Universität Karlsruhe
Fakultät für Informatik

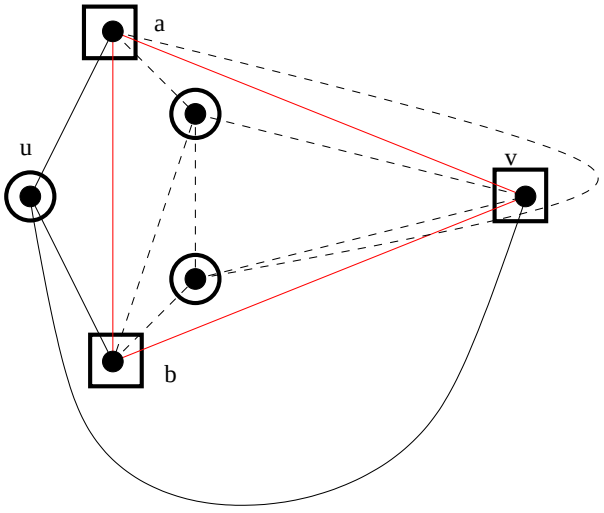
Institut für Theoretische Informatik
Algorithmik I

Vorlesungsskript

Algorithmen für planare Graphen

Dorothea Wagner

Sommersemester 2008



Inhaltsverzeichnis

Vorwort: Dieses Vorlesungsskript ist auf Basis der gleichnamigen Vorlesung entstanden, die ich im Sommersemester 1997 und im Sommersemester 2001 an der Universität Konstanz und im Sommersemester 2006 an der Universität Karlsruhe gehalten habe. Frau Lüthke hat aus meinen handschriftlichen Unterlagen eine erste Version des Skripts erstellt, und Herr Stefan Schmidt hat die Abbildungen angefertigt. Ihnen gilt mein Dank für diese Unterstützung.

1 Planare Graphen – eine anschauliche Einführung

Wir betrachten einen Graph $G = (V, E)$ mit endlicher Menge von Knoten V endlicher Menge von Kanten E.

Beispiel:

$$\begin{aligned} K_4 &= (V, E) \\ V &= \{1, 2, 3, 4\} \\ E &= \big\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\big\} \end{aligned}$$

Der K_4 ist „der“ *vollständige* (ungerichtete einfache) Graph über 4 Knoten.

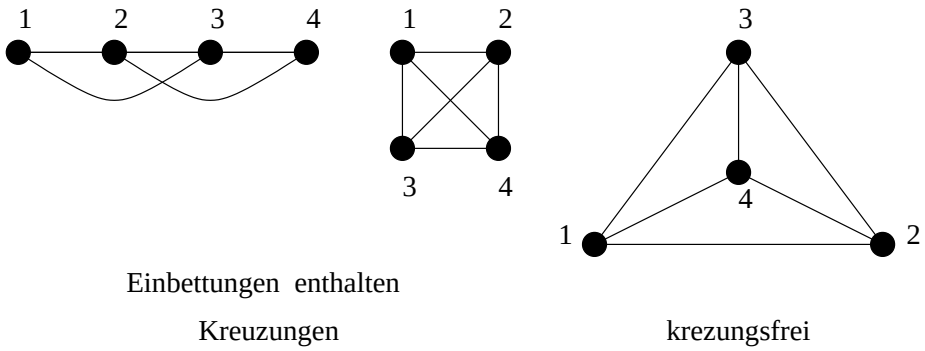


Abbildung 1.1: Verschiedene Einbettungen des K_4

Ein Graph, der kreuzungsfrei (in die Ebene) eingebettet werden kann, heißt *planar*. Der K_4 ist planar. Sind alle (endlichen, einfachen) Graphen planar? Wir „beweisen“, dass K_5 und $K_{3,3}$ nicht planar sind.

$$\begin{aligned} K_5 &= (V, E) \\ V &= \{1, 2, 3, 4, 5\} \\ E &= \big\{\{i,j\} : \ 1 \le i, j \le 5, \ i \ne j\big\} \end{aligned}$$

Angenommen, der K_5 ist planar. Bette o.B.d.A. Knoten 1 und alle zu 1 inzidenten Kanten planar ein. Es gibt die Kante $\{2,4\}$, und diese induziert eine Zerlegung der

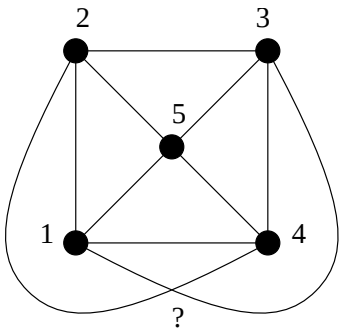


Abbildung 1.2: Einbettung des K_5. Ist die Kreuzung notwendig?

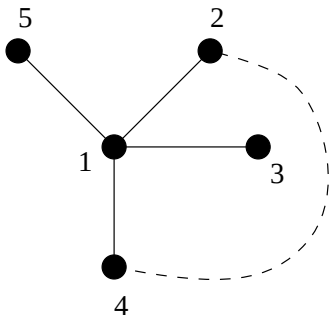


Abbildung 1.3: Zerlegung der Ebene in zwei Gebiete.

Ebene in zwei „Gebiete“, das Innere des Kreises $\big\{\{1,2\},\{2,4\},\{4,1\}\big\} \equiv 1241$ und dessen Äußeres.

Jeder Weg aus dem Inneren von 1241 in das Äußere von 1241 muss den „Rand“ (1241) kreuzen. Die Kante $\{3,5\}$ kann also nicht kreuzungsfrei gezogen werden. (Jordan'scher Kurvensatz).

Wir betrachten nun den *vollständig bipartiten* Graphen auf sechs Knoten, den $K_{3,3}$.

$$\begin{aligned} K_{3,3} &= (V,E) \\ V &= \{1,2,3,\ w,g,s\} \\ E &= \big\{\{i,j\}:\ 1 \le i \le 3,\ j \in \{w,g,s\}\big\} \end{aligned}$$

Der $K_{3,3}$ modelliert das „Wasser-Gas-Strom–Problem“, d.h. es gibt drei „Quellen“ w, g, s und drei „Häuser“ 1, 2, 3. Jedes Haus braucht Leitungen zu allen drei Quellen.

Angenommen der $K_{3,3}$ ist planar. Bette o.B.d.A. den Kreis $w1g2s3w$ kreuzungsfrei ein, wobei $\{1,s\}$ im Inneren eingebettet werde. Dann muss $\{2,w\}$ ins Äußere eingebettet werden. Unabhängig davon wie die Kante $\{3,g\}$ eingebettet wird, kreuzt sie den Kreis $w1s2w$. Damit folgt Lemma 1.1:

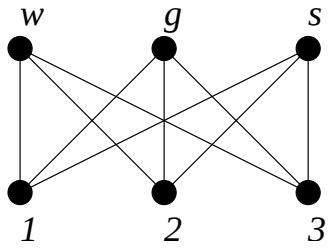


Abbildung 1.4: Eine Einbettung des $K_{3,3}$.

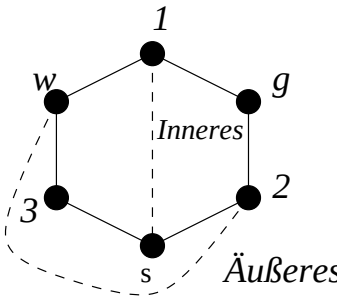


Abbildung 1.5: Inneres und Äußeres des Kreis $w1s2w$.

Lemma 1.1. K_5 *und* $K_{3,3}$ *sind nicht planar.*

Allgemeine Fragen im Zusammenhang mit planaren Graphen sind:

- Woran erkennt man planare Graphen? Kann man „effizient“ entscheiden, ob ein gegebener Graph planar ist?
- Falls man weiß, dass der gegebene Graph planar ist, kann man dann „effizient“ eine kreuzungsfreie/planare Einbettung konstruieren?
- „straight-line embedding“: Ist jeder planare Graph so planar einbettbar, dass alle Kanten gerade sind? .
- Wieviele planare Einbettungen gibt es zu einem planaren Graph?

Es ist leicht zu sehen, dass eine planare Einbettung nicht notwendig eindeutig ist, siehe Abbildung 1.6.

Das Landkartenfärbungsproblem

Färbe jedes Land so, dass benachbarte Länder verschiedene Farben bekommen. Betrachte dazu den *Nachbarschaftsgraphen* (auch Konfliktgraph genannt), bei dem jedes Land einem Knoten entspricht, und zwei Knoten durch eine Kante verbunden werden, wenn sie eine gemeinsame Grenze haben. Der Nachbarschaftsgraph „entspricht“ dem Konzept des *Dualgraph*, das im Zusammenhang mit planaren Graphen oft verwendet wird.

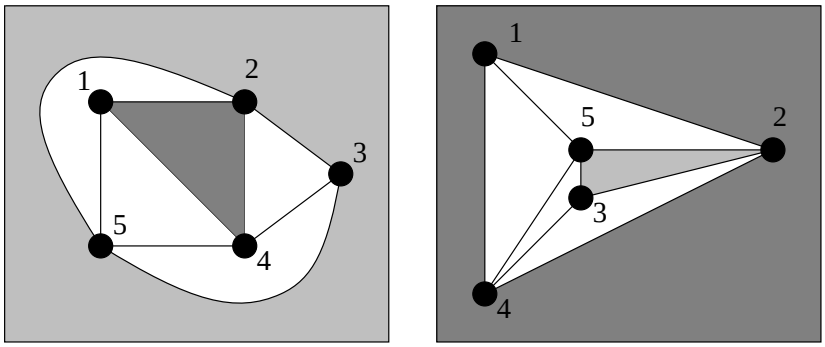


Abbildung 1.6: Das innere Gebiet von 5235 wird zum äußeren Gebiet der Einbettung gemacht.

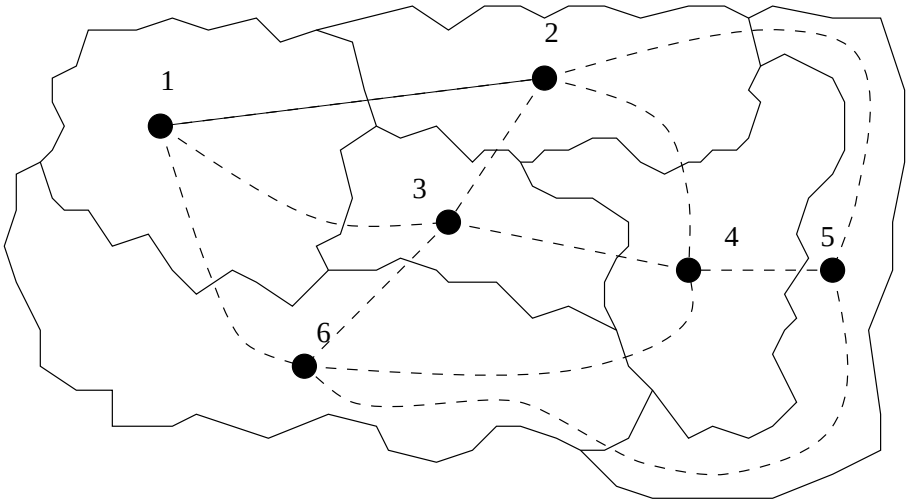


Abbildung 1.7: Landkarte mit Hauptstädten und der Nachbarschaftsgraph.

Äquivalent zum Landkartenfärbungsproblem ist dann folgendes *Graphenfärbungsproblem*: Färbe die Knoten des Nachbarschaftsgraphen so, dass zwei Knoten, die durch eine Kante verbunden sind, verschiedene Farben haben.

Der Nachbarschaftsgraph einer Landkarte ist immer planar. Um dies zu sehen betrachten wir folgende Konstruktion.

> Verbinde jede Hauptstadt sternförmig mit den „Mittelpunkten“ der verschiedenen gemeinsamen Grenzabschnitte mit anderen Ländern. Füge je zwei Verbindungen zu einer Kante zusammen.

Dies könnte man auch allgemeiner für Länder auf dem Globus machen, bzw. die Oberfläche des Globus in eine Landkarte „transformieren“, indem ein beliebiges Land zum Äußeren gemacht wird. Die Größen der Länder verändern sich dabei, d.h. das Land, das dem Äußeren entspricht, wird unendlich groß. Beachte, dass uns dies zu derselben Einbettungsfrage wie vorhin führt.

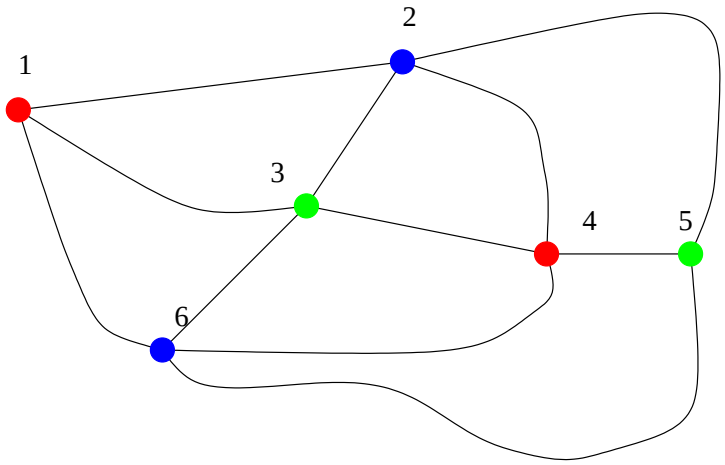


Abbildung 1.8: Färbung des Nachbarschaftsgraphen zur Landkarte aus Abbildung 1.7.

Das Landkartenfärbungsproblem bzw. das Graphenfärbungsproblem lässt sich trivial lösen, indem jedes Land eine eigene Farbe erhält. Eigentlich interessiert man sich jedoch für folgende *Optimierungsversion des Färbungsproblems:* Konstruiere eine Färbung mit minimaler Anzahl an Farben. Betrachtet man dabei den Nachbarschaftsgraphen einer Landkarte, so ist dies das „klassische“ Graphenfärbungsproblem eingeschränkt auf planare Graphen. Für beliebige Graphen ist das Graphenfärbungsproblem $\mathcal{NP}$-schwer. (Siehe Vorlesung „Theoretische Grundlagen der Informatik“.) Es lässt sich jedoch folgender Satz, der *Vierfarbensatz* beweisen.

Satz 1.2. *Jeder planare Graph lässt sich mit höchstens vier Farben färben.*

Dieser Satz wurde bereits 1852 von de Morgan formuliert und 1879 von Kempe der erste falsche Beweis geliefert. Aufgrund dieses „begeisternden“ Resultats wurde Kempe zum „Fellow of the Royal Society“ gewählt. 1890 fand Heawood den Fehler in Kempes Beweis (dazu später mehr). 1977 wurde der Vierfarbensatz von Appel & Haken „endgültig bewiesen“. Dieser Beweis besteht aus einer riesigen Anzahl von Fallunterscheidungen, die durch Computereinsatz gelöst werden und ist daher bei einigen Mathematiker umstritten. 1995 wurde ein wesentlich kürzerer Beweis, der allerdings auch einen Computer benutzt, von Robertson, Sanders, Seymour & Thomas angegeben. Es ist leicht zu sehen, dass man zur Färbung des K_4 auch vier Farben benötigt. Andererseits gibt es planare Graphen, die mit weniger als vier Farben gefärbt werden können. Es ist jedoch auch für planare Graphen $\mathcal{NP}$-vollständig, zu entscheiden, ob drei Farben ausreichen. Es ist übrigens wiederum „leicht“ (auch) für beliebige Graphen zu entscheiden, ob sie mit zwei Farben gefärbt werden können. Die zweifärbbaren Graphen sind nämlich gerade die *bipartiten* Graphen. Eine interessante Frage ist nun allgemein:

> „Hilft“ Planarität bei der Lösung algorithmischer Probleme auf Graphen? Gibt es weitere Optimierungsprobleme, die für beliebige Graphen $\mathcal{NP}$-schwer sind, für planare Graphen aber in $\mathcal{P}$ sind?

Erstaunlicherweise scheint es nicht viele solche Probleme zu geben. Ein weiteres Beispiel eines $\mathcal{NP}$-schweren Problems, das für planare Graphen in $\mathcal{P}$ ist, ist das „MAX-CUT-Problem“. Wir werden dieses Problem hier behandeln. Allgemein scheint die Eigenschaft der Planarität bei „Schnittproblemen“ oder „Zerlegungsproblemen“ in Graphen vorteilhaft zu sein. Betrachte etwa folgendes „Zerlegungsproblem“:

> Gegeben sei ein Graph $G = (V, E)$. Finde eine Kantenmenge S minimaler oder zumindest kleiner Größe, so dass G durch Entfernen von S in disjunkte Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ „zerfällt“ mit
>
> $$|V_i| \leq \alpha \cdot |V| \quad \text{für} \quad i = 1, 2;\ 0 < \alpha < 1.$$

Diese Fragestellung werden wir später wieder aufgreifen, wenn wir das „Planar-Separator-Theorem“ beweisen. Wie kann Planarität bei solchen Schnittproblemen helfen? Es gibt eine schöne Korrespondenz zwischen Kreisen und Schnitten in planaren Graphen:

- *Schnitte* in G korrespondieren zu *Kreisen* im „Dualgraph“ von G.
- Die *Größen von Schnitten* in G korrespondieren zu *Längen von Wegen* im „Dualgraph“ von G.

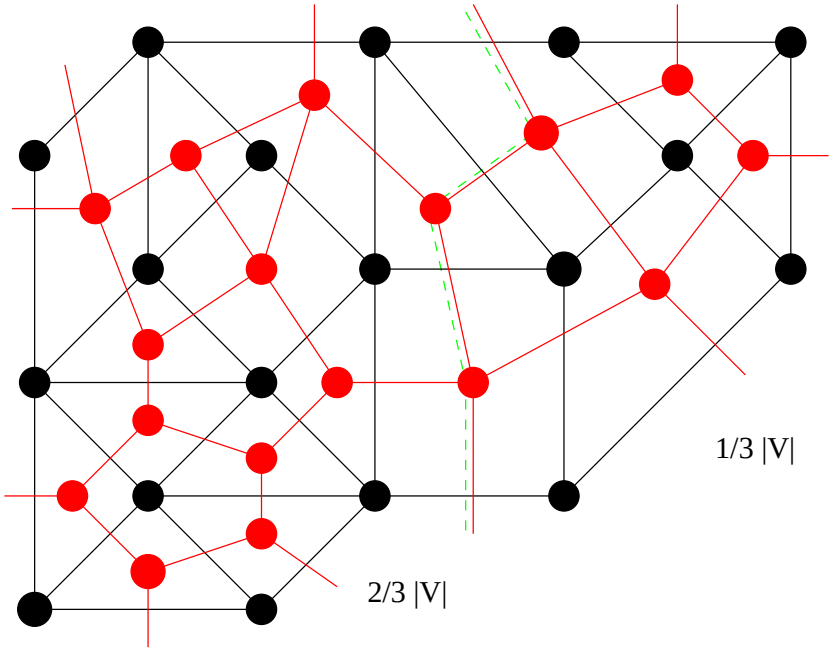


Abbildung 1.9: Eine $\frac{1}{3} - \frac{2}{3}$-Zerlegung durch Wegnahme von 4 Kanten und der entsprechende Weg der Länge 4 im Dualgraph.

Allgemein kann also die Korrespondenz zwischen „Konfigurationen“ im planaren Graph G und entsprechenden „Konfigurationen“ in seinem Dualgraph bei der Lösung algorithmischer Probleme helfen. Ein weiterer Vorteil planarer Graphen besteht darin, dass man

bei gewissen algorithmischen Vorgehensweise wie Tiefen– und Breitensuche eine planare Einbettung des Graphen ausnutzen kann. Dies scheint vor allem bei Wegeproblemen, Steinerbaumpackungsproblemen und dem Menger-Problem eine erfolgreiche Strategie zu sein, wie wir später sehen werden.

- Man kann bezüglich einer beliebigen aber festen Einbettung „Inneres“, Äußeres“, „rechts von“, „links von“ ... zur Konstruktion und Argumentation verwenden.

Weitere Vorteile planarer Graphen sind:

- Wegen ihrer „guten Zerlegbarkeit“ lässt sich sehr gut das „Divide-and-Conquer“-Prinzip anwenden.
- Es gibt Aussagen über
  - die Knotengrade im planaren Graphen,
  - die maximale Kantenzahl,
  - den maximalen Zusammenhang,
  - ...,

  die „direkt“ aus der Planarität folgen.

2 Grundlegende Eigenschaften planarer Graphen

2.1 Grundlegende Eigenschaften

Ein *Graph* $G = (V, E)$ besteht aus einer endlichen Menge V von *Knoten* und einer endlichen Menge E von *Kanten*, sowie einer Vorschrift, die jeder Kante $e \in E$ genau zwei Knoten $u, v \in V$, ihre *Endknoten*, zuordnet ($u = v$ möglich). Sind die beiden Endknoten u, v von e identisch, so heißt e *Schlinge*; wir sagen allgemein, dass Kante e Knoten u und v *verbindet*.

Knoten, die durch eine Kante verbunden sind, heißen *benachbart*. Zu $v \in V$ definiere die *Nachbarschaft* $N(v) := \{u \in V : u \text{ ist zu } v \text{ benachbart}\}$. Ist der Knoten v ein Endknoten der Kante e, so heißen v und e *inzident*. Ebenso heißen zwei Kanten, die einen gemeinsamen Endknoten haben, *inzident*. Haben Kanten $e_1, \dots e_k,\ k \geq 2$ beide Endknoten gemeinsam, so heißen sie *Mehrfachkanten*. Falls $G = (V, E)$ keine Schlingen und keine Mehrfachkanten besitzt, so heißt G *einfach*. In diesem Fall kann E als Teilmenge von $\big\{\{u, v\} : u, v \in V, u \neq v\big\}$ aufgefasst werden. Der *Grad von v*, bezeichnet mit $d(v)$, ist die Anzahl der zu v inzidenten Kanten. In einem einfachen Graphen ist also $d(v) := |N(v)|$.

Eine Folge $v_0 e_1 v_1 e_2 \dots v_{k-1} e_k v_k$ von Knoten und Kanten in G, für die v_{i-1} und v_i Endknoten von e_i sind und $e_i \neq e_{i+1}$, ist ein *Weg* von v_0 (Anfangsknoten) nach v_k (Endknoten). Die *Länge* des Weges ist die Anzahl der durchlaufenen Kanten. Ist G einfach, so geben wir bei einem Weg in G die Kanten nicht explizit an. Ein Weg ohne Knotenwiederholung heißt *einfach*. Ein Weg, für den Anfangs- und Endknoten identisch sind, heißt *Kreis* oder *Zykel*. Ein Kreis, an dem (außer Anfangs- und Endknoten) alle Knoten verschieden sind, heißt wiederum *einfach*. $G = (V, E)$ heißt *zusammenhängend*, wenn es zwischen je zwei Knoten aus V einen Weg in G gibt, ansonsten ist G *unzusammenhängend*. Da die „Wegverbundenheit“ eine Äquivalenzrelation ist, zerfällt jeder Graph eindeutig in zusammenhängende Komponenten, seine *Zusammenhangskomponenten*.

Ein Graph $G = (V, E)$ kann *dargestellt* werden, indem man die Knoten aus V auf Punkte in der Ebene abbildet, und die Kanten aus E als Jordan–Kurven (stetige, sich selbst nicht kreuzende Kurven) zwischen den Endpunkten.

Ein Graph $G = (V, E)$ heißt *planar*, wenn es eine Darstellung von G gibt, in der sich die Kanten nicht kreuzen, also nur in Knoten treffen. Eine solche Darstellung nennen

wir dann auch *planare Einbettung*. Eine planare Einbettung eines Graphen zerlegt die Ebene in *Facetten (Gebiete, Flächen)*.

Offensichtlich gibt es zu jedem Knoten v bzw. zu jeder Kante e eines planaren Graphen eine Einbettung, bei der v bzw. e auf dem Rand der äußeren Facette liegt:

Betrachte eine Einbettung auf der Kugel und „rolle“ diese so in der Ebene aus, dass die/eine Facette, die v bzw. e auf ihrem Rand hat, zur äußeren Facette wird.

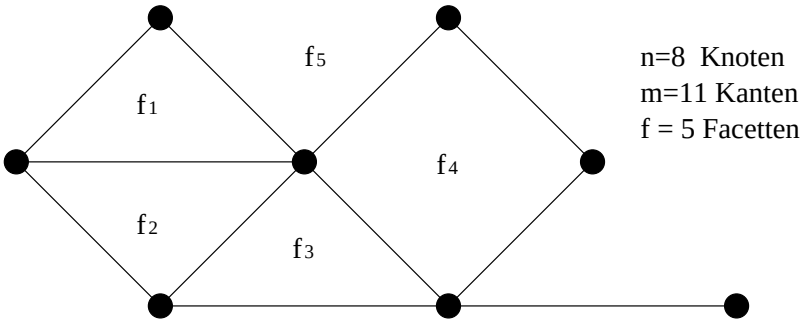


Abbildung 2.1: Knoten, Kanten und Facetten eines planaren Graphen.

Jede Kante, die auf einem einfachen Kreis liegt, grenzt an genau zwei Facetten an; alle anderen an genau eine Facette.

Im folgenden bezeichne immer $n = |V|, m = |E|$ und f die Anzahl der Facetten. Der *Satz von Euler* (bewiesen 1750) beschreibt den Zusammenhang zwischen n, m und f in einem planaren Graphen.

Satz 2.1. Satz von Euler
In einem zusammenhängenden planaren Graph $G = (V, E)$*, mit* $|V| = n$*,* $|E| = m$ *und* f *Anzahl der Facetten gilt für jede seiner planaren Einbettungen*

$$n - m + f = 2.$$

Beweis. Wir führen eine Induktion über m durch. Für $m = 0$ besteht G aus einem einzelnen Knoten und $n - m + f = 1 + 1 = 2$ gilt. Sei nun $m \geq 1$.

Fall 1: Wenn G einen Kreis enthält, so gibt es eine Kante e, die wir aus G entfernen können, so dass $G' = (V, E\backslash\{e\})$ immer noch zusammenhängend ist. Die beiden Facetten von G, die an e angrenzen, werden durch Wegnahme von e zu einer Facette, d.h. die Anzahl der Facetten f' von G' erfüllt $f' = f - 1$. Nach Induktionsvoraussetzung ist

$$n - (m - 1) + f' = 2,$$

also folgt die Behauptung für G.

Fall 2: Enthält G keinen Kreis, so besitzt er genau eine Facette und zerfällt durch Wegnahme einer beliebigen Kante e in zwei zusammenhängende Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ mit $n_1 = |V_1|,\ n_2 = |V_2|,\ m_1 = |E_1|,\ m_2 = |E_2|$ und $n_1+n_2 = n,\ m_1+m_2 = m-1$. Für G_1 bzw. G_2 gilt nach Induktionsvoraussetzung, dass

$$\begin{aligned} n - m + f &= (n_1 + n_2) - (m_1 + m_2 + 1) + 1 \\ &= n_1 - m_1 + n_2 - m_2 \\ &= 1 + 1 = 2 \end{aligned}$$

□

Folgerung 2.2. *Ein zusammenhängender, planarer Graph ohne Kreise besitzt* $n - 1$ *Kanten.*

Ein zusammenhängender Graph ohne Kreis heißt *Baum* und ist offensichtlich immer planar.

Folgerung 2.3. *Ein planarer, einfacher Graph mit* $n \geq 3$ *Knoten hat höchstens* $3n-6$ *Kanten.*

Beweis. Betrachte einen planaren Graphen G. Wir können annehmen, dass G maximal planar ist, d.h. unter allen Graphen mit n Knoten maximale Kantenzahl hat. In einer Einbettung von G muss dann jede Facette durch genau drei Kanten begrenzt sein. Insbesondere ist G zusammenhängend. Da jede Kante zwei Facetten begrenzt, gilt $3 \cdot f = 2 \cdot m$. Also gilt mit Satz 2.1, dass $m = 3n - 6$. □

Lemma 2.4. *Sei* G *ein planarer, einfacher Graph mit* $n \geq 3$ *Knoten,* $d_{max}(G)$ *bezeichne den Maximalgrad in* G *und* n_i *die Anzahl der Knoten in* G *mit Grad* i, $0 \leq i \leq d_{max}(G)$. *Dann gilt*

$$\begin{aligned} 6 \cdot n_0 + 5 \cdot n_1 + 4 \cdot n_2 + 3 \cdot n_3 + 2 \cdot n_4 + n_5 \geq \\ n_7 + 2 \cdot n_8 + 3 \cdot n_9 + \cdots + (d_{max}(G) - 6) \cdot n_{d_{max}} + 12. \end{aligned}$$

Beweis. Offensichtlich ist $n = \sum_{i=0}^{d_{max}(G)} n_i$ und $2 \cdot m = \sum_{i=0}^{d_{max}(G)} i \cdot n_i$.

Da wegen Folgerung 2.3 gilt, dass $6n \geq 2m + 12$ ist, folgt

$$6 \cdot \sum_{i=0}^{d_{max}(G)} n_i \geq \sum_{i=0}^{d_{max}(G)} i \cdot n_i + 12.$$

□

Folgerung 2.5. *Jeder planare, einfache Graph enthält einen Knoten* v *mit* $d(v) \leq 5$.

2.2 Charakterisierung planarer Graphen

Wir wollen nun untersuchen, wie sich planare Graphen charakterisieren lassen. Eine zentrale Rolle spielt dabei der *Satz von Kuratowski*, der planare Graphen anhand von verbotenen Subgraphen charakterisiert. Ein Graph $H = (V(H), E(H))$ heißt *Subgraph (Teilgraph)* von $G = (V, E)$, falls $V(H) \subseteq V$ und $E(H) \subseteq E$. Eine Teilmenge $V' \subseteq V$ *induziert* einen Subgraph $H = (V', E(H))$ von G durch $E(H) := \{e \in E :$ beide Endknoten von e sind in $V'\}$, genannt *knoteninduzierter Subgraph*. Jede Teilmenge $E' \subseteq E$ *induziert* einen Subgraph $H = (V(H), E')$ von G durch $V(H) := \{v \in V :$ es gibt eine Kante $e \in E'$ mit Endknoten $v\}$, genannt *kanteninduzierter Subgraph*.

Ein Graph H ist eine *Unterteilung* von G, wenn H aus G entsteht, indem Kanten von G durch einfache Wege über neu eingefügte Knoten ersetzt werden. Alle neu eingefügten Knoten haben also Grad 2 (siehe Abbildung 2.2).

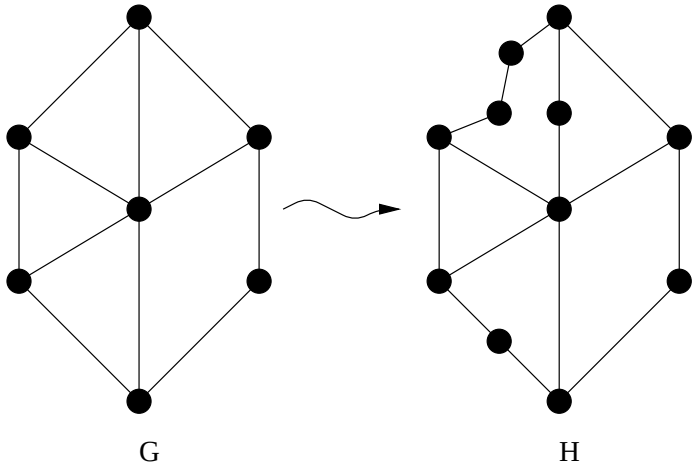


Abbildung 2.2: H ist eine Unterteilung von G.

Bemerkung.

1. *Ein Graph, der einen nicht planaren Subgraph besitzt, ist nicht planar.*
2. *Ein Graph, der eine Unterteilung eines nicht planaren Graphen ist, ist nicht planar.*
3. *Ein Graph, der eine Unterteilung eines nicht planaren Graphen als Subgraph besitzt, ist nicht planar.*

Wir wollen nun die planaren Graphen vollständig durch verbotene Subgraphen charakterisieren. Vorab benötigen wir noch den Begriff des k-fachen Zusammenhangs. Eine Menge $S \subset V$ heißt *Separator* von $G = (V, E)$, falls der durch $V \backslash S$ induzierte Subgraph von G unzusammenhängend ist. S trennt die Knoten $u, v \in V \backslash S$, falls u und v in dem

durch $V \backslash S$ induzierten Subgraph (bezeichnet mit $G - S$) in verschiedenen Zusammenhangskomponenten liegen (siehe Abbildung 2.3).

Wir definieren den *Knotenzusammenhang* $\kappa_G(u, v)$ zweier Knoten u und v bzw. den *Knotenzusammenhang* $\kappa(G)$ des Graphen G wie folgt.

$$\kappa_G(u,v) := \begin{cases} |V| - 1, \text{ falls } \{u,v\} \in E \\ \min\limits_{\substack{S \subset V, \\ S \text{ trennt } u \text{ und } v}} |S|, \text{ sonst.} \end{cases}$$

$$\kappa(G) := \min_{\substack{S \subset V, \\ S \text{ Separator von } G}} \{|S|, |V| - 1\} = \min_{u,v \in V} \kappa_G(u,v)$$

Eine Menge $S \subseteq E$ heißt *Schnitt* von $G = (V, E)$, falls der durch $E \backslash S$ induzierte Subgraph von G unzusammenhängend ist, d.h. in Graphen $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ zerfällt, mit $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, $E_1 \cup E_2 = E \backslash S$, $E_1 \cap E_2 = \emptyset$, wobei alle Kanten aus S einen Endknoten in V_1 und einen Endknoten in V_2 haben. S *trennt* die Knoten $u, v \in V$, falls u und v in dem durch $E \backslash S$ induzierten Subgraph (bezeichnet mit $G - S$) in verschiedenen Zusammenhangskomponenten liegen.

Entsprechend definieren wir den *Kantenzusammenhang* $\lambda_G(u, v)$ zweier Knoten u und v, bzw. den *Kantenzusammenhang* $\lambda(G)$ des Graphen G wie folgt.

$$\lambda_G(u,v) := \min_{\substack{S \subseteq E, \\ S \text{ trennt } u \text{ und } v}} |S|$$

$$\lambda(G) := \min_{\substack{S \subseteq E, \\ S \text{ Schnitt von } G}} |S| = \min_{u,v \in V} \lambda_G(u,v)$$

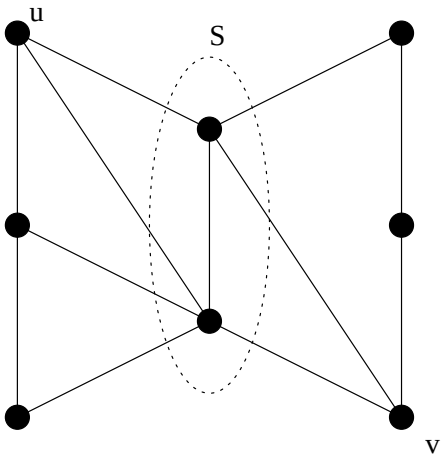


Abbildung 2.3: S trennt u und v.

Ein Schnitt S mit $|S| = 1$ heißt *Brücke*. G heißt *k-fach knoten- bzw. kantenzusammenhängend*, falls $k \leq \kappa(G)$ bzw. $k \leq \lambda(G)$. Zwei Wege in einem Graphen G heißen *(intern) knotendisjunkt*, wenn sie (außer den Endknoten) keine gemeinsamen Knoten enthalten und *kantendisjunkt*, wenn sie keine gemeinsame Kante enthalten.

Satz 2.6 (Satz von Menger (1927), ohne Beweis).
Seien s und t zwei Knoten eines Graphen G, s und t nicht adjazent bei der knotendisjunkten Version.

- $\kappa_G(s,t) \geq k$ *genau dann, wenn es k paarweise intern knotendisjunkte Wege zwischen s und t in G gibt.*
- $\lambda_G(s,t) \geq k$ *genau dann, wenn es k paarweise kantendisjunkte Wege zwischen s und t in G gibt.*

Der Satz von Menger ist ein typisches Beispiel einer *Dualitätsaussage*, d. h. Dualität zwischen einem Minimum (hier Größe des Separators bzw. Schnittes) und einem Maximum (hier Anzahl disjunkter Wege). Die kantendisjunkte Version des Satzes ist ein Spezialfall des Max-Flow-Min-Cut-Theorems (alle Kantenkapazitäten sind eins), das in der Vorlesung „Algorithmentechnik“ bewiesen wurde.

Folgerung 2.7. *Ist $S \subset V$ ein Separator, der s und t trennt und $|S| = \kappa_G(s,t) = k$, so gibt es auch zwei „Bündel“ von jeweils k intern knotendisjunkten Wegen von s nach S und von t nach S, die jeweils zu verschiedenen Knoten in S führen.*

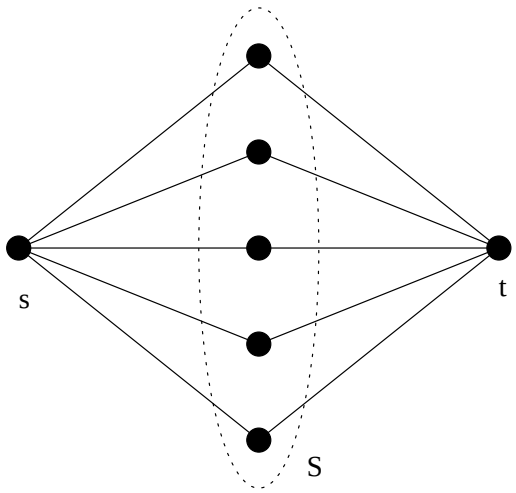


Abbildung 2.4: Illustration zu Folgerung 2.7.

Satz 2.8 (Satz von Kuratowski (1930)).
Ein Graph ist genau dann planar, wenn er keine Unterteilung von K_5 oder $K_{3,3}$ als Subgraph enthält.

Beweis. Wir haben bereits gezeigt, dass K_5 und $K_{3,3}$ nicht planar sind. Damit ist klar, dass ein planarer Graph keine Unterteilung des K_5 bzw. $K_{3,3}$ als Subgraph enthalten kann. Es bleibt zu zeigen, dass jeder Graph, der keine Unterteilung des K_5 bzw. $K_{3,3}$ als Subgraph enthält, planar ist.

Wir führen eine Induktion über die Anzahl n der Knoten des Graphen $G = (V, E)$ durch. Für $n \leq 4$ gilt die Behauptung, da K_4 planar ist. Für $n \geq 5$ führen wir eine Induktion über die Anzahl m der Kanten des Graphen durch.

Wenn der Graph $m = 0$ Kanten enthält, ist G trivialerweise planar. Gelte also die Behauptung für alle Graphen mit weniger als n Knoten oder n Knoten und echt weniger als m Kanten. $G = (V, E)$ sei ein Graph mit $|V| = n \geq 5$ und $|E| = m$. Wir machen eine Fallunterscheidung nach $\kappa(G)$.

Fall 1: $\kappa(G) = 0$

Nach Induktionsvoraussetzung kann jede Zusammenhangskomponente von G planar eingebettet werden; also ist auch G planar. ∎

Fall 2: $\kappa(G) = 1$

Es gibt einen Knoten v, so dass $\{v\}$ ein Separator von G ist. G kann also auch zerlegt werden in zwei kantendisjunkte Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ mit $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \emptyset$. Nach Induktionsvoraussetzung können G_1 und G_2 planar eingebettet werden, und zwar so, dass v jeweils auf dem Rand der äußeren Facette liegt. Aus diesen Einbettungen erhält man dann auch eine planare Einbettung von G (siehe Abbildung 2.5). ∎

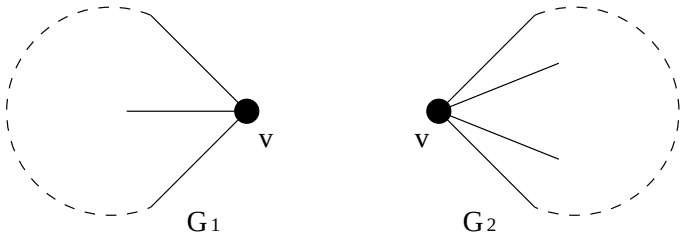


Abbildung 2.5: $\{v\}$ ist Separator von G.

Fall 3: $\kappa(G) = 2$

Es gibt einen Separator $\{u, v\}$ und G kann zerlegt werden in $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ mit $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \{u, v\}$, $E = E_1 \cup E_2$ und $E_1 \cap E_2 = \emptyset$ bzw. $E_1 \cap E_2 = \{\{u, v\}\}$, falls $\{u, v\} \in E$, siehe Abbildung 2.6.

Falls $\{u, v\} \in E$, so können nach Induktionsvoraussetzung G_1 und G_2 so planar eingebettet werden, dass $\{u, v\}$ jeweils auf dem Rand der äußeren Facette liegt. Daraus erhält man dann auch eine planare Einbettung von G.

Falls $e := \{u, v\} \notin E$, so betrachte $G_1 + e$ $(= (V_1, E_1 \cup \{e\}))$ und $G_2 + e$. Wir zeigen, dass auch $G_1 + e$ und $G_2 + e$ planar sind. Nach Induktionsvoraussetzung genügt es dazu, zu zeigen, dass sie keine Unterteilung des K_5 bzw. $K_{3,3}$ enthalten. Falls

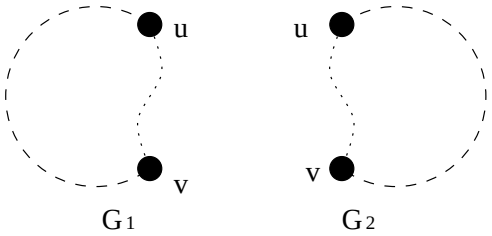


Abbildung 2.6: $\{u, v\}$ ist Separator von G.

$G_i + e$ eine Unterteilung des K_5 oder des $K_{3,3}$ enthalten würde, so müsste diese die Kante e enthalten. Da $\kappa(G) = 2$, ist auch $\kappa(G_i) \geq 2$. Es gibt also in G_i jeweils einen Weg P_i von u nach v, $i = 1, 2$. Dann enthielte aber auch $G_i + P_j$, mit $i \neq j$ eine Unterteilung des K_5 bzw. des $K_{3,3}$, die P_i enthält und somit enthielte G eine solche Unterteilung. Dies ist ein Widerspruch.

Wir können also wieder $G_i + e$ so planar einbetten, dass e auf dem Rand der äußeren Facette liegt, und erhalten daraus eine planare Einbettung für G. ∎

Fall 4: $\kappa(G) \geq 3$

Sei $e = \{u, v\}$ eine beliebige Kante und $G' := G - e$. Wir unterscheiden die beiden Fälle $\kappa_{G'}(u, v) = 2$ und $\kappa_{G'}(u, v) \geq 3$.

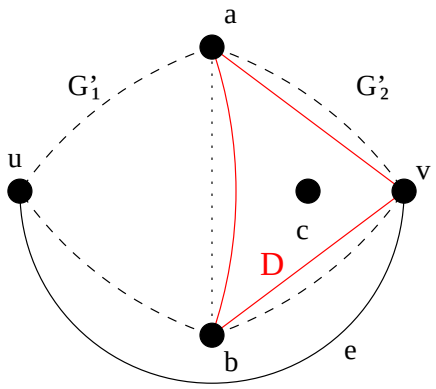


Abbildung 2.7: Illustration von Fall 4.1: $\{a, b\}$ ist Separator von $G - e$.

Fall 4.1: $\kappa_{G'}(u, v) = 2$

Es gibt einen Separator $\{a, b\}$, der u und v in G' trennt. G' kann dann an den Knoten a, b zerlegt werden in Subgraphen $G'_1 = (V_1, E_1)$ und $G'_2 = (V_2, E_2)$ bestehend aus den durch Wegnahme von $\{a, b\}$ induzierten Subgraphen jeweils zusammen mit a, b und den dazu inzidenten Kanten zu Knoten im Subgraph. Es ist also $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \{a, b\}$, $u \in V_1$, $v \in V_2$, $E_1 \cup E_2 = E \backslash \{e\}$ und $E_1 \cap E_2 = \emptyset$ bzw. $E_1 \cap E_2 = \{\{a, b\}\}$, falls $\{a, b\} \in E$.

Da $n \geq 5$ ist, existiert ein weiterer Knoten c, wobei o. B. d. A. c in G'_2 sei. Füge, falls noch nicht vorhanden, die Kanten $\{a, b\}$, $\{a, v\}$ und $\{b, v\}$ ein, und nenne dieses „Dreieck“ D (siehe Abbildung 2.7).

Setze $G_1 := G_1' + D + e$ und $G_2 := G_2' + D$. Dann haben G_1 und G_2 genau D gemeinsam. Für G_1 und G_2 ist die Induktionsvoraussetzung erfüllt, da sie beide echt weniger Knoten als G haben. Wir gehen nun folgendermaßen vor:

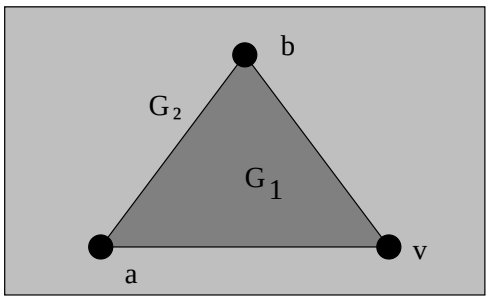


Abbildung 2.8: Illustration von Fall 4.1. Einbettung von G_1 in das Dreieck von G_2.

G_1 und G_2 werden so eingebettet, dass D jeweils eine Facette ist, und zwar die äußere Facette von G_1 und eine innere von G_2. Für G wird dann bewiesen, dass basierend auf diesen Einbettungen ebenfalls eine planare Einbettung existiert. Dabei wird G_1 in G_2 eingebettet (siehe Abbildung 2.8).

Zunächst muss natürlich gezeigt werden, dass G_1 und G_2 planar sind, d.h. D keinen K_5 oder $K_{3,3}$ „zulässt“, und darüber hinaus, dass G_1, G_2 geeignet (wie oben) einbettbar sind.

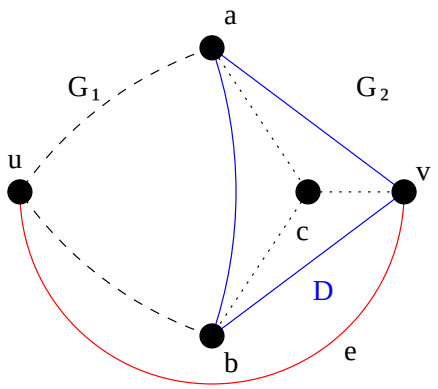


Abbildung 2.9: Illustration von Fall 4.1. $\{a, b, v\}$ ist ein Separator, der u und c trennt.

Da $\kappa(G) \geq 3$ ist, gibt es drei knotendisjunkte Wege in G, die u und c verbinden. Die Menge $\{a, b, v\}$ muss in G ein Separator sein, der u und c trennt. Also gehen alle drei Wege von u nach c durch diesen Separator. Entsprechend existieren die beiden Bündel knotendisjunkter Wege von c nach $\{a, b, v\}$ und von u nach $\{a, b, v\}$ (siehe Abbildung 2.9).

Wir zeigen: Enthielte nun G_1 oder G_2 eine Unterteilung des K_5 oder $K_{3,3}$, die Kanten aus D benutzt, so gäbe es auch in G eine Unterteilung des K_5 bzw. des $K_{3,3}$, die gegebenenfalls über geeignete Wege von c nach a, b und v, bzw. von u nach a, b und v gingen.

Annahme 4.1.1: G_1 enthält eine Unterteilung von K_5 oder $K_{3,3}$, die Kanten aus D benutzt.
Wenn sie nur eine der Kanten aus D benutzt, so kann diese leicht durch einen entsprechenden Weg über c in G simuliert werden. Alle drei Kanten aus D können nur bei einem K_5 benutzt werden, in dem v nicht Unterteilungsknoten ist. Ein solcher K_5 kann aber nicht existieren, da $d_{G_1}(v) = 3$. Ebenso kann es keinen K_5 oder $K_{3,3}$ geben, der zwei Kanten aus D benutzt und den zu diesen beiden Kanten inzidenten Knoten a, b oder v nicht als Unterteilungsknoten benutzt. Werden zwei Kanten aus D für einen K_5 oder $K_{3,3}$ benutzt, so können diese zusammen mit dem Knoten (a, b oder v), der zu beiden Kanten inzident ist, durch c mit geeigneten Wegen simuliert werden. Also führt die Annahme zu einem Widerspruch. ◇

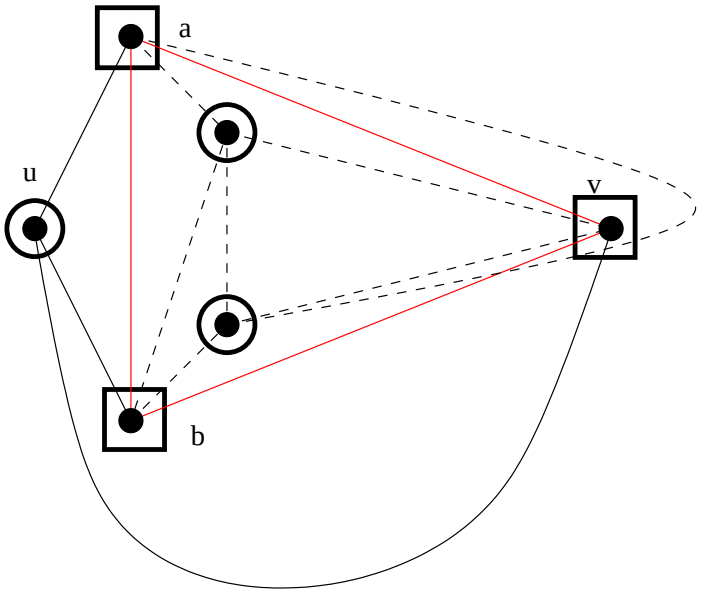


Abbildung 2.10: Illustration zu Fall 4.1.2.

Annahme 4.1.2 G_2 enthält eine Unterteilung von K_5 oder $K_{3,3}$, die Kanten aus D benutzt.
Eine einzelne Kante aus D kann wiederum leicht durch einen Weg über u simuliert werden. Wenn alle drei Kanten aus D benutzt werden, so gibt es einen K_5, also zwei weitere Knoten in G_2, die knotendisjunkte Wege zu a, b und v haben. Dann würde aber u mit diesen beiden Knoten und den Knoten a, b, v und den entsprechenden Wegen eine Unterteilung des $K_{3,3}$ in G sein (siehe Abbildung 2.10).

Werden zwei Kanten aus D für eine Unterteilung des K_5 benutzt, so kann wiederum ähnlich wie eben eine Unterteilung des $K_{3,3}$ in G konstruiert werden. Zwei Kanten aus D zusammen mit dem Knoten, der zu diesen beiden Kanten in einer Unterteilung des $K_{3,3}$ inzident ist, können durch entsprechende Wege über u zusammen mit u simuliert werden. ◇

Wir können nun G_1 und G_2 so planar einbetten, dass D eine Facette begrenzt.

Betrachte dazu eine planare Einbettung von G_1. Angenommen in dieser Einbettung gibt es Knoten x und y im Inneren bzw. Äußeren von D. Da $\kappa_G(x, y) \geq 3$ ist, gibt es drei disjunkte Wege in G von x zu a, b, v bzw. y zu a, b, v. Zusammen mit den Wegen (die nicht in G_1 liegen) von c zu a, b und v gibt es dann aber einen $K_{3,3}$ in G. Die Argumentation ist analog für G_2. ◇

Fall 4.2: $\kappa_{G'}(u, v) \geq 3$

Nach Induktionsvoraussetzung ist G' planar. Betrachte also eine planare Einbettung von G'. Liegen u und v auf dem Rand einer gemeinsamen Facette, so kann die Kante $\{u, v\} = e$ innerhalb dieser Facette ebenfalls planar eingefügt werden.

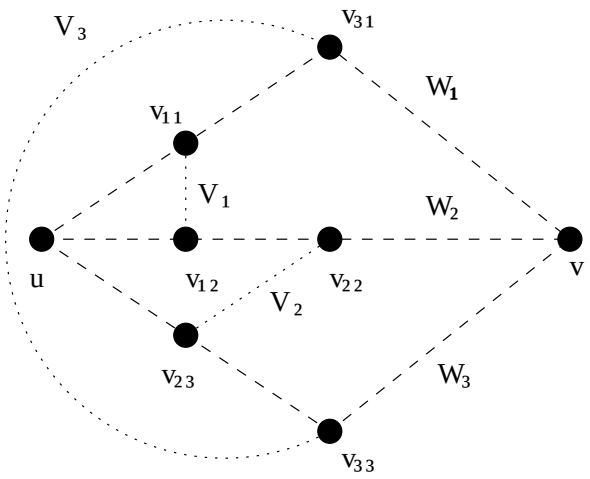


Abbildung 2.11: Illustration zu Fall 4.2.

Den anderen Fall führen wir zum Widerspruch: Betrachte also die Facetten, die an u grenzen. Da $\kappa_{G'}(u, v) \geq 3$, gibt es mindestens drei disjunkte Wege W_1, W_2 und W_3 von u nach v in G', also auch mindestens drei Facetten, die an u grenzen. Die Ränder dieser Facetten induzieren wiederum mindestens drei Wege V_1, V_2 und V_3 um u herum. Wir führen den Beweis für den Fall, dass es genau drei Facetten gibt, die an u angrenzen. Die Argumentation für den Fall, dass es mehr als drei angrenzende Facetten gibt, ist analog.

Sei v_{ij} jeweils der gemeinsame Knoten von V_i und W_j. Wir zeigen, dass je nachdem ob die entsprechenden beiden Knoten aus W_j identisch sind oder nicht, es nun eine Unterteilung des $K_{3,3}$ oder des K_5 in G gibt (siehe Abbildung 2.11).

a) $v_{ij} = v_{lj}$ für alle $j \in \{1, 2, 3\}$
   Dies ist Unterteilung des K_5.
b) $v_{ij} = v_{lj}$ für genau zwei der $j \in \{1, 2, 3\}$
   Dann ergibt sich eine Unterteilung des $K_{3,3}$.
c) $v_{ij} = v_{lj}$ für genau ein $j \in \{1, 2, 3\}$
   Dann ergibt sich eine Unterteilung des $K_{3,3}$.

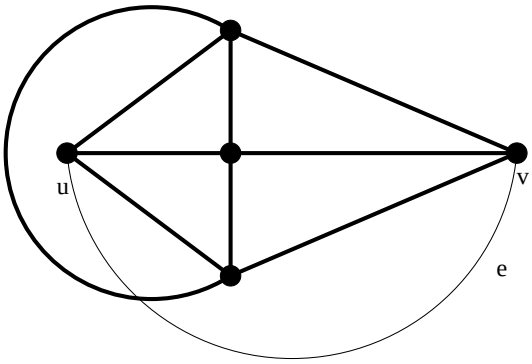


Abbildung 2.12: Fall 4.2 a).

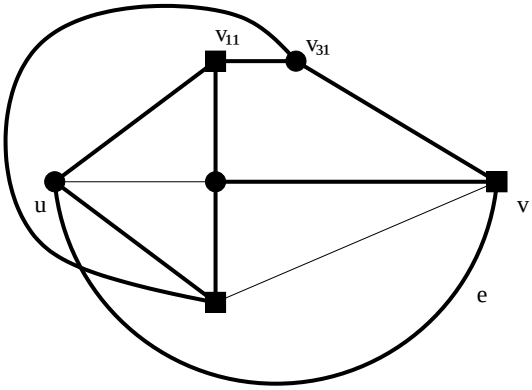


Abbildung 2.13: Fall 4.2 b).

d) keine Gleichheit
Dann ergibt sich eine Unterteilung des $K_{3,3}$. ◇ ■

Somit haben wir gezeigt, dass sich G planar einbetten lässt, falls G keine Unterteilung von K_5 oder $K_{3,3}$ enthält. □

Zu einem beliebigen Graphen $G = (V, E)$ können wir den dazu korrespondierenden einfachen Graphen betrachten, der entsteht, indem alle Schleifen aus G entfernt werden und Kanten mit denselben Endknoten zu einer Kante zusammengefasst werden.

Offensichtlich ist ein Graph planar genau dann, wenn der korrespondierende einfache Graph planar ist.

Der Satz von Kuratowski liefert also auch eine Charakterisierung nicht einfacher, planarer Graphen. Mit dem Satz von Kuratowski haben wir die planaren bzw. nicht planaren Graphen vollständig charakterisiert. Wie nützlich ist der Satz von Kuratowski aus algorithmischer Sicht?

Ein Algorithmus, der basierend auf der Aussage dieses Satzes für einen beliebigen Graphen untersucht, ob dieser planar ist, würde Subgraphen betrachten, und entscheiden,

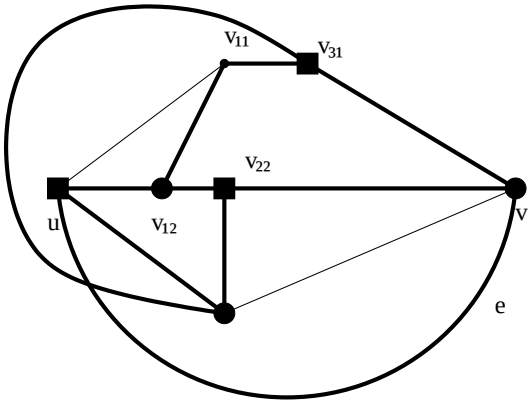


Abbildung 2.14: Fall 4.2 c).

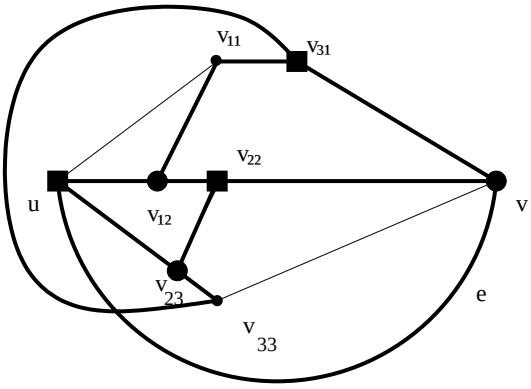


Abbildung 2.15: Fall 4.2 d).

ob diese Unterteilungen des $K_{3,3}$ oder K_5 sind. Es gibt mindestens 2^m Subgraphen. Dieses Verfahren scheint also nicht effizient zu sein. Man könnte anhand des Beweises einen effizienten Algorithmus angeben. Es gibt verschiedene effiziente Algorithmen zum Testen auf Planarität mit Laufzeit $\mathcal{O}(n)$, siehe Kapitel **??**.

2.3 Dualgraph

Betrachte einen planaren Graphen $G = (V, E)$ mit einer festen Einbettung. F sei die Menge der Facetten von G bzgl. dieser Einbettung. Definiere dazu einen Graphen $G^* = (V^*, E^*)$ wie folgt:

Zu jeder Facette aus F gibt es einen Knoten in V^*, und zu jeder Kante $e \in E$ gibt es genau eine duale Kante $e^* \in E^*$, die die beiden Knoten aus V^* verbindet, welche den Facetten aus F entsprechen, an die e angrenzt. G^* heißt *geometrischer Dualgraph* (oder nur Dualgraph) zu G (siehe Abbildung 2.16).

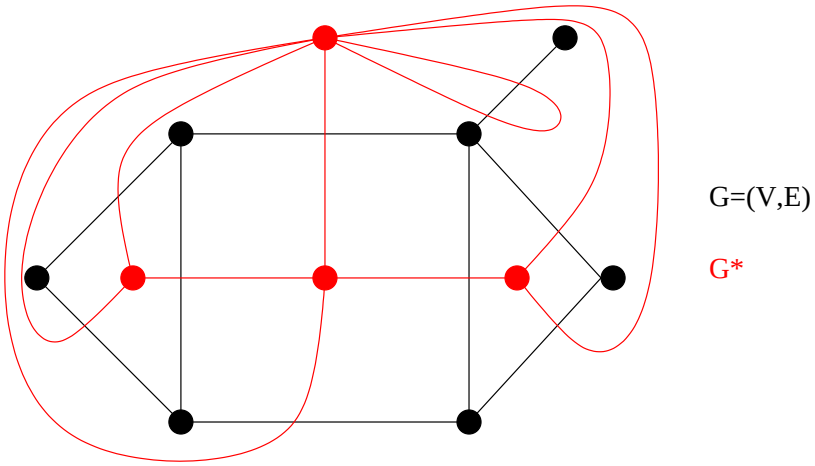


Abbildung 2.16: Ein planar eingebetteter Graph G und sein Dualgraph G^*.

Beobachtung.

1. *Der Dualgraph G^* zu einem einfachen planaren Graphen G ist nicht notwendig einfach. G^* besitzt genau dann Mehrfachkanten, wenn es zwei benachbarte Facetten gibt, die mehr als eine begrenzende Kante gemeinsam haben. G^* enthält genau dann eine Schleife, wenn G eine Brücke enthält.*
2. *Offensichtlich ist G^* wieder planar und $(G^*)^* = G$. Jedoch nur, wenn man G^* mit seiner „kanonischen" Einbettung, die durch die Einbettung von G bestimmt wird, betrachtet.*
3. *Zu einem planaren Graphen G gibt es möglicherweise mehr als einen Dualgraphen G^*, da G ja möglicherweise mehrere verschiedene Einbettungen besitzt.*

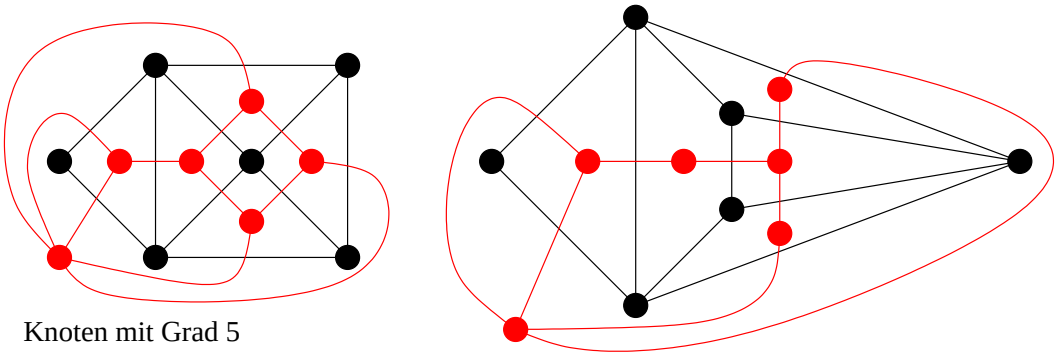


Abbildung 2.17: Verschiedene Dualgraphen desselben Graphen.

Lemma 2.9. *Sei $G = (V, E)$ ein planarer Graph und $G^* = (V^*, E^*)$ sein Dualgraph (bzgl. einer festen Einbettung).*

$S \subseteq E$ bildet einen Schnitt in G genau dann, wenn die Menge der entsprechenden Kanten $S^* \subseteq E^*$ Kreise in G^* bildet.

$S \subseteq E$ *bildet Kreise in* G *genau dann, wenn die Menge der entsprechenden Dualkanten* $S^* \subseteq E^*$ *einen Schnitt in* G^* *bildet.*

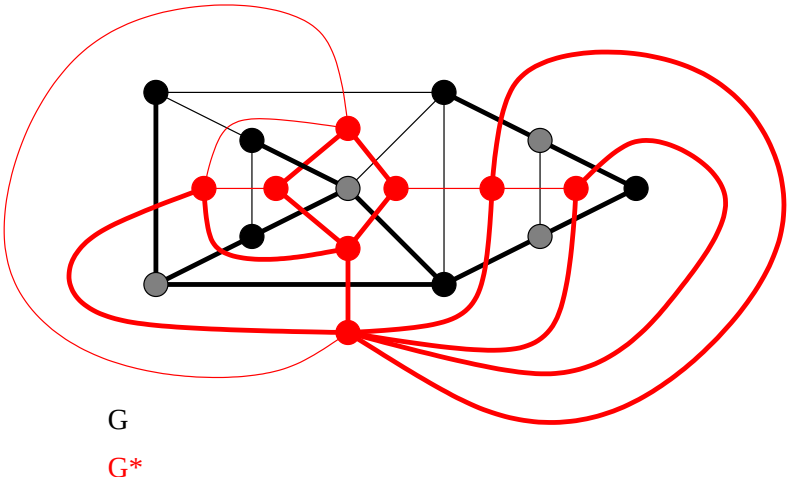


Abbildung 2.18: Illustration zu Lemma 2.9.

Beweis. Da $(G^*)^* = G$ ist, sind die beiden Fälle natürlich äquivalent. Sei o.B.d.A. G zusammenhängend und S^* eine Menge von Kreisen im Dualgraph G^* (bzgl. einer festen Einbettung). Die Kanten aus G, die dual zu S^* sind, trennen gerade alle Knoten von G im Inneren der Kreise S^* von den Knoten des Äußeren der Kreise von S^*. □

2.4 Suchmethoden in planaren Graphen

Bekannte Suchmethoden in Graphen sind die *Tiefensuche* (DFS) und die *Breitensuche* (BFS), und werden in der Grundvorlesung „Algorithmen und Datenstrukturen“ behandelt.

Die Grundidee der Tiefensuche besteht darin, dass der Graph systematisch „durchsucht“ wird, d.h. Knoten und Kanten werden besucht, wobei in einem „Suchschritt“ wenn möglich weitergegangen wird zu einem „neuen“ Knoten. Die Grundidee der Breitensuche besteht darin, dass der Graph systematisch „durchsucht“ wird, wobei in einem Suchschritt zunächst alle Nachbarn eines Knoten besucht werden, bevor von dem als ersten besuchten Nachbarn aus weitergegangen wird. Konkrete Implementationen einer Tiefen- bzw. Breitensuche hängen u.a. davon ab, welches „Ergebnis“ gewünscht wird.

Knoten-DFS. Wähle einen beliebigen Knoten aus, markiere ihn und lege ihn auf einen „Stapel“. Solange der Stapel noch einen Knoten enthält, betrachte den obersten Knoten v auf dem Stapel.

Suchschritt an v:
Falls v einen unmarkierten Nachbarn besitzt, markiere ihn und lege ihn auf den Stapel, und „speichere“ die entsprechende Kante.

backtrack:
Ansonsten entferne v von dem Stapel.

Wahlfreiheit:
Welcher unmarkierte Nachbar bzw. welche Kante zu einem unmarkierten Nachbar wird ausgewählt?

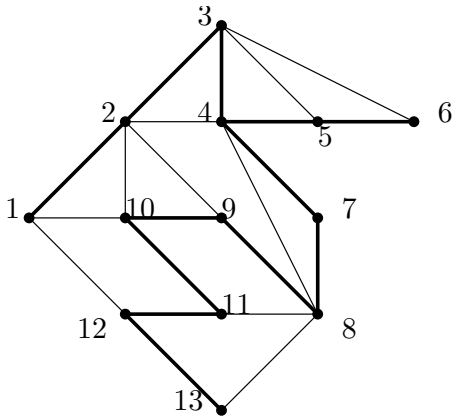


Abbildung 2.19: Ein Beispiel einer Knoten-DFS.

Knoten-BFS. Wähle einen beliebigen Knoten aus, markiere ihn und hänge ihn an eine „Warteschlange“ an. Solange die Warteschlange noch einen Knoten enthält, betrachte den vordersten Knoten v in der Warteschlange.

Suchschritt an v:
Falls v einen unmarkierten Nachbarn besitzt, markiere ihn und hänge ihn an die Warteschlange, und speichere die entsprechende Kante.

backtrack:
Ansonsten entferne v aus der Warteschlange.

Wieder gibt es Wahlfreiheit in jedem Suchschritt. Üblicherweise kann diese Wahl dadurch festgelegt werden, wie der Graph „organisiert“ ist. Das kann z.B. folgendermaßen sein: Für jeden Knoten gebe es eine Liste seiner Nachbarn. Wähle den nächsten unmarkierten Knoten in der Liste. Bei planaren Graphen könnte diese Liste entsprechend der Einbettung des Graphen im „Gegenuhrzeigersinn“ zyklisch angeordnet sein. Als *nächste* Kante kann dann die Kante direkt *links* neben der vorherigen Kante angesehen werden. Dies

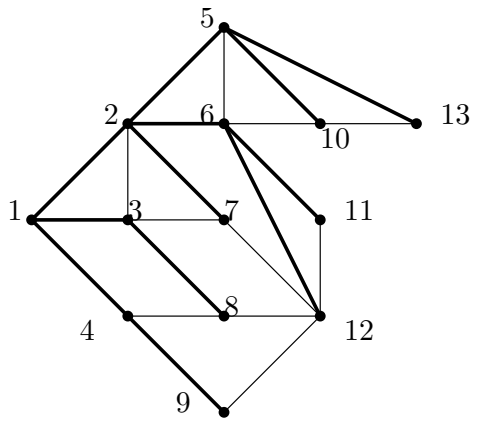


Abbildung 2.20: Ein Beispiel einer Knoten-BFS.

ist gleichzeitig die *rechteste* bisher unbesuchte Kante relativ zur vorherigen Kante. Dies führt zur *RIGHT-FIRST* bzw. *LEFT-FIRST* Auswahl.

RIGHT-FIRST-Kanten-DFS. Wähle eine beliebige Kante e mit Endknoten u und v aus. Orientiere sie von $u \longrightarrow v$ und lege sie auf einen Stapel. Solange der Stapel noch eine Kante enthält, betrachte die oberste Kante e auf dem Stapel.

Suchschritt an e:
Falls der „Einlaufknoten“ von e inzident ist zu einer nichtorientierten Kante, so orientiere die *rechteste* nichtorientierte Kante relativ zu e und lege sie auf den Stapel.

backtrack:
Ansonsten entferne e vom Stapel.

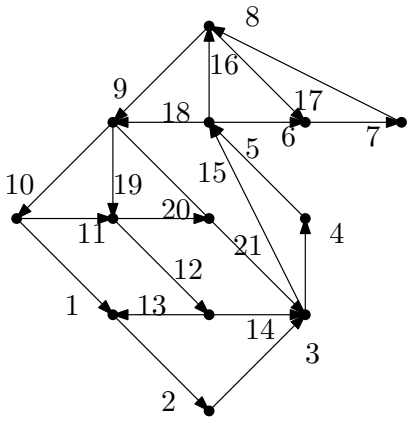


Abbildung 2.21: RIGHT-FIRST Kanten-DFS.

LEFT-FIRST-Kanten-BFS. Wähle eine beliebige Kante mit Endknoten u und v aus. Orientiere sie von $u \longrightarrow v$ und hänge e an eine Warteschlange. Orientiere alle von u ausgehenden Kanten von $u \longrightarrow w$, wobei w anderer Endknoten und hänge sie der Reihe nach von links nach rechts relativ zu e an die Warteschlange an (siehe Abb. 2.22). Solange die Warteschlange nicht leer ist, betrachte den Einlaufknoten v der vordersten Kante e.

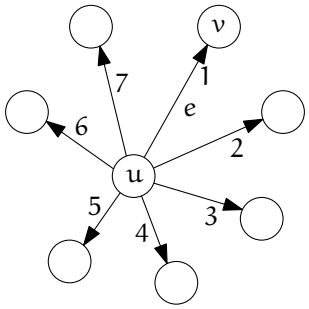


Abbildung 2.22: Reihenfolge der Kanten in der Warteschlange bezüglich Startkante e.

Suchschritt an e:
Falls dieser inzident ist zu einer nichtorientierten Kante, so orientiere die linkeste nichtorientierte Kante relativ zu e von $v \longrightarrow u$, wobei u anderer Endknoten und hänge sie an die Warteschlange an.

backtrack:
Ansonsten entferne e aus der Warteschlange.

3 Färbung planarer Graphen

In der Einführung haben wir bereits das Färbungsproblem angesprochen. In diesem Kapitel werden wir beweisen, dass jeder planare Graph mit fünf Farben gefärbt werden kann. Tatsächlich ist jeder planare Graph sogar vierfärbbar. Der Beweis des Vierfarbensatzes ist allerdings zu aufwendig, um ihn in der Vorlesung zu behandeln.

KNOTENFÄRBUNGSPROBLEM
Gegeben sei ein Graph $G = (V, E)$. Färbe die Knoten aus V mit möglichst wenigen Farben so ein, dass benachbarte Knoten verschiedene Farben haben.

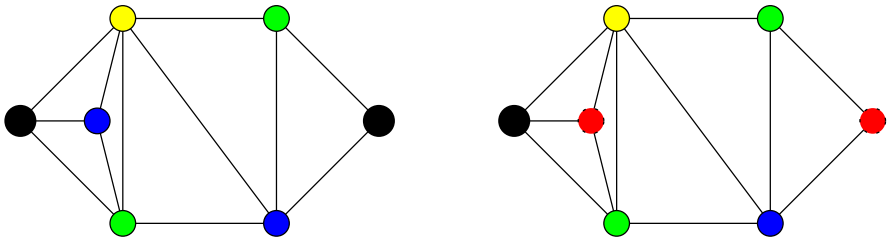


Abbildung 3.1: Beispiele von Knotenfärbungen

Bezeichne $\chi(G)$ die minimale Anzahl an Farben, die benötigt wird um G zulässig zu färben ($\chi(G)$ heißt auch „chromatische Zahl“ von G) und $\mathrm{cl}(G)$ die Cliquenzahl von G, d.h das maximale $t \leq |V|$, so dass G einen K_t als knoteninduzierten Subgraphen enthält. Offensichtlich ist $\chi(G) \geq \mathrm{cl}(G)$.

Satz 3.1. *Jeder planare Graph kann mit fünf Farben zulässig gefärbt werden.*

Beweis. Wir führen eine Induktion über die Anzahl der Knoten n. Für $n \leq 5$ gilt die Behauptung trivialerweise. Sei also jeder planare Graph mit höchstens $n - 1$ Knoten fünffärbbar, und G habe n Knoten. G enthält mindestens einen Knoten v mit $d(v) \leq 5$.

Fall 1: Es existiert ein Knoten v mit $d(v) \leq 4$. Betrachte Einbettung von G, mit $v, w_1, \ldots, w_4$ wie folgt:

Entferne v und die entsprechenden Kanten $\{v, w_i\}$ aus G. Dann entsteht der Graph $G - v$ mit $n - 1$ Knoten, der per Induktionsannahme fünffärbbar ist. Eine Fünffärbung von $G - v$ induziert dann eine Fünffärbung von G, wobei v gerade mit der Farbe gefärbt wird, die für keinen der w_i benutzt wurde.

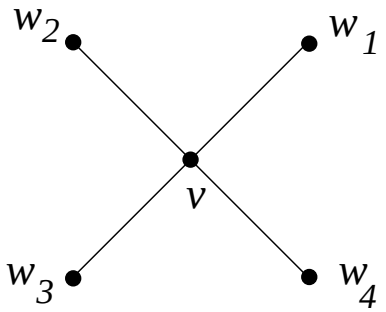


Abbildung 3.2: Illustration von Fall 1.

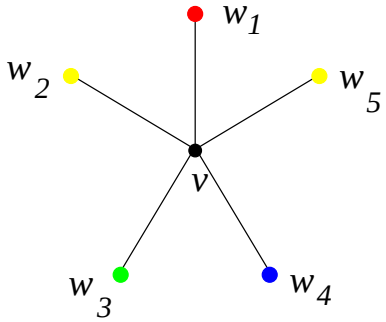


Abbildung 3.3: Illustration zu Fall 2.

Fall 2: Der minimale Grad in G ist fünf und v sei ein Knoten mit $d(v) = 5$. Betrachte wieder eine Einbettung von G wie in Abbildung 3.3.

Entferne v und die entsprechenden Kanten $\{v, w_i\}$ und färbe wieder G induktiv basierend auf einer Fünffärbung von $G - v$.

Fall 2.1: Falls für $w_1, \ldots, w_5$ nicht alle fünf Farben bei einer Fünffärbung von $G - v$ verwendet werden, kann G wie in Fall 1 gefärbt werden.

Fall 2.2: Jedes w_i hat eine eigene Farbe i, $1 \leq i \leq 5$. Betrachte den Subgraph H von $G - v$, der durch die Knoten mit Farben 1 und 4 induziert wird.

Falls w_1 und w_4 in H in verschiedenen Zusammenhangskomponenten H_1 bzw. H_4 liegen, so vertausche in H_1 die Farben 1 und 4 und färbe v in G mit Farbe 1.

Falls w_1 und w_4 in H verbunden sind, betrachte analog w_2 und w_5 und den durch Farben 2 und 5 induzierten Subgraph H'. Wegen der Planarität von G sind w_2 und w_5 in H' nicht verbunden. Durch Vertauschen der Farben 2 und 5 in der Zusammenhangskomponente von w_2 in H' und Färben von v mit Farbe 2 erhält man dann wieder eine Fünffärbung von G.

Dieser Beweis induziert einen einfachen Algorithmus um planare Graphen mit fünf Farben zu färben.

Fünffärbungsalgorithmus

Schritt 1: Sortiere die Knoten in der Reihenfolge $v_1, \ldots, v_n$, so dass $d_{G_i}(v_i) \leq 5$, wobei $G_1 := G$ und $G_i := G - \{v_1, \ldots, v_{i-1}\}$ für $i \geq 2$.

Schritt 2: Färbe dann zunächst G_n, dann G_{n-1} usw. wie im Beweis von Satz 3.1.

Versuch einer Vierfärbung mittels der gleichen Technik wie in Satz 3.1.

Ist diese Vorgehensweise anwendbar um zu einer Vierfärbung eines planaren Graphen zu kommen? Betrachte einen maximal planaren Graphen $G = (V, E)$. Existiert ein $v \in V$, mit $d(v) \leq 3$, so kann wie in Fall 1 vorgegangen werden. Falls ein $v \in V$, mit $d(v) = 4$ existiert, so kann aus einer Vierfärbung von $G - v$ eine Vierfärbung von G analog zum Beweis von Satz 3.1, Fall 2 konstruiert werden. Sei also der minimale Grad eines Knoten 5 und $v \in V$ mit $d(v) = 5$. Angenommen alle vier Farben *rot, blau, gelb, grün* würden für die Vierfärbung von $w_1, \ldots, w_5$ verwendet. Seien o.b.d.A. die Knoten w_2 und w_5 mit der Farbe *gelb* gefärbt, w_1 *rot*, w_3 *grün* und w_4 *blau* und sei H der Subgraph, der durch `rot` und `blau` induziert wird. Sind w_1 und w_4 nicht in H verbunden, so können `rot` und `blau` in einer der Komponenten vertauscht werden. Ebenso kann man vorgehen, wenn w_1 und w_3 nicht in dem entsprechenden Subgraphen verbunden sind.

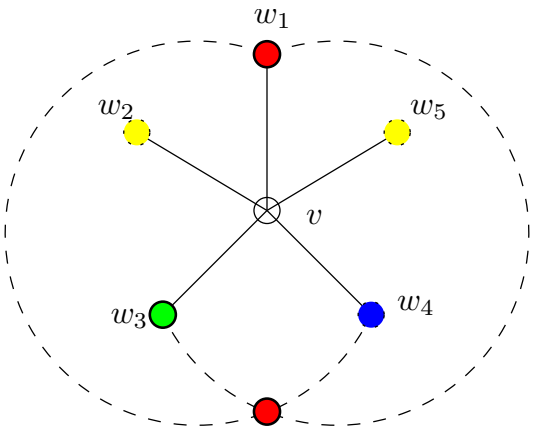


Abbildung 3.4: Illustration zur Vierfärbung.

Seien also w_1 und w_4 und w_1 und w_3 jeweils in dem rot-blauen bzw. rot-grünen Subgraphen verbunden. Dann sind w_2 und w_5 aber weder in dem gelb-blauen noch in dem gelb-grunen Subgraphen verbunden. Die gelb-grüne Komponente H_5, die w_5 enthält, enthält nicht w_2 und w_3, und die gelb-blaue Komponente H_2, die w_2 enthält, enthält nicht w_4 und w_5.

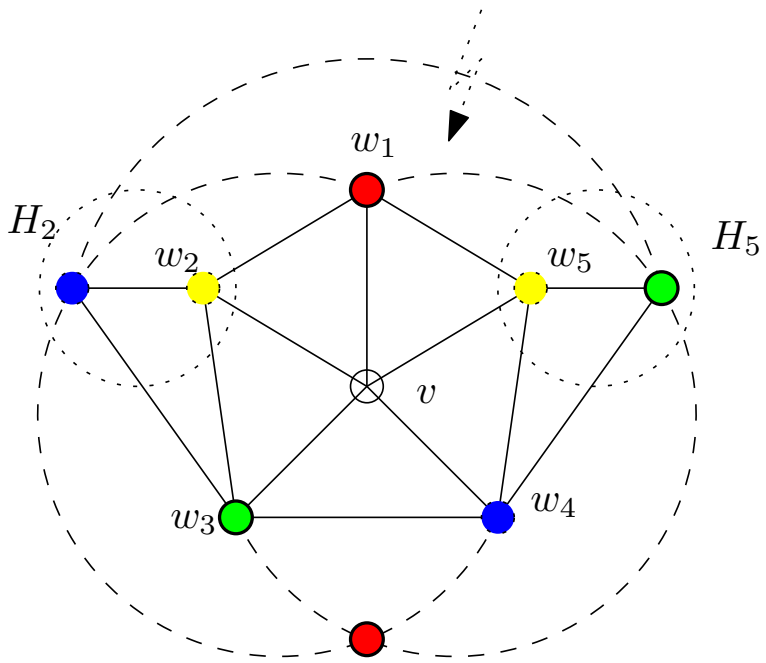


Abbildung 3.5: Gegenbeispiel zur Vierfärbung

Vertausche also in H_5 gelb mit grün und in H_2 gelb mit blau. Dann kann gelb für v verwendet werden. Dies kann jedoch zu einer unzulässigen Knotenfärbung führen, siehe Abbildung 3.5.

4 Separatoren in planaren Graphen

Eine Menge $S \subset V$ heißt *Separator* von $G = (V, E)$, falls der durch $V \backslash S$ induzierte Subgraph von G unzusammenhängend ist. S *trennt* die Knoten $u, v \in V \backslash S$, falls u und v in dem durch $V \backslash S$ induzierten Subgraph (bezeichnet mit $G - S$) in verschiedenen Zusammenhangskomponenten liegen.

Separatoren spielen beispielsweise bei Graphenalgorithmen, die auf dem *Divide-and-Conquer Prinzip* beruhen eine wichtige Rolle. Siehe dazu die Vorlesung „Algorithmentechnik“. Dazu zerlegt man einen Graphen durch Wegnahme eines Separators und wendet den Algorithmus rekursiv auf die entstandenen Subgraphen an. Da die Größe des Separators beim Zusammensetzen der „Teillösungen“ in die Laufzeit eingeht, benutzt man typischerweise *kleine Separatoren*. Für die Gesamtlaufzeit eines solchen Divide-and-Conquer Algorithmus ist jedoch auch die *Rekursionstiefe* des Verfahrens wichtig. Diese hängt von der Größe der entstehenden Subgraphen ab. Ideal wären balancierte Zerlegungen, bei denen die Subgraphen der Zerlegung etwa gleich groß sind. Dies führt zu folgendem Optimierungsproblem.

> Minimum-Balanced-Separator-Problem
> Gegeben sei ein Graph $G = (V, E)$. Finde eine Partition von V in drei Mengen V_1, V_2 und S, wobei S Separator minimaler Kardinalität ist, der V_1 und V_2 trennt mit $|V_1|,\ |V_2| \leq \alpha \cdot |V|$ und $\frac{1}{2} \leq \alpha < 1$ konstant.

Das Problem ist für beliebige Graphen $\mathcal{NP}$-schwer. Ist $\alpha = \frac{1}{2}$, so nennt man das Problem Minimum-Bisection-Problem. Es ist nicht bekannt, ob das Minimum-Bisection-Problem auch für planare Graphen $\mathcal{NP}$-schwer ist. Allerdings lässt sich für planare Graphen in linearer Laufzeit ein Separator finden, für dessen Größe und Balanciertheit der Zerlegung sich noch eine gewisse Garantie beweisen lässt. Dahinter steht der folgende *Satz von Lipton & Tarjan* (bewiesen 1977), der auch als Planar-Separator-Theorem bezeichnet wird.

Satz 4.1 (Planar-Separator-Theorem). *Die Knotenmenge eines zusammenhängenden, planaren Graphen $G = (V, E)$, $n = |V| \geq 5$, kann so in drei Mengen $V_1, V_2, S \subseteq V$ partitioniert werden, dass*

1. $|V_1|,\ |V_2| \leq \frac{2}{3} \cdot n$,
2. *S Separator, der V_1 und V_2 trennt,*
3. $|S| \leq 4 \cdot \sqrt[2]{n}$.

Diese Partition kann in Laufzeit $\mathcal{O}(n)$ berechnet werden.

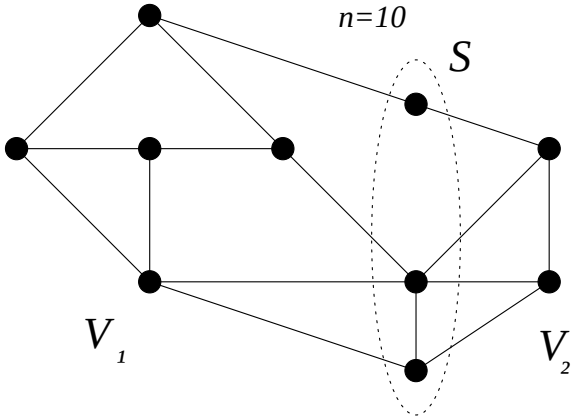


Abbildung 4.1: Ein Separator, der die Bedingungen des Planar-Separator-Theorems erfüllt.

Zur Illustration des Satzes siehe Abb. 4.1. Bevor wir Satz 4.1 beweisen, benötigen wir noch einige Begriffe. Ein Subgraph $T = (V(T), E(T))$ eines Graphen $G = (V, E)$, heißt *aufspannender Baum* von G, falls T Baum ist und $V(T) = V$. Ein beliebiger Knoten eines Baumes T kann als *Wurzel* w ausgezeichnet sein. Dann ist das *Level* oder die *Höhe eines Knotens* v definiert als die Länge des eindeutigen Weges vom Knoten v zur Wurzel und wird mit level(v) bezeichnet. Die *Höhe von* T ist die Länge des längsten Weges von w zu einem Knoten aus T. Sei $G = (V, E)$ ein planarer (eingebetteter) Graph. $G' = (V, E')$ heißt *Triangulierung von* G, falls G' ein kantenmaximaler planarer Graph ist, der G als Subgraph enthält. In einer Einbettung von G' sind alle Facetten Dreiecke. Zu einem eingebetteten planaren Graphen kann in Zeit $\mathcal{O}(n)$ eine Triangulierung konstruiert werden (Übung).

Für den Beweis von Satz 4.1 werden wir folgendes Lemma verwenden.

Lemma 4.2. *Sei $G = (V, E)$ ein planarer, zusammenhängender Graph mit $|V| = n \geq 5$ und $T = (V, E(T))$ ein aufspannender Baum von G mit Wurzel w und Höhe h. Die Knotenmenge von G kann so in drei Mengen V_1, V_2 und S partitioniert werden, dass*

1. $|V_1|, |V_2| \leq \frac{2}{3} \cdot n$,
2. *S Separator, der V_1 und V_2 trennt,*
3. $|S| \leq 2 \cdot h + 1$.

Eine solche Partition kann in $\mathcal{O}(n)$ konstruiert werden.

Beweis. G wird zunächst durch Hinzufügen von Kanten trianguliert, d.h. die Facetten werden zu Dreiecken gemacht. Dies ist in $\mathcal{O}(n)$ möglich. Der so konstruierte Graph enthält also genau $3n - 6$ Kanten. Dementsprechend hat er nach dem Satz von Euler genau $2n - 4$ Facetten. Ein aufspannender Baum T des Ausgangsgraphen ist natürlich auch ein aufspannender Baum des triangulierten Graphen.

Basierend auf einem aufspannenden Baum T der Höhe h mit ausgezeichneter Wurzel suchen wir einen Kreis im triangulierten Graphen, der höchstens Länge $2 \cdot h + 1$ hat, und dessen Inneres und Äußeres jeweils höchstens $\frac{2}{3}n$ Knoten enthalten. Die Knoten dieses Kreises bilden dann den gewünschten Separator S von G. Siehe Abb. 4.2.

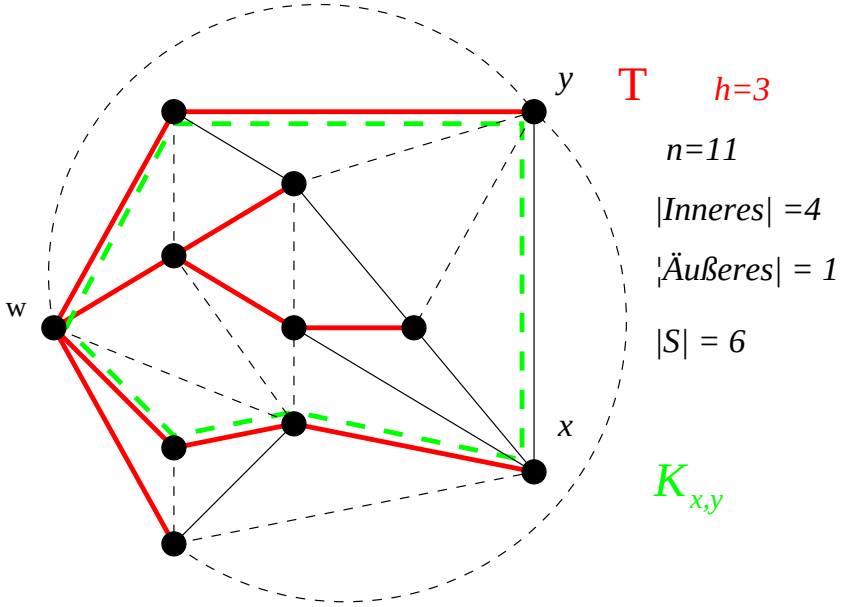


Abbildung 4.2: Illustration eines aufspannenden Baumes in einem planaren Graphen und dem durch eine Nichtbaumkante $\{x, y\}$ induzierten Kreis $K_{x,y}$. Die gestrichelten Kanten sind bei der Triangulierung zum Graphen hinzugefügt worden, die roten, fetten Kanten bilden einen aufspannenden Baum.

Jede Kante $\{x, y\} \in E$ ist entweder auch in $E(T)$, also eine *Baumkante*, oder in $E \setminus E(T)$, also eine *Nichtbaumkante*. Jede Nichtbaumkante $\{x, y\} \in E \setminus E(T)$ induziert einen Kreis, den Kreis $K_{x,y}$ bestehend aus $\{x, y\}$ und den Wegen von x bzw. y zum gemeinsamen Vorgänger maximalen Levels im Baum. Die Anzahl der Knoten auf $K_{x,y}$ ist höchstens $2 \cdot h + 1$.

Betrachte Inneres$(K_{x,y})$ und Äußeres$(K_{x,y})$, d.h. die Knoten und Kanten, die im Inneren von $K_{x,y}$ bzw. in dessen Äußeren eingebettet sind, und bezeichne mit $|\text{Inneres}(K_{x,y})|$ bzw. $|\text{Äußeres}(K_{x,y})|$ die Anzahl der Knoten im Inneren bzw. Äußeren von $K_{x,y}$. Wir wählen eine beliebige Nichtbaumkante $\{x, y\} \in E \setminus E(T)$, wobei o.B.d.A. $|\text{Inneres}(K_{x,y})| \geq |\text{Äußeres}(K_{x,y})|$. Wenn zusätzlich gilt, dass $|\text{Inneres}(K_{x,y})| \leq \frac{2}{3}n$ ist, sind wir fertig.

Sei also $|\text{Inneres}(K_{x,y})| > \frac{2}{3}n$ (beachte: dies impliziert $|\text{Äußeres}(K_{x,y})| < \frac{1}{3}n$). Wir verkleinern nun systematisch das Innere, indem wir eine geeignete Nichtbaumkante e im Inneren von $K_{x,y}$ suchen, für die $|\text{Inneres}(K_e)| \leq \frac{2}{3}n$ wird, und $|\text{Äußeres}(K_e)| \leq \frac{2}{3}n$ bleibt.

Die Kante $\{x, y\}$ begrenzt zwei Dreiecke, von denen eines im Inneren von $K_{x,y}$ liegt. Betrachte den Knoten t, der mit x und y dieses Dreieck bildet.

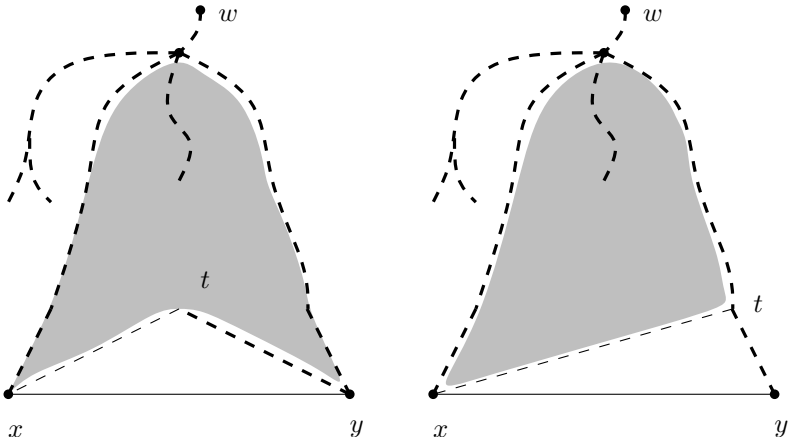


Abbildung 4.3: Illustration von Fall 1, $t \notin K_{x,y}$ bzw. $t \in K_{x,y}$.

Fall 1: Eine der beiden Kanten ist eine Baumkante, o.B.d.A. $\{y, t\} \in E(T)$ (siehe Abb. 4.3).
Ersetze $\{x, y\}$ durch $\{x, t\}$ und betrachte nun $K_{x,t}$. Dann gilt:

$$\text{Falls } t \notin K_{x,y}: \quad \begin{aligned} &|\,\text{Äußeres}(K_{x,t})| = |\,\text{Äußeres}(K_{x,y})| \\ &|\,\text{Inneres}(K_{x,t})| = |\,\text{Inneres}(K_{x,y})| - 1 \end{aligned} \qquad \text{und}$$

$$\text{Falls } t \in K_{x,y}: \quad \begin{aligned} &|\,\text{Äußeres}(K_{x,t})| = |\,\text{Äußeres}(K_{x,y})| + 1 \\ &|\,\text{Inneres}(K_{x,t})| = |\,\text{Inneres}(K_{x,y})| \end{aligned} \qquad \text{und}$$

Ersetzung von $\{x, y\}$ durch $\{x, t\}$ verkleinert also $|\,\text{Inneres}(K_{x,y})|$ bzw. lässt $|\,\text{Inneres}(K_{x,y})|$ zumindest unverändert, und lässt $|\,\text{Äußeres}(K_{x,t})|$ klein genug.

Fall 2: Beide Kanten $\{x, t\}$ und $\{y, t\}$ sind Nichtbaumkanten (siehe Abb. 4.4).
O.b.d.A. sei $|\,\text{Inneres}(K_{x,t})| \geq |\,\text{Inneres}(K_{y,t})|$. Ersetze $\{x, y\}$ durch $\{x, t\}$. Dann gilt

$$\begin{aligned} &|\,\text{Äußeres}(K_{x,t})| \leq n - \frac{1}{2}\,|\,\text{Inneres}(K_{x,y})| \leq \frac{2}{3}n \qquad \text{und} \\ &|\,\text{Inneres}(K_{x,t})| \leq |\,\text{Inneres}(K_{x,y})| - 1 \ . \end{aligned}$$

Ersetzung von $\{x, y\}$ durch $\{x, t\}$ verkleinert also $|\,\text{Inneres}(K_{x,y})|$ und lässt $|\,\text{Äußeres}(K_{x,t})|$ klein genug. Dies kann nun so lange wiederholt werden, bis auch $|\,\text{Inneres}(K_{x,y}) \leq \frac{2}{3}n$ gilt.

Damit haben wir bewiesen, dass sich eine Partition mit den gewünschten Eigenschaften konstruieren lässt. Wir müssen nun noch deren Implementation in linearer Laufzeit sicherstellen.

Implementation in linearer Laufzeit: Da beim Übergang von einer Nichtbaumkante zu einer neuen Nichtbaumkante im Inneren des betrachteten Kreises sich die Anzahl der

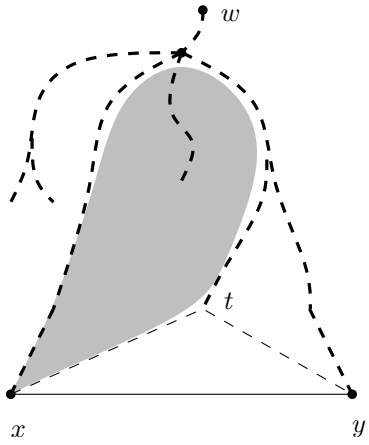


Abbildung 4.4: Illustration von Fall 2.

Dreiecke im Inneren reduziert, endet das Verfahren nach spätestens $2n - 4$ solchen Übergängen. Wäre also jede der Verkleinerungsoperationen in konstanter Zeit realisierbar, wäre damit auch die gesamte Konstruktion in Gesamtlaufzeit $\mathcal{O}(n)$ möglich.

Alle Operationen, bis auf die Entscheidung welches Innere, $|\text{Inneres}(K_{x,t})|$ oder $|\text{Inneres}(K_{y,t})|$ größer ist in Fall 2, bzw. zu entscheiden wann das Innere des betrachteten Kreises nur noch höchstens $\frac{2}{3}n$ Knoten enthält, sind in konstanter Zeit möglich. Allerdings kann eine einzelne Entscheidung ob $|\text{Inneres}(K_{x,t})| < |\text{Inneres}(K_{y,t})|$ bzw. $|\text{Inneres}(K_{x,y})| \leq \frac{2}{3}n$ mehr als konstante Zeit erfordern. Daher führen wir eine amortisierte Analyse durch. Wir überlegen uns zunächst genauer, wie wir für einen Kreis $K_{x,y}$ die Größe $|\text{Inneres}(K_{x,y})|$ bestimmen können.

Zu Beginn des Verfahrens wird der Baum T von den Blättern zur Wurzel w hin durchlaufen, und für jede ausgehende Kante jedes Knotens gespeichert, wieviele Knoten der bezüglich dieser Kante rechte bzw. linke Unterbaum des Knotens enthält. Beginnend mit der gewählten Nichtbaumkante $\{x, y\}$ werden alle Knoten auf dem Weg von x bzw. y zum gemeinsamen Vorgänger im Baum markiert, und gleichzeitig aus den Unterbäumen im Inneren von $K_{x,y}$ der Wert $|\text{Inneres}(K_{x,y})|$ berechnet.

Bei jedem Verkleinerungsschritt entsprechend Fall 2 wird nun zunächst vom Knoten t im Baum nach oben gelaufen bis zum ersten markierten Knoten. Dieser ist der Vorgänger v von t auf $K_{x,y}$. Alle Knoten auf dem Weg von t zu v werden dabei markiert und die Anzahl a_t dieser Knoten sowie die Anzahl r_t der Knoten rechts und l_t links des Weges berechnet (siehe Abb. 4.5).

Um $|\text{Inneres}(K_{x,t})|$ und $|\text{Inneres}(K_{y,t})|$ zu berechnen, werden gleichzeitig die Ränder der beiden Kreise bis v entlanggelaufen und zwar abwechselnd Knoten für Knoten beginnend mit x bzw. y, und die Anzahl der Knoten im Inneren zu I_x bzw. I_y aufaddiert. Sobald mit Erreichen von v der Rand eines der beiden Kreise vollständig abgelaufen ist, wird abgebrochen. Die Anzahl der Knoten im Inneren des anderen Kreises kann nun mit den

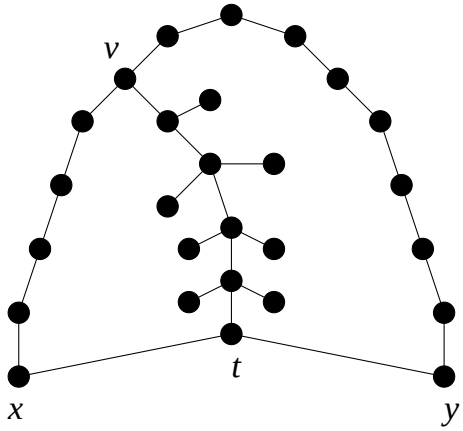


Abbildung 4.5: Illustration der „Aufsummierung“ der Knoten rechts und links des Weges von t zu $K_{x,y}$.

Werten $|\text{Inneres}(K_{x,y})|$ und a_t „rückgerechnet“ werden: Sei o.B.d.A der Knoten v von x aus zuerst erreicht worden, dann gilt

$$|\text{Inneres}(K_{x,t})| = I_x + l_t \qquad \text{und}$$
$$|\text{Inneres}(K_{y,t})| = I_y + r_t = |\text{Inneres}(K_{x,y})| - I_x - a_t - l_t$$

Entscheidend ist nun, dass die Anzahl der Schritte bei dieser Vorgehensweise proportional zur Anzahl der Knoten in dem Teil von $K_{x,y}$ ist, der nicht weiter betrachtet wird. Insgesamt ist die Anzahl der Schritte also amortisiert linear in der Anzahl der Knoten von G, also in $\mathcal{O}(n)$. □

Im Beweis zu Satz 4.1 benutzen wir folgende Eigenschaft eines Breitensuchbaumes (BFS-Baum).

Lemma 4.3. *Zu einem Graph $G = (V, E)$ sei $T = (V, E(T))$ ein BFS-Baum ausgehend von einer beliebigen Wurzel. Eine Nichtbaumkante verbindet Knoten desselben Levels oder direkt aufeinander folgender Level, d.h. für $\{u, v\} \in E \backslash E(T)$ gilt $|\text{level}(u) - \text{level}(v)| \leq 1$.*

Beweis. Angenommen $\{u, v\}$ sei Nichtbaumkante zu einem BFS-Baum mit $|\text{level}(v) - \text{level}(u)| > 1$. O.b.d.A. sei $\text{level}(u) < \text{level}(v)$. Der unmittelbare Vorgänger von v in T muss nach u durchsucht werden, da sein Level mindestens um 1 größer ist als Level u. Wenn $\{u, v\} \in E$ muss v dann in der BFS aber bereits von u aus „entdeckt“ worden sein.

Wir können nun Satz 4.1 beweisen:

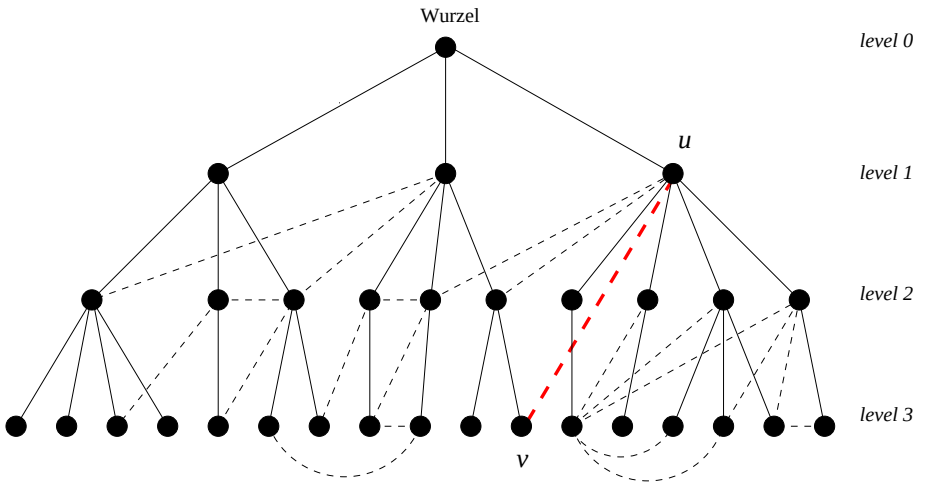


Abbildung 4.6: Illustration zu Lemma 4.3.

Beweis. Konstruiere eine Triangulierung von G und einen BFS-Baum T mit beliebiger Wurzel. Seien dessen Level angefangen mit der Wurzel die Level $0, 1, \ldots, h$ und bezeichne S_i $(0 \leq i \leq h)$ die Menge der Knoten in Level i. Sei μ $(0 \leq \mu \leq h)$ das Level mit der Eigenschaft

$$\sum_{i=0}^{\mu-1} |S_i| \leq \frac{n}{2} \quad \text{und} \quad \sum_{i=0}^{\mu} |S_i| > \frac{n}{2}.$$

Falls $|S_\mu| \leq 4 \cdot \sqrt{n}$ und $\mu < h$, so setze $S := S_\mu$, $V_1 := \bigcup_{i=0}^{\mu-1} S_i$ und $V_2 := \bigcup_{i=\mu+1}^{h} S_i$. Dann ist V_1, V_2 und S eine Partition von V mit den gewünschten Eigenschaften.

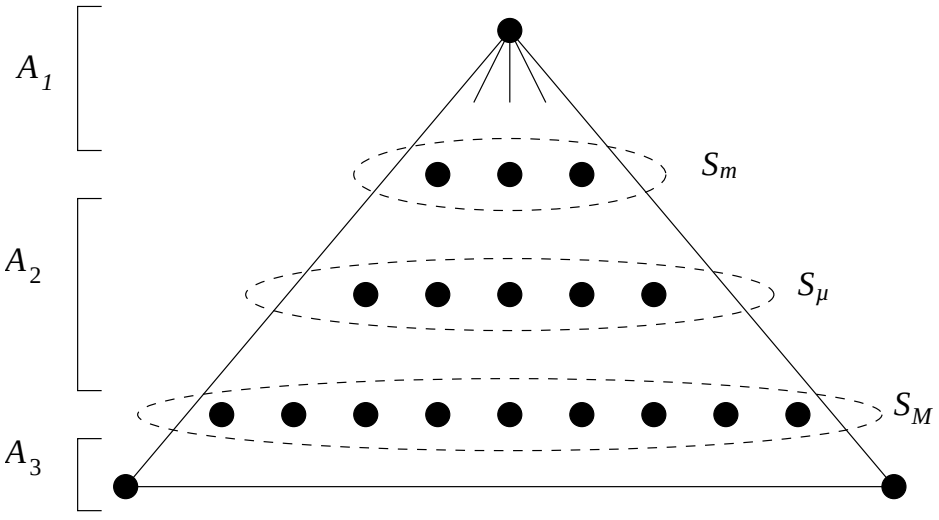


Abbildung 4.7: Illustration der Aufteilung der Level.

Ansonsten sei m das unterste Level oberhalb von Level μ und M das oberste Level unterhalb von Level μ ($0 \leq m \leq \mu \leq M \leq h+1$) mit $|S_m| \leq \sqrt{n}$ und $|S_M| \leq \sqrt{n}$. Beachte, dass möglicherweise $M = h+1$ und $S_M = \emptyset$ ist. Setze (siehe Abb. 4.7)

$$A_1 := \bigcup_{i=0}^{m-1} S_i, \qquad A_2 := \bigcup_{i=m+1}^{M-1} S_i \quad \text{und} \quad A_3 := \bigcup_{M+1}^{h} S_i \ .$$

Basierend auf S_m, S_M und gegebenenfalls weiteren Knoten wird nun ein Separator S mit den gewünschten Eigenschaften konstruiert. Dabei hängt von der Größe von A_2 ab, ob noch weitere Knoten aus dem Bereich zwischen S_m und S_M zu S hinzugefügt werden müssen. Um diese zusätzlichen Knoten zu bestimmen, wird Lemma 4.2 angewendet.

Fall 1: $|A_2| \leq \frac{2}{3}n$

Setze $S := S_m \cup S_M$. S ist Separator von G und zerlegt V in die Knotenmengen A_1, A_2 und A_3. Setze dann $V_1 := A_i$ mit A_i kardinalitätsmaximal unter A_1, A_2 und A_3, und $V_2 := V \backslash (V_1 \cup S)$. Nach Wahl von μ, m und M gilt $|A_1| \leq \frac{n}{2}$ und $|A_3| < \frac{n}{2}$ und nach Vorraussetzung $|A_2| \leq \frac{2}{3}n$. Also ist $|V_1| \leq \frac{2}{3}n$. Zudem gilt

$$|V_2| \leq n - |V_1| \leq n - \frac{|V_2|}{2} \ , \qquad \text{da}$$
$$|V_1| = \max\{|A_1|, |A_2|, |A_3|\} \geq \frac{|V_2|}{2}.$$

Also ist $\frac{3}{2}|V_2| \leq n$ und S, V_1 und V_2 eine Partition von V mit den gewünschten Eigenschaften.

Fall 2: $|A_2| > \frac{2}{3}n$

Verschmelze die Knoten in $A_1 \cup S_m$ zu einem Knoten s durch sukzessives Zusammenziehen von Kanten zwischen Knoten aus $A_1 \cup S_m$. Entferne alle Knoten aus $S_M \cup A_3$ Dadurch entsteht ein Graph $G' = (V', E')$ mit

$$V' := V \backslash \left(\bigcup_{i=0}^{m} S_i \cup \bigcup_{i=M}^{h} S_i \right) \cup \{s\} = A_2 \cup \{s\}$$

und $\{x, y\} \in E'$ genau dann, wenn entweder $x, y \in V' \backslash \{s\}$ und $\{x, y\} \in E$, oder $x, y \in V'$, wobei o.B.d.A. $x = s$ und es existiert ein $z \in S_m$ mit $\{z, y\} \in E$. Dann ist entsprechend $n' := |V'| = |A_2| + 1$. Siehe Abb. 4.8.

Der BFS-Baum T induziert in G' einen BFS-Baum T' mit Wurzel s. Die Höhe h' von T' ist maximal $\sqrt{n}$, da für jedes i, $m < i < M$ gilt $|S_i| > \sqrt{n}$ und $|V'| \leq n$.

Mit Lemma 4.2 existiert eine Zerlegung S', V_1' und V_2' von V' mit $|V_1'|, |V_2'| \leq \frac{2}{3}n' \leq \frac{2}{3}n$ und $|S'| \leq 2 \cdot \sqrt{n} + 1$.

Setze $S := (S' \cup S_m \cup S_M) \backslash \{s\}$. Dann ist S ein Separator von G mit $|S| \leq 4 \cdot \sqrt{n}$, denn sollte $|S'| = 2 \cdot \sqrt{n} + 1$ sein, so enthält S' die Wurzel s von T'.

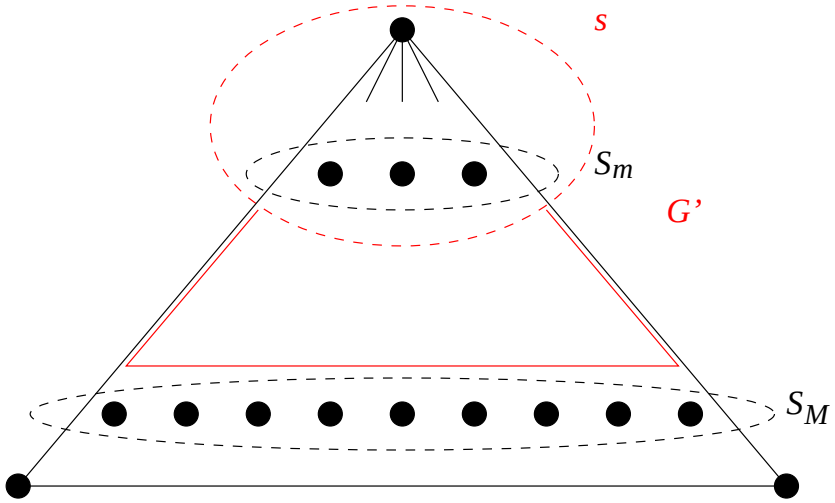


Abbildung 4.8: Illustration zu Fall 2.

Wähle V_1 als die größere der Mengen V_1' und V_2' (gegebenenfalls ohne s) und den Rest von V als V_2 bzw. falls $|V_1'| = |V_2'|$ wähle die Menge, welche nicht s enthält, als V_1, den Rest als V_2 (siehe Abb. 4.9). Dann gilt

$$|V_1| \leq \frac{2}{3}n' \leq \frac{2}{3}n \quad \text{und}$$
$$|V_2| \leq n - (|V_1| + |S|) \leq n - \frac{|A_2|}{2} \leq \frac{2}{3}n. \qquad \square$$

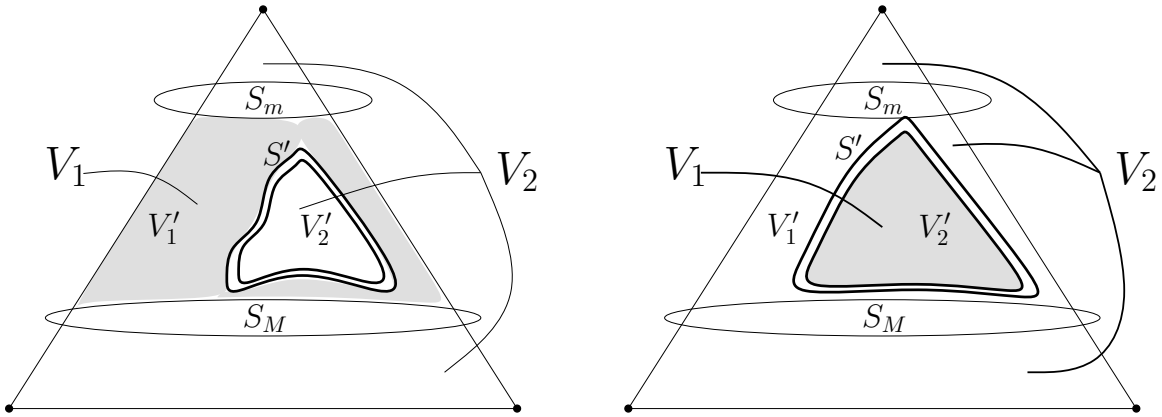


Abbildung 4.9: Konstruktion des Separators im Fall 2; links: $|V_1'| > |V_2'|$, rechts: $|V_2'| > |V_1'|$.

Randfälle des „Planar-Separator-Theorem“:

– Falls $n \leq 2$ oder $G = K_4$ oder $G = K_3$, existiert kein (nichttrivialer) Separator von G. Satz 4.1 ist nur relevant für planare, zusammenhängende Graphen, die einen

(nichttrivialen) Separator besitzen, also ab $n \geq 4$ und $G \neq K_4$ bzw. G Pfad mit drei Knoten. Die entsprechenden Fälle für $n \geq 4$ sind Trivialfälle.

– In Fall 1, Beweis zu Satz 4.1 wird immer eine echte Partition S, V_1, V_2 konstruiert, d.h. S, V_1, $V_2 \neq \emptyset$. Zunächst ist per Voraussetzung $|S_\mu| > 4 \cdot \sqrt{n}$ und daher $|A_2| > 4 \cdot \sqrt{n}$. Andererseits ist $|A_2| \leq \frac{2}{3}n$. Dies ist erst für $n > 36$ möglich. Wäre nun $V_2 = \emptyset$, so müsste $m = 0$ und $M = h$ sein. Da $|S_M| \leq \sqrt{n}$ ist, gilt $|A_2| \geq n - \left(1 + \sqrt{n}\right)$. Für $n > 36$ gilt $1 + \sqrt{n} < \frac{1}{3} \cdot n$, also ist $|A_2| > \frac{2}{3}n$.

Der Beweis des Planar-Separator-Theorems liefert, wie wir gesehen haben, gleichzeitig einen Algorithmus, um einen Separator mit den gewünschten Eigenschaften zu konstruieren. Im Folgenden fassen wir diesen Algorithmus noch einmal zusammen und machen uns dabei klar, dass die Laufzeit tatsächlich in $\mathcal{O}(n)$ ist.

Zusammenfassung des „Separator-Algorithmus“

Schritt 1: Trianguliere G. $\mathcal{O}(n)$

Schritt 2: Berechne einen BFS-Baum. $\mathcal{O}(n)$

Schritt 3: Berechne μ, m und M wie im Beweis zu Satz 4.1. $\mathcal{O}(n)$

Schritt 4: Falls $A_2 = \left| \bigcup_{i=m+1}^{M-1} S_i \right| \leq \frac{2}{3}n$, so berechne S, V_1 und V_2 entsprechend Fall 1 des Beweises zu Satz 4.1. $\mathcal{O}(n)$

Schritt 5: Ansonsten, d.h. falls $\left| \bigcup_{i=m+1}^{M-1} S_i \right| > \frac{2}{3}n$ ist, konstruiere G' wie in Fall 2 des Beweises zu Satz 4.1. $\mathcal{O}(n)$

Schritt 6: Wähle die Nichtbaumkante $\{x, y\}$ und den dadurch induzierten Kreis $K_{x,y}$ in G' wie im Beweis zu Lemma 4.2.
Berechne die Größen der „Unterbäume“ zu allen Knoten in G'.
Berechne $|\text{Inneres}(K_{x,y})|$ und $|\text{Äußeres}(K_{x,y})|$ und sei o.B.d.A. $|\text{Inneres}(K_{x,y})| \geq |\text{Äußeres}(K_{x,y})|$: $\mathcal{O}(n)$

Schritt 7: Solange $|\text{Inneres}(K_{x,y})| > \frac{2}{3}n$ ist, ersetze $\{x, y\}$ und $K_{x,y}$ wie im Beweis zu Lemma 4.2, Fall 1 bzw. Fall 2.
Berechne S, V_1 und V_2 geeignet, d.h. $S := S' \cup S_m \cup S_M$ und o.B.d.A. $V_1 := V_1'$ und $V_2 = V \backslash (V_1 \cup \{s\})$. $\mathcal{O}(n)$

5 Matchings

Das Matching-Problem ist auch für beliebige Graphen in $\mathcal{P}$. Unter Anwendung des Planar-Separator-Theorems kann allerdings ein Divide-and-Conquer Algorithmus für das Matching-Problem in planaren Graphen entworfen werden, der eine kleinere Laufzeit hat, als der effizienteste bekannte Algorithmus zur Bestimmung eines maximalen Matchings in beliebigen Graphen.

In einem Graph $G = (V, E)$ nennt man eine Menge $M \subseteq E$ *Matching*, falls keine zwei Kanten aus M denselben Endknoten haben. Ein Knoten v heißt *ungematcht*, falls v zu keiner Kante aus M inzident ist, ansonsten heißt v *gematcht*.

> Matching-Problem
> Gegeben sei ein Graph $G = (V, E)$ mit Kantengewichtsfunktion $w : E \longrightarrow \mathbb{R}$.
> Finde in G ein Matching M maximalen Gewichts, d.h.
>
> $$w(M) := \sum_{e \in M} w(e)$$
>
> sei maximal unter allen Matchings von G.

Ein Spezialfall dieses Problems besteht darin, ein Matching Maximaler Kardinalität zu berechnen (d.h. $w(e) := 1$ für alle $e \in E$).

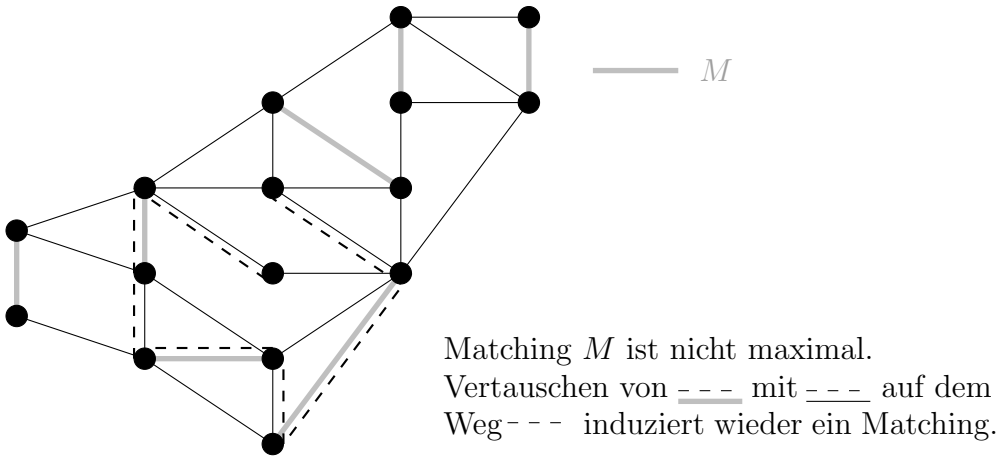


Abbildung 5.1: Illustration der Begriffe Matching und erhöhender Weg.

Ein (bezüglich M) *alternierender Weg* ist ein einfacher Weg oder einfacher Kreis, dessen Kanten abwechselnd in M und in $E\backslash M$ sind. Ein alternierender Weg P, wobei P die Menge der Kanten des Weges bezeichnet, heißt (bezüglich M) *erhöhend* falls

$$\sum_{\substack{e \text{ auf } P,\\ e\in E\backslash M}} w(e) - \sum_{\substack{e \text{ auf } P,\\ e\in M}} w(e) > 0$$

ist, und P entweder ein Kreis gerader Länge ist oder ein Weg, dessen erste und letzte Kante jeweils in M oder inzident zu einem ungematchten Knoten ist (siehe Abb. 5.1).

Beobachtung. *Sei M ein Matching in G und P ein (bezüglich M) erhöhender Weg. Dann ist $M' := (M\backslash P) \cup (P \cap E\backslash M)$ ein Matching von G mit $w(M') > w(M)$.*

Lemma 5.1. *Sei $G = (V, E)$ Graph mit Kantengewichtsfunktion $w : E \longrightarrow \mathbb{R}$. Ein Matching M von G hat genau dann maximales Gewicht, wenn es bzgl. M in G keinen erhöhenden Weg gibt.*

Beweis. Falls es zu M einen erhöhenden Weg gibt, so kann M natürlich nicht maximales Gewicht haben. Umgekehrt nehmen wir an, dass M ein Matching ist, zu dem es einerseits keinen erhöhenden Weg gibt, für das aber andererseits $w(M)$ nicht maximal ist. Dann gibt es ein Matching M^* mit $w(M^*) > w(M)$. Betrachte den Subgraph von G, der durch die Menge $M\Delta M^* := (M \cup M^*)\backslash(M \cap M^*)$ induziert wird. Dieser Graph hat nur Knoten vom Grad 1 oder 2, besteht also aus einfachen Kreisen und Wegen. Wenn nun keiner der Kreise erhöhend bzgl. M ist, so muss es einen inklusionsmaximalen bezüglich M alternierenden Weg P geben mit $w(P \cap M^*) > w(P \cap M)$, da $w(M^*) > w(M)$. Wenn eine Endkante des Weges nicht in M ist, so ist sie in M^*, und daher der entsprechende Endknoten v nicht von M gematcht. Also ist P bzgl. M erhöhend. Widerspruch.

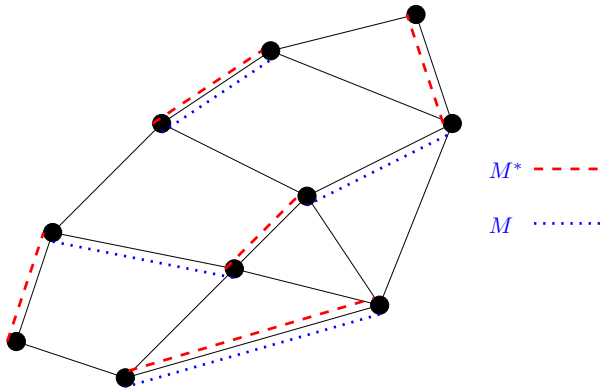


Abbildung 5.2: Illustration von Lemma 5.1.

Lemma 5.2. *Sei $G = (V, E)$ Graph mit Kantengewichtsfunktion $w : E \longrightarrow \mathbb{R}$ und $v \in V$. Weiter sei M ein Matching maximalen Gewichts in $G - v$ (dem durch $V\backslash\{v\}$ induzierten Subgraph von G).*

– *Falls* G *keinen erhöhenden Weg bzgl.* M *mit Endknoten* v *enthält, so ist* M *auch Matching maximalen Gewichts in* G.

– *Ansonsten sei* P *Kantenmenge eines erhöhenden Weges bzgl.* M *in* G *mit* $w(P \cap E\backslash M) - w(P \cap M)$ *maximal unter allen erhöhenden Wegen. Dann ist* $M \Delta P$ *ein Matching maximalen Gewichts in* G.

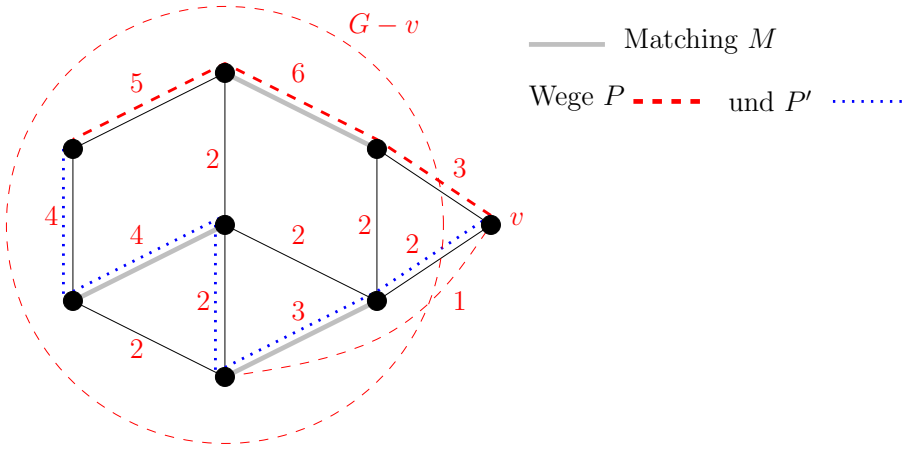


Abbildung 5.3: Illustration von Lemma 5.2.

Beweis. Betrachte ein Matching M maximalen Gewichts in $G-v$. Dann ist M natürlich auch Matching in G. Jeder erhöhende Weg bzgl. M in G muss als Endknoten v haben, ansonsten hätte M nicht maximales Gewicht in $G - v$.

Sei nun M^* ein Matching maximalen Gewichts in G. Wiederum bildet $M \Delta M^*$ eine Menge einfacher bezüglich M^* bzw. M alternierender Wege und Kreise in G. Jeder bezüglich M erhöhende Weg im durch $M \Delta M^*$ induzierten Graph ist auch erhöhend in G. Der durch $M \Delta M^*$ induzierte Graph kann jedoch höchstens einen bezüglich M erhöhenden Weg und zwar mit Endknoten v enthalten, da ansonsten v zu mindestens zwei Kanten aus M^* inzident wäre. Falls P^* ein solcher Weg ist, so hat das durch Erhöhung entlang P^* konstruierte Matching Gewicht

$$w(M) - w(P^* \cap M) + w(P^* \cap E\backslash M) = w(M) - w(P^* \cap M) + w(P^* \cap M^*).$$

Da im durch $M \Delta M^*$ induzierten Graph kein weiterer bezüglich M erhöhender Weg existiert, hat die Menge der Kanten aus M, die nicht auf P^* liegen dasselbe Gewicht wie die Menge der Kanten aus M^*, die nicht auf P^* liegen, d.h. $w(M) - w(P^* \cap M) = w(M^*) - w(P^* \cap M^*)$. Dann hat also das durch Erhöhung entlang P^* konstruierte Matching Gewicht

$$w(M) - w(P^* \cap M) + w(P^* \cap M^*) = w(M^*).$$

Daraus folgt die Behauptung. □

Basierend auf diesem Lemma kann in einem beliebigen Graph $G = (V, E)$, $|E| = m$, aus einem Matching maximaler Kardinalität bzw. maximalen Gewichts in $G - v$ (für beliebiges $v \in V$) ein Matching maximaler Kardinalität bzw. maximalen Gewichts in G konstruiert werden. Die Laufzeit ist $\mathcal{O}(m)$ bzw. $\mathcal{O}(m \log n)$ (siehe Übung (teilweise)). Dies führt für planare Graphen zu einer Laufzeit von $\mathcal{O}(n)$ bzw. $\mathcal{O}(n \log n)$.

Der folgende rekursive Algorithmus findet in planaren Graphen ein Matching maximalen Gewichts bzw. Kardinalität unter Benutzung des „Planar-Separator-Theorems“ und Lemma 5.2.

Divide-and-Conquer-Algorithmus Max-Matching

Schritt 1: Falls G höchstens drei Knoten enthält, bestimme direkt ein Matching maximalen Gewichts.

Schritt 2: Ansonsten zerlege V in V_1, V_2 und S entsprechend dem Planar-Separator-Theorem.
G_1, G_2 bezeichne die durch V_1 bzw. V_2 induzierten Subgraphen von G.
Wende den Algorithmus rekursiv auf G_1 und G_2 an, und berechne so Matchings maximalen Gewichts M_1 bzw. M_2 von G_1 bzw. G_2.
Sei $M := M_1 \cup M_2$, $V' := V_1 \cup V_2$.

Schritt 3: Solange $S \neq \emptyset$ ist, führe aus:
Wähle $v \in S$, und setze $S := S \backslash \{v\}$ und $V' := V' \cup \{v\}$.
Wende Lemma 5.2 an um in dem durch V' induzierten Subgraph von G ein Matching maximalen Gewichts zu berechnen.

Wenn $t'(n)$ die Laufzeit zur Berechnung eines Matchings maximalen Gewichts in einem Graph G mit n Knoten aus einem Matching maximalen Gewichts von $G - v$ ist, und $t(n)$ Laufzeit des Divide-and-Conquer-Algorithmus bezeichnet, so gilt:

$$\begin{aligned} t(n_0) &= c_0, \text{für geeignetes } n_0 \in \mathbb{N} \\ t(n) &\leq t(c_1 \cdot n) + t(c_2 \cdot n) + c_3 \cdot \sqrt{n} \cdot t'(n), \text{für } n > n_0, \end{aligned}$$

wobei c_0, c_1, c_2, c_3 konstant, $c_1, c_2 \leq \frac{2}{3}$ und $c_1 + c_2 < 1$. Man kann mit Techniken zur Analyse von Rekursionabschätzungen (siehe dazu Vorlesung „Algorithmentechnik“) beweisen, dass

$$\begin{aligned} t(n) \in \mathcal{O}(n^{\frac{3}{2}}) \text{ falls } t'(n) \in \mathcal{O}(n), \\ \text{und } t(n) \in \mathcal{O}(n^{\frac{3}{2}} \cdot \log n) \text{ falls } t'(n) \in \mathcal{O}(n \log n). \end{aligned}$$

6 Mixed Max Cut in planaren Graphen und Via-Minimierung

Ein grundlegendes Problem besteht in der Berechnung eines *Schnittes* mit *minimalem* oder mit *maximalem Gewicht*. Die Komplexität dieses Problems ist wesentlich abhängig von der Gewichtsfunktion. Es gibt zahlreiche Anwendungen dieses Problems. Siehe auch Vorlesung „Algorithmentechnik“.

Wir werden einen polynomialen Algorithmus für die Berechnung eines Schnittes mit maximalem Gewicht in planaren Graphen mit beliebigen (positiven und negativen) Kantengewichten konstruieren. Darüber hinaus werden wir eine Anwendung dieses Algorithmus für das Via-Minimierungs-Problem, ein Problem aus dem „VLSI-Design“ (Entwurf hochintegrierter Schaltkreise) kennenlernen.

Eine Menge $S \subseteq E$ heißt *Schnitt* von $G = (V, E)$, falls der durch $E \backslash S$ induzierte Subgraph von G unzusammenhängend ist, d.h. in Graphen $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ zerfällt, mit $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, $E_1 \cup E_2 = E \backslash S$, $E_1 \cap E_2 = \emptyset$, wobei alle Kanten aus S einen Endknoten in V_1 und einen Endknoten in V_2 haben. S *trennt* die Knoten $u, v \in V$, falls u und v in dem durch $E \backslash S$ induzierten Subgraph (bezeichnet mit $G - S$) in verschiedenen Zusammenhangskomponenten liegen. In einem Graph mit Kantengewichtsfunktion $w : E \longrightarrow K$ ist das *Gewicht* eines Schnittes S ist definiert als

$$w(S) := \sum_{e \in S} w(e) \ .$$

> Min-Cut-Problem
> Gegeben sei ein Graph $G = (V, E)$ mit einer Kantengewichtsfunktion $w\colon E \to K$, wobei $K = \mathbb{R}^+$. Finde einen Schnitt $S \subseteq E$ mit $w(S)$ minimal.

Das Min-Cut-Problem ist für beliebige Graphen in polynomialer Zeit lösbar, und zwar in Laufzeit $\mathcal{O}(n \cdot m + n^2 \cdot \log n)$. Siehe Vorlesung „Algorithmentechnik“.

> Max-Cut-Problem
> Gegeben sei ein Graph $G = (V, E)$ mit einer Kantengewichtsfunktion $w : E \longrightarrow K$, wobei $K = \mathbb{R}^+$. Finde einen Schnitt $S \subseteq E$ mit $w(S)$ maximal.

Das Max-Cut-Problem ist für beliebige Graphen $\mathcal{NP}$-schwer.

MIXED-MAX-CUT-PROBLEM
Gegeben sei ein Graph $G = (V, E)$ mit einer Kantengewichtsfunktion $w\colon E \to K$, wobei $K = \mathbb{R}$. Finde einen Schnitt $S \subseteq E$ mit $w(S)$ maximal.

Das MIXED-MAX-CUT-PROBLEM ist für beliebige Graphen natürlich auch $\mathcal{NP}$-schwer. Sowohl MIN-CUT-PROBLEM und MAX-CUT-PROBLEM sind Spezialfälle des MIXED-MAX-CUT-PROBLEMs. Ersetze dazu beim MIN-CUT-PROBLEM $w(e)$ durch $-w(e)$.

6.1 Mixed-Max-Cut in planaren Graphen

Wir werden nun einen Algorithmus für das MIXED-MAX-CUT-PROBLEM in planaren Graphen mit Laufzeit $\mathcal{O}(n^{\frac{3}{2}} \log n)$ angeben. Dieser basiert auf der Berechnung eines Matchings in planaren Graphen.

Wie zu erwarten, nutzt der Algorithmus für das MIXED-MAX-CUT-PROBLEM in planaren Graphen die Planarität ganz entscheidend aus, und zwar die Korrespondenz zwischen einem *Schnitt* in dem (eingebetteten) planaren Graphen G und einer *Menge von Kreisen* in dessen Dualgraph G^*.

Aus Lemma 2.9 folgt, dass das MIXED-MAX-CUT-PROBLEM in $G = (V, E)$ äquivalent ist zu dem Problem, im Dualgraph $G^* = (V^*, E^*)$ (bzgl. einer festen Einbettung von G) eine nichtleere Menge von Kanten $S^* \subseteq E^*$ zu finden, die kantendisjunkte Vereinigung von Kreisen ist, und für die $w(S^*)$ maximal ist, wobei $w(e^*) := w(e)$ für e^* Dualkante zu e. Wir benutzen folgenden *Satz von Euler*.

Satz 6.1 (Satz von Euler). *Für einen Graphen $G = (V, E)$ sind äquivalent*

1. *G ist Eulersch.*
2. *E ist kantendisjunkte Vereinigung einfacher Kreise.*
3. *$d(v)$ ist gerade für alle $v \in V$.*

Dabei heißt ein Graph G *Eulersch*, wenn jede Zusammenhangskomponente von G einen so genannten *Euler-Kreis* enthält, d.h. einen Kreis, der jede Kante genau einmal enthält. Zu einem Graphen $G = (V, E)$ heißt eine Menge $E' \subseteq E$ *gerade* genau dann, wenn in dem durch E' induzierten Subgraph von G jeder Knoten geraden Grad hat.

Das MIXED-MAX-CUT-PROBLEM in planaren Graphen ist also äquivalent zum MIXED-MAX-KREIS-PROBLEM.

MIXED-MAX-KREIS-PROBLEM
Gegeben sei ein planarer Graph $G = (V, E)$ mit einer Kantengewichtsfunktion $w : E \longrightarrow K$, wobei $K = \mathbb{R}$. Finde eine nichtleere gerade Menge $E' \subseteq E$ mit $w(E')$ maximal.

Wir werden weiterhin die Äquivalenz des Mixed-Max-Kreis-Problem zu einem perfekten Matching minimalen Gewichts in einem geeignet definierten Graphen benutzen.

Definition 6.2. *Ein Matchings* M *in einem Graphen mit einer geraden Anzahl* n *von Knoten heißt* perfekt *genau dann, wenn* $|M| = \frac{n}{2}$.

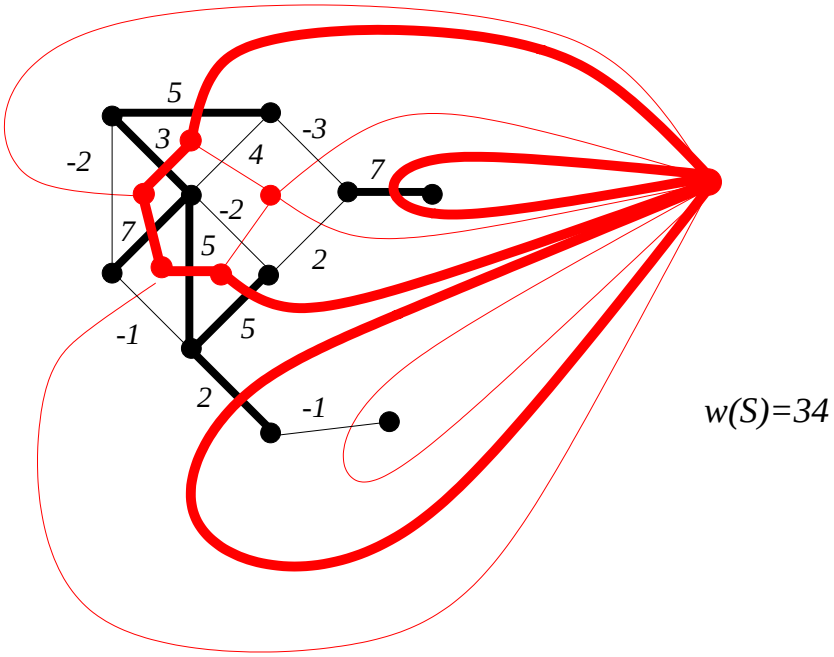


Abbildung 6.1: Illustration der Korrespondenz zwischen Mixed-Max-Cut und Mixed-Max-Kreis.

Der Mixed-Max-Cut-Algorithmus von Shih, Wu & Kuo, 1990

Gegeben sei ein eingebetteter planarer Graph $G = (V, E)$ mit Kantengewichtsfunktion $w : E \longrightarrow \mathbb{R}$.

Schritt 1: Trianguliere G in $\mathcal{O}(n)$ und ordne den hinzugefügten Kanten Gewicht 0 zu.

Schritt 2: Berechne in $\mathcal{O}(n)$ den Dualgraph $G^* = (V^*, E^*)$ zu der Triangulierung von G, wobei $w(e^*) := w(e)$ mit e^* Dualkante zu e. Dann hat in G^* jeder Knoten Grad 3. Eine gerade Menge in G^* ist also eine knotendisjunkte Vereinigung einfacher Kreise in G^*.

Schritt 3: Konstruiere aus G^* in $\mathcal{O}(n)$ einen Graph $G' = (V', E')$ derart, dass ein perfektes Matching minimalen Gewichts in G' eine gerade Menge maximalen Gewichts in G^* induziert.

Schritt 4: Konstruiere in $\mathcal{O}(n^{\frac{3}{2}} \log n)$ ein perfektes Matching M minimalen Gewichts in G'.

Schritt 5: Falls M eine nichtleere gerade Menge in E^* induziert, gib den dazu dualen Schnitt in G aus. Ansonsten berechne in $\mathcal{O}(n^{\frac{3}{2}} \log n)$ aus M eine nichttriviale gerade Menge in G^* maximalen Gewichts.

Ausführung von Schritt 3: Konstruktion von $G' = (V', E')$

G^* ist 3-regulär (d.h., jeder Knoten hat Grad 3). Ersetze jeden Knoten v aus G^* durch einen Graph H_v mit 7 Knoten wie in Abbildung 6.2 und erhalte so $G' = (V', E')$. Die Gewichte der Kanten aus E^* werden dabei auf die entsprechenden Kanten aus E' übertragen und neue Kanten aus E' erhalten Gewicht 0. Wir unterscheiden nicht zwischen den Kanten vom Typ e_1, e_2 und e_3 in G^* und in G'.

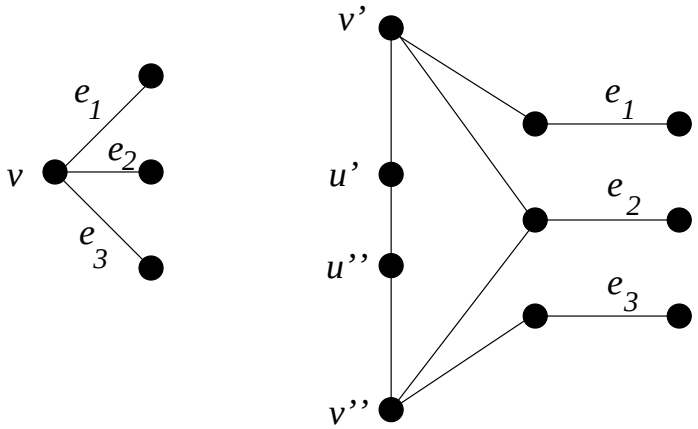


Abbildung 6.2: Ersetzung von v durch H_v.

Beobachtung: Da G^* 3-regulär ist bzw. Dualgraph eines maximal planaren Graphen, ist $|V^*|$ gerade, also auch $|V'|$ gerade. Es existiert also in G' ein perfektes Matching.

Lemma 6.3. *Sei $G' = (V', E')$ entsprechend Abbildung 6.2 aus $G^* = (V^*, E^*)$ konstruierter Graph.*

- *Falls $M \subseteq E'$ ein perfektes Matching in G' ist, so ist die der Menge $E' \backslash M$ entsprechende Menge $M^* \subseteq E^*$ eine gerade Menge in G^*.*
- *Ist andererseits E^*_o eine gerade Menge in G^*, so induziert die der Menge $M^* = E^* \backslash E^*_o$ entsprechende Teilmenge von E' ein perfektes Matching M in G'.*

Beweis. „$\Longrightarrow$“ Sei M ein perfektes Matching in G'. Betrachte für jeden Knoten v in G^* den entsprechenden Subgraphen H_v in G'.

Fall 1: Die Kante $\{u', u''\}$ ist nicht in M. Dann sind die Kanten $\{v', u'\}$ und $\{u'', v''\}$ sowie e_1, e_2, e_3 in M, also e_1, e_2, e_3 nicht in der durch $E' \backslash M$ induzierten Menge $M^* \subseteq E^*$. Also ist $d(v) = 0$ bzgl. M^* und damit M^* gerade Menge. Siehe Abbildung 6.3, links.

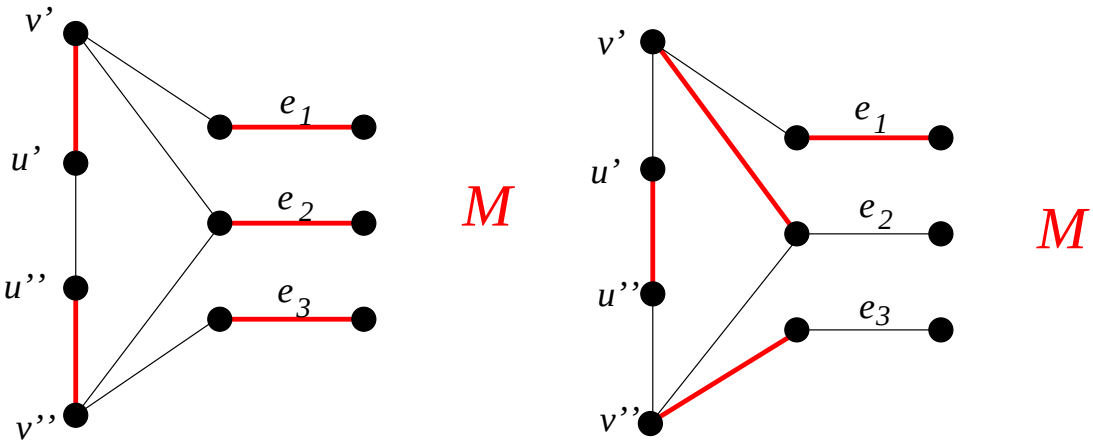


Abbildung 6.3: Illustration von Fall 1 (links) und Fall 2 (rechts).

Fall 2: Die Kante $\{u', u''\}$ ist in M. Dann sind die Kanten $\{v', u'\}$ und $\{u'', v''\}$ nicht in M. Dementsprechend ist jeweils eine der anderen zu v' bzw. v'' inzidenten Kanten in M, sowie genau eine der Kanten e_1, e_2, e_3. Also ist $d(v) = 2$ bzgl. der durch $E' \backslash M$ induzierten Menge $M^* \subseteq E^*$ und damit M^* gerade Menge. Siehe Abbildung 6.3, rechts.

„ $\Longleftarrow$“ Sei E_o^* eine gerade Menge in G^*. Dann haben alle Knoten in dem Subgraphen (V^*, E_o^*) von G^* entweder Grad 0 oder Grad 2.

Fall 1: Der Knoten v habe $d(v) = 0$ bzgl. E_o^*. Dann enthalte M alle drei Kanten e_1, e_2, e_3 und die Kanten $\{v', u'\}$ und $\{v'', u''\}$. Siehe Abbildung 6.3, links.

Fall 2: Der Knoten v habe $d(v) = 2$ bzgl. E_o^*, o.B.d.A. $e_2, e_3 \in E_o^*$. Dann enthalte M die Kante e_1 sowie die Kante $\{u', u''\}$ und die beiden Kanten inzident zu v' und v'', die zu den Kanten e_2 bzw. e_3 adjazent sind. Siehe Abbildung 6.3, rechts.

Dann ist M perfektes Matching in G' und die durch M induzierte Menge $M^* \subseteq E^*$ erfüllt $E^* \backslash M^* = E_o^*$. □

Folgerung 6.4. *Falls $M \subseteq E'$ ein perfektes Matching minimalen Gewichts in G' ist, so ist die der Menge $E' \backslash M$ entsprechende Menge $M^* \subseteq E^*$ eine gerade Menge maximalen Gewichts in G^*. Ist andererseits E_o^* eine gerade Menge maximalen Gewichts in G^*, so induziert die der Menge $M^* = E^* \backslash E_o^*$ entsprechende Teilmenge von E' ein perfektes Matching M minimalen Gewichts in G'.*

Beweis. Es gilt $w(E' \backslash M) = w(E^*) - w(E^* \cap M) = w(E^*) - w(M)$, da alle $e \in M$, mit $e \notin E^*$ Gewicht 0 haben. □

Bemerkung: Die durch M induzierte Menge M^* in G^* kann leer sein! Dazu später.

Ausführung von Schritt 4: Konstruktion eines perfekten Matchings minimalen Gewichts in G'.

Zunächst kann ein perfektes Matching minimalen Gewichts mit einem Algorithmus zur Berechnung eines Matchings maximalen Gewichts folgendermaßen konstruiert werden.

Beobachtung: M ist ein perfektes Matching minimalen Gewichts in einem Graphen $G = (V, E)$ mit Kantengewichten $w : E \longrightarrow \mathbb{R}$, genau dann, wenn M ein perfektes Matching maximalen Gewichts in $G = (V, E)$ mit Kantengewichten $\bar{w} : E \longrightarrow \mathbb{R}$, $\bar{w}(e) := W - w(e)$, wobei W geeignete Konstante.

Wir müssen nun noch die „Perfektheit“ von M bei der Berechnung erzwingen. Wähle dazu W geeignet. Zunächst gilt für ein perfektes Matching M in G, dass

$$\bar{w}(M) = \sum_{e \in M} \bar{w}(e) = \frac{n}{2} \cdot W - \sum_{e \in M} w(e) \geq \frac{n}{2} \cdot (W - w_{\max}),$$

wobei $w_{\max} := \max_{e \in E}(w(e))$. Für ein nicht-perfektes Matching M' gilt andererseits $\bar{w}(M') \leq (\frac{n}{2} - 1) \cdot (W - w_{\min})$, wobei $w_{\min} := \min_{e \in E}(w(e))$. Damit also $\bar{w}(M) > \bar{w}(M')$ gilt für alle perfekten Matchings M und alle nicht-perfekten Matchings M' von G, reicht es aus, W so zu wählen, dass

$$\begin{aligned} & \frac{n}{2}(W - w_{\max}) > (\frac{n}{2} - 1)(W - w_{\min}), \\ \text{also} \quad & W > \frac{n}{2}(w_{\max} - w_{\min}) + w_{\min} \end{aligned}$$

ist. Aus Kapitel 5 kennen wir einen Algorithmus mit Laufzeit $\mathcal{O}(n^{\frac{3}{2}} \log n)$ um in einem planaren Graphen ein Matching maximalen Gewichts, also auch ein perfektes Matching minimalen Gewichts zu bestimmen.

Ausführung von Schritt 5: Konstruktion des Schnitts.

Falls die durch $E' \setminus M$ in E^* induzierte Menge M^* nicht leer ist, gib den entsprechenden dualen Schnitt in G aus.

Wir müssen nun noch den Fall behandeln, dass das berechnete perfekte Matching M minimalen Gewichts in G' die leere Menge in G^* induziert. Dazu berechnen wir in $\mathcal{O}(n^{\frac{3}{2}} \log n)$ aus M eine nichttriviale gerade Menge in G^* maximalen Gewichts. Diese hat dann offensichtlich negatives Gewicht!

Konstruiere zu jedem Knoten $v \in V^*$ aus G' einen Graph G'_v, indem entsprechend Abbildung 6.4 in H_v die Kanten $\{w', u'\}$ und $\{w'', u''\}$ zugefügt werden. Die Kanten $\{w', u'\}$ und $\{w'', u''\}$ erhalten wieder Gewicht 0.

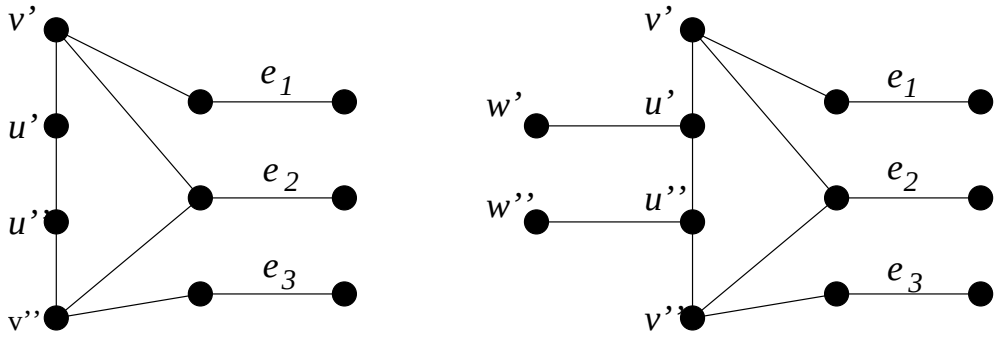


Abbildung 6.4: Konstruktion von G'_v aus G' zu ausgezeichnetem Knoten $v \in V^*$.

Lemma 6.5. *M_v ist ein perfektes Matching minimalen Gewichts in G'_v genau dann, wenn die der Menge $E'_v \setminus M_v$ entsprechende Menge in E^* eine gerade Kantenmenge maximalen Gewichts in G^* ist, die eine zu v inzidente Kante enthält.*

Beweis. Ein perfektes Matching M_v in G'_v enthält immer $\{w', u'\}$ und $\{w'', u''\}$. Ein solches Matching enthält dann genau eine der Kanten e_1, e_2, e_3 zu v. Das Lemma folgt dann analog zu Lemma 6.3 und Folgerung 6.4. □

Ein perfektes Matching M_v minimalen Gewichts in G'_v erhält man in $\mathcal{O}(n \log n)$ aus dem Matching M in G'. Wende dazu zweimal (für w' und w'') den Algorithmus aus Kapitel 5 an. Betrachte nun für alle $v \in V^*$ den Graph G'_v. Eine nichttriviale gerade Menge maximalen Gewichts in G^* wird dann durch die Menge M induziert, für die $w(M) = \min_{v \in V^*} w(M_v)$. Diese Menge M kann „direkt“ in $\mathcal{O}(n^2 \cdot \log n)$ bestimmt werden.

Durch Anwenden des Planar-Separator-Theorems kommt man zu einem effizienteren Algorithmus mit Laufzeit $\mathcal{O}(n^{\frac{3}{2}} \cdot \log n)$ wie folgt.

Vorüberlegung: Wenn für das in Schritt 4 berechnete perfekte Matching M minimalen Gewichts in G' gilt, dass die durch $E' \setminus M$ induzierte Menge in G^* leer ist, so müssen alle Kreise in G^* negatives Gewicht haben. Die gesuchte nichttriviale gerade Menge in G^* besteht also aus einem einfachen Kreis negativen Gewichts, dessen Gewicht maximal ist unter allen Kreisen in G^*.

Lemma 6.6. *In einem 3-regulären planaren Graphen G, der keinen positiven Kreis enthält, kann ein negativer einfacher Kreis maximalen Gewichts in $\mathcal{O}(n^{\frac{3}{2}} \log n)$ bestimmt werden.*

Beweis. Wende folgenden Algorithmus an.

Schritt 1: Berechne eine Partition S, V_1, V_2 in G, die die Bedingungen des Planar-Separator-Theorems erfüllt.

Schritt 2: Berechne rekursiv negative einfache Kreise maximalen Gewichts in den durch V_1 und V_2 induzierten Subgraphen von G. Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält, wie folgt:

Konstruiere zu G den Graphen G' gemäß Schritt 3 des Mixed-Max-Cut-Algorithmus. Für jeden Knoten $v_i \in S$ erweitere G' zu G'_{v_i}, indem v_i durch den durch $\{u', u'', v', v'', w', w''\}$ induzierten Subgraphen (vgl. Abb. 6.4, rechts) ersetzt wird (jeder dieser Graphen enthält also die Knoten w' und w'' genau einmal), und bestimme in $\mathcal{O}(n \log n)$ ein perfektes Matching minimalen Gewichts von G'_{v_i}; berechne den dazu korrespondierenden negativen einfachen Kreis maximalen Gewichts in G.

Schritt 3: Gib den Kreis maximalen Gewichts unter allen konstruierten Kreisen aus. Dieser ist der gewünschte Kreis in G, da jeder einfache Kreis in G entweder ganz in G_1 oder ganz in G_2 liegt oder (mindestens) ein $v_i \in S$ enthält.

Die Gesamtlaufzeit $t(n)$ ist gegeben durch

$$t(n_o) = c_o$$
$$t(n) = t(c_1 \cdot n) + t(c_2 \cdot n) + c_3 \cdot \sqrt{n} \cdot n \cdot \log n,$$

wobei c_0, c_1, c_2, c_3 konstant, $c_1,\ c_2 \leq \frac{2}{3}$ und $c_1 + c_2 < 1$. Damit ist $t(n) \in \mathcal{O}(n^{\frac{3}{2}} \log n)$.□

Damit haben wir insgesamt folgenden Satz bewiesen.

Satz 6.7. *In einem planaren Graphen $G = (V, E)$ mit Kantengewichtsfunktion $w : E \longrightarrow \mathbb{R}$ kann in $\mathcal{O}(n^{\frac{3}{2}} \log n)$ ein Schnitt $S \subseteq E$ konstruiert werden, mit $w(S)$ maximal.*

6.2 Das Via-Minimierungs-Problem

Eine von vielen interessanten Anwendungen des Mixed-Max-Cut-Problems tritt beim Entwurf hochintegrierter Schaltungen auf. Man möchte eine Schaltung möglichst kostengünstig auf einem Chip realisieren. Ein Schritt in dem entsprechenden Entwurfsprozess besteht darin, ein Layout der Schaltung so innerhalb mehrerer Lagen zu realisieren, dass die Anzahl der Lagenwechsel klein ist. Als Basis der Realisierung einer Schaltung wird üblicherweise ein orthogonales Gitter angenommen.

Eine *Schaltung* bestehe aus *Modulen*, auf deren Rändern *Terminale* liegen und *Drähten*, die jeweils vorgegebene Terminale verbinden. Ein *Layout* L ist dann eine Einbettung der Schaltung in ein orthogonales Gitter, bei der Drähte als kantendisjunkte Verbindungen (im allgemeinen *Steiner-Bäume*) entlang Gitterlinien geführt werden. Wir werden uns hier auf den Fall beschränken, dass jeder Draht genau zwei Terminale verbindet, d.h. die Drähte als kantendisjunkte *Wege* eingebettet werden können.

Jede *Lage* ist dann eine Kopie des orthogonalen Gitters, und eine zulässige *Lagenzuweisung* besteht in einer knotendisjunkten Zuordnung der eingebetteten Drahtstücke zu Lagen, d.h. keine zwei Drähte berühren sich in derselben Lage. Lagenwechsel, so genannte *Vias*, sind nur an Gitterpunkten erlaubt. Wenn es zu einem Layout eine zulässige Lagenzuweisung in zwei Lagen gibt, so nennt man das Layout auch *in zwei Lagen realisierbar*. Siehe Abbildungen 6.5 und 6.6.

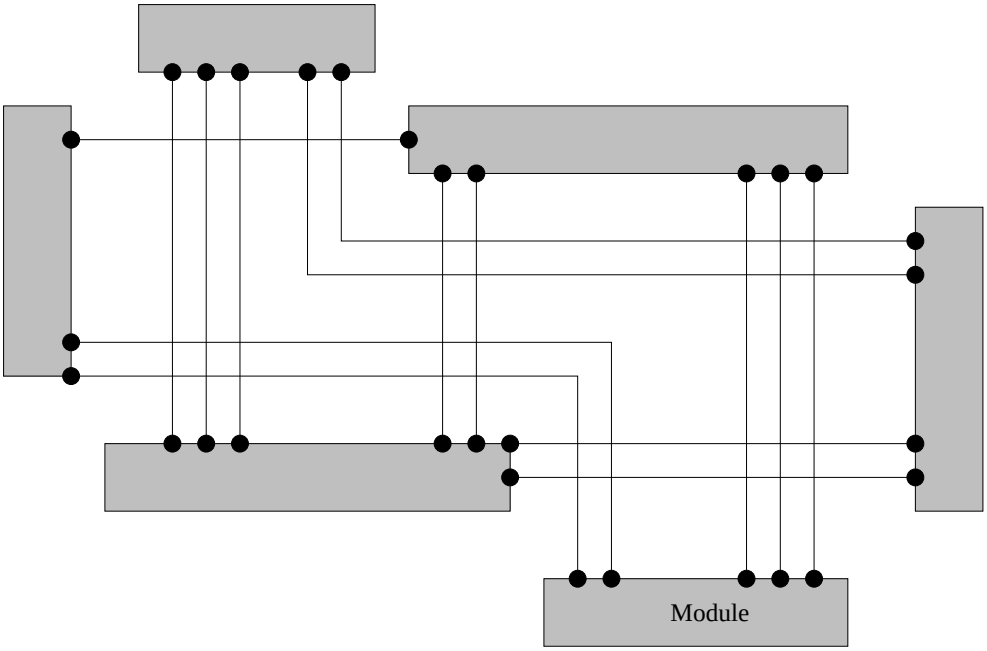


Abbildung 6.5: Ein in zwei Lagen realisierbares Layout.

VIA-MIMIMIERUNGS-PROBLEM
Gegeben sei ein in zwei Lagen realisierbares Layout L. Finde eine Realisierung von L in zwei Lagen mit minimaler Anzahl an Vias.

Bemerkung: Ein Layout, bei dem sich verschiedene Drähte in Gitterpunkten kreuzen dürfen, aber nicht an dem selben Gitterpunkt gegeneinander abknicken dürfen, können immer leicht in zwei Lagen realisiert werden. Dazu ordnet man einfach alle vertikalen Drahtstücke der einen und alle horizontalen Drahtstücke der anderen Lage zu. Solche Layouts werden *Manhattan-Layout* genannt. Eine solche Realisierung in zwei Lagen für das Layout aus Abbildung 6.5 würde vier Vias benötigen. In Abbildung 6.6 ist eine Realisierung dieses Layouts in zwei Lagen mit 3 Vias gezeigt. Ist für das Layout in Abbildung 6.5 eine Realisierung in zwei Lagen mit weniger als 3 Vias möglich?

Vorüberlegung: Eine Realisierung eines Layouts in zwei Lagen entspricht einer Zweifärbung der Drahtstücke. Die Anzahl der Vias entspricht der Gesamtzahl der Farbwechsel von Drähten. Entsprechend werden wir das VIA-MIMIMIERUNGS-PROBLEM als ein Graphenfärbungsproblem modellieren. Zunächst müssen wir die entscheidenden Charakteri-

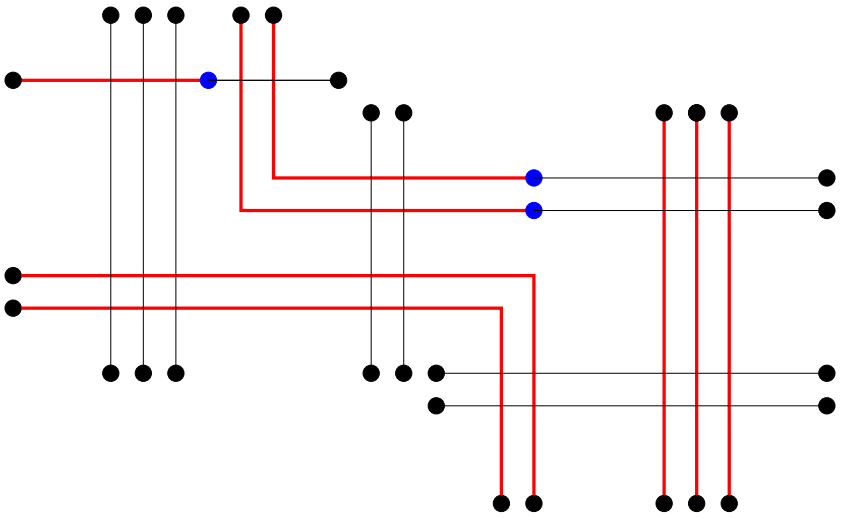


Abbildung 6.6: Realisierung des Layouts aus Abbildung 6.5 in zwei Lagen mit 3 Vias.

stika des Problems, die einerseits in den *Konflikten* zwischen Drahtstücken und andererseits in der geeigneten Wahl von *Lagenwechseln durch Vias* bestehen, durch einen Graph ausdrücken. Dazu definieren wir den *Konfliktgraph* zu einem Layout.

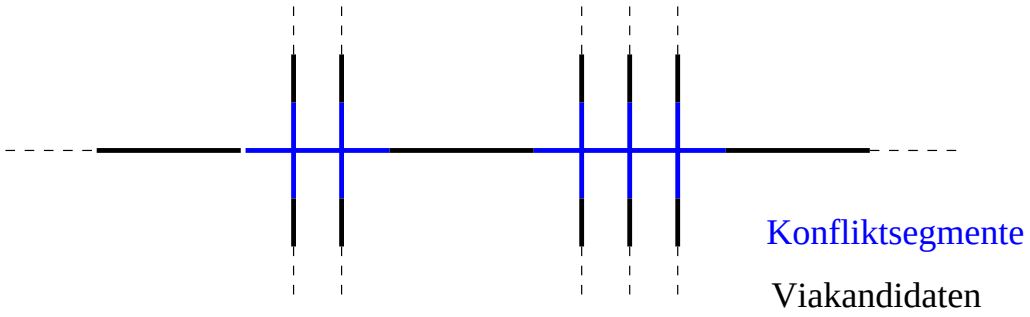


Abbildung 6.7: Ausschnitt eines Layouts und dessen Aufteilung in *Konflikt-Segmente* und *Via-Kandidaten*.

Die Drähte eines Layouts können in zwei Typen von Drahtstücken aufgeteilt werden, in *Konflikt-Segmente* und in *Via-Kandidaten*. Siehe Abbildung 6.7.

- *Konflikt-Segmente* sind Drahtstücke, die in allen Gitterpunkten andere Drähte berühren.
- Die restlichen Drahtstücke sind *Via-Kandidaten*, d.h. maximale Drahtstücke, die über mindestens einen Gitterpunkt gehen, über den kein anderer Draht geht. Dies sind gerade die Drahtstücke, auf denen Vias plaziert werden können.

Zu einem in zwei Lagen realisierbaren Layout L definiere den *Konfliktgraph* $\mathsf{G_c}(\mathsf{L}) = (\mathsf{V_c}, \mathsf{E_c})$ wie folgt.

- $\mathsf{V_c}$ entspricht der Menge aller Konflikt-Segmente

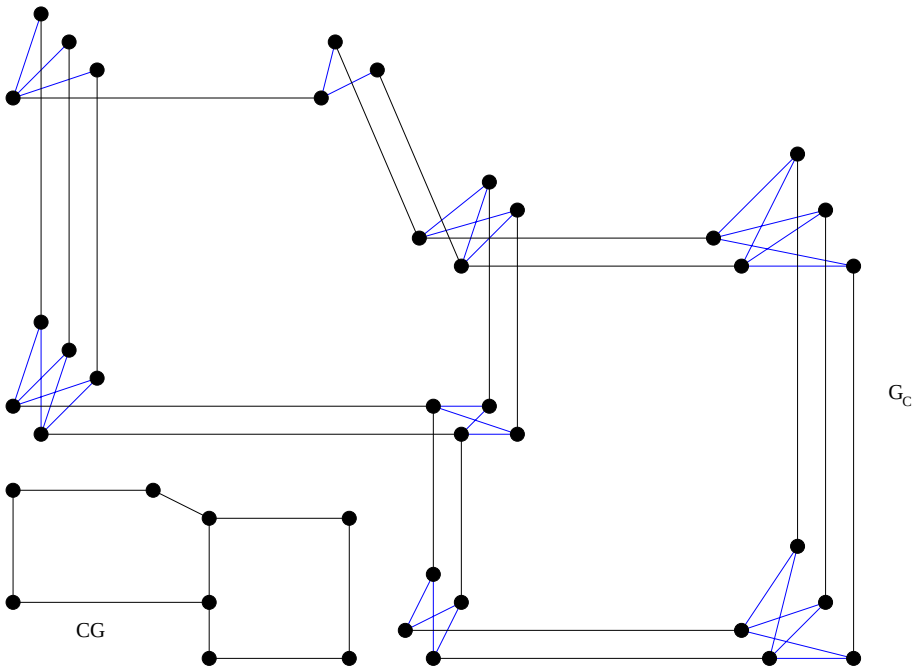


Abbildung 6.8: Der Konfliktgraph zum Layout aus Abbildung 6.5 und der zugehörige Clustergraph.

- E_c enthält zwei Typen von Kanten
  - $\{u, v\} \in E_c$ für Knoten $u, v \in V_c$, welche Konflikt-Segmenten entsprechen, die sich in einem Gitterpunkt berühren, genannt *Konfliktkanten*
  - $\{u, v\} \in E_c$ für Knoten $u, v \in V_c$, welche Konflikt-Segmenten entsprechen, die inzident zu dem selben Via-Kandidaten sind.

Die Subgraphen von G_c, die durch Konfliktkanten induziert sind, d.h. Mengen von Konfliktsegmenten, die sich gegenseitig berühren, können zu *Konfliktclustern* zusammengefasst werden. Dadurch wird ein bewerteter *Clustergraph* $CG := (CV, CE)$ induziert, mit Kantenbewertung $c : CE \to \mathbb{N}$.

- CV entspricht der Menge der Konfliktcluster,
- $\{a, b\} \in CE$, falls in dem Konfliktcluster zu a und dem Konfliktcluster zu b Konfliktsegmente existieren, die durch denselben Via-Kandidaten verbunden sind,
- $c(e) :=$ # der Via-Kandidaten, die e entsprechen für $e \in CE$.

Abbildung 6.8 zeigt den Konfliktgraph und zugehörigen Clustergraph zum Layout aus Abbildung 6.5. Offensichtlich ist CG immer planar.

In einer Realisierung von L in zwei Lagen ist die Lage aller Konfliktsegmente eines Konflikt-Clusters durch die Lage eines einzigen Konfliktsegments dieses Clusters festgelegt. Man kann also einen beliebigen Knoten eines jeden Konflikt-Cluster als *Repräsentanten* der Lagenzuweisung wählen.

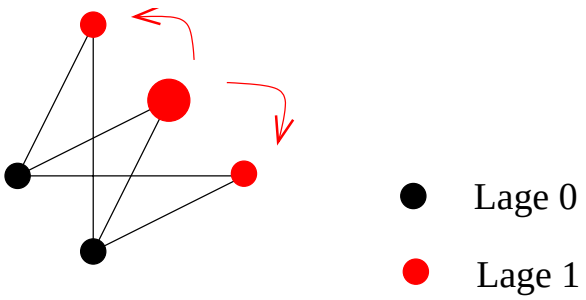


Abbildung 6.9: Festlegung der Lagen aller Konfliktsegmente eines Konflikt-Clusters durch Festlegung des Repräsentanten.

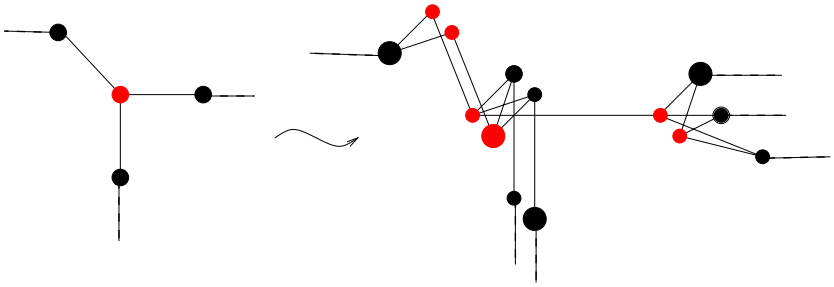


Abbildung 6.10: Festlegung der Lagen entsprechend einer Zweifärbung im Konfliktgraph.

Eine Realisierung von L in zwei Lagen entspricht zunächst einer Zweifärbung der Knoten aus G_c, so dass Knoten, die durch eine Konfliktkante verbunden sind, verschiedene Farben haben. Dies entspricht einer Färbung der Knoten von CG mit zwei Farben, wobei adjazente Knoten die gleiche Farbe haben können. Eine „echte“ Zweifärbung würde dann einer Realisierung von L in zwei Lagen entsprechen ohne Vias, bei entsprechender Wahl der Repräsentanten. Eine „echte“ Zweifärbung eines Graphen existiert genau dann, wenn der Graph keine Kreise ungerader Länge enthält. Solche Graphen heißen *bipartit*. Es ist also leicht zu entscheiden, ob ein Layout ohne Via in zwei Lagen realisierbar ist.

Im Allgemeinen entspricht — bei entsprechender Wahl der Repräsentanten — eine Realisierung von L in zwei Lagen *mit minimaler Anzahl an Vias* einer Zweifärbung der Knoten von CG, bei der das *Gesamtgewicht der Kanten, deren Endknoten dieselbe Farbe haben, minimal* ist. Wenn wir, beginnend bei einem beliebigen Knoten von CG, entlang Wegen abwechselnd mit zwei Farben färben, so erhalten wir auf ungeraden Kreisen Kanten, deren Endknoten dieselbe Farbe haben. Via-Minimierung ist also äquivalent dazu, eine Menge von Kanten mit minimalem Gesamtgewicht zu finden, nach deren Entfernen CG bipartit ist:

Minimum Edge Deletion Bipartization
Gegeben $G = (V, E), c : E \to \mathbb{Z}$. Finde $E' \subseteq E$ so, dass $G' := (V, E \setminus E')$

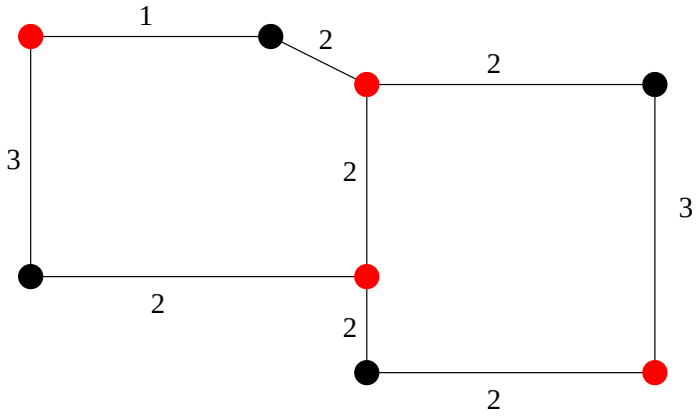


Abbildung 6.11: Zweifärbung des Cluster-Graph.

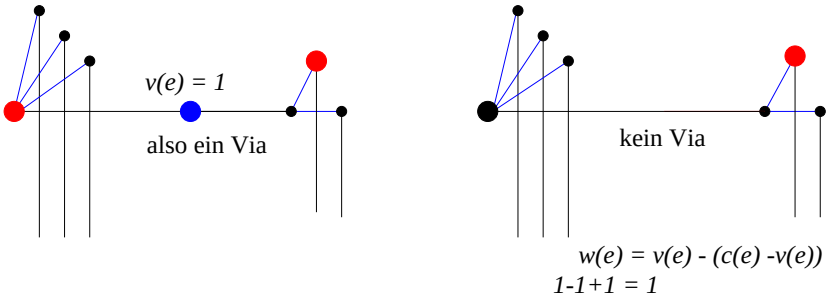


Abbildung 6.12: Reduktion um ein Via bzgl. ν.

bipartit ist und $c(E') := \sum_{e \in E'} c(e)$ minimal.

Dies ist äquivalent zu

Gegeben $G = (V, E), c : E \to \mathbb{Z}$. Finde $E^* \subseteq E$ so, dass $G^* := (V, E^*)$ bipartit ist und $c(E^*) := \sum_{e \in E^*} c(e)$ maximal.

Dies ist aber gerade das *Mixed-Max-Cut-Problem*: der bipartite Graph G^* induziert eine Partition von V in $V_1 und V_2$ so, dass das Gesamtgewicht der Kanten von G, deren Endknoten in verschiedenen Mengen der Partition liegen, maximal ist.

Zurück zum Via-Minimierungsproblem.

Wahl der Repräsentanten

Wähle in G_C einen beliebigen Knoten eines Clusters als Repräsentanten dieses Clusters. Gehe entlang Wegen durch G, bei denen sich Konfliktkanten und Viakanten abwechseln, und wähle jeweils jeden zweiten Knoten als Repräsentanten seines Clusters.

Bemerkung: Dies geht, da solche Kreise in G_C gerade Länge haben.

Färbung und dadurch induzierte Kantengewichte

Färbe alle Knoten aus CG mit derselben Farbe. Die dadurch induzierte Anzahl an Vias entspricht gerade der Gewichtsfunktion $c : CE \to \mathbb{Z}_0^+$ mit $c(e) :=$ # Viakandidaten, die e entsprechen. Max-Cut in CG bzgl. c entspricht einer Realisierung von L in zwei Lagen, bei der gerade auf den Viasegmenten, die Schnittkanten entsprechen, *kein Via* platziert wird.

Bemerkung:

- Wir haben bereits festgestellt, dass CG immer planar ist.
- Kantengewichte sind hier alle nicht-negativ.

Bei einer beliebigen Wahl von Repräsentanten und einer beliebigen Färbung mit zwei Farben erhält man eine Realisierung in zwei Lagen, die eine Kantengewichtsfunktion $\nu_{red} : CE \to \mathbb{Z}$ wie folgt induziert:

Sei $\nu : CE \to \mathbb{Z}^+$ eine Funktion, die zu dieser Realisierung die Anzahl der Vias pro Kante angibt, d. h.

$$\nu(e) = \begin{cases} c(e) & \text{falls Endknoten der durch } e \text{ in } G_C \text{ induzierten Viakanten in verschiedenen Lagen liegen} \\ 0 & \text{sonst} \end{cases}.$$

Dann soll ν_{red} für jede Kante $e \in CE$ angeben, wie groß die ‚Via-Reduktion' ist, wenn bei einem Endknoten von e die Farbe getauscht, d. h. im entsprechenden Cluster in G_C die Lagen vertauscht werden. Es ist offensichtlich

$$\nu_{red}(e) = \begin{cases} c(e) & \text{falls } \nu(e) = c(e) \\ -c(e) & \text{falls } \nu(e) = 0 \end{cases},$$

also $\nu_{red}(e) = \nu(e) - (c(e) - \nu(e))$.

Startend bei einer beliebigen Wahl der Repräsentanten und einer beliebigen Färbung, besteht unser Problem also in der maximalen Via-Reduktion, also der Wahl einer Knotenmenge $X \subseteq CV$ so, dass der zugehörige Schnitt maximales Gewicht hat, d. h.

$$\nu_{red}(X) := \sum_{\substack{e=\{x,y\}\in E \\ x\in X, y\notin X}} \nu_{red}(e) \text{ maximal.}$$

Beachte: ν_{red} kann auch negative Werte haben.

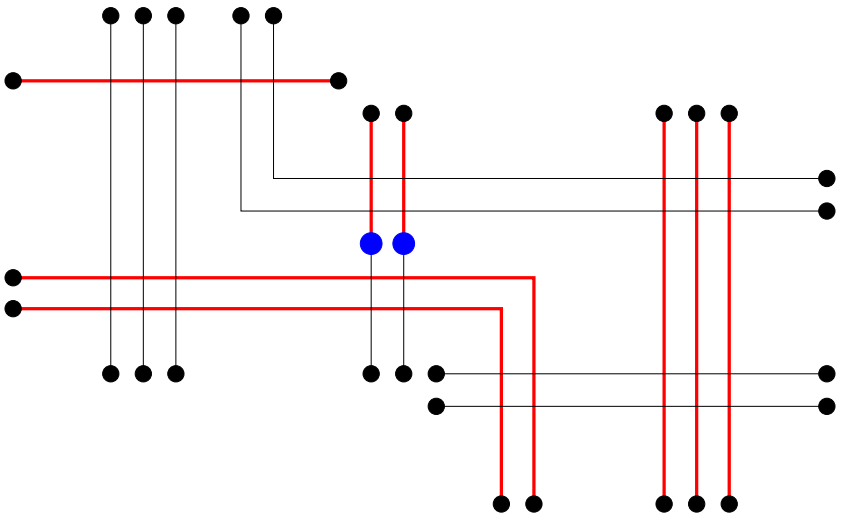


Abbildung 6.13: Realisierung des Layouts aus Abbildung 6.5 in zwei Lagen mit nur 2 Vias.

7 Das Menger-Problem

Das Menger-Problem ist ein „Kernproblem“ vieler Anwendungsprobleme, etwa aus dem Bereich „Verkehrsplanung“, „Schaltkreisentwurf“ oder „Kommunikationsnetzwerke“. Wir werden Linearzeitalgorithmen zur Lösung des kantendisjunkten bzw. knotendisjunkten Menger-Problems in planaren Graphen angeben. Beide Algorithmen beruhen im wesentlichen auf einer Right-First-Tiefensuche.

Eine Menge $S \subset V$ heißt *Separator* von $G = (V, E)$, falls der durch $V \backslash S$ induzierte Subgraph von G unzusammenhängend ist. S trennt die Knoten $u, v \in V \backslash S$, falls u und v in dem durch $V \backslash S$ induzierten Subgraph (bezeichnet mit $G - S$) in verschiedenen Zusammenhangskomponenten liegen. Siehe Abb. 2.3. Wir definieren den *Knotenzusammenhang* $\kappa_G(u, v)$ zweier Knoten u und v bzw. den *Knotenzusammenhang* $\kappa(G)$ des Graphen G wie folgt.

$$\kappa_G(u,v) := \begin{cases} |V| - 1, \text{ falls } \{u,v\} \in E \\ \min\limits_{\substack{S \subset V, \\ S \text{ trennt } u \text{ und } v}} |S|, \text{ sonst.} \end{cases}$$

$$\kappa(G) := \min_{\substack{S \subset V, \\ S \text{ Separator von } G}} \{|S|, |V| - 1\} = \min_{u,v \in V} \kappa_G(u,v)$$

Eine Menge $S \subseteq E$ heißt *Schnitt* von $G = (V, E)$, falls der durch $E \backslash S$ induzierte Subgraph von G unzusammenhängend ist, d.h. in Graphen $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ zerfällt, mit $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, $E_1 \cup E_2 = E \backslash S$, $E_1 \cap E_2 = \emptyset$, wobei alle Kanten aus S einen Endknoten in V_1 und einen Endknoten in V_2 haben. S *trennt* die Knoten $u, v \in V$, falls u und v in dem durch $E \backslash S$ induzierten Subgraph (bezeichnet mit $G - S$) in verschiedenen Zusammenhangskomponenten liegen. Entsprechend definieren wir den *Kantenzusammenhang* $\lambda_G(u, v)$ zweier Knoten u und v, bzw. den *Kantenzusammenhang* $\lambda(G)$ des Graphen G wie folgt.

$$\lambda_G(u,v) := \min_{\substack{S \subseteq E, \\ S \text{ trennt } u \text{ und } v}} |S|$$

$$\lambda(G) := \min_{\substack{S \subseteq E, \\ S \text{ Schnitt von } G}} |S| = \min_{u,v \in V} \lambda_G(u,v)$$

G heißt *k-fach knoten- bzw. kantenzusammenhängend*, falls $k \leq \kappa(G)$ bzw. $k \leq \lambda(G)$. Zwei Wege in einem Graphen G heißen *(intern) knotendisjunkt*, wenn sie (außer den Endknoten) keine gemeinsamen Knoten enthalten und *kantendisjunkt*, wenn sie keine gemeinsame Kante enthalten.

Satz 7.1. Satz von Menger (1927)
Seien s *und* t *zwei Knoten eines Graphen* G*,* s *und* t *nicht adjazent bei der knotendisjunkten Version.*

- $\kappa_G(s,t) \geq k$ *genau dann, wenn es* k *paarweise intern knotendisjunkte Wege zwischen* s *und* t *in* G *gibt.*
- $\lambda_G(s,t) \geq k$ *genau dann, wenn es* k *paarweise kantendisjunkte Wege zwischen* s *und* t *in* G *gibt.*

Menger-Problem
Gegeben sei ein Graph $G = (V, E)$ und Knoten s, $t \in V$. Finde eine maximale Anzahl paarweise kanten- bzw. intern knotendisjunkter Wege, die s und t verbinden.

7.1 Das kantendisjunkte Menger-Problem in planaren Graphen

Der im folgenden ausgeführte Algorithmus zur Lösung des kantendisjunkten Menger-Problems in planaren Graphen wurde 1994 von K. Weihe veröffentlicht. Eine Variante des Algorithmus wurde 1997 von Coupry vorgestellt. Beide Algorithmen haben lineare Laufzeit. Betrachte im folgenden einen planaren Graphen $G = (V, E)$ mit einer planaren Einbettung, bei der t auf der äußeren Facette liegt.

Kantendisjunkter Menger-Algorithmus

Schritt 1: Ersetze in Linearzeit $G = (V, E)$ durch den *gerichteten Graphen* $\overrightarrow{G} = (V, \overrightarrow{E})$, der entsteht, indem jede Kante $\{u, v\} \in E$ ersetzt wird durch die beiden *gerichteten Kanten* (u, v), $(v, u) \in \overrightarrow{E}$.

Schritt 2: Berechne in Linearzeit eine Menge geeigneter einfacher kantendisjunkter gerichteter Kreise $\overrightarrow{C_1}, \ldots, \overrightarrow{C_l}$, und betrachte den Graph $\overrightarrow{G_C}$, der aus $\overrightarrow{G}$ entsteht, indem alle Kanten, die auf einem $\overrightarrow{C_i}$ liegen, umgedreht werden.

Schritt 3: Berechne in Linearzeit eine maximale Anzahl von kantendisjunkten gerichteten s-t-Wegen in $\overrightarrow{G_C}$.

Schritt 4: Berechne in Linearzeit aus der Menge der kantendisjunkten s-t-Wege in $\overrightarrow{G_C}$ eine Menge kantendisjunkter s-t-Wege in G gleicher Kardinalität.

Ausführung von Schritt 1: Konstruktion von $\overrightarrow{G} = (V, \overrightarrow{E})$

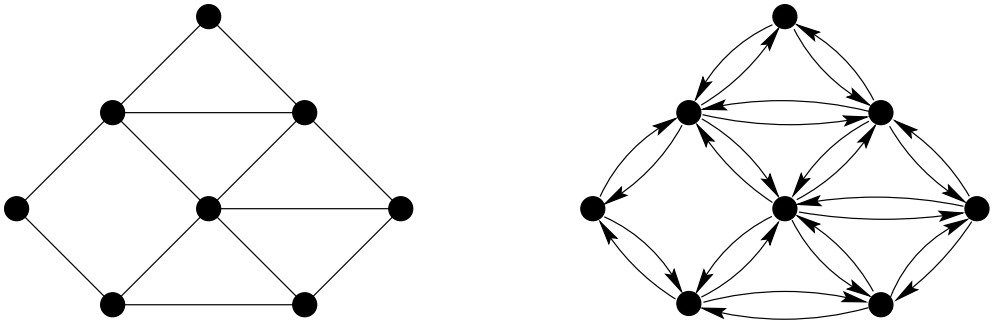


Abbildung 7.1: Ersetzung von $G = (V, E)$ durch $\overrightarrow{G} = (V, \overrightarrow{E})$.

Ersetze $G = (V, E)$ durch den *gerichteten Graphen* $\overrightarrow{G} = (V, \overrightarrow{E})$, der entsteht, indem jede Kante $\{u, v\} \in E$ ersetzt wird durch die beiden *gerichteten Kanten* (u, v), $(v, u) \in \overrightarrow{E}$. Die Begriffe Weg, s-t-Weg und Kreis werden kanonisch übertragen zu den Begriffen *gerichteter Weg, gerichteter* s-t-*Weg* und *gerichteter Kreis*.

Lemma 7.2. *Seien* $p_1, \ldots, p_r$ *kantendisjunkte, gerichtete* s-t-*Wege in* $\overrightarrow{G}$*. Dann enthält die Menge* $P \subseteq E$, $P := \{\{u, v\}$*: genau eine der beiden gerichteten Kanten* (u, v) *oder* (v, u) *gehört zu einem der Wege* $p_i, 1 \leq i \leq r\}$ *gerade* r *kantendisjunkte Wege* s-t-*Wege in* G*.*

Beweis.

1. Wenn p_i beide Kanten (u, v) und (v, u) enthält, so ist p_i nicht einfach. Es gibt zu p_i einen einfachen gerichteten s-t-Weg, der weder (u, v) noch (v, u) enthält.
2. Seien nun p_i, p_j gerichtete s-t-Wege, p_i enthalte die Kante (u, v) und p_j enthalte die Kante (v, u). Dann gibt es zwei gerichtete kantendisjunkte s-t-Wege, die alle Kanten aus p_i und p_j enthalten, außer (u, v) und (v, u).

Damit folgt die Behauptung (siehe Abb. 7.2). □

Folgerung 7.3. *Eine maximale Anzahl kantendisjunkter gerichteter* s-t-*Wege* $p_1, \ldots, p_k$ *in* $\overrightarrow{G}$ *induziert eine maximale Anzahl kantendisjunkter* s-t-*Wege in* G*. Diese können offensichtlich in Linearzeit aus* $p_1, \ldots, p_k$ *konstruiert werden.*

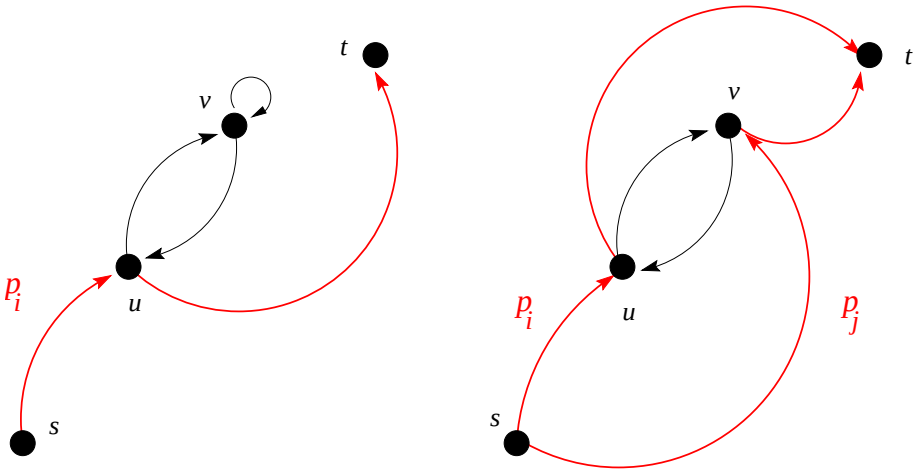


Abbildung 7.2: Illustration von Lemma 7.2

Ausführung von Schritt 2: Konstruktion von $\overrightarrow{G_C}$

Berechne in Linearzeit eine Menge von einfachen kantendisjunkten gerichteten Kreisen $\overrightarrow{C_1}, \dots, \overrightarrow{C_l}$, so dass der Graph $\overrightarrow{G_C}$, der aus $\overrightarrow{G}$ entsteht, durch Ersetzen aller Kanten (u, v), die auf einem der $\overrightarrow{C_i}$ liegen, durch die Kante $(v, u)'$ folgende Eigenschaften hat.

Eigenschaft 1 $\overrightarrow{G_C}$ enthält keinen Rechtskreis, d.h. keinen einfach gerichteten Kreis, dessen „Inneres“ ($\widehat{=}$ Menge der vom Kreis umschlossenen Facetten, die nicht die äußere Facette enthält) rechts vom Kreis liegt.

Eigenschaft 2 Bilde $\overrightarrow{P_C} \subseteq \overrightarrow{E_C}$ Menge von kantendisjunkten gerichteten s-t-Wegen in $\overrightarrow{G_C}$ und $\overrightarrow{P} \subseteq \overrightarrow{E}$ sei definiert als

$$\begin{aligned}\overrightarrow{P} := (&\overrightarrow{P_C} \cap \overrightarrow{E})\\ &\cup \{(u,v) \in \overrightarrow{E} : (u,v) \text{ auf einem der } \overrightarrow{C_i} \text{ und } (v,u)' \notin \overrightarrow{P_C}\}.\end{aligned}$$

Dann soll $\overrightarrow{P}$ genau dann eine maximale Menge kantendisjunkter gerichteter s-t-Wege in $\overrightarrow{G}$ sein, wenn $\overrightarrow{P_C}$ eine maximale Menge kantendisjunkter gerichteter s-t-Wege in $\overrightarrow{G_C}$ ist.

Bemerkung.

1. $\overrightarrow{G_C}$ *kann doppelte Kanten* (v, u) *und* $(v, u)'$ *enthalten.*
2. $\overrightarrow{P}$ *entsteht aus* $\overrightarrow{P_C}$*, indem genau die Kanten „herausgenommen“ werden, die in* $\overrightarrow{P_C}$ *liegen und „umgedreht“ wurden, und die Kanten „hinzugenommen“ werden, die nicht in* $\overrightarrow{P_C}$ *liegen und „umgedreht“ wurden.*

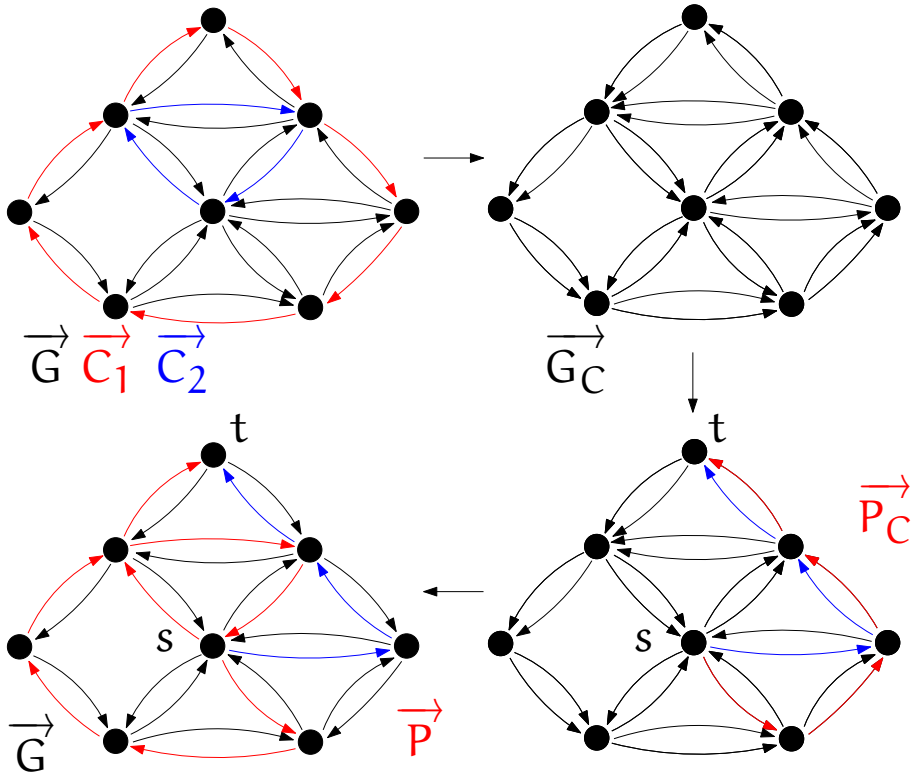


Abbildung 7.3: Illustration von Schritt 2.

Konstruktion der $\overrightarrow{C_1}, \ldots, \overrightarrow{C_l}$

Sei F Menge der Facetten von G. Definiere den *Abstand* einer Facette $f \in F$ von der äußeren Facette f_0, d.h.

$$\text{dist}(f) := \text{Länge eines kürzesten Weges von dem Dualknoten zu } f \text{ zum Dualknoten der äußeren Facette } f_0 \text{ in } G^*.$$

Sei $l := \max_{f \in F} \text{dist}(f)$ und für $1 \leq i \leq l$ sei C_i Vereinigung der einfachen Kreise c_i in G, für die alle Facetten $f \in \text{Inneres}(c_i)$ erfüllen $\text{dist}(f) \geq i$, und alle Facetten $f \in \text{Äußeres}(c_i)$ erfüllen $\text{dist}(f) < i$. Dann seien $\overrightarrow{C_1}, \ldots, \overrightarrow{C_l}$ die $C_1, \ldots, C_l$ entsprechenden Rechtskreise in $\overrightarrow{G}$.

Wir beweisen, dass der durch Umkehren der Kreise $\overrightarrow{C_1}, \ldots, \overrightarrow{C_l}$ in $\overrightarrow{G}$ induzierte Graph $\overrightarrow{G_C}$ die gewünschten Eigenschaften hat.

Zu Eigenschaft 1: Offensichtlich enthält $\overrightarrow{G_C}$ keine Rechtskreise, da von jedem Rechtskreis aus $\overrightarrow{G}$ mindestens eine Kante auf einem der $\overrightarrow{C_i}$ liegen muss.

Zu Eigenschaft 2: Wir benutzen folgendes Lemma über kantendisjunkte s-t-Wege in gerichteten Graphen.

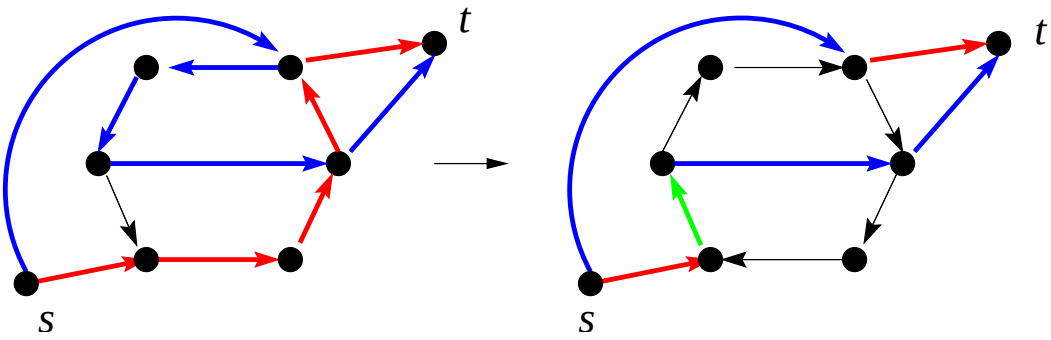


Abbildung 7.4: Illustration von Eigenschaft 2.

Lemma 7.4. *Sei* $\overrightarrow{H} = (V(\overrightarrow{H}), E(\overrightarrow{H}))$ *ein gerichteter zusammenhängender Graph,* $s, t \in V(\overrightarrow{H})$. *Also besteht* $\overrightarrow{H}$ *aus genau* k *kantendisjunkten gerichteten* s-t-*Wegen genau dann, wenn gilt:*

(1) Für alle $v \in V(\overrightarrow{H})\backslash\{s,t\}$ *ist* $d^{\rightarrow}(v) = d^{\leftarrow}(v)$,

wobei $d^{\rightarrow}(v)$:= # *Kanten, die* v *verlassen, und*
$d^{\leftarrow}(v)$:= # *Kanten, die in* v *hereinführen*

(2) $k = d^{\rightarrow}(s) - d^{\leftarrow}(s) = d^{\leftarrow}(t) - d^{\rightarrow}(t)$

Beweis. „$\Longrightarrow$“ Da jeder s-t-Weg für jeden Knoten v dieselbe Anzahl Kanten zu $d^{\rightarrow}(v)$ wie zu $d^{\leftarrow}(v)$ beiträgt, gilt (1). Jeder s-t-Weg trägt genau eine Kante zu $d^{\rightarrow}(s) - d^{\leftarrow}(s)$, und genau eine Kante zu $d^{\leftarrow}(t) - d^{\rightarrow}(t)$. Also gilt auch (2).

„$\Longleftarrow$“ Wenn jeder Knoten aus $V(\overrightarrow{H})\backslash\{s,t\}$ Bedingung 1 erfüllt, und $d^{\rightarrow}(s) - d^{\leftarrow}(s) = d^{\leftarrow}(t) - d^{\rightarrow}(t)$, so gibt es dabei gerade k s-t-Wege. Damit besteht $\overrightarrow{H}$ also aus k nicht notwendig einfachen, kantendisjunkten s-t-Wegen. □

Sei nun $\overrightarrow{P_C}$ Menge kantendisjunkter gerichteter s-t-Wege in $\overrightarrow{G_C}$. Beim Übergang von $\overrightarrow{P_C}$ zu $\overrightarrow{P}$ ändern sich $d^{\rightarrow}(v)$ und $d^{\leftarrow}(v)$ für jeden Knoten $v \in V$ um den gleichen Betrag. Nur für v auf einem C_i ändert sich $d^{\rightarrow}(v)$ bzw. $d^{\leftarrow}(v)$ und zwar für (v,u), (w,v) aus C_i.

$$(u,v)', (v,w)' \in \overrightarrow{P_C} \quad \text{g.d.w.} \quad (v,u), (w,v) \notin \overrightarrow{P} \tag{1}$$
$$(u,v)' \in \overrightarrow{P_C}, (v,w)' \notin \overrightarrow{P_C} \quad \text{g.d.w.} \quad (v,u) \notin \overrightarrow{P}, (w,v) \in \overrightarrow{P} \text{ bzw.} \tag{2}$$
$$(u,v)' \notin \overrightarrow{P_C}, (v,w)' \in \overrightarrow{P_C} \quad \text{g.d.w.} \quad (v,u) \in \overrightarrow{P}, (w,v) \notin \overrightarrow{P}$$
$$(u,v)', (v,w)' \notin \overrightarrow{P_C} \quad \text{g.d.w.} \quad (v,u)\,(w,v) \in \overrightarrow{P} \tag{3}$$

In $\overrightarrow{P_C}$ gilt

$$k = d^{\rightarrow}(s) - d^{\leftarrow}(s) = d^{\leftarrow}(t) - d^{\rightarrow}(t)$$

g.d.w. in $\overrightarrow{P}$ gilt

$$k = d^{\rightarrow}(s) - d^{\leftarrow}(s) = d^{\leftarrow}(t) - d^{\rightarrow}(t) \ .$$

Folgerung 7.5. *Aus* k *kantendisjunkten* s-t-*Wegen in* $\overrightarrow{G_C}$ *können in Linearzeit* k *kantendisjunkte* s-t-*Wege in* $\overrightarrow{G}$ *berechnet werden.*

Die $\overrightarrow{C_1}, \ldots, \overrightarrow{C_l}$ *können in Linearzeit konstruiert werden (Übung).*

Ausführung von Schritt 3: Berechnung einer maximalen Anzahl von s-t-Wegen in $\overrightarrow{G_C}$

Zur Berechnung einer maximalen Anzahl von kantendisjunkten gerichteten s-t-Wegen in $\overrightarrow{G_C}$ in Linearzeit geben wir eine Prozedur an, die auf einer Right-First-Tiefensuche basiert.

RIGHT-FIRST PROZEDUR ($\overrightarrow{G_C}$, $\overrightarrow{P_C}$)

1: Seien $e_1, \ldots, e_r$ Kanten aus $\overrightarrow{G_C}$, die aus s herauslaufen; $\overrightarrow{P_C} \leftarrow \emptyset$.
2: **Für** $i := 1$ bis r **führe aus**
3: $\quad e \leftarrow e_i$
4: $\quad p_i \leftarrow \{e_i\}$
5: $\quad$ **Solange** Einlaufknoten von e nicht s oder t **führe aus**
6: $\quad\quad v \leftarrow$ Einlaufknoten von e
7: $\quad\quad e' \leftarrow$ rechteste freie auslaufende Kante von v bzgl. „Referenzkante“ von v
8: $\quad\quad p_i \leftarrow p_i \cup \{e'\};\ e \leftarrow e'$
9: $\quad\quad$ **Falls** p_i s-t-Weg **dann**
10: $\quad\quad\quad$ setze $\overrightarrow{P_C} \leftarrow \overrightarrow{P_C} \cup \{p_i\}$
11: $\quad\quad$ **Ende** „**Falls**“
12: $\quad$ **Ende** „**Solange**“
13: **Ende** „**Für**“

Version von Weihe: Referenzkante von v ist jeweils die aktuelle in v einlaufende Kante e.

Version von Coupry: Referenzkante von v ist die allererste Kante über die der Knoten v besucht wird. Diese Kante muss also beim ersten Besuch an v abgespeichert werden.

Laufzeit: Die Version von Coupry kann direkt in Linearzeit realisiert werden. Die rechteste freie auslaufende Kante bzgl. der Referenzkante ist in diesem Fall immer die nächste auslaufende Kante nach der zuletzt besetzten in der Adjazenzliste des Knoten, falls diese im Gegenuhrzeigersinn angeordnet ist. Die Gesamtlaufzeit ist also linear in der Anzahl der Kanten in $\overrightarrow{G_C}$ amortisiert über alle Durchläufe $i = 1$ bis r von Schritt 2.

Um die Version von Weihe in Linearzeit zu realisieren, werden die Suchschritte an jedem Knoten als Folge von UNION- und FIND-Operationen ausgeführt. Die hier verwendete Version von UNION-FIND ist linear.

Korrektheit: Wir beweisen die Korrektheit für die Version von Weihe. Die Schleife 5. der RIGHT-FIRST-PROZEDUR endet immer in s oder t, da für jeden Knoten v in $\overrightarrow{G_C}$ per Konstruktion $d^{\rightarrow}(v) = d^{\leftarrow}(v)$. Wir beweisen, dass am Ende $\overrightarrow{P_C}$ eine maximale Anzahl kantendisjunkter gerichteter s-t-Wege enthält. Dazu benutzen wir die gerichtete, kantendisjunkte Version des *Satz von Menger*.

Satz 7.6. *Die Maximalzahl gerichteter kantendisjunkter* s-t-*Wege in einem gerichteten Graphen ist gleich der minimalen Kardinalität eines* s-t-*Schnitts. Dabei ist* $A \subseteq \overrightarrow{E}$ *ein* s-t-*Schnitt in einem (beliebigen) gerichteten Graphen* $\overrightarrow{G} = (V, \overrightarrow{E})$*, wenn* $\overrightarrow{G} - A$ *keinen gerichteten* s-t-*Weg enthält.*

Wir konstruieren basierend auf den in Schleife 5 berechneten $p_1, \ldots, p_r$ (d.h. $\overrightarrow{P_C}$ zusammen mit den p_i, die in s geendet haben) einen gerichteten Kreis $\overrightarrow{K}$ in $\overrightarrow{G_C}$ mit folgenden Eigenschaften:

i) s liegt in Inneres($\overrightarrow{K}$) oder auf $\overrightarrow{K}$,

ii) t liegt in Äußeres($\overrightarrow{K}$),

iii) die Anzahl der Kanten, die von $\overrightarrow{K}$ aus in Äußeres($\overrightarrow{K}$) zeigen, ist gleich der Anzahl der s-t-Wege in $\overrightarrow{P_C}$.

Wenn i – iii gelten, so induziert $\overrightarrow{K}$ einen s-t-Schnitt mit der gewünschten Kardinalität.

Konstruktion von $\overrightarrow{K}$

Seien $p_1, \ldots, p_r$ die Linkskreise und s-t-Wege, die von der RIGHT-FIRST-PROZEDUR berechnet werden. $\overrightarrow{K}$ wird ausgehend von einer Kante (v, s), die von einem der $p_1, \ldots, p_r$ besetzt ist, mittels einer rückwärts gerichteten Suche mit Left-First-Auswahlregel konstruiert. Falls es keine solche Kante (v, s) gibt, so setze $\{s\} := \overrightarrow{K}$. Ansonsten ist $\overrightarrow{K}$ eine Folge von Kanten $(v_r, v_{r-1}) \ldots (v_1, v_0)$ mit $(v_r, v_{r-1}) = (v, s)$. Die Kante (v_{i+1}, v_i) ist die im Uhrzeigersinn nächste Kante nach (v_i, v_{i-1}) in der Adjazenzliste von v_i, die von einem der $p_1, \ldots, p_r$ besetzt ist. (v_1, v_0) ist entweder die erste Kante nach (v_r, v_{r-1}) mit $v_0 = s$ oder die erste Kante in der Folge, für die die nächste zu wählende Kante bereits zu $\overrightarrow{K}$ gehört.

Nach Konstruktion von $\overrightarrow{K}$ gelten offensichtlich i und ii.

Lemma 7.7. *Betrachte* $\overrightarrow{G_C} = (V, \overrightarrow{E_C})$ *und* $\overrightarrow{K}$*. Jede Kante* $(u, v) \in \overrightarrow{E_C}$ *mit* u *auf* $\overrightarrow{K}$ *und* $v \in$ Äußeres($\overrightarrow{K}$) *gehört zu einem der* s-t-*Wege auf* $\overrightarrow{P_C}$

Beweis. Nach Konstruktion von $\overrightarrow{K}$ zeigt keine Kante, die von einem der Wege und Linkskreise $p_1, \ldots, p_r$ besetzt ist, von Äußeres($\overrightarrow{K}$) auf $\overrightarrow{K}$. Dann kann jedoch auch keine Kante $(u, v) \in \overrightarrow{E_C}$ mit u auf $\overrightarrow{K}$ und $v \in$ Äußeres($\overrightarrow{K}$) von einem der Linkskreise aus $p_1, \ldots, p_r$ besetzt sein.

Zu einem Knoten u auf $\overrightarrow{K}$ betrachte die Kante (u, w) auf $\overrightarrow{K}$. Die Referenzkante von u zeigt von Inneres($\overrightarrow{K}$) auf u oder liegt auf $\overrightarrow{K}$, und deshalb liegt (u, v), mit $v \in$ Äußeres($\overrightarrow{K}$) rechts von (u, w) bzgl. der Referenzkante von u. Dann muß aber (u, v) durch einen s-t-Weg aus $\overrightarrow{P_C}$ besetzt sein, ansonsten wäre vor (u, w) die Kante (u, v) im Algorithmus durch ein p_i besetzt worden. □

Aus Lemma 7.7 folgt dann direkt das folgende Lemma und damit die Korrektheit von Schritt 3.

Lemma 7.8. *Die Menge* $A := \{(u, v) \in \overrightarrow{E_C} : u \text{ auf } \overrightarrow{K}, v \in \text{Äußeres}(\overrightarrow{K})\}$ *ist ein* s-t-*Schnitt von* $\overrightarrow{G_C}$ *und* $|A| = |\overrightarrow{P_C}|$.

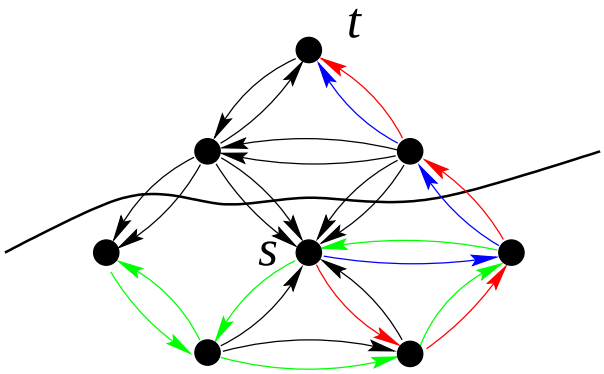


Abbildung 7.5: Wege p_1, p_2 (rot und blau) und der Kreis p_3 in $\overrightarrow{G_C}$. $\overrightarrow{K}$ besteht aus der in s hereinführenden Kante von p_3, gefolgt von der zweiten Kante aus p_1 und vier weiteren Kanten aus p_3, und der durch $\overrightarrow{K}$ induzierte Schnitt A.

Folgerung 7.9. *Der Algorithmus zu Schritt 3 berechnet eine maximale Anzahl kantendisjunkter* s-t-*Wege in* $\overrightarrow{G_C}$. *Damit berechnet der Algorithmus insgesamt in* $\mathcal{O}(n)$ *eine maximale Anzahl kantendisjunkter* s-t- *Wege im planaren Graphen* G.

Ausführung von Schritt 4: Konstruktion einer maximalen Anzahl kantendisjunkter s-t-Wege in G

Berechne in Linearzeit aus der Menge der kantendisjunkten s-t-Wege in $\overrightarrow{G_C}$ eine Menge kantendisjunkter s-t-Wege in G gleicher Kardinalität. Dies geht entsprechend Lemma 7.2 und Folgerung 7.5.

7.2 Das knotendisjunkte Menger-Problem

Betrachte wieder einen planaren Graphen $G = (V, E)$ mit einer planaren Einbettung, bei der t auf der äußeren Facette liegt. Wir behandeln einen Linearzeitalgorithmus zur Lösung des knotendisjunkten Menger Problems.

Knotendisjunkter Menger-Algorithmus

Schritt 1: Ersetze $G = (V, E)$ durch den gerichteten Graphen $\overrightarrow{G} := (V, \overrightarrow{E})$, der entsteht, indem jede Kante $\{u, v\} \in E$ ersetzt wird durch die beiden gerichteten Kanten (u, v), $(v, u) \in \overrightarrow{E}$, falls sie nicht mit s oder t inzident ist; Kanten $\{u, s\} \in E$ nur durch (s, u) und Kanten $\{u, t\} \in E$ nur durch (u, t). Wenn $\overrightarrow{p_1}, \ldots, \overrightarrow{p_l}$ knotendisjunkte s-t-Wege in in $\overrightarrow{G}$ sind, so induzieren diese direkt knotendisjunkte s-t-Wege $p_1, \ldots, p_l$ in G.

Schritt 2: Seien $e_1, \ldots, e_r \in \overrightarrow{E}$ die Kanten aus $\overrightarrow{G}$, die aus s herauslaufen. In einer Schleife über $e_1, \ldots, e_r$ werden knotendisjunkte, gerichtete s-t-Wege mittels einer Suche mit „Right-First“ Auswahlregel konstruiert. Dabei werden „Konflikte“ zwischen bereits besetzten Knoten und dem aktuellen Suchweg geeignet „aufgelöst“.

Ein besetzter Knoten v ist mit genau einer besetzten einlaufenden Kante (u, v) und genau einer besetzten auslaufenden Kante (v, w) inzident. Der Suchweg kann also von links oder von rechts in Bezug auf (u, v), (v, w) auf v treffen.

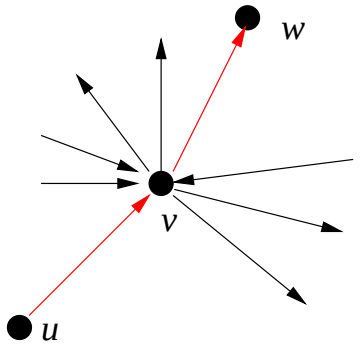


Abbildung 7.6: Eindeutig in v einlaufende und auslaufende besetzte Kanten (u, v) und (v, w).

Behandlung von Konflikten zwischen Suchweg und besetztem Knoten v.

1. Konflikt von links: Der aktuelle Suchweg trifft von links auf einen besetzten Knoten v. Dann wird ein *Backtrack-Remove-Schritt* ausgeführt, d.h. die letzte Kante des Suchwegs

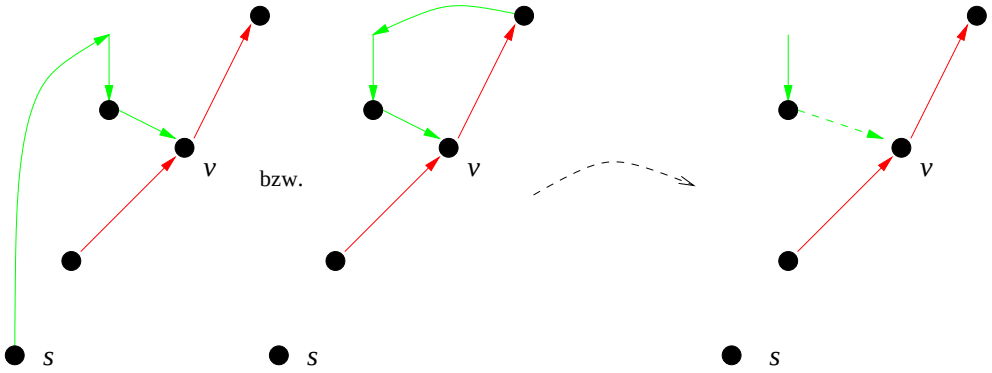


Abbildung 7.7: Konflikt von links.

vom Suchweg und aus $\overrightarrow{G}$ entfernt.

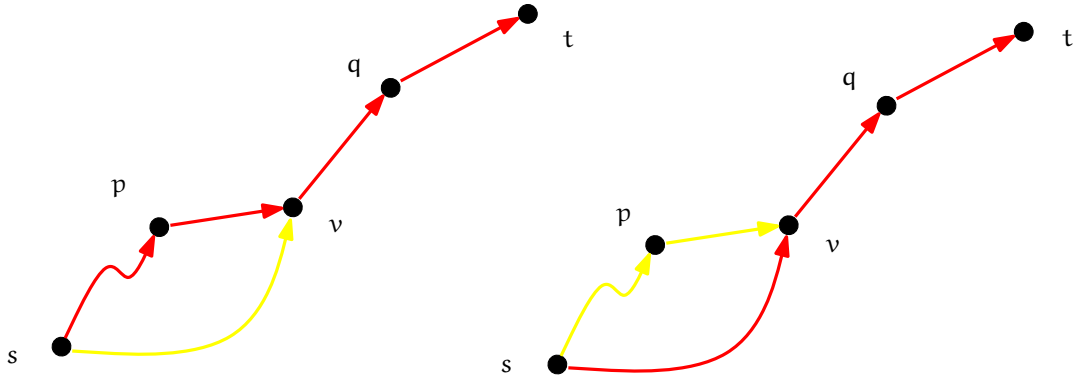


Abbildung 7.8: Umorganisation der Teilwege p, q (rot) und dem aktuellen Suchweg r (grün).

2. Konflikt von rechts: Der aktuelle Suchweg trifft von rechts auf einen besetzten Knoten v. p sei der Teilweg von s nach v, zu dem die besetzte in v einlaufende Kante gehört, q der Teilweg von v nach t, zu dem die besetzte aus v auslaufende Kante gehört, und r der aktuelle Suchweg. Die Teilwege r und q werden zu einem s-t-Weg zusammengesetzt, und p als aktueller Suchweg betrachtet. Dies kann als eine „Umorganisation von Wegen“ angesehen werden. Danach trifft der aktuelle Suchweg von links auf den besetzten Knoten, d.h. es tritt ein Konflikt von links ein. Dies wird wie gehabt durch einen *Backtrack-Remove-Schritt* aufgelöst. Siehe Abb. 7.8.

Voraussetzung dafür, dass diese „Umorganisation von Wegen“ sinnvoll ist, ist die Bedingung, dass die zu v inzidenten besetzten Kanten nicht zu dem aktuellen Suchweg gehören. Daher ist der Algorithmus so angelegt, dass diese Situation erst gar nicht auftritt, d.h. der Suchweg keinen Rechtskreis durchläuft. Beachte, dass $\overrightarrow{G}$ selbst Rechtskreise enthält.

Trick 1: Man kann beweisen, dass für eine Kante (v, w), über die der Suchweg einen Rechtskreis mit Konflikt in v durchlaufen würde, bereits zu einem früheren Zeitpunkt des

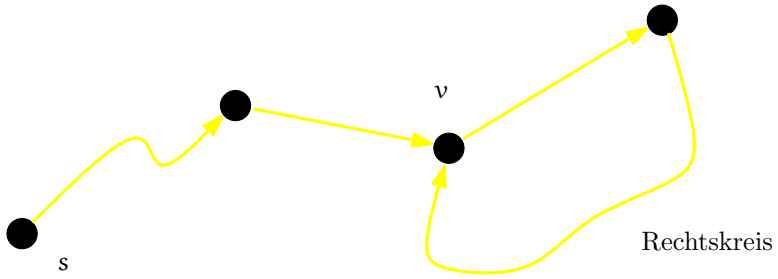
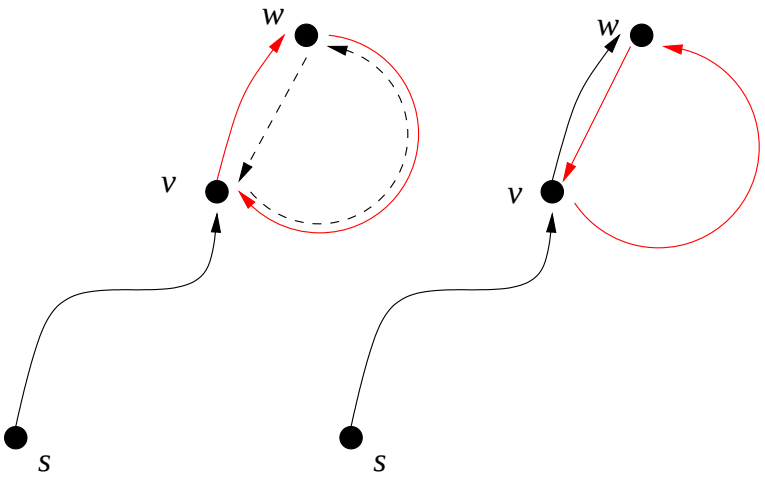


Abbildung 7.9: Konflikt von rechts an einem Knoten v durch einen Rechtskreis im aktuellen Suchweg.



Durchlauf eines Rechtskreis vorher Durchlauf von entsprechendem Linkskreis

Abbildung 7.10: Illustration zu Trick 1.

Algorithmus die umgekehrte Kante (w, v) in einem Linkskreis mit Konflikt in v durchlaufen wurde. Daher wird eine Kante (v, w) entfernt, für die gilt:

> (v, w) würde nach Right-First-Auswahlregel als nächste Kante von dem aktuellen Suchweg belegt, und (w, v) ist zuvor einmal von dem aktuellen Suchweg belegt worden.

Um diese Bedingung abzutesten, muss für eine Kante bekannt sein, ob sie bereits von dem aktuellen Suchweg belegt wurde. Es müsste also zu jeder belegten Kante gespeichert werden, von welchem Weg sie belegt ist.

Problem: Belegte Kanten ändern im Laufe des Algorithmus ihre Zugehörigkeit zu Wegen, da die Wege selbst im Laufe des Algorithmus jeweils bei einem Konflikt von rechts umorganisiert werden. Ein Update für alle betroffenen Kanten bei jeder Umorganisation würde jedoch zu quadratischer Laufzeit führen.

Trick 2: Es wird ein „globaler Zähler“ eingeführt, der immer dann erhöht wird, wenn entweder von s aus ein neuer Suchweg startet, oder wegen eines Konflikts von rechts der Suchweg umorganisiert wird. Jeder Knoten erhält einen „lokalen Zeitstempel“. Wird ein Knoten zum führenden Knoten des aktuellen Suchwegs, so wird sein lokaler Zeitstempel auf den aktuellen Wert des globalen Zählers gesetzt.

Es gilt dann: Falls Kante (w, v) im Algorithmus bereits zuvor von dem aktuellen Suchweg belegt wurde, so sind bei Betrachtung der Kante (v, w) als eventuelle nächste vom Suchweg zu belegende Kante, die lokalen Zeitstempel von v und w identisch.

RIGHT-FIRST-PROZEDUR zu Schritt 2

Insgesamt besteht nun Schritt 2 aus einer RIGHT-FIRST-PROZEDUR (Algorithmus 1). Sie beginnt jeweils bei einer aus s herausführenden Kante, und endet entweder bei t oder wieder bei s. Im Laufe des Verfahrens werden knotendisjunkte s-t-Wege konstruiert. Wegen der wiederholt durchgeführten Umorganisation von Wegen, tritt ein einmal konstruierter s-t-Weg nicht unbedingt als solcher in der endgültigen Lösung auf.

Laufzeit: Die Gesamtzahl der Durchläufe von 6. ist linear in der Anzahl der Kanten von $\overrightarrow{G}$, denn jede einzelne Kante von $\overrightarrow{G}$ ist zunächst unbesetzt, wird im Laufe des Algorithmus höchstens einmal besetzt, und höchstens einmal nach Besetzen („ansonsten“ in 14.) bzw. direkt bei erster Betrachtung (in 18.) entfernt. Die Anzahl der Operationen, die für jede Kante ausgeführt wird, ist konstant. Beachte dazu, dass die Auswahl der rechtesten freien herausführenden Kante bzgl. der führenden Kante des aktuellen Suchwegs in 16. in konstanter Zeit ausgeführt werden kann. Die gesuchte Kante ist immer die nächste freie herausführende Kante in der Adjazenzliste. Denn falls für einen Knoten v dieser Auswahlschritt mehrfach ausgeführt wird, so ist die „Referenzkante“ immer dieselbe. Damit ist die Gesamtlaufzeit des Algorithmus in $\mathcal{O}(n)$.

Algorithmus 1 RIGHT-FIRST-PROZEDUR zu Schritt 2

1: Seien $e_1, \ldots, e_r$ die Kanten von $\overrightarrow{G} = (V, \overrightarrow{E})$, die aus s herausführen. Die Reihenfolge ist beliebig.
2: Setze Zähler $\leftarrow 0$
3: **Für** $i := 1$ bis r **führe aus**
4: Aktueller Suchweg bestehe aus Kante $e_i := (s, v)$.
5: Setze Zähler $\leftarrow$ Zähler $+ 1$.
6: **Solange** $v \notin \{s, t\}$ **führe aus**
7: Setze Zeitstempel(v) $\leftarrow$ Zähler
8: **Falls** der aktuelle Suchweg in v einen Konflikt von links hat **dann**
9: führe „Backtrack-Remove“ aus.
10: **sonst**
11: **Falls** der aktuelle Suchweg in v einen Konflikt von rechts hat **dann**
12: führe „Umorganisation“ aus.
13: Setze Zähler $\leftarrow$ Zähler $+ 1$.
14: **sonst**
15: **Falls** eine unbesetzte aus v herausführende Kante (verschieden von der Gegenkante zur gerade führenden Kante des aktuellen Suchweges) existiert **dann**
16: wähle die rechteste freie herausführende Kante bzgl. der führenden Kante des aktuellen Suchweges. (v, w) sei diese Kante.
17: **Falls** Falls Zeitstempel(v) = Zeitstempel(w) **dann**
18: entferne (v, w).
19: **sonst**
20: füge (v, w) zum aktuellen Suchweg hinzu.
21: **Ende** „**Falls**“
22: **sonst**
23: führe „Backtrack-Remove“ mit der führenden Kante des aktuellen Suchwegs aus.
24: **Ende** „**Falls**“
25: **Ende** „**Falls**“
26: **Ende** „**Falls**“
27: Setze $v \leftarrow$ führender Knoten des aktuellen Suchwegs.
28: **Ende** „**Solange**“
29: **Ende** „**Für**“

Korrektheit: Wir führen hier keinen ausführlichen Korrektheitsbeweis, da dieser relativ aufwendig ist. Die Korrektheit des Algorithmus kann bewiesen werden, indem die folgenden beiden „Invarianten“ für den Algorithmus bewiesen werden.

Invariante 1: Der Suchweg formt zu keinem Zeitpunkt des Algorithmus einen Rechtszykel. (Dies bedeutet, dass Trick 1 und 2 greifen.)

Invariante 2: Die maximale Anzahl knotendisjunkter s-t-Wege in $\overrightarrow{G}$ sei k. $\{a_1, \ldots, a_m\}$ seien die Kanten aus $\overrightarrow{G}$, die im Laufe des Algorithmus (in dieser Reihenfolge) entfernt werden. Dann gilt: Der Graph $\overrightarrow{G_i} := (V, \overrightarrow{E} \backslash \{a_1, \ldots, a_i\}), 1 \leq i \leq m$, enthält ebenfalls k knotendisjunkte s-t-Wege.

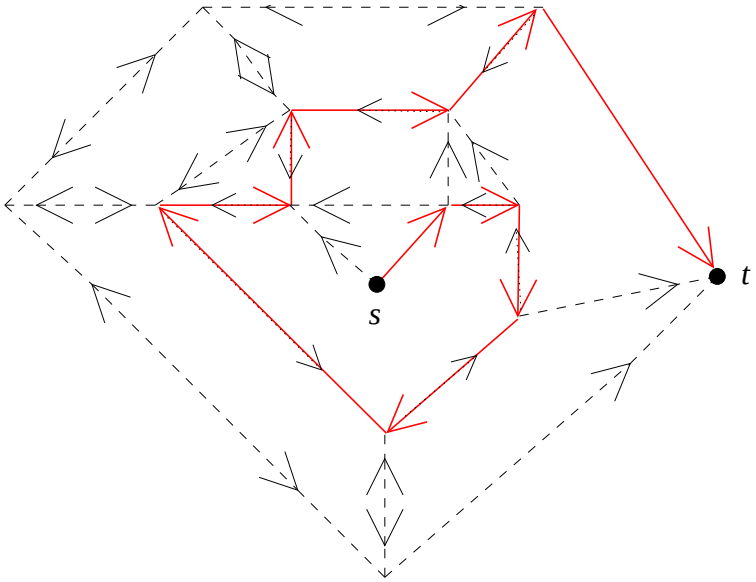


Abbildung 7.11: Die Situation nach dem ersten Durchlauf von 3. mit dem in diesem Durchlauf berechneten s-t-Weg (rot). Zwei gegenläufige Pfeile geben an, dass beide gerichteten Kanten noch im Graph existieren. Beachte, dass rechts des ersten Weges keine auslaufende Kante mehr existiert.

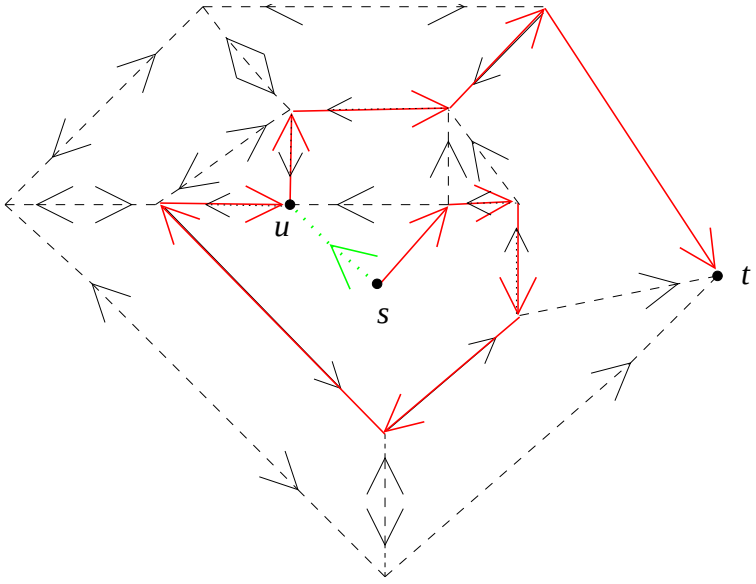


Abbildung 7.12: Die Situation nach dem ersten Suchschritt des zweiten Durchlauf von 3. Der Suchweg (grün gestrichelt), d.h. $s \to u$ trifft im Knoten u von rechts den s-t-Weg, der im ersten Durchlauf konstruiert wurde.

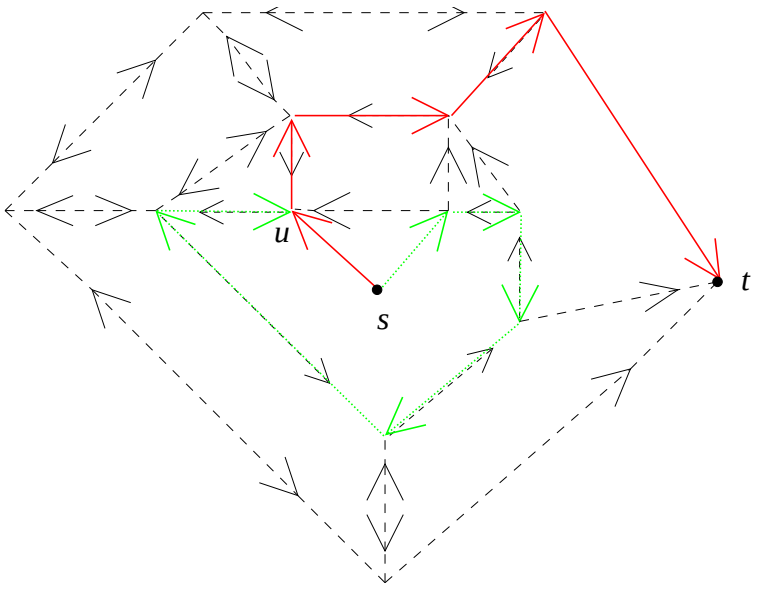


Abbildung 7.13: Der Teilwegvon $s \to u$ des ersten s-t-Weges wird zum neuen Suchweg (grün gestrichelt), und der vorherige Suchweg wird mit dem Teilweg von $u \to t$ des ersten s-t-Weges zu einem s-t-Weg (rot) zusammengesetzt.

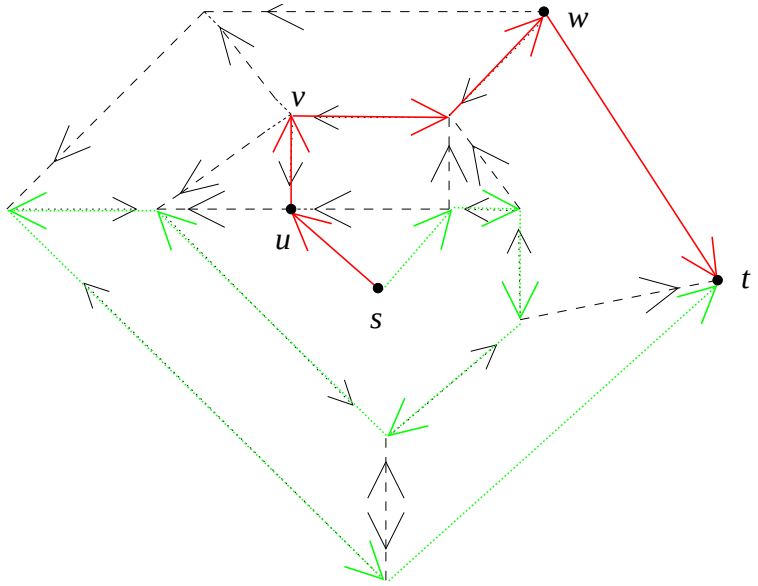


Abbildung 7.14: Alle weiteren Konflikte im zweiten Durchlauf sind Konflikte, bei denen der Suchweg (grün gestrichelt) den konstruierten s-t-Weg (rot) von links trifft, zweimal in v und später einmal in w.

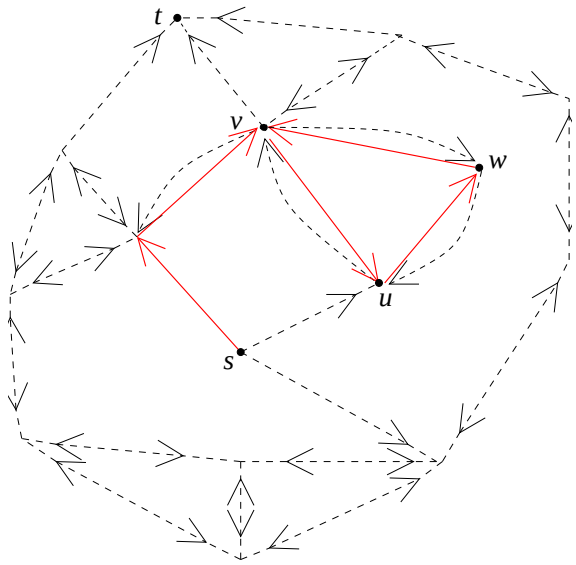


Abbildung 7.15: Ein Beispiel, in dem ein Konflikt auftritt, der den vermeintlichen Durchlauf eines Rechtskreises „ankündigt“.

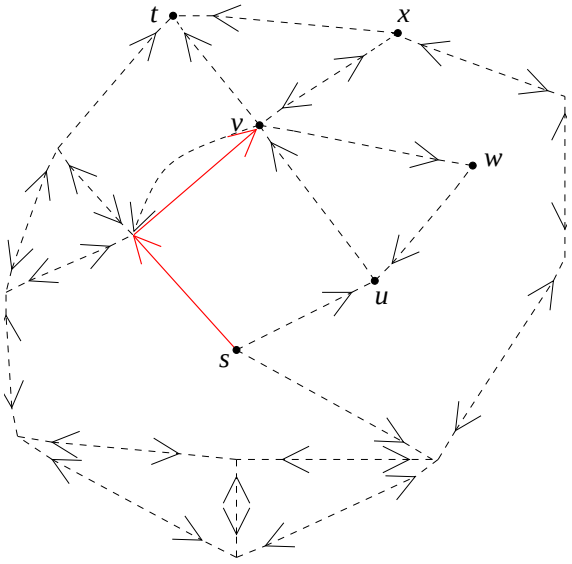


Abbildung 7.16: Nach drei Backtrack-Remove-Schritten. Nun ist (v, w) die nächste Kante, die der Suchweg betrachtet. Die Zeitstempel von v und w sind zu diesem Zeitpunkt identisch, also wird (v, w) nicht gewählt, sondern entfernt und stattdessen (v, x) gewählt. Würde der Algorithmus tatsächlich (v, w) wählen, so würde ein Rechtkreis $v \to w \to u \to v$ durchlaufen. Auflösung des entsprechenden Konflikts von rechts durch Entfernen von (u, v) würde alle optimalen Lösungen zerstören!

8 Das Problem von Okamura und Seymour

Das Menger-Problem kann man als ein „Wegpackungsproblem“ auffassen, d.h. es sollen knotendisjunkte bzw. kantendisjunkte Wege zwischen vorgegebenen Knoten in einen Graph „gepackt“ werden. Ein allgemeineres kantendisjunktes Wegpackungsproblem kann folgendermaßen formuliert werden.

> Kantendisjunktes Wegpackungsproblem
> Gegeben sei ein Graph $G = (V, E)$ und Paare ausgezeichneter Knoten $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, $s_i, t_i \in V$ (nicht notwendig paarweise verschieden). Finde paarweise kantendisjunkte s_i-t_i-Wege p_i in G, $1 \leq i \leq k$. Die s_i, t_i werden *Terminale* genannt, die Mengen $\{s_i, t_i\}$ heißen *Netze*.

Dieses Problem ist $\mathcal{NP}$-vollständig, auch falls G planar ist. Naheliegende Einschränkungen des Problems in planaren Graphen sind:

- Die Lage der s_i, t_i ist „eingeschränkt“, etwa alle s_i, t_i, $1 \leq i \leq k$ liegen auf dem Rand derselben Facette.
- Sei $D := \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ (Menge der Demand-Kanten), dann soll $G + D := (V, E \cup D)$ planar sein.

Ein wichtiges Kriterium für die Lösbarkeit solcher Probleme ist ähnlich wie beim kantendisjunkten Menger-Problem die „Kapazität“ des Graph.

Definition 8.1. *Sei* $G = (V, E)$, $X \subseteq V$. *Dann heißt*

$$\mathrm{cap}(X) := |\{\{u, v\} \in E : u \in X,\, v \in V \backslash X\}|$$

die Kapazität *von* X *(Größe des durch* X *induzierten Schnittes). Zu* $G = (V, E)$ *sei* $D = \{\{s_i, t_i\} : s_i, t_i \in V, 1 \leq i \leq k\}$ *gegeben. Dann heißt*

$$\mathrm{dens}(X) := |\{\{s_i, t_i\} \in D : |\{s_i, t_i\} \cap X| = 1\}|$$

die Dichte *von* X.

$$\mathrm{fcap}(X) := \mathrm{cap}(X) - \mathrm{dens}(X)$$

bezeichne die freie Kapazität *von* X. X *heißt* saturiert*, falls* $\mathrm{fcap}(X) = 0$ *und* übersaturiert*, falls* $\mathrm{fcap}(X) < 0$.

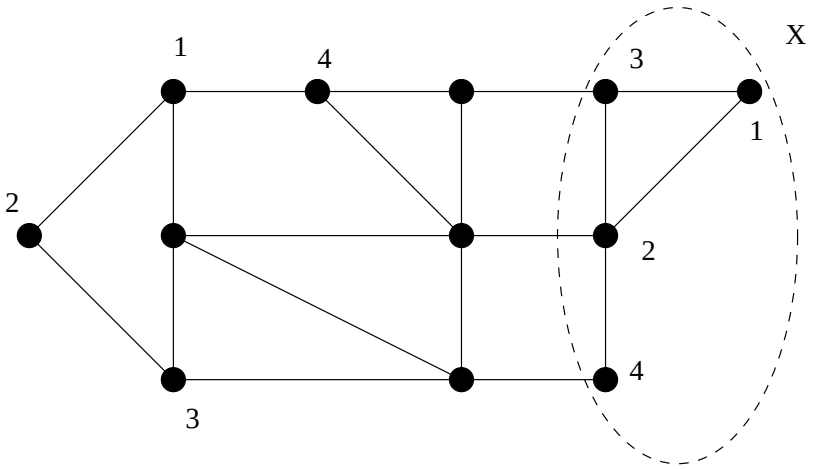


Abbildung 8.1: Ein Problembeispiel mit Schnitt X, für den $\mathrm{cap}(X) = 3$, $\mathrm{dens}(X) = 4$ und $\mathrm{fcap}(X) < 0$ gilt

Eine notwendige Bedingung für die Lösbarkeit des kantendisjunkten Wegpackungsproblems ist offensichtlich

$$\mathrm{fcap}(X) \geq 0 \quad \text{für alle} \quad X \subseteq V.$$

Wir nennen diese Bedingung *Kapazitätsbedingung*. Im allgemeinen ist die Kapazitätsbedingung keine hinreichende Bedingung für die Lösbarkeit des kantendisjunkten Wegpackungsproblems. Siehe Abb. 8.2.

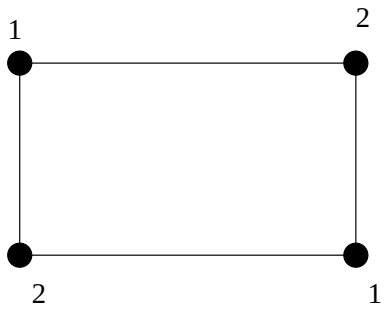


Abbildung 8.2: Ein Problembeispiel, das die Kapazitätsbedingung erfüllt, d.h. $\mathrm{cap}(X) \geq 2$ für alle $\emptyset \neq X \subset V$ und $\mathrm{dens}(X) \leq 2$, aber nicht lösbar ist.

Ein Graph $G = (V, E)$ mit $D = \{\{s_i, t_i\} : s_i, t_i \in V,\ 1 \leq i \leq k\}$ erfüllt die *Geradheitsbedingung* (auch *Euler-Bedingung* genannt), falls $\mathrm{fcap}(X)$ gerade ist für alle $X \subseteq V$.

Wir betrachten ab jetzt folgenden Spezialfall des kantendisjunkten Wegpackungsproblems:

Okamura & Seymour-Problem
Gegeben sei ein planarer Graph $G = (V, E)$ und Paare ausgezeichneter Knoten $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, $s_i, t_i \in V$, wobei alle s_i, t_i auf dem Rand derselben Facette liegen. O.B.d.A. sei G so planar eingebettet, dass s_i, t_i, $1 \leq i \leq k$ auf der äußeren Facette liegen. Außerdem sei die Geradheitsbedingung erfüllt. Finde paarweise kantendisjunkte s_i-t_i-Wege p_i in G, $1 \leq i \leq k$.

Lemma 8.2. *Sei* $G = (V, E)$, $D := \{\{s_i, t_i\} : s_i, t_i \in V, 1 \leq i \leq k\}$. *Es gilt* $\mathrm{fcap}(X)$ *gerade für alle* $X \subseteq V$ *genau dann, wenn* $\mathrm{fcap}(v) := \mathrm{fcap}(\{v\})$ *gerade für alle* $v \in V$.

Beweis. „$\Longrightarrow$“ ist trivial.

„$\Longleftarrow$“ Sei also $\mathrm{fcap}(v)$ gerade für alle $v \in V$. Es gilt für $X \subseteq V$

$$\mathrm{cap}(X) = \sum_{v \in X} \mathrm{cap}(v) - 2 \cdot \left|\left\{\{u, v\} \in E : u, v \in X\right\}\right| \quad \text{und}$$
$$\mathrm{dens}(X) = \sum_{v \in X} \mathrm{dens}(v) - 2 \cdot \left|\left\{\{s_i, t_i\} \in D : s_i, t_i \in X\right\}\right|.$$

Dann ist

$$\begin{aligned} \mathrm{fcap}(X) &= \sum_{v \in X} \mathrm{cap}(v) - \sum_{v \in X} \mathrm{dens}(v) - 2 \cdot \left|\left\{\{u, v\} \in E : u, v \in X\right\}\right| \\ &\quad +2\left|\left\{\{s_i, t_i\} \in D : s_i, t_i \in X\right\}\right| \\ &= \sum_{v \in X} \mathrm{fcap}(v) - 2 \cdot \Big(\left|\left\{\{u, v\} \in E : u, v \in X\right\}\right| \\ &\quad +\left|\left\{\{s_i, t_i\} \in D : s_i, t_i \in X\right\}\right|\Big) \end{aligned}$$

Damit ist $\mathrm{fcap}(X)$ gerade, falls alle $\mathrm{fcap}(v)$ für $v \in X$ gerade sind. □

Satz 8.3 (Satz von Okamura & Seymour, 1981). *Gegeben sei ein planar eingebetteter Graph* $G = (V, E)$ *und* $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$, *wobei* $s_1, \ldots, s_k, t_1, \ldots, t_k$ *auf dem Rand der äußeren Facette von* G *liegen. Die Kapazitätsbedingung und die Geradheitsbedingung seien erfüllt. Dann existieren paarweise kantendisjunkte* s_i-t_i-*Wege in* G.

Der Beweis von Okamura & Seymour zu Satz 8.3 führt direkt zu einem $\mathcal{O}(n^5)$ Algorithmus zur Lösung des kantendisjunkten Wegpackungsproblems in Graphen, die die Bedingungen von Satz 8.3 erfüllen. Dieser kann auf $\mathcal{O}(n^2)$ verbessert werden (Becker & Mehlhorn 1986; Matsumoto, Nishizeki & Saito 1985). Dabei werden explizit Kapazitäten und Dichten von Schnitten untersucht unter Benutzung der Dualität zwischen Schnitten und Kreisen.

Wir werden hier einen Linearzeitalgorithmus behandeln, der wiederum auf einer Tiefensuche mit Right-First-Auswahlregel basiert.

Sei im folgenden $G = (V, E)$ planar eingebettet, so dass $s_1, t_1, \ldots, s_k, t_k$ auf dem Rand der äußeren Facette liegen und die Geradheitsbedingung sei erfüllt. O.B.d.A. sei G zweifach zusammenhängend; diese Annahme dient nur der Vereinfachung der Darstellung.

Der Algorithmus besteht aus zwei Phasen. Zunächst wird mittels einer Right-First-Tiefensuche eine Lösung für das kantendisjunkte Wegpackungsproblem für ein modifiziertes Problembeispiel mit einer „einfacheren“ Struktur bestimmt. Diese Lösung induziert einen gerichteten Hilfsgraphen $\overrightarrow{G}$, in dem dann wieder mittels Right-First-Tiefensuche das Ausgangsproblem gelöst wird.

Linearzeitalgorithmus für das Okamura & Seymour-Problem

Schritt 1: Konstruiere aus $G = (V, E)$ mit $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ ein Problem bestehend aus $G = (V, E)$ mit $D' = \{\{s_1't_1'\}, \ldots, \{s_k', t_k'\}\}$ mit „einfacherer“ Struktur.

Schritt 2: Gegeben sei $G = (V, E)$ und $D' = \{\{s_i', t_i'\}, \ldots, s_k', t_k'\}\}$. Berechne in $\mathcal{O}(n)$ kantendisjunkte s_i'-t_i'-Wege mittels Right-First-Tiefensuche und Orientierung der entsprechenden Wege von s_i' nach t_i'. Diese induzieren einen gerichteten Graph $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$.

Schritt 3: Gegeben sei nun der durch Schritt 2 induzierte Graph $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$ und $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$. Berechne in $\mathcal{O}(n)$ kantendisjunkte gerichtete s_i-t_i-Wege p_i in $\overrightarrow{G}$, welche kantendisjunkte s_i-t_i-Wege in G induzieren.

Definition 8.4. $G = (V, E)$ *mit* $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ *hat* Klammerstruktur*, falls* $G + D$ *so planar eingebettet werden kann, dass die Kanten aus* D *kreuzungsfrei in die äußere Facette der entsprechenden Einbettung von* G *eingebettet sind. Siehe Abb. 8.3.*

Ausführung von Schritt 1: Konstruktion von D' mit Klammerstruktur.

Konstruiere aus $G = (V, E)$ mit $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ ein Problem bestehend aus $G = (V, E)$ mit $D' = \{\{s_1't_1'\}, \ldots, \{s_k', t_k'\}\}$, so dass $\{s_1, \ldots, s_k, t_1, \ldots, t_k\} = \{s_1', \ldots, s_k', t_1', \ldots, t_k'\}$ und die $\{s_1', t_1'\}, \ldots, \{s_k', t_k'\}$ *Klammerstruktur* haben.

Zu einem Problembeispiel $G = (V, E)$ mit $D = \{\{s_1, t_1, \}, \ldots, \{s_k, t_k\}\}$ (ohne Klammerstruktur) kann ein Problembeispiel G, D' mit Klammerstruktur leicht in $\mathcal{O}(n)$ konstruiert werden. Wähle dazu ein beliebiges Terminal (s_i oder t_i) als Startterminal s. Beginnend bei s gehe im Gegenuhrzeigersinn um die äußere Facette. Dem jeweils ersten Terminal eines $\{s_i, t_i\}$ ordne eine öffnende Klammer zu und dem jeweils zweiten eine schließende

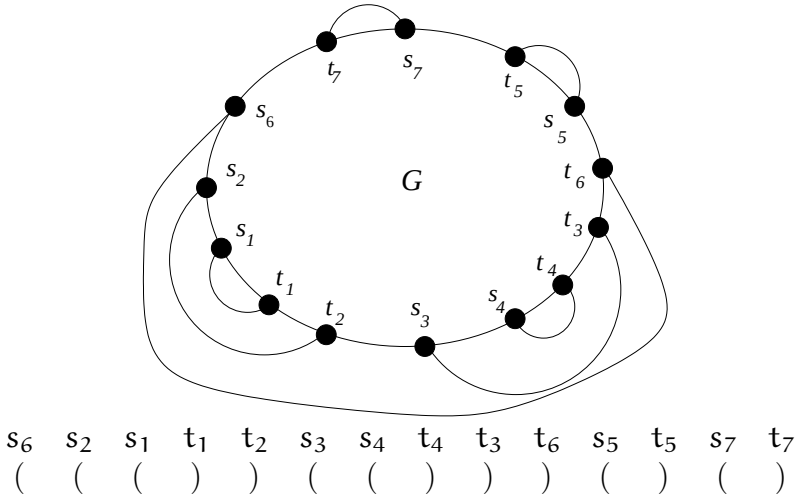


Abbildung 8.3: Beginnend mit einem beliebigen s_i, z.B. s_6 haben die Netze *Klammerstruktur* d.h. Paarung der $\{s_i, t_i\}$ entspricht einer korrekten Klammerung der entsprechenden Klammern.

Klammer. Die korrekte Klammerung dieser Klammern beginnend mit der Klammer zu s induziert D' (Realisierung mit STACK siehe Übung).

Ausführung von Schritt 2: Berechnung der s'_i-t'_i-Wege $q_1, \ldots, q_k$

Gegeben sei $G = (V, E)$ und $D' = \{\{s'_i, t'_i\}, \ldots, s'_k, t'_k\}\}$ mit Klammerstruktur. Berechne in $\mathcal{O}(n)$ kantendisjunkte s'_i-t'_i-Wege mittels Right-First-Tiefensuche und eine Orientierung der entsprechenden Wege von s'_i nach t'_i.

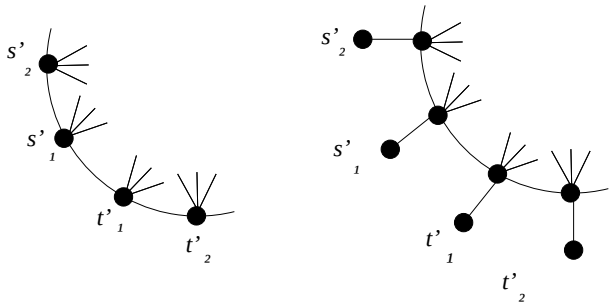


Abbildung 8.4: Aus technischen Gründen hinzugefügte „Dummy-Kanten“.

Die s'_i, t'_i seien, beginnend beim Startterminal s im Gegenuhrzeigersinn so angeordnet, dass jeweils s'_i vor t'_i und t'_i vor t'_{i+1}. Aus technischen Gründen füge jeweils Kanten wie in Abb. 8.4 hinzu.

RIGHT-FIRST-PROZEDUR ($G = (V, E)$, $D' = \{\{s'_i, t'_i\}\}$)

1: **Für** $i := 1$ bis k **führe aus**
2: Füge zu q_i die eindeutige zu s'_i inzidente Kante ausgehend aus s'_i orientiert als führende Kante hinzu.
3: Setze $v \leftarrow$ eindeutiger zu s'_i adjazenter Knoten.
4: **Solange** $v \notin \{s'_j, t'_j, 1 \leq j \leq k\}$ **führe aus**
5: Sei $\{v, w\}$ rechteste freie Kante bzgl. der führenden Kante von q_i.
6: Füge (v, w) zu q_i als führende Kante hinzu.
7: Setze $v \leftarrow w$.
8: **Ende** „**Solange**“
9: **Falls** $v \neq t'_i$ **dann**
10: gib „unlösbar“ aus.
11: **Ende** „**Falls**“
12: **Ende** „**Für**“
13: Gib $q_1, \ldots, q_k$ aus.

Offensichtlich endet wegen der Geradheitsbedingung jeder Durchlauf von 1. bei einem Terminalknoten.

Beobachtung. *Wegen der Right-First-Auswahlregel können an einem Knoten v Reihenfolgen von Kanten wie in Abb. 8.5 nicht vorkommen. Daher gilt für die Wege q_i:*

i. Keine zwei Wege q_i und q_j kreuzen sich.

ii. Kein Weg q_i kreuzt sich selbst.

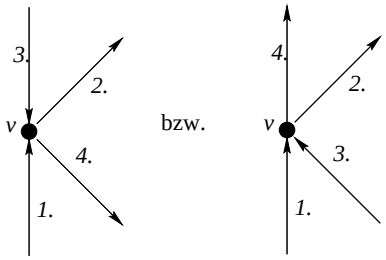


Abbildung 8.5: Reihenfolgen von Kanten, die wegen der Right-First-Auswahlregel nicht vorkommen können.

Sei $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E}), \overrightarrow{V} \subseteq V$, der Graph, der durch die von $q_1, \ldots, q_k$ belegten Kanten zusammen mit der Orientierung induziert wird. Dann gilt für $v \in \overrightarrow{V} \backslash \{s'_i, t'_i : 1 \leq i \leq k\}$, dass $d^{\leftarrow}(v) = d^{\rightarrow}(v)$ in $\overrightarrow{G}$.

Corollar 8.5. $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$ *enthält keinen Rechtskreis.*

Beweis. Wenn $\overrightarrow{C}$ Kanten eines Rechtskreises in $\overrightarrow{G}$ sind, so gehören nicht alle Kanten aus $\overrightarrow{C}$ zu dem selben Weg q_i. Seien q_i, q_j Wege, die Kanten aus $\overrightarrow{C}$ besetzen. Da q_i und

q_j sich nicht kreuzen, müssen die Terminale s'_i, t'_i und s'_j, t'_j in der Reihenfolge s'_i, t'_j, s'_j, t'_i im Gegenuhrzeigersinn auf der äußeren Facette von G liegen. Dies ist ein Widerspruch. Siehe Abb. 8.6. □

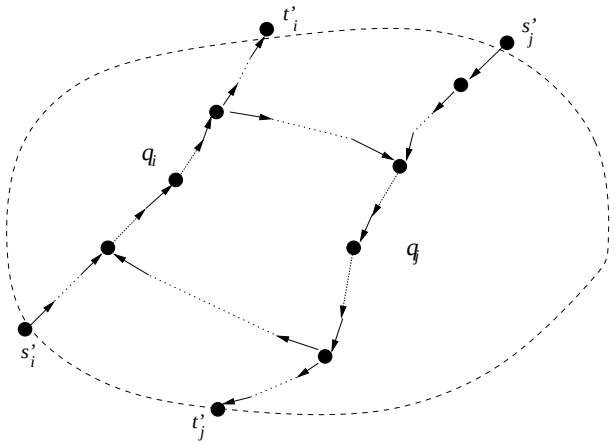


Abbildung 8.6: Durch einen Rechtskreis induzierte Reihenfolge der Terminale s'_i, t'_j, s'_j, t'_i.

Lemma 8.6. *Für ein lösbares Problem des Wegpackungsproblems* $G = (V, E)$ *und* $D' = \{\{s'_1, t'_1\}, \ldots, \{s'_k, t'_k\}\}$*, wobei* D' *Klammerstruktur hat,* s'_i, t'_i *auf dem Rand der äußeren Facette liegen und die Geradheitsbedingung erfüllt ist, berechnet die RIGHT-FIRST-PROZEDUR(*G, D'*) kantendisjunkte* s'_i-t'_i*-Wege* q_i*, für* $1 \leq i \leq k$.

Beweis. Betrachte eine beliebige kreuzungsfreie Lösung $q'_1, ..., q'_k$. Eine solche Lösung existiert immer! Dann können wir für jedes q'_i dessen linke und deren rechte Seite betrachten. Ebenso können wir für die von der RIGHT-FIRST-PROZEDUR konstruierten Wege q_i linke und rechte Seite betrachten.

Wenn $q'_1, ..., q'_k$ kreuzungsfrei ist, so gilt für jedes q'_i, dass es vollständig zur linken Seite aller $q'_1, \ldots, q'_{i-1}$ gehört. Induktiv über i, $1 \leq i \leq k$, gilt für jedes q_i, da es mittels Right-First-Auswahlregel und entsprechend der Klammerstruktur von D' bestimmt wurde, dass die linke Seite von q'_i ganz enthalten ist in der linken Seite von q_i. Daraus folgt insbesondere, dass q_i die Terminale s'_i und t'_i verbindet. □

Laufzeit: Offensichtlich ist Schritt 2 amortisiert in $\mathcal{O}(n)$ realisierbar, da die rechteste freie Kante bzgl. der führenden Kante immer die nächste Kante nach der führenden Kante in der (aktuellen) Adjazenzliste ist, und damit in konstanter Zeit gefunden werden kann.

Ausführung von Schritt 3: Berechnung der s_i-t_i-Wege $p_1, \ldots, p_k$

Gegeben sei nun der durch Schritt 2 induzierte Graph $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$ und $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$, wobei ausgehend vom Startterminal s aus Schritt 1 im Gegenuhrzeigersinn s_i vor t_i liegt, und o.B.d.A. die Indizierung so ist, dass t_i vor t_{i+1}. Berechne in $\mathcal{O}(n)$ kantendisjunkte s_i-t_i-Wege p_i in $\overrightarrow{G}$ mittels gerichteter Right-First-Tiefensuche, und zwar der Reihe nach entsprechend der Indizierung der $\{s_i', t_i'\}$ (also entsprechend dem Auftreten der t_i ausgehend von s im Gegenuhrzeigersinn).

RIGHT-FIRST-PROZEDUR ($\overrightarrow{G}$, D)

1: **Für** $i = 1$ bis k **führe aus**
2: Füge zu p_i die eindeutige aus s_i herausführende Kante von $\overrightarrow{G}$ als führende Kante hinzu.
3: Setze $v \leftarrow$ Einlaufknoten dieser Kante.
4: **Solange** $v \notin \{s_j, t_j : 1 \leq j \leq k\}$ **führe aus**
5: Sei (v, w) die rechteste freie Kante, die aus v herausführt bzgl. der führenden Kante von p_i, dann füge (v, w) zu p_i als führende Kante hinzu.
6: Setze $v \leftarrow w$.
7: **Ende** „**Solange**“
8: **Falls** $v \neq t_i$ **dann**
9: gib aus „unlösbar“.
10: **Ende** „**Falls**“
11: **Ende** „**Für**“
12: Gib $p_1, \ldots, , p_k$ aus.

Korrektheit: Um zu beweisen, dass Schritt 3 korrekte s_i-t_i-Wege p_i konstruiert für ein lösbares Problem geben wir einen Linearzeitalgorithmus an, der für jedes p_i einen Weg p angibt mit der Eigenschaft:

i. Falls p_i korrekt s_i mit t_i verbindet, so induziert p einen saturierten Schnitt in G.
ii. Falls p_i Terminal s_i nicht mit t_i verbindet, aber jedes p_j, $1 \leq j \leq i$ korrekt s_j mit t_j verbindet, so induziert p einen übersaturierten Schnitt in G.

Mit ii. ist im Fall, dass ein p_i nicht s_i mit t_i verbindet, die Kapazitätsbedingung, und damit eine notwendige Bedingung für Lösbarkeit, verletzt.

Sei p_i der i-te Weg, der konstruiert wurde, p_j, $1 \leq j < i$ seien die zuvor konstruierten Wege und verbinden jeweils s_j und t_j. Weg p_i ende bei Knoten t. Dann ist $t \in \{t_i, \ldots, t_k\}$. Folgende Prozedur besteht aus einer „Left-First-Tiefensuche“ rückwärts und berechnet einen Weg p, der einen Schnitt mit den gewünschten Eigenschaften induziert.

SCHNITT-PROZEDUR $(p_1, \ldots, p_i)$

1: p sei die eindeutige in t einlaufende Kante
2: $v \leftarrow$ Auslaufknoten dieser Kante.
3: **Solange** $v \neq \{s_1, \ldots, s_k\}$ **führe aus**
4: Sei (v, x) die letzte zu p hinzugefügte Kante und (u, v) die im Uhrzeigersinn erste Kante nach (v, x), die nicht durch p besetzt ist,
5: dann füge (u, v) zu p hinzu.
6: Setze $v := u$.
7: **Ende** „**Solange**“
8: Gib p aus.

Offensichtlich kann die SCHNITT-PROZEDUR in $\mathcal{O}(n)$ realisiert werden. Wir beweisen, dass die Menge der Kanten $\{u, v\}$ aus G, für die u auf p und v rechts von p liegt, einen saturierten bzw. übersaturierten Schnitt in G, D induzieren. Dazu muss natürlich die rechte Seite von p wohldefiniert sein.

Lemma 8.7. *Der Weg p, der von der SCHNITT-PROZEDUR konstruiert wird, enthält keine Kreuzung. Insbesondere sind seine linke und rechte Seite wohldefiniert.*

Beweis. Angenommen p würde sich selbst kreuzen. Dann gibt es einen einfachen Rechtskreis oder einen einfachen Linkskreis auf p mit Kreuzung in einem Knoten v. Da $\overrightarrow{G}$ nach Korollar 8.5 keinen Rechtskreis enthält, gibt es in v einen einfachen Linkskreis, wobei die entsprechenden Kanten von p, die zu v inzident sind, die in Abb. 8.7 gezeigte Konstellation bilden.

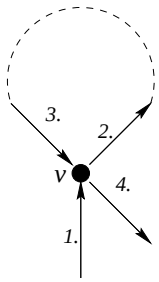


Abbildung 8.7: Illustration zu Lemma 8.7.

Dies ist ein Widerspruch zur Vorgehensweise in Schritt 2 der SCHNITT-PROZEDUR.□

Lemma 8.8. *Sei A die Menge der Kanten $\{u, v\}$ aus G mit u auf p und v rechts von p. Jede Kante $\{u, v\} \in A$ mit u auf p gehört zu $\overrightarrow{G}$, und zwar in der Orientierung (u, v) genau dann, wenn sie von einem der p_j, $1 \leq j < i$, besetzt ist.*

Beweis. Sei $\{u, v\} \in A$ mit u auf p. Falls $\{u, v\}$ von einem der p_j, $1 \leq j < i$ besetzt ist, so ist deren Orientierung (u, v) in $\overrightarrow{G}$, nach Konstruktion von p.

Fall 1: Angenommen $\{u, v\} \in A$ mit u auf p habe Orientierung (u, v) in $\overrightarrow{G}$,und sei nicht durch eines der p_i besetzt.

Betrachte die zu p gehörenden Kanten (x, u), (u, y) inzident zu p, für die $\{u, v\}$ rechts von (x, u) und (u, y) liegt. Die Kante, die bei der Berechnung der $p_1, \ldots, p_i$ unmittelbar vor (u, y) gewählt wurde, ist dann (x, u) oder liegt links von (x, u) und (u, y). Dann ist aber die Wahl von (u, y) ein Widerspruch zur Right-First-Auswahlregel. Siehe Abb. 8.8.

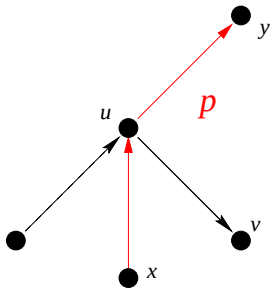


Abbildung 8.8: Illustration zu Fall 1 aus Lemma 8.8.

Fall 2: Betrachte nun eine Kante $\{u, v\} \in A$, mit u auf p, die nicht zu $\overrightarrow{G}$ gehört. Dann können wegen der Right-First Auswahlregel die Kanten (x, u), (u, y) von p, für die $\{u, v\}$ rechts liegt, nicht beide zu dem selben Weg q'_i aus $\overrightarrow{G}$ gehören. Gehöre (x, u) also zu q'_j und (u, y) zu q'_l, $j \neq l$. Dann liegt die Vorgängerkante von (u, y) auf q'_l rechts von $\{u, v\}$ und (u, y), und kann daher von keinem der $p_1, \ldots, p_i$ besetzt sein. Dann muss es eine weitere Kante (z, u) in $\overrightarrow{G}$ geben, die von einem der $p_1, \ldots, p_i$ besetzt ist und links von (x, u) und (u, y) liegt, und eine weitere Kante (u, w), die Nachfolgerkante von (z, u) auf dem entsprechenden Weg q'_r in $\overrightarrow{G}$ ist, und auch links von (x, u), (u, y) liegt. Dann liegt aber die Kante $\{u, v\}$ rechts von q'_r im Widerspruch zur Right-First-Auswahlregel. Siehe Abb. 8.9.

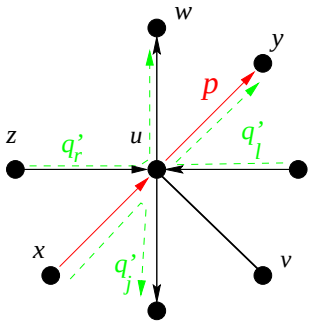


Abbildung 8.9: Illustration zu Fall 2 aus Lemma 8.8.

Lemma 8.9. *Sei $X \subseteq V$ Menge der Knoten rechts von p. Falls p_i ein s_i-t_i-Weg ist, ist X saturiert, ansonsten ist X übersaturiert.*

Beweis. Alle Kanten $\{u, v\}$ mit u auf p und $v \in X$ gehören entweder zu einem Weg p_j, $1 \leq j < i$, mit $s_j \in V \backslash X$ und $t_j \in X$, oder zu einem Weg q'_j, mit $s'_j \in X$ und $t'_j \in V \backslash X$. Wenn p_i ein s_i-t_iWeg ist, dann ist damit

$$\begin{aligned} \operatorname{cap}(X) &= \left| \left\{ \{s_j, t_j\} \colon s_j \in V \backslash X,\ t_j \in X,\ 1 \leq j \leq i \right\} \right| \\ &\quad + \left| \left\{ \{s'_j, t'_j\} \colon s'_j \in X \text{ und } t'_j \in V \backslash X,\ \{s'_j, t'_j\} \notin \left\{ \{s_1, t_1\}, \ldots, \{s_i, t_i\} \right\} \right\} \right| \\ &= \operatorname{dens}(X) \ . \end{aligned}$$

Wenn p_i kein s_i-t_i-Weg ist, so verbindet p_i das Terminal s_i mit einem Terminal $t \in \{t_j;\ i < j \leq k\}$. Da $s_i \in V \backslash X$ und $t_i \in X$ ist, dann gilt

$$\operatorname{cap}(X) \ \leq \ \operatorname{dens}(X) - 1 \ . \qquad \square$$

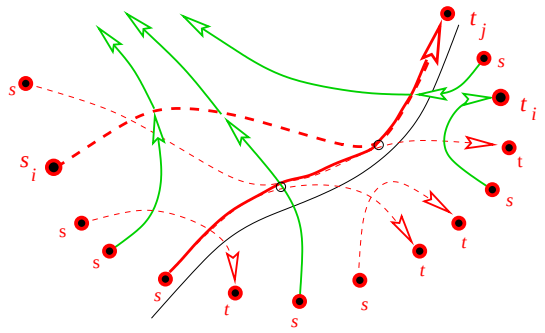


Abbildung 8.10: Illustration der SCHNITT-PROZEDUR. Der Weg p (rot durchgezogen), der von der SCHNITT-PROZEDUR berechnet wird, darunter der durch p induzierte Schnitt, der in diesem Fall übersaturiert ist, da er zusätzlich zu bereits verbundenen Terminalpaaren noch $\{s_i, t_i\}$ trennt. Wege p_i sind rot gestrichelt, Wege q_j grün.

Laufzeit: Mit UNION-FIND kann die RIGHT-FIRST-PROZEDUR zu Schritt 3 in $\mathcal{O}(n)$ realisiert werden.

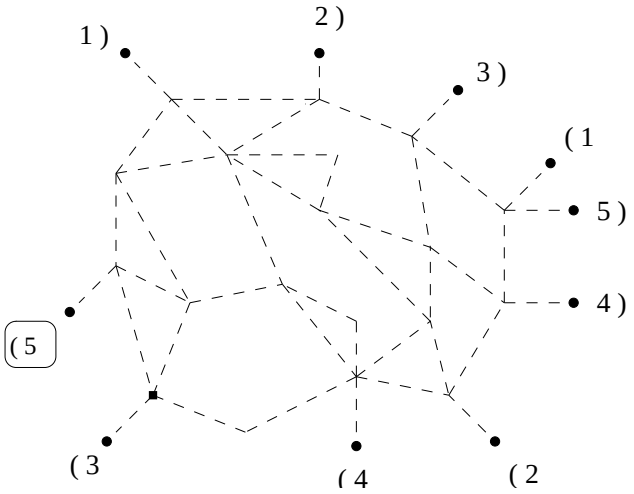


Abbildung 8.11: Ein Problembeispiel für das Problem von Okamura & Seymour und zugehöriges Problembeispiel mit Klammerstruktur bei Wahl von 5 als Startterminal.

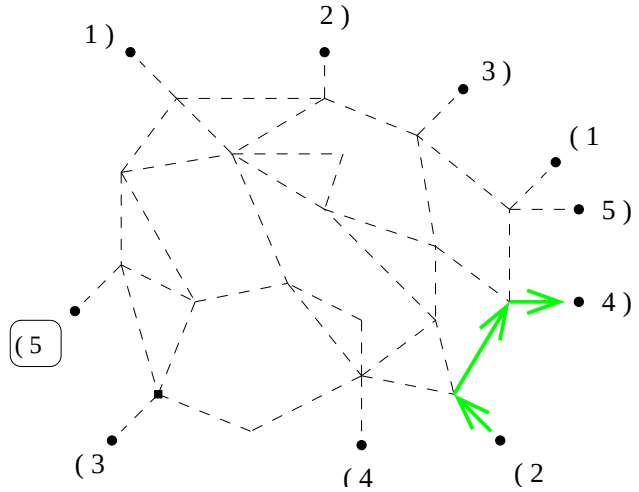


Abbildung 8.12: Der erste Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

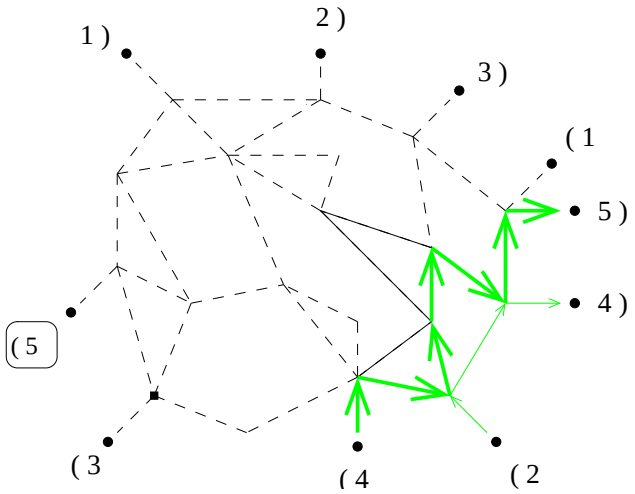


Abbildung 8.13: Der zweite Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

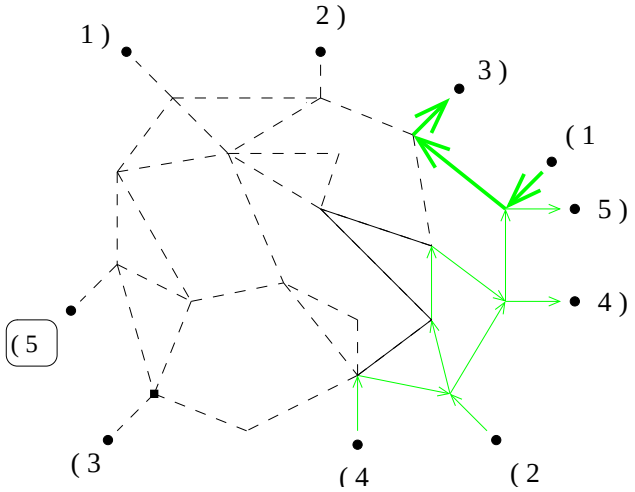


Abbildung 8.14: Der dritte Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

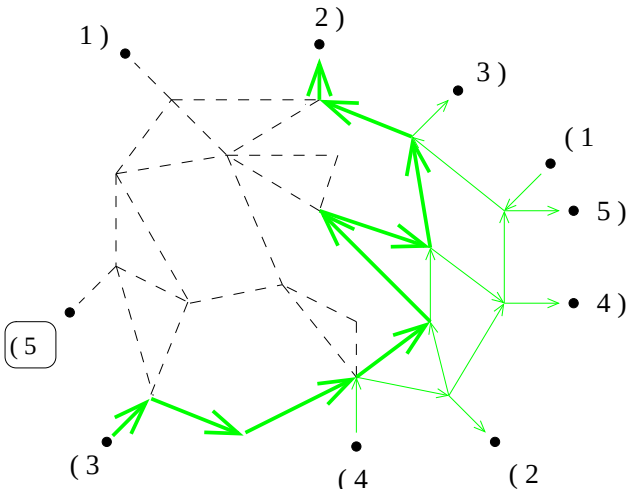


Abbildung 8.15: Der vierte Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

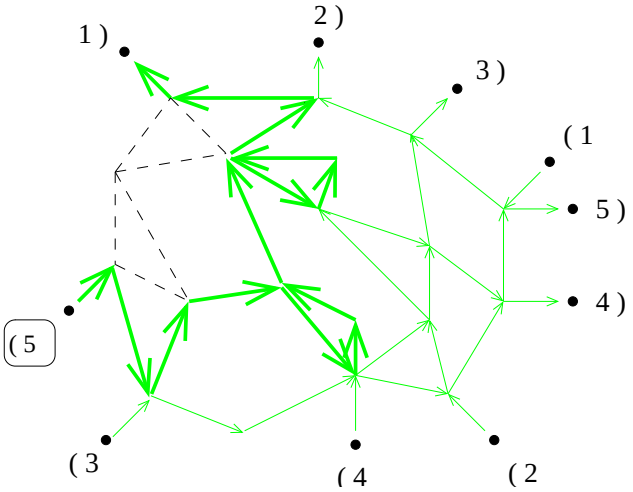


Abbildung 8.16: Der fünfte und letzte Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

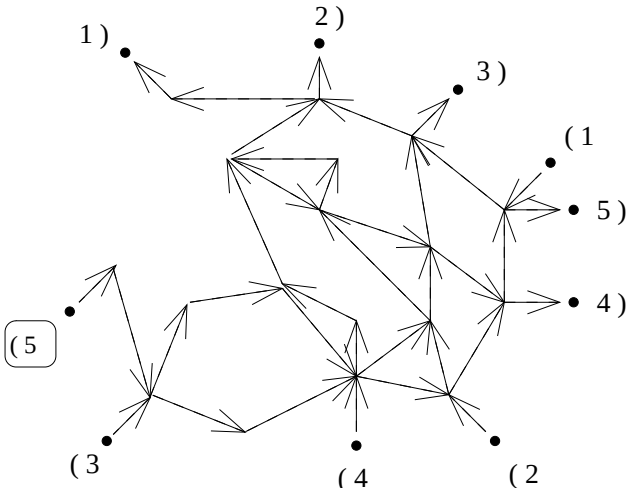


Abbildung 8.17: Der in Schritt 2 berechnete Hilfsgraph $\overrightarrow{G}$, in dem die RIGHT-FIRST-PROZEDUR für G und D arbeitet.

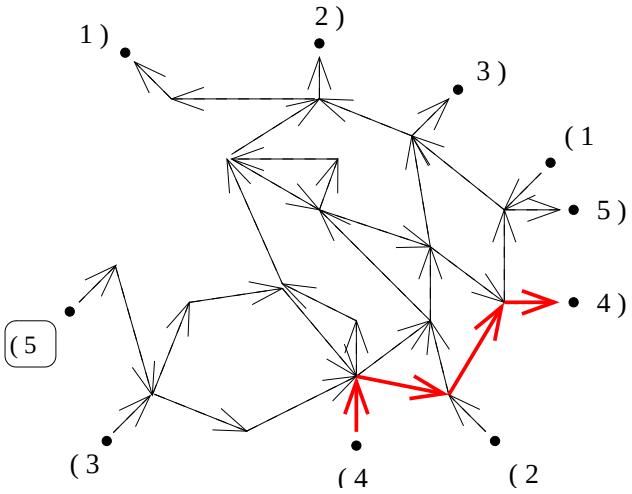


Abbildung 8.18: Der erste Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 4.

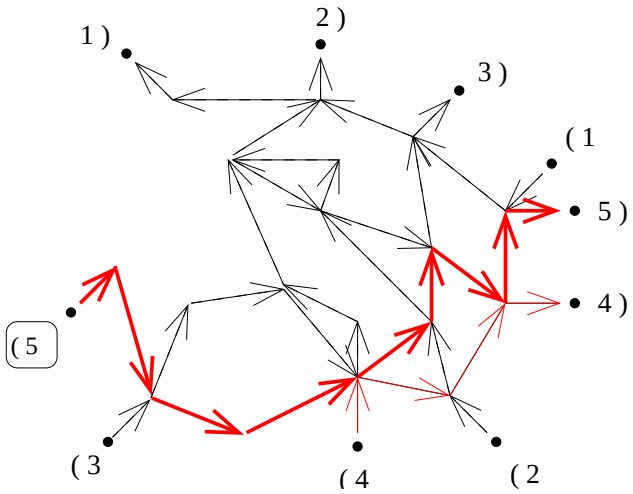


Abbildung 8.19: Der zweite Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 5.

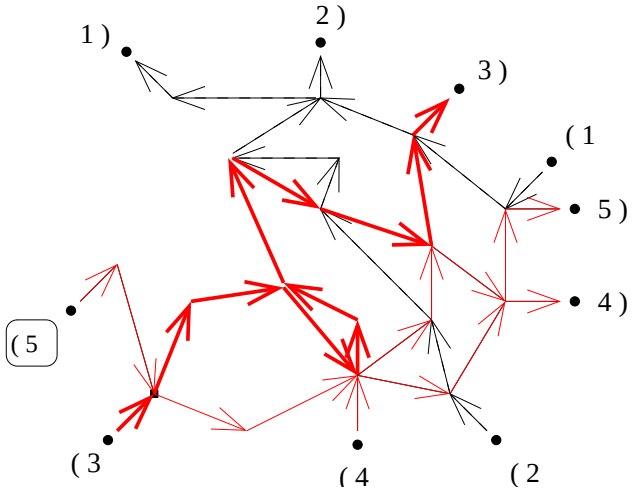


Abbildung 8.20: Der dritte Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 3.

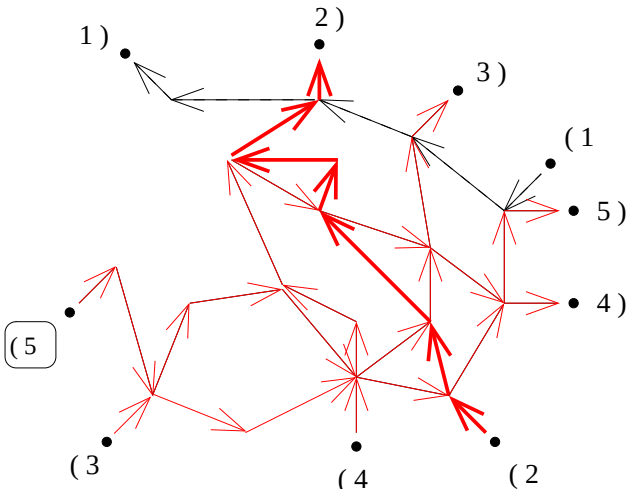


Abbildung 8.21: Der vierte Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 2.

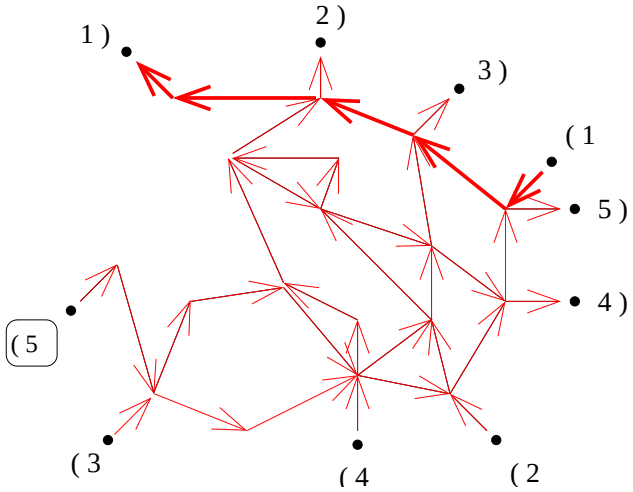


Abbildung 8.22: Der fünfte Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 1.

Literaturliste

Zum Nachlesen von Grundlagenwissen ist [CLR94] ein gutes Buch unter vielen. Siehe auch [Man89] zum Thema Wachstumsraten von Funktionen, [Jun94] zum Thema Graphentheorie, und [GJ79] zum Thema Komplexitätsklassen.

Ein algorithmisch orientiertes Standardwerk zu planaren Graphen ist das Buch [NC88]. [Nis90] schreibt [NC88] fort. [Joh85] ist ein Übersichtsartikel zur Komplexität algorithmischer Probleme auf Graphen und enthält auch einen Abschnitt über planare Graphen.

Literaturangaben zu den Kapiteln der Vorlesung

Kap.	1	Einführung	[Aig84, RSST96]
Kap.	2	Grundlegende Eigenschaften	[Aig84, Kapitel 4]
Kap.	3	Färbung	[Aig84, Kapitel 3]
Kap.	4	Planar Separator Theorem	[LT79]
		(Der in der Vorlesung vorgestellte Beweis ist wesentlich vereinfacht. Die garantierte Maximalgröße eines Separators ist dadurch unwesentlich größer als in der Originalarbeit.)	
Kap.	5	Matching maximalen Gewichts	[LT80]
Kap.	6	Mixed-Max-Cut-Algorithmus	[SWK90]
		Via-Minimierung	[Pin84, KCS88]
Kap.	7	Kantendisjunkter Menger-Algorithmus	[Wei94, Cou97]
		Knotendisjunkter Menger-Algorithmus	[RLWW97]
Kap.	8	Das Okamura-Seymour-Problem	[OS81, WW95]
		Übersichtsartikel zu Wegpackungsproblemen	[Wag93]

Zusätzliche Literaturangaben zu ausgewähltem Übungsstoff

Isomorphie	vergl. [BM76, Abschnitt 1.2]
Einbettung in verschiedene Flächen	[Whi84, Kapitel 5]
Maße für die Nähe zur Planaritt	[Lie96]
Satz von Whitney	[Aig84, Satz 4.7]
Satz von Steinitz	[Grü67, Kapitel 13]
Choosability	[Tho94, Voi93]
Straight line embedding	[Fár48]
Satz von Menger	[Aig84, Satz 4.4 und Folgerung 4.5]
Satz von Vizing	[Aig84, Satz 5.6]
Dualer Graph im Sinne von Whitney	[Aig84, Satz 4.11]

Literaturverzeichnis

[Aig84] Martin Aigner. *Graphentheorie: Entwicklung aus dem 4–Farben–Problem.* Teubner, 1984. Unibib KN: mat 9:ai80/g71.

[BM76] John A. Bondy and Uppaluri R.S. Murty. *Graph theory with applications.* North–Holland, 1976. Unibib KN: mat 9:bo63/g71.

[CLR94] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press and McGraw–Hill, 1994. ISBN 0-262-03141-8, 0-07-013143-0, 0-262-53091-0. Unibib KN: kid 112/c67c, lbs 830/c67.

[Cou97] Laurent Coupry. A simple linear algorithm for the edge-disjoint (s,t)paths problem in undirected planar graphs. Technical report, GREYC, 14032 Université de Caen Cedex, France, 1997.

[Fár48] István Fáry. On straight line representation of planar graphs. *Acta Scientiarum Mathematicarum (Institutum Bolyainum, Universitatis Szegediensis)*, 11:229–233, 1948.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$–Completeness.* W H Freeman & Co Ltd, 1979. Unibib KN: kid 112/g17, lbs 840/g17, y 16053.

[Grü67] Banko Grünbaum. *Convex Polytopes.* Interscience Publishers, a divsion of John Wiley & Sons, 1967. Unibib KN: mat 9:gr91:nb/c66.

[Joh85] David S. Johnson. The $\mathcal{NP}$-Completeness Column: An Ongoing Guide. *J. Algorithms*, 6:434–451, 1985.

[Jun94] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen.* BI-Wissenschaftsverlag, 1994. Unibib KN: kid 114/j96a(3), kid 604/j96, lbs 840/j96.

[KCS88] Y.S. Kuo, T.C. Chen, and W.-K. Shih. Fast algorithm for optimal layer assignment. In *Proceedings of the 25th Design Automation Conference DAC'88*, pages 554–559, 1988.

[Lie96] Annegret Liebers. Methods for Planarizing Graphs—A Survey and Annotated Bibliography. Konstanzer Schriften in Mathematik und Informatik 12, Universität Konstanz, 1996.

[LT79] R.J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36:177–189, 1979.

[LT80] R.J. Lipton and Robert E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9:615–627, 1980.

[Man89] Udi Manber. *Introduction to algorithms: a creative approach.* Addison–Wesley Publishing Company, 1989. Unibib KN: kid 112/m16, lbs 840/m16.

[NC88] Takao Nishizeki and Norishige Chiba. *Planar Graphs: Theory and Algorithms*, volume 32 of *Annals of Discrete Mathematics*. North–Holland, 1988. ISBN 0-444-70212-1. Unibib KN: kid 114/n48.

[Nis90] Takao Nishizeki. Planar graph problems. In Gottfried Tinhofer, Ernst W. Mayr, Hartmut Noltemeier, and Maciej M. Sysło, editors, *Computational Graph Theory*, volume 7 of *Computing Supplementum*, pages 53–68. Springer-Verlag Wien New York, 1990. Unibib KN: kid 2/c61a-7.

[OS81] Haruko Okamura and Paul Seymour. Multicommodity flows in planar graphs. *J. of Combinatorial Theory, Series B*, 31:75–81, 1981.

[Pin84] R.Y. Pinter. Optimal layer assignment for interconnect. *Journal of VLSI and Computer Systems*, 1:123–137, 1984.

[RLWW97] Heike Ripphausen-Lipa, Dorothea Wagner, and Karsten Weihe. The Vertex-Disjoint Menger Problem in Planar Graphs. *SIAM J. Comput.*, 26:331–349, 1997.

[RSST96] Neil Robertson, Daniel P. Sanders, Paul Seymour, and Robin Thomas. A new proof of the four-colour theorem. *ERA Amer. Math. Soc.*, 2(1):17–25, 1996. Available electronically at http://www.ams.org/era/1996-02-01/.

[SWK90] W.-K. Shih, S. Wu, and Y.S. Kuo. Unifying maximum cut and minimum cut of a planar graph. *IEEE Transactions on Computers*, C-39:694–697, 1990.

[Tho94] Carsten Thomassen. Every Planar Graph is 5-Choosable. *J. of Combinatorial Theory, Series B*, 62:180–181, 1994.

[Voi93] Margit Voigt. List colourings of planar graphs. *Discrete Mathematics*, 120: 215–219, 1993.

[Wag93] Dorothea Wagner. Simple Algorithms for Steiner Trees and Paths Packing Problems in Planar Graphs. *CWI Quarterly*, 6:219–240, 1993.

[Wei94] Karsten Weihe. Edge–disjoint $(\mathtt{s},\mathtt{t})$–paths in undirected planar graphs in linear time. In Jan v. Leeuwen, editor, *Second European Symposium on Algorithms, ESA'94*, pages 130–140. Springer-Verlag, Lecture Notes in Computer Science, vol. 855, 1994.

[Whi84] Arthur T. White. *Graphs, Groups and Surfaces*, volume 8 of *North-Holland Mathematics Studies*. North–Holland, 1984. Unibib KN: mat 9:wh48:fe/g71(2).

[WW95] Dorothea Wagner and Karsten Weihe. A linear time algorithm for edge-disjoint paths in planar graphs. *Combinatorica*, 15:135–150, 1995.