technische universität
dortmund

Masterarbeit

Effiziente Berechnung von K_5-Minoren in Graphen

Julian Sauer

30. September 2019

Betreuer:
Prof. Dr. Petra Mutzel
Prof. Dr. Jens Schmidt

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

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im Rahmen dieser Masterarbeit wird ein Algorithmus erklärt und implementiert, der entscheiden kann, ob ein Graph K_5-Minor-frei ist. Er basiert auf einem von Kezdy und McGuinness [14] vorgestellten Algorithmus, der quadratische Laufzeitkomplexität besitzt.

1.1 Motivation und Hintergrund

Durch die Berechnung von K_5-Minoren kann entschieden werden, ob ein Graph K_5-Minor-frei ist. Das ist insofern interessant, als dass einige Algorithmen effizienter auf K_5-Minor-freien Graphen ausgeführt werden können. So ist die Berechnung eines *maximalen Schnittes* (*Max-Cut Problem*) - also das Aufteilen der Knoten eines Graphen in zwei Mengen, sodass die Kanten zwischen diesen beiden Mengen in Summe ein maximales Gewicht besitzen - NP-schwer [12]. Allerdings wird etwa in [2] gezeigt, dass es für Graphen ohne K_5-Minor in Polynomialzeit gelöst werden kann. Viele kombinatorische Optimierungsprobleme wie quadratische 0-1 Probleme können als Max-Cut Probleme formuliert werden [3], sodass K_5-Minor-freie Probleminstanzen deutlich effizienter gelöst werden können.

In [11] transformieren Jünger et al. Ising Spin Glass Probleme zu Max-Cut Problemen, um mit Hilfe von einem Branch-and-Cut Algorithmus eine optimale Lösung zu finden. Diese kann mit den heuristischen Lösungen, die etwa auf dem Quantencomputer *D-Wave 2000Q* berechnet werden, verglichen werden. Dadurch kann beispielsweise bewertet werden, ob für die schnellere Laufzeit des Quantencomputers die heuristische Lösung tragbar ist. Innerhalb des exakten Verfahrens stellen die K_5-Minoren die Schwierigkeit des Problems dar. Deshalb kann das Problem einerseits für K_5-Minor-freie Graphen effizient gelöst werden. Andererseits können einer oder mehrere gefundene K_5-Minoren benutzt werden, um innerhalb des linearen Programms den Suchraum zu beschränken und die Berechnung zu beschleunigen.

1.2 Aufbau der Arbeit

Zunächst werden einige Definition gegeben, die anschließend benutzt werden, um den Algorithmus von Kezdy und McGuinness [14] vorzustellen. Anschließend wird ein auf Wagner [18] zurückzuführendes Strukturtheorem der beiden Autoren genauer betrachtet. Für den Fall, dass ein Graph K_5-Minor-frei ist, kann dadurch ein Zertifikat erzeugt werden, über dass die getroffene Aussage verifiziert werden kann. Letztlich beschäftigt sich die Arbeit mit der Implementierung des Algorithmus im *Open Graph Drawing Framework* sowie einer experimentellen Analyse dieser Implementierung.

Kapitel 2

Definitionen

Vorab werden einige Definitionen und Notationen festgelegt, die im Verlauf der Arbeit verwendet werden. Der Algorithmus arbeitet mit einem ungerichteten Graph $G = (V, E)$, wobei E die Menge der Kanten und V die Knotenmenge ist. Eine Kante $e \in E$, die zwei Knoten u und v verbindet, wird durch $e = (u, v)$ angegeben. Ein Pfad $P(u, v)$ verbindet zwei Knoten u und v über eine Folge von Knoten, die paarweise adjazent sind. Für $P(u, v)$ werden u und v als *Endpunkte*, die übrigen Knoten des Pfades als *innere Knoten* bezeichnet. Falls nicht anders angegeben, wird für jedes Knotenpaar in einem Graphen erwartet, dass es durch maximal eine Kante verbunden wird – Mehrfachkanten sind nicht erlaubt. Genauso sind die betrachteten Graphen frei von Schleifen der Form $e = (v, v)$ mit $e \in E$ und $v \in V$.

Die Menge adjazenter Knoten eines Knotens v wird durch $N(v)$ angegeben. Als *Knotengrad* von v wird die Kardinalität von $N(v)$ bezeichnet. Bei einer *Kantenkontraktion* einer Kante $e = (u, v)$ wird ein neuer Knoten w hinzugefügt, sodass $N(w) = N(u) \cup N(v)$ gilt und anschließend e aus dem Graphen entfernt. In Abb. 2.1 ist das Vorgehen skizziert. Analog kann, wie in Abb. 2.2 gezeigt, ein Pfad kontrahiert werden, indem nacheinander je eine Kante des Pfades kontrahiert wird. Außerdem kann jede Menge von Knoten kontrahiert werden, solange sie einen zusammenhängenden Teilgraph bildet.

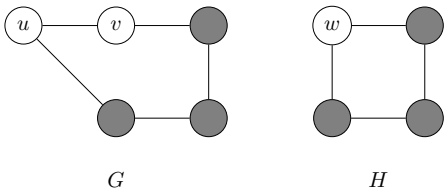


Abbildung 2.1: Die Kante, die u und v in G verbindet, wird kontrahiert, sodass sie in H durch den neuen Knoten w ersetzt wird.

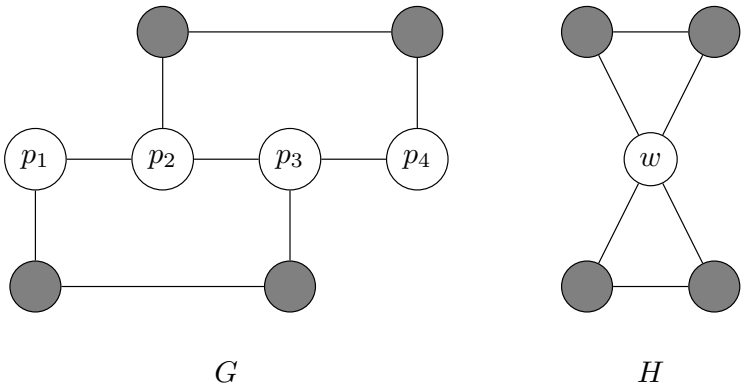


Abbildung 2.2: Der Pfad von p_1 bis p_4 wird kontrahiert. Der neue Knoten w in H enthält alle Nachbarn der Pfadknoten in G.

Ein *Minor* H eines Graphen G bezeichnet einen Graph, der isomorph zu G ist, nachdem eine beliebige Folge an Operationen von Kantenkontraktionen, Kantenentfernungen und Knotenentfernungen durchgeführt wurde. Ein Beispiel dazu findet sich in Abb. 2.3. Jeder Graph ist sein eigener Minor, genauso ist jeder Teilgraph ein gültiger Minor. Dass H ein Minor von G ist, wird dargestellt durch $H \prec_M G$. Im Fließtext wird die Formulierung G *enthält* einen H-Minor verwendet, um $H \prec_M G$ auszudrücken. Ist V eine Menge von Knoten, die einen zusammenhängenden Teilgraphen formen, der durch Kontraktionen durch einen neuen Knoten w ersetzt wurde, dann bezeichnet das *Branch-Set* von w die Menge V. In Abb. 2.3 ist beispielsweise das Branch-Set von g die Menge $\{a, b, c\}$, für f in H_2 ist es $\{f\}$ in G.

Für die Knotenmenge $U \subseteq V$ bezeichnet $G - U$ den Teilgraph, der entsteht, wenn alle Knoten aus U mit ihren inzidenten Kanten aus G entfernt werden. Besteht eine Knotenmenge $\{u\}$ aus nur einem Knoten, wird statt $G - \{u\}$ auch $G - u$ geschrieben. Durch $G(U)$ für $U \subseteq V$ wird der Teilgraph von G, dargestellt, der alle Knoten aus U enthält sowie alle Kanten $e = (u, v) \in E$, für die gilt $u, v \in U$. Für einzelne Knoten u wird statt $G(\{u\})$ vereinfacht $G(u)$ geschrieben. Eine *Subdivision* H eines Graphen G enthält alle Knoten aus $G = (V, E)$, sodass jedes inzidente Knotenpaar $\{u, v\} \subseteq V$ in H durch einen Pfad verbunden ist.

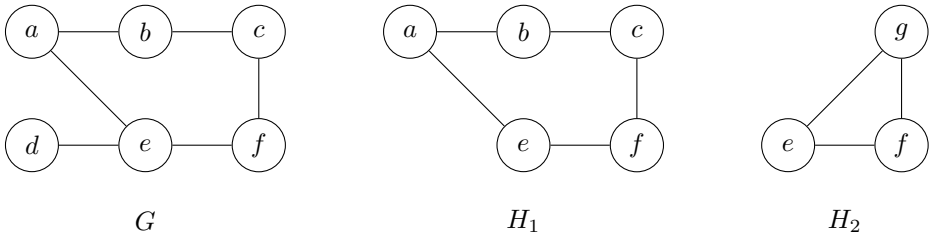


Abbildung 2.3: Ein Graph G mit seinen Minoren H_1 und H_2. Um H_1 zu erhalten, wurde in G die Kante (d, e) und anschließend der Knoten d entfernt. Für H_2 wurden außerdem der Pfad $P(a, c)$ kontrahiert.

Ein Graph wird als *planar* bezeichnet, wenn er sich so in der Ebene einbetten lässt, dass sich keine Kanten kreuzen. Ein K_5 (s. Abb. 2.4) ist ein Graph bestehend aus einer Clique von fünf Knoten. Ein $K_{3,3}$ (s. Abb. 2.5) ist ein vollständig bipartiter Graph mit sechs Knoten, sodass jede Bipartition drei Knoten enthält. Er lässt sich also in zwei Knotenmengen unterteilen (im Folgenden als rote und blaue Menge bezeichnet), sodass alle Knoten der einen Menge paarweise adjazent zu allen Knoten der anderen Menge sind. Nach dem Satz von Kuratowski [13] ist ein Graph genau dann planar, wenn er keine K_5- oder $K_{3,3}$-Subdivision als Teilgraph beinhaltet. Eine alternative Formulierung von Wagner [18] sagt aus, dass ein Graph genau dann planar ist, wenn er keinen K_5- oder $K_{3,3}$-Minor enthält [8].

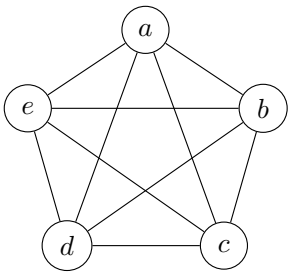


Abbildung 2.4: Der Graph K_5.

Als nächstes wird ein $K_{3,3}$ genauer betrachtet. Sei dessen rote Knotenmenge $R = \{a, b, c\}$ und blaue $B = \{x, y, z\}$. Diese sechs Knoten werden in einer $K_{3,3}$-Subdivision H (analog in einem $K_{3,3}$-Minor) *Branch-Ends* genannt und zeichnen sich dadurch aus, dass sie als einzige Knoten in H den Grad 3 haben. Ein *Branch-Path* in H ist ein Pfad, der zwei Branch-Ends verbindet, beispielsweise $P(a, x)$. Ein *Branch-Fan* bezieht sich immer auf eines der Branch-Ends und wird z. B. für a als $F(a)$ angegeben. Bezeichnet werden dadurch alle Pfade in H, die zu einem anderen Branch-End führen – für a also die Pfade $P(a, x)$, $P(a, y)$ und $P(a, z)$. Zwei Branch-Paths sind *parallel*, wenn ihre Endpunkte vier unterschiedlichen Branch-Ends sind.

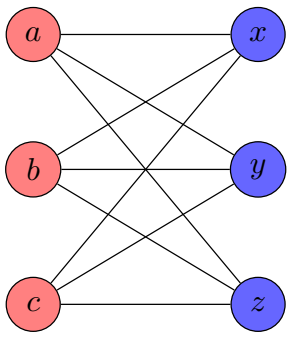


Abbildung 2.5: Der Graph $K_{3,3}$.

Der Graph W bezeichnet einen speziellen Graph, der aus acht Knoten besteht. Seine äußeren Kanten bilden einen Kreis, außerdem sind die Knoten jeweils adjazent zu den

gegenüberliegenden. Eine Darstellung findet sich links in Abb. 2.6. Er enthält einen $K_{3,3}$ als Minor (in der Abbildung rechts angedeutet), jedoch keinen K_5. Als M wird der Graph bezeichnet, der links in Abb. 2.7 zu sehen ist. Er enthält einen $K_{3,3}$-Minor, wenn beispielsweise die Kanten $\{(b,d),(b,e)\}$ entfernt und die Kante (a,b) kontrahiert wird. Er ist aber vor allem interessant, weil er, wie in Abb. 2.7 zu sehen, neben einem $K_{3,3}$-Minor auch einen K_5-Minor enthält.

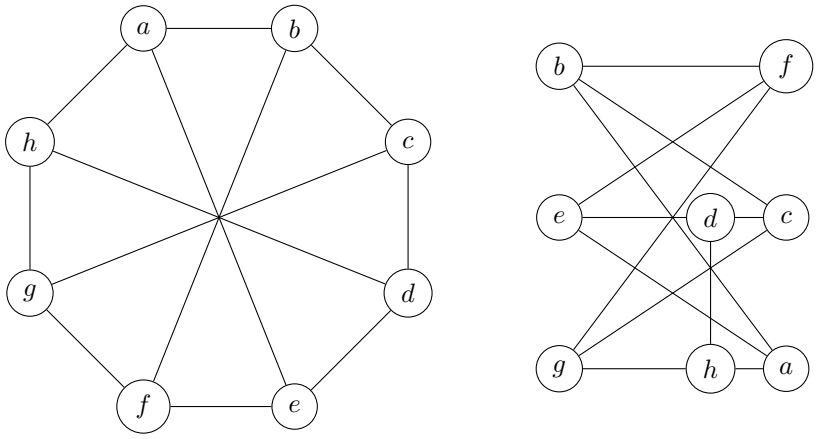
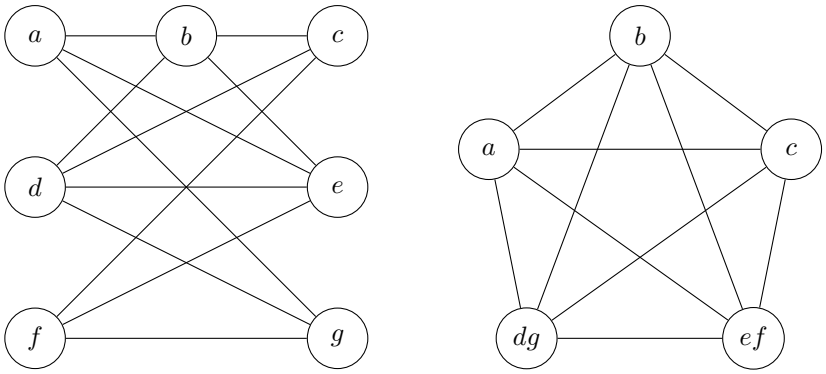


Abbildung 2.6: Der Graph W, links in seiner üblichen Darstellung, rechts mit angedeutetem $K_{3,3}$-Minor.



M K_5-Minor

Abbildung 2.7: Der Graph M sowie ein K_5-Minor aus M.

Als *i-Separator* wird eine Menge S bestehend aus i Knoten in einem zusammenhängenden Graphen G bezeichnet, sodass $G - S$ nicht mehr zusammenhängend ist. Ein Graph ist *k-zusammenhängend*, falls er keinen $(k-1)$-Separator hat. Ein *(i,j)-Separator* ist ein i-Separator, sodass $G - S$ aus mindestens j Zusammenhangskomponenten besteht. In Abb. 2.8 wird ein $(3,3)$-Separator im oberen Graphen gezeigt, der aus den Knoten $C = \{c_1, c_2, c_3\}$ besteht. Der Graph wird durch $G - C$ zunächst in j Zusammenhangskomponenten $Z_1, ..., Z_j$ geteilt. Dann wird für jedes Z_k mit $1 \leq k \leq j$ ein neuer Graph

A_k erzeugt, der aus dem Teilgraphen $Z_k \cup C$ besteht – die Knoten von C in A_k werden paarweise durch Kanten verbunden, falls sie noch nicht adjazent zueinander sind. Jeder der resultierenden Graphen wird als *augmentierte Komponente* des Eingabegraphen G bezeichnet, die durch den Separator C definiert wird.

Ist V eine Menge von Knoten oder Kanten, dann bezeichnet $|V|$ die Kardinalität von V. Sei A_1 ein Graph mit einer Clique C_1 und A_2 ein Graph mit einer Clique C_2, sodass $|C_1| = |C_2| = i$ ist. Dann bezeichnet eine Cliquen-Summe das paarweise Zusammenfügen von je einem Knoten $c_1 \in C_1$ mit einem $c_2 \in C_2$, sodass ein neuer Graph G entsteht, der sowohl einen A_1-, als auch einen A_2-Minor enthält. Innerhalb einer solchen Operation dürfen zusätzlich beliebige Kanten in G gelöscht werden, die zwei Knoten in der Clique verbinden. Dadurch ist es möglich, einen Graphen G durch einen Separator in augmentierte Komponente aufzuteilen und anschließend durch eine Cliquen-Summe wieder einen zu G isomorphen Graph zu erhalten. Dieses Vorgehen wird in Abb. 2.8 gezeigt, dabei wird der obere Graph in die unten abgebildeten augmentierten Komponenten aufgeteilt bzw. von unten nach oben werden drei Graphen durch eine Cliquen-Summe zusammengefügt.

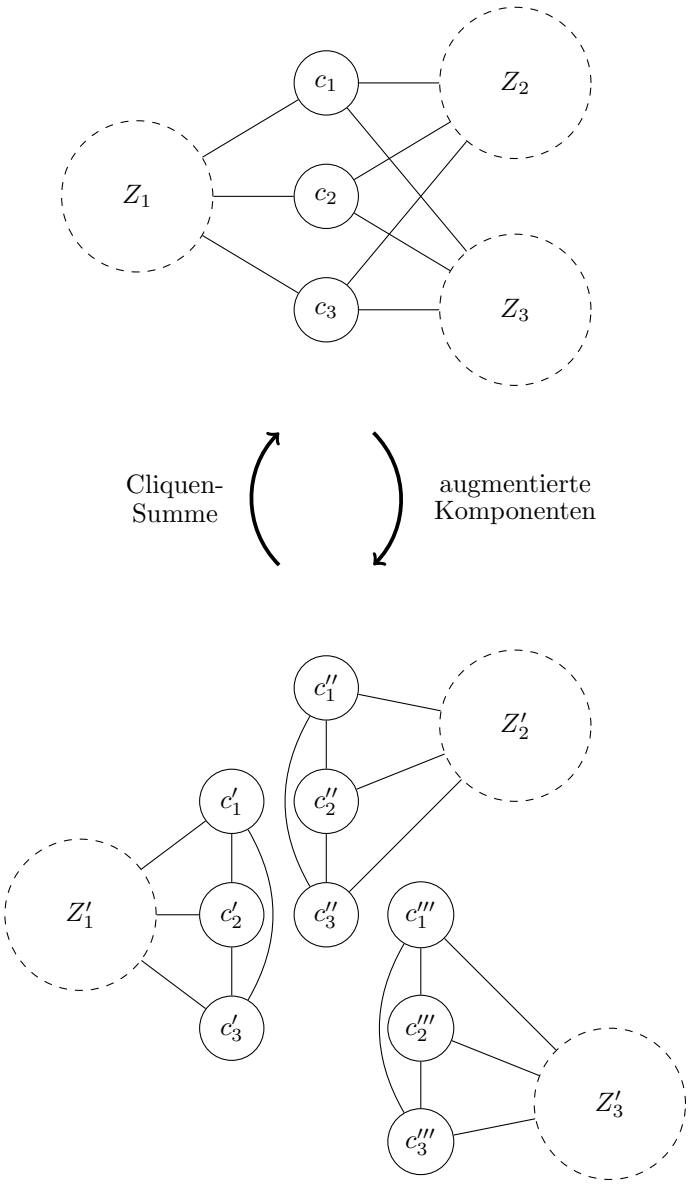


Abbildung 2.8: Der obere Graph wird in drei augmentierte Komponenten durch den $(3,3)$-Separator $C = \{c_1, c_2, c_3\}$ geteilt. Alle Z_i und Z_i' stellen Teilgraphen dar, die zur Übersicht zu einem Knoten zusammengefügt wurden. Aus den drei unteren Graphen kann durch die Cliquen-Summe der Cliquen $\{c_1', c_2', c_3'\}$ sowie $\{c_1'', c_2'', c_3''\}$ und $\{c_1''', c_2''', c_3'''\}$ der oberen Graphen erzeugt werden. Während der Cliquen-Summen Operation dürfen beliebige Kanten, die die Knoten in den Cliquen verbinden, gelöscht werden.

Kapitel 3

Algorithmus von Kezdy und McGuinness

Die Arbeit basiert auf dem in [14] vorgestellten sequenziellen Algorithmus von Kezdy und McGuinness, der im Folgenden erklärt wird. Als Eingabe wird ein ungerichteter Graph ohne Mehrfachkanten und Schleifen erwartet, ausgegeben wird, ob der Graph einen K_5-Minor hat. Für den Fall, dass einer gefunden wurde, kann zusätzlich ausgegeben werden, welche Knoten den Minor formen. Die Laufzeit liegt in $\mathcal{O}(n^2)$.

Planaritätstests können bereits in linearer Laufzeit entscheiden, ob ein Graph einen K_5- oder $K_{3,3}$-Minor enthält [5]. Es muss folglich der Fall behandelt werden, in dem der Test stoppt, weil er einen $K_{3,3}$-Minor gefunden hat. Denn dann kann zunächst keine Aussage darüber getroffen werden, ob zusätzlich ein K_5-Minor enthalten ist. Als Lösung bildet der Algorithmus von Kezdy und McGuinness in einem 3-zusammenhängenden Graphen solange augmentierte Komponenten mit $(3, 3)$-Separatoren, die eine Teilmenge der gefundenen $K_{3,3}$-Minoren sind, bis entweder alle Komponenten planar bzw. isomorph zu W sind oder aber ein K_5-Minor gefunden wurde.

3.1 Behandlung von $K_{3,3}$-Minoren

Die folgenden Theoreme, Lemmata und meisten Beweise sind [14] entnommen.

Um das zentrale Theorem aus [14], welches den $K_{3,3}$-Minor untersucht, zu erklären, wird zunächst die Gültigkeit augmentierter Komponenten behandelt:

3.1.1 Theorem ([14]). *Für $k \geq 3$: Sei G ein k-zusammenhängender Graph und C ein k-Separator in G mit $k \geq 3$. Jede durch C definierte augmentierte Komponente ist ein Minor von G genau dann, wenn es entweder mindestens k Komponenten sind oder mindestens $k - 1$, von denen mindestens zwei jeweils aus mehr als einem Knoten bestehen.*

Beweis. Seien $c_1, c_2, ..., c_k$ die Knoten von C und $Z = \{Z_1, Z_2, ..., Z_k\}$ beziehungsweise $Z = \{Z_1, Z_2, ..., Z_{k-1}\}$ die Zusammenhangskomponenten, die durch $G - C$ entstehen. Die augmentierten Komponenten der Zusammenhangskomponenten seien $A_1, A_2, ..., A_k$ bzw. $A_1, A_2, ..., A_{k-1}$. Betrachtet wird eine beliebige dieser augmentierten Komponenten A_i. Der Definition der augmentierten Komponenten nach finden sich bereits alle Knoten von A_i in G wieder, da Z_i ein Teilgraph von G ist. Weiterhin enthält G mindestens alle Kanten in $A_i - C$ sowie alle Kanten (u, v) mit $u \in Z_i, v \in C$. Jedoch bilden in A_i die Knoten von C eine Clique, es existieren also ggf. Kanten zwischen den Knoten von C in A_i, die es nicht in G gibt. Es bleibt zu zeigen, dass die Kanten, die für diese Clique in A_i nötig sind, durch Kantenkontraktionen in G erzeugt werden können. Dadurch, dass G k-zusammenhängend ist, besitzt jede Zusammenhangskomponente von $G-C$ Kanten zu allen $c_1, c_2, ..., c_k$. Würde eine Kante zu einem Knoten c_j mit $1 \leq j \leq k$ fehlen, wäre ein $(k-1)$-Separator bestehend aus $C - c_j$ möglich, was im Widerspruch zu dem k-Zusammenhang stehen würde. Das Theorem unterscheided nun zwei Fälle, um die fehlenden Kanten bereitstellen zu können:

1. Es existieren k Zusammenhangskomponenten. Wird A_i betrachtet, kommt der Teilgraph $G - Z_i$ infrage, um durch Kantenkontraktionen die fehlenden Kanten für die Clique von C in A_i zu erzeugen. Um die Kanten von C in A_i in G zu erzeugen, kann zunächst Z_1 zu einem Knoten z_1 kontrahiert werden. Er ist adjazent zu allen Knoten von C. Wird (z_1, c_1) kontrahiert, ist folglich c_1 adjazent zu allen Knoten in $C - \{c_1\}$ Dies kann analog für alle Knoten in C und den entsprechenden Zusammenhangskomponenten durchgeführt werden, außer für c_i, da A_i der gesuchte Minor ist. Allerdings ist c_i aufgrund des k-Zusammenhangs mit allen anderen Zusammenhangskomponenten verbunden, sodass C nach den beschriebenen Kontraktionen eine Clique bildet.

2. Es existieren $k-1$ Komponenten von denen mindestens zwei aus mehr als einem Knoten bestehen. Analog zum vorherigen Fall können die Pfade zwischen den Knoten von C und den Zusammenhangskomponenten Z kontrahiert werden. Im Vergleich zum vorherigen Fall liegt jedoch eine Zusammenhangskomponente weniger vor. Allerdings gibt es mindestens eine Zusammenhangskomponente $Z_j \in Z - Z_i$, die aus zwei oder mehr Knoten besteht. Da der Graph k-zusammenhängend ist, sind mindestens zwei der Knoten von Z_j über disjunkte Pfade mit allen Knoten in C verbunden. Seien $u, v \in Z_j$ zwei dieser Knoten. Es können zwei Knoten $c_u, c_v \in C$ gewählt werden und anschließend die Pfade $P(u, c_u)$ und $P(v, c_v)$ kontrahiert werden. Dadurch können zwei statt nur einer Kante erzeugt werden, die inzident zu zwei Knoten in C sind. Werden außerdem alle übrigen Kanten wie im ersten Fall beschrieben, erzeugt, dann sind die Knoten von C im resultierenden Minor A_i eine Clique.

Ein Beispiel, in dem die augmentierten Komponenten keine Minoren sind, findet sich in Abb. 3.1. □

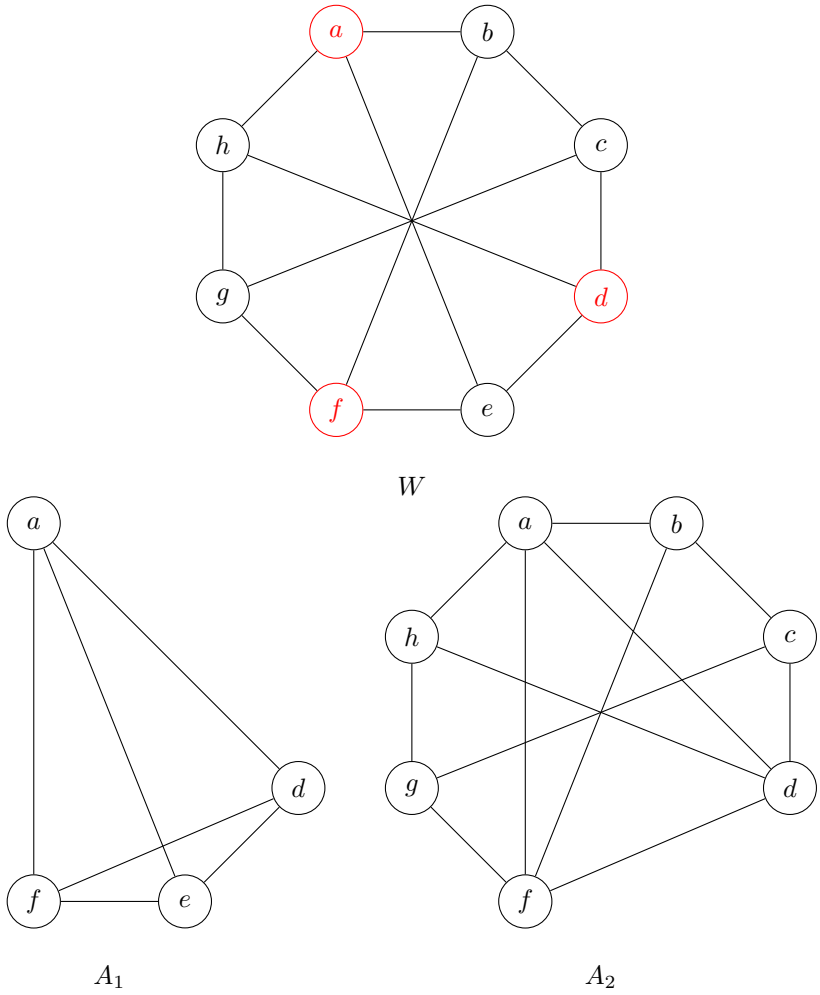


Abbildung 3.1: Gegenbeispiel zu Theorem 3.1.1 für $k = 3$ mit $C = \{a, d, f\}$ in W, wodurch die $k-1$ Komponenten A_1 und A_2 entstehen Es entstehen durch $W - C$ zwei Zusammenhangskomponenten, die die Knoten $\{e\}$ und $\{b, c, g, h\}$ enthalten. Entgegen der Voraussetzungen des Theorems gibt es daher nicht genügend Zusammenhangskomponenten bzw. keine zwei Zusammenhangskomponenten, die je mehr als zwei Knoten enthalten: Die Komponente A_1 ist zwar ein gültiger Minor, da sie etwa durch die Kontraktionen der Kanten $(a, b), (a, h), (f, g)$ und (c, d) aus W erzeugt werden kann. A_2 dagegen kann aber nicht durch Kontraktionen aus W erzeugt werden – wird beispielsweise die Kante (d, e) in W kontrahiert, fehlt die Kante (a, f) in A_2. Analog kann e mit keiner seiner inzidenten Kanten kontrahiert werden, um einen zu A_2 isomorphen Graphen zu erhalten.

Als nächstes stellen Kezdy und McGuinness fest, dass ein Graph in augmentierte Komponenten zerlegt werden kann, wenn ein $(3, 3)$-Separators in ihm gefunden wurde:

3.1.2 Theorem ([14]). *Sei G ein 3-zusammenhängender Graph mit einem $(3, 3)$-Separator C. G hat einen K_5-Minor genau dann, wenn eine der durch C definierten augmentierten Komponenten einen K_5-Minor enthält.*

Beweis ([14]). Zunächst kann festgestellt werden, dass, falls eine der augmentierten Komponenten einen K_5-Minor enthält, dieser laut Theorem 3.1.1 auch ein Minor von G ist. Es bleibt zu zeigen, dass sich ein K_5-Minor nicht auf zwei augmentierte Komponenten erstreckt, sondern sich ausschließlich in einer befindet. Angenommen es gilt $K_5 \prec_M G$ und zwei der Branch-Sets, die den K_5-Minor bilden, befinden sich jeweils vollständig in unterschiedlichen Zusammenhangskomponenten. In diesem Fall wäre C ein 3-Separator in dem gefundenen Minor, was im Widerspruch zu dem 4-Zusammenhang des K_5 steht. □

Das zentrale Theorem ist darauf zurückzuführen, dass jeder Graph ohne K_5-Minor durch Cliquen-Summen von Teilgraphen, die planar oder isomorph zu W sind, gebildet werden kann [18]. Eine genauere Betrachtung findet in Kapitel 4 statt. Es kann benutzt werden, um einen Graph rekursiv in augmentierte Komponenten aufzuteilen, bis der Planaritätstest entweder einen K_5-Minor findet oder alle augmentierten Komponenten planar bzw. isomorph zu W sind.

3.1.3 Theorem ([14]). *Sei G ein 3-zusammenhängender Graph, der ein $K_{3,3}$-Subdivision S enthält, dessen Branch-Ends in die rote Knotenmenge $R = \{a, b, c\}$ und blaue $B = \{x, y, z\}$ unterteilt sind. Mindestens eine der folgenden Bedingungen trifft auf G zu:*

1. *G enthält einen K_5-Minor.*
2. *G ist isomorph zu W.*
3. *$\{a, b, c\}$ bilden einen 3-Separator, sodass $\{x, y, z\}$ in separaten Komponenten liegen.*
4. *$\{x, y, z\}$ bilden einen 3-Separator, sodass $\{a, b, c\}$ in separaten Komponenten liegen.*

Durch die Theoreme 3.1.1 und 3.1.2 wurde gezeigt, dass der Graph in den Fällen 3 und 4 in augmentierte Komponenten zerlegt und darauf ein Planaritätstest ausgeführt werden kann. Anschließend stellen die Autoren einige Lemmata auf, mit denen untersucht wird, ob S einen K_5-Minor enthält – also ob Bedingung 1 zutrifft.

3.1.4 Lemma ([14]). *Sei G ein 3-zusammenhängender Graph und S ein $K_{3,3}$-Subdivision in G. Enthält G drei Pfade von einem Knoten $w \in G - S$ zu drei nicht im selben Branch-Fan liegenden Knoten in S, enthält G einen K_5-Minor.*

Beweis ([14]). Seien t, u, v die drei Endpunkte der Pfade in S. Mindestens einer von ihnen ist ein innerer Knoten, da sonst alle im selben Branch-Fan liegen würden. Sei o. B. d. A. t ein solcher innerer Knoten auf dem Pfad $P(a, x)$. Folglich können u und v nicht beide in $F(a)$ oder $F(x)$ liegen, sonst lägen alle drei im gleichen Branch-Fan. Es wird unterschieden, wo genau die drei Knoten in S liegen – die Fälle sind in den Abbildungen 3.2, 3.3 und 3.4 skizziert.

1. u und v sind nicht im gleichen Branch-Fan wie t. Dann müssen u und v ebenfalls innere Knoten sein, im Beispiel auf den Pfaden $P(y, b)$ bzw. $P(z, c)$. Es kann ein M-Minor durch folgende Kontraktionen erzeugt werden: u mit einem der roten und v mit einem der blauen Knoten (analog u mit blau und v mit rot) sowie $P(w, t)$.

2. u oder v liegen auf $P(a, x)$. Sei o. B. d. A. $u \in P(a, x)$. Da t ebenfalls in diesem Pfad liegt, gilt $\{t, u\} \in F(a) \cup F(x)$, sodass v nicht in diesen beiden Branch-Fans liegen kann. Es können t und v getauscht werden, sodass eine Reduktion auf Fall 1 erreicht wird.

3. Entweder u oder v liegen im gleichen Branch-Fan wie t. Sei o. B. d. A. $u \in F(x) - P(a, x)$, im Beispiel auf dem Pfad $P(b, x)$. Es gilt $\{t, u\} \in F(x)$, weshalb v in einem anderen Branch Fan sein muss. Da alle roten Knoten in $F(x)$ liegen, gilt konkreter $v \in (F(y) \cup F(z)) - \{a, b, c\}$ Es können $P(b, u)$ kontrahiert werden sowie je nach Fall entweder $P(v, y)$ oder $P(v, z)$. Wird $P(w, t)$ ebenfalls kontrahiert, entsteht erneut ein M-Minor. □

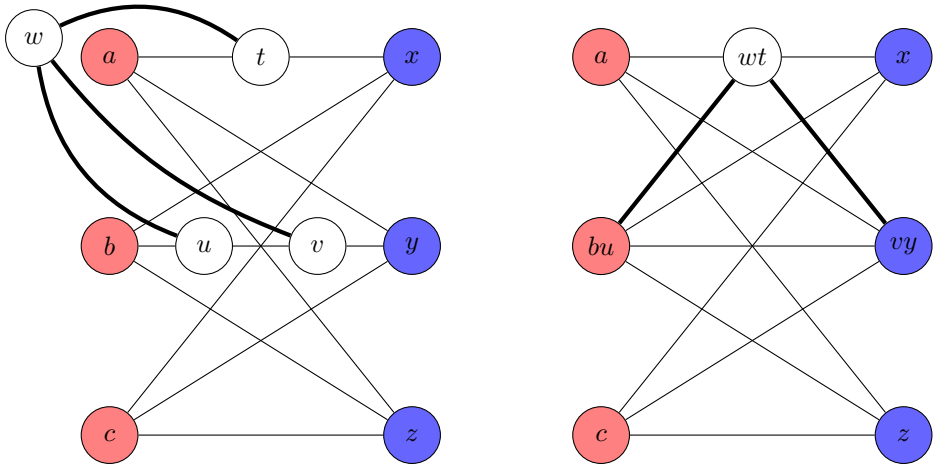


Abbildung 3.2: Beispiel zum ersten Fall von Lemma 3.1.4.

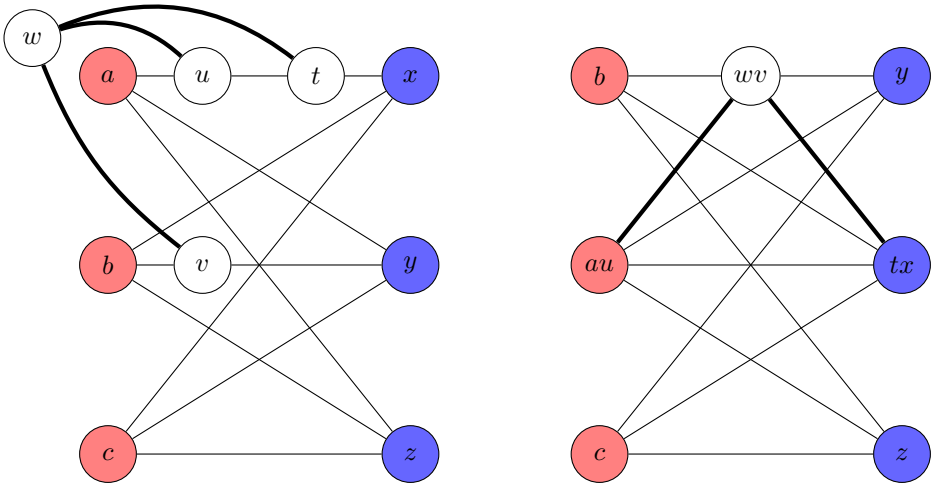


Abbildung 3.3: Beispiel zum zweiten Fall von Lemma 3.1.4.

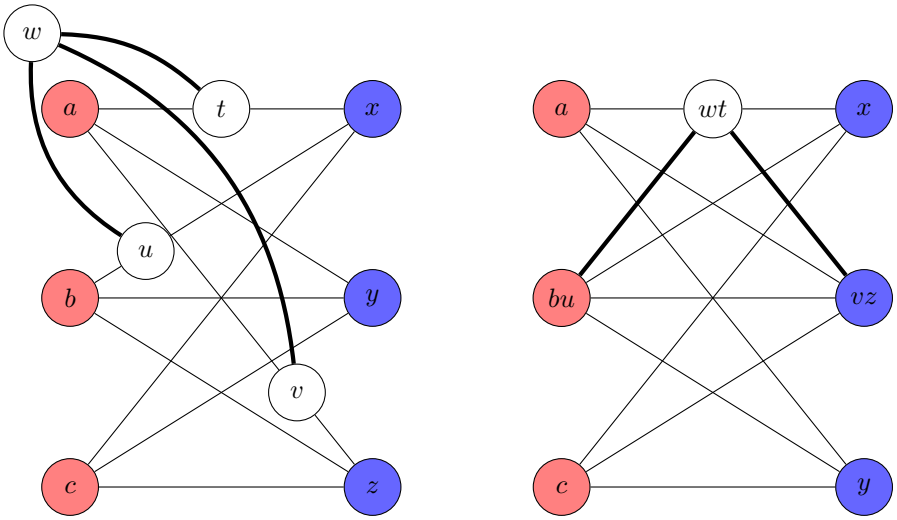


Abbildung 3.4: Beispiel zum dritten Fall von Lemma 3.1.4.

3.1.5 Lemma ([14]). *Sei G ein 3-zusammenhängender Graph und S ein $K_{3,3}$-Subdivision in G. Betrachtet wird ein Pfad außerhalb von S, der zwei Endpunkte hat, die in einem roten Branch-Fan, aber nicht im gleichen Branch-Path liegen. Analog dazu wird ein Pfad außerhalb von S gesucht, dessen Endpunkte in einem blauen Branch-Fan liegen, ohne dass diese auf dem gleichen Branch-Path in S liegen. Existieren diese beiden Pfade in G, dann enthält G einen K_5-Minor.*

Beweis ([14]). Sei P_1 der Pfad, der zwei Knoten in einem roten Branch-Fan verbindet und P_2 der, der zwei in einem blauen Branch-Fan verbindet. O. B. d. A. hat P_1 Endpunkte in $F(a)$ und P_2 in $F(x)$. Da laut Bedingung die Endpunkte nicht in einem einzelnen Pfad

von S liegen, kann a kein Endpunkt von P_1 und x kein Endpunkt von P_2 sein. Es ergeben sich zwei Fälle:

1. Die beiden Pfade haben keine gemeinsamen Knoten. Da P_1 beide Endpunkte in $F(a)$ hat, liegen diese beiden Endpunkte in zwei unterschiedlichen blauen Branch-Fans. Entsprechend sind die Endpunkte von P_2 in unterschiedlichen roten Branch-Fans. Werden die Endpunkte von P_1 je mit den beiden blauen und die von P_2 mit den beiden roten Knoten von S kontrahiert, entsteht ein K_5-Minor. Abb. 3.5 skizziert den Fall beispielhaft.

2. Die beiden Pfade haben einen gemeinsamen Knoten w. Liegt dieser gemeinsame Knoten außerhalb von S, kann Lemma 3.1.4 angewendet werden, da die Endpunkte der Pfade nicht alle im gleichen Branch-Fan liegen. Liegt w innerhalb von S, ist er ein Endpunkt von P_1 und P_2 und muss auf dem Pfad $P(a, x)$ liegen, da dieser der einzige gemeinsame Pfad ist, siehe links in 3.6. Sei $P_1 = P(w, u)$ und $P_2 = P(w, v)$. Da u nicht in $F(x)$ liegt und v nicht in $F(a)$, gibt es einen Pfad von u zu einem blauen Knoten und von v zu einem roten Knoten, die sich nicht kreuzen und daher kontrahiert werden können. Durch die Kontraktion dieser beiden Pfade entsteht, wie in Abb. 3.6 rechts zu sehen, ein M-Minor. □

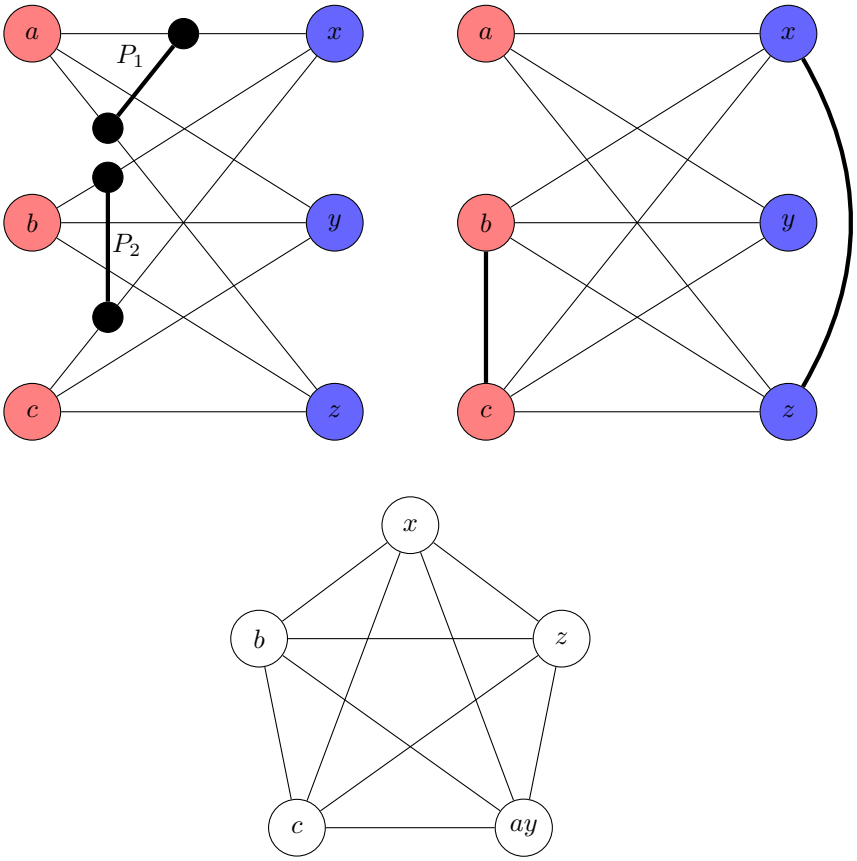


Abbildung 3.5: Beispiel zum ersten Fall von Lemma 3.1.5. Links oben ist S mit zwei zusätzlichen Pfaden aus G abgebildet. P_1 hat beide Endpunkte in $F(a)$ sowie den blauen Branch-Fans $F(x)$ und $F(y)$, mit denen die Endpunkte kontrahiert werden. P_2 hat seine Endpunkte in den roten Branch-Fans $F(b)$ und $F(c)$, mit denen sie kontrahiert werden. Rechts oben ist der dadurch entstehende Minor abgebildet und unten wird der enthaltene K_5-Minor gezeigt.

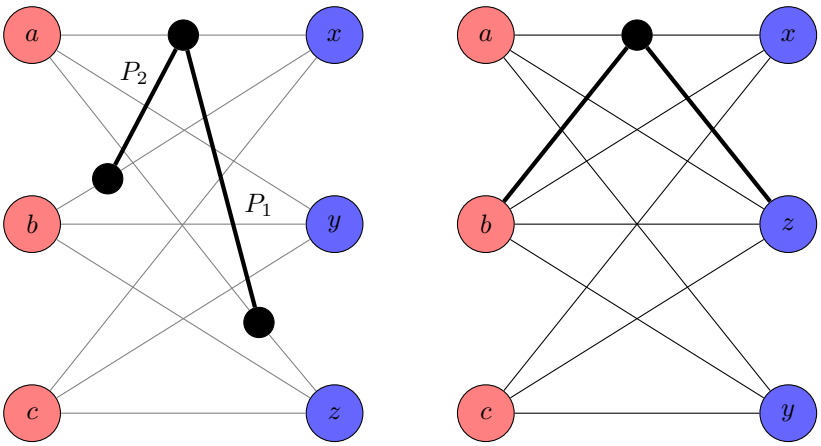


Abbildung 3.6: Beispiel zum zweiten Fall von Lemma 3.1.5. Rechts ist der enthaltene M-Minor abgebildet.

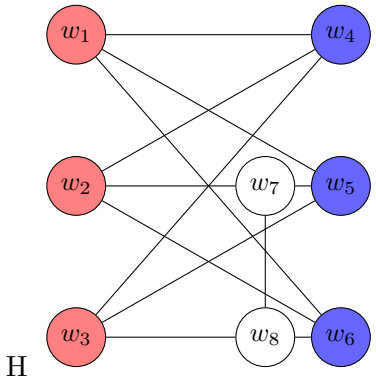


Abbildung 3.7: Darstellung von einem Graph S, der isomorph zu W ist.

3.1.6 Lemma ([14]). *Sei G ein 3-zusammenhängender Graph und S ein $K_{3,3}$-Subdivision in G. Betrachtet wird ein Pfad außerhalb von S, der zwei innere Knoten paralleler Pfade in S verbindet, sowie ein Pfad außerhalb von S, dessen Endpunkte nicht beide im gleichen Pfad von S liegen. Bestehen die Endpunkte der beiden Pfade aus mindestens drei unterschiedlichen Knoten in S, enthält G einen K_5-Minor.*

3.1.7 Lemma ([14]). *Sei G ein 3-zusammenhängender Graph und S ein $K_{3,3}$-Subdivision in G mit den roten Knoten $R = \{a, b, c\}$ und den blauen Knoten $B = \{x, y, z\}$. Bilden weder R noch B einen 3-Separator, enthält G einen K_5-Minor.*

Beweis ([14]). Falls R und B keinen $(3, 3)$-Separator bilden, ist sowohl der Graph $G - R$ als auch $G - B$ zusammenhängend. Sei P_1 ein Pfad, der zwei blaue Branch-Fans in $G - R$ und P_2 einer, der zwei rote Branch-Fans in $G - B$ verbindet. Beide liegen außerhalb von S. Die Endpunkte von P_1 seien u_1 und v_1, die von P_2 seien u_2 und v_2. u_1 und v_1 besitzen jeweils einen Pfad in S zu einem der roten Knoten. Folglich gibt es einen dritten roten Knoten, der keinen solchen Pfad besitzt – u_2 wird so gewählt, dass er in dem Branch-Fan dieses Knotens liegt. Demnach sind u_1, v_1 und u_2 unterschiedliche Knoten. Anschließend kann je nach vorliegendem Fall die Aussage auf eines der vorherigen Lemmata reduziert werden:

1. P_1 oder P_2 verbindet zwei parallele Pfade in S. In dem Fall kann Lemma 3.1.6 angewendet werden und G enthält einen K_5-Minor.

2. Die Endpunkte von P_1 liegen in einem einzelnen roten Branch-Fan – analog liegen die von P_2 in einem blauen. Nach Lemma 3.1.5 enthält G einen K_5-Minor. □

3.1.8 Lemma ([14]). *Sei G ein 3-zusammenhängender Graph mit einer W-Subdivision. Ist G nicht isomorph zu W, enthält G einen K_5-Minor.*

Beweis. G enthält mindestens acht Knoten, die S formen. Sei $G = (V_G, E_G)$ der Graph und $S = (V_S, E_S)$ der Teilgraph mit $V_S \subseteq V_G$ und $E_S \subseteq E_G$, der die W-Subdivision formt. Da W keinen K_5-Minor enthält, besitzt jeder Graph, der isomorph zu W ist, ebenfalls keinen. Es wird deshalb angenommen, dass G nicht isomorph zu W ist. Damit G nicht isomorph zu W ist trifft mindestens einer der Punkte auf G zu:

1. Es gibt eine weitere Kante $e = (w_i, w_j) \in E_G$ mit $\{w_i, w_j\} \in V_S$, aber $e \notin E_G$. Seien die Knoten $w_1, ..., w_8 \in V_S$ o. B. d. A. so gewählt, wie in Abb. 3.7 gezeigt. Seien $R = \{w_1, w_2, w_3\}$ als rote und $B = \{w_4, w_5, w_6\}$ als blaue Knoten bezeichnet. R bildet keinen $(3,3)$-Separator in S, da es den Pfad $P(w_5, w_6)$ in $S - R$ gibt. Analog ist B kein $(3,3)$-Separator wegen dem Pfad $P(w_2, w_3) \in S - B$. Verbindet die zusätzliche Kante zwei Knoten in R, dann ist e eine Kante, deren Endpunkte in einem blauen Branch-Fan, aber nicht auf dem gleichen Branch-Path liegen. Außerdem gibt es durch den Pfad $P(w_5, w_6)$ eine Kante, die zwei Endpunkte in einem roten Branch-Fan verbindet, die nicht auf dem gleichen Branch-Path liegen. Das Problem kann zu Lemma 3.1.5 reduziert werden (erster Fall von 3.1.5). Die Reduktion kann analog für $P(w_2, w_3)$ angewendet werden, wenn e zwei blaue Knoten verbindet. Übrig bleibt der Fall, dass ein Endpunkt von e entweder w_7 oder w_8 ist und der andere Endpunkt einer der roten oder einer der blauen Knoten ist. Sei o. B. d. A. $e = (w_1, w_7)$. Dann liegen die Endpunkte von e in einem blauen Branch-Fan ($F(w_5)$), aber nicht dem gleichen Branch-Path. Außerdem verbinden die Endpunkte der Kante (w_7, w_8) zwei parallele Branch-Paths ($P(w_2, w_5)$ und $P(w_3, w_6)$). Damit kann der Fall zu Lemma 3.1.6 reduziert werden.

2. Es gibt einen weiteren Knoten $v \in V_G$ mit $v \notin V_S$. Er hat aufgrund des 3-Zusammenhangs mindestens drei Pfade zu Knoten von S. Seien diese drei Knoten $\{w_1, w_2, w_3\} \in V_S$ beliebig. Da es in W keine Cliquen der Größe 3 gibt, sind mindestens zwei der drei Knoten nicht adjazent. Sei o. B. d. A. $e = (w_1, w_2) \notin E_S$. Da $\{(w_1, v), (v, w_2)\} \in E_G$ gilt, kann eine dieser beiden Kanten kontrahiert werden, sodass durch die jeweils andere w_1 und w_2 adjazent sind. Der Minor S' mit $E'_S \cup e$ für $e = (w_1, w_2)$ kann auf den vorherigen Fall reduziert werden, da eine Kante außerhalb von S gefunden wurde. □

Als nächstes folgt der Beweis zu 3.1.3.

Beweis ([14]). Gezeigt wird, dass, falls S keinen $(3,3)$-Separator bildet, G entweder einen K_5-Minor enthält oder isomorph zu W ist. Falls kein K_5-Minor enthalten ist, gilt nach Lemma 3.1.7, dass $G - R$ oder $G - B$ nicht zusammenhängend ist. Demnach ist B ein 3-Separator, der den Graph teilt, aber die Knoten aus R liegen nicht alle in unterschiedlichen Zusammenhangskomponenten. Deshalb muss es außerhalb von S mindestens einen Pfad P_1 geben, der zwei der roten Knoten in $G - B$ verbindet. Analog gibt es einen Pfad P_2,

der zwei blaue Knoten in $G - R$ verbindet. Da P_1 zwei rote Branch-Fans verbindet, liegen seine Endpunkte in zwei verschiedenen Pfaden von S. Gleiches gilt für die Endpunkte von P_2. Liegen die Endpunkte von P_1 beide in einem einzelnen blauen Branch-Fan und die von P_2 in einem einzelnen roten, dann enthält G laut Lemma 3.1.5 einen K_5-Minor. Liegen die Endpunkte von P_1 in parallelen Pfaden von S, enthält G laut Lemma 3.1.6 einen K_5-Minor, da die Endpunkte von P_2 nicht auf einem Pfad von S liegen (analog falls P_2 auf parallelen Pfaden liegt). Übrig bleibt die Möglichkeit, dass die Endpunkte der beiden Pfade paarweise identisch sind, siehe $P(w_7, w_8)$ in Abb. 3.7. Dann ist G ein Subdivision zu W und enthält laut Lemma 3.1.8 keinen K_5-Minor bei Isomorphie zu W. □

3.2 Sequenzieller Algorithmus zum Finden von K_5-Minoren

Da die Theoreme größtenteils auf 3-zusammenhängenden Graphen arbeiten, muss der Eingabegraph ggf. zunächst angepasst werden, bevor der Planaritätstests angewendet werden kann. Ist der Graph nicht 2-zusammenhängend, gibt es einen Knoten, der einen $(1, j)$-Separator für $j \geq 2$ bildet. Genauso müssen zwei Knoten existieren, die einen $(2, j)$-Separator bilden, falls der Graph nicht 3-zusammenhängend ist. In beiden Fällen kann der Separator benutzt werden, um den Graphen in j augmentierte Komponenten zu zerlegen. Anschließend kann der 3-Zusammenhang der einzelnen Komponenten rekursiv geprüft werden. Sind die Komponenten alle 3-zusammenhängend, kann auf jede ein Planaritätstest angewendet werden. Kezdy und McGuinness verwenden den Williamson-Algorithmus [19], welcher in Linearzeit für einen Graphen eine K_5- bzw. $K_{3,3}$-Subdivision ausgibt oder feststellt, dass der Graph planar ist. In der Implementierung wird stattdessen der Planaritätstest von Boyer und Myrvold [5] verwendet, der bereits in OGDF existiert. Ergibt der Planaritätstest, dass eine Komponente planar ist, wird sie nicht weiter beachtet. Enthält sie einen K_5-Minor, kann der Algorithmus stoppen und diesen ausgeben. Wird ein $K_{3,3}$-Minor gefunden, wird geprüft, welcher der vier Fälle aus Theorem 3.1.3 zutrifft. Bei Isomorphie zu W wird die Komponente nicht weiter beachtet. Ist der $K_{3,3}$-Minor ein $(3, 3)$-Separator in der untersuchten Komponente, kann sie in augmentierte Komponenten zerlegt und der Algorithmus rekursiv darauf angewendet werden. Andernfalls müssen genügend Pfade in der Komponente existieren, sodass der $K_{3,3}$-Minor auch einen K_5-Minor bildet; der Algorithmus kann ihn ausgeben und terminieren. Diese Schritte werden solange wiederholt, bis alle augmentierten Komponenten planar bzw. isomorph zu W sind oder ein K_5-Minor gefunden wurde.

Für einen Graphen G sind die einzelnen Schritte zusammengefasst:

1. Gibt es einen 1-Separator in G, dann bilde damit augmentierte Komponenten und wende den Algorithmus rekursiv auf jede an.

2. Gibt es einen 2-Separator in G, dann bilde damit augmentierte Komponenten und wende den Algorithmus rekursiv auf jede an.

3. Planaritätstest:

   (a) G ist planar: Gib zurück, dass kein K_5-Minor gefunden wurde.

   (b) G enthält ein K_5-Subdivision: Gib den K_5-Minor zurück und beende den Algorithmus.

   (c) G enthält ein $K_{3,3}$-Subdivision: Weiter zu Schritt 4

4. G ist isomorph zu W: Gib zurück, dass kein K_5-Minor gefunden wurde. G ist nicht isomorph zu W: Weiter zu Schritt 5

5. Seien $R = \{a, b, c\}$ die roten und $B = \{x, y, z\}$ die blauen Branch-Ends der $K_{3,3}$-Subdivision. Ist R einen $(3, 3)$-Separator, sodass alle Knoten aus B in unterschiedlichen Zusammenhangskomponenten sind, dann bilde damit augmentierte Komponenten und wende Schritt 3 rekursiv auf jede an. (Analog für B als Separator)

6. Ist dieser Schritt erreicht, enthält G einen K_5-Minor bestehend aus der $K_{3,3}$-Subdivision, dem Pfad, der zwei rote, sowie dem Pfad, der zwei blaue Knoten verbindet.

3.2.1 Laufzeit

Der Laufzeitbeweis ist aus [14]. Zunächst werden die einzelnen Schritte für sich betrachtet. Beide benötigen Linearzeit [10] [9]. Für den Planaritätstest in Schritt 3 wird der Algorithmus von Boyer und Myrvold verwendet, welcher ebenfalls Linearzeit benötigt [5]. Schritt 4 ist ein Isomorphietest zu einem konstanten Graph und kann daher in konstanter Zeit durchgeführt werden. Für Schritt 5 werden maximal 6 Tiefensuchen in Linearzeit ausgeführt, um alle Knoten des $K_{3,3}$-Minors zu prüfen. Der letzte Schritt besteht hauptsächlich aus der Ausgabe, dass ein K_5-Minor gefunden wurde.

Jeder der einzelnen Schritte kann also in Linearzeit durchgeführt werden. Allerdings wird nach Schritt 5 ggf. Schritt 3 nach dem Divide-and-Conquer Prinzip rekursiv auf die augmentierten Komponenten angewendet. Es kann jedoch keine Aussage darüber getroffen werden, wie groß die Komponenten sind. So wird im Worst-Case ein Graph G mit n Knoten durch einen $(3, 3)$-Separator C in drei augmentierte Komponenten aufgeteilt. Es kann nicht ausgeschlossen werden, dass zwei Zusammenhangskomponenten, die durch $G - C$ entstehen, nur je einen Knoten haben. Dann besteht die dritte augmentierte Komponente

aus $n-2$ Knoten, sodass erneut Linearzeit ab Schritt 3 gebraucht wird. Für einen Graphen mit n Knoten ist die Laufzeit der Unterprobleme:

$$T(n) = T(n_1) + T(n_2) + T(n - n_1 - n_2) + 6 + cn$$

Der Wert 6 entsteht, da für einen $(3, 3)$-Separator 3 Knoten entfernt und 9 hinzugefügt werden. Durch cn wird die Linearzeit dargestellt, die außerhalb der Rekursion verbraucht wird. Die Größe der drei Unterprobleme liegt bei $n_1, n_2, n - n_1 - n_2 + 6 \leq n - 2$, da jedes Unterproblem mindestens einen Knoten hat und diese folglich in jedem anderen Unterproblem fehlen. Wenn n_1 und n_2 konstante Werte sind, wird das dritte Unterproblem pro Rekursionsschritt nur konstant kleiner. Dadurch liegt die Laufzeitkomplexität insgesamt in $\mathcal{O}(n^2)$ [14].

Kapitel 4

Wagner-Struktur

In diesem Kapitel soll Theorem 3.1.3 genauer betrachtet werden. Zunächst werden einige Baumstrukturen eingeführt, die den 3-Zusammenhang herstellen können. Anschließend werden $(3,3)$-Separatoren behandelt und eine Formulierung des Theorems von Wagner, das in Theorem 3.1.3 benutzt wurde, vorgestellt.

4.1 2-Zusammenhang

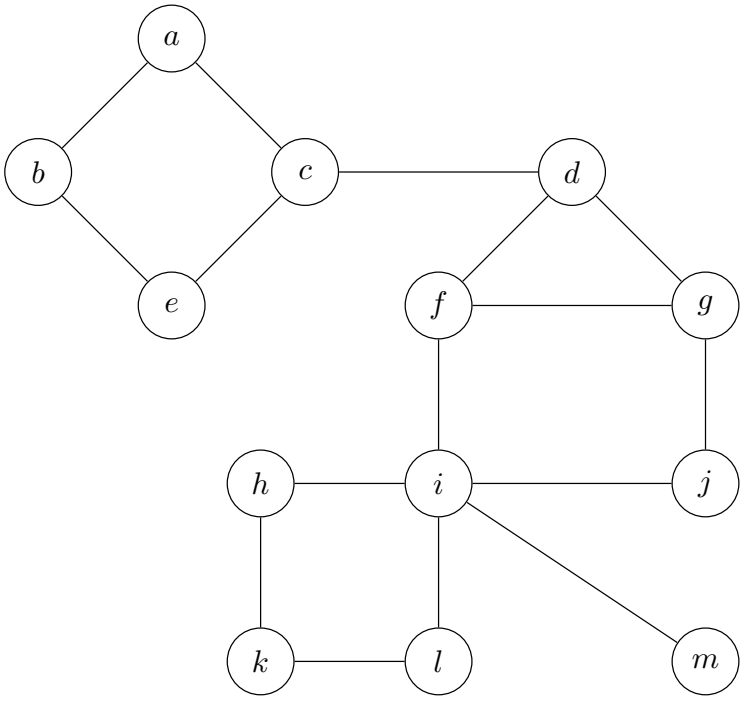


Abbildung 4.1: Ein nicht 2-zusammenhängender Graph G_1.

4.1.1 Definition ([4]). Als *Block* eines Graphen G wird jeder seiner maximalen Teilgraphen bezeichnet, der keinen 1-Separator enthält und für dessen Knotenmenge V' gilt:

$|V'| \geq 2$. Je zwei Blöcke haben maximal einen gemeinsamen Knoten, der einen 1-Separator in G bildet. Der *Block-Cut Tree* $T_{bc} = (V_{bc}, E_{bc}, B)$ von G ist ein Baum, der für jeden Block $b_v \in B$ von G einen assoziierten Knoten $v \in V_{bc}$ besitzt. Sind $\{b_1, ..., b_k\} \subseteq B$ eine Menge von Blöcken, sodass alle den gleichen gemeinsamen Knoten c haben, dann gibt es die Kanten $(b_i, b_j) \in E_{bc}$ für $1 \leq i \leq k$, $j \in [1, k]$, $i \neq j$. Dadurch ist b_j ein beliebig, aber fester Block, der adjazent zu allen $b_i \in \{b_1, ..., b_k\}$ außer sich selbst ist.

Als Beispiel wird in Abb. 4.1 ein Graph mit dazugehörigem Block-Cut Tree in Abb. 4.2 gezeigt. Die Knoten des Block-Cut Tree sind als durchgezogene Linien angegeben, in jedem dieser Knoten findet sich ein 2-zusammenhängender Teilgraph aus G_1.

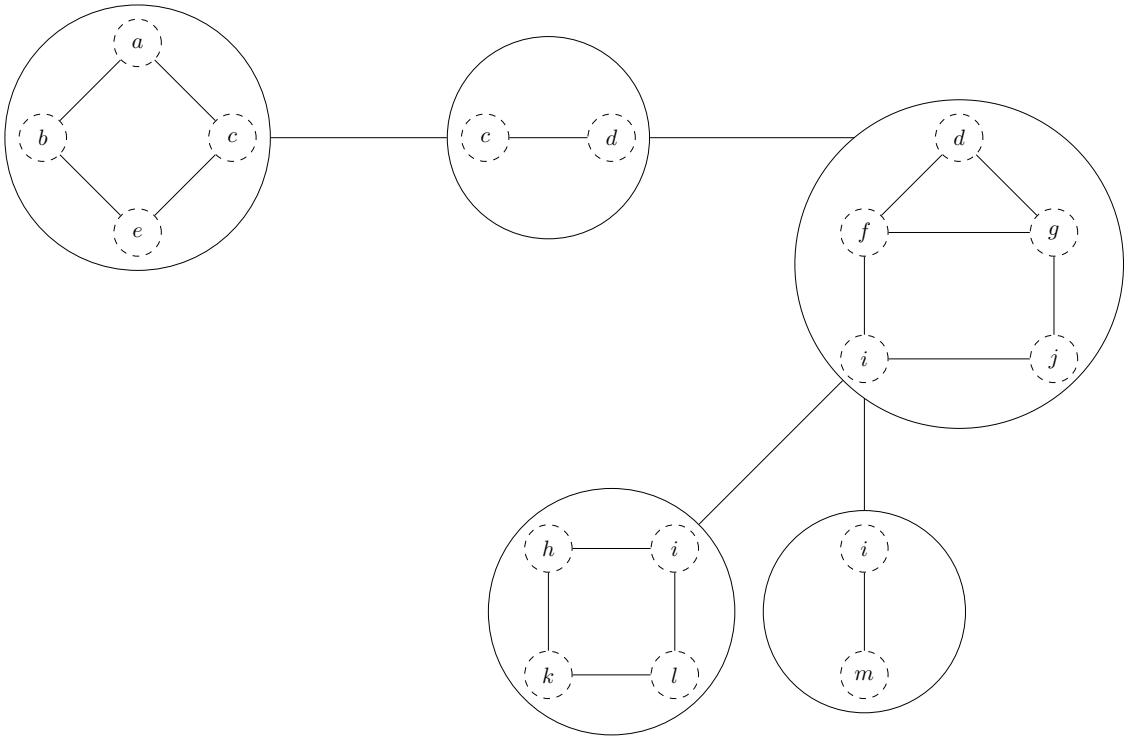


Abbildung 4.2: Der Block-Cut Tree zu G aus Abb. 4.1.

Reed und Li stellen in [16] und [17] Erweiterungen zum Block-Cut Tree vor, die für 1-, 2- und $(3, 3)$-Separatoren definiert sind. Der *1-Block-Tree* unterscheidet sich im Wesentlichen vom Block-Cut Tree dadurch, dass für 1-Separatoren zusätzliche Knoten im Baum angelegt werden.

4.1.2 Definition ([16]). Ein *1-Block-Tree* ist ein Baum $T = (V_T, E_T, \mathcal{G})$ für einen zusammenhängenden Graphen G. V_T ist die Menge der Knoten, E_T die der Kanten. Die Menge $\mathcal{G} = \{G_t\}_{t \in V_T}$ hat folgende Eigenschaften:

- Für alle Knoten $t \in V_T$ ist G_t ein Graph.

- Ist G 2-zusammenhängend, hat der 1-Block-Tree einen einzelnen Knoten t mit $G_t = G$.
- Andernfalls gibt es einen als *Cliquenknoten* bezeichneten $t \in V_T$:
  - Ist c ein $(1, j)$-Separator in G für $j \geq 2$, dann ist $G_t = G(c)$.
  - Seien $A_1, ..., A_j$ die augmentierten Komponenten definiert durch $G - c$. Dann gibt es $T_1, ..., T_j$ Teilbäume in $T - t$, sodass T_i ein 1-Block-Tree für $G_i = A_i$ ist.
  - Für jeden $t_i \in V_T$, der c als Teilgraph enthält, gibt es eine Kante $(t, t_i) \in E_T$.
- Jeder Knoten, der kein Cliquenknoten ist, wird als *Blockknoten* bezeichnet.

Sind T_{bc} der Block-Cut-Tree und T_1 der 1-Block-Tree für einen Graphen G, dann gibt es für jeden $v \in V_{bc}$ mit assoziiertem Block $b_v \in B$ einen Blockknoten $t \in V_T$, dessen Graph G_t isomorph zu b_v ist. Ein Beispiel findet sich in Abb. 4.3.

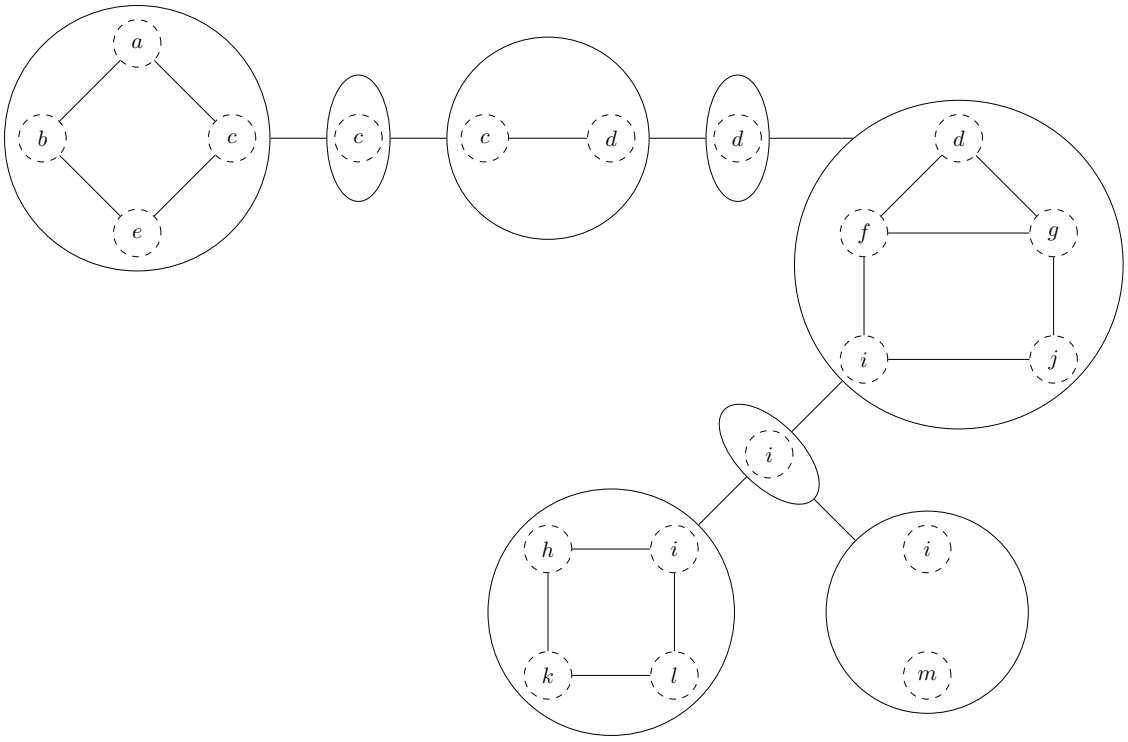


Abbildung 4.3: Der 1-Block-Tree zu G aus Abb. 4.2. Die Cliquenknoten sind ovalförmig dargestellt.

4.2 3-Zusammenhang

Nachdem durch Block-Cut Trees bzw. 1-Block-Trees der 2-Zusammenhang in allen Blockknoten hergestellt wurde, können die darin enthaltenen Teilgraphen jeweils als Eingabe für

einen SPQR-Baum genutzt werden, um 3-zusammenhängende Komponenten zu erzeugen. Die folgende Definition dazu ist [9] entnommen.

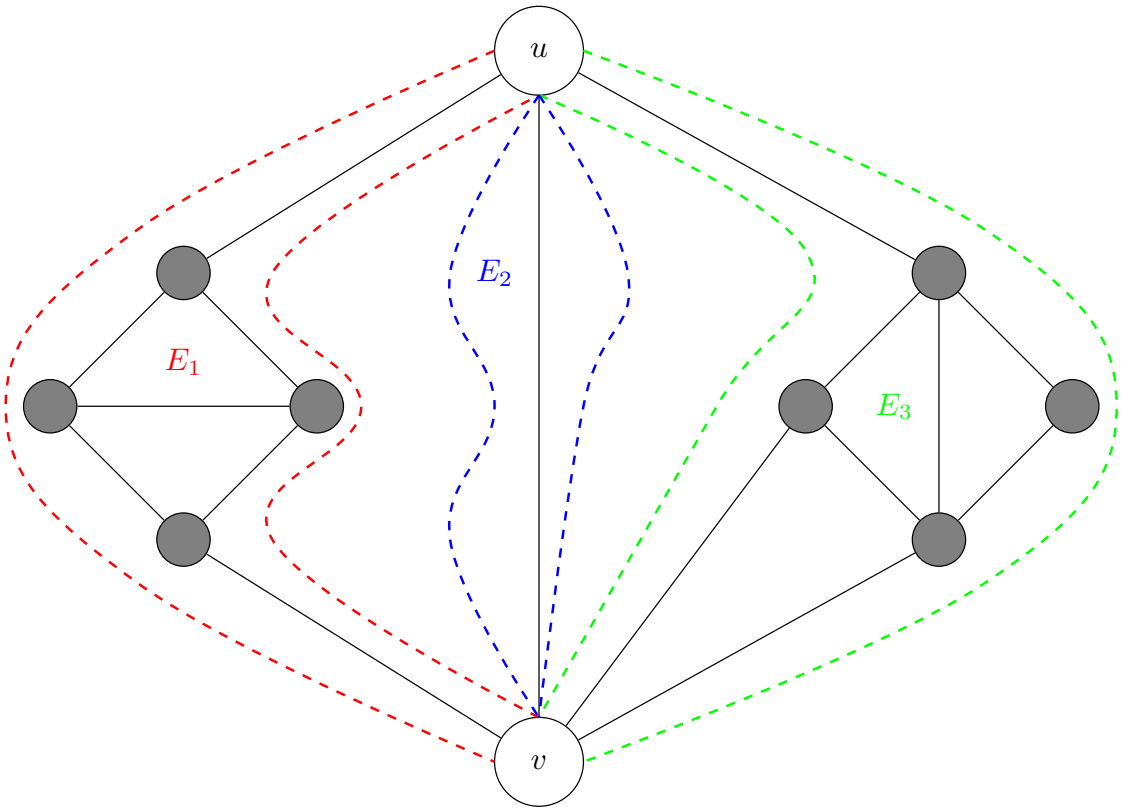


Abbildung 4.4: Unterteilung der Kanten in drei Mengen E_1, E_2 und E_3 anhand des Split Pairs $\{u, v\}$.

4.2.1 Definition. Sei $G = (V, E)$ ein 2-zusammenhängender Graph. Als *Split Pair* $\{u, v\}$ wird entweder ein adjazentes Knotenpaar oder ein 2-Separator bezeichnet, ein *maximales Split Pair* $\{s, t\}$ für $\{u, v\}$ teilt den Graphen so auf, dass u, v, s, und t in einer 3-zusammenhängenden Komponente liegen. Eine *Split Component* für ein Split Pair ist entweder eine Kante zwischen diesem oder der maximale Teilgraph, für den es kein Split Pair mehr ist. Ein Split Pair teilt die Kanten des Graphen in die Mengen $E_1, ..., E_k$, sodass eine Menge E_i alle Pfade enthält, die u oder v höchstens als Endpunkte haben. Dabei enthält E_i entweder einen Teilgraphen von G, der nur die beiden Knoten enthält, oder einen, für den das Split Pair kein 2-Separator ist. Eine Skizze dazu findet sich in Abb. 4.4. Der SPQR-Baum T_{SPQR} von G ist ein Baum, der G anhand von jedem Split Pair $\{u, v\}$ rekursiv in Minoren aufteilt, die mit den Knoten des Baumes assoziiert und mit S, P, Q oder R markiert werden:

- **Q**: Tritt der Randfall auf, dass G nur eine einzige Kante (u, v) besitzt, dann enthält der SPQR-Baum einen einzelnen Q-Knoten, der auf G verweist.

- **P**: Entstehen durch das Split Pair die Kantenmengen $E_1, ..., E_k$ mit $k \geq 3$, dann wird ein P-Knoten hinzugefügt. Der assoziierte Graph besteht aus dem Split Pair sowie k Kanten zwischen diesem.

- **S**: Andernfalls teilen u und v die Kanten in zwei Mengen, sodass es eine Kante $(u, v) \in E$ und einen weitere Pfad gibt, der das Split Pair verbindet. Der hinzugefügte S-Knoten enthält die Kante und den Pfad.

- **R**: Tritt keiner der obigen Fälle ein, dann seien $\{s_1, t_1\}, ..., \{s_k, t_k\}$ für $k \geq 1$ die maximalen Split Pairs für u, v und G_i für alle $1 \leq i \leq k$ die Vereinigung aller Split Components für $\{u, v\}$ außer der, die die Kante (u, v) enthält. Der neue R-Knoten enthält G, wobei jeder Teilgraph G_i durch die Kante (s_i, t_i) ersetzt wird.

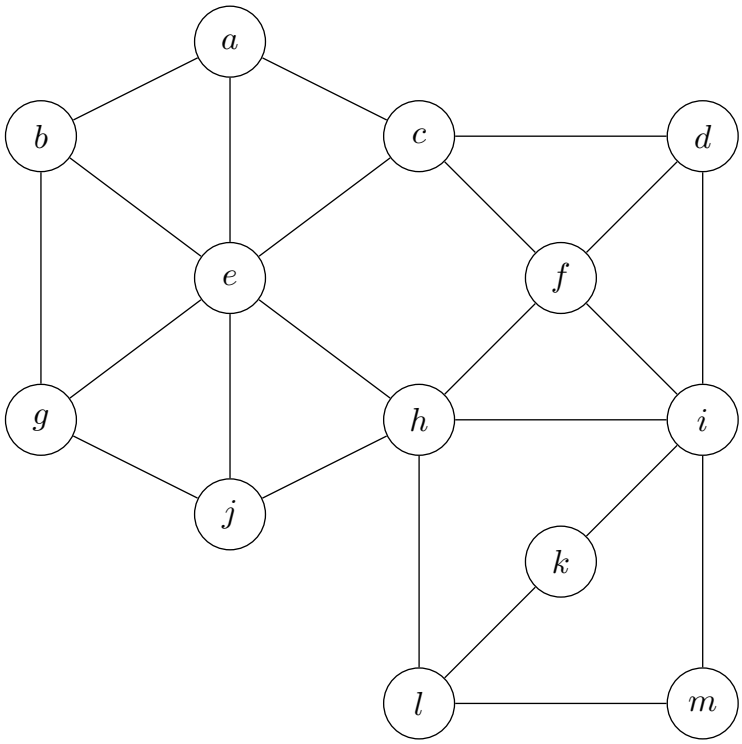


Abbildung 4.5: Ein 2-zusammenhängender Graph G_2.

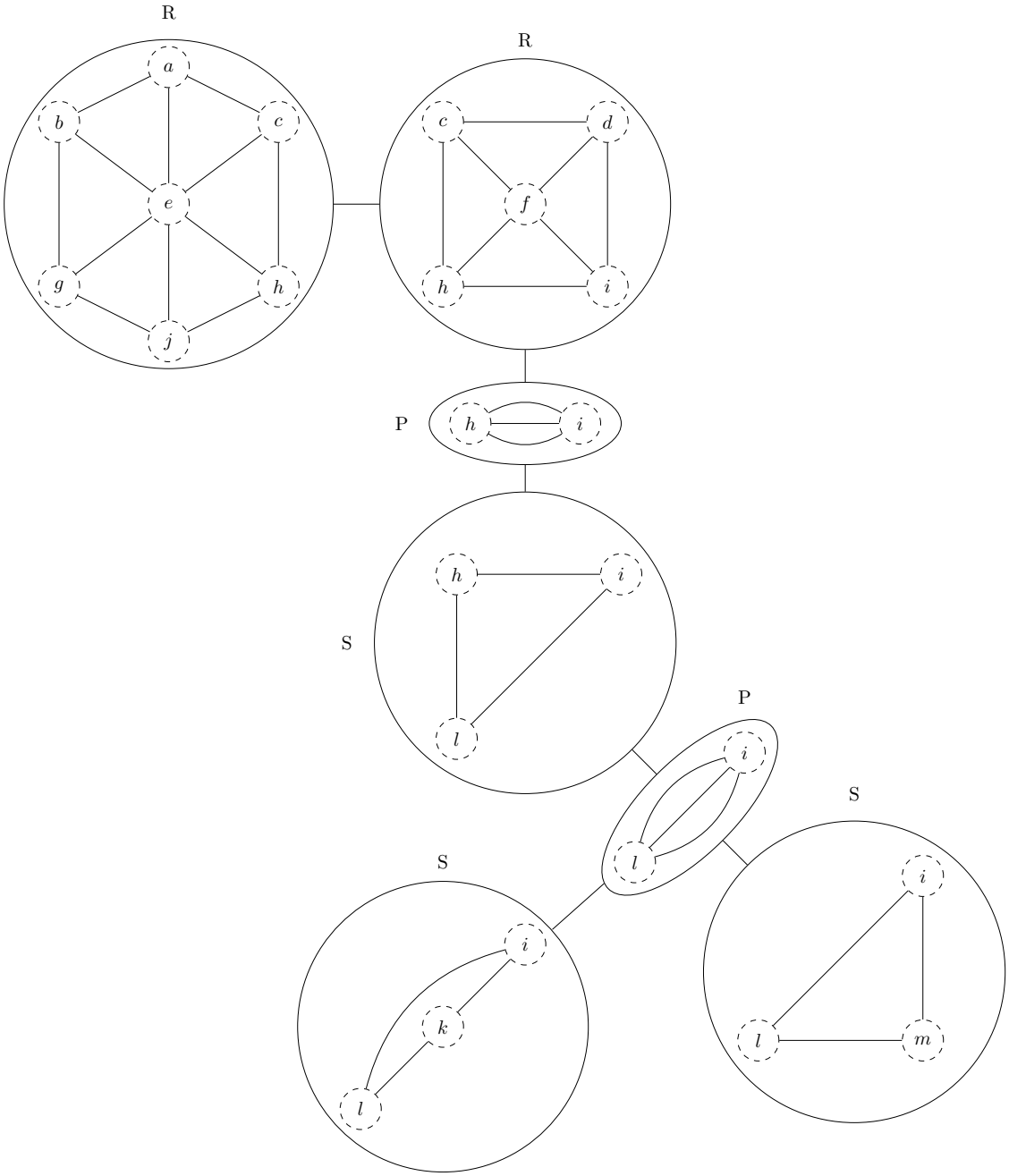


Abbildung 4.6: SPQR Baum zum Graphen G_2 aus Abb. 4.5.

Zu dem Graphen in Abb. 4.5 ist in Abb. 4.6 der zugehörige SPQR Baum skizziert. Besonders interessant sind die R-Knoten, die 3-zusammenhängende Minoren von G_2 sind. Zwar gilt das auch für die drei S-Knoten, allerdings sind Kreise im Allgemeinen nicht 3-zusammenhängend, aber immer planar.

Analog zum 1-Block-Tree stellen Li und Reed in [17] den *2-Block-Tree* vor. Er unterscheidet sich deutlich vom SPQR-Baum, enthält aber ebenfalls 3-zusammenhängende Minoren.

4.2.2 Definition ([17]). Ein *2-Block-Tree* ist ein Baum $T = (V_T, E_T, \mathcal{G})$ für einen 2-zusammenhängenden Graphen G. V_T ist die Menge der Knoten, E_T die der Kanten. Die Menge $\mathcal{G} = \{G_t\}_{t \in V_T}$ hat folgende Eigenschaften:

- Für alle Knoten $t \in V_T$ ist G_t ein Graph.
- Ist G 3-zusammenhängend, hat der 2-Block-Tree einen einzelnen Knoten t mit $G_t = G$.
- Andernfalls gibt es einen Cliquenknoten $t \in V_T$:
  - Ist $C = \{c_1, c_2\}$ ein $(2, j)$-Separator in G für $j \geq 2$, dann ist $G_t = G(C)$. Es werden Kanten zu G_t hinzugefügt, bis G_t eine Clique ist.
  - Seien $A_1, ..., A_j$ die augmentierten Komponenten definiert durch $G - C$. Dann gibt es $T_1, ..., T_j$ Teilbäume in $T - t$, sodass T_i ein 2-Block-Tree für $G_i = A_i$ ist.
  - Für jeden $t_i \in V_T$, der C als Teilgraph enthält, gibt es eine Kante $(t, t_i) \in E_T$.
- Jeder Knoten, der kein Cliquenknoten ist, ist ein Blockknoten.

Ein 2-Block-Tree zu G_2 aus Abb. 4.5 ist in Abb. 4.7 zu sehen. Es ist zu beobachten, dass Knoten wie etwa h oder i Teil mehrerer Separatoren sein können und somit nicht nur in mehreren Block-, sondern auch in mehreren Cliquenknoten enthalten sind.

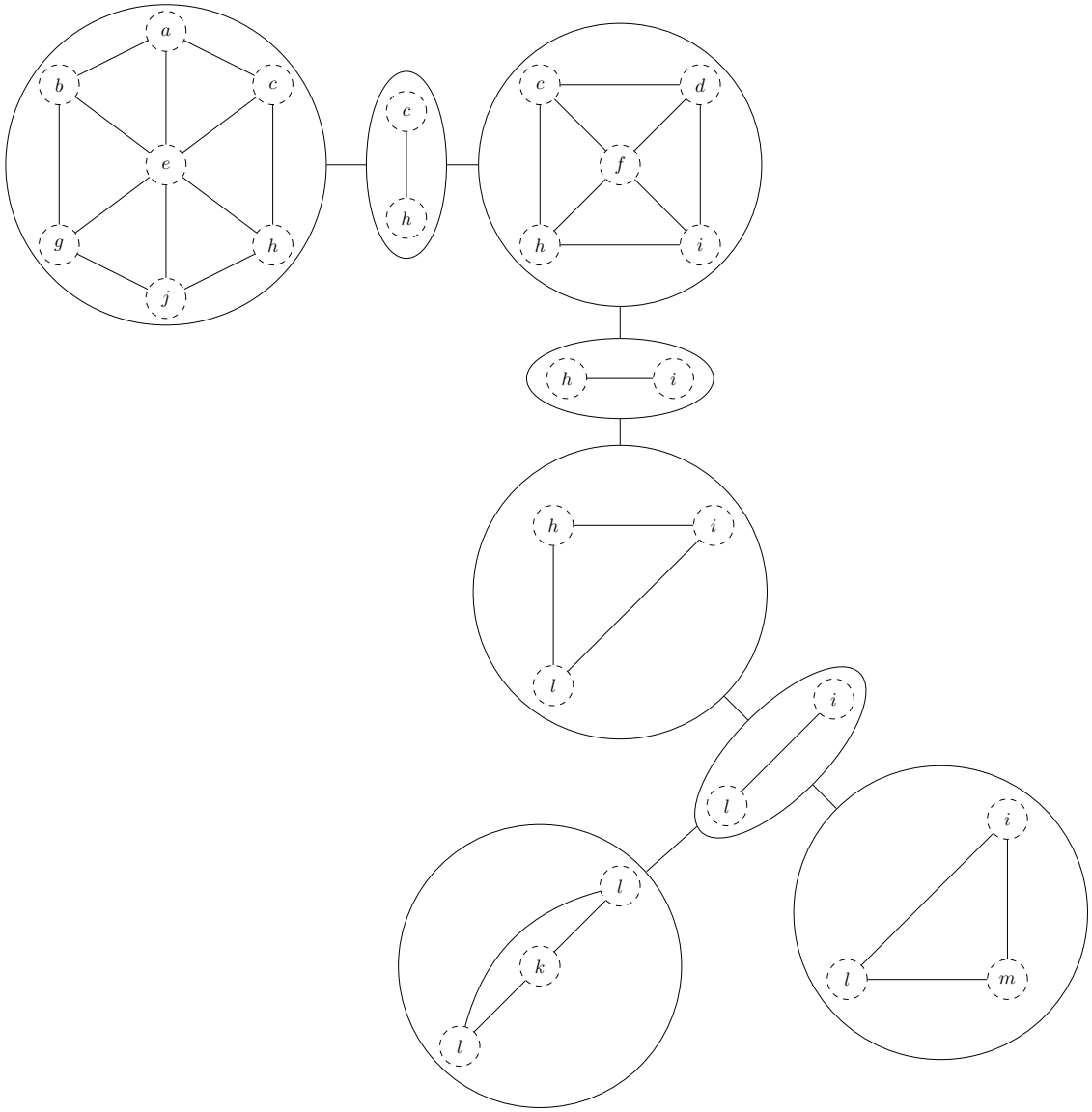


Abbildung 4.7: 2-Block-Tree zum Graphen aus Abb. 4.5. Die Cliquenknoten sind ovalförmig dargestellt und enthalten immer genau zwei Knoten, alle übrigen sind Blockknoten.

4.3 $(3,3)$-Separatoren

Letztlich wird in [17] eine weitere Baumstruktur für $(3,3)$-Separatoren eingeführt:

4.3.1 Definition ([17]). Ein *(3, 3)-Block-Tree* ist ein Baum $T = (V_T, E_T, \mathcal{G})$ für einen 3-zusammenhängenden Graphen G. V_T ist die Menge der Knoten, E_T die der Kanten. Die Menge $\mathcal{G} = \{G_t\}_{t \in V_T}$ hat folgende Eigenschaften:

- Für alle Knoten $t \in V_T$ ist G_t ein Graph.
- Hat G keinen $(3,3)$-Separator, hat der $(3,3)$-Block-Tree einen einzelnen Knoten t mit $G_t = G$.
- Andernfalls gibt es einen Cliquenknoten $t \in V_T$:
  - Ist $C = \{c_1, c_2, c_3\}$ ein $(3,j)$-Separator in G für $j \geq 3$, dann ist $G_t = G(C)$. Es werden Kanten zu G_t hinzugefügt, bis G_t eine Clique ist.
  - Seien $A_1, ..., A_j$ die augmentierten Komponenten definiert durch $G - C$. Dann gibt es $T_1, ..., T_j$ Teilbäume in $T - t$, sodass T_i ein $(3,3)$-Block-Tree für $G_i = A_i$ ist.
  - Für jeden $t_i \in V_T$, der C als Teilgraph enthält, gibt es eine Kante $(t, t_i) \in E_T$.
- Jeder Knoten, der kein Cliquenknoten ist, ist ein Blockknoten.

Ein Beispiel ist in Abb. 4.8 skizziert.

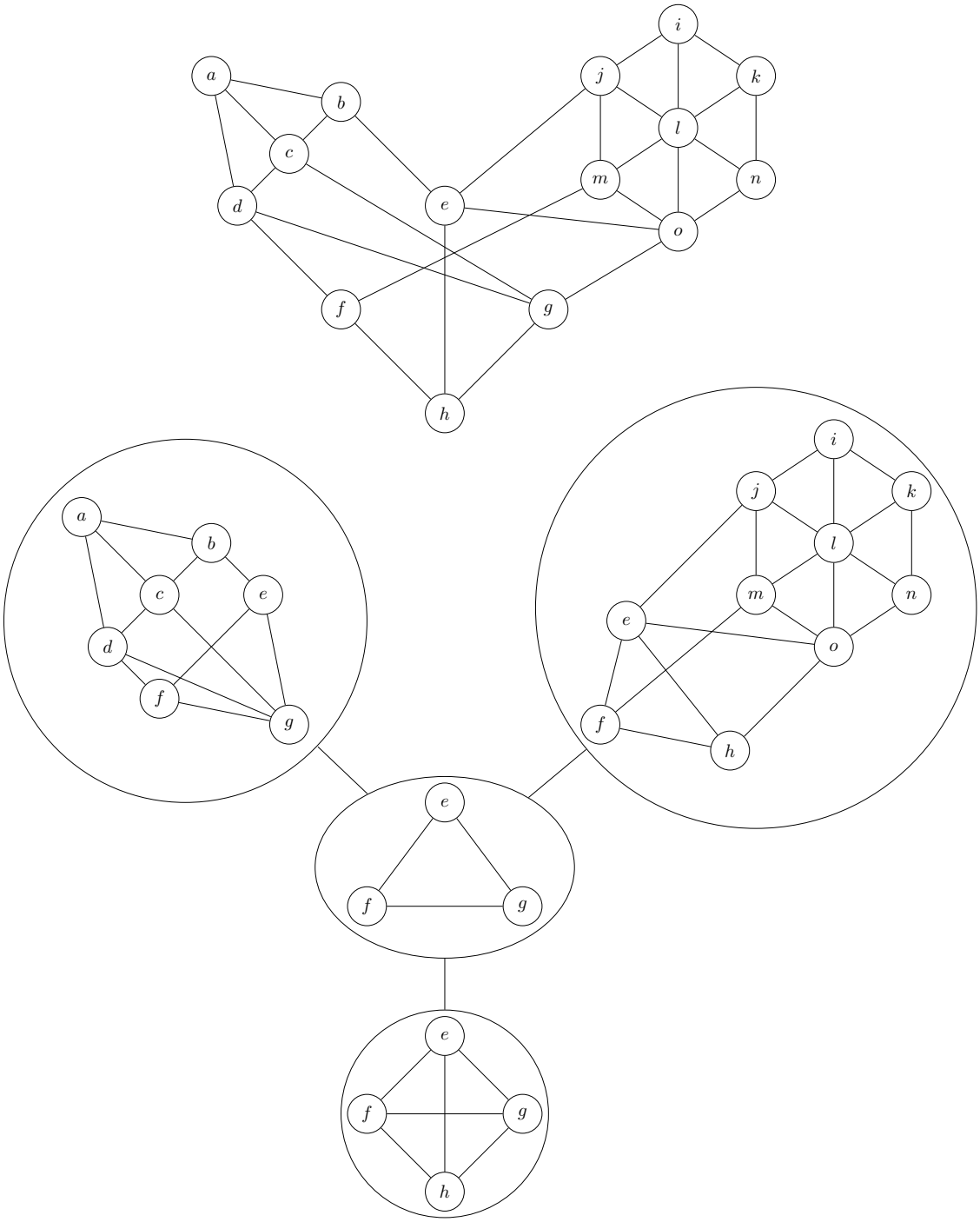


Abbildung 4.8: Ein 3-zusammenhängender Graph mit dazugehörigem $(3, 3)$-Block-Tree. Der Separator wird durch $\{e, f, g\}$ gebildet. Es ist zu sehen, dass der Eingabegraph nicht planar ist, da ein $K_{3,3}$-Minor enthalten ist. Die Blockknoten enthalten jedoch alle planare Minoren des ursprünglichen Graphs.

4.4 Wagner-Struktur

4.4.1 Definition ([16]). Sei G ein zusammenhängender Graph. Die *Wagner-Struktur* von G ist ein 1-Block-Tree $R = (V, E, \mathcal{G})$ und ein 2-Block-Tree $S_r = (V_r, E_r, \mathcal{G}_r)$ für jeden Blockknoten $r \in V$ und ein $(3,3)$-Block-Tree $T_{s_t} = (V_{s_t}, E_{s_t}, \mathcal{G}_{s_t})$ für jeden Blockknoten $s \in V_r, r \in V$.

Dadurch kann das Theorem von Wagner wie folgt formuliert werden:

4.4.2 Theorem ([18]). *Ein zusammenhängender Graph enthält genau dann keinen K_5-Minor, wenn für ihn eine Wagner-Struktur existiert, sodass für alle $(3,3)$-Block-Trees $T_{s_t} = (V_{s_t}, E_{s_t}, \mathcal{G}_{s_t}$ gilt: Jeder Graph $G \in \mathcal{G}_{s_t})$ ist entweder planar oder isomorph zu W.*

Trifft Theorem 4.4.2 auf die Wagner-Struktur eines Graphen G nicht zu, wird sie als *ungültig* bezeichnet und G enthält einen K_5-Minor. Ebenso werden $(3,3)$-Block-Trees als ungültig bezeichnet, wenn in ihnen ein K_5-Minor gefunden wird. Eine Wagner-Struktur kann ebenfalls für einen nicht-zusammenhängenden Graphen G aufgebaut werden. In dem Fall besteht sie aus einem Wald, der einen 1-Block-Tree für jede Zusammenhangskomponente von G hat.

Als Beispiel ist dazu in Abb. 4.9 ein Graph G zu sehen, der nicht planar ist, aber keinen K_5-Minor enthält. In Abb. 4.10 sind die 3-zusammenhängenden Graphen G_1, G_2, G_3, G_4, G_5 und G_6 abgebildet, von denen ebenfalls keiner einen K_5-Minor enthält. Außerdem ist G_6 nicht planar. Einige Cliquen sind in beiden Abbildungen mit je einer Farbe hervorgehoben, sodass durch Cliquen-Summen ein Graph G' erzeugt werden kann, der G als Teilgraphen enthält. In Abb. 4.11 ist die Wagner-Struktur zu G skizziert. Der 1-Block-Tree wurde nicht eingezeichnet, da er aufgrund des 2-Zusammenhangs von G aus nur einem Blockknoten besteht. Foglich existiert ein einzelner 2-Block-Tree, der aus blauen Knoten und Kanten besteht. Für jeden seiner Blockknoten existiert ein $(3,3)$-Block-Tree (rot). Es kann beobachtet werden, dass die Cliquenknoten aller Block-Trees genau die farblich markierten Cliquen aus Abb. 4.10 enthalten und in G Separatoren bilden. Außerdem ist zu sehen, dass die Blockknoten aller $(3,3)$-Block-Trees entweder planar oder isomorph zu W sind. Daraus folgt, dass G keinen K_5-Minor enthält.

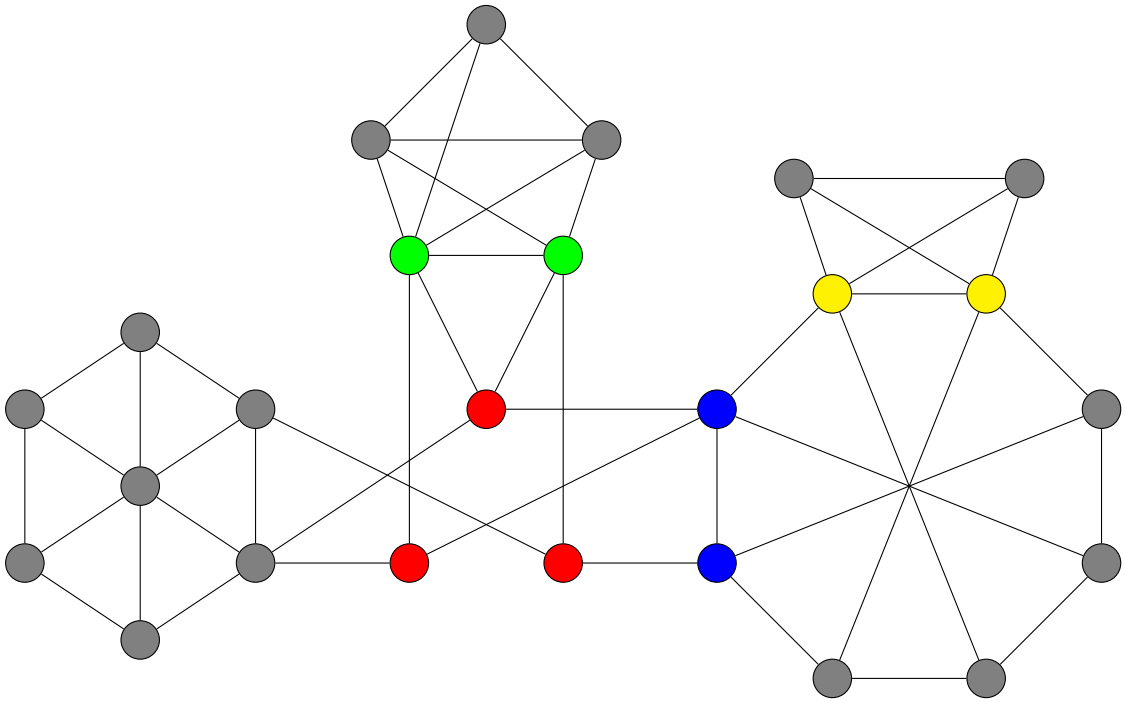


Abbildung 4.9: Ein nicht-planarer Graph G, der keinen K_5-Minor enthält.

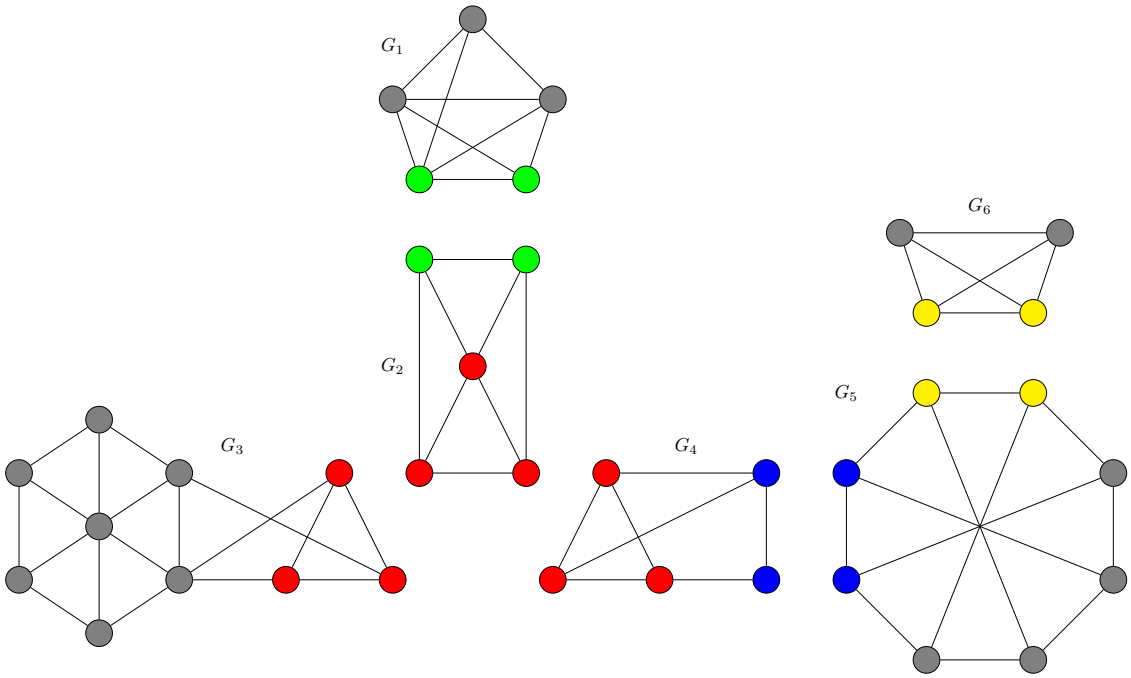


Abbildung 4.10: Mehrere Graphen, die planar oder isomorph zu W sind.

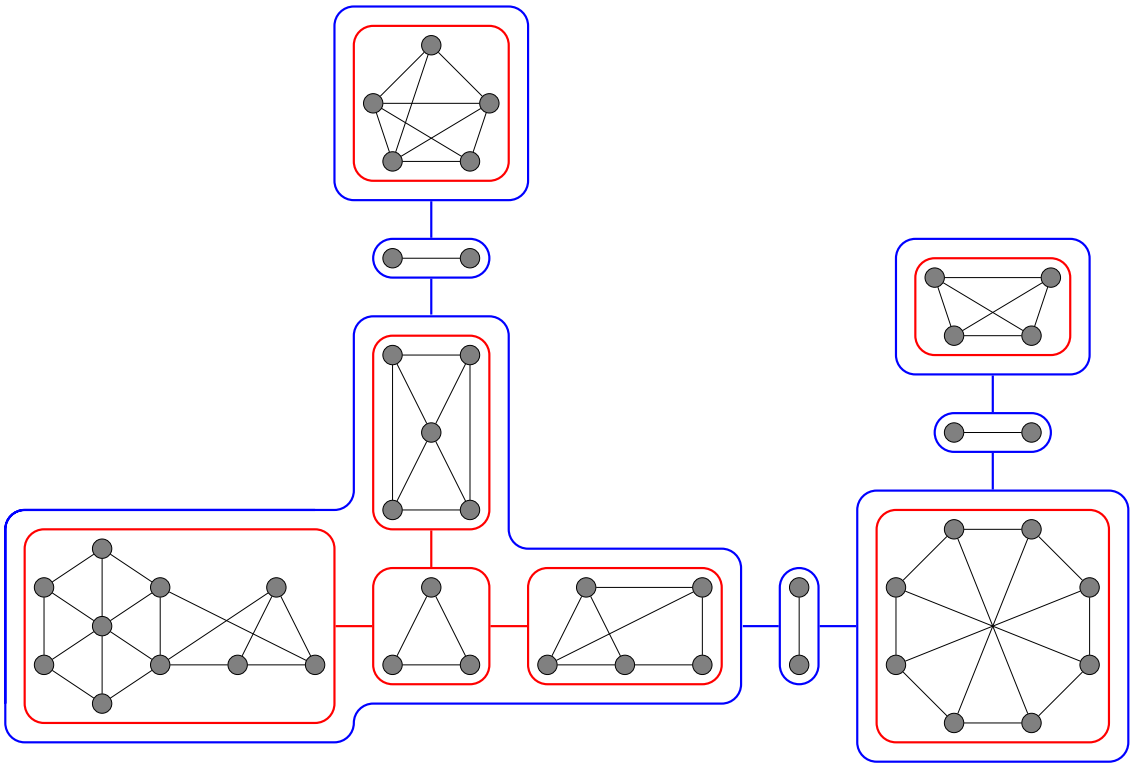


Abbildung 4.11: Wagner-Struktur von G aus Abb. 4.9. Da G bereits 2-zusammenhängend ist, enthält der 1-Block-Tree einen einzelnen Blockknoten für den kompletten Graphen und wurde nicht eingezeichnet. Die Knoten und Kanten des 2-Block-Trees sind blau, die der $(3, 3)$-Block-Trees rot hervorgehoben.

Kapitel 5

Implementierung

Im Folgenden werden Details zur Implmentierung im *Open Graph Drawing Framework* erklärt. Dazu wird zunächst kurz das Framework vorgestellt, anschließend wird erklärt, wie der Algorithmus von Kezdy und McGuinness zusammen mit einer Wagner-Struktur umgesetzt werden kann, um ein Zertifikat zu liefern und welche Schwierigkeiten durch Separatoren auftreten.

Das *Open Graph Drawing Framework (OGDF)* ist ein in C++ geschriebenes Framework, das Algorithmen und Datenstrukturen für Graphen enthält, wobei ein besonderes Augenmerk auf dem Zeichnen von Graphen liegt. OGDF kann unabhängig von anderen Frameworks und Bibliotheken genutzt werden und läuft sowohl unter Linux als auch unter Windows und MacOS. Es wurde 2005 unter der GNU General Public License als Open Source Projekt veröffentlicht [1] [6].

5.1 Algorithmus von Kezdy und McGuinness als Wagner-Struktur

Im Folgenden wird die Wagner-Struktur mit dem Algorithmus von Kezdy und McGuinness verglichen und dieser so modifiziert, dass er sie erzeugen kann. Daraufhin werden augmentierte Komponenten für 1- und 2-Separatoren durch Block-Cut-Trees und SPQR-Bäume erzeugt und abschließend ein Zertifikat erklärt.

5.1.1 Block-Trees

Im Algorithmus von Kezdy und McGuinness wird der Graph in augmentierte Komponenten aufgeteilt, wenn ein $(1,2)$-, $(2,2)$- oder $(3,3)$-Separator gefunden wird. Er wird so lange ausgeführt, bis die Blockknoten aller $(3,3)$-Block-Trees der Wagner-Struktur planar bzw. isomorph zu W sind oder ein K_5-Minor gefunden wurde. Um eine Wagner-Struktur aufzubauen, wird zunächst für jede Zusammenhangskomponente G des Eingabegraphen

ein 1-Block-Tree angelegt, der einen einzelnen auf G verweisenden Blockknoten v_b enthält. Eine Funktion, um einen 1-Separator C in G zu finden, existiert bereits in OGDF. Sind alle Graphen der Blockknoten 2-zusammenhängend, kann mit dem nächsten Schritt fortgefahren werden. Andernfalls werden die augmentierten Komponenten $A_1, ..., A_j$ gebildet, die durch den gefundenen 1-Separator C in G, auf den der Blockknoten v_b verweist, definiert werden. Dann wird v_b durch einen Cliquenknoten v_c für C sowie einem Blockknoten $v_{b_{A_i}}$ für jede augmentierte Komponente ersetzt. Anschließend werden Kanten der Form $(v_c, v_{b_{A_i}})$ hinzugefügt, um jede neue augmentierte Komponente mit dem neuen Cliquenknoten zu verbinden. War v_b adjazent zu einem anderen Cliquenknoten v'_c, der auf einen Teilgraph G_C verweist, dann gibt es jetzt genau eine augmentierte $v_{b_{A_i}}$, die den Teilgraph G_C enthält. Die Kante $(v'_c, v_{b_{A_i}})$ wird hinzugefügt. Letztlich kann dieser Schritt auf alle neu entstandenen augmentierten Komponenten rekursiv angewendet werden, bis keine 1-Separatoren mehr gefunden werden.

Das gleiche Verfahren kann genutzt werden, um 2-Block-Trees für alle Blockknoten des 1-Block-Trees zu erzeugen – eine Suche nach 2-Separatoren ist ebenfalls bereits implementiert. Wird ein Cliquenknoten im 2-Block-Tree angelegt, muss ggf. eine zusätzliche Kante im assoziierten Graph eingefügt werden, damit die beiden Separatorknoten eine Clique formen. Da die Blockknoten die augmentierten Komponenten enthalten, ist die Kante dort immer vorhanden.

Als nächstes wird ein $(3, 3)$-Block-Tree T für jeden Blockknoten v_b der 2-Block-Trees aufgebaut. Zunächst wird ein neuer Blockknoten in T angelegt, der auf den mit v_b assozierten Graph G verweist. Der bereits implementierte Planaritätstest von Boyer und Myrvold [5] kann auf G angewendet werden. Wird dadurch ein K_5-Minor gefunden, kann der Algorithmus stoppen – die Wagner-Struktur ist ungültig und der Eingabegraph enthält einen K_5-Minor. Wird in G ein $K_{3,3}$-Minor gefunden, wird geprüft, ob G isomorph zu W ist. Ist das der Fall, dann enthält T zu diesem Zeitpunkt immer genau einen Blockknoten, der auf den zu W isomorphen Graph verweist – hätte T weitere Knoten, dann hätte nach Lemma 3.1.8 bereits ein Rekursionsschritt zuvor ein K_5-Minor gefunden werden müssen. Somit ist T bei Isomorphie zu W immer ein gültiger $(3, 3)$-Block-Tree. Andernfalls wird geprüft ob drei Knoten des $K_{3,3}$-Minors einen $(3, 3)$-Separator C formen. Falls ja, wird analog zu den anderen beiden Block-Trees G in augmentierte Komponenten aufgeteilt und der $(3, 3)$-Block-Tree entsprechend angepasst. Erneut ist darauf zu achten, dass die Knoten von C als Clique in dem Graph gespeichert werden, mit dem der neue Cliquenknoten assoziiert wird. Anschließend wird der Planaritätstest rekursiv auf die augmentierten Komponenten angewendet. Bildet der $K_{3,3}$-Minor dagegen keinen $(3, 3)$-Separator, dann kann nach den Lemmata aus Kapitel 3 ein K_5-Minor konstruiert werden und der Algorithmus stoppen. Findet der Planaritätstest in keinem Graphen der Blockknoten von T weitere K_5- oder $K_{3,3}$-Minoren, dann ist T ein gültiger $(3, 3)$-Block-Tree, der aus planaren

Blöcken besteht. Wurden alle $(3,3)$-Block-Trees aufgebaut, sodass die Wagner-Struktur gültig ist, dann ist der Eingabegraph K_5-Minor-frei.

5.1.2 Block-Cut-Trees und SPQR-Bäume

Wie in Kapitel 4 angedeutet, werden statt 1-Block-Trees Block-Cut-Trees und statt 2-Block-Trees SPQR-Bäume verwendet werden. Für beide gibt es bereits Implementierungen in OGDF, die in Linearzeit laufen. Der Nachteil ist, dass wenn ein 1-Block-Tree durch einen Block-Cut-Tree aufgebaut wird, jedes adjazente Knotenpaar des Block-Cut-Trees nach dem gemeinsamen Knoten durchsucht werden muss, der den 1-Separator bildet. Vor allem für einen 2-Block-Tree gestaltet sich die Suche nach 2-Separatoren als schwierig, da die Graphen der Blockknoten des 2-Block-Trees nicht identisch zu den Komponenten der R-Knoten des SPQR-Baumes sind. Ohne weitere Anpassungen scheint es nicht möglich, die 2-Separatoren aus dem SPQR-Baum direkt zu gewinnen. Stattdessen kann auf die 3-Zusammenhangssuche in OGDF zurückgegriffen werden, um 2-Separatoren zu finden. Sie brachte in der Implementierung jedoch weitere Schwierigkeiten mit sich, da sie nicht auf unzusammenhängenden Graphen funktioniert. Außerdem wird pro Suchdurchlauf nur ein 2-Separator gefunden, sodass durch mehrfaches Ausführen bereits in diesem Schritt eine quadratische Laufzeit entstehen kann. Wie im nächsten Kapitel beschrieben, sind die Cliquenknoten relevant für das Zertifikat. Die experimentelle Analyse in Kapitel 6 testet dagegen vor allem die Laufzeit des Algorithmus – die teils aufwändige Zertifikatsberechnung wurde nur für K_5-Minor-freie Graphen umgesetzt.

5.2 Zertifizierender Algorithmus

Nachdem im vorherigen Abschnitt erklärt wurde, wie der Algorithmus entscheidet, ob ein K_5-Minor enthalten ist, wird nun darauf eingegangen, wie die vom Algorithmus getroffene Aussage zertifiziert werden kann.

5.2.1 Virtuelle Kanten

Werden für einen 2- oder $(3,3)$-Separator C in G augmentierte Komponenten erzeugt, enthält jede Komponente die Knoten von C als Clique. Für den Fall, dass die Knoten in G keine Clique formen, wurden Kanten hinzugefügt, die als *virtuell* bezeichnet werden. Neben der Beantwortung der Frage, ob ein K_5-Minor enthalten ist, ist es für einige Anwendungsfälle wie das Ausgeben eines Zertifikates interessant, diesen in G zu finden: Das *Modell* eines K_5-Minoren in G besteht aus fünf Teilgraphen, die in G zusammenhängend und paarweise durch mindestens einen Pfad verbunden sind. Dafür ist es nicht nur wichtig, virtuelle Kanten zu markieren, sondern auch zu wissen, aus welchen Pfaden sie in G bestehen. Wie im Beweis von Theorem 3.1.1 bereits festgestellt, korrespondiert jede

virtuelle Kante zu einem Pfad: Sei beispielsweise $\{u, v\}$ ein 2-Separator, der einen zusammenhängenden Graph G in zwei Zusammenhangskomponenten Z_1 und Z_2 teilt. Ist A_1 die augmentierte Komponente, die den Teilgraph $Z_1 \cup \{u, v\}$ enthält, dann ist die Kante (u, v) ein Pfad in G, der durch den Teilgraph Z_2 verläuft.

Um etwa den Teilgraph in G zu finden, für den in einem Blockknoten eines $(3, 3)$-Block-Trees ein K_5-Modell K gefunden wurde, müssen die Knoten und Kanten von K auf G abgebildet werden:

5.2.1 Definition. Sei $G = (V_G, E_T)$ ein Graph und $G_T = (V_T, E_T)$ der Graph, der zu einem Block- oder Cliquenknoten eines Block-Trees von G gehört. Dann ist jeder Knoten $v_T \in V_T$ aus genau einem Knoten $v_G \in V_G$ durch (wiederholtes) Kopieren entstanden. Dies sei dargestellt durch $v_G = V_G[v_T]$ bzw. für Mengen $\{v_{T_1}, v_{T_2}\} \subseteq V_T$ durch $\{v_{G_1}, v_{G_2}\} = V_G[\{v_{T_1}, v_{T_2}\}]$. Für alle $v_{T_1}, v_{T_2} \in V_T$ gilt $V_G[v_{T_1}] \neq V_G[v_{T_2}]$.

Jeder Knoten von K kann also auf genau einen Knoten in G abgebildet werden. Um den Pfad zu finden, aus dem eine virtuelle Kante in K besteht, kann die Wagner-Struktur benutzt werden. Zwar ist sie ungültig in dem Sinne, dass der Graph nicht K_5-Minor-frei ist. Allerdings bilden die 2- und $(3, 3)$-Block-Trees durch die zu Cliquenknoten adjazenten Blockknoten ab, welche Pfade in G benutzt werden können. Sei $T = (V_T, E_T, \mathcal{G})$ ein $(3, 3)$-Block-Tree, der einen Blockknoten $t \in V_T$ mit Graph G_t enthält. Hat G_t eine virtuelle Kante $e_v = (c_1, c_2)$, dann gibt es entweder einen Cliquenknoten t_c mit Graph G_{t_c}, sodass $(t, t_c) \in E_T$ und G_{t_c} die Knoten $\{c_1, c_2\}$ hat. Oder es gibt einen Cliquenknoten in dem 2-Block-Tree, der T als Blockknoten enthält. Wird zu jeder virtuellen Kante eine Referenz zu dem Cliquenknoten gespeichert, für den sie erzeugt wurde, ist das Verfahren analog und wird für einen Cliquenknoten des $(3, 3)$-Block-Trees erklärt: Es gibt einen Blockknoten $t' \in V_T$ mit Graph G'_t, der adjazent zu t_c ist und $t \neq t'$. Um den Pfad in G zu finden, den e_v darstellt, kann ein beliebiger Pfad $P(c_1, c_2) = \{(c_1, v_1), (v_2, v_3), ..., (v_k, c_2)\}$ in G'_t gesucht werden, sodass (c_1, v_1) und (v_k, c_2) keine virtuellen Kanten sind. Dass es einen solchen Pfad gibt, wurde in Theorem 3.1.1 gezeigt. Ist $e'_v = (v_i, v_j) \in P(c_1, c_2)$ eine virtuelle Kante, sodass sie weder inzident zu c_1 noch zu c_2 ist, dann gibt es einen Blockknoten, dessen Graph den Pfad $P(v_i, v_j)$ hat. Durch Rekursion können die Schritte solange angewendet werden, bis $P(c_1, c_2)$ keine virtuellen Kanten enthält.

5.2.2 Zertifikat

Ist G der Eingabegraph, dann besteht das Zertifikat grundsätzlich aus Teilgraphen von G. Einfachheitshalber wird angenommen, dass keine virtuellen Kanten enthalten sind. Ist der Graph G K_5-Minor-frei, dann kann die aufgebaute Wagner-Struktur als Zertifikat dienen, um die Aussage nach dem Theorem von Wagner [18] zu belegen. Sei $G_T = (V_T, E_T)$ der Graph, der mit dem Block- oder Cliquenknoten eines Block-Cut-Trees assoziiert wird und $G = V_G, E_T$ der Graph, der als Eingabe benutzt wurde. Es werden die Blockknoten aller

$(3,3)$-Block-Trees sowie die Cliquenknoten aller Block-Trees der Wagner-Struktur wie folgt geprüft:

1. **Planarer Blockknoten**: Sei G' für G folgender Teilgraph: $G' = G(V_G[V_T])$ Dann ist G' planar, was z. B. durch einen Planaritätstest geprüft werden kann.

2. **Zu W isomorpher Blockknoten**: Es gilt $|V_T| = 8$. Der Teilgraph $G' = G(V_G[V_T])$ besteht also aus acht Knoten, die in G eine W-Subdivision formen. Der für den Algorithmus von Kezdy und McGuinness implementierte Isomorphietest kann in Kombination mit z. B. einer Tiefensuche zum Prüfen verwendet werden.

3. **Cliquenknoten**: Die Knotenmenge $V_G[V_T]$ bildet in G einen Separator. Sei a die Anzahl der Zusammenhangskomponenten von G. Für $|V_T| \leq 2$ enthält $G - V_G[V_T]$ mindestens $a + 1$ Zusammenhangskomponenten. Für $|V_T| = 3$ enthält $G - V_G[V_T]$ mindestens $a + 2$ Zusammenhangskomponenten.

Andernfalls kann ein *Modell* des gefunden K_5-Minoren zur Verifikation dienen. Wird jeder der fünf Teilgraphen zu einem Knoten kontrahiert und jeder Pfad, der zwei dieser Knoten verbindet, zu einer Kante kontrahiert, dann entsteht nach dem Entfernen von Mehrfachkanten ein Graph, der isomorph zu K_5 ist. So kann z. B. durch eine Tiefensuche leicht geprüft werden, ob jeder der fünf Teilgraphen in G zusammenhängend ist, und ob sie paarweise miteinander verbunden sind.

Kapitel 6

Experimentelle Analyse

Die Implementierung des Algorithmus von Kezdy und McGuinness wurde mit dem Planaritätstest von Boyer und Myrvold in OGDF verglichen. Um den Fehler durch äußere Einflüsse bei geringer Laufzeit zu minimieren, wurde die Zeit von mehreren Iterationen gemessen und anschließend gemittelt. Einerseits wurden beide Algorithmen auf der *Rome-Library*, einer in [7] Testmenge von Graphen, ausgeführt. Die insgesamt 11528 Graphen bestehen aus 10 bis 100 Knoten. 8249 sind nicht planar, davon enthalten 7897 einen K_5-Minor.

Andererseits wurde ein in OGDF enthaltener Generator genutzt, um zufällige Graphen mit bis zu 10000 Knoten und variierender Kantenanzahl zu erzeugen. Um vor allem den Kern des Kezdy-McGuinness Algorithmus, dem Behandeln von $K_{3,3}$-Minoren, zu Testen, wurden ausschließlich 3-zusammenhängende Graphen erzeugt. Dadurch beeinflussen weder Block-Cut Trees noch SPQR-Bäume die Laufzeit. Außerdem wird kein Zertifikat für den Fall erstellt, dass ein K_5-Minor gefunden wurde. Ist kein K_5-Minor im Eingabegraph enthalten, liegt ein Zertifikat vor, da während des Algorithmus die Wagner-Struktur aufgebaut wurde.

Getestet wurde auf einem Intel Core i7-9700k mit 3GHz und 32GB RAM. Als Compiler wurde der GCC 7.3.0 verwendet.

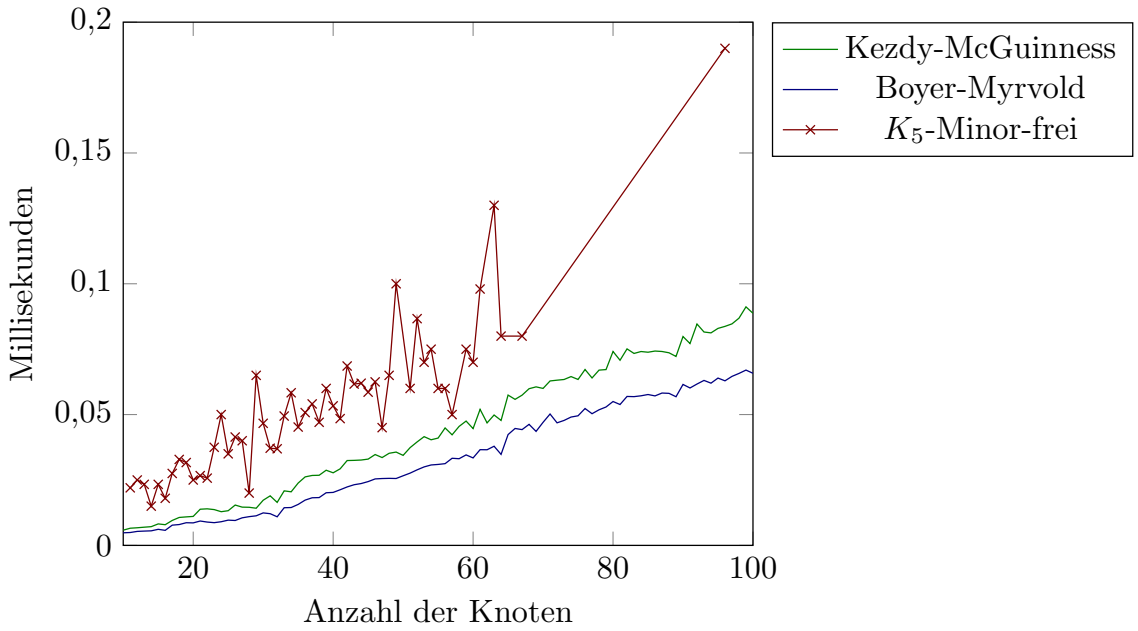


Abbildung 6.1: Benchmark mit den Graphen der Rome Library.

In Abb. 6.1 ist die Laufzeit in Millisekunden pro Graph zu sehen. Es fällt auf, dass sich die Laufzeit vom Kezdy-McGuinness-Algorithmus (grün) stark an der des Planaritätstests (blau) orientiert und damit eher linear als quadratisch ist. Die theoretische Worst-Case-Laufzeit tritt in diesem Anwendungsfall selten auf, da, wie in Abb. 6.2 zu sehen, die meisten Graphen entweder planar sind oder einen K_5-Minor enthalten. Ist ein Graph planar, kann der Algorithmus nach einmaligem Ausführen des Planaritätstests beendet werden, sodass die Laufzeit fast identisch zu der des Planaritätstests ist. Der Mehraufwand begründet sich in dem Aufbauen der nötigen Datenstrukturen wie der Wagner-Struktur oder einer Kopie des Eingabegraphen, die aus 3-zusammenhängende Komponenten besteht. Außerdem findet der Planaritätstest oft direkt einen K_5-Minor oder einen $K_{3,3}$-Minor, der kein $(3,3)$-Separator ist, sodass auch in den beiden Fällen der Kezdy-McGuinness-Algorithmus nach nur einem Rekursionsschritt terminieren kann. Abb. 6.2 ist zu entnehmen, dass viele Graphen entweder einen K_5-Minor enthalten oder planar sind. Deshalb wurde in Abb. 6.1 der Kezdy-McGuinness-Algorithmus (rot) zusätzlich nur auf den K_5-Minor-freien Graphen, die nicht planar sind, ausgeführt. Da nicht für alle Knotenanzahlen solche Graphen vorliegen, wurden gemessenen Werte durch Kreuze markiert. Hier wird deutlich, dass der Algorithmus deutlich länger läuft, weil der Planaritätstest $(3,3)$-Separatoren findet und rekursiv auf die augmentierten Komponenten angewendet wird. Generell ist in Abb. 6.2 aber zu sehen, dass $(3,3)$-Separatoren nur in weniger als 10% der Graphen vorkommen.

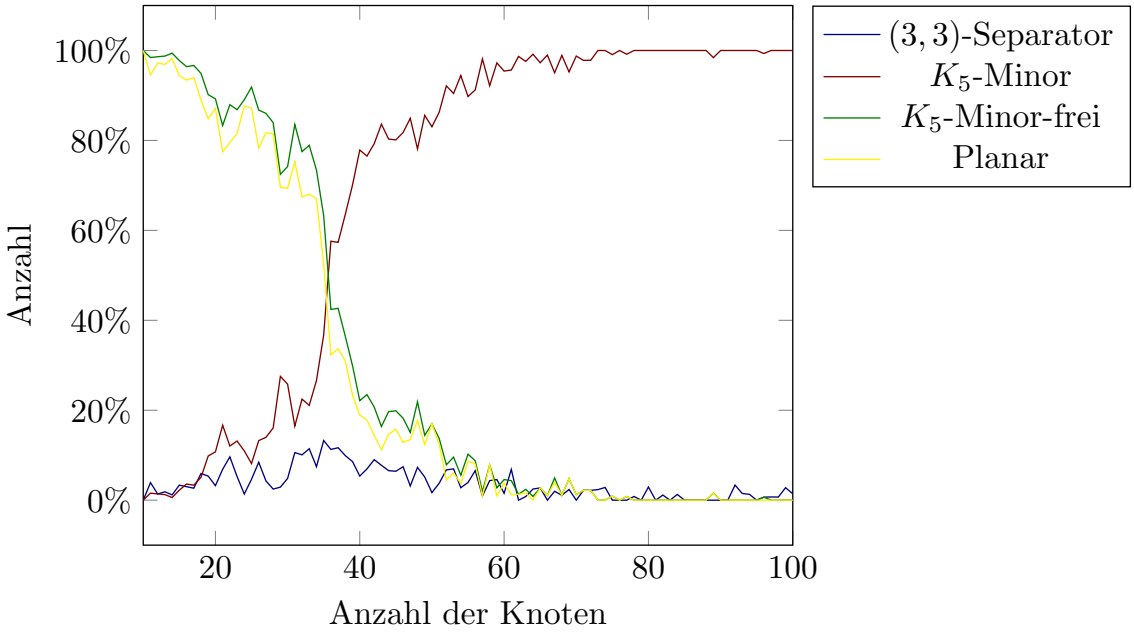


Abbildung 6.2: Angaben, wie viele Graphen $(3, 3)$-Separatoren und/oder K_5-Minor enthalten bzw. wie viele planar und/oder K_5-Minor-frei sind.

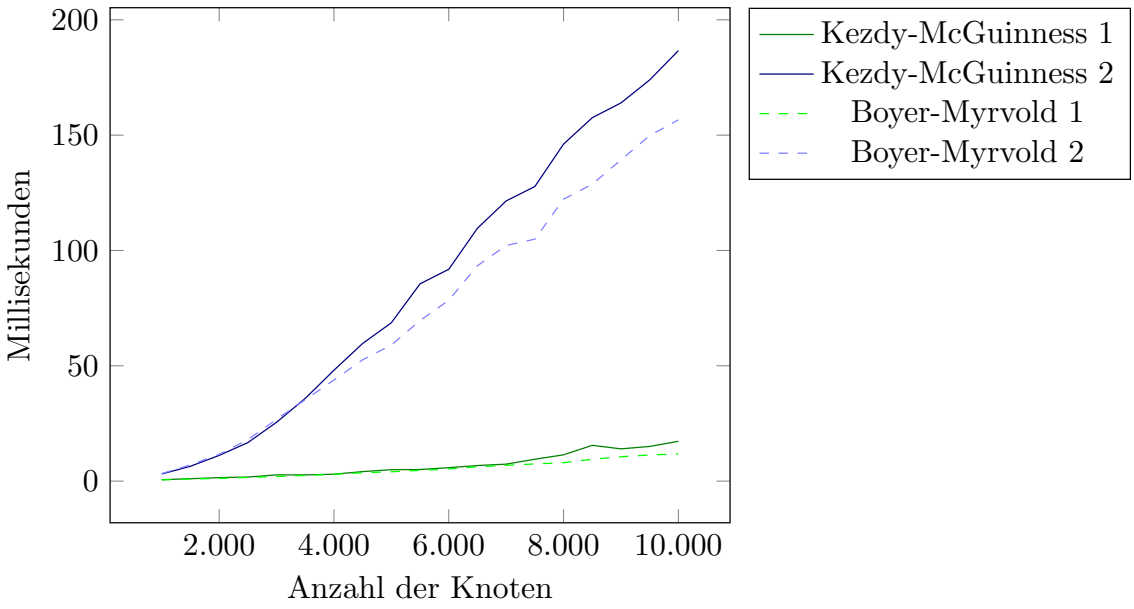


Abbildung 6.3: Benchmark 3-zusammenhängender Graphen mit n Knoten und $2 * n$ Kanten (grün) bzw. $10 * n$ Kanten.

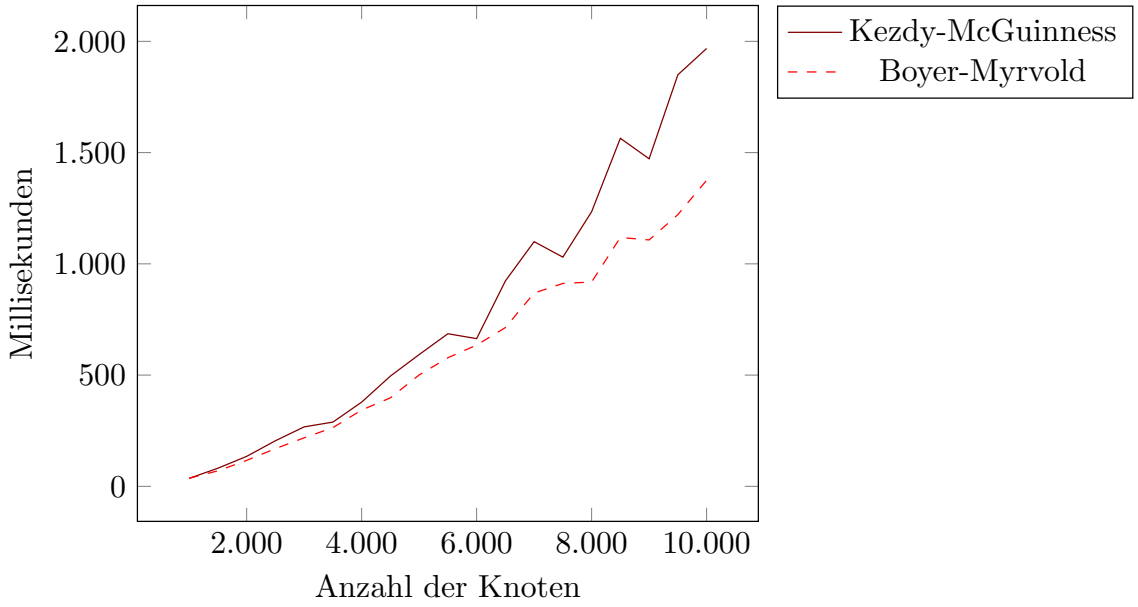


Abbildung 6.4: Benchmark 3-zusammenhängender Graphen mit n Knoten und $f * n$ Kanten für $30 \leq f \leq 100$.

In Abb. 6.3 und Abb. 6.4 sind die Laufzeiten für größere Graphen zu sehen. Es wurden für 20 unterschiedliche Knotengrößen jeweils 25 Graphen erzeugt und die Messwerte gemittelt. Die Messungen in Abb. 6.3 decken sich mit den Erkennntnissen, die aus der Rome-Library gewonnen wurden. Die Graphen in Abb. 6.4 für größere Kantenanzahlen variieren dagegen deutlicher. Vermutlich wird für die langsameren Instanzen durch den Planaritätstest ein $K_{3,3}$-Minor gefunden. Würde er einen $(3,3)$-Separator bilden, müsste die Laufzeit deutlich langsamer sein, wie etwa die Messungen für K_5-Minor-freie Graphen in Abb. 6.1 zeigen. Allerdings funktioniert der Test, ob der $K_{3,3}$-Minor ein $(3,3)$-Separator ist, über Tiefensuchen, die durch die deutlich höhere Kantenanzahl länger laufen. Es sei außerdem erwähnt, dass durch die hohe Kantenanzahl immer ein K_5-Minor enthalten ist.

Zuletzt wurden in Abb. 6.5 Graphen mit wenigen Knoten und Kanten getestet. Es wurden nur 10 Graphen pro Knotenanzahl erzeugt, dafür jedoch in einem kleineren Intervall. Dadurch konnte eine stärkere Varianz in der Laufzeit gemessen werden. In Abb. 6.6 wurde angegeben, wie viele $(3,3)$-Separatoren durchschnittlich gefunden wurden. Es zeichnet sich ab, dass der Kezdy-McGuinness-Algorithmus schneller ist, umso weniger $(3,3)$-Separatoren gefunden werden. Beispielsweise enthielten von den Graphen mit 700 Knoten im Schnitt 30% einen $(3,3)$-Separator, die Laufzeit lag bei $0,685ms$. Demgegenüber enthielt keiner der Graphen mit 720, bis 740 Knoten einen $(3,3)$-Separator, sodass eine schnellere Laufzeit abgelesen werden kann. Die Laufzeit für Graphen mit 740 Knoten

lag beispielsweise bei nur $0,532ms$ und liefen damit ähnlich schnell, wie die für kleineren Instanzen mit 580 Knoten.

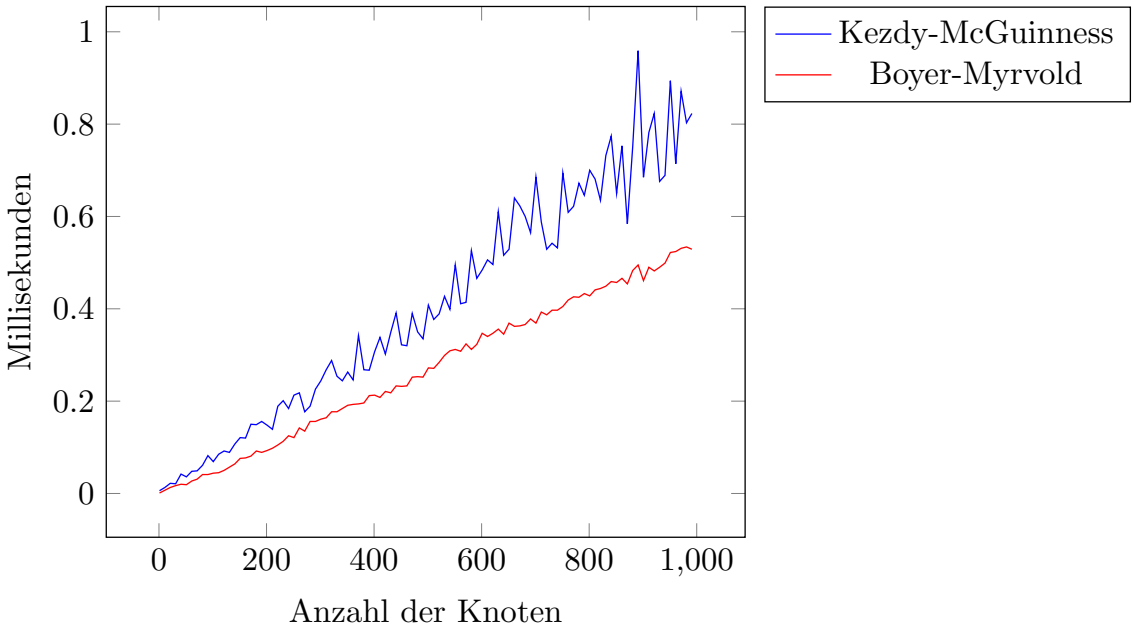


Abbildung 6.5: Benchmark für Graphen mit n Knoten und $2 * n$ Kanten.

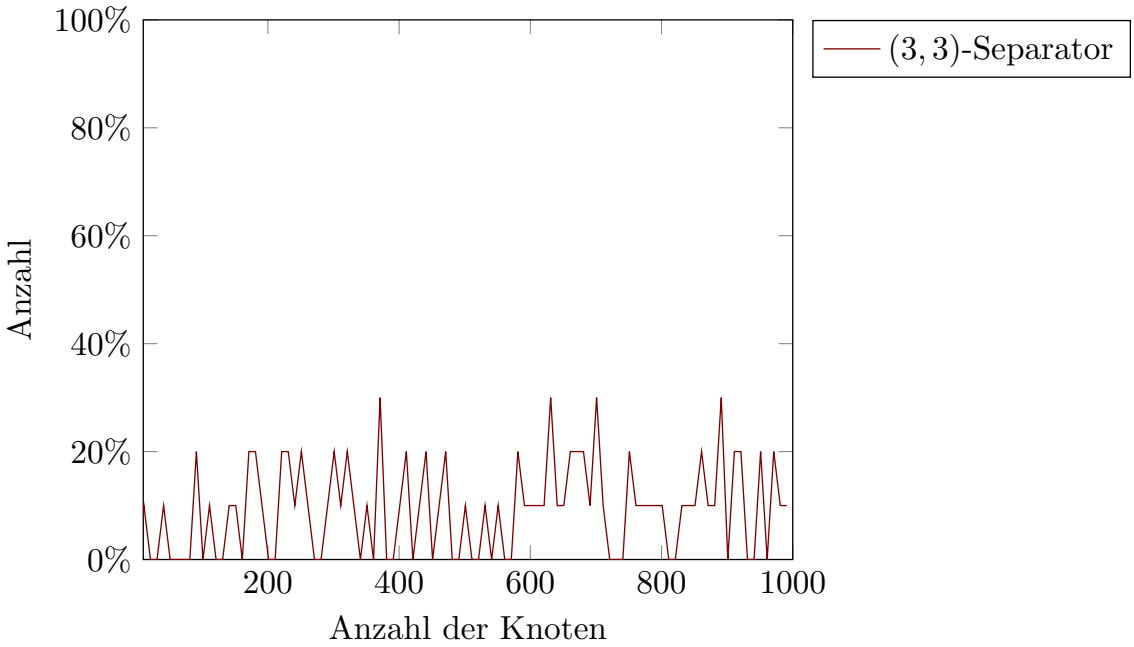


Abbildung 6.6: Angabe, wie viele Graphen mit n Knoten und $2 * n$ Kanten einen $(3, 3)$-Separatoren enthalten.

Kapitel 7

Zusammenfassung und Ausblick

Es wurde ein Verfahren vorgestellt, das entscheiden kann, ob ein Graph einen K_5-Minor enthält. Außerdem wurde erklärt, wie sich K_5-Minor-freie Graphen nach dem Theorem von Wagner aufteilen lassen und eine Baumstruktur gezeigt, die diese Aufteilung widerspiegelt. Für die Implementierung in OGDF wurden im Wesentlichen Block-Cut Trees und SPQR-Bäume verwendet, um 3-zusammenhängende Graphen zu erzeugen, und anschließend der Boyer-Myrvold Planaritätstest, um solange $(3, 3)$-Separatoren zu finden, bis ein K_5-Minor gefunden wurde oder der Graph eindeutig K_5-Minor-frei ist. Bisher werden Zertifikate für 3-zusammenhängende Graphen, die K_5-Minor-frei sind, erzeugt. Vor allem die Berechnung von K_5-Modellen im Eingabegraph kann im Hinblick auf den praktischen Nutzen von wesentlicher Bedeutung sein. Auch kann es für einige praktische Anwendungen sinnvoll sein, den Algorithmus zu erweitern, sodass mehrere K_5-Minoren in Graphen zu berechnen werden können. Dafür könnte der Algorithmus z. B. so angepasst werden, dass in gefundenen K_5-Minoren einzelne Kanten verändert (kontrahiert, entfernt) werden, um bei einem erneuten Durchlauf einen anderen K_5-Minor zu finden. So könnten viele K_5-Minoren gefunden werden, die sich allerdings kaum unterscheiden und eine hohe Laufzeit verursachen. Ein weiterer Ansatz könnte daraus bestehen, den Algorithmus nicht zu terminieren, wenn in einer augmentierten Komponente ein K_5-Minor gefunde wurde. Wird stattdessen in den übrigen Komponenten weitergesucht, ist es möglich, K_5-Minoren zu finden, die sich stark voneinander unterscheiden. Tests während der experimentellen Analyse haben jedoch angedeutet, dass auf diese Art meist nur einstellige Mengen von K_5-Minoren in den getesteten Graphen gefunden werden konnten. Da es sich um ein heuristisches Verfahren handelt, wurde es nicht mit in die Arbeit aufgenommen – einige praktische Anwendungen, wie die Berechnung des maximalen Schnittes in Graphen, die nicht K_5-Minor-frei sind, könnten jedoch davon profitieren. Es steht aus, diese beiden Ansätze in einer experimentellen Analyse für solche speziellen Anwendungsfälle zu prüfen.

Darüber hinaus gibt es Ansätze in [15], [17] und [16], die in theoretisch linearer Laufzeit entscheiden, ob ein Graph K_5-Minor-frei ist. Allerdings bleibt die Frage offen, ob eine

Implementierung in der Praxis eine bessere Laufzeit aufweisen würde, da teils mit großen Konstanten gearbeitet wird.

Letztlich wären weitere Laufzeittests speziell für Graphen, die nicht planar sind und keinen K_5-Minor enthalten, interessant, da vor allem dann hohe Laufzeiten in der Praxis auftreten können. Für alle getesteten Graphen konnte der Algorithmus dagegen in wenigen Sekunden entscheiden, ob ein K_5-Minor in einem Graph enthalten ist.

Literaturverzeichnis

[1] *OGDF - About.* `https://ogdf.uos.de/about/`. Stand 22. September 2019.

[2] Barahona, F.: *The Max-cut Problem on Graphs Not Contractible to K5.* Oper. Res. Lett., 2(3):107–111, 1983.

[3] Barahona, F., M. Jünger und G. Reinelt: *Experiments in quadratic 0-1 programming.* Math. Program., 44(1-3):127–137, 1989.

[4] Bondy, J. A. und U. S. R. Murty: *Graph Theory.* Springer Publishing Company, Incorporated, 1st Auflage, 2008.

[5] Boyer, J. M. und W. J. Myrvold: *On the Cutting Edge: Simplified O(n) Planarity by Edge Addition.* J. Graph Algorithms Appl., 8(3):241–273, 2004.

[6] Chimani, M., C. Gutwenger, M. Jünger, G. W. Klau, K. Klein und P. Mutzel: *The Open Graph Drawing Framework (OGDF).* In: Tamassia, R. (Herausgeber): *Handbook of Graph Drawing and Visualization*, Kapitel 17. CRC Press, 2014.

[7] Di Battista, G., A. Garg, G. Liotta, R. Tamassia, E. Tassinari und F. Vargiu: *An Experimental Comparison of Four Graph Drawing Algorithms.* Comput. Geom., 7:303–325, 1997.

[8] Diestel, R.: *Graph Theory, 4th Edition*, Band 173 der Reihe *Graduate texts in mathematics.* Springer, 2012.

[9] Gutwenger, C. und P. Mutzel: *A Linear Time Implementation of SPQR-Trees.* In: *Graph Drawing, 8th International Symposium, GD 2000, Colonial Williamsburg, VA, USA, September 20-23, 2000, Proceedings*, Seiten 77–90, 2000.

[10] Hopcroft, J. E. und R. E. Tarjan: *Efficient Algorithms for Graph Manipulation [H] (Algorithm 447).* Commun. ACM, 16(6):372–378, 1973.

[11] Jünger, M., E. Lobe, P. Mutzel, G. Reinelt, F. Rendl, G. Rinaldi und T. Stollenwerk: *Performance of a Quantum Annealer for Ising Ground State Computations on Chimera Graphs.* CoRR, abs/1904.11965, 2019.

[12] Karp, R. M.: *Reducibility Among Combinatorial Problems.* In: *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, Seiten 85–103, 1972.

[13] Kuratowski, C.: *Sur le problème des courbes gauches en Topologie.* Fundamenta Mathematicae, 15(1):271–283, 1930.

[14] Kézdy, A. E. und P. McGuinness: *Sequential and Parallel Algorithms to Find a K_5 Minor.* In: *Proceedings of the Third Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 27-29 January 1992, Orlando, Florida, USA.*, Seiten 345–356, 1992.

[15] Li, Z.: *Tree decompositions and linear time algorithms.* Doktorarbeit, School of Computer Science, McGill University, Montreal, Quebec, 12 2011.

[16] Reed, B. A. und Z. Li: *A Linear Time Algorithm for Testing if G has a K_5 Minor.*

[17] Reed, B. A. und Z. Li: *Optimization and Recognition for K_5-minor Free Graphs in Linear Time.* In: *LATIN 2008: Theoretical Informatics, 8th Latin American Symposium, Búzios, Brazil, April 7-11, 2008, Proceedings*, Seiten 206–215, 2008.

[18] Wagner, K.: *Über eine Eigenschaft der ebenen Komplexe.* Mathematische Annalen, 114:570–590, 1937.

[19] Williamson, S. G.: *Depth-First Search and Kuratowski Subgraphs.* J. ACM, 31(4):681–693, 1984.

Eidesstattliche Versicherung

Sauer, Julian	197859
Name, Vorname	Matr.-nr.

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

Effiziente Berechnung von K_5-Minoren in Graphen

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

Dortmund, den 30. September 2019	
Ort, Datum	Unterschrift

Belehrung:

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

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

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

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Dortmund, den 30. September 2019	
Ort, Datum	Unterschrift