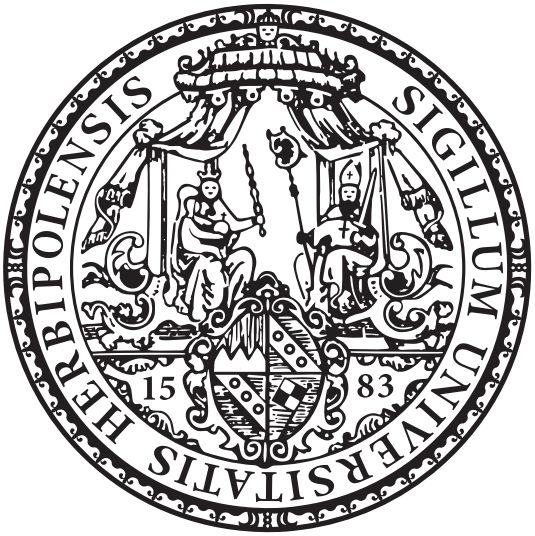
Masterarbeit

Die schwache Geradenüberdeckungszahl in 3D für eingeschränkte Graphenklassen

Alexander Erhardt

Abgabedatum:	22. Dezember 2021
Betreuer:	Prof. Dr. Alexander Wolff
	Dr. Oleg Verbitsky
Überarbeitung:	03. Mai 2022



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Zusammenfassung

Die lineare Knotenarborizität eines Graphen ist die kleinste Anzahl an Teilmengen, in die die Knoten eines Graphen unterteilt werden können, sodass jede dieser Teilmengen einen linearen Wald induziert. Sie ist gleich der schwachen Geradenüberdeckungszahl. Dies ist die Anzahl an Geraden, die benötigt werden, um die Knotenmenge eines geradlinig und kreuzungsfrei gezeichneten Graphen im $\mathbb{R}^3$ zu überdecken. In dieser Arbeit wird die schwache Geradenüberdeckungszahl und die Komplexität ihrer Berechnung untersucht. Es wird gezeigt, dass es bereits für planare Graphen und sogar planare Graphen mit Maximalgrad 6 NP-vollständig ist, zu entscheiden, ob ihre lineare Knotenarborizität höchstens 2 ist. Außerdem wird bewiesen, dass das Problem auf allgemeinen Graphen bereits ab Maximalgrad 5 NP-vollständig ist. Abschließend wird die parametrisierte Komplexität des Entscheidungsproblems, ob die lineare Knotenarborizität eines Graphen gleich 2 ist, untersucht und gezeigt, dass es mit Baumweite als Parameter parametrisierbar ist.

Abstract

The linear vertex arboricity of a graph is the smallest number of sets into which the vertices of a graph can be partitioned, so that each of these induces a linear forest. This is the same as the weak line cover number, that is, the minimum number of lines that are needed to cover the vertex set of a crossing-free straight-line drawing of the graph in $\mathbb{R}^3$. In this thesis, we investigate the complexity of computing the weak line cover number. We show that for planar graphs (even if restricted to a maximum degree of 6) it is NP-hard to decide whether the linear vertex arboricity is at most 2. We also show that for general graphs, the same problem is already NP-hard if restricted to a maximum degree of 5. Finally, we prove that deciding whether the linear arboricity of a graph is equal to 2 is fixed-parameter tractable with treewidth as the parameter.

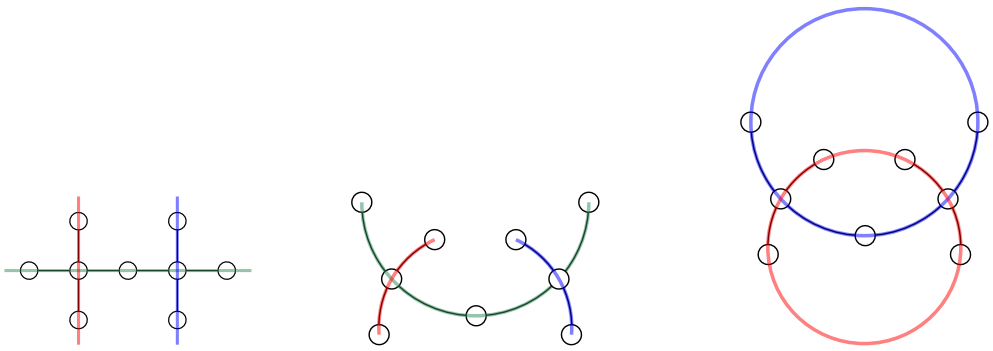
Inhaltsverzeichnis

1 Einführung

Das Graphzeichnen ist ein beliebtes Teilgebiet der Graphentheorie, da Informationen häufig als Graphen dargestellt werden. So findet es Anwendung im Modellieren von Geschäfts- oder Softwareprozessen. Aber auch in alltäglichen Dingen, wie zum Beispiel dem Zeichnen von U-Bahnplänen werden Forschungsergebnisse aus diesem Gebiet benutzt. Da es für uns oft einfacher ist, Bilder mit geringer Komplexität zu verstehen, ist das Ziel meistens eine möglichst einfache Darstellung zu gewährleisten, die aber noch alle benötigten Informationen widerspiegelt.

Im Allgemeinen beschäftigt sich die Forschung auf dem Gebiet mit der Frage, ob Graphen unter bestimmten Bedingungen in der Ebene oder im Raum zeichenbar sind. Die wahrscheinlich bekannteste Bedingung ist die Planarität. Unter dieser müssen Graphen so in der Ebene gezeichnet und eingebettet werden, dass sich keine zwei Kanten kreuzen. Eine weitere interessante Eigenschaft ist, ob man die Kanten eines Graphen durch bestimmte Objekte repräsentieren oder überdecken kann. In der Regel wird hierbei eine möglichst geringe Anzahl an Objekten angestrebt. Dadurch ist die Darstellung einfacher zu verstehen und wird von einem Betrachter als angenehmer empfunden.

Dujmović et al. [DESW07] betrachteten die Anzahl an Segmenten, die benötigt werden, um einen planaren, geradlinigen Graphen zu zeichnen. Dabei ist ein Segment die maximale Anzahl an Kanten, die eine geradlinige Strecke bilden; siehe Abbildung 1.1a. Später untersuchte Schulz [Sch15] dies für Kanten, die als Kreisbögen dargestellt werden sollen; siehe Abbildung 1.1b. Kryven et al. [KRW19] betrachteten die kleinste Anzahl an benötigten Kreisen (im $\mathbb{R}^2$) bzw. Kugeln (im $\mathbb{R}^3$), die einen gegebenen Graph, dessen Kanten Kreisbögen sind, überdecken; siehe Abbildung 1.1c. Dabei musste die Zeichnung kreuzungsfrei sein.



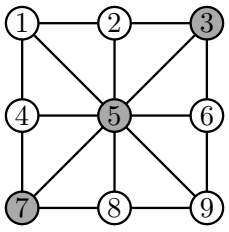
(a) Graph mit drei Segmenten, die die Kanten überdecken.

(b) Graph mit drei Kreisbögen, die die Kanten überdecken.

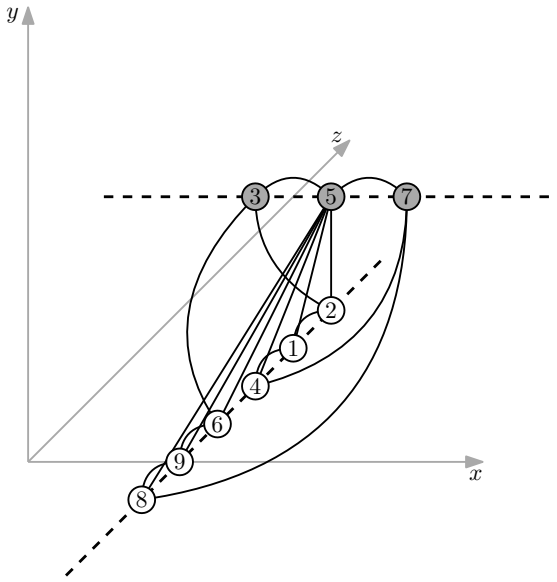
(c) Graph mit zwei Kreisen, die die Kanten überdecken.

Abb. 1.1: Überdeckung der Kanten eines Graphen mit verschiedenen Objekten.

Eine weitere interessante Fragestellung, ist die kleinste Anzahl an Geraden oder Ebenen, die benötigt werden, um einen Graphen kreuzungsfrei so im $\mathbb{R}^3$ zu zeichnen, dass alle seine Knoten und Kanten auf diesen Geraden oder Ebenen liegen. Chaplick et al. [CFL$^+$20] behandelten diese und weitere Fragestellungen bereits ausführlich. Die minimale Anzahl der Geraden wurde als Geradenüberdeckungszahl bezeichnet, die minimale Anzahl an Ebenen als Ebenenüberdeckungszahl. Wird die Bedingung, dass alle Kanten auf den Geraden liegen müssen, weggelassen, so spricht man von der schwachen Geradenüberdeckungszahl. Diese ist also die kleinste Anzahl an Geraden, die benötigt werden, so dass in einer kreuzungsfreien Zeichnung im $\mathbb{R}^3$ alle Knoten eines Graphen auf diesen liegen. Die schwache Geradenüberdeckungszahl eines Graphen im $\mathbb{R}^3$ entspricht der linearen Knotenarborizität eines Graphen. Die lineare Knotenarborizität ist die kleinste Anzahl an Teilmengen, in die die Knotenmenge eines Graphen aufgeteilt werden kann, so dass jede Teilmenge einen linearen Wald bzw. eine Menge von Pfaden induziert. In Abbildung 1.2a ist ein Graph G und eine mögliche Aufteilung seiner Knoten in zwei Teilmengen dargestellt. In Abbildung 1.2b ist eine kreuzungsfreie Zeichnung von G im $\mathbb{R}^3$ dargestellt, wobei Knoten aus einer Teilmenge auf einer Geraden liegen.



(a) Graph G mit linearer Knotenarborizität von 2 und eine mögliche Knotenaufteilung in zwei Teilmengen.



(b) Graph G kreuzungsfrei im $\mathbb{R}^3$ gezeichnet und die zwei Teilmengen auf zwei Geraden platziert.

Abb. 1.2: Darstellung von Graph G im $\mathbb{R}^2$ und $\mathbb{R}^3$.

In einer weiteren Veröffentlichung ([CFL+17]) wurde gezeigt, dass es NP-schwer ist zu entscheiden, ob die Knotenmenge eines gegebenen Graphen so auf zwei Geraden im $\mathbb{R}^3$ platziert werden kann, dass die resultierende Zeichnung kreuzungsfrei ist. Dazu wurde das Ergebnis von Farrugia [Far04] verwendet. Dieses besagt, dass es für zwei gegebene Graphenklassen $\mathcal{P}$ und $\mathcal{Q}$, die unter Knoten-disjunkter Vereinigung und induzierten Teilgraphen abgeschlossen sind, NP-schwer ist zu entscheiden, ob die Knoten eines gegebenen Graphen so in zwei Teilmengen A und B aufgeteilt werden können, dass $G[A] \in \mathcal{P}$ und $G[B] \in \mathcal{Q}$ gilt. Dabei sind $G[A]$ und $G[B]$ die durch die Mengen A bzw. B induzierten Teilgraphen in G. Chaplick et al. setzten $\mathcal{P} = \mathcal{Q}$ auf die Klasse der linearen Wälder und bewiesen somit die NP-Schwere. Es wurde ebenfalls gezeigt, dass das analoge Problem für Ebenen NP-schwer ist. Aus dem zur Reduktion verwendeten Gadget folgt sogar, dass es bereits für Graphen mit einem Maximalgrad von 6 NP-schwer ist zu entscheiden, ob die Knoten und Kanten des gegebenen Graphen im $\mathbb{R}^3$ durch zwei Ebenen überdeckt werden können.

1.1 Grundlagen und Definitionen

Wir wollen nun einige Grundlagen festlegen, die für den Rest der Arbeit gültig sind. Sei G ein Graph und V die Menge seiner Knoten. Ein *linearer Baum* ist ein Baum, dessen Maximalgrad 2 ist, also ein Pfad. Wir werden häufig von nicht-linearen Bäumen reden und meinen damit einen Baum, der mindestens einen Knoten mit mehr als zwei Nachbarn enthält. Ein *linearer Wald* ist ein Wald, der aus linearen Bäumen besteht. Anders gesagt, ist ein linearer Wald eine Menge von Pfaden.

Definition 1. *Die* lineare Knotenarborizität *vom G ist die kleinste Anzahl an Teilmengen, in die V aufteilbar ist, sodass jede von ihnen einen linearen Wald induziert. Sei* lva(G) *die lineare Knotenarborizität von G.*

Zur Vereinfachung werden wir von Knotenfärbungen reden und meinen damit die Teilmengenzugehörigkeit. Dabei sind Knoten, die gleich gefärbt sind, stets in der selben Teilmenge und Knoten, die unterschiedlich gefärbt sind, in verschiedenen Teilmengen; siehe Abbildungen 1.2a und 1.2b.

Somit dürfen Knoten höchstens zwei Nachbarn ihrer eigenen Farbe haben. Sonst würden sie zusammen mit ihren gleich gefärbten Nachbarn einen nicht-linearen Baum induzieren und damit die Eigenschaft der linearen Knotenarborizität verletzen. Ferner darf es keinen Kreis geben, der nur aus Knoten der gleichen Farbe besteht. Wir bezeichnen einen Kreis, der aus drei Knoten besteht, als Dreieck.

Da für uns insbesondere die Fragestellung, ob ein Graph G eine schwache Geradenüberdeckungszahl von 2 hat, interessant ist, definieren wir es als Entscheidungsproblem. Dazu wird die Tatsache ausgenutzt, dass die schwache Geradenüberdeckungszahl von G und $\text{lva}(G)$ übereinstimmen.

2-Linear Vertex-Arboricity	
Gegeben:	Graph G mit Knotenmenge V.
Entscheide:	Kann V in zwei disjunkte Teilmengen aufgeteilt werden, sodass jede von ihnen einen linearen Wald in G induziert?

Ebenso soll an dieser Stelle bereits die allgemeine Version des Problems für c Teilmengen definiert werden.

c-Linear Vertex-Arboricity	
Gegeben:	Graph G mit Knotenmenge V.
Entscheide:	Kann V in c disjunkte Teilmengen aufgeteilt werden, sodass jede von ihnen einen linearen Wald in G induziert?

1.2 Eigener Beitrag

Da der Beweis von Farrugia [Far04] nur auf allgemeine Graphen gültig ist, soll in dieser Arbeit die Komplexität von 2-Linear Vertex-Arboricity auf planaren Graphen untersucht werden. Es wird bewiesen, dass 2-Linear Vertex-Arboricity bereits für planare Graphen (mit Maximalgrad 6) NP-vollständig ist. Ferner wird gezeigt, dass dies auf allgemeinen Graphen bereits für Maximalgrad 5 gilt. Matsumoto [Mat90] hat bereits bewiesen, dass alle Graphen mit Maximalgrad 4, die nicht isomorph zu K_5 sind, eine lineare Knotenarborizität von höchstens 2 haben. Somit können ihre Knoten durch zwei Geraden im $\mathbb{R}^3$ überdeckt werden. Dadurch ist die Lücke zwischen Maximalgrad 4 und 5 auf allgemeinen Graphen geschlossen. Außerdem wird das Problem als ganzzahliges lineares Programm modelliert. Mit diesem ist es möglich, für einen gegebenen Graphen das 2-Linear Vertex-Arboricity Problem zu entscheiden. Der resultierende Algorithmus ist natürlich nicht effizient. Im positiven Fall stellen die Werte der Variablen des ganzzahligen linearen Programms eine gültige Aufteilung der Knotenmenge in zwei Teilmengen dar. Dabei induzieren beide Teilmengen einen linearen Wald. Im Anschluss beschäftigen wir uns noch mit der parametrisierten Komplexität des Problems. Wir zeigen, dass 2-Linear Vertex-Arboricity und c-Linear Vertex-Arboricity mit den Parametern Knotenüberdeckungszahl und Baumweite parametrisierbar sind.

2 NP-Vollständigkeit von 2-Linear Vertex-Arboricity auf eingeschränkten Graphenklassen

In diesem Kapitel wird bewiesen, dass das 2-Linear Vertex-Arboricity Problem bereits auf planaren Graphen, planaren Graphen mit Maximalgrad 6 und allgemeinen Graphen mit Maximalgrad 5 NP-vollständig ist. Dazu reduzieren wir von verschiedenen planaren 3-SAT Problemen von denen bekannt ist, dass sie NP-vollständig sind. Dabei besteht ein planares 3-SAT Problem stets aus einer planaren booleschen Formel ϕ und dem zugehörigen planaren bipartiten Inzidenzgraph G. Für jede Variable v_i und jede Klausel c_i in ϕ hat G einen Knoten v_i bzw. c_i. Kommt eine Variable in einer Klausel vor, so sind die entsprechenden Knoten in G mit einer Kante verbunden.

2.1 Komplexität auf planaren Graphen

Im Folgenden soll das Clause-Linked Planar 3-SAT Problem, welches NP-vollständig ist, auf das Problem 2-Linear Vertex-Arboricity auf planaren Graphen reduziert werden. Wir übernehmen die Definition von Fellows et. al. [FKMP95].

Clause-Linked Planar 3-SAT	
Gegeben:	Boolesche 3-KNF-Formel ϕ und der zugehörige bipartite planare Inzidenzgraph $G(\phi)$ mit Einbettung. Die Klausel-Knoten in G erlauben eine lineare Anordnung, sodass jeweils zwei Klausel-Knoten durch eine zusätzliche Kante verbunden werden können, ohne die Planarität von G zu verletzen.
Entscheide:	Ist ϕ erfüllbar?

Bei der Reduktion soll (ϕ, G) in polynomieller Zeit so in einen planaren Graphen H umgewandelt werden, dass eine gültige Aufteilung der Knoten von H in zwei Teilmengen, die jeweils einen linearen Wald in H induzieren, eine erfüllende Belegung für ϕ impliziert.

2.1.1 Variablen-Gadget

Zuerst entwickeln wir das Variablen-Gadget. Jede Variable aus ϕ bzw. jeder Variabel-Knoten aus G wird durch ein solches Gadget ersetzt. Sei G_a das Variablen-Gadget der Variable a. Dieses enthält einen Knoten a, der zu allen Knoten von zwei Dreiecken $1, 2, 3$ und $4, 5, 6$ benachbart ist, siehe Abbildung 2.1.

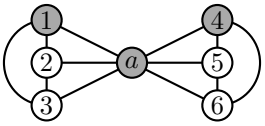


Abb. 2.1: Variablen-Gadget G_a

Seien T_1 und T_2 die beiden Teilmengen, in die die Knoten aufgeteilt werden. Ohne Beschränkung der Allgemeinheit soll festgelegt werden, dass Knoten a in T_1 liegt. Zur Vereinfachungen sei festgelegt, dass Knoten aus T_1 grau gefärbt und Knoten aus T_2 weiß gefärbt werden. Da $1, 2, 3$ ein Dreieck ist, muss mindestens einer der Knoten von $1, 2, 3$ grau sein. Sonst würden die Knoten $1, 2, 3$ einen monochromen Kreis induzieren und somit die Eigenschaft verletzen, dass alle induzierten Teilgraphen Pfade sein müssen. Dasselbe gilt für das Dreieck $4, 5, 6$. Somit ist sichergestellt, dass mindestens zwei Nachbarn von a grau gefärbt sind, z. B. Knoten 1 und 4. Außerdem müssen jeweils genau zwei der Knoten aus den Dreiecken weiß sein. Sonst hätte a mehr als zwei graue Nachbarn und ein nicht-linearer grauer Baum wäre induziert. Knoten a hat also genau zwei graue Nachbarn.

Diese Konstruktion hat den Vorteil, dass alle Knoten, die zusätzlich an a gehängt werden, weiß sein müssen. (Sonst würden die Knoten $1, 4, a$ und der zusätzliche Knoten einen nicht-linearen Baum induzieren). Dadurch kann eine Negation, wie in Abbildung 2.2 dargestellt, konstruiert werden. Es ist sichergestellt, dass der einzige Nachbar von $\overline{a}$ immer anders gefärbt ist als $\overline{a}$ selbst. (Somit kann $\overline{a}$ für weitere Konstruktionen so behandelt werden, als hätte er keine Nachbarn).

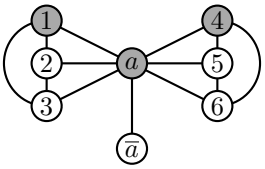
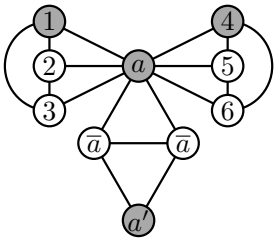
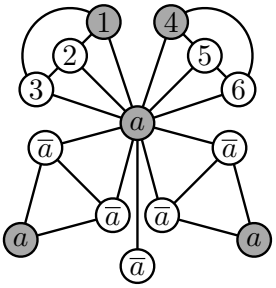


Abb. 2.2: G_a mit einer Negation

Da eine Variable in mehreren Klauseln vorkommen kann, werden auch mehrere Kopien von a benötigt. Außerdem muss sichergestellt werden, dass alle Kopien eines a-Knoten im gesamten Graph die selbe Farbe, wie der originale a-Knoten haben. Eine Kopie wird dadurch erzeugt, dass zwei Negationen zusammen mit dem Kopie-Knoten ein Dreieck bilden, siehe Abbildung 2.3a. Sei a' eine Kopie des a-Knotens. Da im Dreieck $\overline{a}, \overline{a}, a'$ die beiden $\overline{a}$-Knoten die gleiche Farbe haben müssen, muss a' eine andere haben. Diese ist die gleiche Farbe, die der originale a-Knoten hat. Es können beliebig viele Kopien oder Negationen an den originalen Knoten angehängt werden. Ein möglicher Aufbau ist in Abbildung 2.3b dargestellt.



(a) G_a mit einer Kopie a' des a-Knotens



(b) G_a mit zwei Kopien und einer Negation

Abb. 2.3

2.1.2 Klausel-Gadget

In diesem Abschnitt soll das Klausel-Gadget entwickelt werden. Für jede Klausel aus ϕ bzw. jeden Klausel-Knoten in G gibt es in H genau ein Klausel-Gadget. Dieses enthält Gadgets für die Variablen, die in dieser Klausel vorkommen. Es wird die Tatsache ausgenutzt, dass ein Kreis die Eigenschaft der linearen Knotenarborizität zerstört, wenn alle seine Knoten die selbe Farbe haben. Da die Variablen-Knoten von G in beliebiger Reihenfolge an einem Klausel-Knoten von G hängen können, muss das Gadget alle möglichen Reihenfolgen erlauben. Ferner soll es möglich sein, alle Variablen-Knoten im Klausel-Gadget mit einer der beiden Farben zu färben, aber nicht mit der anderen. Außerdem muss diese Farbe für alle Klausel-Gadgets gleich sein. Dies ist äquivalent zu der Tatsache, dass z. B. bei der Klausel $(a \vee b \vee c)$ zwar allen Variablen gleichzeitig der Wert wahr zugeordnet werden kann, aber nicht allen der Wert falsch.

Dies wird durch 0-Knoten erreicht. Sie werden so konstruiert, dass alle 0-Knoten im gesamten Graph H immer dieselbe Farbe haben müssen. Ebenso existieren $\overline{0}$-Knoten, die die Eigenschaft haben, alle mit genau der andere Farbe gefärbt sein zu müssen als die 0-Knoten. Ein Beispiel ist in Abbildung 2.4 dargestellt. So würde ein Klausel-Gadget in H aussehen, das einem Klausel-Knoten in G entspricht. Dabei wäre die Kantenreihenfolge im Uhrzeigersinn im ursprünglichen Klausel-Knoten von G: Variablen-Knoten a, Klausel-Knoten C_1, Variablen-Knoten c, Variablen-Knoten b, Klausel-Knoten C_3. Abbildung 2.7 soll dies verdeutlichen.

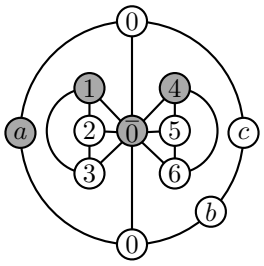


Abb. 2.4: Klausel-Gadget

Da die Klausel-Knoten in G so angeordnet werden können, dass jeweils ein Klausel-

Knoten eine Kante zum nächsten Klausel-Knoten hat, sollen diese Kanten genutzt werden, um die Färbung der 0-Knoten „weiterzuleiten“. Dadurch wird sichergestellt, dass alle 0-Knoten von H gleich gefärbt werden. Die Negation aus Abbildung 2.2 funktioniert auch in umgekehrter Reihenfolge. Diese Tatsache nutzen wir zur Konstruktion eines Negations-Gadgets aus. In Abbildung 2.5 ist ein solches Gadget dargestellt. Diese hat den Vorteil, dass der negierte Knoten mit zwei Knoten seiner Farbe benachbart sein muss.

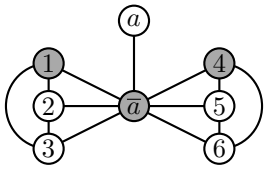


Abb. 2.5: Negations-Gadget mit der Eigenschaft, dass der negierte Knoten genau zwei Nachbarn seiner eigenen Farbe hat.

Die Knoten a und $\overline{a}$ müssen unterschiedlich gefärbt werden. Angenommen sie wären gleich gefärbt. Da Knoten $1, 2, 3$ ein Dreieck bilden, müsste mindestens einer davon die gleiche Farbe wie a und $\overline{a}$ haben. Das gleiche gilt für das Dreieck $4, 5, 6$. Dadurch hätte $\overline{a}$ drei Nachbarn seiner eigenen Farbe und würde mit ihnen zusammen einen monochromen nicht-linearen Baum induzieren. Somit müssen a und $\overline{a}$ unterschiedliche Farben haben. Ferner hat $\overline{a}$ genau zwei Nachbarn seiner eigenen Farbe.

Wir wollen nun eine Weiterleitung der 0-Knoten-Farbe von einem Klausel-Gadget zum nächsten zeigen. Dazu soll zuerst sichergestellt werden, dass der untere und obere 0-Knoten in einem Klausel-Gadget die gleiche Farbe haben muss. Dies wird mit Hilfe eines Negations-Gadgets innerhalb des Klausel-Gadgets gewährleistet; siehe Abbildung 2.4. Ohne Beschränkung der Allgemeinheit sei festgelegt, dass der obere 0-Knoten weiß ist. Er wird mit dem $\overline{0}$-Knoten des Negations-Gadgets verbunden und dieser wird mit dem unteren 0-Knoten des Klausel-Gadgets verbunden. Der $\overline{0}$-Knoten muss somit grau sein und der untere 0-Knoten weiß. Folglich müssen beide 0-Knoten in einem Klausel-Gadget gleich gefärbt sein.

Die Weiterleitung der 0-Knoten-Farbe zwischen zwei Klausel-Gadgets funktioniert auf die gleiche Weise. Das Negations-Gadget befindet sich nun zwischen den beiden Klausel-Gadgets. Der untere 0-Knoten des ersten Klausel-Gadgets wird mit dem $\overline{0}$-Knoten des Negations-Gadget verbunden und dieser mit dem oberen 0-Knoten des zweiten Klausel-Gadgets. Dies ist in Abbildung 2.6 dargestellt.

Somit wurde gezeigt, dass alle 0-Knoten im gesamten Graph H die gleiche Farbe haben müssen und H planar ist.

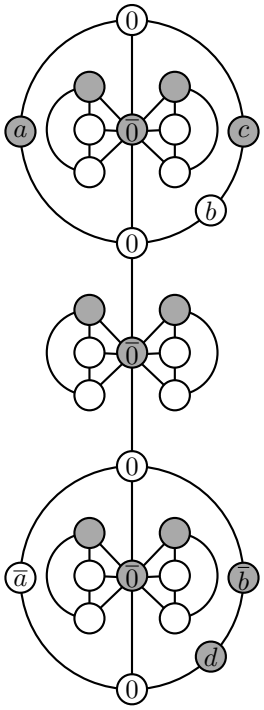


Abb. 2.6: Weiterleitung der 0-Knoten-Farbe zwischen zwei Klausel-Gadgets mit Hilfe eines Negations-Gadgets.

2.1.3 Vollständige Konstruktion und Beweis

Im Folgenden wird der komplette Graph H aus G und der Einbettung konstruiert. Jeder Variablen-Knoten v aus G wird durch ein Gadget G_v, wie in Abschnitt 2.1.1 beschrieben, ersetzt. Jeder Klausel-Knoten aus G wird durch ein Klausel-Gadget aus Abschnitt 2.1.2 ersetzt. Die ursprünglichen Kanten zwischen Klausel-Knoten und Variablen-Knoten in G enthalten nun die Kanten, der neu erzeugten Kopie-Knoten einer Variable bzw. die Negationen dieser Variable. Diese Knoten liegen selber im Klausel-Gadget und bilden zusammen mit den 0-Knoten einen Ring.

Ein Beispiel ist in Abbildung 2.8 gegeben. Es ist der neu konstruierte Graph H für $\phi = (a \vee b \vee c) \wedge (\overline{a} \vee d \vee \overline{b})$ und der Einbettung für G aus Abbildung 2.7. An dieser Stelle soll ignoriert werden, dass alle Formeln mit zwei Klauseln und vier Variablen immer erfüllbar sind.

Satz 2. *Es ist NP-vollständig zu entscheiden, ob ein gegebener planarer Graph lineare Knotenarborizität 2 hat.*

Beweis. Es kann in polynomieller Zeit festgestellt werden, ob eine gegebene Aufteilung der Knoten in Teilmengen gültig ist. Daraus folgt die Mitgliedschaft in NP. Die Reduktion erfolgt, wie in Kapitel 2.1 beschrieben. Sie kann in polynomieller Zeit durchgeführt werden. Der neu entstandene Graph H ist planar.

Sei (ϕ, G) eine Instanz von Clause-Linked Planar 3-SAT und sei H eine Instanz von 2-Linear Vertex-Arboricity. Es muss gezeigt werden, dass genau dann, wenn (ϕ, G) eine Ja-Instanz von Clause-Linked Planar 3-SAT ist, und damit erfüllbar ist, dann ist H eine Ja-Instanz für das 2-Linear Vertex-Arboricity Problem.

Sei eine Ja-Instanz von Clause-Linked Planar 3-SAT gegeben. Das bedeutet, es gibt für jede Variable v_i in ϕ einen Wahrheitswert, sodass die Formel ϕ erfüllt ist. Für jede Variable v_i aus (ϕ, G) gibt es Knoten im Graph H, die dieser entsprechen. Alle Kopien eines v_i Knotens in H, die in (ϕ, G) mit wahr belegt wurden, gehören zur Teilmenge T_1. Alle Kopien eines v_i Knotens in H, die mit falsch belegt wurden, gehören zur Teilmenge T_2. Alle $\overline{v_i}$ Knoten in H werden in die jeweils andere Teilmenge, als v_i eingeordnet. Aus den zwei Dreiecken, die an einem Variablen-Gadgets G_{v_i} befestigt sind, wird jeweils genau ein Knoten der selben Teilmenge zugeordnet wie v_i. Die anderen beiden Knoten werden der anderen Teilmenge zugeordnet. Für die 0-Knoten und $\overline{0}$-Knoten müssen beide Teilmengen ausprobiert werden. Dies ist kein Problem, da in polynomieller Zeit festgestellt werden kann, ob eine Aufteilung gültig ist. Dadurch wird allen Eingangs- und Ausgangsknoten eines Negation-Gadgets eine Teilmenge zugeordnet. Die anderen Knoten im Gadget sind immer so in zwei Teilmengen aufteilbar, dass keine Eigenschaft der linearen Knotenarborizität verletzt wird. Dadurch wurde jedem Knoten in H eine Teilmenge zugewiesen. Beide Teilmengen induzieren einen linearen Wald. Also ist die lineare Knotenarborizität von H gleich 2.

Sei eine Ja-Instanz für das Entscheidungsproblem 2-Linear Vertex-Arboricity gegeben. Das bedeutet, jedem Knoten aus H wurde eine Teilmenge T_1 oder T_2 zugeordnet, sodass die beiden Teilmengen einen linearen Wald induzieren. Es ist sichergestellt, dass ein Knoten a sich immer in einer anderen Teilmenge als $\overline{a}$ befindet. Außerdem ist durch den Klausel-Ring sichergestellt, dass alle Knoten der Klausel zwar gleichzeitig einer Teilmenge, nicht aber der anderen Teilmenge zugeteilt werden können. Die Teilmenge, für die es unmöglich ist, ist genau die Teilmenge, in der sich die 0-Knoten befinden. Es werden nun alle Variablen aus ϕ, deren Knoten in H in T_1 liegt, auf den Wert wahr gesetzt; alle anderen auf den Wert falsch. Falls dies ϕ nicht erfüllt, werden die Wahrheitswerte für alle Variablen vertauscht. Dies ist in polynomieller Zeit möglich.

Es wurde eine gültige Reduktion gezeigt. Somit ist das Problem zu entscheiden, ob die lineare Knotenarborizität eines planaren Graphen 2 ist und damit 2-Linear Vertex-Arboricity, NP-vollständig.

□

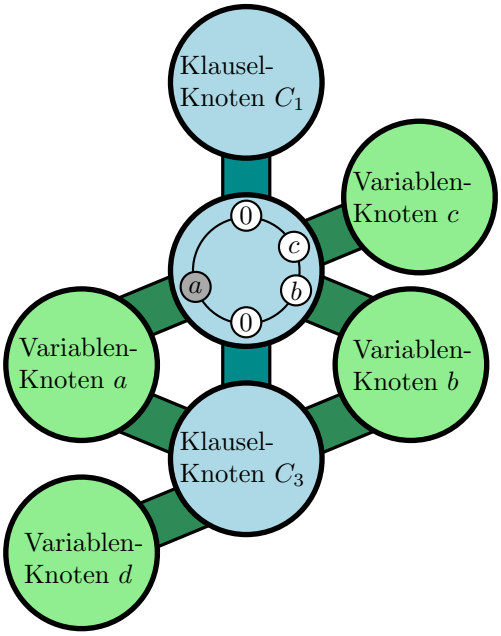


Abb. 2.7: Graph G mit einem eingesetzten Klausel-Gadget

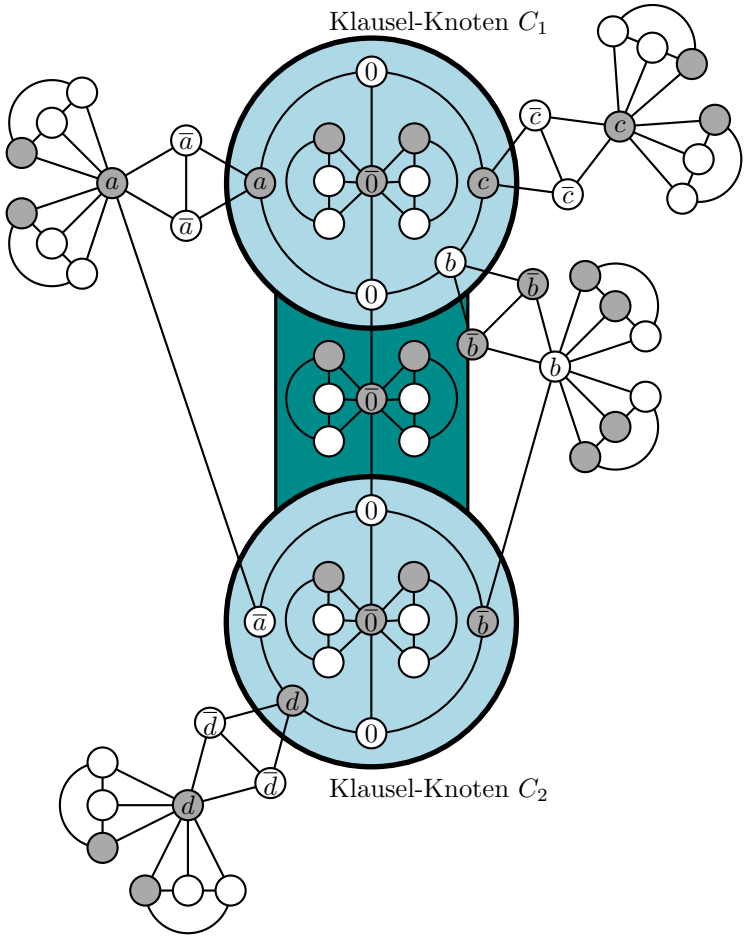


Abb. 2.8: Der Graph H für $\phi = (a \vee b \vee c) \wedge (\overline{a} \vee d \vee \overline{b})$ unter Berücksichtigung der Einbettung von G

2.2 Komplexität auf planaren Graphen mit Maximalgrad 6

In diesem Abschnitt wird gezeigt, dass das 2-Linear Vertex-Arboricity Problem aus Satz 2 auch für planare Graphen mit Maximalgrad 6 NP-vollständig ist. Dazu werden neue Gadgets konstruiert. Wir müssen sicherstellen, dass die Knotengrade in den Gadgets höchstens 6 sind. Der Grund dafür, dass die Knotengrade der Gadgets in Abschnitt 2.1 beliebig groß sein können, ist, dass eine Variable in ϕ in beliebig vielen Klauseln vorkommen kann.

Es wird also eine Variante des planaren 3-SAT Problems benötigt, die die Anzahl der Klauseln, in der eine Variable vorkommen darf, beschränkt und NP-vollständig ist. Fellows et. al. [FKMP95] haben bewiesen, dass Clause-Linked-Planar-3-Sat selbst unter zusätzlichen Einschränkungen NP-vollständig bleibt. Wir bezeichnen dieses Problem als Clause-Linked-Planar-Exactly-3-Bounded-3-SAT.

Clause-Linked-Planar-Exactly-3-Bounded-3-SAT	
Gegeben:	Boolesche 3-KNF-Formel ϕ' und der zugehörige bipartite planare Inzidenzgraph $G'(\phi')$ mit Einbettung. Die Klausel-Knoten in G' erlauben eine lineare Anordnung, sodass jeweils zwei Klausel-Knoten durch eine zusätzliche Kante verbunden werden können, ohne die Planarität von G' zu verletzen. Jede Variable aus ϕ' muss in genau drei Klauseln auftauchen, davon einmal negativ und zweimal positiv. Jede Klausel hat zwei oder drei Variablen. Falls eine Klausel drei Variablen hat, so sind sie alle positiv in dieser Klausel.
Entscheide:	Ist ϕ' erfüllbar?

Wieder soll bei der Reduktion (ϕ', G') in polynomieller Zeit so in einen planaren Graphen H' umgewandelt werden, dass eine gültige Aufteilung der Knoten von H' in zwei Teilmengen, die jeweils einen linearen Wald in H' induzieren, eine erfüllende Belegung für ϕ' impliziert. Außerdem ist der Maximalgrad von H' höchstens 6.

2.2.1 Variablen-Gadget

Wieder beginnen wir damit, ein Variablen-Gadget zu konstruieren. Dieses soll sicherstellen, dass immer drei Variablen-Knoten existieren, von denen zwei in einer und einer in genau der andere Teilmenge liegen müssen. Diese entsprechen den drei Variablen in Clause-Linked-Planar-Exactly-3-Bounded-3-SAT und der Tatsache, dass zwei von ihnen positiv und eine negiert vorkommen müssen. Zuerst erstellen wir ein Basis-Gadget B, das uns sowohl als Grundgerüst für das Variablen-Gadget, als auch zur Weiterleitung der 0-Knoten dienen soll; siehe Abbildung 2.9

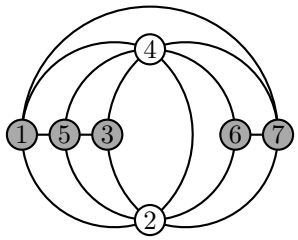


Abb. 2.9: Basis-Gadget B

Es gibt nur eine Möglichkeit die Knoten in B so in zwei Teilmengen aufzuteilen, dass beide einen linearen Wald induzieren. Zum besseren Verständnis werden wir im Folgenden wieder von Färbungen reden und meinen damit die Zugehörigkeit zu den beiden Teilmengen. Seien T_1 und T_2 die beiden Teilmengen und seien Knoten aus T_1 grau gefärbt und Knoten aus T_2 weiß gefärbt. Ferner sei ohne Beschränkung der Allgemeinheit festgelegt, dass Knoten 1 in T_1 liegt und somit grau ist. Die Knoten 1, 3, 5, 6, 7 müssen alle grau sein. Die Knoten 2 und 4 müssen weiß sein.

Aus dem Dreieck 2, 3, 4 muss genau ein Knoten grau sein. Knoten 3 und 4 können nicht beide gleichzeitig grau sein, da es sonst keine gültige Färbung für die Knoten im

Dreieck 2, 6, 7 gäbe. Wären Knoten 3 und 4 beide grau, so müsste Knoten 2 weiß sein, da 2, 3, 4 sonst ein graues Dreieck wäre. Da Knoten 4 dann bereits zwei graue Nachbarn (Knoten 1 und Knoten 3) hätte , müssten Knoten 6 und 7 weiß gefärbt werden. (Sonst wäre ein nicht linearer Baum induziert.) Dadurch würden die Knoten 2, 6, 7 ein weißes Dreieck bilden. Somit dürfen Knoten 3 und 4 nicht gleichzeitig grau sein.

Ebenso können Knoten 2 und 4 nicht beide gleichzeitig grau sein, da sonst Knoten 1, 2, 4 ein graues Dreieck bilden würden.

Knoten 2 und 3 können nicht gleichzeitig grau sein, da sonst Knoten 2 bereits zwei graue Nachbarn (Knoten 1 und 3) hätte. Dies würde implizieren, dass Knoten 4, 6 und 7 weiß sein müssten, da sie sonst zusammen mit Knoten 2 einen nicht-linearen Baum induzieren würden. Dadurch wäre 4, 6, 7 ein weißes Dreieck. Also dürfen Knoten 2 und 3 nicht gleichzeitig grau sein.

Somit muss genau ein Knoten aus dem Dreieck 2, 3, 4 zusammen mit Knoten 1 grau sein. Es kann nicht Knoten 2 sein, weil Knoten 5 weiß sein müsste, da sonst Knoten 1, 2, 5 ein graues Dreieck bilden würden. Da Knoten 3 und 4 ebenso weiß wären, würden sie zusammen mit Knoten 5 ein weißes Dreieck bilden.

Ebenso kann Knoten 4 nicht der grau gefärbte Knoten aus dem Dreieck 2, 3, 4 sein. Da Knoten 5 grau sein müsste, weil sonst die Knoten 2, 3, 5 ein weißes Dreieck bilden würden. Dadurch würde Knoten 5 zusammen mit den Knoten 1 und 4 ein graues Dreieck bilden.

Somit muss Knoten 3 grau gefärbt werden und Knoten 2 und 4 weiß. Daraus folgt, dass Knoten 5 ebenfalls grau ist. Da 2, 4, 6 und 2, 4, 7 Dreiecke sind, müssen Knoten 7 und 6 in weiß sein.

Dadurch wurde gezeigt, dass dies die einzig gültige Aufteilung der Knoten in zwei Teilmengen ist, ohne die Eigenschaft der linearen Knotenarborizität zu verletzen.

Die Eigenschaft, die ausgenutzt werden soll, ist, dass Knoten 1 und 7 beide zwei graue Nachbarn haben. Somit muss jeder weitere an Knoten 1 oder 7 angehängte Knoten weiß sein. Dadurch ist es möglich, zusätzlich drei Knoten so anzuhängen, dass zwei der Knoten eine und der dritte Knoten die andere Farbe haben müssen. Sei a eine Variable in $\phi^{'}$. Wir bezeichnen das Variablen-Gadget von a als G_a^6. An Knoten 1 werden zwei weitere Knoten angehängt, diese sind die zwei a-Knoten und müssen beide weiß gefärbt sein. An Knoten 7 werden ebenfalls zwei weitere Knoten ($a^{'}$, $a^{''}$) angehängt, die ebenfalls weiß sein müssen. Knoten $a^{'}$ und $a^{''}$ werden miteinander und einem weiteren Knoten $\overline{a}$ verbunden. Da diese drei Knoten ein Dreieck bilden, muss $\overline{a}$ eine andere Farbe als $a^{'}$ und $a^{''}$ haben. Das Variablen-Gadget ist planar und hat einen Maximalgrad von 6. In Abbildung 2.10 ist G_a^6 dargestellt.

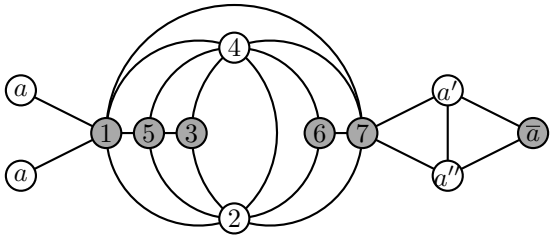


Abb. 2.10: Variablen-Gadget G_a^6 für Variable a aus ϕ'

Die zwei äußeren a-Knoten und der äußere $\overline{a}$-Knoten erfüllen alle geforderten Eigenschaften. Die beiden a-Knoten müssen in einer und der $\overline{a}$-Knoten in genau der anderen Teilmenge liegen. Ferner sind sie nur mit Knoten aus der jeweils anderen Teilmenge verbunden. Dadurch ist es möglich, sie in einem Klausel-Gadget zu verwenden ohne, dass ihre Nachbarn stören könnten. Für jede Variable aus ϕ' erstellen wir ein solches Variablen-Gadget mit drei Variablen-Knoten und platzieren diese wieder in einen Klausel-Ring.

2.2.2 Klausel-Gadget

Das Klausel-Gadget wird ähnlich wie in Abschnitt 2.1.2 aufgebaut. Das Grundgerüst des Klausel-Gadgets ist in Abbildung 2.11 dargestellt. Das Basis-Gadget aus dem vorherigen Abschnitt wird dabei wiederverwendet um die 0-Knoten weiterzuleiten. Zuerst soll wieder sichergestellt werden, dass die beiden 0-Knoten eines Klausel-Gadgets immer die selbe Farbe haben. Dazu wird das Basis-Gadget um 90 Grad gedreht und in die Mitte des Klausel-Gadgets platziert; siehe Abbildung 2.11. Ohne Beschränkung der Allgemeinheit sei angenommen, der obere 0-Knoten des Klausel-Gadgets ist weiß gefärbt. Er wird mit Knoten 1 des Basis-Gadgets, der sich direkt unter ihm befindet, verbunden. Da Knoten 1 immer zwei Nachbarn seiner eigenen Farbe (Knoten 5 und 7) hat, muss er grau sein. (Sonst würde der obere 0-Knoten, Knoten 1, Knoten 5 und Knoten 7 einen weißen nicht-linearen Baum induzieren.) Im vorherigen Abschnitt wurde bereits gezeigt, dass die restliche Färbung des Basis-Gadgets impliziert ist. Somit müssen Knoten 1, 3, 5, 6, 7 grau und Knoten 2 und 4 weiß sein. Knoten 7 hat zwei graue Nachbarn. Also müssen alle weiteren Knoten, die mit ihm verbunden sind, weiß sein. Diese Tatsache nutzen wir für die Weiterleitung der 0-Knoten-Farbe aus. Dazu wird Knoten 7 mit dem unteren 0-Knoten des Klausel-Rings verbunden. Dieser muss dadurch weiß sein. Somit müssen beide Knoten eines Klausel-Gadgets die gleiche Farbe haben.

Jetzt muss noch gezeigt werden, dass alle 0-Knoten in H' weiß sein müssen. Wieder soll das Basis-Gadget aus dem vorherigen Abschnitt verwendet werden. Dieses Mal um die 0-Knoten-Farbe zwischen den Klausel-Gadgets weiterzuleiten. Da von Clause-Linked-Planar-Exactly-3-Bounded-3-SAT reduziert wird, muss es möglich sein, die Klausel-Gadgets so anzuordnen, dass eine Weiterleitung der 0-Knoten-Farbe zwischen ihnen existieren kann, ohne die Planarität zu zerstören.

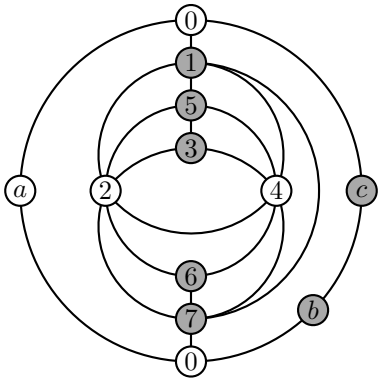


Abb. 2.11: Klausel-Gadget

Zur Weiterleitung zwischen den Klausel-Gadgets wird jeweils ein Basis-Gadget zwischen zwei Klausel-Gadgets platziert. Der untere 0-Knoten des oberen Klausel-Gadgets wird mit Knoten 1 des Basis-Gadgets verbunden, siehe Abbildung 2.12. Wieder ist dadurch die Färbung im Basis-Gadget impliziert. Knoten 7 wird mit dem oberen 0-Knoten des nächsten Klausel-Gadgets verbunden. Dadurch muss dieser die gleiche Farbe wie die 0-Knoten im vorherigen Klausel-Gadget haben. Somit lässt sich eine Weiterleitung der 0-Knoten-Färbung durch den gesamten Graph sicherstellen. Das führt dazu, dass alle 0-Knoten immer gleich gefärbt sein müssen.

Die Variablen-Gadgets und Klausel-Gadgets werden ähnlich wie in Abbildung 2.7 und Abbildung 2.8 in den ursprünglichen Graphen $G^{'}$ platziert. Der Beweis für die Gültigkeit der Reduktion ist analog zu Satz 2. Somit ist das 2-Linear Vertex-Arboricity Problem auch für planare Graphen mit einem Maximalgrad 6 NP-vollständig.

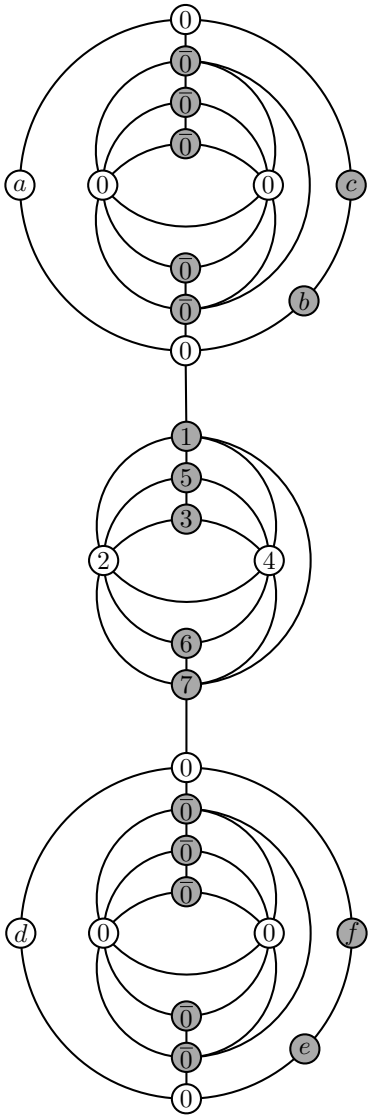


Abb. 2.12: Weiterleitung der 0-Knoten-Farbe zwischen zwei Klausel-Gadgets

2.3 Komplexität auf Graphen mit Maximalgrad 5

In diesem Abschnitt wird gezeigt, dass das 2-Linear Vertex-Arboricity Problem aus Satz 2 auch für Graphen mit Maximalgrad 5 NP-vollständig ist. Matsumoto [Mat90] hat bereits bewiesen, dass Graphen mit Maximalgrad 4 immer eine lineare Knotenarborizität von 2 haben müssen. Die einzige Ausnahme ist der vollständige Graph K_5. Seine lineare Knotenarborizität beträgt 3. Wie in Abschnitt 2.2, werden neue Variablen- und Klausel-Gadgets konstruiert. Diese haben diesmal einen Maximalgrad von 5, sind aber nicht mehr planar. Wir wollen wieder von Clause-Linked-Planar-Exactly-3-Bounded-3-SAT reduzieren und bezeichnen eine Instanz davon als (ϕ', G').

2.3.1 Variablen-Gadget

Sei K_5^- der maximal planare Graph, der durch das Löschen einer Kante aus K_5 entsteht; siehe Abbildung 2.13. Da die lineare Knotenarborizität von K_5^- 2 ist, können die Knoten so in zwei Teilmengen aufgeteilt werden, dass beide einen linearen Wald induzieren. Allerdings müssen die beiden Knoten mit Grad 3 (Knoten 1 und 2) in der selben Teilmenge liegen. Wäre das nicht der Fall, würde eine zusätzliche Kante zwischen den beiden Knoten nichts an der linearen Knotenarborizität ändern und implizieren, dass K_5 eine lineare Knotenarborizität von 2 hat. Diese Tatsache nutzen wir zur Konstruktion eines Variablen-Gadgets aus und reden im Folgenden wieder von Farben und meinen damit die Teilmengenzugehörigkeit.

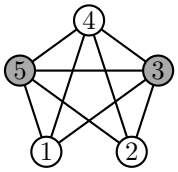


Abb. 2.13: Graph K_5^- (nicht planare Zeichnung)

Ohne Beschränkung der Allgemeinheit sei festgelegt, dass die beiden Grad-3-Knoten 1 und 2 weiß gefärbt sind. Das Dreieck aus den Knoten 3, 4, 5 muss ebenfalls einen weißen Knoten enthalten. Allerdings ist noch nicht eindeutig, welcher der weiße Knoten sein muss. Das Ziel ist es wieder drei Knoten zu erzeugen, von denen zwei gleich und einer unterschiedlich gefärbt sein müssen. Ferner müssen die Knoten alle höchstens Grad 3 haben (um sie im Klausel-Ring einbauen zu können) und dürfen nur mit anders gefärbten Knoten benachbart sein. Dazu erweitern wir K_5^-, wie in Abbildung 2.14 dargestellt.

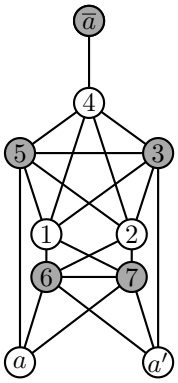


Abb. 2.14: Variablen-Gadget der Variable a

Knoten 6 und 7 müssen grau gefärbt werden, da sie sonst jeweils mit Knoten 1, 2 und dem weißen Grad-4-Knoten einen weißen Kreis bilden würden. Knoten a und $a^{'}$ müssen beide weiß sein, da sie sonst jeweils mit Knoten 6 und 7 ein graues Dreick bilden würden. Zusätzlich verbinden wir Knoten a mit Knoten 5 und Knoten $a^{'}$ mit Knoten 3. Dadurch

ist sichergestellt, dass Knoten 3 und 5 grau sein müssen, da sonst zwei weiße nicht-lineare Bäume induziert wären. Somit wissen wir, dass Knoten 4 der weiße Grad-4-Knoten sein muss und bereits zwei weiße Nachbarn (Knoten 1 und 2) hat. Wir verbinden Knoten $\overline{a}$ mit Knoten 4. Also muss $\overline{a}$ grau gefärbt sein, da sonst die Knoten 1, 2, 4, $\overline{a}$ einen weißen nicht-linearen Baum induzieren würden.

Knoten a und $a^{'}$ müssen also immer gleich und Knoten $\overline{a}$ anders gefärbt sein. Sie haben alle einen Knotengrad von höchstens 3 und sind mit nur anders gefärbten Knoten benachbart. Dadurch ist es möglich, sie wie in Abschnitt 2.1.2 und 2.2.2 in einen Klausel-Ring mit einzubauen. Dabei stellen a und $a^{'}$ die beiden positiven und $\overline{a}$ das negative Variablenvorkommen aus $\phi^{'}$ dar.

2.3.2 Klausel-Gadget

Das Klausel-Gadget ist ähnlich, wie in Abschnitt 2.1.2 und 2.2.2 aufgebaut. Allerdings besteht es nun aus einem Ring mit drei 0-Knoten und der Weiterleitung der 0-Knoten. Die Weiterleitung unterscheidet sich, da nun ein Maximalgrad von 5 gefordert ist und der Graph nicht mehr planar sein muss. Deswegen werden keine weiteren Knoten mehr innerhalb des Rings benötigt. In Abbildung 2.15 ist die Weiterleitung dargestellt.

Es wird angenommen, dass die drei obersten 0-Knoten alle gleich gefärbt sind. Später wird gezeigt warum es so sein muss. Ohne Beschränkung der Allgemeinheit sei festgelegt, dass die oberen drei 0-Knoten des Klausel-Gadgets aus Abbildung 2.15 weiß sind. Da der mittlere 0-Knoten zwei weiße Nachbarn (die anderen beiden 0-Knoten) hat, müssen alle weiteren mit ihm benachbarten Knoten grau sein. Somit sind Knoten 1 und 3 grau. Knoten 2 bildet zusammen mit den drei 0-Knoten einen Kreis. Somit muss er grau sein, da sonst ein weißer Kreis induziert wäre.

Der linke untere 0-Knoten bildet zusammen mit den grauen Knoten 1, 2 und 3 einen Kreis und muss somit weiß sein. Das gleiche Argument gilt für den rechten unteren 0-Knoten. Knoten 2 hat bereits zwei graue Nachbarn (Knoten 1 und 3). Somit muss der mittlere untere 0-Knoten weiß sein, da sonst ein nicht-linearer grauer Baum induziert wäre. Dadurch wurde eine Weiterleitung der 0-Knoten-Farbe gezeigt.

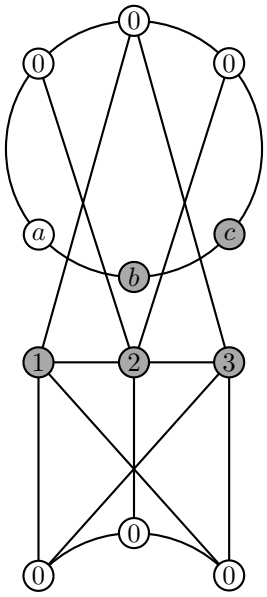


Abb. 2.15: 0-Knoten Weiterleitung für Maximalgrad 5

Wir wollen nun zeigen, wie sichergestellt werden kann, dass die drei 0-Knoten des ersten Klausel-Gadgets die gleiche Farbe haben müssen. Dazu werden drei Basis-Gadgets K_5^- verwendet. Sie werden wie in Abbildung 2.16 miteinander verbunden. Knoten 1 und 2 sind die beiden Grad-3-Knoten des ersten Basis-Gadgets. Knoten 3, 4 und 5 sind die drei Grad-4-Knoten des ersten Basis-Gadgets. Knoten 6 und 7 sind die beiden Grad-3-Knoten des zweiten und Knoten 8 und 9 die beiden Grad-3-Knoten des dritten Basis-Gadgets.

Knoten 1 und 2 müssen gleich gefärbt sein. Ohne Beschränkung der Allgemeinheit sei festgelegt, dass sie grau sind. Da Knoten 1 und 2 den selben grauen Nachbarn (Knoten 3, 4 oder 5) haben, müssen Knoten 6 und 9 weiß sein. Sonst wären zwei graue Kreise induziert. Die zwei Grad-3-Knoten in K_5^- müssen gleich gefärbt sein. Somit müssen auch Knoten 7 und 8 weiß sein. Sie werden miteinander verbunden und haben somit beide jeweils zwei weiße Nachbarn und werden mit jeweils einer Kante mit Knoten 10 und 11 verbunden. Knoten 10 und 11 werden verbunden und müssen grau sein, weil sonst nicht-lineare weiße Bäume induziert wären. Knoten 3 und 5 werden mit ihnen verbunden. Dadurch wird sichergestellt, dass Knoten 3 und 5 weiß und Knoten 4 grau sein muss, da sonst nicht-lineare graue Bäume induziert wären. Da Knoten 10 und 11 mit jeweils einem 0-Knoten (linker und rechter 0-Knoten des ersten Klausel-Gadgets) zwei Dreiecke bilden, müssen die beiden 0-Knoten weiß sein. Knoten 4 wird mit dem dritten (mittleren) 0-Knoten verbunden. Somit muss dieser weiß sein, da sonst Knoten 1, 2, 4 und der 0-Knoten einen grauen nicht-linearen Baum induzieren würden.

Zusammen mit der Weiterleitung haben alle 0-Knoten im gesamten Graph die gleiche Farbe. Die Konstruktionen verletzten zwar die Planarität, ändert aber nichts an der Tatsache, dass eine gültige Aufteilung der Knoten in zwei Teilmengen eine Lösung für Clause-Linked-Planar-Exactly-3-Bounded-3-SAT implizieren würde. Der Beweis für eine gültige Reduktion lässt sich aus Satz2 übertragen. Somit ist das

Entscheidungsproblem 2-Linear Vertex-Arboricity bereits für Graphen mit Maximalgrad 5 NP-vollständig.

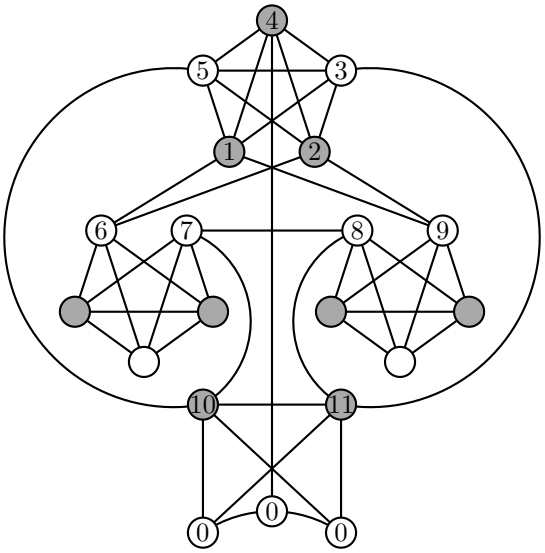


Abb. 2.16: Konstruktion aus drei K_5^-, die sicherstellt, dass die drei 0-Knoten im ersten Klausel-Gadget alle gleich gefärbt sind.

3 Modellierung von 2-Linear Vertex-Arboricity als ganzzahliges lineares Programm

In diesem Abschnitt wird 2-Linear Vertex-Arboricity als ganzzahliges lineares Programm (ILP) modelliert. Dabei stellen die (Un)gleichungen sicher, dass eine gültige Lösung des ganzzahligen linearen Programms einer gültigen Aufteilung der Knotenmenge in zwei Teilmengen entspricht. Die beiden Teilmengen induzieren jeweils lineare Wälder.

3.1 Eliminieren von Kreisen und nicht-linearen Bäumen

Am Anfang soll sichergestellt werden, dass sich ein Knoten in genau einer der beiden Teilmengen befinden muss. Seien diese T_1 und T_2. Sei jedem Knoten des Graphen eine Zahl zugeordnet, die seiner Zeile in der Adjazenzmatrix entspricht. Für jeden Knoten i sollen nun folgende Bedingungen bzw. Gleichung gelten:

$$t_{i1} \in \{0, 1\} \tag{3.1}$$

$$t_{i2} \in \{0, 1\} \tag{3.2}$$

$$t_{i1} + t_{i2} = 1 \tag{3.3}$$

Die Bedingungen 3.1 und 3.2 erzwingen einen Wert von 0 oder 1 für t_{i1} und t_{i2}. Dabei liegt ein Knoten i in T_1, wenn $t_{i1} = 1$ gilt und in T_2, wenn $t_{i2} = 1$ gilt. Durch Gleichung 3.3 ist sichergestellt, dass Knoten i in genau einer der beiden Teilmengen liegen muss. Am Ende soll es möglich sein, aus t_{i1} und t_{i2} die Zugehörigkeit eines Knotens i zu den Teilmengen T_1 und T_2 zu bestimmen.

Da die Zerlegung eines Graphen in zwei Teilmengen, die lineare Wälder induzieren, impliziert, dass eine Teilmenge keine nicht-linearen Bäume oder Kreise enthält, müssen diese durch Ungleichungen verboten werden. Sei K ein Kreis, auf dem die Knoten $k_1, k_2, \ldots, k_m$ liegen. Dabei entspricht ein Knoten k_i den Variablen t_{i1} und t_{i2}. Sei $|K| = m$ die Anzahl der Knoten auf K. Es soll nun unmöglich sein, allen Variablen $t_{11}, t_{21}, \ldots, t_{m1}$ bzw. allen Variablen $t_{12}, t_{22}, \ldots, t_{m2}$ gleichzeitig den Wert 1 oder 0 zu zuordnen. Dies wird durch folgende Ungleichungen erreicht:

$$t_{11} + t_{21} + \cdots + t_{m1} \leq |K| - 1 \tag{3.4}$$

$$t_{12} + t_{22} + \cdots + t_{m2} \leq |K| - 1 \tag{3.5}$$

Diese sind äquivalent zu der Aussage, dass auf einem Kreis mindestens einer der Knoten in einer anderen Teilmenge liegen muss als die anderen. Es ist nicht nötig durch extra Ungleichungen sicherzustellen, dass die Summe aus $t_{11}, t_{21}, \ldots, t_{m1}$ bzw. $t_{12}, t_{22}, \ldots, t_{m2}$ mindestens 1 ist. Dies wird bereits durch die Kombination aus den (Un)gleichungen 3.1, 3.2, 3.3 und 3.4, 3.5 impliziert. Für jeden Kreis im Graphen müssen also jeweils zwei solcher Ungleichungen aufgestellt werden.

Ein nicht-linearer Baum hat mindestens einen Knoten mit Knotengrad 3 oder höher. Es muss sichergestellt werden, dass für jeden Knoten und alle möglichen Kombinationen aus drei seiner Nachbarn, höchstens zwei dieser Nachbarn in der selben Teilmenge liegen wie der Knoten selber. Sei i ein Knoten und sei $\{j, k, l\}$ eine solche Kombination aus drei Nachbarn von i. Für diese würden die Ungleichungen wie folgt lauten:

$$t_{j1} + t_{k1} + t_{l1} - 2t_{i1} - 3t_{i2} \leq 0 \tag{3.6}$$

$$t_{j2} + t_{k2} + t_{l2} - 2t_{i2} - 3t_{i1} \leq 0 \tag{3.7}$$

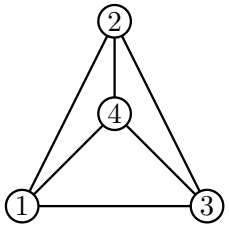
Es ist bekannt, dass von den Variablen t_{i1} und t_{i2} genau eine den Wert 1 und eine den Wert 0 hat. Durch die Subtraktion um $3t_{i2}$ am Ende der linken Seite von Ungleichung 3.6 ist es möglich, dass $t_{i2} = 1$ und $t_{j1} + t_{k1} + t_{l1} = 3$ entspricht. Dies ist äquivalent zu der Aussage, dass Knoten i in T_2 und die Knoten j, k, l in T_1 liegen. Allerdings ist es nicht möglich, dass mehr als zwei der Nachbarknoten in der selben Teilmenge liegen wie Knoten i. Angenommen Knoten i, j, k, l würden alle in T_1 liegen, dann würde das in Ungleichung 3.6 eingesetzt Folgendes ergeben:

$$1 + 1 + 1 - 2 \cdot 1 - 3 \cdot 0 \leq 0 \tag{3.8}$$

Die Ungleichung 3.8 gilt natürlich nicht. Somit wird durch die beiden Ungleichungen 3.6 und 3.7 sichergestellt, dass höchstens zwei Nachbarn eines Knotens in der selben Teilmenge wie der Knoten selber liegen.

3.2 Beispiel

Wir wollen das Ganze nun an einem Beispiel ausprobieren. Dazu betrachten wir den Graph zu K_4; siehe Abbildung 3.1a. Dieser hat vier Knoten und sieben Kreise (vier mit Länge 3 und drei mit Länge vier). Da für die Kreisungleichungen (3.4 und 3.5) nur Knoten relevant sind, müssen wir nur fünf verschiedene Kreise betrachten. (Alle Kreise der Länge 4 haben die selben Knoten und somit würden die drei Ungleichungen dieselben sein). Die Adjazenzmatrix für K_4 ist in Abbildung 3.1b dargestellt.



(a) Graph K_4.

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

(b) Adjazenzmatrix von K_4.

Abb. 3.1: K_4 und zugehörige Adjazenzmatrix.

Somit ergeben sich folgende (Un)gleichungen für die Tatsache, dass jeder Knoten von K_4 in genau einer Teilmenge liegen muss.

$$\begin{aligned} 0 \leq t_{11} \leq 1 \quad & 0 \leq t_{12} \leq 1 \quad & t_{11} + t_{12} = 1 \\ 0 \leq t_{21} \leq 1 \quad & 0 \leq t_{22} \leq 1 \quad & t_{21} + t_{22} = 1 \\ 0 \leq t_{31} \leq 1 \quad & 0 \leq t_{32} \leq 1 \quad & t_{31} + t_{32} = 1 \\ 0 \leq t_{41} \leq 1 \quad & 0 \leq t_{42} \leq 1 \quad & t_{41} + t_{42} = 1 \end{aligned}$$

Für die fünf Kreise, die wir betrachten, wird durch folgende Ungleichungen sichergestellt, dass sich nicht alle Knoten eines Kreises in der gleichen Teilmenge befinden.

$$\begin{aligned} t_{11} + t_{21} + t_{31} \leq 2 \quad & t_{12} + t_{32} + t_{42} \leq 2 \\ t_{12} + t_{22} + t_{32} \leq 2 \quad & t_{21} + t_{31} + t_{41} \leq 2 \\ t_{11} + t_{21} + t_{41} \leq 2 \quad & t_{22} + t_{32} + t_{42} \leq 2 \\ t_{12} + t_{22} + t_{42} \leq 2 \quad & t_{11} + t_{21} + t_{31} + t_{41} \leq 3 \\ t_{11} + t_{31} + t_{41} \leq 2 \quad & t_{12} + t_{22} + t_{32} + t_{42} \leq 3 \end{aligned}$$

Zum Schluss muss noch sichergestellt werden, dass für jeden Knoten und alle möglichen Kombinationen aus drei Nachbarknoten höchstens zwei der Nachbarknoten in der selben Teilmenge, wie der Knoten liegen. In K_4 gibt es für jeden Knoten nur eine mögliche Kombination aus drei Nachbarn.

$$\begin{aligned} t_{21} + t_{31} + t_{41} - 2t_{11} - 3t_{12} \leq 0 \quad & t_{11} + t_{21} + t_{41} - 2t_{31} - 3t_{32} \leq 0 \\ t_{22} + t_{32} + t_{42} - 2t_{12} - 3t_{11} \leq 0 \quad & t_{12} + t_{22} + t_{42} - 2t_{32} - 3t_{31} \leq 0 \\ t_{11} + t_{31} + t_{41} - 2t_{21} - 3t_{22} \leq 0 \quad & t_{11} + t_{21} + t_{31} - 2t_{41} - 3t_{42} \leq 0 \\ t_{12} + t_{32} + t_{42} - 2t_{22} - 3t_{21} \leq 0 \quad & t_{12} + t_{22} + t_{32} - 2t_{42} - 3t_{41} \leq 0 \end{aligned}$$

Eine gültige Belegung der Variablen, die alle (Un)gleichungen erfüllt, wäre zum Beispiel:

$$t_{11} = 1 \qquad t_{12} = 0$$
$$t_{21} = 1 \qquad t_{22} = 0$$
$$t_{31} = 0 \qquad t_{32} = 1$$
$$t_{41} = 0 \qquad t_{42} = 1$$

Somit befinden sich genau zwei Knoten in T_1 und genau zwei Knoten in T_2 und die lineare Knotenarborizität von K_4 ist 2; siehe Abbildung 3.2. Auf K_5 würde das lineare ganzzahlige Programm keine Lösung liefern, da es bereits für die Kreisgleichungen unmöglich ist, eine gültige Belegung der Variablen zu finden. Im Allgemeinen ist die Laufzeit, die zum Lösen des Programms benötigt wird, natürlich exponentiell und nicht praktikabel. Außerdem kann es exponentiell viele Ungleichungen geben. Somit kann das ILP nicht einmal effizient hingeschrieben werden.

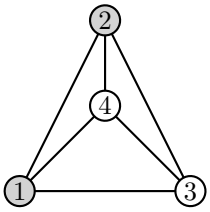


Abb. 3.2: K_4 mit zwei Knoten in T_1 und zwei Knoten in T_2.

4 Parametrisierte Komplexität

In diesem Abschnitt soll die parametrisierte Komplexität von 2-Linear Vertex-Arboricity und c-Linear Vertex-Arboricity untersucht werden. Im Allgemeinen ist ein Problem bezüglich eines Parameters k parametrisierbar (d. h. es liegt in der Klasse FPT), wenn es einen Algorithmus gibt, der in $f(k)\cdot \text{poly}(n)$ Zeit das Problem entscheidet. Dabei ist $f(k)$ eine beliebige berechenbare Funktion und $\text{poly}(n)$ ein Polynom von n.

4.1 c-Linear Vertex-Arboricity mit Parameter c

Wir fangen mit einer einfachen Fragestellung an. Das Problem c-Linear Vertex-Arboricity ist wie folgt definiert. Sei ein Graph G gegeben. Es soll entschieden werden, ob $\text{lva}(G) = c$ ist.

Satz 3. *Das c-Linear Vertex-Arboricity Problem ist nicht in FPT bezüglich des Parameters c.*

Beweis. Angenommen das Problem wäre in FPT. Dann müsste es einen Algorithmus geben, der das Problem für jedes feste c in $\text{poly}(n)$ Zeit entscheidet. Dies würde implizieren, dass für einen planaren Graph in $\text{poly}(n)$ Zeit entschieden werden kann, ob die lineare Knotenarborizität 2 ist. Das ist ein Widerspruch zu Satz 2. □

4.2 2-Linear Vertex-Arboricity mit Parameter Knotenüberdeckungszahl

In diesem Abschnitt werden wir zeigen, dass 2-Linear Vertex-Arboricity parametrisierbar bezüglich der Knotenüberdeckungszahl ist. Dies bedeutet also, dass für einen Graphen G mit Knotenüberdeckungszahl k in $f(k)\cdot \text{poly}(n)$ Zeit entschieden werden kann, ob G lineare Knotenarborizität 2 hat. Sei C die Menge der Knoten in einer minimalen Knotenüberdeckung. Sei V die Knotenmenge von G.

Wir wollen zuerst einige allgemeine Beobachtungen über eine gegebene Knotenüberdeckung C festhalten. Der kürzeste Pfad zwischen zwei Knoten a und b aus C hat Länge höchstens 2 oder enthält mindestens einen weiteren Knoten d aus C; siehe Abbildung 4.1. Ferner sind Knoten, die nicht in C liegen, nur zu Knoten aus C benachbart. Sonst gäbe es eine Kante, die durch keinen Knoten überdeckt wird. Dies impliziert, dass jeder Knoten, der nicht in C liegt, Knotengrad 1 hat oder zu zwei oder mehr Knoten aus C benachbart ist.

Die Menge aller ungeordneter Knotenpaare $\{a, b\}$ mit $a, b \in C$ bezeichnen wir mit $\binom{C}{2}$. Sei $\overline{C} = V \backslash C$ und seien $N_{\overline{C}}(a)$ die Nachbarknoten von a, die nicht in C sind. Sei

$\{a, b\} \in \binom{C}{2}$ und sei $S(a, b) = (N_{\overline{C}}(a) \cap N_{\overline{C}}(b))$. Somit sind $S(a, b)$ die Knoten zwischen a und b, die nicht in C sind, siehe Abbildung 4.2.

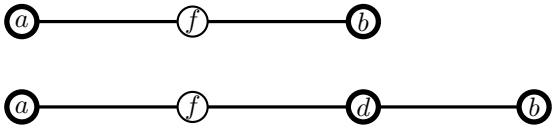


Abb. 4.1: Pfad zwischen zwei Knoten aus C. (Knoten aus C sind stärker umrandet.)

Das Problem ist, dass die Menge $S(a, b)$ beliebig groß sein kann, siehe Abbildung 4.2.

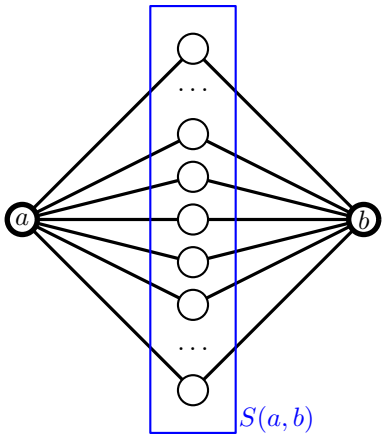


Abb. 4.2: Zwei Knoten $\{a, b\} \in \binom{C}{2}$ mit beliebig vielen Knoten in $S(a, b)$.

Wir wollen nun zeigen, dass es ausreicht, für eine Knotenmenge U, deren Größe durch eine Funktion $f(k)$ beschränkt ist, alle möglichen Kombinationen durchzuprobieren. Für die restlichen Knoten aus V, die nicht in U sind, ist die Aufteilung impliziert. Dazu konstruieren wir einen Algorithmus, der aus zwei Teilen besteht. Im ersten Teil werden bestimmte Knoten ausgewählt, die die Menge U bilden, sodass noch $|U| \leq f(k)$ gilt. Im zweiten Teil werden alle möglichen Färbungen von U durchprobiert. Aus einer Färbung von U wird die Farbe für die Knoten in $V \backslash U$ abgeleitet oder es gibt keine gültige Färbung und es wird eine andere Färbung für U ausprobiert.

Konstruktion von U Die Menge U wird wie folgt gebildet.

$$U = C \cup \bigcup_{\substack{\{a,b\} \in \binom{C}{2}: \\ |S(a,b)| \leq 4}} S(a, b)$$

Es gilt $\left|\binom{C}{2}\right| \leq k^2/2$. Somit ist $|U| \leq k + 4\binom{k}{2}$. Für die restlichen Knotenpaare $\{a, b\} \in \binom{C}{2}$ gilt also, dass $|S(a, b)| > 4$.

Färbealgorithmus Die Färbung funktioniert nun wie folgt. Es werden alle möglichen Färbungen der Knoten aus U durchprobiert. Falls sie bereits auf der Menge U ungültig ist, wird abgebrochen und die nächste probiert. Wir betrachten alle $\{a, b\} \in \binom{C}{2}$, für die gilt, dass es mindestens einen Knoten in $S(a, b)$ gibt, der nicht in U liegt. Haben a und b verschiedene Farben erhalten, so ist keine gültige Färbung mehr möglich und es wird eine andere Färbung probiert. Dies liegt daran, dass zwischen a und b mehr als vier Knoten liegen und keine gültige Färbung möglich ist, ohne dass ein Knoten mehr als zwei Nachbarn seiner eigenen Farbe hat, siehe Abbildung 4.3. Dabei können einige der Knoten zwischen a und b bereits zu U hinzugefügt worden sein und schon eine Farbe erhalten haben. Das ist die Knotenmenge $S(a, b) \cap U$.

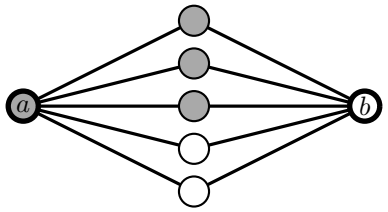


Abb. 4.3: a und b unterschiedlich gefärbt.

Haben a und b dieselbe Farbe erhalten und sind adjazent, so müssen alle Knoten in $S(a, b)$, die nicht in U sind und noch keine Farbe erhalten haben, anders als a und b gefärbt werden. Gäbe es einen Knoten d in $S(a, b)$ mit der selben Farbe wie a und b, dann würden a, b, d ein monochromes Dreieck bilden. Sind a und b keine Nachbarn und haben dieselbe Farbe erhalten, werden alle Knoten aus $S(a, b)$, die nicht in U sind und noch keine Farbe erhalten haben, mit der anderen Farbe gefärbt. Da a und b nicht adjazent sind, könnte genau ein Knoten h in $S(a, b)$ dieselbe Farbe wie a und b erhalten. Dies ist allerdings nur möglich, falls dieser Knoten keine weiteren Nachbarn der gleichen Farbe wie a und b hat. In diesem Fall ist es günstiger, h anders als a und b zu färben, weil dadurch sichergestellt ist, dass damit die momentane Färbung gültig bleibt.

Laufzeit Um die Menge U zu konstruieren, müssen alle möglichen Knotenpaare $\{a, b\} \in \binom{C}{2}$ und ihre gemeinsamen Nachbarn betrachtet werden. Für jedes Paar werden alle gemeinsamen Nachbarn von a und b betrachtet und, falls es weniger als fünf sind, zu U hinzugefügt. Dies ist also in $\mathcal{O}(k^2/2) \cdot \text{poly}(n)$ Zeit möglich.

Der Färbealgorithmus überprüft alle möglichen Färbungen von U. Wir wissen, dass U höchstens $k+4\binom{k}{2}$ Knoten enthält. Es gilt $\binom{k}{2} \leq k^2/2$. Somit ist $|U| \leq k + 2k^2$, und es gibt höchstens 2^{k+2k^2} verschiedene Möglichkeiten, die Knoten aus U zu färben. Es kann in $\text{poly}(n)$ Zeit festgestellt werden, ob eine Färbung gültig ist. Ebenso können die Knoten aus $S(a, b)$ in $\text{poly}(n)$ Zeit gefärbt werden. Somit kann in $\mathcal{O}(2^{k+2k^2} + k^2/2) \cdot \text{poly}(n)$ Zeit entschieden werden, ob die lineare Knotenarborizität eines gegebenen Graphen 2 ist. Dabei ist k die Größe einer minimalen Knotenüberdeckung.

Satz 4. 2-Linear Vertex-Arboricity *ist fest-parameter-berechenbar bezüglich der Knotenüberdeckungszahl: Für einen Graphen mit n Knoten und Knotenüberdeckungszahl k lässt sich in* $\mathcal{O}^*(2^{k+2k^2})$ *Zeit entscheiden, ob seine lineare Knotenarborizität höchstens 2 ist.*

4.3 c-Linear Vertex-Arboricity mit Parameter Knotenüberdeckungszahl

Wir wollen nun zeigen, dass c-Linear Vertex-Arboricity für ein konstantes c in FPT bezüglich der Knotenüberdeckungszahl liegt. Dazu erweitern wir den Algorithmus aus Abschnitt 4.2 auf c verschiedene Farben. Wieder sei C die Menge der Knoten in der Knotenüberdeckung. Sei $\binom{C}{c}$ die Menge aller c-elementigen Mengen aus C. Sei $t \in \binom{C}{c}$ eine solche c-elementige Menge, also $t = \{v_1, v_2, \ldots, v_c\}$. Sei $S(t) = \bigcap_{i=1}^{c} N_{\overline{C}}(v_i)$. Die Menge U wird wie folgt gebildet.

$$U = C \cup \bigcup_{\substack{t \in \binom{C}{c}: \\ |S(t)| \leq 2c}} S(t)$$

Der Färbealgorithmus geht wieder alle möglichen Färbungen von U durch. Diesmal betrachten wir alle $t \in \binom{C}{c}$, für die gilt, dass es mindestens einen Knoten in $S(t)$ gibt, der nicht in U ist. Zusätzlich wird für jeden Knoten, der nicht in U ist, gespeichert, mit welchen Farben er nicht gefärbt werden darf. Wurden die Knoten in t mit c verschiedenen Farben gefärbt, so gibt es keine gültige Färbung mehr und es muss die nächste ausprobiert werden. Wurden die Knoten in t mit weniger als c Farben gefärbt, speichern wir in jedem Knoten in $S(t)$ diese Farben. Das sind genau die Farben, mit denen dieser Knoten nicht gefärbt werden darf. Gibt es am Ende für ein t einen Knoten aus $S(t)$, der mit c Farben nicht gefärbt werden darf, wird die nächste Färbung für U ausprobiert. Wieder gilt, dass es einzelne Knoten in $S(t)$ geben könnte, die mit der gleichen Farbe wie Knoten in t gefärbt werden könnten. Allerdings ist es auch hier günstiger, sie mit der Farbe zu färben, die nicht in t vorkommt, weil dadurch sichergestellt ist, dass die Färbung gültig bleibt.

Die Menge U besteht aus maximal $k + 2c\binom{k}{c}$ Knoten. Somit hat der Algorithmus eine Laufzeit von $\mathcal{O}^*\left(2^{k+2c\binom{k}{c}}\right)$.

Satz 5. c-Linear Vertex-Arboricity *ist für jedes konstante c fest-parameter-berechenbar bezüglich der Knotenüberdeckungszahl.*

4.4 c-Linear-Vertex-Arboricity mit Parameter Baumweite

In diesem Abschnitt wird gezeigt, dass c-Linear Vertex-Arboricity bezüglich der Baumweite als Parameter in FPT liegt. Dazu soll die Tatsache ausgenutzt werden, dass Grapheneigenschaften, die mit Hilfe der erweiterten monadischen Prädikatenlogik zweiter Ordnung (MSO_2) ausgedrückt werden können, auf Graphen mit beschränkter Baumweite in polynomieller Zeit entschieden werden können [Cou90, CE12]. Wir werden uns zuerst auf 2-Linear Vertex-Arboricity beschränken und im Anschluss zeigen, wie das Ergebnis auf c-Linear Vertex-Arboricity verallgemeinert werden kann. Chaplick et al. [CKL$^+$18] haben gezeigt, dass Closed Outer K-Planarity in MSO_2 ausdrückbar ist. Analog dazu wird im Folgenden gezeigt, dass 2-Linear Vertex-Arboricity auf eine ähnliche Weise in MSO_2 dargestellt werden kann. Diese Logik bietet vier verschiedene Variablentypen: Knoten, Kanten, Mengen von Knoten und Mengen von Kanten. Die Prädikate über diesen Variablen können die Variablen auf Gleichheit, Mengenzugehörigkeit und Kanten-Knoten-Inzidenz testen. Sei U eine Menge von Knoten, seien v und w Knoten und sei e eine Kante. Wir wollen zuerst ein Prädikat konstruieren, das den Zusammenhang einer Knotenmenge U ausdrückt.

$$\textsc{Connected}(U) \equiv (\forall U' \subsetneq U)(U' \neq \emptyset \implies \exists v \in U' \wedge w \in U \backslash U' \exists e \in E)[I(e,v) \wedge I(e,w)]$$

Für alle möglichen Teilmengen U' von V, die nicht leer sind, muss gelten, dass es mindestens eine Kante von einem Knoten aus U' zu einem Knoten gibt, der nicht in U' liegt. Dabei drückt $I(e,v)$ die Inzidenz des Knotens v zur Kante e aus.

Da bei der linearen Knotenarborizität keine Kreise der gleichen Teilmenge erlaubt sind, brauchen wir ein Prädikat, das Kreise verbietet. Dazu erstellen wir zuerst ein Prädikat für eine Menge von Kreisen.

$$\textsc{Cycle-Set}(U) \equiv (\forall v \in U)(\exists^{=2} w \in U)(\exists e \in E)[I(e,v) \wedge I(e,w)]$$

Das heißt also, dass jeder Knoten v aus U genau zwei Nachbarn in U haben muss. Das Prädikat ist wahr, falls $G[U]$ eine Menge von Kreisen ist. Kombiniert man nun beide Prädikate, so erhält man die Definition für genau einen Kreis.

$$\textsc{Cycle}(U) \equiv \textsc{Cycle-Set}(U) \wedge \textsc{Connected}(U)$$

Da durch die beiden Teilmengen der linearen Knotenarborizität jeweils Mengen von Pfaden induziert werden müssen, darf es keinen Knoten geben, der mit mehr als zwei Nachbarn aus seiner Teilmenge benachbart ist. Wir bezeichnen die Konstruktion, bei der ein Knoten genau drei Nachbarn hat, als Stern.

$$\textsc{Star}(U) \equiv (\exists^{=1} v \in U)(\exists^{=3} w \in U \backslash \{v\})(\exists e \in E)[I(e,v) \wedge I(e,w)]$$

Hierdurch wird also ausgedrückt, dass U einen Stern hat.

Eine Menge U von Pfaden impliziert, dass U weder Kreise noch Sterne hat. Dies muss für jede Teilmenge von U gelten.

$$\textsc{Path-Set}(U) \equiv (\forall U' \subseteq U)[\neg \textsc{Cycle}(U') \wedge \neg \textsc{Star}(U')]$$

Dadurch wird also sichergestellt, dass $G[U]$ nur aus Pfaden besteht. Abschließend muss nur noch sichergestellt werden, dass die Knoten so in zwei Teilmengen aufgeteilt werden können, dass beide jeweils nur Mengen von Pfaden induzieren.

$$\text{LVA2}(U) \equiv (\exists U' \subsetneq U)[\textsc{Path-Set}(U') \wedge \textsc{Path-Set}(U \backslash U')]$$

Sei G nun ein Graph mit Knotenmenge V, dann beschreibt das Prädikat $\text{LVA2}(V)$ genau die Tatsache, dass die lineare Knotenarborizität von G gleich 2 ist.

Satz 6. *Die Eigenschaft, dass die Knotenmenge eines Graphen so in zwei Teilmengen aufgeteilt werden kann, dass beide einen linearen Wald induzieren, ist in MSO_2 ausdrückbar. Somit ist 2-Linear Vertex-Arboricity bezüglich der Baumweite fest-parameter-berechenbar.*

Bemerkung 7. Da die Baumweite durch die Pfadweite beschränkt ist, ist 2-Linear Vertex-Arboricity auch fest-parameter-berechenbar bezüglich der Pfadweite. Ebenso ist die Baumweite durch die Knotenüberdeckungszahl beschränkt. Die Laufzeit des Algorithmus für MSO_2 ist allerdings deutlich schlechter als die Laufzeit des Algorithmus aus Satz 4.

Chaplick et al. [CKL$^+$18] entwickelten das Prädikat $\textsc{Vertex-Partition}(U_1, U_2, \ldots, U_c)$. Dieses übernehmen wir als $\text{VP}(U_1, U_2, \ldots, U_c)$ und zeigen damit, dass auch c-Linear Vertex-Arboricity fest-parameter-berechenbar bezüglich der Baumweite ist.

$$\text{VP}(U_1, U_2, \ldots, U_c) \equiv (\forall v)\left[\left(\bigvee_{i=1}^{k} v \in U_i\right) \wedge \left(\bigwedge_{i \neq j} \neg(v \in U_i \wedge v \in U_j)\right)\right]$$

Dadurch wird ausgedrückt, dass jeder Knoten in genau einer der c Teilmengen vorkommen muss.

$$\text{LVAC}(U) \equiv (\exists U_1, U_2, \ldots, U_c)\left[\left(\text{VP}(U_1, U_2, \ldots, U_c)\right) \wedge \left(\bigwedge_{i=1}^{c} \textsc{Path-Set}(U_i)\right)\right]$$

$\text{LVAC}(U)$ drückt also die Tatsache aus, dass U so in c disjunkte Teilmengen aufgeteilt werden kann, dass jede davon eine Menge von Pfaden induziert. Somit ist die Eigenschaft, dass die Knotenmenge eines Graphen so in c Teilmengen aufgeteilt werden kann, dass jede davon einen linearen Wald induziert, in MSO_2 ausdrückbar.

Satz 8. *Das c-Linear Vertex-Arboricity Problem ist fest-parameter-berechenbar bezüglich der Baumweite.*

Bemerkung 9. Da die Baumweite durch die Pfadweite beschränkt ist, ist c-Linear Vertex-Arboricity auch fest-parameter-berechenbar bezüglich der Pfadweite.

5 Fazit

In dieser Arbeit wurde gezeigt, dass es für planare Graphen NP-vollständig ist zu entscheiden, ob die lineare Knotenarborizität 2 ist. Dazu wurde in Abschnitt 2.1 eine Polynomialzeitreduktion von Clause-Linked Planar 3-SAT auf 2-Linear Vertex-Arboricity vorgestellt. Ebenso haben wir bewiesen, dass 2-Linear Vertex-Arboricity bereits für planare Graphen mit Maximalgrad 6 NP-vollständig ist. In Abschnitt 2.2 wurde dafür eine Polynomialzeitreduktion von Clause-Linked-Planar-Exactly-3-Bounded-3-SAT auf 2-Linear Vertex-Arboricity entwickelt. Dabei sind die entstehenden Instanzen von 2-Linear Vertex-Arboricity planar und haben Maximalgrad 6. Auf allgemeinen Graphen ist 2-Linear Vertex-Arboricity bereits für Graphen mit Maximalgrad 5 NP-vollständig. Dies wurde in Abschnitt 2.3 gezeigt.

Somit ist das Entscheidungsproblem, ob die Knoten eines Graphen im $\mathbb{R}^3$ mit zwei Geraden überdeckt werden können, für planare Graphen (mit Maximalgrad 6) und für allgemeine Graphen mit Maximalgrad 5 NP-vollständig. Dabei muss die Zeichnung des Graphen kreuzungsfrei und geradlinig sein.

Anschließend wurde in Kapitel 3 ein ganzzahliges lineares Programm vorgestellt, das in der Lage ist, für einen gegebenen Graph eine gültige Teilmengenaufteilung für 2-Linear Vertex-Arboricity zu finden. Hat das Programm keine Lösung, so ist die lineare Knotenarborizität des Graphen größer als 2. Es sollte möglich sein, das ganzzahlige lineare Programm so zu erweitern, dass es c-Linear Vertex-Arboricity entscheiden kann.

Zum Abschluss wurde die parametrisierte Komplexität des Problems in Kapitel 4 betrachtet. Wir konnten zeigen, dass 2-Linear Vertex-Arboricity und c-Linear Vertex-Arboricity fest-parameter-berechenbar bezüglich der Parameter Knotenüberdeckungszahl und Baumweite sind.

6 Offene Probleme

In Kapitel 2.2 wurde gezeigt, dass es NP-schwer ist zu entscheiden, ob ein planarer Graph mit Maximalgrad 6 eine lineare Knotenarborizität von 2 hat. In Kapitel 2.3 wurde dies für allgemeine Graphen mit Maximalgrad 5 bewiesen. Daraus resultiert die natürliche Frage, wie schwer es für einen planaren Graph mit Maximalgrad 5 ist und ob es überhaupt einen planaren Graph mit Maximalgrad 5 und linearen Knotenarborizität 3 gibt. Eine interessante Tatsache ist, dass K_5^- sich nur so in zwei Teilmengen, die beide jeweils einen linearen Wald induzieren, aufteilen lässt, dass die zwei Knoten mit Grad 3 in der selben Teilmenge liegen müssen. Ferner liegen die beiden Knoten in allen möglichen Zeichnungen in verschiedenen Facetten, sodass sich für eine beliebige Konstruktion eines planaren Graphen nur einer der Knoten gleichzeitig nutzen lässt ohne die Planarität zu zerstören. Dies könnte ein Hinweis darauf sein, dass ein solcher Graph nicht existieren kann. Eine Computeranalyse ergab, dass planare Graphen mit zwölf Knoten, die 3-zusammenhängend sind und Maximalgrad 5 haben, alle eine lineare Knotenarborizität von 2 haben.

Eine weitere offene Fragestellung ist es, dass c-Linear-Vertex-Arboricity Problem als „precoloring extension“ Variante zu betrachten. Also wäre für eine Instanz des Problems für einige Knoten die Teilmengenzugehörigkeit bzw. Färbung von Anfang an festgelegt. Dieses Problem ist im allgemeinen Fall natürlich NP-schwer, könnte aber eventuell für bestimmte Graphklassen oder bestimmte Instanzen schnell gelöst werden. So z. B. auf (fast) vollständigen Graphen. Jede gültige Vorfärbung eines vollständigen Graphen lässt sich in linearer Zeit vervollständigen.

Ebenso könnte es interessant sein, ob das Problem zwei Knoten in einem Graphen zu finden, die unter allen möglichen Teilmengenaufteilungen in unterschiedlichen oder gleichen Teilmengen liegen müssen, NP-schwer ist. Dies ist z. B. für die beiden Grad-3-Knoten in K_5^- und zwei Teilmengen der Fall.

Das Entscheidungsproblem, ob die Knoten eines Graphen so in zwei Teilmengen aufgeteilt werden können, dass beide einen linearen Wald induzieren und sich die Anzahl der Knoten in den Teilmengen um höchstens eins unterscheidet, ist NP-schwer. Es sollte vermutlich möglich sein, den Algorithmus aus Abschnitt 4.2 so anzupassen, dass auch dieses Entscheidungsproblem fest-parameter-berechenbar bezüglich der Knotenüberdeckungszahl ist.

Literaturverzeichnis

[CE12] Bruno Courcelle und Joost Engelfriet: *Monadic second-order logic*, Seite 315–426. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012, 10.1017/CBO9780511977619.007.

[CFL+17] Steven Chaplick, Krzysztof Fleszar, Fabian Lipp, Alexander Ravsky, Oleg Verbitsky und Alexander Wolff: The Complexity of Drawing Graphs on Few Lines and Few Planes. In: Faith Ellen, Antonina Kolokolova und Jörg Rüdiger Sack (Herausgeber): *Workshop on Algorithms and Data Structures (WADS)*, Band 10389 der Reihe *Lecture Notes in Computer Science*, Seiten 265–276, Cham, 2017. Springer-Verlag, 10.1007/978-3-319-62127-2_23.

[CFL+20] Steven Chaplick, Krzysztof Fleszar, Fabian Lipp, Alexander Ravsky, Oleg Verbitsky und Alexander Wolff: Drawing Graphs on Few Lines and Few Planes. *Journal of Computational Geometry*, 11(1):433–475, 2020, 10.20382/jocg.v11i1a17.

[CKL+18] Steven Chaplick, Myroslav Kryven, Giuseppe Liotta, Andre Löffler und Alexander Wolff: Beyond Outerplanarity. In: Fabrizio Frati und Kwan Liu Ma (Herausgeber): *Graph Drawing and Network Visualization (GD)*, Band 10692 der Reihe *Lecture Notes in Computer Science*, Seite 546–559. Springer-Verlag, 2018, 10.1007/978-3-319-73915-1_42.

[Cou90] Bruno Courcelle: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990, 10.1016/0890-5401(90)90043-H.

[DESW07] Vida Dujmović, David Eppstein, Matthew Suderman und David R. Wood: Drawings of planar graphs with few slopes and segments. *Computational Geometry*, 38(3):194–212, 2007, 10.1016/j.comgeo.2006.09.002.

[Far04] Alastair Farrugia: Vertex-Partitioning into Fixed Additive Induced-Hereditary Properties is NP-hard. *Electr. J. Comb.*, 11(1), 2004, 10.37236/1799.

[FKMP95] Michael R. Fellows, Jan Kratochvíl, Matthias Middendorf und Frank Pfeiffer: The complexity of induced minors and related problems. *Algorithmica*, 13:266–282, 1995, 10.1007/BF01190507.

[KRW19] Myroslav Kryven, Alexander Ravsky und Alexander Wolff: Drawing Graphs on Few Circles and Few Spheres. *Journal of Graph Algorithms and Applications*, 23(2):371–391, 2019, 10.7155/jgaa.00495.

[Mat90] Makoto Matsumoto: Bounds for the vertex linear arboricity. *Journal of Graph Theory*, 14(1):117–126, 1990, 10.1002/jgt.3190140113.

[Sch15] André Schulz: Drawing Graphs with Few Arcs. *Journal of Graph Algorithms and Applications*, 19(1):393–412, 2015, 10.7155/jgaa.00366.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 22. Dezember 2021

. . . . . .
Alexander Erhardt