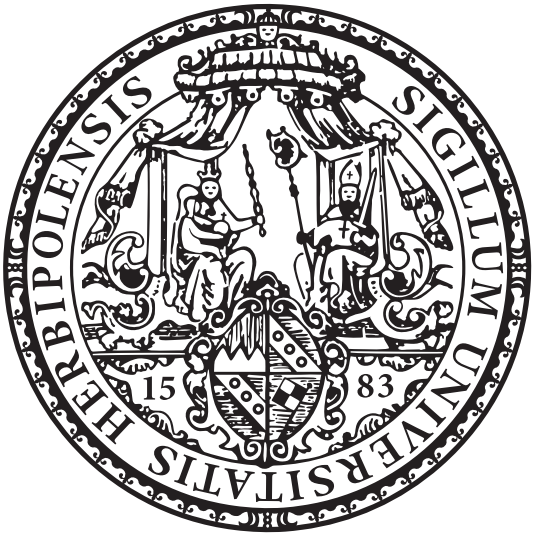
Bachelorarbeit

1-Hindernis-Sichtbarkeitsrepräsentation von kubischen Graphen

Christian Goldschmied

Abgabedatum:	22. April 2021
Überarbeitungsdatum:	12. Juni 2021
Betreuer:	Prof. Dr. Alexander Wolff
	Felix Klesen, M. Sc.



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

Zusammenfassung

In dieser Arbeit beschäftigen wir uns mit der Generierung von kubischen Graphen und der Darstellung dieser mithilfe von Hindernissen, genauer einem einzigen Außenseitenhindernis in der Ebene. Dazu definieren wir Operationen mit welchen wir rekursiv aus kleineren kubischen Graphen Größere erzeugen können. Danach erläutern wir die Funktionsweise von McKays Kanonischer-Konstruktionspfadmethode, die uns erlaubt isomorphiefreie Mengen von Objekten zu generieren. Wir fahren fort, indem wir das zuvor eingeführte Verfahren für den Generierungsalgorithmus umsetzen, um isomorphiefreie Mengen von kubischen Graphen zu erhalten. Wir benutzen eine SAT-Formulierung für unsere generierten kubische Graphen, die uns eine Permutation von Knoten ausgibt, die wir in einem konvexen Polygon anordnen können. Diese Permutation nutzen wir dann um eine 1-Hindernis-Darstellung der kubischen Graphen zeichnen zu können.

Inhaltsverzeichnis

1. Motivation

Als Francis Guthrie im Jahr 1852, die uns als *Vier-Farben-Satz* bekannte Vermutung erstmals aufstellte, als er eine Karte der Grafschaften Englands einfärben wollte und drei Farben offensichtlich nicht ausreichten, konnten man noch nicht erahnen, dass diese Vermutung mehr als 100 Jahre später zum ersten computergestützten mathematischen Beweis führen würde. Zum allerersten Mal in der Geschichte der Mathematik wurde also im Jahre 1976 von Appel und Haken [AH89] zum Beweisen des Vier-Farben-Satzes ein mathematischer Beweis mithilfe von Computern geführt, bei welchem eine sehr große Anzahl von Fallunterscheidungen geprüft werden mussten. Über die folgenden Jahre konnte weiterhin die Anzahl der Fallunterscheidungen des Beweises immer weiter verbessert werden [RSST96].
Auch wir wollen in dieser Arbeit über das Gebiet der Graphentheorie eine Brücke zwischen Mathematik und Informatik schlagen. Wir werden in dieser Arbeit nicht nur Mengen kubischer Graphen isomorphiefrei generieren, sondern auch gewünschte Darstellungen dieser erzeugen.
Wir können unter Zuhilfenahme von Computern durch einfache Art und Weise auch für große Mengen kubischer hamiltonscher Graphen zeigen, dass sie alle eine notwendige Bedingung für die Darstellung mit nur einem Hindernis erfüllen, wodurch sich unsere Vermutung begründet, dass sogar alle Graphen der Menge $\mathcal{S}$, die alle kubischen hamiltonschen Graphen enthält, mit nur einem Hindernis C darstellen lassen. Dafür nutzen wir eine SAT-Formulierung des Problems, die einen Graphen $G = (V, E)$, wobei $|V| = n$ in Form einer Adjazenzmatrix als Eingabe erhält und uns ein n-stelliges Tupel mit den Zahlen von $1, 2, \ldots, n$, wobei diese die Anordnungen oder Reihenfolge der Knoten beschreibt, als Ausgabe zurückgibt. Die Klasse dieser Graphen kann dabei nicht nur von theoretischem Interesse für mathematische Probleme sein, sondern auch in der modernen Chemie werden häufig Kohlenstoffverbindungen erforscht, die als kubischer Graph modelliert werden können. Diese binden sich mit jeweils drei anderen Kohlenstoffatomen, sodass Atome als Knoten und die Verbindung als Kanten dargestellt werden. Ein Beispiel sind die mit einem Nobelpreis ausgezeichneten Fullerene [KHO$^+$85]. Aber auch in der Graphentheorie sind oftmals kubische Graphen das kleinste oder einfachste Gegenbeispiel für viele theoretische Probleme. Aus diesen Gründen ist die Erforschung dieser Graphen und deren Darstellung mit einem oder auch mehreren Hindernissen interessant.

1.1. Problemstellung

Zuallererst wollen wir diesen Abschnitt beginnen, indem die benötigten Begriffen definiert werden, um die Problemstellung, mit der wir uns beschäftigen, formulieren zu können. Wir beginnen mit der Einführung und Definition einer geradlinigen Darstellung

von Graphen in der Ebene. Diese Darstellung wird nur Gebrauch von der Position der Punkte und Hindernisse machen.

Definition 1 (Hindernis-Sichtbarkeitsgraphen)**.** *Gegeben sei nun ein topologisch offenes, sich nicht selbst schneidendes Polygon C und eine endliche Menge P von Punkten in allgemeiner Lage im Komplement von C. Der* Sichtbarkeitsgraph $G' = (C, P)$ *hat einen Knoten für jeden Punkt $p \in P$ und für zwei beliebige Punkte p und q genau dann eine Kante pq, wenn die geradlinige Verbindung der Punkte p und q nicht durch das Hindernis unterbrochen wird, also wenn $C \cap pq = \emptyset$. Jede Kante pq, die durch das Hindernis behindert wird, nennen wir* Nicht-Kante*. Eine geradlinige Zeichnung von $G' = (C, P)$ nennen wir* Sichtbarkeitszeichnung*. Wir nennen diese Darstellungen im Rahmen dieser Arbeit auch* Hindernisdarstellungen*.*

Sei $G = (V, E)$ ein Graph mit Knotenmenge E und Kantenmenge V. Wir suchen demnach nach einer Hindernisdarstellung $G' = (C, P)$ des Graphen G, also ein Hindernis C und eine Menge von Punkten P, sodass gilt $G = G'$.

Wir können dabei zusätzlich zwischen Hindernissen in der unbeschränkten Komponente und im Komplement der unbeschränkten Komponente unterscheiden. Wir sprechen von *Außenseitenhindernis* und *Innenseitenhindernis*.

Nun wollen wir die Familie der kubischen Graphen einführen, mit der wir uns auseinandersetzen werden.

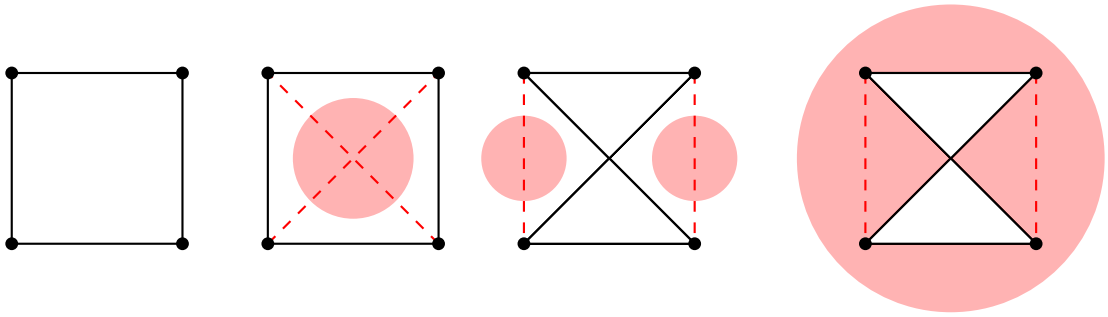
Definition 2 (Kubische Graphen)**.** *Ein* kubischer *Graph $G = (V, E)$ ist ein Graph, dessen Knoten $v \in V$ alle genau* Knotengrad 3 *haben. Kubische Graphen werden auch als* 3-reguläre Graphen *bezeichnet. Dabei bezeichnet der Knotengrad eines Knoten $v \in V$ die Summe aller zu v inzidenten Kanten.*

In dieser Arbeit soll insbesondere eine Teilmenge $\mathcal{S}$ der kubischen Graphen betrachtet werden. Ein kubischer Graph G sei genau dann Element dieser Klasse $\mathcal{S}$, wenn er zusätzlich die Eigenschaft hat *hamiltonsch* zu sein. Ein Graph ist hamiltonsch, wenn er einen Zyklus enthält, der jeden Knoten genau ein mal besucht.

Entfernen wir die Restriktion aus unserer Definition von Hindernis-Sichtbarkeitsgraphen von einem einzelnen Hindernis zu einer Menge von vielen Hindernissen $C \in \mathcal{C}$, so wird das Problem zu einem Optimierungsproblem. Wir suchen somit eine minimale Anzahl von Hindernissen, die eine Darstellung eines Graphen G als Sichtbarkeitsgraph erlaubt.

Definition 3 (Hinderniszahl)**.** *Die Hindernisanzahl $\mathcal{H}(G)$ eines Graphen $G = (V, E)$ ist die kleinste Anzahl einer Menge von Hindernissen $\mathcal{C}$ in einer beliebigen Hindernisdarstellung von G.*

Für unsere definierte Menge $\mathcal{S}$ soll nun gezeigt werden, dass jeder kubische Graph $G \in \mathcal{S}$ mit genau einem Hindernis, genauer einem Außenseiten-Hindernis C, also 1-Hindernis darstellbar ist. Wir vermuten, dass die gesamte Graphenklasse *Hinderniszahl* 1 hat, also $\mathcal{H}(\mathcal{S}) = 1$. Dazu werden wir eine SAT-Formulierung der Problemstellung benutzen, mit



Einfacher Graph C_4 | Innenseiten-Hindernis-Darstellung | Außenseiten-Hindernis-Darstellung mit 2 Hindernissen | Außenseiten-Hindernis-Darstellung mit einem Hindernis

Abb. 1.1.: Verschiedene Darstellungen des Graphen C_4 mit verschiedenen Hindernissen

der wir die Realisierbarkeit in konvexer Lage prüfen können und falls möglich, eine gültige Permutation als Ausgabe erhalten. Die genaue Vorgehensweise wollen wir im folgenden Abschnitt 1.3 erläutern.

1.2. Verwandte Arbeiten

Erstmalig wurde der Begriff der Hinderniszahl eines Graphen von Alpert et al. [AKL10] etabliert. Viele damit einhergehende Fragen über die Hinderniszahl von Graphen und verschiedener Graphenklassen konnten auch schon beantwortet werden. Während es offensichtlich ist, dass die Hindernisdarstellung eines Graphen stets durch eine obere Schranke von $O(n^2)$ Hindernissen, wobei n die Anzahl der Knoten des Graphen beschreibt, beschränkt ist, existieren auch Graphen, die eine Mindestanzahl an Hindernissen benötigen. Beispielsweise gilt für die Graphen, die wir mit $G_{k,m}$ bezeichnen, für geeignete Wahl von k, dass $\mathcal{H}(G_{k,m}) = m$. Dieser setzt sich aus einem vollständigen Graphen G_k und für jede Teilmenge S, die genau $2m + 2$ Knoten enthält, wird ein zusätzlicher Knoten hinzugefügt und mit den Knoten aus S verbunden. Es stellt sich aber hierdurch auch die Frage, ob es möglich ist die Hinderniszahl sogar durch eine lineare Funktion zu beschränken. Bisher konnte diese Schranke auf $O(n \log n)$ reduziert werden [BCV15]. Als Familie von Graphen mit kleiner Hinderniszahl konnten bisher außenplanare Graphen, die darüber hinaus eine Außenseiten-Hindernis Darstellung haben, klassifiziert werden [AKL10]. Darüber hinaus konnte die Klasse von Graphen mit konvexer Hinderniszahl 1, bei welcher nur konvexe Hindernisse erlaubt sind, von Alpert et al. [AKL10] vollständig als sogenannte nicht-doppelüberlappende kreisförmige Bogen-Graphen charakterisiert werden. Viele interessante Problemstellungen sind aber auch offen geblieben. Gibt es beispielsweise für jede Zahl $n \geq 3$ einen Graphen mit der Hinderniszahl genau n? Falls ja, wie viele Knoten muss dieser Graph mindestens haben? Was sind die Hinderniszahlen von planaren Graphen, insbesondere des Ikosaeders? Gibt es sogar (außen)planare Graphen mit beliebig großer konvexer Hinderniszahl? Mit

welcher Komplexität können Graphen mit konvexer Hinderniszahl 1 entschieden werden? Chaplick et al. [CLPW16] konnten einige dieser Probleme bereits lösen. So kann man zeigen, dass jeder beliebige Graph mit maximal sieben Knoten eine Außenseiten-Hindernis Zahl von genau 1 hat, also eine Darstellung mit nur einem Hindernis. Erstaunlich ist dabei, dass Graphen mit nur 8 Knoten existieren, die eine Hinderniszahl von 2 haben, womit eine scharfe Schranke gefunden werden konnte. Weiterhin kann festgehalten werden, dass die Klassen der Graphen mit Innenseiten-Hindernis-Darstellung und Außenseiten-Hindernis-Darstellung nicht vergleichbar sind. Dabei wurde ein Beispiel für einen Graphen mit 11 Knoten mit gültiger Innenseiten-Hindernis-Darstellung, aber ohne Außenseiten-Hindernis-Darstellung gefunden [CLPW16].

1.3. Vorgehensweise

Für die Realisierung unseres Vorhabens 1-Außenhindernis-Darstellung kubischer Graphen angeben zu können, werden wir also zunächst Algorithmen benötigen, die alle kubischen Graphen mit n Knoten ausgeben können. Zusätzlich wollen wir bei der Generierung der Graphen sicherstellen, dass alle Graphen paarweise nicht isomorph zueinander sind, da allein eine einzelne Berechnung der SAT-Formulierung für kubische Graphen ab einer gewissen Größe eine sehr rechen- und zeitintensive Operation darstellt und deshalb eine mehrfache Berechnung des selben Graphen vermieden werden soll. Im ersten Schritt werden wir die Generierung und die damit einhergehenden Schwierigkeiten der Generierung von kubischen Graphen theoretisch erläutern. Im zweiten Schritt nutzen wir die Menge der generierten isomorphiefreien kubischen Graphen, um für alle kubischen Graphen eine SAT-Formulierung unserer Problemstellung mithilfe des Constraint Program Optimizers von IBMs CPLEX Studio zu berechnen. Diese liefert, falls vorhanden, eine Permutation und mögliche Lösung der Knoten als Ergebnis aus, die notwendig ist, von der aber bisher noch nicht klar ist, ob sie auch hinreichend ist. Diese Permutation von Knoten können wir dann in konvexer Lage angeben und somit eine 1-Außenhindernis-Darstellung finden.

2. Kubische Graphen

Wie bereits erwähnt, werden in dieser Arbeit insbesondere 1-Hindernis Darstellungen der Teilmenge $\mathcal{S}$, die alle hamiltonschen kubischen Graphen enthält, betrachtet. Sei ein beliebiger kubischer Graph $G = (V, E)$ mit $|V| = n$ Knoten gegeben, dann konvergiert der Anteil der kubischen Graphen, die hamiltonsch sind, gegen eins, wenn n gegen unendlich geht [GFFS21]. Einfach gesagt, ist es sehr wahrscheinlich, dass ein kubischer Graph mit n Knoten zusätzlich einen Hamiltonkreis enthält. Daher spricht nichts dagegen, auch nicht-hamiltonsche kubische Graphen auf Außenseitenhindernis-Darstellungen zu untersuchen. Wir werden jedoch feststellen, dass es Beispiele für nicht-hamiltonsche kubische Graphen gibt, für die sich zunächst sicher keine 1-Hindernis-Darstellung findet, in welcher die Knoten alle in konvexer Lage liegen, jedoch trotzdem 1-Außenseiten-Hindernis darstellbar sind.

Deswegen beschränken wir uns auf die Teilmenge $\mathcal{S}$ von kubischen Graphen, für die eine 1-Außenseitenhindernis-Darstellung in konvexer Lage vermutet wird. Graphen der Klasse $\mathcal{S}$ können präzise mithilfe der *LCF-Notation* beschrieben werden. Wir wollen hiermit die Notation einführen.

2.1. LCF-Notation

Die *LCF-Notation* ist eine praktische Notation, die von Lederberg [Led65] für die Darstellung von kubischen Hamilton-Graphen entworfen wurde. Die Notation wurde 1976 von Frucht und Coxeter [Fru77] erweitert und wird daher als LCF-Notation bezeichnet. Die Notation gilt nur für Hamilton-Graphen, da sie ihre Symmetrie und Prägnanz dadurch erreicht, dass ein Hamilton-Zyklus in ein konvexes Polygon eingebettet wird und die Knoten durch ein perfektes Matching ergänzt werden.

Definition 4 (LCF-Syntax)**.** *Die Beschreibung eines kubischen Graphen in LCF-Notation hat folgende Form:* $n, [s_0, s_1, \dots s_k]^p$
Semantik: Wir konstruieren zunächst einen Zyklus der Länge n mit den Knoten $\{v_0, v_1, \dots, v_{n-1}\}$ *und den Kanten* $v_i v_{i+1}$ *für* $0 \leq i \leq n-1$*, wobei alle Indizes modulo n zu lesen sind. Zusätzlich werden nun die Sehnenkanten* $(v_i, v_{(s_{i\%k})\%n})$ *hinzugefügt.*

Darüber hinaus enthalten einige Programme zur Bearbeitung von Graphen Funktionen zur Erstellung eines Graphen aus seiner LCF-Notation. Beispielhaft sehen wir in Abbildung 2.1 wie ein kubischer Graph durch den Code generiert werden kann.

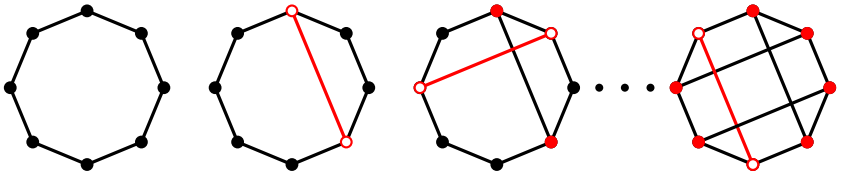


Abb. 2.1.: Sukzessiver Aufbau des kubischen Graphen mit dem LCF-Code $8, [3, -3]^4$

2.2. Generierung von kubischen Graphen

Um ein ausschöpfendes Testen aller kubischen Graphen zu ermöglichen, ist eine effiziente Generierung im Zuge dieser Arbeit notwendig, da die Anzahl von Graphen in Abhängigkeit der Knoten exponentiell schnell wächst. Viele Methoden und Algorithmen zur Generierung von kubischen Graphen sind bereits seit vielen Jahrzehnten bekannt. Das erste Verfahren, auf welchem heute auch die schnellsten Algorithmen zur Generierung basieren, wurde erstmals schon Ende des 19. Jahrhunderts von De Vries [DV89] beschrieben.

Kanteneinfügungen

Die von de Vries verwendete Idee ist, dass aus einem kubischen Graphen $G' = (V', E')$ mit $n-2$ Knoten ein kubischer Graph G mit n Knoten erzeugt werden kann. Dazu nutzen wir eine Operation, die wir *Kanteneinfügung* nennen. Bei dieser wählen wir zwei Kanten des Graphen e'_1 und e'_2 aus E',wobei $e'_1 \neq e'_2$ und unterteilen diese jeweils durch neue Knoten, die wir v_{e_1} und v_{e_2} nennen. Wir verbinden diese zwei neuen Knoten schließlich durch eine neue Kante $v_{e_1}v_{e_2}$. Es ist einfach zu erkennen, dass der resultierende Graph nach dieser Operation auch wieder kubisch ist, da der Knotengrad der Knoten für alle Knoten $v \in V'$ nicht beeinflusst wird.

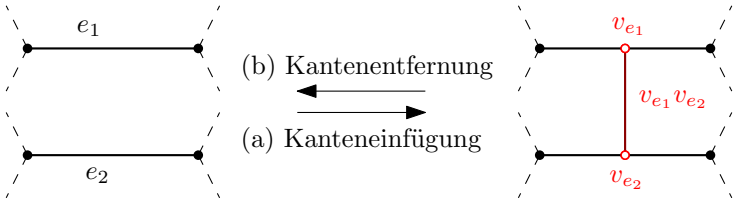


Abb. 2.2.: Kanteneinfügungsoperation mit der neuen Kante $v_{e_1}v_{e_2}$

Definition 5 (Kanteneinfügung)**.** *Gegeben seien zwei verschiedene Kanten uv, $ab \in E$ eines Graphen $G = (V, E)$. Unterteile die Kanten uv und ab durch jeweils einen neuen Knoten $x \notin V$ beziehungsweise $y \notin V$. Füge schließlich die Kante xy zur Kantenmenge E und die Knoten x, y zur Knotenmenge V hinzu.*

Entsprechend können wir die zur Kanteneinfügung inverse Operation *Kantenentfernung* definieren.

Definition 6 (Kantenentfernung)**.** *Eine Kante $e \in E$ und beide inzidenten Knoten lassen sich aus einem kubischen Graphen $G = (V, E)$ entfernen, wenn der resultierende Graph kubisch ist und auch nach Ausführung der Kantenentfernung nicht in mehrere nicht-zusammenhängende Komponenten zerfällt oder Multikanten enthält. Wir nennen solche Kanten* reduzibel.

Mit den beiden Operationen Kanteneinfügung und Kantenentfernung besitzen wir nun unsere Werkzeuge, die wir zu isomorphiefreien Generierung von kubischen Graphen benötigen. Die genaue Funktionsweise unserer isomorphiefreien Generierung wollen wir in Kapitel 3 und die genaue Implementierung der Kanteneinfügungsoperation in Kapitel 4 erläutern. Weiterhin wollen wir in diesem Abschnitt weitere zum Verständnis des Generierungsalgorithmus notwendige Begriffe und Lemmata einführen.

Kubische Primgraphen

Wir beginnen mit der Definition einer besonderen Teilmenge kubischer Graphen, die nur aus irreduziblen Kanten besteht.

Definition 7 (Kubische Primgraphen)**.** *Ein kubischer Graph $G = (V, E)$ ist prim, wenn er keine reduziblen Kanten besitzt. Wir nennen solche Graphen auch* kubische Primgraphen.

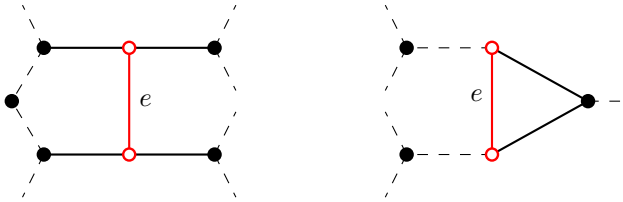
Wir charakterisieren die Klasse der kubischen Primgraphen $\mathcal{C}_\mathcal{P}$ mit folgendem Lemma nach Brinkmann et al. [BGM11].

Wir benutzen folgende Definitionen für den Beweis der nachfolgenden Lemmata.

Definition 8 (K_4-Varianten)**.** *Wir bezeichnen den Teilgraphen von K_4, von dem eine beliebige Kante e entfernt wurde als K_4^-. Wir nennen ihn auch* Drachen *und nennen die beiden Knoten mit Grad 2* Verbindungsknoten*. Analog dazu sei K_4^+ derjenige Graph, bei dem eine beliebige Kante e des Graphen K_4 durch einen neuen Knoten v unterteilt wurde.*

Lemma 9 (Charakterisierung kubischer Primgraphen)**.** *Ein kubischer Graph $G = (V, E)$ ist genau dann prim, wenn jede Kante $e \in E$ eine Brücke ist, oder einer der zu e inzidenten Knoten in einem K_4^- Drachen enthalten ist.*

Beweis. Sei $xy \in E$ eine Kante, die keine Brücke ist und keiner der zu e inzidenten Knoten x, y in einem K_4^- Drachen enthalten ist. In der Abbildung 2.3 sehen wir die beiden Fälle die nun auftreten können. Wenn die inzidenten Knoten der Kante e nun nicht in einem Dreieck enthalten sind, dann ist e reduzierbar, wie in Fall (a) zu sehen. Andererseits kann $x \in V$ in einem Dreieck enthalten sein, aber nicht in einem K_4^-, dann ist aber jede Kante des Dreiecks reduzierbar, siehe Fall (b). Die Rückrichtung ist trivial, denn weder Brücken noch Kanten, die einen Knoten in K_4^- haben, können reduziert werden, da der Graph sonst entweder in mehr als eine Zusammenhangskomponente zerfällt oder es Multikanten zwischen Knoten geben würde. □



(a) e hat keine Knoten in einem Dreieck

(b) e ist in einem Dreieck enthalten

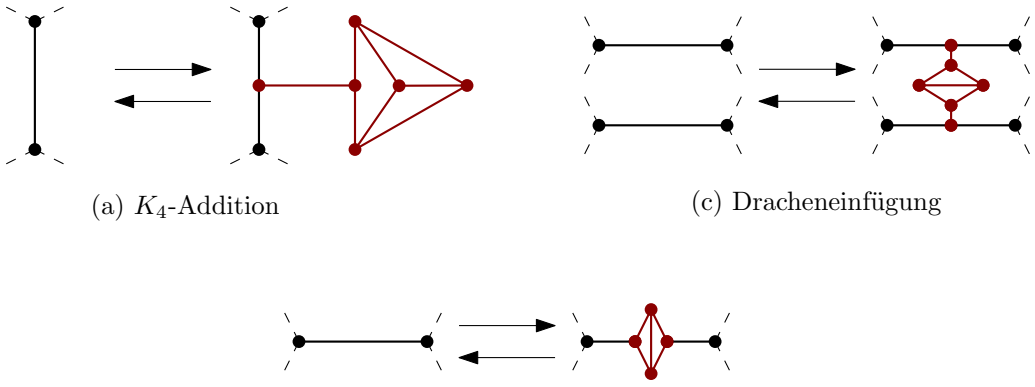
Abb. 2.3.: Beide Fälle, bei denen die Kante e auch reduzibel ist

Wir nutzen diese Charakterisierung von Primgraphen, um die Menge aller Primgraphen durch drei verschiedene Operationen, die wir *Primoperationen* nennen, generieren zu können.

Primoperationen

Wir beginnen folgenden Abschnitt durch Definition und Einführung der Primoperationen. Die Definition der Operationen sei durch die Abbildung 2.4 gegeben. Wir werden durch das folgende Lemma zeigen, dass wir mit Hilfe der Primoperationen auf einfache Weise alle kubischen Primgraphen generieren können.

Lemma 10 (Generierung kubischer Primgraphen)**.** *Die Klasse der Primgraphen kann durch rekursive Anwendung der Primoperationen auf dem Graphen K_4 generiert werden, sodass alle durch die Konstruktionsoperation entstehenden kubischen Graphen, alle existierenden Primgraphen enthalten.*



(a) K_4-Addition

(c) Dracheneinfügung

(b) Drachenunterteilung

Abb. 2.4.: Alle drei verschiedenen Primoperationen

Beweis. Wir müssen also zeigen, dass jeder Primgraph $G = (V, E)$ mit mehr als 4 Knoten auf einen kleineren Primgraphen durch eine der Reduktionen zurückzuführen ist.

Dazu werden wir alle auftretenden Fälle abarbeiten.
Angenommen G enthält eine Kante, dessen inzidente Knoten beide jeweils an verschiedenen Teilen eines K_4^+ oder zwei K_4^- enthalten sind. Falls einer der Knoten der Kante e in einem K_4^- endet, so können wir diese auf die zweite Operation (b) zurückführen. Es ist offensichtlich, dass der resultierende Graph im ersten Fall entweder eine Brücke enthält oder im Falle, dass beide Knoten in einem K_4^- enthalten waren, selbst wieder einen Knoten in K_4^- besitzt. Für den Fall, dass beide Knoten in einem K_4^+ enthalten sind, kommt einzig Operation (a) in Frage.
Für den nächsten Fall nehmen wir an, es gebe keine Kanten mit Knoten in K_4^+ oder K_4^-. Dann hat jede Kante, die nicht in einem K_4^- enthalten ist nur einen Knoten in K_4^- oder ist selbst eine Brücke. Die Existenz von K_4^+ impliziert, dass Operation (a) ausgeführt werden kann und die neue Kante wird entweder eine Brücke oder ihre beiden Endpunkte jeweils in einem K_4^- haben, womit der reduzierte Graph auch wieder prim ist.
Wenn es keine Teilgraphen K_4^+ gibt, dann gibt es eine Kante e mit einem Endpunkt in einem K_4^- die auf einem Zyklus liegt. Dann kann die Reduktion (c) auf K_4^- angewendet werden, der einen Knoten von e enthält. Der resultierende Graph ist zusammenhängend, weil e keine Brücke ist, und wird selbst auch prim sein, weil die neuen Kanten Brücken sind oder die Knoten eines K_4^- enthalten, wenn dies für die inzidenten Kanten auch galt. □

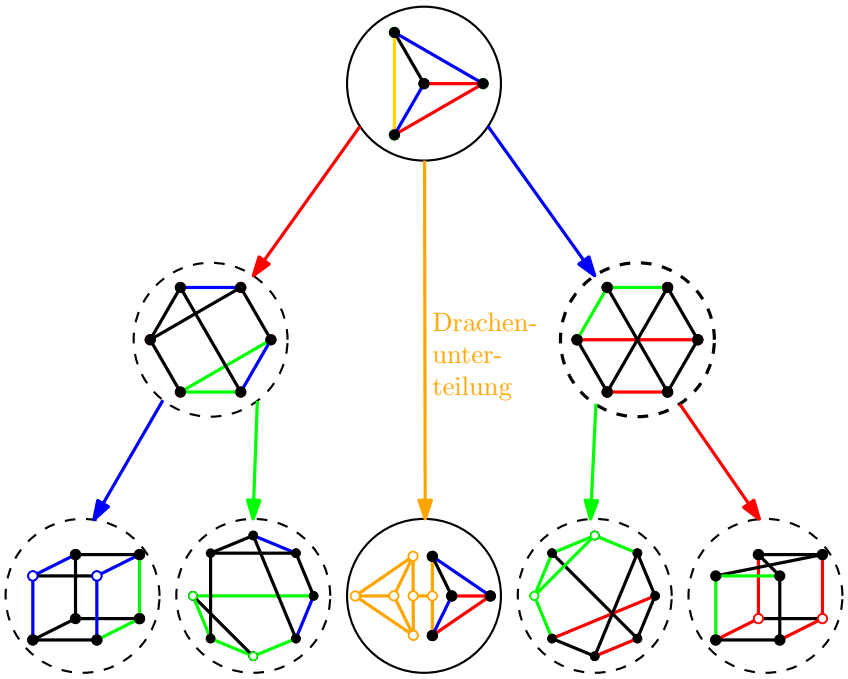


Abb. 2.5.: Konstruktionsbaum aller kubischer Graphen mit bis zu 8 Knoten. Wobei die Knoten mit durchgezogener Linie die beiden Primgraphen darstellen und die Einfärbung der Kanten angibt für welche Kanten eine Kanteneinfügung vorgenommen wird.

Es ist dabei anzumerken, dass die Klasse der kubischen Primgraphen unter der Konstruktionsoperationen nicht abgeschlossen ist und das Lemma nur besagt, dass wir durch die Konstruktionsoperationen sicher alle Primgraphen abgesucht haben. Somit muss jeder durch die Primoperationen erzeugte kubische Graph nachträglich auf reduzible Kanten geprüft werden. Führen wir beispielsweise eine Dracheneinfügung für den Graphen K_4 aus, so erhalten wir den kubischen Graphen mit 10 Knoten und dem LCF-Code $[4, 2, 3, -2, -4, -3, 2, 2, -2, -2]$, der nicht prim ist. Beide anderen Primoperationen führen jedoch zu den beiden einzigen Primgraphen mit jeweils 8 und 10 Knoten.

Nach Definition dieser Graphenklasse von kubischen Primgraphen können wir durch unsere Kanteneinfügungsoperation alle kubischen Graphen generieren. Wir konstruieren die Klasse kubischer Graphen, indem wir alle Primgraphen generieren und dann für diese Kanteneinfügungen vorzunehmen.

Beispielhaft soll nun der Konstruktionsbaum für alle kubischen Graphen bis 8 Knoten gegeben sein, siehe Abbildung 2.5.

Schwierigkeiten

Die Klasse der zusammenhängenden kubischen Graphen ist unter der Kanteneinfügeoperation abgeschlossen, sodass die einzige Schwierigkeit die zeitaufwändigen Routinen sind, die sicherstellen, dass nur paarweise nicht-isomorphe Graphen erzeugt werden. Wir verwenden die in Kapitel 3 beschriebene kanonische Konstruktionspfadmethode. Wir werden diese zunächst im folgenden Abschnitt allgemein und auf unser Beispiel angewendet beschreiben.

Eine weitere Schwierigkeit sind kubische Graphen, die keinen Hamiltonkreis enthalten. Denn auch für diese ist das Finden von Hamiltonkreisen ein NP-schweres Problem [GFFS21]. Allerdings können wir diese zunächst ignorieren, da wir uns auf die Graphen der Menge $\mathcal{S}$ fokussieren.

Um dieses Problem zu lösen, benutzen wir eine Methode, die auf McKay zurückgeht.

3. Kanonische Konstruktionspfadmethode

Die Kanonische Konstruktionspfadmethode ist ein Verfahren, genauer gesagt eine Algorithmenskizze von McKay [McK98], mit welcher viele unterschiedliche kombinatorische Objekte, die eine rekursive oder induktive Konstruktionsstruktur aufweisen, sogar mit Ausschluss von Isomorphie erzeugt werden können. In einigen Fällen kann auf das Testen von Isomorphie komplett verzichtet werden, in anderen Fällen beschränkt sich dieses jedoch auf sehr kleine Teilmengen von Objekten. Kombinatorische Objekte können dabei beispielsweise auch Graphen sein. Wir wollen diese Methode, die wir zur Generierung der kubischen Graphen nutzen, parallel theoretisch und auch praktisch für unser Beispiel einführen. Wir beziehen uns dabei auf die Zusammenfassung von Stolee [Sto12].

3.1. Objekte, Augmentierungen und Reduktionen

Gegeben sei eine Familie von kombinatorischen beschrifteten Objekten $\mathcal{L}$, aus welcher wir nach bestimmten Objekten suchen. Sei eine passende Definition eines Isomorphismus zwischen diesen kombinatorischen beschrifteten Objekten gegeben. Die Äquivalenzklassen unter der Definition des Isomorphismus bezeichnen wir mit $\mathcal{U}$ als die Menge der unbeschrifteten Objekte. Wir definieren zusätzlich eine Abbildung $P : \mathcal{L} \to \{0, 1\}$, die für ein Objekt $X \in \mathcal{L}$ genau dann $P(X) = 1$ ist, wenn X eine bestimme Eigenschaft erfüllt. Wir wollen also alle Objekte X generieren, für die $P(X) = 1$ ist. Wir nehmen an, dass für alle unbeschrifteten Objekte $\mathcal{X} \in \mathcal{U}$ und alle $X, X' \in \mathcal{X}$ stets die Gleichung $P(X) = P(X')$ gilt.

Wir benötigen eine Basismenge von Objekten $\mathcal{B} \subset \mathcal{L}$, wobei für je zwei Elemente $X, Y \in \mathcal{B}$ gelten soll, dass X und Y nicht isomorph sein dürfen. Aus dieser Basismenge können wir durch folgende Operationen unter Einbezugnahme der Abbildung P sukzessiv unsere gewünschten Objekte generieren.

Für jedes einzelne beschriftete Objekt $X \in \mathcal{L}$ definieren wir sowohl eine Menge von Augmentierungen $\mathcal{A}(X)$ als auch eine Menge von Reduktionen $\mathcal{R}(X)$. Beide Mengen von Operationen sollen jeweils genug Informationen enthalten, um zu beschreiben wie kleinere Objekte zu Größeren augmentiert werden, respektive wie größere Objekte zu Kleineren reduziert werden können.
Darüber hinaus muss es zwischen beiden Operationen eine Bijektion $\delta : \bigcup_{X \in \mathcal{L}} \mathcal{A}(X) \to \bigcup_{Y \in \mathcal{L}} \mathcal{R}(Y)$ geben, sodass eine für eine Augmentierung $A \in \mathcal{A}(X)$ gilt $\delta(A) = R \in \mathcal{R}(Y)$ genau dann, wenn Y aus X durch Augmentierung hervorgeht und X aus Y durch Reduktion von Y zurückzuführen ist.

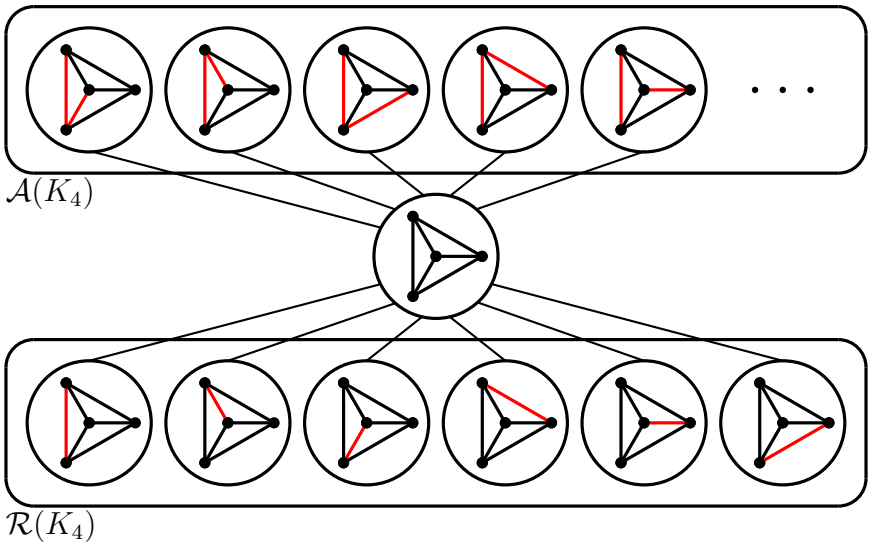


Abb. 3.1.: Augmentierungs- und Reduktionsoperationen für das kleinste kubische Beispielobjekt des Graphen K_4.

Beispiel: Als Basismenge $\mathcal{B}$ benötigen wir zunächst die Menge aller kubischen Primgraphen, auf der wir unsere Augmentierungs- und Reduktionsoperationen ausführen können. Der Grund, warum wir nicht parallel kubische Graphen und kubische Primgraphen generieren können, erläutern wir in Abschnitt 3.1, wenn wir auf die auftretenden Fallstricke aufmerksam machen.

Eine Augmentierung eines kubischen Primgraphen oder eines allgemeinen kubischen Graphen haben wir bereits als Kanteneinfügung definiert. Wir können also eine Augmentierung $\mathcal{A}(G)$ eines kubischen Graphen $G = (V, E)$ als einfaches Tripel des Graphen und zweier Kanten (G, uv, xy) angeben, wobei $uv, xy \in E$. Analog dazu definieren wir die inverse Operation der Reduktion, welche wir als Kantenentfernung eingeführt haben. Diese geben wir als einfaches Tupel (G, uv) an, wobei $uv \in E$. In Abbildung 3.1 sehen wir beispielhaft die Menge aller Augmentierungsoperationen mit $|\mathcal{A}(K_4)| = 15$ und die Menge aller Reduktionsoperationen mit $|\mathcal{R}(K_4)| = 6$. Allgemein gilt für einen kubischen Graphen $G = (V, E)$ der Größe $|V| = n$: $|\mathcal{A}(G)| = (n(n-1))/2$ und $|\mathcal{R}(G)| = 3n/2$.

Selbstverständlich führen viele verschiedene Augmentierungen zu paarweise isomorphen Graphen. In diesem Fall führen sogar alle Reduktionsoperationen zu ungültigen Ergebnissen, da K_4 der kleinste kubische Graph ist. Praktisch ist dabei die Definition der Kanteneinfügungsoperation, da wir aus Abschnitt 2.2 wissen, dass ein kubischer Graph G nach einer Augmentierung selbst wieder kubisch ist, was dazu führt dass die Eigenschaft der Funktion $P(G)$ stets erfüllt bleibt. Auch die Bijektion zwischen beiden Operationen ist einfach nachzuvollziehen.

3.2. Augmentierungen und Orbits

Wie wir bereits an Abbildung 3.1 gesehen habe, treten weiterhin Probleme auf, die dazu führen können, dass paarweise isomorphe Objekte generiert werden. Viele Augmentierungen oder auch Reduktionen führen zu paarweise isomorphen Objekten.

Wir wollen dazu zunächst beschreiben, wie wir genau einen Repräsentanten eines jeden unbeschrifteten Objektes generieren und wie dieser Isomorphismus zu behandeln ist.

Sei ein unbeschriftetes Objekt $\mathcal{X} \in \mathcal{U}$ und zwei beschriftete Objekte X und $X' \in \mathcal{X}$, die folgende Bijektionen aufweisen: $\pi_{X,X'} : \mathcal{A}(X) \to \mathcal{A}(X')$ und $\rho_{X,X'} : \mathcal{R}(X) \to \mathcal{R}(X')$, sodass folgendes gilt:

- Für alle $Y \in \mathcal{A}(X)$ gilt: Sei W das Objekt, sodass $\delta(Y) \in \mathcal{R}(W)$, und W' das Objekt, sodass $\delta(\pi_{X,X'}(Y)) \in \mathcal{R}(W')$, dann sind W und W' isomorph zueinander.
- Für alle $Z \in \mathcal{R}(X)$ gilt: Sei W das Objekt, sodass $\delta^{-1}(Z) \in \mathcal{A}(W)$, und W' das Objekt, sodass $\delta(\rho_{X,X'}(Z)) \in \mathcal{A}(W')$, dann sind W und W' isomorph zueinander.

Diese Eigenschaft erlaubt uns augmentierte und reduzierte Objekte $A(\mathcal{X})$ und $R(\mathcal{X})$ auch für unbeschriftete Objekte $\mathcal{X} \in \mathcal{U}$ zu definieren. Wir wollen dies erneut an unserem Beispiel zeigen.

Beispiel: Seien zwei kubische Graphen $G = (V, E)$ und $H = (V', E')$ gegeben. Diese seien isomorph zueinander über eine Bijektion $\tau : V \to V'$, dann wird eine Augmentierung $(G, uv, xy) \in \mathcal{A}(G)$ auf (H, ab, cd) abgebildet, wobei $\{ab, cd\} = \{\tau(e) : e \in \{uv, xy\}\}$. Analog werden die Kantenentfernungen definiert. Dann haben jeweils zwei isomorphe Graphen G und H eine Bijektion $\pi_{G,H} : \mathcal{A}(G) \to \mathcal{A}(H)$ zwischen beiden Augmentierungen.

Darüber hinaus müssen wir nun auch Augmentierugen entfernen, die zum selben größeren Objekt führen würden. Falls ein Objekt X über zwei verschiedene Augmentierungen $A, A' \in \mathcal{A}(X)$ augmentiert werden soll, müssen wir also zuerst ausschließen, dass die beiden Augmentierungen A und A' zueinander isomorph sind, da ansonsten auch $\delta(A)$ und $\delta(A')$ isomorph sind. Durch die Berechnung von Automorphismengruppen, können wir diese Fälle ausschließen.

Beispiel: Leider kann es passieren, dass dennoch Duplikate erzeugt werden. Wir zeigen an unserem Beispiel die zu umgehenden Schwierigkeiten auf. Es kann zwei voneinander disjunkte Pfade im Konstruktionsbaum von nicht-isomorphen Augmentierungen geben, die das gleiche unbeschriftete Objekt erzeugen. Wie bereits erwähnt erzeugt eine Dracheneinfügung für den Graphen K_4 den Graphen mit dem LCF-Code $[4, 2, 3, -2, -4, -3, 2, 2, -2, -2]$, jedoch ist dieser nicht prim, und wird ebenfalls besucht werden, wenn wir eine Kanteneinfügung für einen Graphen G mit 8 Knoten ausführen. Diese Problematik umgehen wir, indem wir zunächst nur die Klasse der kubischen Primgraphen aufbauen. Jedoch ist nicht ausgeschlossen, dass dieser Fallstrick ebenfalls bei zwei Nicht-Primgraphen auftreten kann.

3.3. Kanonische Label

Um diese Problematiken zu vermeiden, die dazu führen, dass mehrere beschriftete Repräsentanten einer Isomorphieklasse bei der Suche besucht werden, benutzen wir die Berechnung von Automorphismengruppen, die es uns erlauben Invarianten zu berechnen.

Sei eine Familie von kombinatorischen beschrifteten Objekten $\mathcal{L}$ gegeben, wobei $\mathcal{U}$ die Menge der unbeschrifteten Objekte darstellt. Ein *Kanonische Beschriftung* ist eine Funktion $\ell : \mathcal{L} \to \mathcal{L}$, wobei für $U \in \mathcal{U}$ und jedes beschriftete Objekt $L, L' \in U$ die beiden beschrifteten Objekte $\ell(L)$ und $\ell(L')$ identisch sein müssen. Mithilfe dieser Abbildung, die aus einem beschrifteten Objekt L ein anderes beschriftetes Objekt $\ell(L)$ erzeugt werden kann, können wir also Invarianten für unbeschriftete Objekte berechnen.

Um solche Berechnungen durchführen zu können, wurde die nauty-Bibliothek von McKay entwickelt, um kanonische Beschriftungen effizient berechnen zu können. Der Algorithmus definiert die Funktion ℓ, und ist nicht trivial. Eine vollständige Beschreibung des nauty-Algorithmus und der kanonischen Beschriftungsfunktion ℓ ist im Paper von Hartke und Radcliffe nachzulesen. [HR09]

3.4. Kanonische Löschfunktion

Um das fundamentale Konzept zur Vermeidung von Isomorphismen dieser Such- und Aufbautechnik zu beschreiben, kommen wir schließlich zur *kanonischen Löschfunktion.* Wir definieren diese folgendermaßen:

Definition 11. *Die kanonische Löschfunktion ist eine Abbildung mit* $del(X) : \mathcal{L} \to \bigcup_{X \in \mathcal{L}} \mathcal{R}(X)$*, sodass* $del(X) \in \mathcal{R}(X)$ *und für zwei isomorphe Objekte* X *und* X'*, sodass gilt* $del(X) \simeq del(X')$*.*

Durch die Rückwärtsverfolgung der kanonischen Löschungen $del(X)$ ist eine eindeutige Folge von Augmentierungen rekonstruierbar, die diesen Löschungen nur in umgekehrter Reihenfolge entsprechen. Die genaue Definition und Erklärung der Löschfunktion ist in McKays Paper ersichtlich [McK98].

4. Implementierung der Kanteneinfügungsoperation

Die Implementierung des Generierungsalgorithmus mit dem Namen snarkhunter wurde 2011 von Brinkmann et al. [BGM11] entworfen und umgesetzt. Es existieren viele weitere Algorithmen zur Generierung von kubischen Graphen wie beispielsweise minibaum von [Bri96]). Jedoch handelt es sich bei Brinkmanns Algorithmus um den bisher schnellsten.

In diesem Abschnitt erklären wir die theoretische Implementierung der Kanteneinfügungsoperation, die wir mithilfe der Kanonischen-Konstruktionspfad-Methode umsetzen.

Wir unterteilen die Generierung in zwei Teile, indem wir zwischen kubischen Graphen mit und ohne reduzierbaren Dreiecken unterscheiden. Diese Unterteilung erlaubt es uns, da ein bedeutend großer Anteil kubischer Graphen mindestens ein *reduzierbares Dreieck* enthält, die Effizienz des Algorithmus zu erhöhen. Dazu definieren wir zunächst reduzierbare Dreiecke.

Definition 12 (Reduzierbare Dreiecke)**.** *Gegeben sei ein kubischer Graph G mit einem Kreis der Länge 3. Dieser Kreis, der aus drei Kanten und drei Knoten besteht, ist genau dann reduzierbar, wenn mindestens eine der Kanten reduzierbar ist. Wir nennen solche Kreise der Länge 3* reduzierbare Dreiecke*.*

In Abbildung 4.1 können wir ein reduzierbares Dreieck sehen. Genauer können wir sagen, dass ein zufälliger kubischer Graph mit Wahrscheinlichkeit von ungefähr $0,74$ ein reduzierbares Dreieck enthält. Genaueres zum Auftreten von reduzierbaren Dreiecken in kubischen Graphen beschreibt Wormald et al. [Wor99] im Paper über allgemein reguläre Graphen.

4.1. Kubische Graphen mit reduzierbaren Dreiecken

Betrachten wir die Kanteneinfügungsoperation für zwei Kanten, die einen gemeinsamen Knoten besitzen, so können wir diese auch so interpretieren, als dass der gemeinsame Knoten durch ein Dreieck ersetzt wird. Siehe Abbildung 4.1. Infolgedessen ist die Kanteneinfügung nicht mehr von beiden gewählten Kanten abhängig, sondern nur noch durch den gemeinsamen Knoten der durch das Dreieck ersetzt wird. Wir nennen diese Operation *Dreieckseinfügung*. Analog dazu können wir die hierzu inverse Operation einer *Dreiecksersetzung*, bei der wir ein reduzierbares Dreieck durch einen einzigen Knoten ersetzen.

Schließlich ist auch die zur Dreiecksreduzierung inverse Operation zu definieren. Gegeben sei also ein Graph $G = (V, E)$ mit reduzierbaren Dreiecken, wobei $|V| = n$ und $n \geq 6$.

Abb. 4.1.: Dreieckseinfügung und Dreiecksersetzung an den Kanten uv und uw

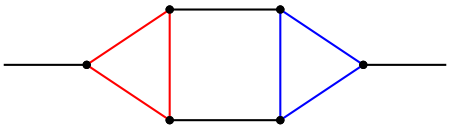


Abb. 4.2.: Sonderfall des Teilgraphen $ext(K_4)$, bei der nur eines der Dreiecke reduziert werden kann.

Nehmen wir eine Dreieckseinfügung für alle Knoten $S \subseteq V$ vor, so kann es vorkommen, dass paarweise isomorphe Graphen benötigt werden. Um das Testen auf Isomorphie zu umgehen, definieren wir eine Äquivalenzrelation für die erweiterbare Knotenmenge und führen Dreieckseinfügungen für jeweils einen Repräsentanten jeder Äquivalenzklasse ein.

Definition 13 (Erweiterbare Knotenmenge)**.**

- *Falls es einen Automorphismus γ von G mit $\gamma(S) = S'$, dann ist $S \equiv S'$*
- *Falls $|S| = |S'|$ und $(S \setminus S') \cup (S' \setminus S)$ die Menge der Verbindungsknoten eines K_4^- aus G ist, sodass jeweils eine Menge S, S' genau einen der Knoten enthält, dann ist $S \equiv S'$.*

Lemma 14 (Isomorphismus der Dreieckseinfügung)**.** *Gegeben seien ein kubischer Graph $G = (V, E)$ und zwei erweiterbare Knotenmengen S und S'. Wir bezeichnen die Graphen, die wir nach Ausführung der Dreieckseinfügung erhalten mit $D_{(G,S)} = (V_{(G,S)}, E_{(G,S)})$ und $D_{(G,S')} = (V_{(G,S')}, E_{(G,S')})$ für die erweiterbaren Knotenmengen S und S'. Dann sind die resultierenden Graphen $D(G, S)$ und $D(G, S')$ genau dann isomorph, falls $S \equiv S'$.*

Beweis. Falls $S \equiv S'$ können wir nach Definition der erweiterbaren Knotenmenge einfach einen Isomorphismus zwischen beiden Graphen konstruieren. Für die Rückrichtung nehmen wir an, dass ein Isomorphismus γ zwischen beiden Graphen $D(G, S)$ und $D(G, S')$ existiert. Wir betrachten vorher den Teilgraphen $ext(K_4^-)$ (siehe Abbildung 4.2) eines kubischen Graphen, bei dem nur eines der reduzierbaren Dreiecke reduziert werden kann, wobei offensichtlich ist, dass unabhängig von der Wahl, beide Reduktionen den gleichen Graphen erzeugen. Die Abbildung muss dabei die Menge der Teilgraphen von $ext(K_4^-)$ von $D(G, S)$ auf die Menge der Teilgraphen $ext(K_4^-)$ von $D(G, S')$ abbilden und analog alle reduzierbaren Dreiecke von $D(G, S)$ auf die reduzierbaren Dreiecke von $D(G, S')$. Wir betrachten nun eine Abbildung $\phi : V_{(G,S)} \to V_{(G,S)}$, indem die grüne Kante siehe

Abbildung 4.3 kontrahiert und die entsprechenden Knoten auf die verbleibenden Knoten abgebildet werden. Ebenso verfahren wir mit den Kanten von reduzierbaren Dreiecken. Analog definieren wir die Abbildung $\phi' : V_{(G,S'))\to V}$. Wir erhalten durch Anwendung des Isomorphismus γ und der beiden Abbildungen ϕ und ϕ' eine Permutation γ_0 der Knoten V von G, sodass $\gamma_0(\phi(v)) = \phi'(\gamma(v))$ für alle $v \in V_{G,S}$, die ein Automorphismus für G ist. Genauer wird S durch die Abbildung γ_0 auf S' abgebildet. Ausnahmen sind falls die Menge S einen Knoten w aus einem Drachen enthält und keinen anderen. In beiden Fällen ist nach Definition jedoch $S \equiv S'$. □

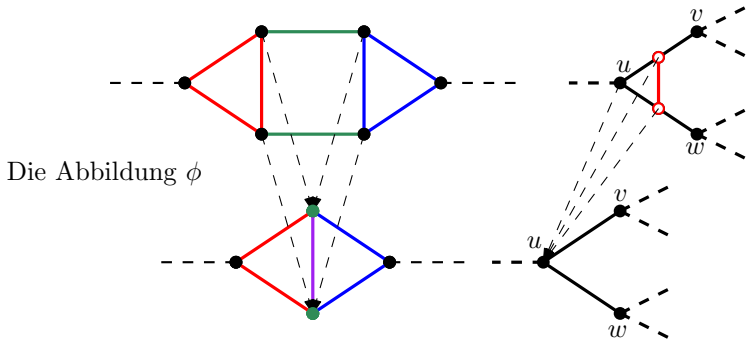


Abb. 4.3.: Abbildung der Knoten von $\phi' : V_{(G,S'))\to V}$

Durch diese Eigenschaft können wir jedes reduzierbare Dreieck gebündelt reduzieren und definieren daher eine *gebündelte Dreiecksreduktion* als gleichzeitiges Ausführen der Dreiecksreduktion für alle reduzierbaren Dreiecke. Da jeder kubische Graph mit reduzierbaren Dreiecken, der aus einer gebündelten Dreiecksreduktion hervorgeht, daher eindeutig bestimmt ist, halten wir folgenden Erkenntnis in einem Korollar fest.

Korollar 15 (Generierung durch Dreieckseinfügung)**.** *Sei jeweils ein Vertreter jeder Isomorphieklasse zusammenhängender kubischer Graphen bis zu $n-2$ Scheitelpunkten gegeben. Dann erzeugt eine gebündelte Dreieckseinfügung für jeweils einen Repräsentanten jeder Äquivalenzklasse der erweiterbaren Knotenmengen, die zu einem kubischen zusammenhängenden Graphen mit n Scheitelpunkten führt, genau einen Repräsentanten für jede Isomorphieklasse von kubisch zusammenhängenden Graphen mit n Knoten, die reduzierbare Dreiecke enthalten.*

Als nützliche Konsequenz aus dem Korollar können wir folgern, dass für Graphen mit reduzierbaren Dreiecken also kein Test auf Isomorphie benötigt wird. Es bleibt daher nur die Berechnung Automorphismengruppen, um jeweils einen Vertreter der Äquivalenzrelation über die erweiterbaren Knotenmengen zu finden.

Die Berechnung dieser Automorphismengruppen wird dabei durch das Unterprogramm nauty vorgenommen. Eine genaue Beschreibung der Berechnung von Automorphismengruppen kann im Paper von Mckay [MP14] nachgeschlagen werden. Brinkmann et al. [BGM11] erklären in ihrem Paper weitere Eigenschaften zur Optimierung der Laufzeit, indem die rechenintensive Berechnung der Automorphismengruppen durch nauty möglichst vereinfacht wird.

4.2. Kubische Graphen ohne reduzierbare Dreiecke

Bisher haben wir uns trotz der ausführlichen Einführung der Kanteneinfügungsoperation in Kapitel 2 und der beispielhaften Erklärung der Kanonischen-Konstruktionspfadmethode in Kapitel 3 die Implementierung nur durch Dreieckseinfügungen durchgeführt. Nun wollen wir die Kanteneinfügungsoperation so einsetzen, dass wir sicherstellen, dass keine reduzierbaren Dreiecke bei der Augmentierung entstehen. Wir nennen ein solches Kantenpaar *augmentierbar*. Wieder müssen wir jedoch sicherstellen, dass nur paarweise nicht-isomorphe Graphen generiert werden. Um dies umzusetzen, definieren die Autoren ein 7-Tupel $(x_0, x_1, x_2, x_3, x_4, x_5, x_6)$ für jede reduzible Kante, die Invarianten der kombinatorischen Objekte darstellen. Dabei ist die Wertigkeit lexikografisch absteigend angeordnet, sodass x_0 die größte Wertigkeit hat. Für die genaue Umsetzung verweisen wir auf das Paper von Brinkmann et al. [BGM11].

Korollar 16 (Isomorphismus reduzibler Kanten)**.** *Es seien zwei zusammenhängende kubische Graphen $G = (V_G, E_G)$ und $H = (V_H, E_H)$ mit reduziblen Kanten gegeben. Seien e_G, e_H diejenigen reduziblen Kanten, von G und H, die beide die lexikografisch größte Tupel zugewiesen bekommen haben. Dann ist $\gamma(e_G)$ im gleichen Orbit wie e_H unter der Automorphismengruppe von H. Die Reduktion beider Kanten e_G, e_H führt zu zueinander isomorphen Graphen.*

Der Algorithmus würde auch funktionieren, wenn nur die letzten zwei Stellen des 7-Tupels mithilfe von der Automorphismengruppen mit nauty berechnet werden, jedoch führen Brinkmann et al. Optimierungen durch die zusätzlichen Invarianten x_0 bis x_4 ein, die sich schneller berechnen lassen und damit den Algorithmus beschleunigen [BGM11].

Lemma 17 (Generierung durch Kanteneinfügung)**.** *Wenn für einen Vertreter jeder Isomorphieklasse aller kubischen Graphen mit $n-2$ Knoten die Kanteneinfügeoperation auf ein Kantenpaar in jedem Orbit von erweiterbaren Kantenpaaren angewendet wird, wird ein nicht primer kubischer Graph mit n Knoten und ohne reduzierbare Dreiecke genau dann akzeptiert, wenn das Tupel $(x_0, \ldots, x_6)$ maximal ist. Dann wird genau ein Vertreter jeder Isomorphie Klasse von zusammenhängenden nicht-primen Graphen mit n Knoten und ohne reduzierbare Dreiecke konstruiert und akzeptiert.*

Aus beiden Korollaren können wir folgenden Satz folgern:

Satz 18 (Generierungsalgorithmus)**.** *Seien alle kubischen Primgraphen bis n Knoten gegeben. Dann generiert der beschriebene Algorithmus genau einen Repräsentanten jeder Isomorphieklasse über alle kubischen Knoten mit bis zu n Knoten.*

Beginnen wir mit allen kubischen Primgraphen mit bis zu $n-2$ Knoten, so können wir durch rekursive Anwendung der soeben beschrieben Operationen von Dreieckseinfügungen und Kanteneinfügungen jeden kubischen Graphen mit genau einem Repräsentanten jeder Isomorphieklasse bis zu n Knoten konstruieren. Dabei konstruieren wie beschrieben aus allen Graphen mit i Knoten alle Graphen mit $i+2$ Knoten, wobei $4 \leq i \leq n-2$.

5. Durchführung und SAT-Formulierung

5.1. Generierung mit snarkhunter

Wir nutzen die aktuellste Implementierung des snarkhunter Algorithmus, der unter [BGM] ebenfalls zum Download zur Verfügung steht. Die Ausgabe erfolgt in Form von einzelnen Dateien als Menge von Adjazenzmatrizen jeweils für alle Graphen mit n Knoten. Diese Adjazenzmatrizen können wir als Eingabe für die SAT-Formulierung nutzen.

5.2. Constraint Programm: SAT-Formulierung

Die SAT-Formulierung geht auf Klesen [Kle] zurück und basiert auf folgender logischer Formel, die sich die geschlossene Anordnung der Knoten in einem Zyklus in einer Permutation zunutze macht. Die Knoten werden auf einem Kreis oder genauer auf einem Polygon eingebettet.

$$\bigwedge_{x,y\in V, xy\notin E} \left[\bigvee_{v\in[x,y)} \left(\bigwedge_{u\in[x,v],w\in(v,y]} uw \notin E \right) \vee \bigvee_{v\in[y,v)} \left(\bigwedge_{u\in[y,v],w\in(v,x]} uw \notin E \right) \right]$$

Für jedes Knotenpaar der Knoten x und y, die nicht durch eine Kante verbunden sind, also für jede Nicht-Kante soll mindestens eine der folgenden beiden Statements erfüllbar sein. Es muss einen Knoten v innerhalb der Folge der Knoten von x und y geben, sodass alle Knoten von x bis v und ab v bis y Nicht-Kanten sind. Analog funktioniert der rechte Teil der Formel.

Wir können nun aus der logischen Formel eine SAT-Formulierung herausarbeiten, die wir entsprechend implementieren können. Die Implementierung der SAT-Formulierung erfolgt in IBMs CPLEX STUDIO, der einen Constraint Program Solver anbietet, der die Erfüllbarkeit der Formulierung testet. Ist die Formel erfüllbar, so erhalten wir eine Permutation als Ausgabe. Gibt es keine Lösung für die Formulierung, so gibt es auch keine 1-Außenhindernis-Darstellung des Graphen mit Knoten in konvexer Lage. Die Formulierung ist also eine notwendige Bedingung.

6. Auswertung und Ausblick

6.1. Ergebnisse

Durch die Generierung mit snarkhunter erhielten wir zunächst folgende Mengen von kubischen Graphen und deren Anzahl, die folgender Tabelle entnommen werden können. Dabei ist der Anteil der kubischen Primgraphen äußerst klein und das Verhältnis nimmt mit steigender Knotenanzahl immer weiter ab [BGM11].

|V|	Kubische Primgraphen	Kubische Graphen
4	1	1
6	0	2
8	1	5
10	1	19
12	1	85
14	3	509
16	2	4060
18	5	41301

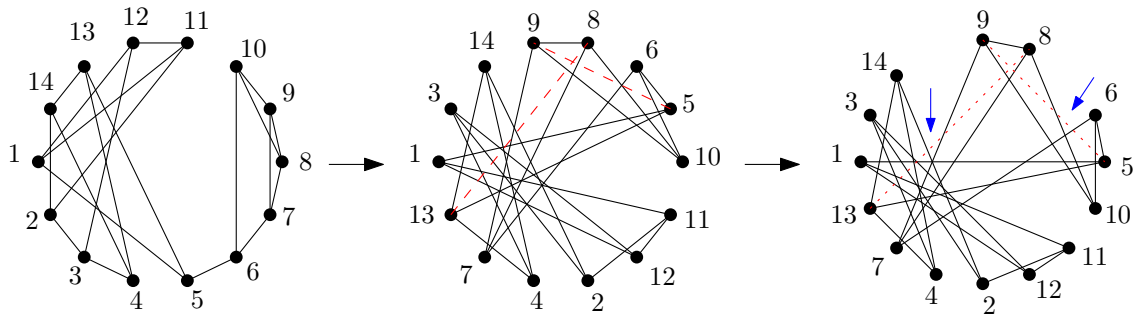


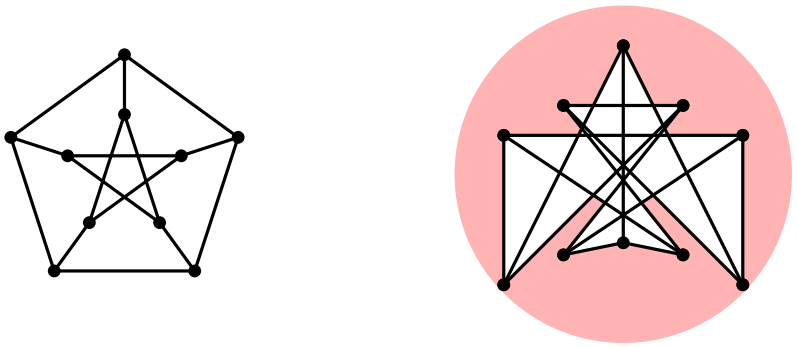
Abb. 6.1.: 1-Außenhindernisdarstellung in konvexer Lage eines kubischen Graphs mit 14 Knoten. (a) Anordnung der Knoten wie sie dem Algorithmus übergeben werden. (b) Ausgabe der Permutation durch die SAT-Formulierung und Anordnung in einem regelmäßigen 14-Eck. Wobei die zwei roten gestrichelten Kanten noch zu Nicht-Kanten gemacht werden müssen. (c) Vergrößerung der Lücken zwischen den Knoten 5 und 6, sowie 9 und 14, sodass die zwei roten gepunkteten Nicht-Kanten auch durch das Außenhindernis behindert werden.

Durch Ausführen der SAT-Formulierung für alle kubischen Graphen bis 16 Knoten mithilfe von CPLEX Studio, konnten wir zeigen, dass wir für alle kubischen Graphen,

bis auf den Petersen-Graphen eine 1-Hindernis-Darstellung mit Knoten in konvexer Lage nicht ausschließen können. Wir wissen jedoch, dass die Außenseiten-Hinderniszahl des Petersen-Graphen dennoch 1 ist. Berman et al. [BCF+17] geben dazu eine Zeichnung mit Knoten an (siehe Abbildung 6.2), die allerdings nicht in konvexer Lage liegen. Zusammen mit der Erkenntnis dieser Tatsache kann bisher festgehalten werden:

Satz 19 (Generierungsalgorithmus)**.** *Die Außenseiten-Hinderniszahl eines Graphen stimmt im Allgemeinen* nicht *mit der Außenseiten-Hinderniszahl eines Graphen mit Knoten in konvexer Lage überein.*

Um zu zeigen, dass es für alle anderen kubischen Graphen bis zu 10 Knoten eine 1-Hindernisdarstellung gibt, haben wir die Sichtbarkeitsrepräsentationen aller dieser Graphen im Anhang (siehe Abbildung B.1) angehängt.
Viele dieser kubischen Graphen können sogar so dargestellt werden, dass die Knoten in einem gleichseitigen n-Eck angeordnet werden. In einigen Fällen ist jedoch eine kleine Verschiebung der Knoten notwendig. In seltenen Fällen kommt es aber auch dazu, dass Knoten stark von ihrem ursprünglichem Platz verschoben werden müssen. Dabei ist nicht trivial, welche Knoten wie verschoben werden müssen, da es nach dem Verschieben dazu kommen kann, dass andere Nicht-Kanten entstehen.



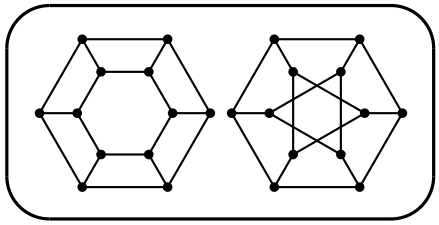
Petersen-Graph 1-Außenseitenhindernis-Darstellung des Petersen-Graphen

Abb. 6.2.: Standarddarstellung des Petersen-Graphen und 1-Außenseitenhindernis-Darstellung des Petersen-Graphen, bei dem die Knoten allerdings *nicht* in konvexer Lage liegen.

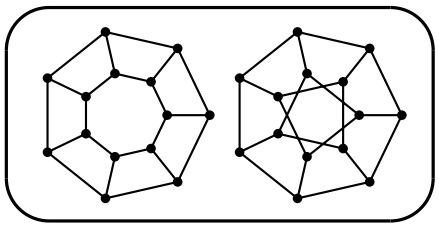
Allgemeiner Petersen-Graph

Das Auftreten des Petersen-Graphen, für den dadurch nachweislich keine Außenseiten-Hindernisdarstellung mit Knoten in konvexer Lage existiert, verleitet zu der Vermutung, dass die Familie der Allgemeinen Petersen-Graphen weiterer solcher Fälle hervorbringen könnte. Die verallgemeinerte Petersen-Graphenfamilie wurde 1950 von Coxeter [Cox50] eingeführt und erhielt 1969 ihren Namen von Watkins [Wat69].

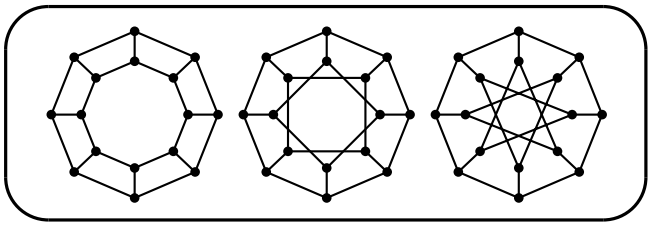
Definition 20 (Allgemeine Petersen-Graphen)**.** *Ein Graph $G = (V, E)$ aus der Familie der allgemeinen Petersen-Graphen ist definiert als Tupel $G(n, k)$ durch die Knotenmenge $V = \{u_0, u_1, \dots u_{n-1}, v_0, v_1, \dots v_{n-1}\}$ und die Kantenmenge $E = \{u_i u_{i+1}, u_i v_i, v_i u_{i+k}\}$. Die Indizes sind dabei modulo n angegeben und $k < n/2$.*



$G(6,1)$ und $G(6,2)$



$G(7,1)$ und $G(7,2)$



$G(8,1)$, $G(8,2)$ und $G(8,3)$

Abb. 6.3.: Alle kubischen Graphen der allgemeinen Petersen-Graphfamilie mit 12, 14 und 16 Knoten.

Der bekannte Petersen-Graph wird somit als $G(5,2)$ angegeben. Doch alle Graphen mit $G(9,k)$, wobei $k < n/2$ besitzen eine Darstellung mit nur einem einzigen Außenseitenhindernis in konvexer Lage.

Auch die Vermutung, dass allgemeine nicht-hamiltonsche Petersen-Graphen, wobei der kleinste nach dem Petersen-Graphen, der nicht-hamiltonsche $G(11,2)$ ist, konnte auch durch Testen des Graphen mit 22 Knoten gezeigt werden, dass wir auch in diesem Fall eine Lösung nicht ausschließen können. Jedoch konnten wir sogar eine 1-Außenhindernis-Darstellung in konvexer Lage finden, siehe Abbildung B.2 im Anhang.

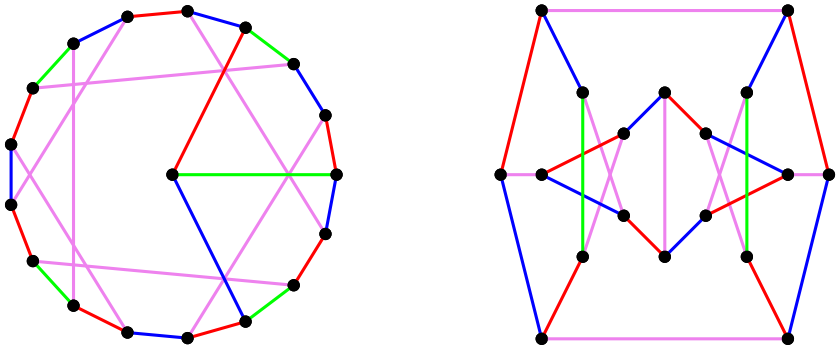
Snarks

Auch die Auswertung anderer besonderer kubischer Graphen, den sogenannten Snarks, zu welchen auch der Petersen-Graph als kleinstes Beispiel zählt, führte wieder zu gültigen Lösungen für die SAT-Formulierung, sodass wieder die notwendige Bedingung für eine 1-Außenhindernis-Darstellung in konvexer Lage des Graphen erfüllt ist.

Dabei ist ein kubischer Graph genau dann ein Snark, wenn sein chromatischer Index 4 ist. Beispiele für die nächsten kleinen Snarks mit jeweils 18 Knoten sind die beiden Blanuša-Graphen und deren 1-Hindernis-Darstellung. Siehe Abbildung 6.4.

6.2. Ausblick

Wir halten also fest, dass wir allein für den bekannten Petersen-Graphen $G(5,2)$ ausschließen können, dass eine Hindernisdarstellung mit Knoten in konvexer Lage existiert. Zusammen mit der Tatsache, dass der Graph dennoch Hinderniszahl 1 hat, wissen



Erster Blanuša Snark Zweiter Blanuša Snark

Abb. 6.4.: Beide Blanuša-Graphen mit valider Kantenfärbung mit vier Farben

wir, dass die Außenseiten-Hinderniszahl eines Graphen im Allgemeinen *nicht* mit der Außenseiten-Hinderniszahl eines Graphen mit Knoten in konvexer Lage übereinstimmt.

Interessant bleibt, ob weitere kubischen Graphen existieren, bei denen eine 1-Hindernis-Darstellung mit Knoten in konvexer Lage ausgeschlossen ist, oder ob der Petersen-Graph der einzige Graph mit dieser Eigenschaft ist. Es bietet sich vor allem die Untersuchung von weiteren nicht-hamiltonschen kubischen Graphen an oder auch die weitere Untersuchung größerer Vertreter der Familie der Snarks oder der Graphen mit Umfang (engl. girth) größer oder gleich 5, welches beides Eigenschaften sind, die auch der Petersen-Graph erfüllt.

Es lassen sich aber noch viele weitere interessante kubische Graphen mit größerer Knotenanzahl untersuchen. Auch der Pappus-Graph mit 18 Knoten liefert eine Permutation für die wir eine 1-Außenhindernis-Darstellung finden können, die sogar auf einem gleichmäßigen 18-Eck eingebettet werden kann.

Ist eine Verbesserung der SAT-Formulierung zu erreichen, lassen sich durch Parallelisierung vielleicht auch alle kubischen Graphen mit 18, 20 oder sogar 22 Knoten testen.

Zusätzlich ließen sich vielleicht auch Algorithmen entwickeln, die inkrementell Knoten verschieben können, um die Keile die in der 1-Hindernis-Darstellung zu vergrößern bis eine Lösung gefunden werden kann. So könnte man den letzten Schritt auch automatisieren und auch schneller zu mehr 1-Hindernis-Darstellungen gelangen.

Literaturverzeichnis

[AH89] Kenneth Appel und Wolfgang Haken: *Every Planar Map is Four Colorable*. American Mathematical Society, 1989. https://doi.org/10.1090/conm/098.

[AKL10] Hannah Alpert, Christina Koch und Joshua Laison: Obstacle Numbers of Graphs. *Discret. Comput. Geom.*, 44(1):223–244, 2010. https://doi.org/10.1007/s00454-009-9233-8.

[BCF+17] Leah Wrenn Berman, Glenn G. Chappell, Jill R. Faudree, John Gimbel, Chris Hartman und Gordon I. Williams: Graphs with Obstacle Number Greater than One. *J. Graph Algorithms Appl.*, 21(6):1107–1119, 2017. https://doi.org/10.7155/jgaa.00452.

[BCV15] Martin Balko, Josef Cibulka und Pavel Valtr: Drawing Graphs Using a Small Number of Obstacles. Band 9411 der Reihe *Lecture Notes in Computer Science*, Seiten 360–372. Springer, 2015. https://doi.org/10.1007/978-3-319-27261-0_30.

[BGM] Gunnar Brinkmann, Jan Goedgebeur und Brendan McKay. Homepage des snarkhunter Algorithmus: http://caagt.ugent.be/cubic/.

[BGM11] Gunnar Brinkmann, Jan Goedgebeur und Brendan D. McKay: Generation of Cubic graphs. *Discret. Math. Theor. Comput. Sci.*, 13(2):69–80, 2011. http://dmtcs.episciences.org/551.

[Bri96] Gunnar Brinkmann: Fast generation of cubic graphs. *J. Graph Theory*, 23(2):139–149, 1996.

[CLPW16] Steven Chaplick, Fabian Lipp, Ji-won Park und Alexander Wolff: Obstructing Visibilities with One Obstacle. Band 9801 der Reihe *Lecture Notes in Computer Science*, Seiten 295–308. Springer, 2016. https://doi.org/10.1007/978-3-319-50106-2_23.

[Cox50] Self-dual configurations and regular graphs. *Bulletin of the American Mathematical Society*, Seiten 413–455, 1950, ISSN 1088-9485.

[DV89] Jan De Vries: Over vlakke configuraties waarin elk punt met twee lijnen incident is. *Verslagen en Mededeelingen der Koninklijke Akademie voor Wetenschappen, Afdeeling Natuurkunde (3)*, 6:382–407, 1889.

[Fru77] Roberto W. Frucht: A canonical representation of trivalent hamiltonian graphs. *J. Graph Theory*, 1(1):45–60, 1977. https://doi.org/10.1002/jgt.3190010111.

[GFFS21] Behrooz Bagheri Gh., Tomás Feder, Herbert Fleischner und Carlos S. Subi: Hamiltonian cycles in planar cubic graphs with facial 2-factors, and a new partial solution of Barnette's Conjecture. *J. Graph Theory*, 96(2):269–288, 2021. https://doi.org/10.1002/jgt.22612.

[HR09] Stephen G Hartke und AJ Radcliffe: Mckay's canonical graph labeling algorithm. 2009.

[KHO+85] Harold Walter Kroto, J. R. Heath, S. C. O'Brien, R. F. Curl und R. E. Smalley: C60: Buckminsterfullerene. *Nature*, 318(6042):162–163, Nov 1985, ISSN 1476-4687. https://doi.org/10.1038/318162a0.

[Kle] Felix Klesen: SAT-Formulierung für die Hindernisdarstellung kubischer Graphen mit Knoten in konvexer Lage. Unveröffentlichtes Manuskript.

[Led65] Joshua Lederberg: DENDRAL-64: A System for Computer Construction, Enumeration and Notation of Organic Molecules as Tree Structures and Cyclic Graphs. *Part II: Topology of Cyclic Graphs. Interim Report to the National Aeronautics and Space Administration. Grant NsG*, 1965.

[McK98] Brendan D. McKay: Isomorph-Free Exhaustive Generation. *J. Algorithms*, 26(2):306–324, 1998. https://doi.org/10.1006/jagm.1997.0898.

[MP14] Brendan D. McKay und Adolfo Piperno: Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. https://doi.org/10.1016/j.jsc.2013.09.003.

[RSST96] Neil Robertson, Daniel P. Sanders, Paul Seymour und Robin Thomas: Efficiently four-coloring planar graphs. In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing - STOC '96*. ACM Press, 1996. https://doi.org/10.1145/237814.238005.

[Sto12] Derrick Stolee: Introduction to Canonical Deletion. ComputationalCombinatorics:https://computationalcombinatorics.wordpress.com/2012/08/13/canonical-deletion/, 2012.

[Wat69] Mark E. Watkins: A theorem on tait colorings with an application to the generalized Petersen graphs. *Journal of Combinatorial Theory*, 6(2):152–164, 1969, ISSN 0021-9800. https://www.sciencedirect.com/science/article/pii/S002198006980116X.

[Wor99] Nicholas Charles Wormald: *Models of random regular graphs*, Band 267 der Reihe *London Mathematical Society Lecture Note Series*, Seiten 239–298. 1999.

A. Ausgabe

Beispielhaft zeigen wir die Ausgabe des Constraint Solvers des Graphen aus Abbildung 6.1. Es werden Laufzeit, Speicherverbrauch, Suchtypus etc. ausgegeben.

```
<<< setup

<<< generate
 ! ---------------------------------------------------- CP Optimizer 20.1.0.0 --
 ! Satisfiability problem - 196 variables, 2 constraints
 ! Initial process time : 20.01s (19.95s extraction + 0.06s propagation)
 !  . Log search space  : 156.0 (before), 156.0 (after)
 !  . Memory usage      : 157.6 MB (before), 157.6 MB (after)
 ! Using parallel search with 16 workers.
 ! ------------------------------------------------------------------------------
 !               Branches  Non-fixed    W       Branch decision
                    1,000          4    1   F     0 != p#2#11
                    1,000          4    2   F     0 != p#2#11
                    1,000          2    3         0  = p#13#12
                    1,000          4    4         1 != p#13#8
 *                    487  32.92s       5         0  = p#3#6
 ! ------------------------------------------------------------------------------
 ! Search completed, 1 solution found.
 ! ------------------------------------------------------------------------------
 ! Number of branches     : 22,107
 ! Number of fails        : 10,309
 ! Total memory usage     : 2.8 GB (2.8 GB CP Optimizer + 0.0 GB Concert)
 ! Time spent in solve    : 37.24s (17.29s engine + 19.95s extraction)
 ! Search speed (br. / s) : 1,279.3
 ! ------------------------------------------------------------------------------

<<< solve

OBJECTIVE: no objective
perm =   [1 13 7 4 2 12 11 10 5 6 8 9 14 3]

<<< post process

<<< done
```

B. Sichtbarkeitsrepräsentationen verschiedener kubischer Graphen

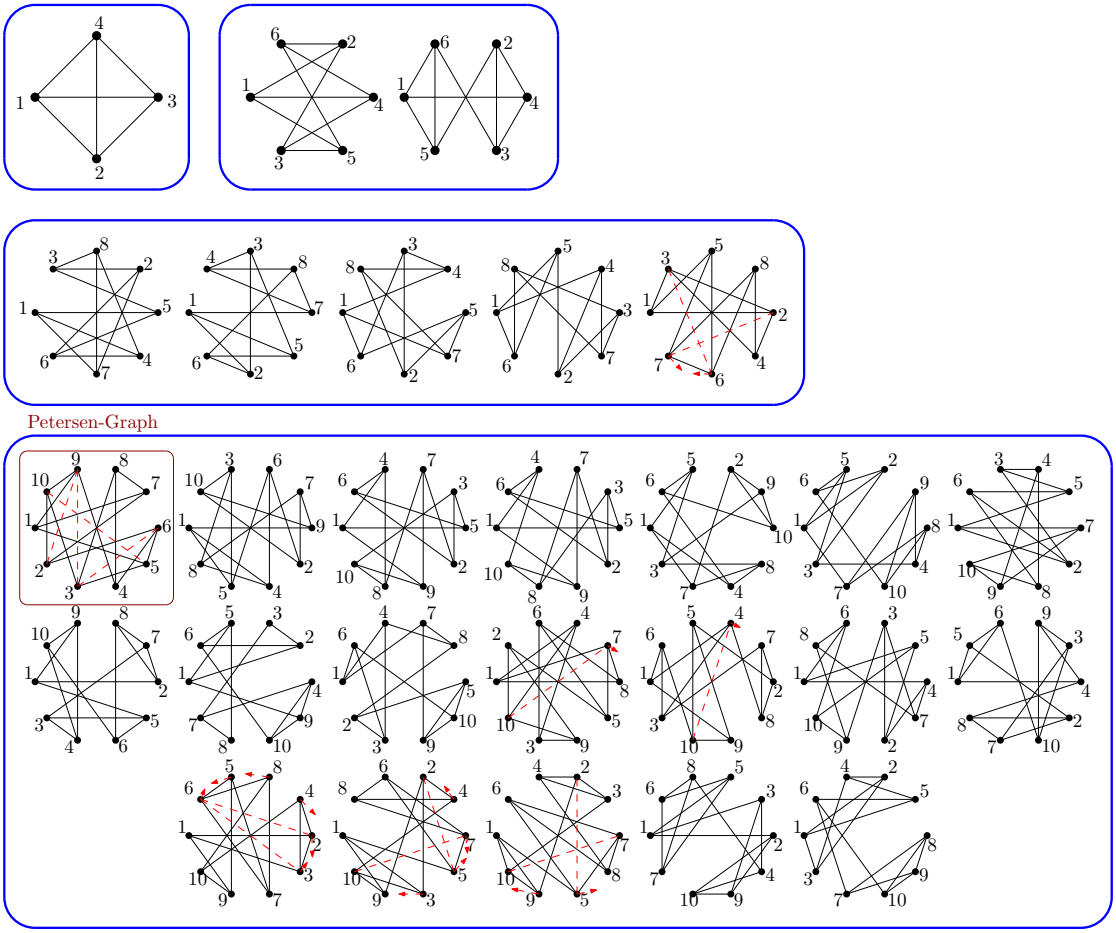


Abb. B.1.: Sichtbarkeitsrepräsentation aller kubischer Graphen mit bis zu 10 Knoten (aufgelistet von rechts nach links und von oben nach unten gemäß der Generierungsreihenfolge) mit Knoten in konvexer Lage. Mit zusätzlicher Angabe der kleinen Verschiebungskorrekturen. Mit zusätzlicher Zeichnung des Petersen-Graphen (oben links) für den es keine Darstellung gibt.

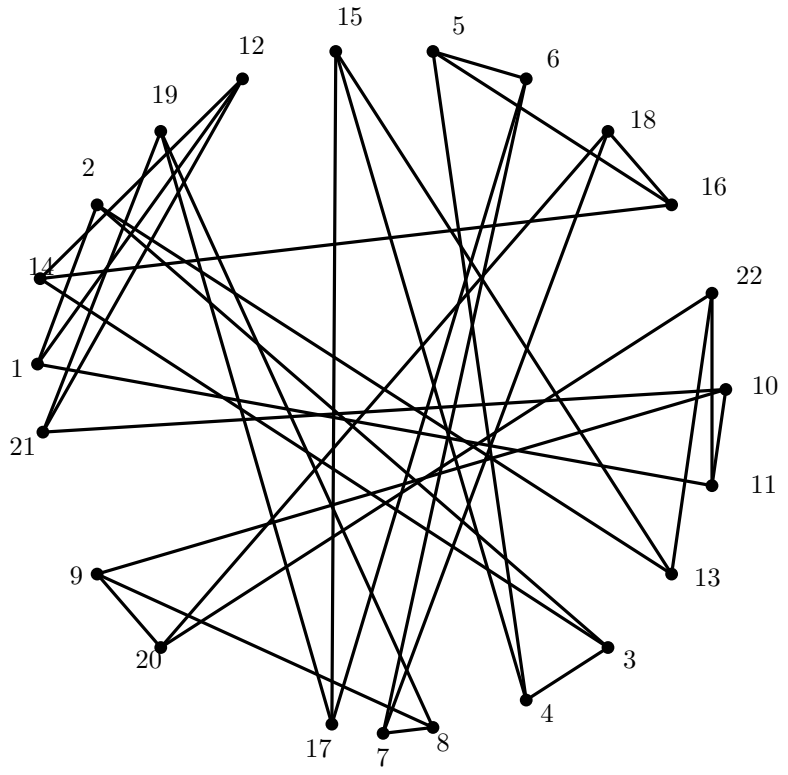


Abb. B.2.: $G(11, 2)$ der zweitkleinste nichthamiltonsche kubische Graph aus der Familie der Allgemeinen Petersen-Graphen mit seiner 1-Hindernis-Darstellung in einem nicht regelmäßigen 22-Eck.

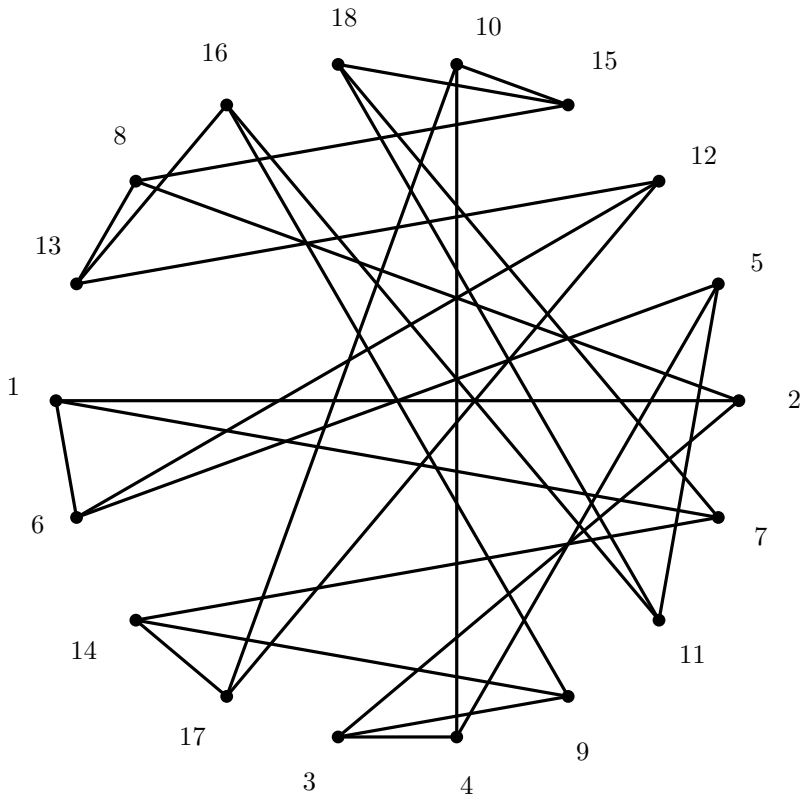


Abb. B.3.: Pappus-Graph mit $|V| = 18$ Knoten, bei dem die Knoten sogar auf einem gleichseitigem 18-Eck eingebettet werden können.

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 12. Juni 2021

. . . . . .
Christian Goldschmied