der in Abschnitt 2.1 erhaltenen Residualnetzwerke.

2.3 Von der Kettendarstellung zum Kaktus

Nun wollen wir aus der in Abschnitt 2.2 erhaltenen Kettendarstellung $P_2, \ldots, P_n$ eine Kaktusdarstellung gewinnen. Dieser Ansatz beruht auf Karzanov und Timofeev [15].

$G = (V, E, c)$ sei weiterhin der betrachtete Graph und $\mathcal{R} = (R, \varphi)$ mit $R = (W, E)$ eine normale Kaktusdarstellung für $\mathcal{C}(G)$. $G_i = (V_i, E_i, c_i)$ bezeichne den durch die Knotenmenge $V_i = \{v_1, \ldots, v_i\}$ induzierten Subgraphen von G. Weiter sei $2 \leq r \leq n$ der Index für den G_r der kleinste Graph mit Wert eines minimalen Schnittes λ ist. Dabei ist r der größte Index mit dieser Eigenschaft. Es ist zu bemerken, dass der Wert eines minimaler Schnitt in G_i, für $1 \leq i \leq r$, ebenfalls den Wert λ besitzt. Im Folgenden konstruieren wir die Kaktusdarstellungen $\mathcal{R}_i$ von $\mathcal{C}(G_i)$ rekursiv für $i = r, \ldots, 2$.

Wir beginnen mit $\mathcal{R}_r$. Die minimalen Schnitte von G_r sind genau die minimalen (V_{r-1}, v_r)–Schnitte. Diese sind durch die Kette P_r repräsentiert. Somit ist $\mathcal{R}_r = M_r$ eine Kaktusdarstellung für $\mathcal{C}(G_r)$. Für die Gewinnung der folgenden Kaktusdarstellungen ist das nächste Lemma hilfreich.

Lemma 2.2 *Stellen $v_1, \ldots, v_n$ eine Adjazenzordnung dar. Für jedes nicht leere M_{i+1} gibt es einen Pfad in R_i mit den Eigenschaften, dass er mit jedem Kreis höchstens eine Kante gemeinsam hat und durch den alle minimalen Schnitte in M_i gehen.*

Beweis. Wir zeigen, dass es der kürzeste Weg zwischen in R_i, der den Knoten, der v_{i+1} enthält, mit dem Knoten, der V_i enthält, tut. Offensichtlich gehen alle Schnitte in M_i durch diesen Pfad. Der Pfad kann besitzt aus jedem Kreis höchstens eine Kante. Andernfalls besitzt der Pfad mindestens zwei Kanten, die auf einem gemeinsamen Kreis liegen. Da wir den Pfad als kürzesten Pfad gewählt haben, muss dieser Kreis aus mindestens vier Kanten bestehen. Somit kann man sich kreuzende Schnitte erzeugen, die durch diesen Pfad gehen. Das steht im Widerspruch zu Lemma 2.1. □

Zieht man nun den in Lemma 2.2 beschriebenen Pfad zusammen in einen Knoten zusammen, so löscht man aus $\mathcal{R}_i$ genau die minimalen Schnitte in M_{i+1}. Das Ergebnis ist die Kaktusdarstellung $\mathcal{R}_{i+1}$.

Unsere Aufgabe ist es nun, die Kaktusdarstellungen in umgekehrter Reihenfolge zu konstruieren, wobei wir den Anfang schon mit $\mathcal{R}_r$ gemacht haben. Angenommen, wir sind dabei, die Kaktusdarstellung $\mathcal{R}_i$ aus $\mathcal{R}_{i+1}$ zu gewinnen. Sei $q \in R_{i+1}$ der Knoten, der die Knoten $\{v_1, \ldots, v_{i+1}\}$ enthält, die die Kette P_{i+1} der Kettendarstellung der minimalen Schnitte in M_{i+1} darstellen. q wird durch die Kette P_{i+1} ersetzt und die Kanten, die zu q inzident waren, werden mit den entsprechenden Mengen von P_{i+1} verbunden. Anschließend muss die Abbildung φ_i aus φ_{i+1} gewonnen werden. Die so erhaltene Kaktusdarstellung wird in eine kreisartige Kaktusdarstellung transformiert.

Wir beschreiben die durchgeführten Schritte formal:

(i) **Ersetze q durch die Kette P_{i+1}:** Ist $P_{i+1} = X_1, \ldots, X_k$, dann lösche q und alle dazu inzidenten Kanten und füge für k neue Knoten $q_1, \ldots, q_k$ mit Kanten (q_j, q_{j+1}), für $1 \leq j < k$, ein.

(ii) **Verbinde die Kette P_{i+1} mit R_{i+1}:** Für jede Baum- oder Kreiskante (q, w) in R_{i+1} sei $W = \varphi_{i+1}^{-1}(w)$, falls w nicht leer ist. Ansonsten sei w' ein nicht leerer Knoten, der von w aus auf einem Pfad, der nicht zu dem Kreis mit (q, w) gehört, erreichbar ist, und setzte $W = \varphi_{i+1}^{-1}(w')$. Verbinde nun w mit demjenigen q_j, so dass $W \subseteq X_j$ gilt.

(iii) **Erstelle φ_i:** Sei Q die Menge der Knoten, die unter φ_{i+1} auf q abgebildet werden. Setze nun $\varphi_i^{-1}(q_j) = X_j \cap Q$ für alle $1 \leq j \leq k$.

(iv) **Erstelle eindeutige Kaktusdarstellung:** Lösche elle leeren Knoten von Grad 2 und transformiere alle leeren 3–Kreuzungsknoten in Kreise der Länge 3.

Die Schritte (i)–(iii) benötigen konstante Zeit. Somit tragen sie über den gesamten Algorithmus eine Laufzeit von $\mathcal{O}(n^2)$ bei. In Schritt (iv) werden insgesamt höchstens $\mathcal{O}(n)$ Knoten gelöscht. Ein Kreis, der einmal erstellt wurde, wird nie zerstört. Da es in einem Kaktus höchstens $\mathcal{O}(n)$ Kreise in einem Kaktus gibt, führt dies zu einer Laufzeit von $\mathcal{O}(n)$ über den gesamten Algorithmus. Da das Finden aller minimalen Schnitte die Konstruktion von $\mathcal{R}$ aus der Kettendarstellung der minimalen Schnitte überwiegt, benötigt man $\mathcal{O}(|V||E| \log {}^{|V|^2}/_{|E|})$ Zeit, um eine kreisartige Kaktusdarstellung $\mathcal{R}$ der minimalen Schnitte von $G = (V, E)$ zu erstellen.

Kapitel 3

Baum–Cluster–Zeichnung

Im Folgenden werden wir ein Verfahren erarbeiten, das aus einer Kaktusdarstellung der minimalen Schnitte eines Graphen eine Clusterhierarchie erstellt. Wir werden sehen, dass der so gewonnene Clustergraph c–planar ist und eine zugehörige Clusterzeichnung die minimalen Schnitte visualisiert. In Abschnitt 3.2 werden wir einen Algorithmus kennen lernen, der eine knickminimale OGRC–Zeichnung erstellt.

In diesem Kapitel beschränken wir uns auf Graphen, deren Kaktusdarstellung der minimalen Schnitte keine Kreise der Länge größer 3 besitzen. Dies ist nach [18] äquivalent mit der Aussage, dass sich die Kaktusdarstellung in eine kreuzungsartige Kaktusdarstellung ohne Kreise transformieren lässt. In Kapitel 6 werden wir sehen, dass z. B. triangulierte, chordale und gewisse Brücken–Graphen diese Forderung erfüllen. In Kapitel 4 werden wir die Einschränkung durch eine Erweiterung des Algorithmus aufheben können.

3.1 Erstellen einer Clusterhierarchie

Voraussetzung

Sei $G = (V, E)$ ein planarer Graph und $\mathcal{R} = (R, \varphi)$ eine Kaktusdarstellung der minimalen Schnitte von G, wobei $R = (W, F)$ der zu $\mathcal{R}$ gehörende Kaktus ist und keine Kreise der Länge größer 3 enthält.

Ausgabe

Algorithmus 3.1 erstellt eine Clusterhierarchie $(G, \mathcal{T}(G))$, aus der die minimalen Schnitte von G rekonstruiert werden können. Dabei ist $\mathcal{T}(G) = ((W', F'), \tilde{\varphi})$.

Algorithmus 3.1 Erstellen der Cluster–Struktur anhand des Kaktus

Voraussetzung: G=(V,E) planarer Graph, $\mathcal{R} = ((W, F), \varphi)$ zugehörige Kaktusdarstellung

Rückgabe: $\mathcal{T} = ((W', F'), \tilde{\varphi})$ ist Baumdarstellung der minimalen Schnitte von G.

1: kopiere $\mathcal{R} = ((W, F), \varphi)$ nach $\mathcal{T} = ((W', F'), \tilde{\varphi})$;
2: **für alle Kreise** $C \subseteq (W', F')$ **führe aus**
3:    erstelle neuen Kreisknoten w_C;
4:    **für alle Knoten** $w \in W'(C)$ **führe aus**
5:       verbinde w mit w_C;
6:    $F' \leftarrow F' - F'(C)$; {Lösche die Kanten von C aus F'}
7: **für alle Knoten** $v \in W'$ mit $d(v) = 2$ und $\tilde{\varphi}^{-1}(v) = \varnothing$ **führe aus**
8:    ersetze Kanten $\{u, v\}$ und $\{v, w\}$, $u \neq w$, durch $\{u, w\}$;
9:    lösche v;
10: **für alle Knoten** $v \in W'$ mit $d(v) = 1$ **führe aus**
11:    **für alle Knoten** $s \in \varphi^{-1}(v)$ **führe aus**
12:       verbinde neuen Knoten $v_s \in W'$ mit v;
13:       $\tilde{\varphi}(s) \leftarrow v_s$
14: bestimme die Wurzel *root*;

Korrektheit und Laufzeit

Aus der Kaktusdarstellung erhält man mit Algorithmus 3.1 eine Cluster–Hierarchie, dargestellt durch den Baum $T = (W', F')$. Als erstes wird eine Kopie $\mathcal{T}$ von $\mathcal{R}$ erstellt. In Zeile 2 bis 6 wird aus der Kaktusdarstellung eine kreuzungsartige Kaktusdarstellung gewonnen. In Zeile 7 bis 9 werden redundante minimale Schnitte entfernt. Anschließend werden die Knoten von G als Blätter von T identifiziert und eine Wurzel für T bestimmt.

Angenommen die Schleife, die in Zeile 2 beginnt, wird mit dem Kreis $C = (w_1, \ldots, w_k)$ durchlaufen, dabei sei k die Länge von C. Es wird ein neuer Knoten $w_C \in W$ erstellt. Die Kanten von C werden entfernt, anschließend werden die Kanten $\{w_i, w_C\}, i = 1, \ldots, k$, eingefügt. Dadurch werden keine neuen Kreise eingeführt. Insbesondere gehört keine der neu eingefügten Kanten zu einem Kreis. Somit erhält man eine kumulierte Laufzeit von $\mathcal{O}(n)$.

Im nächsten Schritt werden die leeren Knoten vom Grad 2 entfernt und deren adjazente Knoten jeweils durch eine neue Kante verbunden. Dieser Schritt vermeidet unnötig viele verschiedene Darstellungen ein und desselben minimale Schnittes im Graphen. Offensichtlich werden dabei keine neuen Kreise erzeugt.

Da jeder Durchlauf konstante Zeit benötigt und $|W'| = \mathcal{O}(n)$ gilt, beträgt die Laufzeit der gesamten Schleife $\mathcal{O}(n)$.

Jeder Baumknoten $w \in T$ repräsentiert die Knotenmenge $\varphi^{-1}(w)$. Für jeden dieser Knoten $s \in \varphi^{-1}(w)$ wird ein neuer Knoten $w_s \in T$ erzeugt, der mit w durch eine neue Kante verbunden wird. Nachdem alle Knoten in φ^{-1} bearbeitet wurden, ist w bezüglich $\tilde{\varphi}$ ein leerer Knoten und es gilt $\{v\} = \tilde{\varphi}^{-1}(w_v)$ für alle $v \in \varphi^{-1}(w)$. Insbesondere gilt die Äquivalenz

$$v \in G \iff \{v\} = \tilde{\varphi}^{-1}(w_v).$$

Wir haben also mit $\tilde{\varphi}$ eine Bijektion von Knoten in G auf die Blätter von T.

Die so erhaltene Struktur von T stellt schon fast eine Clusterhierarchie dar. Es fehlt nur noch eine geeignete Wurzel. Diese kann nach unterschiedlichen Kriterien gewählt werden. Im Folgenden wählen wir einen leeren Knoten $root \in T$ als Wurzel, so dass für jeden Cluster $\nu \in T$ die Ungleichung $|V(\nu)| \leq \left|\overline{V(\nu)}\right|$ gilt. Dies hat den Vorteil, dass ein minimaler Schnitt $\{S, \overline{S}\}$, für den $\left||S| - \left|\overline{S}\right|\right|$ minimal ist, leicht erkannt werden kann. Eine andere Variante wäre, einen Knoten aus dem Zentrum von T als Wurzel zu wählen. Damit würde man die Höhe des Baumes minimieren. Hierbei ist genauso darauf zu achten, dass *root* ein leerer Knoten ist. Beide Möglichkeiten lassen eine Wahl der Wurzel in linearer Zeit zu.

Satz 3.1 *Eine gegebene planare Einbettung eines gewichteten planaren Graphen G kann zu einer c–planaren Clusterzeichnung des Clustergraphen $(G, \mathcal{T}(G))$ erweitert werden, die die minimalen Schnitte von G visualisiert.*

Beweis. Wir werden zuerst sehen, dass die so konstruierte Clusterhierarchie die minimalen Schnitte von G repräsentiert. Anschließend werden zeigen, dass der Clustergraph $(G, \mathcal{T}(G))$ c–planar ist.

Sei nun $root \in T$ mit $\tilde{\varphi}^{-1}(root) = \emptyset$. Wir wollen nun zeigen, dass jeder minimale Schnitt durch einen Cluster repräsentiert ist. Dazu sei $\{\nu, \mu\} \in F$, wobei ν, μ leere Knoten sind. Da wir es mit einer modifizierten kreuzungsartigen Kaktusdarstellung zu tun haben, besteht $F - \{\nu, \mu\}$ aus genau zwei Zusammenhangskomponenten, deren Urbilder unter φ einen minimalen Schnitt $\left\{S, \overline{S}\right\}$ in G darstellen. Auf diese Weise bekommt man alle minimalen Schnitte in G. Liege nun μ näher an *root* als ν. $W(\nu)$ besteht aus den Blättern des Unterbaumes von T mit Wurzel ν und es gilt $S = \varphi^{-1}(W(\nu))$. Somit wird jeder minimale Schnitt in G durch einen Cluster in T repräsentiert.

Nun zur c–Planarität von $(G, \mathcal{T}(G))$. Seien $G^* = (V^*, E^*)$ der Dualgraph von G, $E_S = \left\{\{v, w\} \in E \mid v \in S, w \in \overline{S}\right\}$ die Menge der von $\left\{S, \overline{S}\right\} \in \mathcal{C}(G)$ induzierten Schnittkanten und $E_S^* \subset E^*$ die zu diesen Schnittkanten dualen Kanten. Der durch E_S^* induzierte Subgraph (V_S^*, E_S^*) von G^* ist ein einfacher Kreis. Eine Kante $e \in E_S^*$ ist keine Baumkante, da die Endknoten der zu ihr korrespondierende Kante in G entweder beide in S oder beide in $\overline{S}$ liegen, somit ist e keine Schnittkante. Es gibt auch keine Knoten in V^*, die zu mehr als einem Kreis gehören, da sonst S oder $\overline{S}$ unzusammenhängend wären (s. Lemma 1.5).

Für eine beliebige aber feste planare Einbettung von G bedeutet dies, dass die Knoten von $\overline{S}$ in der äußeren Facette der Zeichnung von S liegen. Hiermit sind die Voraussetzungen für Satz 1.2 erfüllt und die c–Planarität von $(G, \mathcal{T}(G))$ ist gezeigt. $\square$

3.2 Erstellen einer Clusterzeichnung

Nun wollen wir eine Clusterzeichnung für den im vorigen Abschnitt gewonnen Clustergraphen $(G, \mathcal{T})$ anfertigen. Dabei wird der (eingebettete) Graph G zu G' erweitert, so dass zu jedem Cluster $\nu \in \mathcal{T} - root$ die Clustergrenze durch einen Kreis in G' dargestellt wird. Im ersten Schritt werden die Kanten, die zu ν inzident sind, durch einen Pfad der Länge zwei ersetzt. Die so entstandenen neuen Knoten heißen *Clustergrenzknoten*. Diese werden dann in „kanonischer“ Reihenfolge durch *Clustergrenzkanten* miteinander verbunden. Abschließend wird G' mit einer modifizierten Version des Algorithmus von Tamassia knickminimal gezeichnet, so dass Clustergrenzen durch Rechtecke dargestellt werden.

Voraussetzung

Gegeben sei ein c–planarer Clustergraph $(G, \mathcal{T})$.

Ausgabe

Algorithmus 3.2 erstellt eine c–planare Clusterzeichnung von $(G, \mathcal{T})$.

Korrektheit und Laufzeit

Wir erweitern $G = (V, E)$ mit Hilfe der Algorithmen 3.2 und 3.3 zu einem Graphen $G' = (V', E')$, so dass G' planar bleibt und gewisse Kreise in G' Clustergrenzen darstellen. Auf diesen kann man dann einen einbettungserhaltenden Algorithmus anwenden um eine c–planare Clusterzeichnung von $(G, \mathcal{T})$ zu erhalten. Um den Graphen G zu erhalten initialisieren wir G' mit einer Kopie von G.

Einfügen der Clustergrenzknoten Wir beginnen mit Algorithmus 3.2, der die Knoten der Clustergrenzen einfügt. Sei $e = \{v, w\} \in E$ und $\nu = \varphi(v)$ sowie $\mu = \varphi(w)$. Des Weiteren sei κ der *kleinste gemeinsame Vorgänger von ν und μ*, d. h. $V(\nu) \cup V(\mu) \subseteq V(\kappa)$ und für alle $\kappa' \in T(\kappa) - \kappa$ gilt $V(\nu) \cup V(\mu) \nsubseteq V(\kappa')$. Die Kante e ist zu allen Clustern inzident, die auf dem Pfad zwischen $\varphi(v)$ nach κ bzw. zwischen $\varphi(w)$ nach κ liegen. Für jede Kante–Cluster–Inzidenz wird ein *Clustergrenzknoten* s' eingefügt und $\psi(s')$ gibt den Cluster an, bei dem die Inzidenz auftritt. Da $(G, \mathcal{T})$ c–planar ist, besitzt jede Kante höchstens eine Inzidenz mit einem gegebenen Cluster.

Algorithmus 3.2 Einfügen der Clustergrenzknoten

1: **für alle Knoten** $v \in V'$ **führe aus**
2:     $\psi(v) \leftarrow \varphi(v)$;
3: **für alle Kanten** $\{s,t\} \in E$ **führe aus**
4:     $\kappa \leftarrow$ *kleinster gemeinsamer Vorgänger* von $\psi(s)$ und $\psi(t)$;
5:     **solange** $\psi(s) \neq \kappa$ **führe aus**
6:         erstelle neuen Knoten $s' \in V'$;
7:         ersetze in E' die Kante $\{s,t\}$ durch $\{s,s'\}$ und $\{s',t\}$;
8:         $\psi(s') \leftarrow$ Vorgänger von $\psi(s)$;
9:         $s \leftarrow s'$;
10:     **solange** $\psi(t) \neq \kappa$ **führe aus**
11:         erstelle neuen Knoten $t' \in V'$;
12:         ersetze in E' die Kante $\{s,t\}$ durch $\{s,t'\}$ und $\{t',t\}$;
13:         $\psi(t') \leftarrow$ Vorgänger von $\psi(t)$;
14:         $t \leftarrow t'$;

Es ist zu bemerken, dass $\psi^{-1}(root) = \emptyset$ gilt, denn $V = V(root)$ und somit ist der *root*–Cluster zu keiner Kante inzident.

Die Schleifen in Zeilen 7 und 12 benötigen jeweils $h(T)$ Zeitaufwand, wobei $h(T)$ die Höhe des Baumes der Clusterhierarchie ist. Die Schleife in Zeile 4 wird $\mathcal{O}(n)$ mal wiederholt. Die Laufzeit von Algorithmus 3.2 beträgt somit $\mathcal{O}(nh(t))$. Es gibt keine Cluster, die nur einen Untercluster besitzen, denn diese würden durch einen leeren Knoten vom Grad 2 repräsentiert, die aber in Algorithmus 3.1 entfernt wurden. Somit gilt $\mathcal{O}(h(n)) \subseteq \mathcal{O}(n)$.

Bei jedem Durchlauf der Zeilen 7 und 12 wird jeweils ein neuer Knoten und eine zusätzliche Kante eingefügt. Daher gilt $|V'| \in \mathcal{O}(nh(n))$.

Algorithmus 3.3 Füge Clusterumrandung für Cluster ν ein

wähle einen Knoten u aus $\psi^{-1}(\nu)$;
$u_1 \leftarrow u$;
wiederhole
    wähle einen zu u adjazenten Knoten v, wobei $\psi(v)$ Nachfolger von ν ist;
    $f \leftarrow$ Facette links von (u,v);
    wiederhole
        wähle $w \neq u$ mit f liegt links von (v,w);
        $u \leftarrow v$; $\quad v \leftarrow w$;
    bis $\psi(v) \equiv \nu$
    splitte f durch Einfügen der Kante $\{u,v\}$;
bis $v \equiv u_1$

Einfügen der Clustergrenzkanten Sei nun $\nu \in T$ ein Cluster für den die Clustergrenzen eingefügt werden sollen. Zuerst stellen wir sicher, dass die Clustergrenzen der Untercluster von ν bereits eingefügt sind, indem nacheinander Algorithmus 3.3 mit den Nachfolgern von ν als Argument aufgerufen wird. Seien $\psi^{-1}(\nu) = \{u_1, \ldots, u_k\}$ die Clustergrenzknoten, die aufgrund der zu ν inzident Kanten eingefügt wurden. Noch ist die Reihenfolge nicht bekannt, in der die Knoten um den Cluster ν angeordnet sind. Jeder Knoten u_i, $i = 1, \ldots, k$, ist vom Grad 2. Wir beginnen mit $u \in \psi^{-1}(\nu)$ und suchen den zu u adjazenten Knoten v, der innerhalb des Clusters ν liegt — sei es als Graphknoten $v \in V(\nu)$ oder Clustergrenzknoten $v \in \psi^{-1}(\mu)$ für einen Untercluster μ von ν. Man geht nun an der Facette entlang, die links von der Kante (u, v) liegt, bis $v \in \psi^{-1}(\nu)$ gilt. Die Facette f wird durch Einfügen der Clustergrenzkante $\{u_1, u\}$ gesplittet. Ist ν nur zu einer bzw. zwei Kanten inzident, so entsteht ein Schleife oder Doppelkanten. Um dies zu vermeiden ersetzen wir die zuletzt eingefügte Kante durch einen Pfad bestehend aus drei bzw. zwei Kanten.

Dieses Verfahren wiederholt man mit $u \leftarrow v$, bis v gleich dem Knoten u des ersten Durchlaufs ist. Es werden genau die Clustergrenzen von ν eingefügt und im weiteren Verlauf werden die soeben berührten Kanten nicht nochmals abgetastet. Da für jeden Clustergrenzknoten eine Clustergrenzkante eingefügt wird, ist die Laufzeit in $\mathcal{O}(|E'|) = \mathcal{O}(|V'|)$.

Orthogonale Darstellung Nun wollen wir aus G' mit Hilfe des Algorithmus von Tamassia eine orthogonale Repräsentation erstellen. Das Flussnetzwerk wird wie in Abschnitt 1.4 beschrieben aufgebaut. Wie in Abschnitt 1.3 erwähnt, ist es möglich eine Clusterzeichnung anzufertigen, so dass die Clustergrenzen rechteckig sind. Um dies zu garantieren, muss das Flussnetzwerk so modifiziert werden, dass sich Clustergrenzkanten nur in eine bestimmte Richtung knicken können. Dazu sei $\nu \in \mathcal{T} - root$ ein Cluster und $e \in E'$ eine Clustergrenzkante aus der Clustergrenze von nu. Seien f_1 bzw. f_2 die Facetten, die an e grenzen. Desweitern liege f_1 in dem Gebiet innerhalb des Clusters ν. Um rechteckige Clustergrenzen zu garatieren, muss sich e nach Außen knicken, d. h. es darf kein Fluß von f_2 nach f_1 fließen. Dies wird durch Löschen der Kante (f_2, f_1) aus dem Flußnetzwerk erreicht. Dieses Verfahren wird nun auf jede Clusterkante angewendet. Die Entscheidung, ob eine Kante gelöscht werden soll, kann in konstanter Zeit durchgeführt werden, wenn man für jede Facette die Tiefe des aktuellen Clusters abspeichert. Somit kann die Modifikation in $\mathcal{O}(|V'|)$ durchgeführt werden.

Auf das so modifizierte Flußnetzwerk wird der Algorithmus von Tamassia angewendet. Mit $N = |V'|$ führt dies zu der Laufzeit $\mathcal{O}(N^{7/4}\sqrt{\log N})$.

Kapitel 4

Kaktus–Cluster–Zeichnung

In Kapitel 3 haben wir einen Algorithmus kennengelernt, der eine hierarchische Clusterzeichnung der minimaler Schnitte eines planaren Graphen G anfertigt. Die Korrektheit ist aber nur gegeben, wenn G keine Kuchenpartition größer 3 enthält. In diesem Kapitel werden wir uns mit einer Erweiterung beschäftigen, die solche Kuchenpartitionen in G zulässt.

Was läuft schief, wenn wir das Verfahren der Baum–Cluster–Zeichnung auf einen Graphen anwenden, der Kuchenpartitionen der Größe mindestens vier enthält? Wir wollen dies anhand des gewichteten Graphen $G = (V, E)$ in Abbildung 4.1(a) verdeutlichen. Der Wert eines minimalen Schnittes beträgt $\lambda = 4$. Es gibt eine Kuchenpartition der Größe 4, nämlich $\pi = (V_1, V_2, V_3, V_4) = (\{4\}, \{1\}, \{0\}, \{2, 3, 5, 6\})$. In Abbildung 4.1(b) ist die Clusterhierarchie gezeigt, die von Algorithmus 3.1 ausgegeben wird. Dabei ist der Kreisknoten γ, der für den Kreis der Länge (der π erzeugt) eingefügt wurde, ist gesondert gekennzeichnet. Betrachten wir die Zeichnung in Abbildung 4.1(c), die von den Algorithmen 3.2 und 3.3 erzeugt wurde. Nun werden wir sehen, dass nicht alle sechs minimalen Schnitte, die durch π induziert werden, in der Zeichnung erkannt werden können. Die minimalen Schnitte der Form $\{V_i, \overline{V_i}\}$ werden für $i = 1, \ldots, 4$ korrekt dargestellt. Hingegen werden die minimalen Schnitte $\{V_1 \cup V_2, V_3 \cup V_4\}$ und $\{V_2 \cup V_3, V_1 \cup V_4\}$ nicht visualisiert. Dies liegt daran, dass γ nicht als Kreisknoten, sondern als Kreuzungsknoten behandelt wird. Eine Möglichkeit ist, die Rekonstruktion der verloren gegangenen minimalen Schnitte dem Betrachter zu überlassen. Dazu machen wir die folgende Betrachtung.

Sei $(W_1, \ldots, W_r)$ eine Kuchenpartition der Größe $r \geq 4$ und liege o. E. bei der zugehörigen Baum–Cluster–Zeichnung der minimalen Schnitte $W_1 \cup \ldots \cup W_{r-1})$ innerhalb des Clusters γ und W_r außerhalb davon. Weiter repräsentieren die Cluster ν_i, für $i = 1, \ldots, r-1$, die minimalen Schnitte der Form $\{W_i, \overline{W_i}\}$. Die Cluster $\nu_1, \ldots, \nu_{r-1}$ bilden einen „Pfad“, denn für $i = 2, \ldots, r-2$ hat ν_i genau zu den

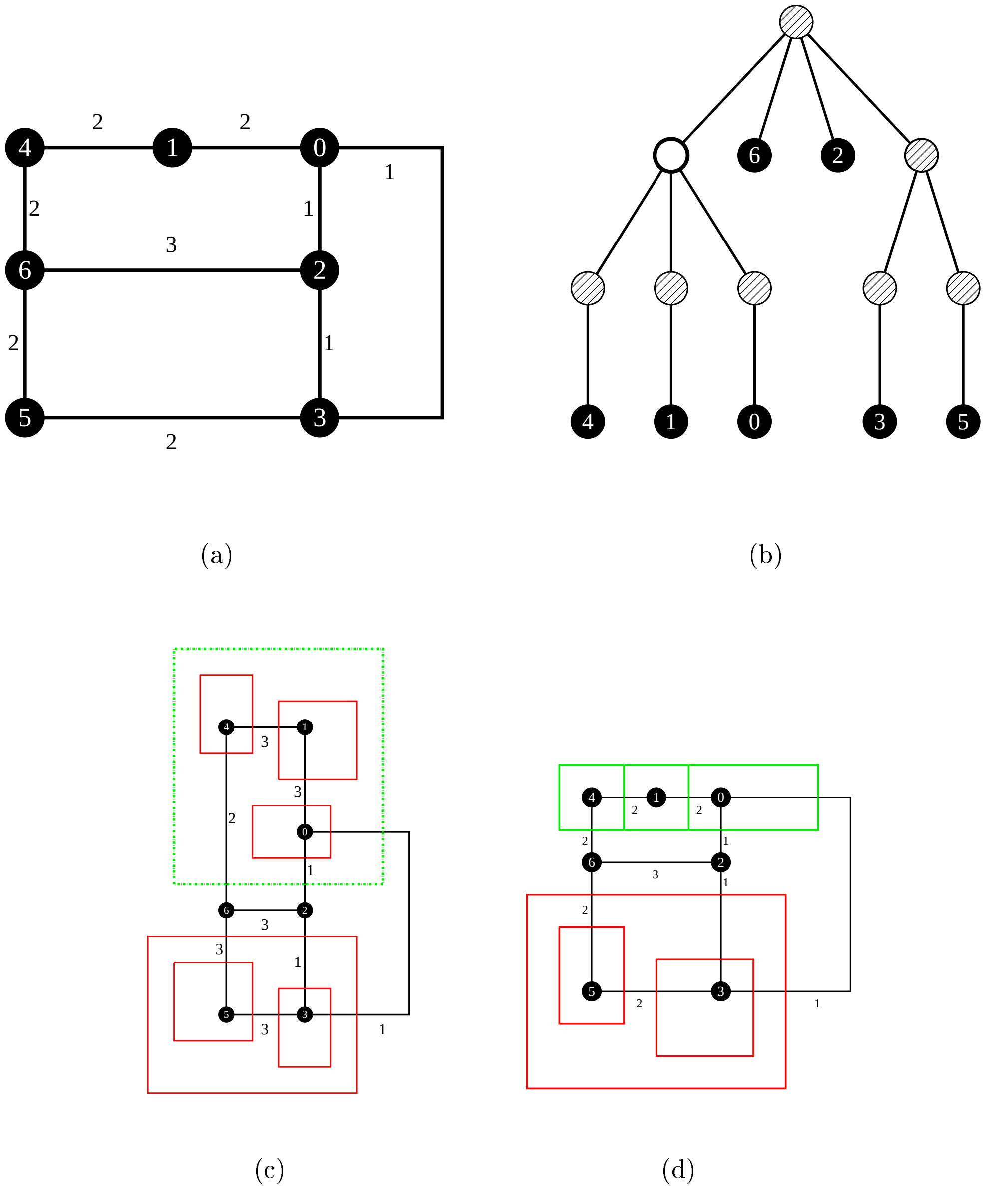


Abbildung 4.1: (a) Der gewichtete Graph „`simple_graph.gw`“. Die verschiedenen Typen der Cluster–Knoten werden in den Cluster–Kaktus–Zeichnung verdeutlicht. Dabei liegt der Zeichnung in (b) ein kreuzungsartiger und in (c) ein kreisartiger Kaktus zugrunde.

Clustern ν_{i-1} und ν_{i+1} inzidente Kanten. Minimale Schnitte der Form

$$\left\{\bigcup_{i=a}^{b} W_i,\ V - \bigcup_{i=a}^{b} W_i\right\},$$

für $1 \leq a < b \leq r-1$, können rekonstruiert werden, indem man die Cluster $\nu_a, \ldots, \nu_b$ zusammenfasst. Dies ist aber recht unbefriedigend, da man bei großen Graphen schnell den Überblick verlieren kann.

4.1 $r-2$ Facettenknoten vom Grad 3

Dieses Problem kann man beheben, in dem man eine Zeichnung wie in Abbildung 4.1(d) gezeigt anstrebt. Mit der Bezeichnung von oben folgt, dass es genau zwei Facetten f_1 und f_2 gibt, an die die Cluster $\nu_1, \ldots, \nu_{r-1}$ grenzen. Wir ersetzen nun die beiden Clustergrenzkanten, die γ begrenzen und zu f_1 bzw. f_2 gehören, durch Pfade p_1 bzw. p_2 der Länge $r-2$. Nun ziehen wir alle Kanten, die zwischen den Clustern $\gamma, \nu_1, \ldots, \nu_{r-1}, \gamma$ verlaufen, zusammen und löschen auftretende Doppelkanten, sowie die Clustergrenzkanten der ν_i, die an f_1 oder f_2 grenzen. Nun verbinden wir sukzessive einen Knoten aus p_1 mit einem der $r-2$ Clustergrenzknoten zwischen den Clustern $\nu_1, \ldots, \nu_{r-1}$, die auf der Facette f_1 liegen. Dasselbe passiert mit den Knoten aus p_2 und den Clustergrenzknoten auf der Facette f_2. Aus der so gewonnenen, in Abbildung 4.1(d) gezeigten Zeichnung, lassen sich alle geforderten minimalen Schnitte ablesen.

Die gezeigte Zeichnung ist aber nicht knickminimal unter den bisher gestellten Forderungen. Bisher haben wir lediglich verlangt, dass die einzelnen Cluster recht-

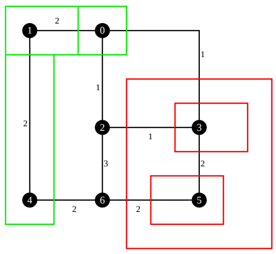


Abbildung 4.2:

eckige Clustergrenzen besitzen. Dies führt lediglich zu der in Abbildung 4.2 gezeigten Zeichnung, die knickminimal ist. Um dies zu verhindern, ist es notwendig, sowohl Knicke auf gewissen Clustergrenzkanten zu verbieten wie auch die an Clustergrenzknoten auftretenden Winkel festzulegen. Es ist nicht ohne weiteres ersichtlich, dass solche Einschränkungen einen Fluss im zugehörigen Flussnetzwerk zulassen. Dazu wählen wir einen anderen Zugang zu dieser Zeichnung, die aus der hierarchischen Cluster–Zeichnung gewonnen wird. Wir werden zunächst mehr Einschränkungen als notwendig vornehmen und zeigen, dass mit diesen Einschränkungen eine Zeichnung möglich ist.

Gegeben:

- $G = (V, E)$ ein planar eingebetteter 4–Graph,
- $\mathcal{R} = (R, \varphi)$ eine Kaktusdarstellung von $\mathcal{C}(G)$,
- $\mathcal{T} = (T, \psi)$ die aus $\mathcal{R}$ gewonnene Clusterhierarchie,
- $G' = (V', E')$ erweiterter Clustergraph, der die Clustergrenzen in $G, \mathcal{T})$ modelliert.
- Flussnetzwerk $N(G') = ((W, A); b, l, u, cost)$,
- X Fluss in $N(G')$, der rechteckige Clustergrenzen garantiert.

Des Weiteren sei $\gamma \in T$ ein Kreisknoten mit den Unterclustern $\nu_1, \ldots \nu_r$, für ein $r \geq 3$, die die Knotenmengen $V_1, \ldots, V_r$ enthalten. Wir bezeichnen mit $V_0 = V - (V_1 \cup \ldots \cup V_r)$ das Komplement der in γ enthaltenen Knoten.

Die Cluster ν_i, für $i = 1, \ldots, r$, seien durch k_i Clustergrenzknoten $c_{i,1}, \ldots, c_{i,k_i}$ in dieser Reihenfolge begrenzt. Die Clustergrenzknoten von ν_i, $i = 1, \ldots, r$, können in zwei Intervalle unterteilt werden: die Clustergrenzknoten $c_{i,1}, \ldots, c_{i,\alpha_i}$ sind zum Cluster ν_{i-1} inzident, wobei die Clustergrenzknoten $c_{i,\alpha_i+1}, \ldots, c_{i,k_i}$ zum Cluster ν_{i+1} inzident sind. Dabei identifizieren wir ν_0 und ν_{n+1} mit γ. In Abbildung 4.3 wird die Bezeichnung verdeutlicht.

Vorgehen: Mit Algorithmus 4.1 werden die Knicke der Kanten so verändert, dass folgendes garantiert ist:

(i) Kanten der Form $(c_{i,j}, c_{i+1,k_i-j+1})$, für $i = 1, \ldots, r-1$ und $j = \alpha_i + 1, \ldots, k_i$, besitzen keine Knicke.

(ii) Clustergrenzkanten der Form $(c_{i,j}, c_{i,j+1})$, für $i = 1, \ldots, r$ und $j = 1, \ldots, \alpha_i - 1, \alpha_i + 1, \ldots, k_i - 1$, besitzen keine Knicke.

(iii) Clustergrenzkanten der Form $(c_{i,\alpha_i}, c_{i,\alpha_i+1})$ und $(c_{i,k_i}, c_{i,1})$, für $i = 1, \ldots, r-1$, besitzen genau zwei Knicke nach außen.

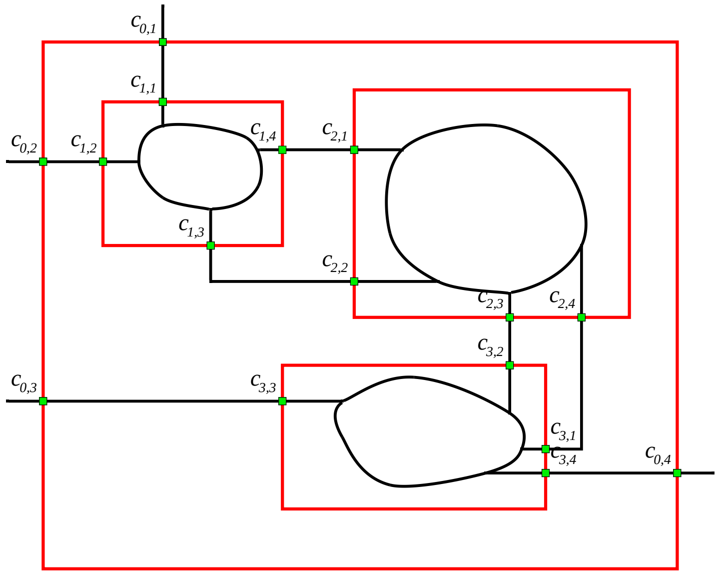


Abbildung 4.3: Ausschnitt eines Graphen in Gestalt einer einfachen Kaktus–Cluster–Zeichnung.

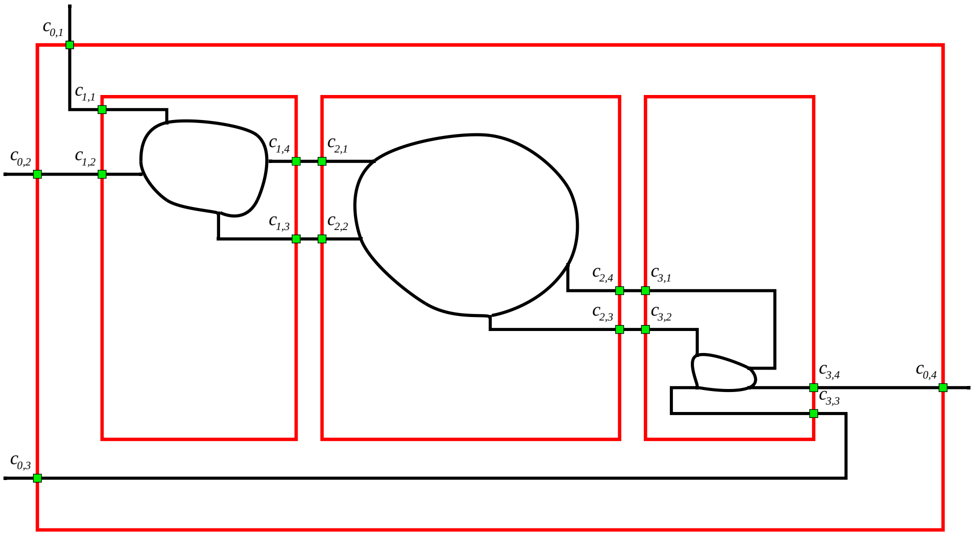


Abbildung 4.4: Der Graph aus Abbildung 4.3 nach Anwenden der Schritte 1–9 von Algorithmus 4.1

Im Folgenden wird Algorithmus 4.1 und die von ihm aufgerufenen Prozeduren beschrieben.

Algorithmus 4.1 Die Schleife in Zeile 1 begradigt die Kanten auf der „Vorderseite“ des Clusters ν_1. Diese Begradigung ist nur bei dem ersten Cluster, denn wie wir sehen werden, sind die Kanten auf der „Vorderseite“ des Clusters ν_{i+1} nach dem i–ten Durchlauf der Schleife in Zeile 3 schon begradigt.

Angenommen die Schleife in Zeile 3 befinde sich im i–ten Durchlauf und die Clustergrenzen $(c_{i,j}, c_{i,j+1})$ seien für $i = 1, \dots, \alpha_i - 1$ begradigt. Der Clustergrenze $(c_{i,\alpha_i}, c_{i,\alpha_i+1})$ werden nun zwei Knicke zugeordnet. Dies kann dazu führen, dass die Clusterbegrenzung von ν_i zeitweise nicht rechteckig ist. Wir werden aber sehen, dass sie nach Durchlauf der Schleife rechteckig ist. Im nächsten Schritt werden die Clustergrenzen $(c_{i,j}, c_{i,j+1})$ für $i = \alpha_i{+}1, \dots, k_i{-}1$ begradigt. Die bisher betrachteten Clustergrenzen besitzen insgesamt 2 Knicke. Da wir von einem Fluss mit rechteckigen Clusterbegrenzungen ausgegangen sind und ein Rechteck 4 Knicke besitzt, besitzt die Kante $(c_{i,k_i}, c_{i,1})$ nach Normalisieren folglich die beiden übrigen Knicke. Die Schleife in Zeile 7: begradigt nun die Kanten, die die Cluster ν_i und ν_{i+1} verbinden. Nun wird klar, warum die Clustergrenzen auf der „Vorderseite“ von ν_{i+1} nach diesem Schritt begradigt sind: Die von $(c_{i+1,l}, c_{i,j})$, $(c_{i,j}, c_{i,j+1})$, $(c_{i,j+1}, c_{i+1,l-1})$ und $(c_{i+1,l-1}, c_{i+1,l}$ begrenzte Facette hat Grad vier und somit einen Flussbedarf von 4. Jeder daran angrenzende Knoten versorgt die Facette mit einer Flusseinheit, so dass der Bedarf schon gedeckt ist. Da die ersten drei Kanten begradig sind, tragen sie keinen Fluss in die Facette. Das bedeutet, dass die letzte Kante, wenn sie normalisiert ist, keinen Knick besitzen kann.

Nun werden die Cluster mit Prozedur 4.4 „zusammengezogen“.

Algorithmus 4.1 Modifiziere zu erweiterter Kaktus–Cluster–Darstellung

---

1: **für** $j = 1, \dots, \alpha_1 - 1$ **führe aus** {Vorbereitung für die Schleife in 3:}
2:     begradige $(c_{1,j}, c_{1,j+1})$ durch $c_{1,j+1}$; {Clusterkanten auf der „Vorderseite“}
3: **für** $i = 1, \dots, n - 1$ **führe aus**
4:     setze Anzahl der Knicke der Kante $(c_{i,\alpha_i}, c_{i,\alpha_i+1})$ durch c_{i,α_i+1} auf 2;
5:     **für** $j = \alpha_i + 1, \dots, k_i - 1$ **führe aus**
6:         begradige $(c_{i,j}, c_{i,j+1})$ durch $c_{i,j+1}$; {Clusterkanten auf der „Rückseite“}
7:     **für** $j = \alpha_i + 1, \dots, k_i$ **führe aus**
8:         $l \leftarrow k_{i+1} - j + 1$;
9:         begradige $(c_{i,j}, c_{i+1,l})$ über $c_{i+1,l}$; {Graphkante zwischen Clustern}
10: füge Cluster aneinander;

Prozedur 4.2 Diese Prozedur setzt die Anzahl der Knicke einer Kante $e = (v, w)$

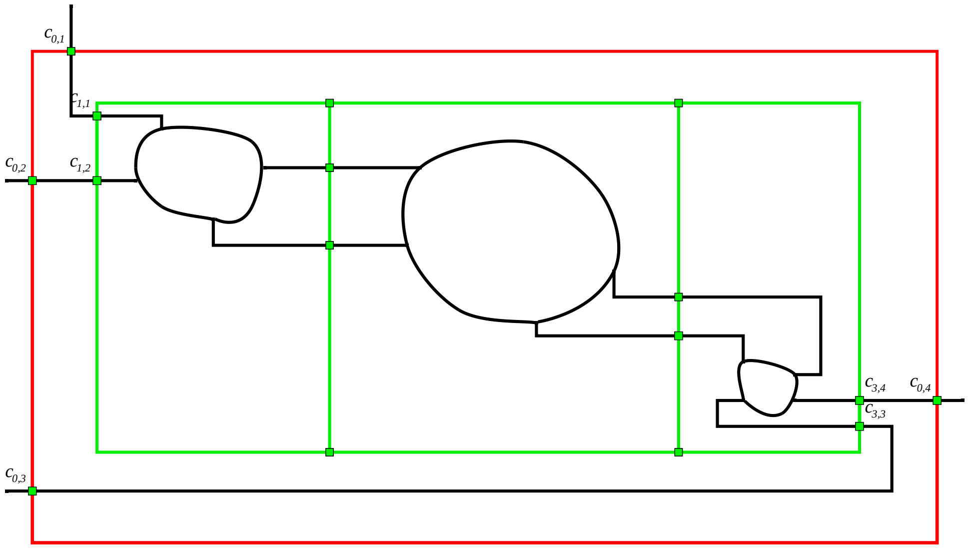


Abbildung 4.5: Der Graph aus Abbildung 4.4 nach zusammenziehen der Cluster

auf b. Ist der Wert b positiv, so erhält e die Anzahl von b Knicken nach links, sonst nach rechts. Im Fall $b = 0$ sprechen wir vom *begradigen* von e.

In Schritt 2–5 wird bestimmt um wieviel der Wert des Flusses auf den Dualkanten von e geändert werden muss. Dabei bezeichnet Δ_+ die Flusserhöhung der Dualkante, die e von links nach rechts überquert und Δ_- die Flusserhöhung der Dualkante, die e von rechts nach links überquert. Die Schleife in Schritt 6 aktualisiert den Fluss zwischen Facette f_i und f_j, wobei $j = i + 1 \mod 4$. Dabei wird der Fluss auf $X(f_1, f_2)$ bzw. $X(f_2, f_1)$ um Δ_+ bzw. Δ_- erhöht, wobei höchstens einer der beiden Werte von Null verschieden ist. Um nicht unnötig viel Fluss zu produzieren, wird in Schritt 10 Fluss, der sich auf beiden Dualkanten befindet, abgezogen.

Offensicht ist X am Ende der Prozedur wieder ein gültiger Fluss. Denn wird der Fluss erhöht, der in einen Facettenknoten fließt, dann wir auch der den Facettenknoten verlassende Fluss erhöht. Desweitern werden keine anderen als die in Abbildung 4.6(a) gezeigten Kanten verwendet.

Prozedur 4.3 Fließt auf beiden Dualkanten der Kante e Fluss, so kann man den kleineren der beiden Werte davon abziehen und erhält einen Fluss mit geringeren Kosten. (s. Abbildung 4.6(b))

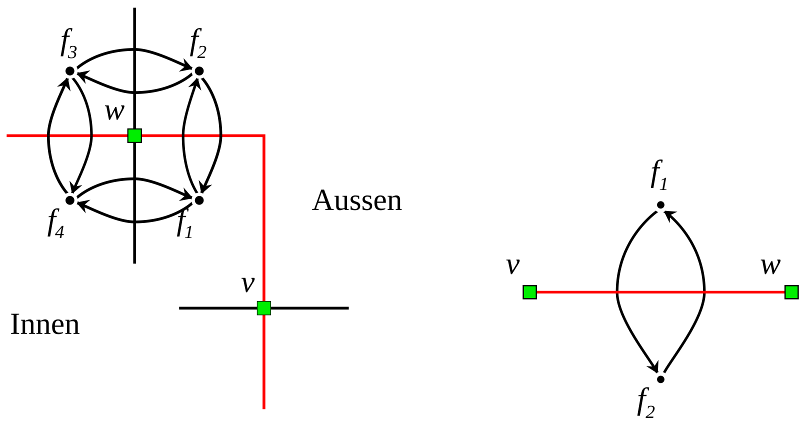
Prozedur 4.4 Hier werden die Cluster $\nu_1, \ldots, \nu_r$ aneinandergefügt. Das Vorgehen wird schnell deutlich, wenn man die Abbildungen 4.4 und 4.5 vergleicht.

Prozedur 4.2 setze Knickzahl von $e = (v, w)$ durch w auf b

Voraussetzung: Kante $e = (v, w)$, $f_1, \ldots, f_4$ in dieser Reihenfolge zu w adjazente Facetten, wobei f_1 bzw. f_2 links bzw. rechts von e liegt. X ist Flussnetzwerk.

Rückgabe: X' ist gültiges Flussnetzwerk, bei dem b die Anzahl der Knicke von e ist. Es sind nur die in Abbildung 4.6(a) gezeigten Kanten des Flussnetzwerks betroffen.

$X' \leftarrow X$;
falls $X(f_1, f_2) - X(f_2, f_1) > b$ **dann führe aus**
  $\Delta_+ \leftarrow 0$; $\Delta_- \leftarrow X(f_1, f_2) - X(f_2, f_1) - b$;
sonst
  $\Delta_- \leftarrow 0$; $\Delta_+ \leftarrow -(X(f_1, f_2) - X(f_2, f_1) - b)$;
für $i \leftarrow 1, \ldots, 4$ **führe aus**
  $j \leftarrow (i \bmod 4) + 1$;
  $X'(f_i, f_j) \leftarrow X(f_i, f_j) + \Delta_+$;
  $X'(f_j, f_i) \leftarrow X(f_j, f_i) + \Delta_-$;
  normlisiere (X', f_i, f_j);
$X \leftarrow X'$;



(a) (b)

Abbildung 4.6: In Prozedur 4.2 bzw. 4.3 wird ausschließlich Fluss auf den in (a) bzw. (b) gezeigten Kanten verändert.

Prozedur 4.3 normalisiere Fluss X zwischen f_1 und f_2

Voraussetzung: Kante $e = (v, w)$, Facette f_1 liegt links von e, f_2 rechts. X ist Flussnetzwerk.

Rückgabe: X' ist gültiges Flussnetzwerk, bei dem mindestens eine der Dualkanten von e keinen Fluss trägt. Es sind nur die in Abbildung 4.6(b) gezeigten Kanten des Flussnetzwerks betroffen.

falls $X(f_1, f_2) \geq X(f_2, f_1)$ **dann führe aus**
  $X'(f_2, f_1) \leftarrow 0;$
  $X'(f_1, f_2) \leftarrow X(f_1, f_2) - X(f_2, f_1);$
sonst
  $X'(f_1, f_2) \leftarrow 0;$
  $X'(f_2, f_1) \leftarrow X(f_2, f_1) - X(f_1, f_2);$

Prozedur 4.4 Füge Cluster aneinander

Voraussetzung: Orthogonale Beschreibung, wie von Algorithmus 4.1 erzeugt.

Rückgabe: Orthogonale Beschreibung, die eine erweiterte Kaktus–Cluster–Zeichnung darstellt.

Identifiziere c_0^a mit $c_{1,1}$ und c_0^b mit c_{1,α_1};
für $i = 1, \ldots, n-1$ **führe aus**
  für $j = \alpha_i + 1, \ldots, k_i$ **führe aus**
    kontrahiere die Kante $\{c_{i,j}, c_{i+1,k_i-j+1}\}$ zu Knoten c_{i+1,k_i-j+1};
    lösche entstandene Doppelkanten;
  erstelle neue Knoten c_i^a, c_i^b;
  ersetze $\{c_{i-1}^a, c_{i,\alpha_i+1}\}$ durch $\{c_{i-1}^a, c_i^a\}$;
  ersetze $\{c_{i,k_i}, c_{i-1}^b\}$ durch $\{c_i^b, c_{i-1}^b\}$;
  ersetze $\{c_{i+1,\alpha_{i+1}}, c_{i+1,\alpha_{i+1}+1}\}$ durch $\{c_i^a, c_{i+1,\alpha_{i+1}+1}\}$;
  ersetze $\{c_{i+1,k_{i+1}}, c_{i+1,1}\}$ durch $\{c_{i+1,k_{i+1}}, c_i^b\}$;
  füge Kanten $\{c_i^a, c_{i+1,\alpha_{i+1}}\}$ und $\{c_i^b, c_{i+1,1}\}$ ein;
  passe Fluss X an;

Es ist zu bemerken, dass Algorithmus 4.1 keine Clusterkanten antastet, die nicht zu $\nu_1, \ldots, \nu_n$ gehören. Dies macht einen rekursiven Aufruf möglich, bei dem von den Blättern von T aus die Cluster zur Wurzel hin bearbeitet werden.

Algorithmus 4.1 lässt sich nun auf alle Cluster anwenden, die aus Kreisen im Kaktus entstanden sind, und zwar werden die Cluster von den Blättern zur Wurzel hin bearbeitet. Ist γ wiederum Untercluster eines Kreisknotens, so wird seine Clustergrenze bei der Bearbeitung von dessen übergeordneten Clusters behandelt. Ist hingegen γ Untercluster eines Kreuzungsknotens, dann ist dessen Clustergrenze überflüssig und kann entfernt werden. Jede bestehende Kante in G' wird nur einmal angetastet. Das gleiche gilt für die $\mathcal{O}(|V'|)$ vielen neu eingefügten Kanten. Somit ist die Laufzeit linear in der Größe von G'.

4.2 Zwei Facettenknoten vom Grad r

Hierbei handelt es sich um eine Variante der Kaktus–Cluster–Zeichnung aus dem vorangegangenen Abschnitt. Dabei werden die Pfade aus $r-2$ Facettenknoten, die zu einem Cluster gehören, verschmolzen. Dabei entstehen zwei Facettenknoten vom Grad r. Ein Nachteil, den diese Variante mit sich bringt, ist dass der erweiterte Cluster–Graph nicht mehr unbedingt 4–planar ist. Man muss dann z. B. eine Erweiterung wie in [16] verwenden

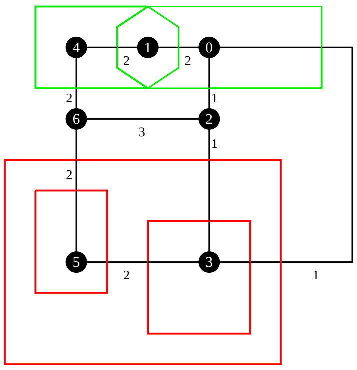


Abbildung 4.7: (a) zeigt eine Kaktus–Cluster–Zeichnung des gewichteten Graphen aus Abbildung 4.1(a). (b) stellt die erweiterte Zeichnung dar, bei der die Facettenknoten durch mehrere Knoten vom Grad drei ersetzt wurden.

Kapitel 5

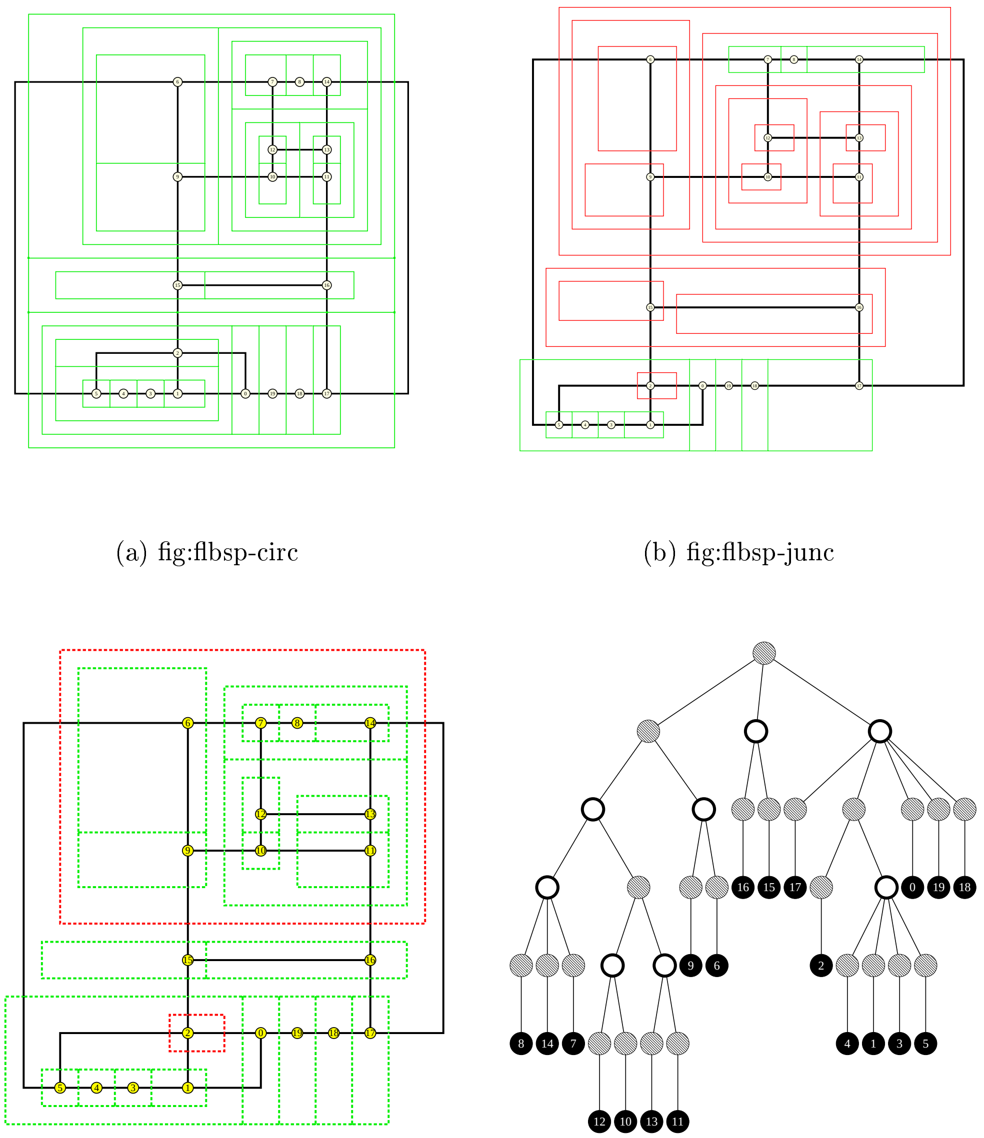
Kuchenpartitionen der Größe drei

In Kapitel 3 haben wir schon einmal die Eigenschaft der Kaktusdarstellung verwendet, dass Kuchenpartitionen der Größe drei des zugehörigen Graphen auf zwei unterschiedliche Arten dargestellt werden können. Eine Kuchenpartition der Größe drei kann sowohl durch einen Kreis der Länge drei im Kaktus, sowie durch einen leeren 3–Kreuzungsknoten, repräsentiert werden. Das in Kapitel 3 beschriebene Verfahren zur Anfertigung einer Clusterzeichnung setzt sogar voraus, dass gar keine Kreise der Länge größer drei im Kaktus auftreten. Das bedeutet, dass jede Kuchenpartition der Größe drei durch einen leeren 3–Kreuzungsknoten repräsentiert wird. Sind hingegen Kreise im Kaktus erlaubt, so wie es in Kapitel 4 der Fall ist, kann die Art der Darstellung von Kuchenpartitionen der Größe drei frei gewählt werden.

In Abbildung 5.1 sind drei Varianten einer Clusterzeichnung desselben Clustergraphen gezeigt. Der Unterschied liegt in der Art der Darstellung von Kuchenpartitionen der Größe drei durch die Kaktusdarstellung der minimalen Schnitte des Graphen. Für die in Abbildung 5.1(a) dargestellte Kaktus–Cluster–Zeichnung, der ein kreisartiger Kaktus zugrundeliegt, werden $|V'| = 132$ Knoten und $|E'| = 233$ Kanten benötigt. Der Zeichnung in Abbildung 5.1(b) liegt ein kreuzungsartiger Kaktus zugrunde. Dies resultiert in $|V'| = 116$ Knoten und $|E'| = 211$ Kanten. Legt man die Darstellung von Kuchenpartition der Größe drei in geeigneter Weise fest, so kann die Größe von $|G'|$ reduziert werden. Wendet man die Kriterien an, die wir in diesem Kapitel erarbeiten werden, so erhält man eine Kaktus–Cluster–Zeichnung mit $|V'| = 104$ Knoten und $|E'| = 182$ Kanten.

Wie wir sehen werden, sind die Einsparungen linear in $|V'|$ und $|E'|$ und führen somit nicht zu einer Verbesserung der asymptotischen Laufzeit für das Erstellen der Kaktus–Cluster–Zeichnung. Das Ziel soll vielmehr sein, eine optisch ansprechende Darstellung der minimalen Schnitte zu erzielen. Wie die Gegnüberstellung der Kaktus–Cluster–Zeichnungen in Abbildung 5.1 zeigt, ist es unübersichtlich, wenn ein minimaler Schnitt mehrfach dargestellt wird.

Das Verfahren wird formal in Algorithmus 5.1 gezeigt.



(a) fig:flbsp-circ

(b) fig:flbsp-junc

(c) fig:flbsp-redu

(d) fig:flbsp-redu-tree

Abbildung 5.1: Kaktus–Cluster–Zeichnungen des Graphen aus Abbildung 1.1(a) kreisartiger Kaktus als Grundlage, (b) kreuzungsartiger Kaktus als Grundlage, (c) Kriterien zur Verringerung von $|V'|$ und $|E'|$ angewendet, (d) Clusterhierarchie zu (c) (schraffierte Knoten: Kreuzungsknoten, weiße Knoten: Kreisknoten)

5.1 Betrachtungen für Baum–Cluster–Graphen

In diesem Abschnitt gehen wir von folgender Situation aus. Sei G ein planarer Graph und $\mathcal{R}$ die kreuzungsartige Kaktusdarstellung der minimalen Schnitte von G. Weiter sei $(G, \mathcal{T})$ der zugehörige Kaktus–Cluster–Graph, wobei $\mathcal{T} = (F, W, \varphi)$ die mit Algorithmus 3.1 erhaltene Clusterhierarchie ist. Des Weiteren sei $K \subseteq W$ die Menge der Kreisknoten der Clusterhierarchie. Nach Voraussetzung ist jeder Cluster $\nu \in T - root$ mit genau zwei Unterclustern ein Kreuzungsknoten, ebenso der $root$–Cluster, falls er drei Untercluster besitzt.

Für jeden Cluster, der ein 3–Kreuzungsknoten ist, wollen wir entscheiden können, ob eine Transformation in einen Kreisknoten eine Reduzierung der Kanten- und Knotenzahl der zugehörigen Kaktus–Clusterzeichnung des durch $V(\mu)$ induzierten Subgraphen von G führt. Dazu müssen wir im ersten Schritt unterscheiden, ob μ die Wurzel ist oder nicht, und die Konstellation dessen Untercluster betrachten.

5.1.1 Betrachtungen für alle Cluster $\mu \neq root$

Sei μ nicht die Wurzel und habe die beiden Nachfolger ν_1 und ν_2. Weiter seien $V_1 = V(\nu_1)$, $V_2 = V(\nu_2)$ und $V_0 = V \setminus (V_1 \cup V_2)$. Die Kanten, zwischen den jeweiligen Mengen seien mit

$$E_{i,j} = \{\{v, w\} \in E \mid v \in V_i,\ w \in V_j\}$$

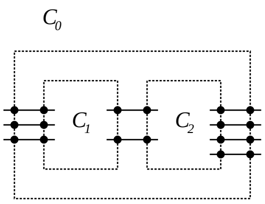
für $i = 0, 1, 2$ und $j = (i + 1) \mod 3$. Wir werden nun die verschiedenen Kombinationen der Untercluster von μ diskutieren. Die Ergebnisse werden in Tabelle 5.1 zusammengefasst.

ν_1 und ν_2 sind Kreuzungsknoten

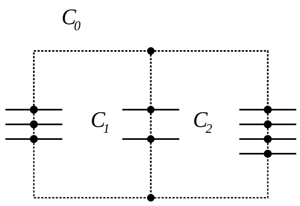
Seien desweiteren ν_1 und ν_2 Kreuzungsknoten. Stellen wir die Veränderung der Anzahl von Knoten und Kanten bei der Transformation von μ in einen Kreisknoten auf. Die Veränderung der Kanten- und Knotenzahl von G' kann man Abbildung 5.2(a) und Abbildung 5.2(b) entnehmen. Für jedes $i = 1, 2, 3$ werden jeweils $|E_{i,(i+1)\text{mod } 3}|$ Graphkanten und Knoten und $|E_{i,(i+1)\text{mod } 3}| - 1$ Clustergrenzkanten eingespart, wobei zwei Clustergrenzknoten hinzukommen. Die Veränderung der Knotenzahl $\Delta_{|V'|}$ und die Veränderung der Kantenzahl $\Delta_{|E'|}$ beträgt jeweils

$$\begin{aligned}
\Delta_{|V'|} &= 2 - (|E_{0,1}| + |E_{1,2}| + |E_{2,0}|) \quad \text{und} \\
\Delta_{|E'|} &= 3 - (2|E_{0,1}| + 2|E_{1,2}| + 2|E_{2,0}|)\,.
\end{aligned}$$

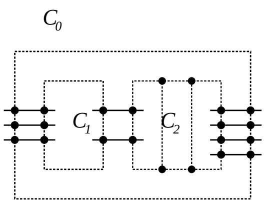
Es gilt $E_{i,j} \neq \emptyset$, für $i = 0, 1, 2$ und $j = i + 1 \bmod 3$, und somit $\Delta_{|V'|} < 0$ und $\Delta_{|E'|} < 0$. Das heißt, dass eine Transformation von μ in einen Kreisknoten zu einer Verkleinerung von G' führt.



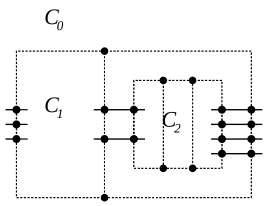
(a) fig:nonroot-nnn-j



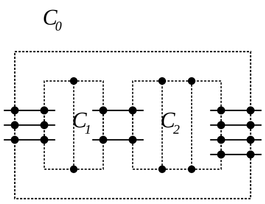
(b) fig:nonroot-nnn-c



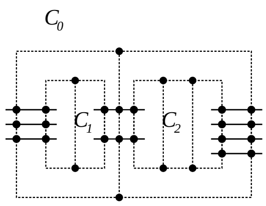
(c) fig:nonroot-nnc-j



(d) fig:nonroot-nnc-c



(e) fig:nonroot-ncc-j



(f) fig:nonroot-ncc-c

Abbildung 5.2:

C_1	C_2	Veränderung	Bedingung
nKK	nKK	$\Delta_{\lvert V'\rvert}$=$2-(m_{0,1}+m_{1,2}+m_{2,0})$	falls ω Kreuzungsknoten,
		$\Delta_{\lvert E'\rvert}$=$3-2(m_{0,1}+m_{1,2}+m_{2,0})$	oder $m_{1,2} \geq 2$ (oder so)
nKK	KK	$\Delta_{\lvert V'\rvert}$=$2-m_{0,1}$	$m_{0,1} \geq 2$
		$\Delta_{\lvert E'\rvert}$=$3-2m_{0,1}$	
KK	KK	$\Delta_{\lvert V'\rvert}$=$2+m_{1,2}$	nie
		$\Delta_{\lvert E'\rvert}$=$3+2m_{1,2}$	

Tabelle 5.1: Veränderung der Knotenzahl bei Transformation von C_0 in einen Kreisknoten (abhängig von der Konstellation und falls C_0 nicht die Wurzel ist). Dabei gilt $m_{i,j} = |E_{i,j}|$.

Jeweils ein Kreuzungs- und ein Kreisknoten

Sei nun o. E. ν_1 ein Kreuzungs- und ν_2 ein Kreisknoten. Wird μ in einen Kreisknoten transformiert, so werden jeweils $|E_{0,1}|$ Graphkanten und Clusterknoten, sowie $|E_{0,1}| - 1$ Clustergrenzkanten eingespart. Des Weiteren kommen 2 Clusterknoten und 2 Clustergrenzkanten hinzu (s. Abbildung 5.2(c) und Abbildung 5.2(d)). Für die Veränderungen in G' ergibt sich

$$\begin{aligned} \Delta_{|V'|} &= 2 - |E_{0,1}| \quad \text{und} \\ \Delta_{|E'|} &= 3 - 2|E_{0,1}|. \end{aligned}$$

Somit erhält man für $|E_{0,1}| \geq 2$ eine Verkleinerung von G'.

ν_1 und ν_2 sind Kreisknoten

Für den Fall, dass die beiden Untercluster von μ Kreisknoten sind, führt eine Konvertierung von mu in einen Kreisknoten stets zu einer Veröߟerung von G'. In diesem Fall werden ausschließlich zusätzliche Clusterkanten und -knoten erstellt. Die Bilanz sieht wie folgt aus,

$$\begin{aligned} \Delta_{|V'|} &= 2 + |E_{1,2}| \quad \text{und} \\ \Delta_{|E'|} &= 3 + 2|E_{1,2}|. \end{aligned}$$

Siehe hierzu auch die Abbildungen 5.2(e) und 5.2(f).

5.1.2 Betrachtungen für die Wurzel *root*

Nun betrachten wir die Veränderung von G' bei der Transformation von *root* in einen Kreisknoten. Es folgt eine Betrachtung abhängig von der Anzahl der Kreisknoten unter den Unterclustern ν_1, ν_2, ν_3 von *root*.

Untercluster von *root* sind Kreuzungsknoten

Seien die Untercluster ν_1, ν_2, ν_3 von *root* Kreuzungsknoten. Die Veränderung der Größe von G' hängt nun von der gewählten Zeichnung ab. Die Zeichnung in Abbildung 5.3(b) stellt die Cluster in der Reihenfolge ν_1, ν_2, ν_3 dar. Wird *root* in einen Kreisknoten transformiert, so werden $|E_{1,2}|$ bzw. $|E_{2,3}|$ viele Graphkanten entfernt, das sind die Kanten, die die Cluster ν_1 und ν_2 bzw. ν_2 und ν_3 verbinden, und genausoviele Clustergrenzknoten. Des Weiteren fallen $|E_{1,2}| - 1$ bzw. $|E_{2,3}| - 1$ Clustergrenzkanten weg. Unabhängig von der Kantenanzahl kommen 4 Clustergrenzknoten und 4 Clustergrenzkanten hinzu. Dies ergibt eine Veränderung von

$$\begin{aligned} \Delta_{|V'|} &= 4 - (|E_{1,2}| + |E_{2,3}|) \quad \text{und} \\ \Delta_{|E'|} &= 6 - 2(|E_{1,2}| + |E_{2,3}|). \end{aligned}$$

root besitzt genau einen Kreisknoten als Untercluster

Habe nun *root* zwei Untercluster, die Kreuzungsknoten sind und einen Untercluster, der ein Kreisknoten ist. In diesem Fall sind wieder mehrere Zeichnungen möglich. Abbildung 5.3(d) zeigt eine Zeichnung, bei der der als Kreisknoten dargestellte Cluster am Rand liegt, wobei dieser in Abbildung 5.4(b) in der Mitte liegt. Während es im ersten Fall je nach Voraussetzung möglich ist, Kanten und Knoten in der Zeichnung einzusparen, entstehen im zweiten Fall ausschließlich zusätzliche Kanten und Knoten.

Betrachten wir den ersten Fall näher. Sei ν_3 der Kreisknoten. Bei der Transformation von *root* in einen Kreisknoten werden $|E_{1,2}|$ Graphkanten und Clustergrenzknoten sowie $|E_{1,2}| - 1$ Clustergrenzkanten entfernt. Andererseits kommen $|E_{3,1}|$ Graphkanten und Clustergrenzknoten hinzu, sowie $|E_{3,1}| - 1$ Clustergrenzkanten. Einmalig entstehen zusätzlich 4 Clustergrenzknoten und 6 Clustergrenzkanten. Dies ergibt die Bilanz

$$\begin{aligned} \Delta_{|V'|} &= 4 - |E_{1,2}| + |E_{3,1}| \quad \text{und} \\ \Delta_{|E'|} &= 6 - 2|E_{1,2}| + 2|E_{3,1}|. \end{aligned}$$

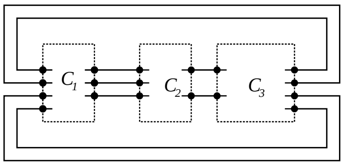
Die Bedingung für eine Verkleinerung von G', d. h. $\Delta_{|V'|}, \Delta_{|E'|} < 0$, ist

$$|E_{1,2}| \;>\; 4 + |E_{3,1}|.$$

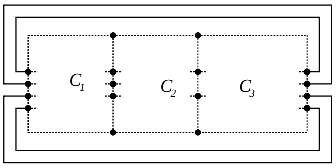
Im zweiten Fall entstehen 4 Clustergrenzknoten und 6 Clustergrenzkanten, unabhängig von der Anzahl der Kanten, die die Cluster verbinden. Der Algorithmus zur Erstellung der Zeichnung sollte daher die Anordnung der Cluster berücksichtigen.

root besitzt mindestens zwei Kreisknoten als Untercluster

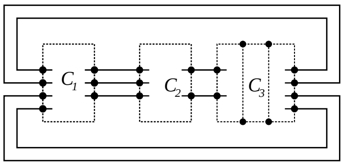
Dies ist in den Abbildungen 5.4(a) bis 5.4(h) gezeigt. Hier enstehen ausschließlich zusätzliche Kanten.



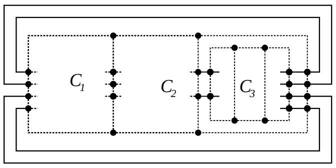
(a) fig:root-jjj-j



(b) fig:root-jjj-c

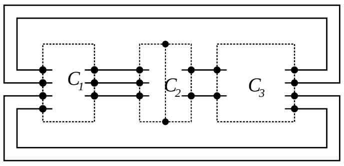


(c) fig:root-jjc-j

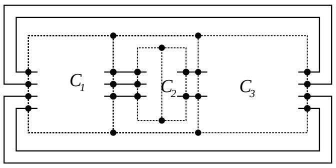


(d) fig:root-jjc-c

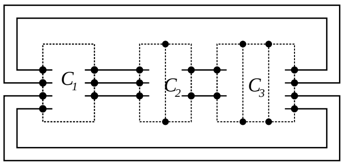
Abbildung 5.3: Von links nach rechts wird der *root*–Cluster von einem Kreuzungsknoten in einen Kreisknoten transformiert. In Teil (a) und (b) sind alle Nachfolger von *root* Kreuzungsknoten. In Teil (c) und (d) ist nur der „rechte“ Untercluster ein Kreisknoten. Abhängig von der Anzahl der Kanten, die die Cluster verbinden, kann die Transformation von *root* in einen Kreisknoten Kanten und Knoten einsparen.



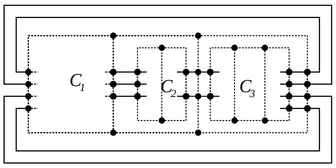
(a) fig:root-jcj-j



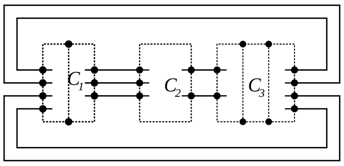
(b) fig:root-jcj-c



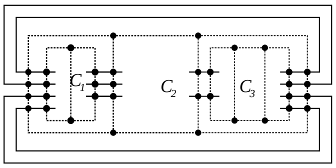
(c) fig:root-jcc-j



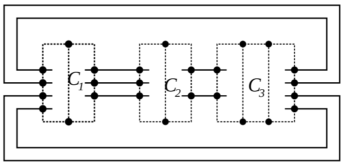
(d) fig:root-jcc-c



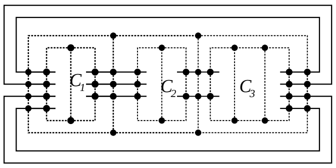
(e) fig:root-cjc-j



(f) fig:root-cjc-c



(g) fig:root-ccc-j



(h) fig:root-ccc-c

Abbildung 5.4: Von links nach rechts wird der *root*–Cluster von einem Kreuzungsknoten in einen Kreisknoten transformiert. In Teil (a) ist nur der „mittlere“ Untercluster ein Kreisknoten. In Teil (c) und (e) sind jeweils genau zwei Untercluster Kreisknoten. In Teil (g) sind alle Untercluster von *root* Kreisknoten. In allen vier Fällen ist es ungünstiger, *root* als Kreisknoten zu wählen.

C_1	C_2	C_3	Veränderung	Verkleinerung
nKK	nKK	nKK	$\Delta_{\lvert V'\rvert}$=$4 - m_{1,2} - m_{2,3}$ $\Delta_{\lvert E'\rvert}$=$6 - 2\ m_{1,2} - 2\ m_{2,3}$	$m_{1,2} + m_{2,3} \geq 4$
nKK	nKK	KK	$\Delta_{\lvert V'\rvert}$=$4 - m_{1,2} + m_{3,0}$ $\Delta_{\lvert E'\rvert}$=$6 - 2m_{0,1} + 2m_{3,0}$	$m_{1,2} > m_{3,0} + 4$
nKK	KK	nKK	$\Delta_{\lvert V'\rvert}$=4 $\Delta_{\lvert E'\rvert}$=6	nie
nKK	KK	KK	$\Delta_{\lvert V'\rvert}$=$4 + m_{2,3} + m_{3,1}$ $\Delta_{\lvert E'\rvert}$=$6 + 2m_{2,3} + 2m_{3,1}$	nie
KK	nKK	KK	$\Delta_{\lvert V'\rvert}$=$4 + 2m_{3,1}$ $\Delta_{\lvert E'\rvert}$=$6 + 4m_{3,1}$	nie
KK	KK	KK	$\Delta_{\lvert V'\rvert}$=$4 + m_{1,2} + m_{2,3} + 2m_{3,1}$ $\Delta_{\lvert E'\rvert}$=$6 + 2m_{1,2} + 2m_{2,3} + 4m_{3,1}$	nie

Tabelle 5.2: Veränderung der Knotenzahl bei Transformation von C_0 in einen Kreisknoten (abhängig von der Konstellation und falls C_0 die Wurzel ist). Dabei gilt $m_{i,j} = |E_{i,j}|$.

5.2 Betrachtungen für ungewichtete Graphen

Bei ungewichteten planaren Graphen vereinfachen sich die Ergebnisse, die in Tabelle 5.1 und 5.2 zusammengefasst sind. Ist G ein ungewichteter planarer Graph, der Kuchenpartitionen der Größe mindestens drei besitzt, so gilt für den Wert eines minimalen Schnittes $\lambda \in \{2, 4\}$. Dies folgt aus $\lambda \leq 5$ (s. Korollar 1.4) und der Voraussetzung, dass $\lambda/2$ ganzzahlig ist. Behalten wir die Bezeichnungen aus Abschnitt 5.1 bei, so gilt $|E_{i,j}| = \lambda/2$ für i, j wie oben gewählt. Wir betrachten die beiden Fälle für λ getrennt.

Fall $\lambda = 2$

Sei $\lambda = 2$ der Wert eines minimalen Schnittes in G. Ist $\mu \neq root$ ein Clusterknoten mit den Unterclustern ν_1 und ν_2, so gilt $|E_{i,j}| = 1$ für $i, j \in \{0, 1, 2\}$ und $i \neq j$. Wird μ in eine Kreisknoten transformiert, so können Einsparungen für G' nur dann erzielt werden, wenn ν_1 und ν_2 Kreuzungsknoten sind. Besitzt der *root*–Cluster genau drei Untercluster, so ergeben sich in keinem Fall Einsparungen. Hierzu wären mindestens 6 Kanten nötig, die die Untercluster von *root* verbinden.

2. Fall $\lambda = 4$

Sei nun $\lambda = 4$ der Wert eines minimalen Schnittes in G und wieder $\mu \neq root$ ein Clusterknoten mit den Unterclustern ν_1 und ν_2. Es gilt $|E_{i,j}| = 2$ für $i, j \in \{0, 1, 2\}$

und $i \neq j$. Damit ist garantiert, dass eine Transformation von μ in einen Kreisknoten den Graph G' der Clusterzeichnung verkleinert sobald einer der Untercluster von μ ein Kreuzungsknoten ist. Ist *root* ein Clusterknoten mit den drei Unterclustern ν_1, ν_2, ν_3. Bei einer Transformation von *root* in einen Kreisknoten erhält man garantiert Einsparungen für G', denn $|E_{1,2} \cup E_{2,3}| = 4$ ist erfüllt.

Algorithmus 5.1 Es werden die Cluster von $(G, \mathcal{T})$ von den Blättern zu Wurzel hin untersucht, ob eine Darstellung als Kreis- oder Kreuzungsknoten die Größe von $G' = (V', E')$ gering zu halten.

Voraussetzung: Seien $(G, \mathcal{T})$ ein kreuzungsartiger Kaktus–Clustergraph und *root* die Wurzel der Clusterhierarchie mit den Unterclustern $\nu_1, \ldots, \nu_r$, für ein $r \geq 3$, wobei genau s Untercluster Kreisknoten seien.
Weiter seien $E_{i,j} = \{\{v, w\} \in E \mid v \in V(\nu_i),\ w \in V(\nu_j)\}$, für $i = 1, \ldots, r$ und $j = (i \pm 1) \bmod r$, die Kantenmengen, die $V(\nu_i)$ mit $V(\nu_j)$ verbinden.

Rückgabe: $\nu \in \mathcal{T}$ wird zu einem Kreisknoten, falls dadurch die Anzahl an Kanten und Knoten in der zugehörigen Cluster–Kaktus–Zeichnung reduziert werden kann.

für $\nu = \nu_1, \ldots, \nu_r$ **führe aus**
  falls ν mindestens zwei Untercluster hat **dann führe aus**
    rufe Prozedur 5.2 mit ν als Argument auf.
falls $r \equiv 3$ **und** $s \equiv 1$ **und** $|E_{1,2}| + |E_{2,3|} \geq 4$ **dann führe aus**
  mache *root* zu Kreisknoten;
falls $r \equiv 3$ **und** $s \equiv 0$ **und** $|E_{1,2}| > |E_{3,0|} + 4$ **dann führe aus**
  mache *root* zu Kreisknoten;

Prozedur 5.2 Entscheidet, ob $\mu \neq root$ ein Kreisknoten oder ein Krezungsknoten sein

Voraussetzung: Seien $(G, \mathcal{R})$ Kaktus–Clustergraph und *root* die Wurzel der Clusterhierarchie. Als Argument wird der Cluster $\mu \neq root$ übergeben, der ein Kreuzungsknoten ist und die Untercluster $\nu_1, \ldots, \nu_r$, für ein $r \geq 2$, besitze, wobei s Untercluster Kreisknoten seien. Desweiteren seien $V_i = V(\nu_i)$, für $i = 1, \ldots, r$, und $V_0 = V \setminus \bigcup_{i=1}^{r} V_i$ und $E_{i,j} = \{\{v, w\} \in E \mid v \in V_i,\ w \in V_j\}$, für $i = 0, \ldots, r$ und $j = (i \pm 1) \bmod r + 1$.

Rückgabe: Falls $r \equiv 3$, so wird μ zu einem Kreisknoten, falls dadurch Cluster–Kaktus–Zeichnung reduziert werden kann.

für alle $\nu = \nu_1, \ldots, \nu_r$ **führe aus**
    falls ν mindestens zwei Cluster als Nachfolger hat **dann führe aus**
        rufe Prozedur 5.2 mit ν als Argument auf.
falls $r \equiv 2$ **und** $s \equiv 0$ **dann führe aus**
    mache μ zu Kreuzungsknoten;
falls $r \equiv 2$ **und** $s \equiv 1$ **dann führe aus**
    $i \leftarrow$ Index des Untercluster von μ, der Kreisknoten ist;
    falls $|E_{i,0}| \geq 2$ **dann führe aus**
        mache μ zu Kreuzungsknoten;

Kapitel 6

Graph der Kaktusdarstellung

Enthält ein Graph Kuchenpartitionen der Größe mindestens vier, so ist bei der Visualisierung seiner minimalen Schnitte ein zusätzlicher Aufwand zu betreiben (s. Kapitel 3). Dies ist genau dann der Fall, wenn die Kaktusdarstellung der minimalen Schnitte einen Kreis der Länge mindestens vier besitzt, oder anders ausgedrückt, wenn die kreuzungsartige Kaktusdarstellung der minimalen Schnitte keine Kreise enthält. Nun liegt es nahe zu fragen, welche Klasse von Graphen keine Kuchenpartitionen der Größe mindestens vier besitzen. Eine maximale Klasse, die diese Forderung erfüllt, wurde konnte bisher nicht bestimmt werden. Dies hält uns nicht davon ab, ein paar Graphklassen zu betrachten, deren minimale Schnitte mit dem in Kapitel 3 vorgestellten Verfahren visualisiert werden können.

Definition 6.1 Seien $G = (V, E)$ ein Graph und $\mathcal{R} = ((W, F), \varphi)$ eine Kaktusdarstellung von $\mathcal{C} \subseteq \mathcal{C}(G)$. $\mathcal{R}$ heißt *Baumdarstellung* von $\mathcal{C}$, falls (W, F) ein Baum ist.

Besitzt eine normale Kaktusdarstellung keine Kreise der Länge größer vier, so kann man sie nach Satz 1.1 in eine Baumdarstellung überführen.

Definition 6.2 Seien $G = (V, E)$ ein Graph, $G^* = (V^*, E^*)$ der Dualgraph zu G und $\{X, \overline{X}\} \in \mathcal{C}$. Dann bezeichne $E^*_X \subseteq E^*$ die Menge der Dualkanten, die $\{X, \overline{X}\}$ induzieren und $V^*_X \subseteq V^*$ deren Endknoten. Der Subgraph $K^*_X = (V^*_X, E^*_X)$ heißt *Schnittkreis*.

Da sich die Schnittkreise sich kreuzender minimaler Schnitte in zwei Facetten vom Grad mindestens vier kreuzen, scheinen Graphen erfolgversprechend zu sein, bei denen nur „wenige“ Facette von Grad ≥ 4 vorkommen, oder, im ungewichteten Fall, diese eine groß genuge Distanz im Dualgraphen besitzen.

6.1 Ungewichtete Graphen

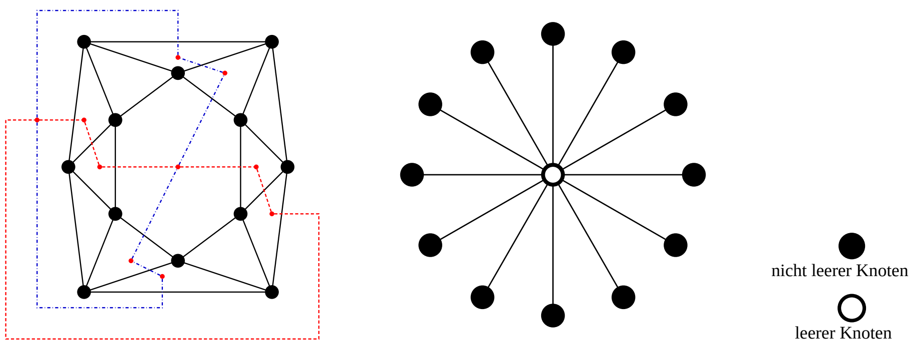
6.1.1 Facetten vom Grad ≥ 4

Sei $G = (V, E)$ ein ungewichteter planarer Graph, der die Eigenschaft hat, dass jeder Knoten $v \in V$ auf höchstens einer Facette vom Grad ≥ 4 liegt, und $G^* = (V^*, E^*)$ der zugehörige Dualgraph. Seien weiter $\{X, \overline{X}\}, \{Y, \overline{Y}\} \in (G)$ zwei sich kreuzende minimale Schnitte, und K_X^* bzw. K_Y^* die zugehörigen Schnittkreise im Dualgraph, die sich in $f_1, f_2 \in V^*$ kreuzen.

Die Schnittkreise haben Länge 2 oder 4, da dies die möglichen Werte λ eines minimalen Schnittes sind.

Des Weiteren muss der Grad von f_1 und f_2 mindestens vier sein, da die Schnittkreise kantendisjunkt sind. Nach Voraussetzung gibt es keinen Knoten in G, der sowohl auf f_1 und auf f_2 liegt. Dies schließt den Fall $\lambda = 2$ schon aus. Im Fall $\lambda = 4$ sei $p: \; f_1, f', f_2$ ein Pfad von f_1 nach f_2 auf einem der Schnittkreise. Die Facette f' hat aber nach Definition Grad 3, so dass es einen Knoten gibt, der auf allen drei Facetten liegt. Dies ist ein Widerspruch zur Voraussetzung.

Daraus folgt, dass der Wert eines minimalen Schnittes mindestens sechs ist. Das ist aber ein Widerspruch zu der Tatsache, dass der Wert eines minimalen Schnittes in planaren Graphen höchstens fünf ist.

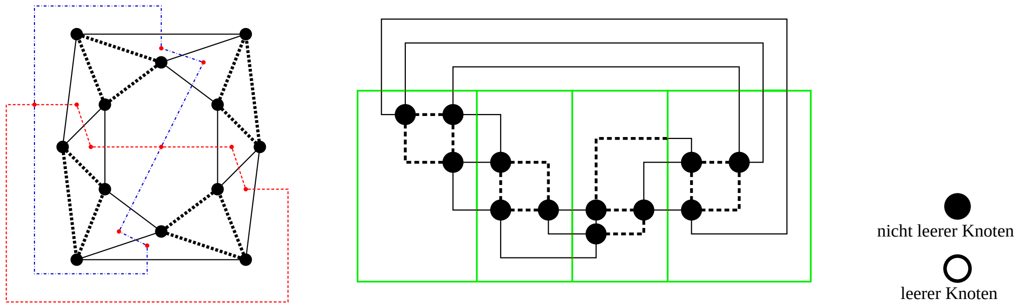


(a) (b)

Abbildung 6.1: (a) Der Graph besitzt keine Knoten, die zu mehr als einer Facette vom Grad ≥ 4 gehören. Die eingezeichneten Schnittkreise induzieren keine minimalen Schnitte. (b) Der zugehörige Kaktus ist kreisfrei.

Gewichteter Fall führt nicht zum Ziel

Lässt man Kantengewichte größer eins zu, so kann man Kuchenpartitionen der Größe drei nicht verhindern. Dies ist in Abbildung 6.3 gezeigt. Der Wert eines minimalen Schnittes beträgt $\lambda = 6$.



(a) (b)

Abbildung 6.2: (a) Eine gewichtete Variante des Graphen aus Abbildung 6.1(a). Dabei haben die breiten, gestrichelten Kanten Gewicht 3. (b) Die zugehörige Kaktus–Cluster–Zeichnung verdeutlicht, die Kuchenpartition der Größe vier besitzt.

Schwächere Forderung führt nicht zum Ziel

Abbildung 6.3 (a) zeigt einen Graphen, bei dem keine Kante zu zwei Facetten vom Grad ≥ 4 gehört. Dies ist eine Abschwächung zu der oben genannten Forderung, da es hier Knoten gibt, die zu zwei Facetten vom Grad ≥ 4 gehören. Die zugehörige Kaktusdarstellung der minimalen Schnitte (s. Abbildung 6.3(b)) gezeigt, besitzt einen Kreis der Länge 8.

6.1.2 Brückengraphen mit $\lambda \in \{2, 4\}$

Sei $G = (V, E)$ ein ungerichteter Graph und $C = v_1, \ldots, v_r$ ein Kreis in G. Unter einer *Brücke von C* verstehen wir einen Pfad P, der zwei Knoten v_i und v_j, für $i < j - 1$, in C verbindet und dessen Länge kürzer ist als die kürzere der beiden Strecken zwischen v_i und v_j auf dem Kreis C. Anders ausgedrückt bedeutet dies

$$d_G(v_i, v_j) < d_C(v_i, v_j).$$

G heißt *Brückengraph*, falls jeder Kreis eine Brücke besitzt. Das bedeutet, dass jeder Kreis zwei Knoten besitzt, die durch einen Pfad verbunden werden können,

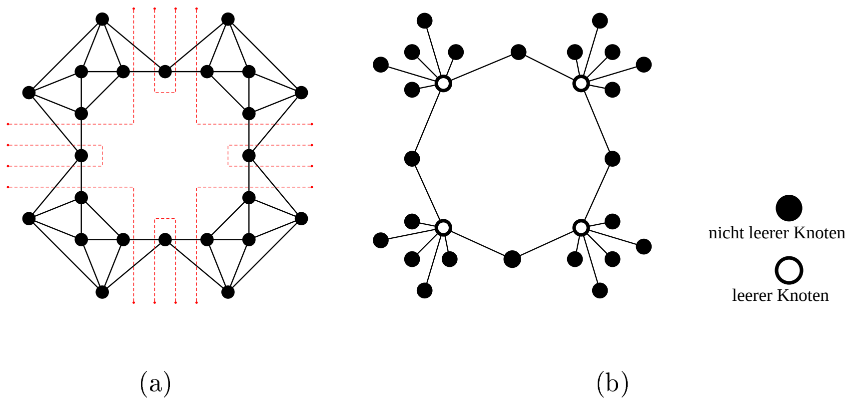


Abbildung 6.3: (a) Ein Graph, der zwar keine adjazenten Facetten vom Grad ≥ 4, aber dennoch sich kreuzende minimalen Schnitte besitzt. (b) Der Kaktus enthält einen Kreis.

dessen Länge kürzer ist als deren Distanz auf dem Kreis. Ein Satz aus [7] besagt, dass für einen Kreis C und einen Knoten $v \in V(C)$ gilt:

- die Nachbarn von v sind adjazent, oder
- C besitzt eine Brücke P, die in v beginnt.

Besitzt C eine Brücke, so besitzt C eine *echte* Brücke P, in dem Sinn, dass P und C bis auf die Anfangs- und Endknoten von P disjunkt sind.

Wir wollen zeigen, dass ein ungewichteter Brückengraph G mit Wert eines minimalen Schnittes $\lambda \in \{2, 4\}$ eine Baumdarstellung der minimalen Schnitte besitzt. Angenommen, eine normale Kaktusdarstellung der minimalen Schnitte von G besitze einen Kreis der Länge r. Dazu gehöre eine maximale Kuchenpartition $V_1, V_2, V_3, U_4, \ldots, U_r$. Insbesondere erhalten wir mit $V_4 = \bigcup_{i=4}^{r} U_i$ eine Kuchenpartition (V_1, V_2, V_3, V_4) der Größe 4. Sei nun $C = v_1, \ldots, v_r, v_1$ ein Kreis, der aus jeder Menge $V_1, \ldots, V_4$ in dieser Reihenfolge mindestens einen Knoten besitzt und unter den Kreisen mit dieser Eigenschaft minimale Länge hat. Offensichtlich gilt $r \geq 4$ und nach Voraussetzung besitzt C eine echte Brücke P. Sei nun o. E. $v_1 \in V_1$ der Punkt, in dem P beginne, und v_j der Endknoten von P, mit $j > 2$. Wir unterscheiden nun vier Fälle, abhängig von der Partition, in der v_j liegt.

$v_j \in V_1$: Ersetzt man das Wegstück $v_1, \ldots, v_j$ auf C durch die Brücke P, so erhält man wieder einen Kreis, der eine kleinere Länge als C hat und auch mindestens einen Knoten aus $V_1, \dots, V_4$ besitzt. Das ist ein Widerspruch dazu, dass die Länge von C minimal ist unter dieser Eigenschaft.

$v_j \in V_2$: Ist $\lambda = 2$, so kann dieser Fall nicht eintreten, da sonst zwei Kanten V_1 und V_2 verbinden würden, im Widerspruch zu $c(V_1, V_2) = \lambda/2$ (s. Lemma 1.6). Im Fall $\lambda = 4$ kann man den Kreis C wieder mit Hilfe von P abkürzen.

$v_j \in V_3$: Hier ist nun zu unterscheiden, ob P Knoten besitzt, die in V_2 oder V_4 liegen, oder nicht. Im ersten Fall lässt sich C wie gehabt abkürzen ohne die geforderte Eigenschaft zu verletzen. Besitzt P hingegen nur Knoten aus V_1 und V_3, so gilt $c(V_1, V_3) \neq 0$, was im Widerspruch zu Lemma 1.6 steht.

$v_j \in V_4$: Dieser Fall verläuft analog zu dem Fall $v_j \in V_2$.

Somit ist die Annahme, dass eine normale Kaktusdarstellung des Brückengraphen G einen Kreis der Länge mindestens vier besitzt, widerlegt. G besitzt also eine Baumdarstellung der minimalen Schnitte.

6.2 Gewichtete Graphen

6.2.1 $\lambda \in \mathbb{N}$ ungerade

Sei G ein gewichteter Graph mit ganzzahligen Kantengewichten und der Wert λ eines minimalen Schnittes in G sei ungerade. Weiter sei $\mathcal{R} = ((W, F), \varphi)$ eine Kaktusdarstellung der minimalen Schnitte $\mathcal{C}(G)$. Wir zeigen, dass der Kaktus G keine sich kreuzenden minimalen Schnitte besitzt. Andernfalls seien $\{X, \overline{X}\}, \{Y, \overline{Y}\} \in \mathcal{C}(G)$ sich kreuzende minimale Schnitte. Aus Lemma 1.6 folgt

$$\lambda(X \cap Y, \overline{X} \cap \overline{Y}) = \lambda/2.$$

Dies steht im Widerspruch zu der Tatsache, dass G nur ganze Kantengewichte besitzt. Somit besitzt (W, F) keine Kreise der Länge größer 3 und $\mathcal{R}$ kann in eine Baumdarstellung der minimalen Schnitte von G transformiert werden.

6.2.2 Chordale Graphen

Sei $G = (V, E)$ ein gewichteter, ungerichteter Graph. Eine Brücke, die aus einer Kante besteht heißt *Sehne* (englisch: *chord*). G heißt *chordaler Graph* , falls jeder Kreis der Länge größer 3 eine Sehne besitzt.

Sei G ein chordaler Graph und R sei der Graph einer normalen Kaktusdarstellung der minimalen Schnitte von G. Angenommen, R besitzt einen Kreis der Länge größer 3. Dann gibt es in G eine (nicht notwendig maximale) Kuchenpartition $\pi = (V_1, V_2, V_3, V_4)$. Sei $C = v_1, \dots .v_r$ ein Kreis in G, der aus jeder der Mengen

$V_1, \ldots, V_4$, in dieser Reihenfolge, mindestens einen Knoten besitzt und minimale Länge besitzt. C besitzt mindestens vier Knoten, hat also eine Sehne. Liegen die Endknoten der Sehne innerhalb einer der Teilmengen V_i, für $i = 1, \ldots, 4$, oder in Teilmengen mit aufeinanderfolgenden Indizes, so ist dies ein Widerspruch zur minimalen Länge von C. Verbindet die Sehne die beiden Teilmengen V_1 und V_3 bzw. V_2 und V_4, so erhält man einen Widerspruch zu $c(V_i, V_i + 1) = 0$ für $i = 1, 2$.

6.3 Diskussion

Wie wir gesehen haben, besitzen gewichtete chordale Graphen eine Baumdarstellung der minimalen Schnitten. Es konnte leider weder bewiesen noch widerlegt werden, ob sich diese Eigenschaft auch auf gewichtete Brückengraphen verallgemeinern lässt.

Desweiteren ist zu bemerken, dass der Graph aus Abbildung 6.1(a) kein Brückengraph ist. Der Kreis auf der äußeren Facette besitzt keine Brücke. Dies ist angesichts der genringen Größe der Graphen recht leicht nachvollziehbar. Interessant ist, dass dieser Graph nicht einmal zur Klasse der *cop–win* Graphen gehört. Es gibt mehrere äquivalente Definitionen für cop–win Graphen. Eine davon stammt von Nowakowski und Winkler [19], aus der Spieltheorie, und betrachtet einen Graphen G als Spielfeld mit den zwei Figuren (*robber* und *cop*). Dabei werden die Knoten des Graphen als Felder identifiziert und die zu einem Knoten v adjazenten Kanten beschreiben einen möglichen Zug, den eine Figur von v aus tätigen kann (Aussetzen ist auch erlaubt — dies modelliert man durch Einfügen von Schleifen). Jede Figur macht abwechselnd einen Zug, wobei der *robber* beginnt. Gelingt es dem *cop* bei jeder Anfangsaufstellung, den *robber* zu fangen (d. h. am Ende seines Zuges auf dem Feld des *robber* zu stehen), so heißt der zugehörige Graph *cop–win*. Wie man sieht, schafft es ein *cop* in Abbildung 6.1(a) niemals, den *robber* zu fangen.

Amsted und Farber [1] identifizierten Brückengraphen als cop–win–Graphen, die keine induzierten Kreise der Länge 4 bzw. 5 enthalten.

Diese Bemerkungen sollen Hoffnung machen bei der Suche nach Graphen mit Baumdarstellung der minimalen Schnitte.

Kapitel 7

Zusammenfassung und Aussichten

Wir haben uns mit der Visualisierung von minimalen Schnitten in planaren Graphen befasst. Wir haben gesehen, dass sich Visualisierung der minimalen Schnitte auf das Erstellen einer Clusterzeichnung reduziert, sobald die kreuzungsartige Kaktusdarstellung der minimalen Schnitte keine Kreise enthält. Während der Vorbereitung dieser Arbeit haben wir erfahren, dass das hier vorgestellte Verfahren zur Erstellung einer knickminimalen Clusterzeichnung von Lütke–Hüttmann in [17] schon behandelt wurde. Analog zu den dort beschriebenen Erweiterungen können wir in beiden Fällen die Visualisierung der minimalen Schnitte auf allgemeine planaren Graphen erweitern.

In Kapitel 5 haben wir eine Heuristik angegeben, die die Erkennbarkeit der minimalen Schnitte für den Betrachter vereinfachen soll. Sie beruht darauf, die Anzahl der Knoten und Kanten zu reduzieren, die für Zeichnung verwendet werden. Die angegebenen Kriterien sind nicht optimal, können sich die Ergebnisse sehen lassen.

In Kapitel 6 haben wir schließlich Graphklassen untersucht, die eine Baum–Cluster–Zeichnung der minimalen Schnitte zulassen und Hinweise für die Suche nach weiteren Graphklassen gegeben, die diese Eigenschaft besitzen.

Die in dieser Arbeit angefertigten Zeichnungen wurden größtenteils mit dem `C++`–Programm „`cluster`“ erstellt, das in Zusammenarbeit von Sabine Cornelsen und dem Autor entstanden ist. Es beruht auf der von Sabine Cornelsen entwickelten Implementation „`cactus`“ einer Variante des in Kapitel 2 vorgestellten Verfahrens, das die kreisartige Kaktusdarstellung der minimalen Schnitte eines Graphen berechnet. Als Grundlage dienen die `C++`–Bibliotheken LEDA[1] (Version 4.1) und AGD[2] (Version 0.9[3]), die am Max–Planck–Institut für Informatik entwickelt wurden.

[1] `http://www.mpi-sb.mpg.de/LEDA`

[2] `http://www.mpi-sb.mpg.de/AGD`

[3] Es wurde ein vom Autor entwickelter Patch angewendet, um eine Verträglichkeit mit LEDA Version 4.1 zu ermöglichen.

Index

Literaturverzeichnis

[1] R. P. Anstee, M. Farber, On bridged graphs and cop–win graphs, J. Combin. Theory B, 44 (1988), 22–28.

[2] G. Battista, P. Eades, R. Tamassia, I. G. Tollis Graph Drawing — Algorithms for the Visualization of Graphs, Prentice Hall (1999)

[3] A. Brandstädt, Van Bang Le, J. P. Spinrad, Graph calsses – a survey, *SIAM,* (1999)

[4] S. Cornelsen, U. Brandes, D. Wagner, How to Draw the Minimum Cuts of a Planar Graph, *to be published,* (2000)

[5] E. A. Dinits, A. V. Karzanov, M. V. Lomonosov, On the structure of a family of minimal weighted cuts in a graph. In A. A. Fridman (Hsg.), Studies in Discrete Optimization, 290–306, Moskau Nauka, 1976 (in russisch)

[6] M. Farber, R. E. Jamison, On local convexity in graphs, *Discrete Math.,* 66 (1987), 231-247

[7] M. Farber, Bridged graphs and geodesic convexity, *Discrete Math.,* 66 (1987), 249-257

[8] P. Eades, Q. Feng, H. Nagamochi, Drawing clustered graphs on an orthogonal grid, Journal of Graph Algorithms and Applications, 3(4):3–29

[9] L. Fleischer, Building chain and cactus representations of all minimum cuts from Hao–Orlin in the same asyptotic run time, Journal of Algorithms (1999), 33(1), 51–72

[10] H. N. Gabow, Applications of poset representation to edge connectivity and graph rigidity, Proc. 32nd IEEE Symp. Found. Comp. Sci., 1991, 812–821

[11] A. Garg und R. Tamassia, A new minimum cost flow algorithm with application to graph drawing, Graph Drawing (Proc. GD 1996)

[12] A. V. Goldberg und R. E. Tarjan, A new approach to the maximum flow problem, Journal of the ACM (1988), 35, 921–940 connectivity and graph rigidity, Proc. 32nd IEEE Symp. Found. Comp. Sci., 1991, 812–821

[13] J. Hao und J. B. Orlin, A faster algorithm for finding the minimum cut in a graph, Proc of 3rd ACM–SIAM Symp. on Discrete Algorithms (1992), 165–174

[14] D. R. Karger und C. Stein, A new approach to the minimum cut problem, Journal of the ACM (1996), 43(4), 601–640

[15] A. V. Karzanov und E. A. Timofeev, Efficient algorithm for finding all minimal edge cuts of a nonorientated graph. *Cybernetics (englische Übersetzung von Kibernetika),* (1986), 156–162

[16] G. W. Klau, Quasiorthogonales Zeichnen planarer Graphen mit wenigen Knicken, Diplomarbeit, Universität des Saarlandes, Max–Planck–Institut für Informatik, Januar 1997

[17] D. Lütke–Hüttmann, Knickminimales Zeichnen 4–planarer Clustergraphen, Diplomarbeit, Universität des Saarlandes, Max–Planck–Institut für Informatik, 2000

[18] H. Nagamochi und T. Kameda, Canonical Cactus Representation for Minimum Cuts, Japan J. Indust. Appl. Math., **11** (1994), 343–361

[19] R. Nowakowski und P. Winkler, Vertex–to–vertex pursuit in a graph, *Discrete Math.,* 43 (1983), 235–239

[20] R. Tamassia, On embedding a graph in the grid with the minimum number of bends. SIAM Journal of Computing (1987), 16, 421–444

[21] D. Wagner, Graphenalgorithmen, Mitschrieb zur Vorlesung, Universität Konstanz (1998)