Über eine Methode zur Konstruktion von Algorithmen für die Berechnung von Invarianten in endlichen ungerichteten Hypergraphen

Von der Fakultät für Mathematik und Informatik

der Technischen Universität Bergakademie Freiberg

genehmigte

DISSERTATION

zur Erlangung des akademischen Grades

doctor rerum naturalium

(Dr. rer. nat.)

vorgelegt

von Dipl.-Math. André Pönitz,

geboren am 21. Mai 1971 in Schlema.

Gutachter: Prof. Dr. rer. nat. habil. Christoph Helmberg, Chemnitz
Prof. Dr. rer. nat. habil. Ingo Schiermeyer, Freiberg
Prof. Dr. rer. nat. Peter Tittmann, Mittweida

Tag der Verleihung: 5. Mai 2004

Die in dieser Arbeit vorgestellte *Kompositionsmethode* beschäftigt sich damit, bestimmte Aufgabenstellungen aus dem Bereich der Berechnung von Graphenkenngrößen und Grapheninvarianten in endlichen ungerichteten Graphen und Hypergraphen in ein einheitliches Schema einzuordnen und so die Umsetzung in Algorithmen zu erleichtern. Dabei werden zwei Hauptziele verfolgt. Zum einen soll die Menge der mit der Methode lösbaren Aufgaben möglichst groß sein, und zum anderen sollen die entstandenen Algorithmen tatsächliche Berechnungen in einigen Netzen praxisrelevanter Größe ermöglichen.

Die Kompositionsmethode belegt mit ihren Zielen somit den Bereich zwischen zwei Extremen der Algorithmenentwicklung: Auf der einen Seite steht die Erzeugung von Spezialalgorithmen, die oft so stark an bestimmte Eigenschaften der zu berechnenden Größen gekoppelt sind, dass eine Anpassung an leicht veränderte Aufgabenstellungen nur schwer möglich ist bzw. unter Umständen der Entwicklung eines völlig neuen Algorithmus gleichkommt; auf der anderen Seite stehen die allgemeingültigen Ansätze, deren Umsetzung häufig zu Algorithmen führt, die bereits für sehr kleine Netze nicht mehr praktisch durchführbar sind.

Die gestellten Ziele werden durch eine Formalisierung der Aufgabenstellungen erreicht, deren Ergebnisse direkt in Algorithmen umgesetzt werden können. Dabei müssen jeweils nur wenige aufgabenspezifische Details formuliert werden, die anschließend in einen von der konkreten Aufgabe unabhängigen Rahmenalgorithmus eingebunden werden. Ein solches Verfahren ist aus Sicht eines Anwenders aus der Praxis besonders interessant, da der Rahmenalgorithmus nur ein einziges Mal implementiert werden muss und somit bei wiederholter Verwendung der Methode der Entwicklungsaufwand für die erzeugten *Kompositionsalgorithmen* erheblich sinkt.

Bislang wurden mit Hilfe der Kompositionsmethode zirka dreißig Problemstellungen von der Berechnung chromatischer Invarianten über das Zählen von Hamiltonkreisen bis hin zur Bestimmung von Zuverlässigkeitskenngrößen von stochastischen Netzen bearbeitet. Die von der Methode erzeugten Algorithmen sind dabei in aller Regel nicht optimal in Bezug auf Laufzeit und Speicherbedarf. Dieser Nachteil wird allerdings durch den extrem geringen Entwicklungsaufwand und durch die Anwendbarkeit der Methode auf neue Aufgabenstellungen, für die noch keine Spezialalgorithmen existieren, kompensiert. Besonders bei der Berechnung bestimmter Zuverlässigkeitskenngrößen sowie bei der Lösung von #P-vollständigen Abzählproblemen können die Kompositionsalgorithmen aber auch aktuelle Spezialalgorithmen übertreffen.

Diese Arbeit wurde gefördert durch die Deutsche Forschungsgemeinschaft im Rahmen des Schwerpunktprogrammes 1126 „Algorithmik großer und komplexer Netzwerke“.

Inhaltsverzeichnis

1 Einleitung

Reale Netze wie Straßen- oder Kommunikationsnetze werden häufig durch Graphen oder Hypergraphen modelliert, um das Netz betreffende Fragestellungen durch die Berechnung einer Graphenkenngröße des Modells beantworten zu können. Nicht selten sind diese Kenngrößen dadurch gekennzeichnet, dass ihre Berechnung zur Klasse der NP-schwierigen Probleme gehört und somit bekannte Algorithmen eine exponentiell mit der Problemgröße – in unstrukturierten Graphen also mit der Kanten- bzw. der Knotenzahl – wachsende Laufzeit aufweisen.

Entsprechend der Vielfalt der praktischen Fragestellungen kommt es häufig vor, dass zur Lösung einer bestimmten Aufgabe keine geeigneten Spezialalgorithmen bekannt sind, dass das Problem nicht effizient auf ein bekanntes transformiert werden kann und dass sehr allgemeine Ansätze zum Erzeugen von Algorithmen wie zum Beispiel bestimmte Varianten der vollständigen Enumeration an der Größe des zu berechnenden Netzmodells scheitern.

Einen Ausweg aus dieser Situation könnte eine Methode bieten, die es ermöglicht, unkompliziert neue Algorithmen zur Lösung komplexer Probleme in Hypergraphen oder gewöhnlichen Graphen zu finden. Die dabei entstehenden Algorithmen sollten leistungsfähiger als offensichtliche, „einfache“ Algorithmen sein und die Berechnung der Kenngrößen für Graphen praxisrelevanter Größe ermöglichen.

Natürlich ist die Existenz einer solchen Methode in dieser Allgemeinheit ohne Einschränkung der möglichen Fragestellungen unwahrscheinlich. Es hat sich aber gezeigt, dass die in dieser Arbeit vorgestellte *Kompositionsmethode* einen Teil dieser Aufgabe löst, indem sie es ermöglicht, bestimmte Problemstellungen in ein einheitliches Schema einzuordnen. Dieses Schema erleichtert die Umsetzung in Algorithmen erheblich, da an Stelle vollständig neuer Algorithmen jeweils nur einige wenige problemspezifische Details gefunden und implementiert werden müssen.

Die resultierenden Algorithmen sind prinzipiell für alle endlichen ungerichteten Graphen und Hypergraphen (im Rahmen dieser Einleitung kurz zusammengefasst als „Graphen“ bezeichnet) beliebiger Struktur einsetzbar. Ihre Laufzeit und ihr Speicherbedarf hängen allerdings sehr stark von der konkreten Struktur der zu berechnenden Graphen und natürlich von der jeweiligen Problemstellung ab. Für die Klasse der Graphen geringer Wegweite lässt sich dabei meist ein besonders gutes Verhalten nachweisen; umgekehrt sind die mit dieser Methode konstruierten *Kompositionsalgorithmen* für Graphen mit verhältnismäßig hoher Wegweite eher selten praktikabel. Graphen geringer Wegweite sind, anschaulich gesprochen, Graphen, die sich durch wiederholtes Aufspalten an trennenden Knotenmengen geringer Mächtigkeit „in Scheiben zerlegen“ lassen. Die Wegweite eines Graphen entspricht dann der maximalen Mächtigkeit einer solchen trennenden Knotenmenge. Eine genauere Definition folgt in Kapitel 2. Tatsächlich ist für eine hinreichend groß gewählte Schranke jeder endliche Graph ein Graph beschränkter Wegweite und damit für Kompositionsalgorithmen geeignet.

Für Problemstellungen, die prinzipiell effizient auf Graphen beschränkter Baumweite lösbar sind, können in der Regel Kompositionsalgorithmen gefunden werden, die effizient auf Gra-

phen beschränkter Wegweite sind. Die Baumweite ist dabei eine ähnliche Größe wie die Wegweite, allerdings können hier die „Scheiben“ ggf. mehr als zwei Schnittflächen haben. Auch hier sei auf das folgende Kapitel 2 mit exakten Definitionen sowie auf die ausführliche Einführung zum Thema Baumweite in den Arbeiten von Bodlaender [Bod98, Bod95] verwiesen.

Viele der in dieser Arbeit zur Illustration der Methode angeführten Beispiele fallen in diese Kategorie der effizient auf Graphen beschränkter Baumweite lösbaren Probleme. Diese berühren die Gebiete

- Enumeration: Kreisüberdeckungen, Hamiltonkreise, Hamiltonwege, aufspannende Wälder, azyklische Orientierungen, Graphentiere, Matchings.
- Polynominvarianten: chromatisches Polynom, Dominationspolynom, Unabhängigkeitspolynom, Matchingpolynom, Tutte-Polynom, verallgemeinertes chromatisches Polynom, Zuverlässigkeitspolynom, Rekonstruktionspolynom, Cliquenpolynom, Stanleys chromatische symmetrische Funktion.
- Ganzzahlige Grapheninvarianten: chromatische Zahl, Cliquenzahl, Unabhängigkeitszahl.
- Zuverlässigkeitskenngrößen: Zusammenhangswahrscheinlichkeit, paarweise Zusammenhangswahrscheinlichkeit, Rekonstruktionswahrscheinlichkeit.
- Optimierungsprobleme: Rundreiseproblem, minimale Knotenfärbung, minimale Listenfärbung, maximale unabhängige Menge, maximales Matching, minimale dominierende Menge, optimale Steinerbäume, Prize Collecting Steiner Trees.

Eine Gemeinsamkeit einiger dieser Probleme kann in [See85], [Cou90] und [Sch89] gefunden werden. Dort wurde gezeigt, dass Grapheneigenschaften, die sich durch eine monadische Logik zweiter Stufe (*monadic second order logic*, MSOL) beschreiben lassen, effizient in Graphen beschränkter Baumweite bestimmbar sind. Scheffler gab einen auf dynamischer Optimierung basierenden Algorithmus an, der mittels MSOL beschreibbare Grapheneigenschaften effizient erkennt. Die direkte Umsetzung logischer Beschreibungen führt jedoch im Allgemeinen zu wenig effizienten Algorithmen, die nicht den hier gestellten Anforderungen genügen. Die Ursache liegt insbesondere in der sehr stark exponentiell mit der Baumweite wachsenden Konstanten. Für viele Probleme lassen sich auf diese Weise Grapheneigenschaften nur in Graphen sehr geringer Baumweite (drei bis höchstens sechs, siehe [Flu97]) tatsächlich ermitteln.

Die Differenz zwischen der Klasse der Graphen beschränkter Baumweite, für die theoretisch gutes Verhalten vorhergesagt werden kann, und der Klasse der Graphen beschränkter Wegweite, für die praktisch gute Kompositionsalgorithmen erzeugt werden können, ist für den Anwender selten relevant. Obwohl erstere Klasse formal eine Obermenge der letzteren ist, hat jeder konkrete endliche Graph sowohl beschränkte Baum- als auch beschränkte Wegweite. Darüber hinaus existiert auch hier keine „theoretische Lücke“, da der von der Methode verwendete „wegorientierte“ Ansatz prinzipiell auf einen „baumorientierten“ erweiterbar ist, auch wenn dieser aus den im folgenden Absatz genannten Gründen in der vorliegenden Arbeit nicht weiter verfolgt wird.

In umfangreichen Beispielrechnungen hat sich gezeigt, dass die auf Baumzerlegung basierenden Algorithmen in der Realisierung deutlich ineffizienter sind als ihre auf Wegzerlegungen

basierenden Pendants – und zwar auch in den praktisch selten vorkommenden Fällen, in denen die Wegweite der Beispielgraphen um einen Faktor zwei bis drei über ihrer Baumweite liegt. Die Verfolgung eines baumorientierten Ansatzes wäre in der Praxis also nur bedeutsam, wenn man regelmäßig mit Netzen zu tun hat, deren Wegweite erheblich über der entsprechenden Baumweite liegt. Für dieses etwas unerwartete Ergebnis konnten zwei Gründe identifiziert werden. Zum einen benötigt die interne Verwaltung der Zerlegungen in der Baumvariante komplexere Strukturen mit entsprechenden Konsequenzen für Laufzeit und Speicherbedarf. Zum anderen werden in den hier vorgeschlagenen Algorithmen automatisch strukturelle Besonderheiten des Graphen (wie zum Beispiel Planarität) zur Reduktion eines so genannten *Zustandsraums* genutzt, ohne dass dies explizit implementiert werden muss. Die Größe dieses Zustandsraumes hat direkten Einfluss auf die Komplexität der Algorithmen. Im Gegensatz dazu verwenden die bekannten auf Baumzerlegungen basierenden Verfahren den vollen Zustandsraum oder müssen entsprechendes Wissen über strukturbedingte Optimierungsmöglichkeiten explizit im Algorithmus berücksichtigen. Da ein erklärtes Ziel der Kompositionsmethode die Erzeugung *praktikabler* Algorithmen ist, wird in der Folge ausschließlich die auf Wegzerlegungen basierende Variante dargestellt.

Ein Problem haben Algorithmen, die auf Weg- bzw. Baumzerlegung basieren, gemeinsam: Die Bestimmung einer Baum- oder Wegzerlegung minimaler Weite ist selbst ein NP-schwieriges Problem, für das es bisher keine praktisch befriedigende Lösung gibt. Selbst eine gute Approximation, die für die hier vorliegenden Zwecke völlig ausreichen würde, ist kaum möglich, da die besten bekannten unteren und oberen Schranken für die Baumweite teilweise weit auseinander liegen. Ein Überblick zu algorithmischen Tests auf diesem Gebiet liefert [KBH01]. Einen praktikablen, theoretisch aber unbefriedigenden Ausweg bieten hier mit heuristischen Algorithmen gewonnene Zerlegungen. Durch eine Kombination von dynamischer Optimierung mit eingeschränkter Suchtiefe und Nachbesserungsverfahren lassen sich schnell Wegzerlegungen bestimmen, die in den nachprüfbaren Fällen nur wenig von optimalen Zerlegungen abweichen (vgl. dazu die Diskussion verschiedener Ansätze in [JuT02]).

Ein Universalalgorithmus für einen solch großen Problemkreis wie dem angesprochenen wird sicher nicht in jedem Fall optimal sein. Im Gegenteil ist zu erwarten, dass im Fall, dass für ein bestimmtes Problem ein Spezialalgorithmus existiert, dieser erheblich besser ist als die entsprechende Variante des Universalalgorithmus. Diese Erwartung wird auch bei der vorgeschlagenen Kompositionsmethode (leider) oft erfüllt. Insbesondere für Optimierungsaufgaben wie dem Rundreiseproblem oder der Bestimmung minimaler Steinerbäume sind andere Verfahren weitaus besser geeignet. Dennoch gibt es Bereiche, in denen Kompositionsalgorithmen die Grenzen des derzeit mit Spezialalgorithmen Machbaren erreichen bzw. sogar übertreffen. Das betrifft insbesondere #P-schwierige Enumerationsprobleme auf Graphen und Probleme aus dem Gebiet der Berechnung von Zuverlässigkeitkenngrößen stochastischer Netzstrukturen. Einige Beispiele seien genannt:

- Die Berechnung des chromatischen Polynoms ist deutlich (um Größenordnungen) besser als professionelle Programme, die zum Beispiel in Maple oder Mathematica verwendet werden. Das ist auch dann so, wenn die Implementierung in diesen Computeralgebrasystemen erfolgt, also der „natürliche“ Geschwindigkeits- und Größenvorteil der kompilierten C++-Algorithmen gegenüber den in der Regel von Interpretern verarbeiteten CAS-Sprachen entfällt (vgl. Anhang C.4).

- Für Zuverlässigkeitsprobleme in Kommunikationsnetzen konnten unter anderem Schranken für die Zusammenhangswahrscheinlichkeit mit der Genauigkeit von 10^{-8} in Graphen mit einer Wegweite von bis zu 120 erzielt werden. Auch die exakte Bestimmung von Zuverlässigkeitskenngrößen geht weit über die aktuellen Ergebnisse von Yeh, Lu und Kuo [YLK02] bzw. die besseren Resultate von Carlier und Lucet [CaL96] (Wegweite bis ca. 10) hinaus, selbst wenn die der damaligen Zeit entsprechende Hardware verwendet wird.
- Die Berechnung der Rekonstruktionswahrscheinlichkeit von Secret Sharing Schemes nach [LYCK99] konnte beschleunigt werden. Eine leichte Modifikation des Kompositionsalgorithmus eröffnet die Möglichkeit, ein verallgemeinertes (und praktisch interessantes) Problem zu lösen. Eine ähnliche Modifikation ist für den ursprünglichen Spezialalgorithmus nicht offensichtlich möglich [SaP01].
- Durch die Lösung von Enumerationsproblemen in regulären Graphen (speziell in Gittergraphen) konnten bekannte Zahlenfolgen aus Sloanes „On-Line Encyclopedia of Integer Sequences“ erweitert werden, so zum Beispiel die Anzahl der Hamiltonkreise in $2n \times 2n$ Gittern (Sloanes Sequenz A003763 [Slo03]). Bei der Enumeration von azyklischen Orientierungen und Wäldern konnten Ergebnisse von Calkin *et al* [CMN03] durch Anwendung der hier vorgestellten Methode innerhalb weniger Minuten ergänzt werden (Sloanes Sequenz A080691 [Slo03]).

Bei einigen dieser Algorithmen konnten Entwicklungszeiten von deutlich unter einer Stunde erreicht werden – einschließlich der Formulierung des Problems und der Implementierung. Einen wesentlichen Beitrag dazu liefert die sehr stark an der Implementierung orientierte, aber sehr flexible Problembeschreibung durch *Konstellationen*. So umfasst die komplette Implementierung der Zählung der spannenden Wälder eines Graphen 35 Zeilen C++-Quellcode üblicher Länge, von denen lediglich vier(!) für die Implementierung problemspezifischer Details verwendet werden.

In Fällen, wo bereits formale Beschreibungen wie zum Beispiel Dekompositionsformeln der Form

$$\begin{aligned} R(G) &= a(e)\,R(G_{-e}) + b(e)\,R(G_e) \\ R(\overline{K}_n) &= r(n) \end{aligned} \tag{1.1}$$

bekannt sind, ist der Aufwand sogar noch geringer. Hierbei bezeichnen G_{-e} bzw. G_e die Graphen, die durch Entfernen bzw. Kontraktion einer Kante e aus G hervorgehen; a und b sind gegebene Funktionen auf der Kantenmenge des Graphen und $r(n)$ gibt eine Anfangsbedingung für den kantenleeren Graphen an. Das Verfahren zur Bestimmung eines Kompositionsalgorithmus ausgehend von einer Kantendekompositionsformel der Form (1.1) wurde bereits in [Poe03] vorgestellt und wird hier in Abschnitt 5.1 in etwas abgewandelter Form ausführlich behandelt. Eine entsprechende Anwendung für die Berechnung des Zuverlässigkeitspolynoms benötigt beispielsweise nur drei problemspezifische Zeilen Quellcode. Eine Implementation dieser Algorithmen ist in [Poe02] zu finden.

Im Folgenden soll die Funktionsweise von Kompositionsalgorithmen kurz umrissen werden.

Der Name der Kompositionsalgorithmen stammt von der Art der Bildung des Ergebnisses, welches Schritt für Schritt aus Teilergebnissen, die jeweils nur einen kleinen Teil des Graphen

beschreiben, zusammengesetzt wird. Ausgangspunkt der Konstruktion ist eine Wegzerlegung des Graphen. Entlang dieser Wegzerlegung wird, beginnend bei der leeren Knotenmenge, eine „Scheibe“ des Graphen bearbeitet. Diese kann durchaus so klein sein, dass sie nur einen einzelnen Knoten oder eine einzelne Kante umfasst. Nach der Bearbeitung einer Scheibe werden deren Knoten *aktiviert* genannt. Falls ein bereits aktivierter Knoten zu einem bestimmten Zeitpunkt zu keiner der Kanten in den nachfolgenden Scheiben mehr inzidiert, so wird er aufgrund einer besonderen Eigenschaft der Bearbeitungsschritte keinerlei Einfluss mehr auf den weiteren Verlauf der Berechnung haben. Er wird deshalb *deaktiviert*. Damit besteht zu jeder Zeit eine (in nichtzusammenhängenden Graphen möglicherweise leere) Menge von aktiven Knoten, die den Graphen in einen bearbeiteten und einen unbearbeiteten Teil trennen. Parallel zu dieser aktiven Knotenmenge wird eine Menge von so genannten *Zuständen* durch eine *Transformation* genannte Operation umgeformt. Diese Zustandsmenge enthält die zur Berechnung des Endresultats notwendige Information aus dem bearbeiteten Teil des Graphen. Dementsprechend startet die Zustandsmenge mit der entsprechenden Information über den leeren Graphen und endet mit ausreichenden Informationen über den Gesamtgraphen, aus denen durch eine geeignete, *Projektion* genannte Operation die Zielgröße ermittelt werden kann. Der „Trick“ dabei ist, den größten Anteil der Arbeit bereits in den Transformationsschritten durchzuführen, und dabei diese Information von der konkreten Struktur des deaktivierten Teils zu lösen, um somit möglichst frühzeitig die in Form der Zustandsmenge gesammelte Information zu komprimieren.

Die Schwierigkeit bei der Konstruktion eines Kompositionsalgorithmus für ein gegebenes Problem besteht in der Suche nach einer geeigneten Struktur, die als Zustand benutzt werden kann, sowie nach der notwendigen Transformationsfunktion. Tatsächlich ist der Ablauf des Algorithmus eindeutig durch eine Wegzerlegung eines Graphen und die genannten Strukturen bestimmt, so dass nach deren Festlegung der Algorithmus völlig automatisch erzeugt und natürlich auch ausgeführt werden kann. Die Suche nach den geeigneten Strukturen bleibt jedoch im Wesentlichen „Handarbeit“. Ein großer Teil der vorliegenden Arbeit beschäftigt sich deshalb mit Möglichkeiten, diesen manuellen Aufwand zu reduzieren.

Als Teil der Lösung dieser Aufgaben wurde die Beschreibung von Graphenkenngrößen durch *Konstellationen* gefunden. Konstellationen sind Funktionen auf der Knoten- und Kantenmenge eines Graphen mit Werten in endlichen Mengen natürlicher Zahlen. Für viele Anwendungen kann eine Konstellation als eine Partition der Knoten- und Kantenmenge eines Graphen betrachtet werden, indem der Wert der Funktion als Nummer eines Blockes einer Knoten- bzw. Kantenpartition interpretiert wird. Der Begriff der Konstellation ist jedoch allgemeiner, da auch Markierungen und bestimmte Bewertungen der Knoten und Kanten berücksichtigt werden können. Im Gegensatz zu einer Beschreibung durch logische Ausdrücke können dabei auch beliebige reelle oder noch komplizierter strukturierte Kanten- und Knotengewichte problemlos verwendet werden, ohne dass die Komplexität der Algorithmen wesentlich ansteigt.

Eine Graphenkenngröße $R(G)$ eines Graphen G ist genau dann mit Hilfe der Kompositionsmethode berechenbar, wenn es gelingt, eine Funktion val zu definieren, die jeder Konstellation C eines bestimmten *Konstellationsraumes* $\mathcal{C}(G)$ ein Element $\mathrm{val}(C)$ eines geeigneten kommutativen Monoides $(\mathcal{W}, \oplus)$ zuordnet, so dass gilt:

$$R(G) \quad = \quad \bigoplus_{C \in \mathcal{C}(G)} \mathrm{val}(C)$$

Das Monoid hat häufig die gleiche Struktur wie die zu berechnende Graphenkenngröße, im Falle von Polynominvarianten handelt es sich also meist um einen Polynomring, bei Wahrscheinlichkeiten um reelle Zahlen usw. Die Funktion val kann theoretisch immer gefunden werden, da es eine Triviallösung für dieses Problem gibt. Diese triviale Lösung liefert allerdings keine brauchbaren Algorithmen und ist somit nur für Existenzaussagen von theoretischem Interesse.

Diese zunächst einfache Darstellung von Graphenkenngrößen durch einige wenige Strukturen und Funktionen gestattet schließlich nach einigen Verfeinerungen eine einheitliche Beschreibung der neuen Kompositionsalgorithmen. Teile dieser Beschreibung können auf einige der bekannten Splitting-, Reduktions- und Dekompositionsverfahren ausgedehnt werden kann.

Die weiteren Kapitel dieser Arbeit beschäftigen sich mit einer ausführlicheren Darstellung der Kompositionsmethode sowie notwendiger Hilfsmittel. Kapitel 2 enthält Definitionen der in der weiteren Arbeit verwendeten Bezeichnungen inklusive der notwendigen Strukturen und Relationen. In Kapitel 3 werden bekannte allgemeine Ansätze zur Berechnung von Graphenkenngrößen wie Splitting-, Reduktions- und Dekompositionsverfahren beschrieben und einige Gemeinsamkeiten und Unterschiede der Verfahren dargestellt. Kapitel 4 enthält schließlich die Beschreibung der Kompositionsmethode in ihrer allgemeinen Form sowie zwei aus praktischer Sicht besonders wichtige Spezialfälle, die *Kanten-* bzw. *Knotenkomposition.* In Kapitel 5 finden sich exemplarisch einige Anwendungen der Kompositionsmethode. Der Anhang beinhaltet neben experimentellen Ergebnissen einen kurzen Einblick in das Problem der Suche nach günstigen Wegzerlegungen, das formal zwar nicht Bestandteil der Kompositionsmethode, praktisch aber eine wichtige Voraussetzung für die Ausführung der Kompositionsalgorithmen ist.

Die Ergebnisse des Autors bestehen im Wesentlichen aus der Beschreibung der Kompositionsmethode (Kapitel 4), ihrer theoretischen (Kapitel 5) und praktischen Umsetzung (Anhang C). Dazu kommt die Grundlage für die einheitliche Beschreibung von Splitting-, Reduktions- und Dekompositionsalgorithmen in Form von Kontaktzerlegungen, Konstellationen und Indizierungen. Der übrige Inhalt des Kapitel 2 sind bekannte Grundlagen der Graphentheorie. Die im Kapitel 3 beschriebenen Reduktions-, Dekompositions-, Splittingverfahren stellen ebenfalls bekanntes Material dar, neu ist hier lediglich die einheitliche Beschreibung aller vier Ansätze durch Konstellationen und ggf. Indizierungen.

2 Notation und Grundlagen

In diesem Kapitel sollen die in der restlichen Arbeit verwendeten Begriffe definiert werden. Die gewählte Notation entspricht dabei über weite Strecken der in der Literatur üblichen. An einigen Stellen wurde allerdings bewusst davon abgewichen, insbesondere wenn mehrere ähnliche Größen die gleiche oder aber leicht verwechselbare Bezeichnungen erhalten hätten.

Neben bekannten Strukturen wie Graphen, Mengenpartitionen, Baum- und Wegzerlegungen werden hier speziell für die Umsetzung der Kompositionsalgorithmen benötigte Strukturen in Form von Konstellationen, Konstellationsräumen, Indizierungen sowie Kanten- und Knotenzerlegungen definiert.

Einfache Größen wie zum Beispiel ganze Zahlen oder Kanten und Knoten eines Graphen werden wie üblich mit einzelnen kursiven lateinischen Buchstaben bezeichnet, Mengen dagegen meist mit Großbuchstaben. Für Mengen im Sinne von „Grundmengen“ oder Definitions- und Wertebereich bestimmter Funktionen sowie die für die Kompositionsmethode besonders wichtigen *Index-* und *Zustandsmengen* wird die kalligraphische Form verwendet. Kleine griechische Buchstaben sind Mengenpartitionen und Permutationen vorbehalten. Bei Funktionen mit „sprechenden Namen“, die aus mehr als einem Buchstaben bestehen, wird dieser aufrecht gesetzt. Für Indizes gelten die gleichen Regeln, allerdings kommen hier auch einbuchstabige „sprechende Namen“ vor, die ebenfalls aufrecht geschrieben werden. Als Beispiel soll der Ausdruck $\pi = \mathrm{ind}_A^{\mathrm{V}}(C)$ dienen. Entsprechend der genannten Konvention wird „π“ für eine Mengenpartition stehen, „A“ und „C“ sind gewöhnliche Mengen und „V“ ist eine nähere Bezeichnung einer Funktion „ind“. Es wurde darüber hinaus versucht, Bezeichnungen möglichst konsistent zu halten. So bezeichnet zum Beispiel überall „V“ die Knotenmenge eines (Hyper-)Graphen und „A“ eine bestimmte „interessante“ Teilmenge davon. Diese typischen Bedeutungen sind am Ende der Arbeit im Symbolverzeichnis im Anhang zusammengefasst.

Aufgrund des beschränkten Vorrats an Symbolen lassen sich Konflikte aber nicht vermeiden, so dass sich zur genannten Regel viele Ausnahmen finden werden.

2.1 Mengen und Partitionen

2.1.1 Mengen

Bezeichnen M und J Mengen von Elementen aus einer Grundmenge Ω und v ein einzelnes Element von Ω, so wird in der Arbeit anstelle von $M \cup \{v\}$ bzw. $M \setminus \{v\}$ verkürzend $M + v$ bzw. $M - v$ verwendet, um die teilweise etwas umfangreicher werdenden Formeln leichter lesbar zu machen. Aus Gründen der Konsistenz wird dann auch $M - J$ anstelle von $M \setminus J$ für die Bildung von Mengendifferenzen verwendet. Für eine Menge Ω bezeichnet 2^{Ω} die Menge aller Teilmengen von Ω. Die Symbole $\cup$ und $\cap$ werden wie üblich für die Vereinigung bzw. den Durchschnitt von Mengen verwendet. Die Anzahl der Elemente einer Menge M wird mit $|M|$ bezeichnet.

2.1.2 Multimengen

Eine *Multimenge* ist eine Menge von geordneten Paaren der Form (v, c), wobei v Element einer Grundmenge Ω und c eine nichtnegative ganze Zahl ist. Für die Operationen auf Multimengen sollen die gleichen Symbole wie für die gewöhnlichen Mengen verwendet werden. Die Tatsache, dass M eine Multimenge mit der Grundmenge Ω ist, soll durch die Bezeichnung $M \in \Omega^*$ ausgedrückt werden.

Multimengen werden im Folgenden nur selten verwendet. In jedem Fall wird darauf explizit hingewiesen werden. Der Einfachheit halber kann eine Multimenge $\{(v_1, c_1), (v_2, c_2), \ldots\}$ auch als „Menge mit mehrfachen Elementen“ $\{\underbrace{v_1, \ldots, v_1}_{c_1}, \underbrace{v_2, \ldots, v_2}_{c_2}, \ldots\}$ geschrieben werden.

Die *Kardinalität* $|M|$ einer Multimenge $M = \{(v_1, c_1), (v_2, c_2), \ldots\}$ ist durch $|M| = \sum_i c_i$ definiert.

2.1.3 Mengenpartitionen

Eine *Mengenpartition* π einer Menge A ist eine Menge nichtleerer disjunkter Teilmengen $B_1, \ldots, B_{|\pi|}$ von A, so dass $\bigcup_{i=1}^{|\pi|} B_i = A$ gilt. Die Mengen B_i werden *Blöcke* von π genannt. Die Menge aller Mengenpartitionen einer Menge A wird mit $\mathcal{P}(A)$ bezeichnet.

Folgende Operationen mit Mengenpartitionen $\pi = \{B_1, \ldots, B_{|\pi|}\} \in \mathcal{P}(A)$ werden definiert:

- *Selektion des Blockes*, der das Element $v \in A$ enthält:

$$\pi_v \quad = \quad B_i : v \in B_i$$

- *Erweiterung der Partition* π um ein Element $v \notin A$ in einem neuen Block zu einer Partition der Menge $A + v$:

$$\pi_{+v} \quad = \quad \pi + \{v\}$$

- *Erweiterung der Partition* π um die Elemente aus $M = \{v_1, \ldots, v_t\}$ mit $M \cap A = \emptyset$ in neuen Blöcken zu einer Partition der Menge $A \cup M$:

$$\pi_{+M} \quad = \quad \pi + \{v_1\} + \ldots + \{v_t\}$$

- *Erweiterung der Partition* π um ein Element $v \notin A$ in einem vorhandenen Block $B \in \pi$ zu einer Partition der Menge $A + v$:

$$\pi_{+v,B} \quad = \quad (\pi - B) + (B + v)$$

- *Entfernung eines Elements* v aus der Partition π resultierend in einer Partition von $A - v$:

$$\pi_{-v} \quad = \quad \begin{cases} (\pi - \pi_v) + (\pi_v - v) & \text{falls } |\pi_v| > 1 \\ \pi - \pi_v & \text{falls } |\pi_v| = 1 \end{cases}$$

- *Kontraktion einer Menge $M \subseteq A$* in der Partition π:

$$\pi_M \quad = \quad (\pi - \{\pi_u : u \in M\}) + \bigcup_{u \in M} \pi_u$$

Die *Verschmelzung* zweier Partitionen π_1 und π_2 mit derselben Grundmenge A kann iterativ aus π_1 durch Kontraktion der Blöcke von π_2 gewonnen werden. Diese Operation wird mit $\pi_1 \vee \pi_2$ bezeichnet. Man überzeugt sich, dass sie kommutativ ist.

Die Menge $\mathcal{P}(A)$ aller Mengenpartitionen einer Menge A bildet einen Verband, dessen Supremumbildung über die Verschmelzung erklärt wird. Maximalelement ist dabei die Partition $\{A\}$, Minimalelement die Partition $\{\{u\} : u \in A\}$.

Eine Partition $\pi \in \mathcal{P}(A')$ einer echten Teilmenge $A' \subset A$ heißt *unvollständige Partition* von A. Die Elemente von $A - A'$ heißen in diesem Zusammenhang *ausfallende Elemente* von π.

Mengenpartitionen werden in der vorliegenden Arbeit nur für Grundmengen benötigt, die Teilmengen der Knotenmenge bestimmter Graphen oder Hypergraphen sind.

2.2 Logische Operationen

Bezeichne p eine logische Aussage. Die Notation $p \,?\, a : b$ ist eine Kurzschreibweise für die *Selektion*:

$$p \,?\, a : b \quad = \quad \begin{cases} a & \text{falls } p \text{ erfüllt ist} \\ b & \text{sonst} \end{cases}$$

Die *Indikatorfunktion* $[p]$ für eine logische Aussage p ist wie folgt definiert:

$$[p] \quad = \quad \begin{cases} 1 & \text{falls } p \text{ erfüllt ist} \\ 0 & \text{sonst} \end{cases}$$

Schließlich bezeichne $z \cdot x$ für $x \in \{0, 1\}$ und für ein Element z einer Menge Z folgenden Wert aus 2^Z:

$$z \cdot x \quad = \quad \begin{cases} \{z\} & \text{falls } x = 1 \\ \emptyset & \text{falls } x = 0 \end{cases}$$

Insbesondere gilt also für eine logische Aussage p:

$$z \cdot [p] \quad = \quad \begin{cases} \{z\} & \text{falls } p \text{ erfüllt ist} \\ \emptyset & \text{falls } p \text{ nicht erfüllt ist} \end{cases}$$

Falls Z eine Multimenge, $z \in Z$ und $x \in \mathbb{N}$ sind, soll $z \cdot x$ die Multimenge $\{(z, x)\}$ bezeichnen.

Falls $\mathcal{W}$ ein Monoid mit dem Nullelement O ist, sei für beliebiges $w \in \mathcal{W}$ vereinbart:

$$w \cdot [p] \quad = \quad \begin{cases} w & \text{falls } p \text{ erfüllt ist} \\ O & \text{falls } p \text{ nicht erfüllt ist} \end{cases}$$

2.3 Graphen

2.3.1 Hypergraphen

Ein *(endlicher) Hypergraph* H ist ein Paar (V, E) bestehend aus einer Menge $V = \{v_1, \ldots, v_n\}$ von $n \in \mathbb{N}$ verschiedenen *Knoten* und einer Multimenge $E = \{e_1, \ldots, e_m\}$ aus $m \in \mathbb{N}$ *Kanten*. Jede Kante $e \in E$ ist eine Multimenge von Knoten aus V mit einer Kardinalität von mindestens zwei. Zwei Knoten u und v heißen *adjazent*, wenn es eine Kante $e \in E$ mit $\{u, v\} \subseteq e$ gibt. Ein Kante e ist *inzident* zu einem Knoten v, wenn $v \in e$ gilt. Die Menge aller Hypergraphen wird mit $\mathcal{H}$ bezeichnet. Die Anzahl der zu einem Knoten v inzidenten Kanten heißt *Grad* des Knotens und wird mit $\deg(v)$ bezeichnet. Ein Knoten vom Grad 0 heißt *isoliert*.

Die mehrfachen Elemente von E heißen *Mehrfachkanten*. Tritt ein Knoten in einer Kante mehrfach auf, so heißt diese Kante *Schlinge*. Ein Hypergraph ist *schlicht*, wenn er weder Mehrfachkanten noch Schlingen enthält. Die hier vorgestellten Algorithmen funktionieren unverändert auch für Hypergraphen mit Mehrfachkanten und Schlingen. Diese haben aber wenig Bedeutung für die gezeigten Anwendungen. Darüber hinaus verkompliziert sich die Darstellung durch die Behandlung dieser Spezialfälle, ohne dass sich wesentliche inhaltliche Änderungen ergeben. Deshalb wird im Folgenden in der Regel davon ausgegangen, dass die betrachteten Hypergraphen schlicht sind. Ausnahmen werden explizit gekennzeichnet.

In einigen Fällen ist eine lineare Ordnung für Knoten bzw. Kanten eines Hypergraphen nötig. In diesen Fällen sei Folgendes vereinbart:

1. Die Knotenmenge $V = \{v_1, \ldots, v_n\}$ ist so geordnet, dass $v_j < v_k$ genau dann gilt, wenn $j < k$ ist.
2. Sei $e = \{v_1, v_2, \ldots, v_t\} \in E$ eine Kante und π eine Permutation der Menge $\{1, \ldots, t\}$, so dass $v_{\pi_j} < v_{\pi_k}$ genau dann, wenn $j < k$ gilt. Das der Kante e *zugeordnete Wort* ist die Folge $v_{\pi_1} v_{\pi_2} \ldots v_{\pi_t}$ von Knoten aus e. Die Ordnung der Kanten wird durch die lexikographische Ordnung der ihnen zugeordneten Wörter bestimmt.

Ein Hypergraph $H' = (V', E')$ heißt *Unterhypergraph* eines Hypergraphen $H = (V, E)$, wenn $V' \subseteq V$ und $E' \subseteq E$ gilt.

Eine alternierende Folge $v_0, e_1, v_1, \ldots, e_k, v_k$ von Knoten und Kanten mit $\{v_{i-1}, v_i\} \subseteq e_i$, $i = 1, \ldots, k$, heißt *Pfad (der Länge k)* im Hypergraphen von v_0 nach v_k.

Zwei Knoten u und v heißen *verbunden*, wenn es einen Pfad von u nach v gibt. Ein Hypergraph $H = (V, E)$ ist *zusammenhängend* (kurz mit *zsh.* bezeichnet), wenn je zwei seiner Knoten verbunden sind. Die Relation „verbunden“ ist reflexiv, transitiv und symmetrisch, so dass Äquivalenzklassen miteinander verbundener Knoten entstehen. Diese bilden zusammen mit den jeweiligen inzidenten Kanten die *Zusammenhangskomponenten* $\mathrm{comp}(H)$ des Hypergraphen, deren Anzahl $|\mathrm{comp}(H)|$ mit $k(H)$ bezeichnet werden soll. Der Hypergraph H heißt *K-zusammenhängend* für eine Teilmenge K der Knoten von H, wenn die Knoten aus K zusammenhängend sind, also alle Knoten aus K in der gleichen Zusammenhangskomponente von H liegen.

Bezeichne $\mathrm{norm}(F) = \{e \in F : |e| \geq 2\}$ für eine beliebige Kantenmenge F. Es werden folgende spezielle Mengen von Kanten und Knoten eines Hypergraphen $H = (V, E)$ definiert:

- Die Menge der *Nachbarknoten* eines Knotens v:

$$N_H(v) \quad = \quad \bigcup_{e \in E : v \in e} e - v$$

- Die Menge der *Nachbarknoten* einer Knotenmenge $J \subseteq V$:

$$N_H(J) \quad = \quad \bigcup_{e \in E : e \cap J \neq \emptyset} e - J$$

- Die Menge der *Nachbarkanten* eines Knotens $v \in V$:

$$I_H(v) \quad = \quad \{e \in E : v \in e\}$$

- Die Menge der *Nachbarkanten* einer Knotenmenge $J \subseteq V$:

$$I_H(J) \quad = \quad \{e \in E : e \cap J \neq \emptyset\}$$

Hypergraphen $H = (V, E)$ können auf verschiedene Weise manipuliert werden:

- *Löschen einer Kante* $e \in E$:

$$H_{-e} \quad = \quad (V, E - e)$$

- *Löschen eines Knotens* $v \in V$:

$$H_{-v} \quad = \quad (V - v, \mathrm{norm}(\{e - v : e \in E\}))$$

- *Löschen einer Knotenmenge* $J \subseteq V$:

$$H_{-J} \quad = \quad (V - J, \mathrm{norm}(\{e - J : e \in E\}))$$

- *Kontraktion einer Knotenmenge* $J \subseteq V$ zu einem Knoten v in einem Hypergraphen:

$$H_{J,v} \quad = \quad (V - J + v, \mathrm{norm}(\{e - J + v : e \in E \wedge e \cap J \neq \emptyset\}) \cup \{e \in E : e \cap J = \emptyset\})$$

- *Kontraktion einer Kante* $e \in E$ in einem Hypergraphen:

$$H_e \quad = \quad H_{e,v} \text{ für ein beliebiges } v \in e$$

- *Einschränkung* eines Hypergraphen auf eine Teilmenge $F \subseteq E$ seiner Kantenmenge:

$$H{:}F \quad = \quad (V, F)$$

- *Einschränkung* eines Hypergraphen auf eine Teilmenge $U \subseteq V$ seiner Knotenmenge:

$$H[U] \quad = \quad (U, \mathrm{norm}(\{e \cap U : e \in E\}))$$

Die *Kontraktion einer Partition* $\pi \in \mathcal{P}(V)$ *der Knotenmenge* V *eines Hypergraphen* $H = (V, E)$ ist als Ergebnis einer Folge von Kontraktionen der Blöcke von π im Hypergraphen definiert.

Ist eine Teilmenge $A \subseteq V$ der Knotenmenge von H gegeben, so erzeugen die Komponenten des Hypergraphen die *induzierte Partition* $\text{part}(H, A) \in \mathcal{P}(A)$ *in* H *auf* A durch

$$\text{part}(H, A) \quad = \bigcup_{(W,F)\in\text{comp}(H):W\cap A\neq\emptyset} \{W \cap A\}.$$

Wenn ein Hypergraph H' aus einem Hypergraphen H durch eine Folge von Kantenlöschungen, Kantenkontraktionen und Knotenlöschungen hervorgeht, nennt man ihn *Minor* von H und schreibt $H' \leq H$.

2.3.2 Graphen

Ein Hypergraph $G = (V, E)$, für den $|e| = 2$ für alle Kanten $e \in E$ gilt, heißt *(gewöhnlicher) Graph.* Jede Kante $e \in E$ eines Graphen ist also ein ungeordnetes Paar zweier Knoten u und v, das gewöhnlich als $e = (uv)$ geschrieben wird. Die Knoten u und v heißen dabei *Endknoten der Kante.* Die Menge aller Graphen wird mit $\mathcal{G}$ bezeichnet.

Ein Graph $G' = (V', E')$ heißt *Untergraph* eines Graphen $G = (V, E)$, wenn $V' \subseteq V$ und $E' \subseteq E$ gilt. Die Begriffe *Pfad, zusammenhängend, Zusammenhangskomponenten, K-zusammenhängend, Minor* sowie die angeführten besonderen Mengen und Operationen werden für Graphen genauso definiert wie für Hypergraphen.

Ein schlichter Graph $G = (V, E)$ ist *vollständig*, wenn je zwei seiner Knoten durch eine Kante verbunden sind. Ein vollständiger Graph mit $|V| = n$ Knoten wird mit K_n bezeichnet.

Ein Pfad $v_0, e_1, v_1, \ldots, e_k, v_k$ in einem Graphen $G = (V, E)$, für den $v_i \neq v_j$ für alle $i \neq j$ gilt, heißt *Weg*; falls dagegen $v_0 = v_k$ und $v_i \neq v_j$ für alle $i \neq j : i, j \in \{1, \ldots, k\}$ gilt, so heißt der Pfad *Kreis.* Ein zusammenhängender Graph ohne Kreise heißt *Baum*. Die Knoten vom Grad höchstens eins eines Baumes heißen *Blätter*, Knoten höheren Grades *innere Knoten*.

Ein Graph der Form $(\{u, v_1, \ldots, v_t\}, \{(uv_1), \ldots, (uv_t)\})$ heißt *Stern*. Der Knoten u wird *Zentralknoten* des Sterns genannt, die Knoten v_i seine *Blätter.*

Zusätzlich zu den Operationen auf Hypergraphen gibt es für gewöhnliche Graphen:

- *Einfügen* eines mit einer Knotenmenge $J \subseteq V$ verbundenen Knotens v:

$$G_{+v,J} \quad = \quad (V + v, E + \{(uv) : u \in J\})$$

- Bildung des *Komplements* eines schlichten Graphen G:

$$\overline{G} \quad = \quad (V, \{(uv) : u, v \in V \wedge u \neq v\} - E)$$

2.3.3 Bewertungen

Sei (V, E) ein Hypergraph, Y eine Menge und $y : V \cup E \to Y$ eine Funktion. Dann heißt das Tripel (V, E, y) *bewerteter Hypergraph*. Falls es sich bei dem Hypergraphen um einen gewöhnlichen Graphen handelt, wird dieser *bewerteter Graph* genannt. Wenn im Folgenden allgemein von Graphen und Hypergraphen gesprochen wird, sind oft bewertete Graphen oder Hypergraphen gemeint, ohne dass dies im Interesse einer kompakten Darstellung besonders gekennzeichnet wird. Die jeweiligen Bewertungsfunktionen ergeben sich aus dem Kontext.

2.3.4 Kontaktgraphen

Ein Tripel $T = (V, E, K)$ heißt *Kontaktgraph*, wenn (V, E) ein Hypergraph und K eine Teilmenge von V ist. Der Hypergraph heißt in diesem Fall *Träger* von T. Falls es sich bei dem Hypergraphen um einen gewöhnlichen Graphen handelt, sind ähnliche Konstruktionen aus der Literatur (zum Beispiel [Bod98]) bekannt. Die Elemente von K werden dabei häufig *Terminalknoten* genannt. Diese Bezeichnung kollidiert jedoch mit einer anderen Verwendung des gleichen Ausdrucks in mehreren der Anwendungen im Kapitel 5 und trifft auch nicht in jedem Fall die in der Literatur übliche Bedeutung, so dass die Knoten aus K in dieser Arbeit ausschließlich als *Kontaktknoten* bezeichnet werden. Dementsprechend heißen Elemente von $V - K$ auch *Nichtkontaktknoten.* Die Menge aller Kontaktgraphen wird mit $\mathcal{T}$ bezeichnet.

Sei $T = (V, E, K)$ ein Kontaktgraph. Ein Kontaktgraph $T' = (V', E', K')$ mit $V' \subseteq V$, $E' \subseteq E$ und $V' \supseteq K' \supseteq V' \cap K$ heißt *Unterkontaktgraph* von T.

Ist $H_1 = (V_1, E_1)$ ein Hypergraph und $T_2 = (V_2, E_2, K)$ ein Kontaktgraph mit $E_1 \cap E_2 = \emptyset$ und $V_1 \cap V_2 = K$, so bezeichnet die *Summe* $H_1 + T_2$ den Hypergraphen $(V_1 \cup V_2, E_1 \cup E_2)$. Es sei darauf hingewiesen, dass diese Operation nicht kommutativ ist und nicht für beliebige Hyper- und Kontaktgraphen als Operanden definiert ist. Die Summe $(\emptyset, \emptyset) + (V_2, E_2, \emptyset)$ ist allerdings immer wohldefiniert und hat den Hypergraphen (V_2, E_2) als Wert.

Ein Hypergraph $H = (V, E) \in \mathcal{H}$ wird im Folgenden manchmal mit dem Kontaktgraphen $T = (V, E, \emptyset) \in \mathcal{T}$ identifiziert, ohne dass dies noch einmal besonders erwähnt wird. Das gilt insbesondere auch, wenn Hypergraphen bzw. Kontaktgraphen als Funktionsargumente oder als Indizes verwendet werden.

2.4 Zerlegungen

2.4.1 Kontaktzerlegung

Definition 2.1. Sei $H = (V, E)$ ein Hypergraph. Ein Paar (H_1, H_2) zweier Unterhypergraphen $H_1 = (V_1, E_1)$ und $H_2 = (V_2, E_2)$ von H heißt *Zerlegung* des Hypergraphen H an der Knotenmenge $K \subseteq V$, falls gilt:

$$\begin{aligned} V_1 \cup V_2 &= V \\ V_1 \cap V_2 &= K \\ E_1 \cup E_2 &= E \\ E_1 \cap E_2 &= \emptyset \end{aligned}$$

Für eine Zerlegung wird auch die Notation $H = H_1 \boxplus_K H_2$ verwendet.

Im Folgenden betrachten wir eine Verallgemeinerung dieses Zerlegungsbegriffes:

Definition 2.2. Sei $H = (V, E)$ ein Hypergraph. Eine Folge $(H_1, T_2, \ldots, T_t)$ bestehend aus einem Hypergraphen $H_1 = (V_1, E_1)$ und Kontaktgraphen $T_i = (V_i, E_i, K_i)$, $i = 2, \ldots, t$ heißt *Kontaktzerlegung* des Hypergraphen H, falls gilt:

1. $\bigcup_{i=1}^{t} V_i = V$,
2. $(V_i - K_i) \cap (V_j - K_j) = \emptyset \quad \forall i \neq j$,
3. $\bigcup_{i=1}^{t} E_i = E$,
4. $E_i \cap E_j = \emptyset \quad \forall i \neq j$,
5. $K_i = \{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\}$.

Auch eine Folge $(T_1, T_2, \ldots, T_t)$ bestehend aus Kontaktgraphen $T_i = (V_i, E_i, K_i)$, $i = 1, \ldots, t$, wird *Kontaktzerlegung* genannt, falls die Bedingungen 1.-5. erfüllt sind und $T_1 = (V_1, E_1, \emptyset)$ gilt.

Eine Kontaktzerlegung $(H_1, T_2, \ldots, T_t)$ eines Hypergraphen H bestimmt eine Folge $(A_1, \ldots, A_t)$ *aktiver Knotenmengen* durch

$$A_i \quad = \quad \{v \in \bigcup_{j=1}^{i} V_j : \exists e \in E_k : k > i \wedge v \in e\}, \quad i = 1, \ldots, t, \tag{2.1}$$

und eine Folge $(D_1, \ldots, D_t)$ *zu deaktivierender Knoten* durch $A_0 = \emptyset$ und

$$D_i \quad = \quad (A_{i-1} \cup V_i) - A_i, \quad i = 1, \ldots, t. \tag{2.2}$$

Die Zahl $\max_{i=1}^{t} |A_i|$ wird als *Weite* der Kontaktzerlegung $(T_1, \ldots, T_t)$ bezeichnet.

Lemma 2.3. *Ist $(H_1, T_2, \ldots, T_t)$ eine Kontaktzerlegung eines Hypergraphen H, so gilt mit den Bezeichungen aus Definition 2.2 für $i = 2, \ldots, t$:*

$$K_i \subseteq A_{i-1}.$$

Beweis. Es gilt

$$\begin{aligned} K_i &= \{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\} \\ &\subseteq \{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_k : k > i-1 \wedge v \in e\} \\ &= A_{i-1}. \end{aligned}$$

□

2.4.2 Knotenzerlegung

Zwei Klassen von Kontaktzerlegungen mit eingeschränkter Struktur haben eine besondere Bedeutung für die Kompositionsmethode. In diesem Abschnitt soll die erste dieser Klassen charakterisiert werden.

Sei $n = |V|$ und $(v_{k_1}, \ldots, v_{k_n})$ eine Permutation der Knoten V eines Hypergraphen $H = (V, E)$. Bezeichne $\mathrm{ord}(v)$ für $v \in V$ den Index i, für den gilt $v_{k_i} = v$, und bezeichne $\mathrm{last}(e)$ für $e \in E$ den Wert $\max_{v \in e} \mathrm{ord}(v)$. Für $i = 1, \ldots, n$ sei $T_i = (V_i, E_i, K_i)$ mit

$$\begin{aligned} V_i &= \bigcup_{e \in E : \mathrm{last}(e) = i} e + v_{k_i} \\ E_i &= \{e \in E : \mathrm{last}(e) = i\} \\ K_i &= V_i - v_{k_i}. \end{aligned} \tag{2.3}$$

Lemma 2.4. *Die durch (2.3) gegebenen Tripel T_i sind Kontaktgraphen.*

Beweis. Das Paar (V_i, E_i) stellt einen Hypergraphen dar, da für alle $e \in E_i$ gilt $e \subseteq V_i$. Andererseits ist $K_i \subseteq V_i$ offensichtlich. □

Satz 2.5. *Die Folge $(T_1, T_2, \ldots, T_n)$ der Kontaktgraphen aus Lemma 2.4 bildet eine Kontaktzerlegung.*

Beweis. Durch Überprüfen der Definition 2.2: Es gilt $K_1 = \emptyset$.

1. $\bigcup_{i=1}^n V_i \supseteq \bigcup_{i=1}^n \{v_{k_i}\} = V, \quad \bigcup_{i=1}^n V_i \subseteq V$ ist klar.
2. Es gilt $(V_i - K_i) \cap (V_j - K_j) = \{v_{k_i}\} \cap \{v_{k_j}\} = \emptyset$ für alle $i \neq j$.
3. Für alle $e \in E$ gilt $e \in E_{\mathrm{last}(e)}$. Damit ist jede Kante in einer der Kantenmengen enthalten und damit $\bigcup_{i=1}^n E_i \supseteq E$. Andererseits ist $\bigcup_{i=1}^n E_i \subseteq E$.
4. Eine Kante $e \in E$ gehört zu genau einer der Kantenmengen, und zwar zu $E_{\mathrm{last}(e)}$. Damit ist $E_i \cap E_j = \emptyset$ für alle $i \neq j$.
5. Zu zeigen ist $\{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\} = V_i - v_{k_i}$. (a) Sei $v \in \{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\}$. Damit gehört v zu einer Kante aus E_i und damit zu V_i. Andererseits ist $v \neq v_{k_i}$, da $v_{k_i} \notin \bigcup_{j=1}^{i-1} V_j$. (b) Wir unterscheiden zwei Fälle. Im ersten Fall soll $E_i \neq \emptyset$ gelten. Sei $v \in V_i - v_{k_i}$ beliebig. Dann ist $\mathrm{ord}(v) < i$ und $v \in V_{\mathrm{ord}(v)}$. Damit ist einerseits $v \in \bigcup_{j=1}^{i-1} V_j$ und andererseits $v \in e$ für ein $e \in E_i$. Es verbleibt der Fall $E_i = \emptyset$. Dann ist aber $V_i = \{v_{k_i}\}$ und somit $\{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\} = \emptyset = V_i - v_{k_i}$. □

Definition 2.6. Die durch Gleichung (2.3) konstruierte Kontaktzerlegung wird *Knotenzerlegung* genannt.

Lemma 2.7. *Wenn $H = (V, E)$ ein gewöhnlicher Graph ist, so sind die durch eine Knotenzerlegung $(v_{k_1}, \ldots, v_{k_m})$ bestimmten Kontaktgraphen $(T_1, \ldots, T_t)$ Sterne, wobei der Zentralknoten von T_i der Knoten v_{k_i} ist.*

Beweis. Es reicht zu zeigen, dass E_i aus Kanten der Form (xv_{k_i}) besteht. Sei $e = (uw) \in E_i$ beliebig. Da nach Definition $E_i = \{e \in E : \text{last}(e) = i\}$ gilt, ist $i = \text{last}(e) = \max_{v \in e} \text{ord}(v) = \max\{\text{ord}(u), \text{ord}(w)\}$. Es gilt also entweder $i = \text{ord}(u)$ oder $i = \text{ord}(w)$, das heißt, entweder u oder w ist gleich v_{k_i}. □

2.4.3 Kantenzerlegung

Lemma 2.8. *Sei $n = |V|$, $m = |E|$ und $(e_{k_1}, \ldots, e_{k_m})$ eine Permutation der Kanten eines Hypergraphen $H = (V, E)$. Definiere für $i = 1, \ldots, m$:*

$$\begin{aligned} V_i^{\mathrm{E}} &= e_{k_i} \\ E_i^{\mathrm{E}} &= \{e_{k_i}\} \\ K_i^{\mathrm{E}} &= e_{k_i} \end{aligned} \tag{2.4}$$

Die durch (2.4) gegebenen Tripel $T_i^{\mathrm{E}} = (V_i^{\mathrm{E}}, E_i^{\mathrm{E}}, K_i^{\mathrm{E}})$ sind Kontaktgraphen.

Beweis. Überprüfen der Definition 2.2: $K_i^{\mathrm{E}} \subseteq V_i^{\mathrm{E}}$ ist klar. $(V_i^{\mathrm{E}}, E_i^{\mathrm{E}})$ ist ein Hypergraph mit einer einzelnen Kante, die inzident zu allen Knoten ist. □

Satz 2.9. *Sei $(T_1^{\mathrm{E}}, \ldots, T_m^{\mathrm{E}})$ die durch (2.4) gegebene Folge von Kontaktgraphen. Sei $V^{\mathrm{S}} = V - \bigcup_{e \in E} e$ die Menge der isolierten Knoten von H, und bezeichne T_i^{S} den Kontaktgraphen $(\{v_i^{\mathrm{S}}\}, \emptyset, \emptyset)$ für $v_i^{\mathrm{S}} \in V^{\mathrm{S}}$. Sei weiterhin $V_i^{\mathrm{A}} = \{v_{i,1}^{\mathrm{A}}, v_{i,2}^{\mathrm{A}}, \ldots, v_{i,|V_i^{\mathrm{A}}|}^{\mathrm{A}}\}$ die Menge $V_i^{\mathrm{E}} - \bigcup_{j=1}^{i-1} V_j^{\mathrm{E}}$ für $i = 1, \ldots, m$ und schließlich bezeichne $T_{i,s}^{\mathrm{A}}$ den Kontaktgraphen $(\{v_{i,s}^{\mathrm{A}}\}, \emptyset, \emptyset)$ für $i = 1, \ldots, m$, $s = 1, \ldots, |V_i^{\mathrm{A}}|$.*

Dann bildet die Folge

$$F = (T_1^{\mathrm{S}}, \ldots, T_{|V^{\mathrm{S}}|}^{\mathrm{S}}, \underbrace{T_{1,1}^{\mathrm{A}}, \ldots, T_{1,|V_1^{\mathrm{A}}|}^{\mathrm{A}}, T_1^{\mathrm{E}}}, \underbrace{T_{2,1}^{\mathrm{A}}, \ldots, T_{2,|V_2^{\mathrm{A}}|}^{\mathrm{A}}, T_2^{\mathrm{E}}}, \ldots, \underbrace{T_{m,1}^{\mathrm{A}}, \ldots, T_{m,|V_m^{\mathrm{A}}|}^{\mathrm{A}}, T_m^{\mathrm{E}}})$$

eine Kontaktzerlegung von H.

Beweis. Seien $T_1, \ldots, T_t$ mit $T_i = (V_i, E_i, K_i)$ eine andere Benennung für die Glieder der Folge $F = (T_1^{\mathrm{S}}, \ldots, T_m^{\mathrm{E}})$ in der gleichen Reihenfolge. Bezeichne $\text{ord}(e)$ den Index i, für den gilt $e_{k_i} = e$. Wir überprüfen wieder die Definition 2.2. Es gilt:

1. Jeder Knoten v ist entweder isoliert, womit er zu V^{S} und damit zu einem der T_i^{S} gehört, oder aber er ist inzident zu einer Kante e, womit er in $V_{\text{ord}(e)}^{\mathrm{E}}$ liegt. Damit ist $\bigcup_{i=1}^{t} V_i \supseteq V$. Andererseits ist $\bigcup_{i=1}^{t} V_i \subseteq V$ klar, es gilt also $\bigcup_{i=1}^{t} V_i = V$.

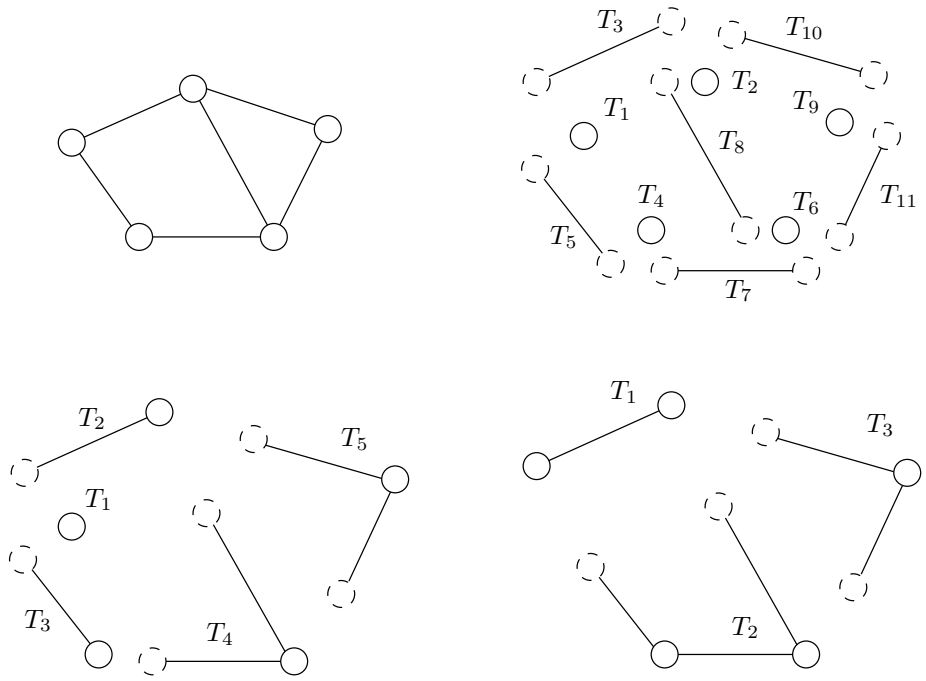


Abbildung 2.1: Ein Graph mit Kanten-, Knoten- und „unregelmäßiger“ Kontaktzerlegung. Die gestrichelten Kreise stellen Kontaktknoten dar, die durchgezogenen sind Nichtkontaktknoten.

2. Ein gegebener Kontaktgraph $T \in (T_1, \ldots, T_t)$ hat nur dann Nichtkontaktknoten, wenn es ein $T^{\mathrm{A}}_{k,l}$ mit $T^{\mathrm{A}}_{k,l} = T$ oder ein T^{S}_i mit $T^{\mathrm{S}}_i = T$ gibt. Die Nichtkontaktknotenmengen dieser Graphen sind aber gleich den entsprechenden (jeweils einelementigen) Knotenmengen, und diese wiederum sind disjunkt für zwei verschiedene Graphen dieser Typen. Für die T^{E} sind die Nichtkontaktknotenmengen nach Definition leer. Damit ist aber $V_i \cap V_j = \emptyset$ für alle $i \neq j$.

3. Sei $e \in E$. Dann ist $e \in E^{\mathrm{E}}_{\mathrm{ord}(e)}$ und folglich $\bigcup_{i=1}^{t} E_i \supseteq E$. Andererseits ist $\bigcup_{i=1}^{t} E_i \subseteq E$ klar, es gilt also $\bigcup_{i=1}^{t} E_i = E$.

4. Seien $i, j \in \{1, \ldots, t\}$ mit $i \neq j$ gegeben. Falls T_i oder T_j von der Form $T^{\mathrm{A}}_{k,l}$ oder T^{S}_s für geeignetes k, l oder s ist, ist die entsprechende Kantenmenge und folglich auch $E_i \cap E_j$ leer. Es verbleibt der Fall, dass T_i und T_j Kontaktgraphen der Form T^{E}_k und T^{E}_l für geeignetes k und l bezeichnen. Deren jeweils nur aus einer Kante bestehenden Kantenmengen sind aber disjunkt für $k \neq l$. Auch in diesem Fall gilt also $E_i \cap E_j = \emptyset$.

5. Zu zeigen ist $\{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\} = K_i$. Sei $i \in \{1, \ldots, t\}$ fixiert. Für den Fall, dass der Kontaktgraph T_i die Form T^{S}_j oder $T^{\mathrm{A}}_{k,l}$ hat, ist $T_i = (\{v_r\}, \emptyset, \emptyset)$ für ein geeignetes r und der Knoten v_r ist in keiner Knotenmenge eines T_j mit $j < i$ enthalten. Es gilt also $\{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\} = \emptyset = K_i$ wie gefordert. Es verbleibt der Fall, dass der Kontaktgraph T_i die Form T^{E}_p hat. Es gilt also $T_i = (e_r, \{e_r\}, e_r)$ für ein geeignetes r. Jeder Knoten v aus e_r ist aber nach Konstruktion spätestens in einem der unmittelbar vorausgehenden $T^{\mathrm{A}}_{i,s}$ enthalten, so dass $v \in \bigcup_{j=1}^{i-1} V_j$ ist. Aus $v \in e_r \in E_i$ folgt

sofort $K_i \subseteq \{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\}$. Sei nun $v \in \{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\}$. Da E_i aus einer einzigen Kante e_r besteht, ist $v \in e_r \in E_i$ und somit $v \in K_i$. Damit gilt $K_i \supseteq \{v \in \bigcup_{j=1}^{i-1} V_j : \exists e \in E_i : v \in e\}$.

Das erste Glied T_1 in F hat die Form T_1^{S} oder, falls H keine isolierten Knoten hat, $T_{1,1}^{A}$. In beiden Fällen hat T_1 keine Kontaktknoten, woraus die Behauptung folgt. □

Definition 2.10. Die eben konstruierte Kontaktzerlegung wird *Kantenzerlegung* genannt.

Bemerkung 2.11. Die Reihenfolge der T_k^{E} in der Kontaktzerlegung ist durch die Permutation $(e_{k_1}, \ldots, e_{k_m})$ eindeutig bestimmt. Wählt man eine lineare Ordnung $(v_1, \ldots, v_n)$ für die Knoten und bestimmt für $T_{i,j}^{\mathrm{A}} = (\{u\}, \emptyset, \emptyset\}$ und $T_{i,k}^{\mathrm{A}} = (\{v\}, \emptyset, \emptyset\}$, dass $T_{i,j}^{\mathrm{A}} < T_{i,k}^{\mathrm{A}}$ ist, falls $u < v$ bezüglich dieser Ordnung ist, so ist sogar die gesamte Kantenzerlegung eindeutig bestimmt. Da die konkrete Reihenfolge der $T_{i,j}^{\mathrm{A}}$ kaum Auswirkungen auf die praktische Umsetzung hat, wird im Folgenden die formale Abhängigkeit der Kontaktzerlegung von der Knotenordnung nicht weiter beachtet und vielmehr die Kantenzerlegung mit der Permutation der Kanten gleichgesetzt.

In Abbildung 2.1 ist ein Graph zusammen mit jeweils einer seiner Knoten-, Kanten- und „unregelmäßiger“ Kontaktzerlegungen dargestellt.

2.4.4 Baumzerlegung

In Ergänzung zu den vorstehend beschriebenen Formen von Kontaktzerlegungen, die speziell für die Kompositionsmethode konstruiert wurden, sollen noch zwei andere Zerlegungen von Graphen erwähnt werden. Diese wurden bereits im Zusammenhang mit der Konstruktion von Graphenalgorithmen verwendet (vgl. zum Beispiel [RoS86, ArP94, Bod95] und viele andere) und weisen einige zum Teil sehr weitreichende Gemeinsamkeiten mit den Kontaktzerlegungen auf. Obwohl diese „klassischen“ Zerlegungen in der Literatur meist nur im Kontext von gewöhnlichen Graphen verwendet werden, können sie auf Hypergraphen verallgemeinert werden und sollen hier der besseren Vergleichbarkeit wegen in dieser Form angegeben werden:

Definition 2.12. Sei $H = (V, E)$ ein Hypergraph. Ein Paar $(\mathcal{X}, B)$, bei dem $B = (I, F)$ ein (gewöhnlicher) Baum und $\mathcal{X} = \{X_i \subseteq V\}_{i \in I}$ eine Menge von Teilmengen von V ist, heißt *Baumzerlegung* von H, falls gilt:

1. $\bigcup_{i \in I} X_i = V$,
2. $\forall e \in E\ \exists i \in I\colon e \subseteq X_i$ und
3. $X_i \cap X_k \subseteq X_j$ für alle $i, j, k \in I$ bei denen j auf einem Weg von i nach k in B liegt.

Die Zahl $\max_{i \in I} |X_i| - 1$ wird als *Weite* der Baumzerlegung $(\mathcal{X}, B)$ bezeichnet. Das Minimum der Weiten aller Baumzerlegungen heißt *Baumweite* $\mathrm{tw}(H)$ *eines Hypergraphen*. Ein Hypergraph mit einer Baumweite von höchstens k wird *partieller k-Baum* genannt. Eine Baumzerlegung $(\mathcal{X}, B)$ mit $B = (I, F)$ und $\mathcal{X} = \{X_i\}_{i \in I}$ heißt *schön*, wenn $|X_i - X_j| + |X_j - X_i| = 1$ für alle $(ij) \in F$ und $|X_k| = 1$ für alle Blattknoten k von B gilt.

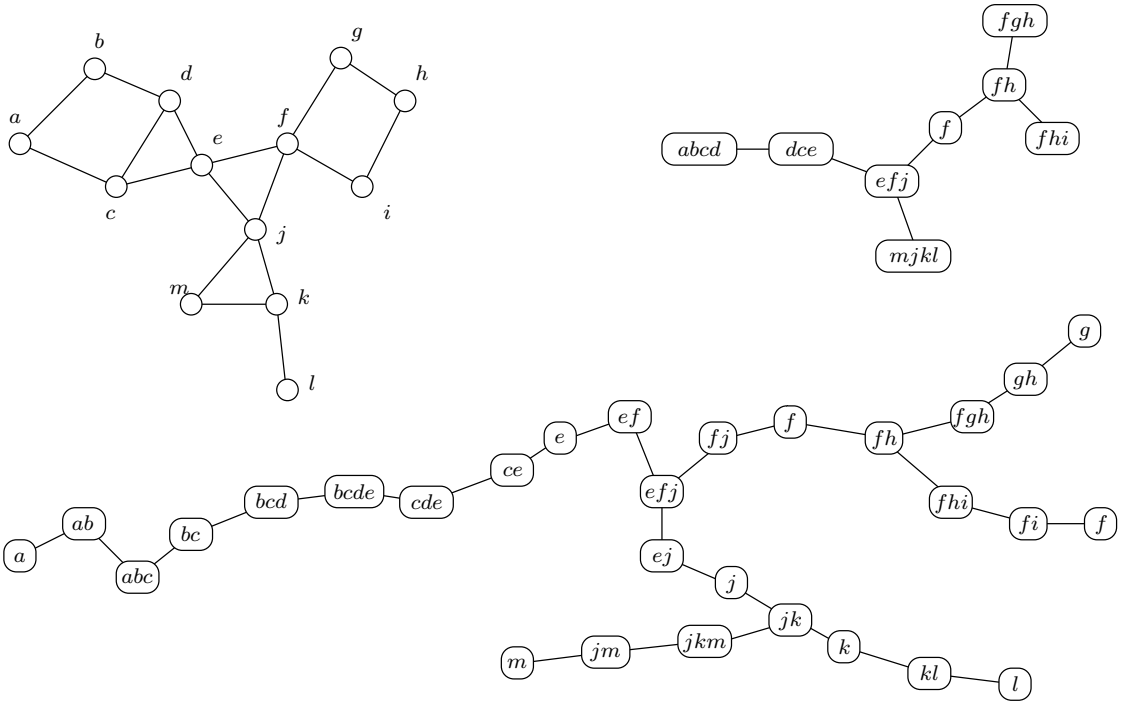


Abbildung 2.2: Graph mit Baumzerlegung und schöner Baumzerlegung

Ein Beispiel für einen Graphen mit einer Baumzerlegung und einer schönen Baumzerlegung ist in Abbildung 2.2 zu sehen.

Der theoretische algorithmische Nutzen einer begrenzten Baumweite zur Berechnung von Grapheninvarianten ist zum Beispiel aus [RoS86] bekannt.

Es ist außerdem bekannt, dass die Bestimmung der Baumweite NP-schwierig ist. Es gibt aber eine polynomiale $\mathcal{O}(\log_2 |V|)$ Approximation (vgl. dazu [Bod95]), die jedoch für unsere Zwecke aufgrund der schlechten (insbesondere von der Gesamtknotenzahl und nicht nur von der tatsächlichen Baumweite abhängenden) Approximationsgüte praktisch unbenutzbar ist, genauso wie ein exakter $\mathcal{O}(|V|^{k+2})$-Algorithmus zur Konstruktion einer Baumzerlegung für einen Graph der Baumweite k (vgl. [ACP87]), der ebenfalls nur bis zu einer Baumweite von fünf oder sechs praktikabel ist.

Wegzerlegung

Wenn der Baum B in einer Baumzerlegung $(\mathcal{X}, B)$ ein Weg ist, so bezeichnet man diese Zerlegung auch als *Wegzerlegung*. Die Zahl $\max_{i \in I} |X_i| - 1$ heißt *Weite* der Wegzerlegung $(\mathcal{X}, B)$. Das Minimum der Weiten aller Wegzerlegungen eines Hypergraphen H wird als *Wegweite* $\mathrm{pw}(H)$ bezeichnet. Hypergraphen mit einer Wegweite von höchstens k heißen *partielle k-Wege*. Auch für die Bestimmung der Wegweite sind polynomiale Approximationen bekannt, so wird zum Beispiel in [Bod95] eine $\mathcal{O}(\log_2^2 |V|)$-Approximation angegeben.

Da die Menge der Wegzerlegungen eines Hypergraphen eine Teilmenge der Menge seiner Baumzerlegungen ist, gilt $\mathrm{pw}(H) \geq \mathrm{tw}(H)$. Im Allgemeinen kann $\mathrm{pw}(H) - \mathrm{tw}(H)$ beliebig

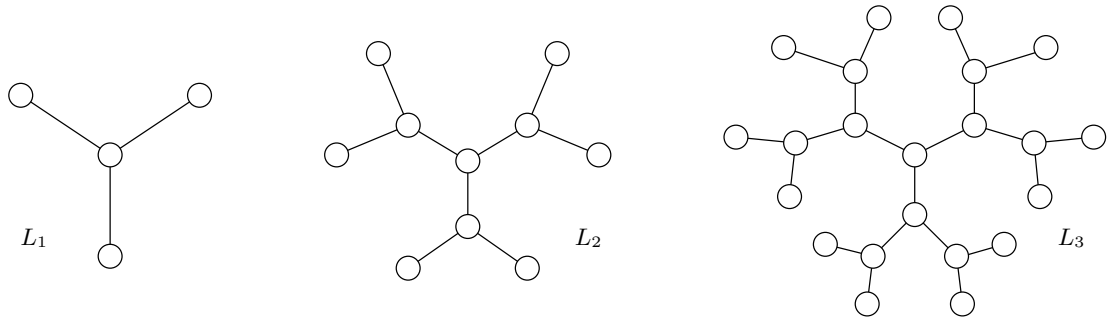


Abbildung 2.3: Folge von Bäumen mit $\mathrm{tw}(L_i) = 1$ und $\mathrm{pw}(L_i) = i$

groß werden. Ein Beispiel für eine Folge $L_1, \ldots, L_l$ von Graphen beschränkter Baumweite, aber beliebig wachsender Wegweite ist wie folgt gegeben: $L_1 = (\{v, x, y, z\}, \{(vx), (vy), (vz)\})$, und L_{i+1} entsteht für $i \geq 1$ aus L_i, indem für jedes Blatt v_k von L_i zwei neue Knoten u_k und w_k sowie zwei neue Kanten $(v_k u_k)$ und $(v_k w_k)$ eingefügt werden (vgl. dazu Abb. 2.3). Es gilt $\mathrm{tw}(L_i) = 1$ und $\mathrm{pw}(L_i) = i$.

Bemerkung 2.13. Der wesentliche Unterschied zwischen einer Wegzerlegung in der dargestellten Form und einer Kontaktzerlegung besteht in der Festlegung der Kontaktknoten der Kontaktgraphen, so dass eine Kontaktzerlegung als Erweiterung einer Wegzerlegung für Hypergraphen angesehen werden kann. Die Knotenmengen X_i der Wegzerlegung werden dabei durch die Mengen $V_i \cup A_i$ aus der Kontaktzerlegung gebildet.

2.5 Konstellationen

2.5.1 Konstellationen

Gegeben seien ein Kontaktgraph $T = (V, E, K)$ und zwei natürliche Zahlen c_V und c_E. Eine $(c_\mathrm{V}, c_\mathrm{E})$-*Konstellation von* T ist eine Funktion C, die den Nichtkontaktknoten $V - K$ von T eine natürliche Zahl aus dem Bereich 1 bis c_V und den Kanten von T eine natürliche Zahl aus dem Bereich 1 bis c_E zuordnet. Die einem Knoten v zugeordnete Zahl $C(v)$ heißt dabei *Label* des Knotens v, die Zahl $C(e)$ analog *Label* der Kante e. Eine solche Konstellation wird als Menge $\bigcup_{x \in (V-K) \cup E} \{(x, C(x))\}$ von Wertepaaren geschrieben. Als *Konstellation eines Hypergraphen* $H = (V, E)$ bezeichnet man eine Konstellation des Kontaktgraphen $T = (V, E, \emptyset)$.

Die Menge aller $(c_\mathrm{V}, c_\mathrm{E})$-Konstellationen des Kontaktgraphen T wird $(c_\mathrm{V}, c_\mathrm{E})$-*Konstellationsraum* von T genannt und mit $\mathcal{C}_T(c_\mathrm{V}, c_\mathrm{E})$ bezeichnet. Analog wird die Menge aller $(c_\mathrm{V}, c_\mathrm{E})$-Konstellationen des Hypergraphen H auch $(c_\mathrm{V}, c_\mathrm{E})$-*Konstellationsraum* von H genannt und mit $\mathcal{C}_H(c_\mathrm{V}, c_\mathrm{E})$ bezeichnet. In allen Fällen kann das Attribut $(c_\mathrm{V}, c_\mathrm{E})$ weggelassen werden, wenn dessen konkreter Wert aus dem Kontext klar oder aber uninteressant ist. Im Weiteren bezeichne $\mathcal{C}(c_\mathrm{V}, c_\mathrm{E})$ die Menge $\bigcup_{T \in \mathcal{T}} \mathcal{C}_T(c_\mathrm{V}, c_\mathrm{E})$ und $\mathrm{def} : \mathcal{C}(c_\mathrm{V}, c_\mathrm{E}) \to \mathcal{T}$ die Funktion, die jedem $C \in \mathcal{C}(c_\mathrm{V}, c_\mathrm{E})$ denjenigen Kontaktgraphen $T \in \mathcal{T}$ zuordnet, für den $C \in \mathcal{C}_T(c_\mathrm{V}, c_\mathrm{E})$ gilt. Die Mengen $\mathcal{C}_T(c_\mathrm{V}, c_\mathrm{E})$ sind für unterschiedliche T disjunkt, da die Definitionsbereiche zweier

zu verschiedenen Kontaktgraphen gehörenden Konstellationen verschieden sind. Damit ist die Funktion def wohldefiniert.

Ist $C = \bigcup_{x \in I} \{(x, C(x))\}$ mit $I = (V - K) \cup E$ eine Konstellation, so heißt die Konstellation $C|_{I'} := \bigcup_{x \in I'} \{(x, C(x))\}$ *Teilkonstellation,* falls $I^{'} \subseteq I$ gilt, und *echte Teilkonstellation*, wenn $I^{'} \subset I$ ist. Wenn $T^{'} = (V^{'}, E^{'}, K^{'})$ ein Unterkontaktgraph von T ist, so wird auch die Notation $C|_{T'}$ anstelle von $C|_{(V'-K')\cup E'}$ verwendet.

Sei $T = (V, E, K)$ ein Kontaktgraph und $T^{'} = (V^{'}, E^{'}, K^{'})$ ein Unterkontaktgraph von T. Sei $C \in \mathcal{C}_T(c_\mathrm{V}, c_\mathrm{E})$ eine Konstellation von T. Dann wird $C|_{T'} = \bigcup_{x \in (V'-K')\cup E'} \{(x, C(x))\}$ die *Einschränkung* von C auf $T^{'}$ genannt. Wenn $V^{'} \subseteq V - K$ ist, so wird $C|_{V'} = C|_{(V',\emptyset,\emptyset)}$ *Einschränkung der Konstellation C auf die Knotenmenge $V^{'}$* genannt.

Bemerkung 2.14. Für zwei häufig benötigte Spezialfälle sollen noch abkürzende Schreibweisen eingeführt werden. Der $(2,1)$-Konstellationsraum eines Kontaktgraphen $T = (V, E, K)$ besteht aus denjenigen Konstellationen, die allen Kanten von T das gleiche Label 1 und den Nichtkontaktknoten von T eines von zwei möglichen Labels (1 oder 2) zuordnen. Damit gibt es eine Bijektion $C \leftrightarrows \{v \in V - K : C(v) = 1\}$ zwischen diesen *Knotenkonstellationen* C und den Teilmengen der Menge der Nichtkontaktknoten von T, so dass anstelle von $(2,1)$-Konstellationen eines Kontaktgraphen im Folgenden einfach Teilmengen der Menge der Nichtkontaktknoten des Graphen benutzt werden. Dabei wird die einer Konstellation C zugeordnete Teilmenge der Knotenmenge mit $\hat{C}$ bezeichnet. Analog werden $(1,2)$-Konstellationen eines Kontaktgraphen mit Kantenmengen des Kontaktgraphen identifiziert und als *Kantenkonstellationen* bezeichnet. Die Bezeichnung $\hat{C}$ steht in diesem Fall für die der $(1,2)$-Konstellation C zugeordnete Kantenmenge.

Konstellationen und Konstellationsräume dienen in der folgenden Abhandlung dazu, die Vorstellung einer „vollständigen Fallunterscheidung“ formal zu fassen. Eine Konstellation beschreibt dabei einen konkreten Fall, der Konstellationsraum dagegen die Menge aller Fälle einer Fallunterscheidung. Die vorgestellten Algorithmen „leben“ davon, ihre Arbeit durch eine Fallunterscheidung zu zerlegen und die besonderen Fallvoraussetzungen zu einer effizienten Beurteilung des Beitrags des Einzelfalls zum Gesamtergebnis zu nutzen. Die Begriffe der Konstellation und des Konstellationsraums sind damit zentral für die folgenden Ausführungen.

2.5.2 Indizierungen

Definition 2.15. Sei $\mathcal{I}$ eine beliebige Menge und seien c_V und c_E zwei natürliche Zahlen. Eine Funktion $\mathrm{ind} : \{(C, A) : A \subseteq V, C \in \mathcal{C}_H(c_\mathrm{V}, c_\mathrm{E}) \text{ für } H = (V, E) \in \mathcal{H}\} \to \mathcal{I}$ wird *Indizierung* von $\mathcal{C}(c_\mathrm{V}, c_\mathrm{E})$ genannt. Die Menge $\mathcal{I}$ heißt *Indexbereich*, die Elemente von $\mathcal{I}$ *Indizes* und der Wert $\mathrm{ind}(C, A)$ der *Index der Konstellation $C \in \mathcal{C}_H(c_\mathrm{V}, c_\mathrm{E})$ mit der Grundmenge H und Zielmenge A*. Für einen Hypergraphen $H = (V, E)$ und eine Zielmenge $A \subseteq V$ wird mit $\mathcal{I}_{H,A}$ die Menge $\{\mathrm{ind}(C, A) : C \in \mathcal{C}_H(c_\mathrm{V}, c_\mathrm{E})\}$ und mit $\mathrm{ind}^{-1}_{H,A}(i)$ die Menge $\{C \in \mathcal{C}_H(c_\mathrm{V}, c_\mathrm{E}) : \mathrm{ind}(C, A) = i\}$ der von *i repräsentierten Konstellationen* bezeichnet.

Eine Indizierung ind eines Konstellationsraum $\mathcal{C}_H = \mathcal{C}_H(c_\mathrm{V}, c_\mathrm{E})$ eines Hypergraphen $H = (V, E)$ mit der Zielmenge $A \subseteq V$ zerlegt diesen Konstellationsraum in $|\mathcal{I}_{H,A}|$ Äquivalenzklassen, wobei Konstellationen zur selben Äquivalenzklasse gehören, wenn sie denselben Index

induzieren. Diese Äquivalenzklassen werden als *Konstellationsklassen* bezeichnet. In jedem Fall gilt für beliebiges $A \subseteq V$:

$$\mathcal{C}_H = \bigcup_{C \in \mathcal{C}_H} \{C\} = \bigcup_{i \in \mathcal{I}_{H,A}} \bigcup_{C \in \mathrm{ind}_{H,A}^{-1}(i)} \{C\} \tag{2.5}$$

Bemerkung 2.16. Indizierungen dienen im Folgenden nur zur Beschreibung von Äquivalenzrelationen in Konstellationsräumen, also der Zuordnung von Konstellationen zu Konstellationsklassen. Im Prinzip wäre es dafür ausreichend, die Konstellationsklassen durchzunumerieren und als Index einer Konstellation die entsprechende Konstellationsklassennummer zu verwenden, mit anderen Worten $\mathcal{I}_{H,A} = \{1, 2, \ldots, |\mathcal{I}_{H,A}|\} \subset \mathbb{N}$ zu wählen. Alternativ ist es prinzipiell ebenso möglich, die Konstellationsklassen selbst (also jeweils die Gesamtheit aller zu einer Konstellationsklasse gehörenden Konstellationen) als Indizes zu verwenden, da auch in diesem Fall Konstellationen genau dann den gleichen Index erhalten, wenn sie zur gleichen Konstellationsklasse gehören.

Für die praktische Umsetzung von Rechenverfahren, die auf Konstellationsklassen beruhen, sind diese beiden „extremen“ Indizierungen jedoch wenig vorteilhaft. Im ersten Fall geht zu viel „Strukturinformation“ verloren, so dass die später benötigten Operationen auf den Indizes unnötig kompliziert werden; im zweiten Fall führt die explizite Auflistung zu inakzeptablem Speicherbedarf und Laufzeitverhalten. Praktikable Lösungen sind dagegen in der Regel Indizes, die aus Teilmengen oder Mengenpartitionen der Zielmenge A oder aber Kombinationen solcher Objekte bestehen.

Im Verlauf eines der im Kapitel 4 beschriebenen Kompositionsalgorithmen werden mehrere verschiedene Zielmengen verwendet; die in Abschnitt 3.3 behandelten Splittingalgorithmen verwenden dagegen lediglich eine Zielmenge. In beiden Fällen hat die Wahl der Zielmengen entscheidenden Einfluss auf die Leistung der Algorithmen. Als Regel gilt, dass sie so klein wie möglich zu wählen sind. Aufgrund zusätzlicher Restriktionen existiert aber eine „natürliche“ untere Schranke für die Mächtigkeit der Zielmengen. Diese Schranke kann immer erreicht werden, so dass es sich anbietet, die Zielmengen so zu wählen, dass dieses Minimum realisiert wird.

2.5.3 Graphenkenngrößen

Definition 2.17. Sei $\mathcal{W}$ ein Menge von *Werten*, die zusammen mit einer zweistelligen kommutativen Operation $\oplus : \mathcal{W} \times \mathcal{W} \to \mathcal{W}$ einen Monoid bildet, dessen neutrales Element als O geschrieben wird. Eine Funktion $R : \mathcal{H} \to \mathcal{W}$ wird im Folgenden als *Graphenkenngröße* bezeichnet.

Definition 2.18. Sei $R : \mathcal{H} \to \mathcal{W}$ eine Graphenkenngröße. Gibt es eine Funktion $\mathrm{val} : \mathcal{C}(c_\mathrm{V}, c_\mathrm{E}) \to \mathcal{W}$, so dass

$$R(H) = \bigoplus_{C \in \mathcal{C}_H(c_\mathrm{V}, c_\mathrm{E})} \mathrm{val}(C) \quad \forall H \in \mathcal{H} \tag{2.6}$$

gilt, so wird das Tupel $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val})$ als *formale Beschreibung in Grundform* (oder kurz *Grundform*) der Graphenkenngröße R bezeichnet. Die Funktion val heißt *Bewertungsfunktion* und der Wert $\mathrm{val}(C)$ *Bewertung* der Konstellation C.

Bemerkung 2.19. Jede Graphenkenngröße hat mindestens eine formale Beschreibung in Grundform, da eine solche zum Beispiel mit $c_V = c_E = 1$, $\mathcal{W} = \{R(H) : H \in \mathcal{H}\}$, $\oplus$ beliebig, und $\text{val}(C_0) = R(H)$ für das einzige Element C_0 des $(1,1)$-Konstellationsraumes von H gegeben ist. Diese *triviale Grundform* ist für die Erzeugung neuer Algorithmen allerdings wertlos, da zur Bestimmung von $\text{val}(C_0)$ der Wert $R(H)$ bestimmt, also das ursprüngliche Problem immer noch gelöst werden muss.

Bemerkung 2.20. Wie der Name „Grundform“ schon andeutet, werden noch weitere formale Beschreibungen von Graphenkenngrößen eingeführt werden. Einen Überblick über diese gibt Abbildung 2.4. Diese formalen Beschreibungen bilden die Grundlage für die nachfolgend behandelten Algorithmen. Die durchgezogenen Pfeile in der Abbildung geben an, in welche Richtungen formale Beschreibungen automatisch umgewandelt werden können, der gestrichelte Pfeil zeigt einen Übergang, der nur sehr wenig „Handarbeit“ erfordert.

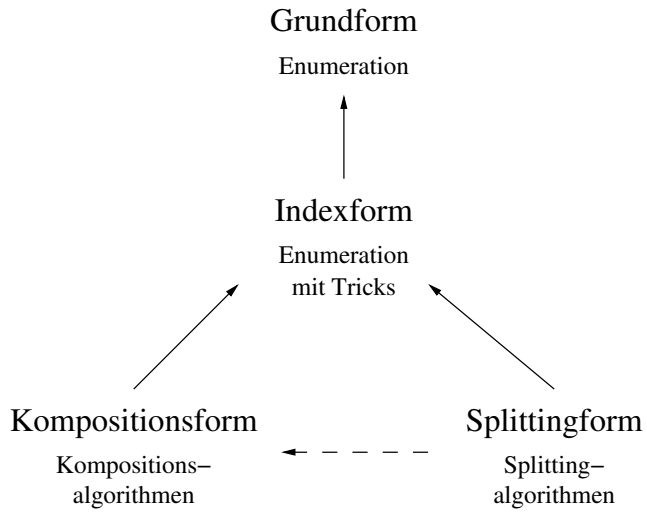


Abbildung 2.4: Formale Beschreibungen und resultierende Algorithmen

Bemerkung 2.21. Normalerweise sind für eine Graphenkenngröße keine formalen Beschreibungen gegeben, die direkt für die Umsetzung in Algorithmen geeignet sind. Es ist aber in der Regel nicht schwer, zumindest eine Grundform zu konstruieren.

Die Werte für c_V und c_E können oft unmittelbar aus einer verbalen Beschreibung der Graphenkenngröße gewonnen werden. Wenn in einer solche Beschreibung nur Kanten erwähnt werden, und diese jeweils einen von zwei möglichen Zuständen annehmen können, einzelne Knoten aber uninteressant sind, wird meist $c_V = 1$ und $c_E = 2$ genügen usw. Mit Blick auf die nachfolgend konstruierten Algorithmen sollte für praktische Anwendungen versucht werden, c_V und c_E (und damit die Mächtigkeit des gewählten Konstellationsraums) so klein wie möglich zu halten, da diese Größe einen direkten Einfluss auf die Komplexität der erzeugten Algorithmen hat.

Die Menge $\mathcal{W}$ umfasst zunächst $\{R(H) : H \in \mathcal{H}\}$, wobei es später durch die Wahl der Operation $\oplus$ aufgrund der notwendigen Abgeschlossenheit von $\mathcal{W}$ unter $\oplus$ dazu kommen

kann, dass auch Elemente in $\mathcal{W}$ aufgenommen werden müssen, die nicht als Funktionswert $R(H)$ für einen geeigneten Hypergraphen H dargestellt werden können. Falls es in dem so konstruierten $\mathcal{W}$ kein neutrales Element gibt, so wird $\mathcal{W}$ um ein Element O erweitert, welches $O \oplus w = w \oplus O = w$ für alle $w \in \mathcal{W}$ erfüllen soll. Falls es dagegen bereits ein neutrales Element gibt, so wird dieses einfach als O bezeichnet.

Die Funktionen val und $\oplus$ erhält man schließlich aus der Überlegung, welchen Einfluss eine Konstellation auf das Endergebnis hat. Bei Optimierungsproblemen verbirgt sich hinter $\oplus$ in der Regel eine Minimums- oder Maximumsbildung, bei Entscheidungsproblemen dagegen oft nur eine Auswahl zwischen zwei potentiellen Lösungskandidaten.

Bemerkung 2.22. Es ist insbesondere bei Optimierungs- und Entscheidungsproblemen häufig zu beobachten, dass es für eine bestimmte Aufgabenstellung eine natürliche, problembezogene Interpretation der Konstellationen gibt. Dabei kann es allerdings vorkommen, dass nicht jede mögliche Konstellation einer im Sinne der Aufgabenstellung zulässigen Lösung zugeordnet werden kann. An dieser Stelle ist es möglich, den folgenden „Trick“ bei der Konstruktion des Monoides anzuwenden: Die Menge $\mathcal{W}$ wird zunächst wie obenstehend konstruiert. Danach wird $\mathcal{W}$ um ein Element $\mathbb{O}$ erweitert und $\mathbb{O} \oplus w = w \oplus \mathbb{O} = w$ definiert. Das neue Element wird also zum neutralen Element – unabhängig davon, ob bereits ein neutrales Element in $\mathcal{W}$ existiert hat (das nun nicht mehr neutral ist) oder nicht. Formal werden nun alle Konstellationen aus dem entsprechenden Konstellationsraum zugelassen, allerdings wird bei der Definition der Funktion val dafür gesorgt, dass für die nichtzulässigen Konstellationen $\mathrm{val}(C) = \mathbb{O}$ gilt. Damit kann die Summation in (2.6) immer noch über den kompletten Konstellationsraum laufen, die nichtzulässigen Konstellationen liefern allerdings keinen Beitrag zum Endergebnis. Dieser „Trick“ wird im Folgenden auch als *Abschlussmethode* bezeichnet.

2.5.4 Zustände

Definition 2.23. Sei R eine Graphenkenngröße mit der Grundform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val})$ und ind eine Indizierung von $\mathcal{C}(c_{\mathrm{V}}, c_{\mathrm{E}})$ mit dem Indexbereich $\mathcal{I}$. Dann wird das Tupel $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ *formale Beschreibung in Indexform* (bzw. kurz *Indexform*) von R genannt.

Bemerkung 2.24. Die formale Beschreibung in Grundform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val})$ einer Graphenkenngröße R kann auf viele verschiedene Weisen zu einer Indexform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ ergänzt werden. Eine zulässige Wahl stellt dabei immer die Wahl der *trivialen Indexform* $\mathrm{ind} : C \mapsto C$ dar.

Sei R eine Graphenkenngröße mit der Indexform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$, $H = (V, E) \in \mathcal{H}$ ein Hypergraph und $A \subseteq V$ eine fixierte Teilmenge seiner Knotenmenge.

Definition 2.25. Die *Bewertung* $\mathrm{ev}_{H,A}(i)$ eines Index $i \in \mathcal{I}_{H,A}$ wird definiert durch

$$\mathrm{ev}_{H,A}(i) \quad = \bigoplus_{C \in \mathrm{ind}_{H,A}^{-1}(i)} \mathrm{val}(C). \tag{2.7}$$

Die Menge

$$\mathcal{Z}_R \quad = \quad \bigcup_{H=(V,E)\in\mathcal{H}} \bigcup_{\substack{i\in\mathcal{I}_{H,A} \\ A\subseteq V}} \{(i, \mathrm{ev}_{H,A}(i))\}$$

heißt *Zustandsraum* der Indexform $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$.

Mit Hilfe einer *Projektion* genannten Funktion

$$\mathrm{prs} \quad : \quad (i, w) \mapsto w$$

kann Gleichung (2.6) für beliebige Hypergraphen $H = (V, E) \in \mathcal{H}$ und Zielmengen $A \subseteq V$ wie folgt umgeschrieben werden:

$$\begin{aligned} R(H) &\overset{(2.6)}{=} \bigoplus_{C\in\mathcal{C}_H(c_\mathrm{V},c_\mathrm{E})} \mathrm{val}(C) \\ &\overset{(2.5)}{=} \bigoplus_{i\in\mathcal{I}_{H,A}} \bigoplus_{C\in\mathrm{ind}^{-1}_{H,A}(i)} \mathrm{val}(C) \\ &= \bigoplus_{i\in\mathcal{I}_{H,A}} \mathrm{prs}(i, \bigoplus_{C\in\mathrm{ind}^{-1}_{H,A}(i)} \mathrm{val}(C)) \\ &\overset{(2.7)}{=} \bigoplus_{i\in\mathcal{I}_{H,A}} \mathrm{prs}\,(i, \mathrm{ev}_{H,A}(i)) \end{aligned} \tag{2.8}$$

Die dabei auftretenden Paare (i, w), die aus jeweils einem Index $i \in \mathcal{I}_{H,A}$ und seiner Bewertung $w = \mathrm{ev}_{H,A}(i) \in \mathcal{W}$ bestehen, heißen *Zustände.* Die Menge $\mathcal{Z}_{H,A} = \mathcal{Z}_{H,A}(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ der Zustände, die aus der Zerlegung des $(c_\mathrm{V}, c_\mathrm{E})$-Konstellationsraumes eines Hypergraphen H gemäß einer Indizierung ind resultieren, wird die *von* ind *induzierte Zustandsmenge von H mit der Zielmenge A* genannt.

Für eine Multimenge $Z = \{(i_1, w_1), \ldots, (i_s, w_s)\}$ von Paaren, die jeweils aus einem Index und einem Wert bestehen, wird die Notation $Z = (i_1, w_1) \uplus \ldots \uplus (i_s, w_s)$ verwendet. Aufgrund der Äquivalenz der entsprechenden Multimengen gilt:

$$\begin{aligned} (i_1, w_1) \uplus (i_2, w_2) &= (i_2, w_2) \uplus (i_1, w_1) \\ ((i_1, w_1) \uplus (i_2, w_2)) \uplus (i_3, w_3) &= (i_1, w_1) \uplus ((i_2, w_2) \uplus (i_3, w_3)) \end{aligned}$$

Wir vereinbaren darüber hinaus:

$$(i, w_1) \uplus (i, w_2) \quad = \quad (i, w_1 \oplus w_2)$$

Zwei derartige Multimengen sollen als gleich gelten, wenn sie nur durch Anwendung dieser Rechenregeln ineinander überführt werden können. Insbesondere gelten diese Rechenregeln auch für Zustandsmultimengen.

Eine Zustandsmultimenge heißt *normal*, wenn sie keine Zustände mit gleichem Index und keine Zustände mit Wert O enthält. Durch Zusammenfassen von Zuständen mit gleichem Index bzw. Entfernen von Zuständen mit dem Wert O, die keinen Einfluss auf den Wert einer

Zustandsmultimenge haben, kann jede beliebige Zustandsmultimenge *normalisiert* werden. Zustandsmultimengen sollen im Weiteren immer als normalisiert angenommen werden. Da nach der Normalisierung keine gleichen Elemente mehr enthalten sein können, wird im Folgenden anstelle von Zustandsmultimengen einfach nur noch von *Zustandsmengen* gesprochen.

Als *Wert* einer Zustandsmenge $Z \subseteq \mathcal{Z}_{H,A}$ wird der Wert

$$\mathrm{pr}(Z) \quad = \quad \bigoplus_{(i,w)\in Z} \mathrm{prs}(i,w) \tag{2.9}$$

bezeichnet.

Die Gleichung (2.8) erhält mit (2.9) die Form:

$$\begin{aligned} R(H) &= \bigoplus_{i\in\mathcal{I}_{H,A}} \mathrm{prs}\left(i, \mathrm{ev}_{H,A}(i)\right) \\ &= \mathrm{pr}\left(\biguplus_{i\in\mathcal{I}_{H,A}} (i, \mathrm{ev}_{H,A}(i))\right) \\ &= \mathrm{pr}\left(\biguplus_{z\in\mathcal{Z}_{H,A}} z\right) \end{aligned} \tag{2.10}$$

Diese Identität vom Wert einer Graphenkenngröße und dem Wert bestimmter Zustandsmengen stellt die Basis für die nachfolgend behandelten Algorithmen dar.

Bemerkung 2.26. Die Mächtigkeit der Zustandsmenge nach einer Indizierung ist ein wesentlicher Faktor sowohl für den Rechen- als auch für den Speicheraufwand bei der Ausführung von Splitting- und Kompositionsalgorithmen. Deshalb ist die in Bemerkung 2.24 genannte *triviale Indizierung* aus praktischer Sicht nicht für die Umsetzung in Algorithmen geeignet, da sie zu einer Zustandsmenge führt, in der jeder Zustand genau eine Konstellation repräsentiert – und damit zur schlechtesten aller möglichen Lösungen. „Gute“ Indizierungen zeichnen sich hingegen dadurch aus, dass die Zahl der erzeugten Zustände möglichst gering ist, sie aber noch zu einer der im Folgenden behandelten *Splitting-* oder *Kompositionsformen* erweiterbar ist.

Bemerkung 2.27. Die Bezeichnungen „val“, „ev“ und „pr“ stammen von den englischen Worten „value“, „evaluation“ und „projection“. Der Suffix „s“ (von „single“) in „prs“ soll kenntlich machen, dass es sich bei dem Argument um einen einzelnen Zustand handelt, wogegen sich die Bezeichnung „pr“ immer auf eine Menge solcher Objekte bezieht.

3 Bekannte Verfahren

Für die Berechnung von Graphenkenngrößen sind verschiedene Algorithmen entwickelt worden, die mehr oder weniger stark spezielle Eigenschaften dieser Kenngrößen oder der zu berechnenden Graphen bzw. Hypergraphen benutzen. Einige wenige Methoden sind aber auch „universell“ einsetzbar, in dem Sinne, dass unter Umständen „ähnliche“ Aufgabenstellungen durch „ähnliche“ Algorithmen gelöst werden können. Vier dieser Methoden, die auch Einfluss auf die Entwicklung der Kompositionsmethode hatten, sollen in diesem Kapitel überblicksmäßig beschrieben werden. Diese Methoden werden häufig nur für den Fall gewöhnlicher Graphen verwendet, allerdings ist die Verallgemeinerung auf Hypergraphen problemlos möglich. Im Interesse einer besseren Vergleichbarkeit mit der im folgenden Kapitel beschriebenen, auf Hypergraphen bezogenen Kompositionsmethode wird deshalb versucht, die angesprochenen Methoden ebenfalls für Hypergraphen darzustellen.

Die vorliegende Auswahl von vier Methoden ist keineswegs als vollständige Auflistung zu betrachten, sondern soll nur bei der Einordnung der Kompositionsmethode in ihr „natürliches Umfeld“ helfen. Dabei stellt sich heraus, dass die auf den ersten Blick sehr verschiedenen Verfahren eine wesentliche Gemeinsamkeit aufweisen, die hier etwas in den Vordergrund gerückt werden soll. Diese Gemeinsamkeit besteht in der expliziten oder impliziten Verwendung einer oder mehrere Indizierungen eines für die Beschreibung der Graphenkenngröße geeigneten Konstellationsraumes und damit letztendlich in der Berechnung der Kenngröße nach Formel (2.10).

Es soll in der Folge davon ausgegangen werden, dass für einen gegebenen Hypergraphen $H = (V, E) \in \mathcal{H}$ der Wert $R(H)$ einer Kenngröße R berechnet werden soll. Die Kenngröße soll bereits in Form einer formalen Beschreibung in Grundform als $R = (c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val})$ gegeben sein. Die einzelnen Methoden unterscheiden sich in der Art und Weise, wie diese Grundform zu einer Indexform erweitert wird. Allen gemeinsam ist das Ziel, einerseits die Zahl $|\mathcal{I}_{H,A}|$ der durch die Indizierung entstehenden Zustände klein zu halten, und andererseits die Werte $\mathrm{ev}_{H,A}(i)$ der Zustände direkt aus dem jeweiligen Index $i \in \mathcal{I}_{H,A}$ zu bestimmen, ohne den Weg über die individuellen Konstellationen aus $\mathrm{ind}_{H,A}^{-1}(i)$ gehen zu müssen. Besonders für das zweite Ziel ist eine kompaktere Darstellung des Index hilfreicher als es die explizite Auflistung der Elemente der Menge $\mathrm{ind}_{H,A}^{-1}(i)$ ist. Ein weiteres Unterscheidungsmerkmal besteht darin, ob während des Ablaufs der erzeugten Algorithmen Grund- und Zielmenge der verwendeten Indizierung gleich bleiben oder ob verschiedene Grund- und Zielmengen verwendet werden.

Als Beispiel wird in diesem Kapitel die Berechnung der Zusammenhangswahrscheinlichkeit eines (gewöhnlichen) stochastischen Graphen dienen. Bei diesem Problem geht es darum, die Wahrscheinlichkeit dafür zu bestimmen, dass ein gegebener stochastischer Graph G nach einem Ausfallereignis noch zusammenhängend ist. Dabei wird vorausgesetzt, dass die Kanten von G unabhängig voneinander mit vorgegebenen, möglicherweise aber individuell verschiedenen Wahrscheinlichkeiten p_e funktionieren. Eine ausführlichere Beschreibung des Problems findet sich im Rahmen der Anwendungen in Abschnitt 5.6.1. An dieser Stelle soll

lediglich angemerkt werden, dass eine formale Beschreibung des Problems in Grundform durch das Tupel $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val})$ mit $c_\mathrm{V} = 1$, $c_\mathrm{E} = 2$, $\mathcal{W} = \mathbb{R}$, $\oplus = +$ und $\mathrm{val} : C \mapsto \prod_{e\in\hat{C}} p_e \cdot \prod_{e\in E-\hat{C}}(1-p_e) \cdot [(V,\hat{C}) \text{ ist zh.}]$ gegeben ist. Die dabei auftretenden $(1,2)$-Konstellationen C von Graphen $G = (V, E)$ werden, wie bereits in Bemerkung 2.14 angesprochen, als Kantenmenge $\hat{C} = \{e \in E : C(e) = 1\} \subseteq E$ geschrieben. Kanten $e \in \hat{C}$ heißen dabei *funktionierend* in der Konstellation C, die Elemente von $E - \hat{C}$ dagegen *ausgefallen*.

3.1 Enumeration

Ein einfacher Ansatz zur Berechnung einer Graphenkenngröße R mit der formalen Beschreibung in Grundform $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val})$ für einen Hypergraphen $H = (V, E)$ besteht darin, diese direkt gemäß (2.6) durchzuführen, also der Reihe nach alle Konstellationen zu erzeugen, zu bewerten und die Resultate dieser Bewertungen zusammenzufassen. Dies entspricht der Wahl der trivialen Indizierung $\mathrm{ind} : (C, A) \mapsto C$ mit der Zielmenge $A = V$. Der Konstellationsraum $\mathcal{C}_H$ wird von der trivialen Indizierung in $|\mathcal{I}_{H,A}|$ Konstellationsklassen mit jeweils einem Element zerlegt. Der Wert w eines Zustands $z = (i, w)$ entspricht dem Wert $\mathrm{val}(C)$ der einzigen Konstellation $C \in \mathrm{ind}^{-1}_{H,A}(i)$ in der durch den Index i repräsentierten Konstellationsklasse. Damit kann die Berechnung direkt nach folgendem simplen Algorithmus erfolgen:

```
Algorithmus Enumeration(Hypergraph H)
  w := O
  Für alle C in C_H:
    w := w ⊕ val(C)
  Rückgabe w
```

Die Berechnung ist dabei derart möglich, dass immer nur ein Wert (die Zwischensumme w) und eine Konstellation gleichzeitig im Speicher gehalten werden müssen, so dass dieses Verfahren sehr sparsam bezüglich des Speicherverbrauchs ist (in der Regel $\mathcal{O}(1)$, gemessen in Einheiten von $\mathcal{W}$ bzw. $\mathcal{C}$).

Dieser geringe Speicherbedarf und die Einfachheit des Algorithmus sind die wesentlichen Vorteile der Methode. Entscheidender Nachteil ist der Rechenaufwand: Der $(c_\mathrm{V}, c_\mathrm{E})$-Konstellationsraum von $H = (V, E)$ hat $c_\mathrm{V}^{|V|} \cdot c_\mathrm{E}^{|E|}$ Elemente, die alle erzeugt und untersucht werden müssen, so dass der Berechnungsaufwand im Regelfall (das heißt $c_\mathrm{V} > 1$ oder $c_\mathrm{E} > 1$) exponentiell mit der Knoten- bzw. Kantenzahl des Graphen wächst.

Für einige Problemstellungen und sehr kleine Graphen kann dieser Nachteil aufgrund der sehr einfachen Implementation jedoch in Kauf genommen werden, insbesondere wenn c_V und c_E sehr klein sind und die Bewertung einer Konstellation einfach ist.

Bemerkung 3.1. Alle folgenden Ansätze können als Verfeinerung dieser Methode verstanden werden. Verbesserungen entstehen dabei durch die Wahl der Indizierung durch Zusammenfassen von möglichst großen Gruppen von Konstellationen zu Konstellationsklassen. Können Konstellationsklassen evaluiert werden, ohne auf die einzelnen Konstellationen zurückgreifen zu müssen, besteht die Hoffnung, dass der durchschnittliche Aufwand pro Konstellation und

damit der Gesamtaufwand sinkt. Die verfeinerten Methoden unterscheiden sich dabei vor allem in der Indizierung, das heißt in der Art der Bildung dieser Gruppen und danach, ob sich die Zielmengen der verwendeten Indizierung (und damit die Konstellationsgruppen) im Verlaufe des Algorithmus ändern oder nicht.

3.2 Dekomposition

Wenn eine formale Beschreibung $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val})$ einer Graphenkenngröße R gegeben ist, für die $c_\mathrm{E} > 1$ und $c_\mathrm{V} = 1$ gilt, so lassen sich die Konstellationen für einen zu berechnenden Hypergraphen rekursiv durch folgenden *Kantendekompositionsalgorithmus* erzeugen und bewerten:

```
Funktion Rekursion
    (Hypergraph H = (V,E), Kantenmenge F, Teilkonstellation C)
  Falls F = E:
    Rückgabe val(C)
  Sonst:
    Wähle Pivotelement e ∈ E − F
    w := O
    Für alle i von 1 bis c_E:
      w := w ⊕ Rekursion(H, F + e, C ∪ {(e,i)})
    Rückgabe w

Algorithmus Kantendekomposition(Hypergraph H)
  Rückgabe Rekursion(H, ∅, ∅)
```

Ein ähnlicher Algorithmus lässt sich für den Fall $c_\mathrm{V} > 1$ und $c_\mathrm{E} = 1$ finden, dieser wird *Knotendekompositionsalgorithmus* genannt. Durch Kombination beider Algorithmen sind schließlich die verbleibenden Fälle, in denen $c_\mathrm{V} > 1$ und $c_\mathrm{E} > 1$ gilt, berechenbar.

Ein Dekompositionsalgorithmus ist zunächst also nichts anderes als ein Enumerationsverfahren, bei dem die Reihenfolge der Bewertung der Konstellationen in einem bestimmten Rahmen festgelegt ist, nämlich durch eine Tiefensuche in einer Baumstruktur, dem so genannten *Dekompositionsbaum*. Die Blätter des Baumes entsprechen dabei den Konstellationen. Wird der Dekompositionsbaum vollständig abgearbeitet, kann also die Dekomposition nicht besser sein als die Enumeration. Das ändert sich allerdings, wenn man kenngrößenspezifische Eigenheiten ausnutzt, um den Dekompositionsbaum zu beschneiden.

Beispiel 3.2. Für den Fall der Berechnung der Zusammenhangswahrscheinlichkeit eines stochastischen Graphen $G = (V, E)$ gilt Folgendes: Wenn im Laufe der Dekomposition ein Aufruf der Rekursion mit einer Teilkonstellation C erfolgt, für die $(V, E - \{e : C(e) = 2\})$ bereits nicht zusammenhängend ist, wird das zwangsläufig auch für alle Erweiterungen von C gelten. Alle diese Erweiterungen würden mit 0 bewertet werden. Damit kann die Rekursion an dieser Stelle bereits abgebrochen werden und als (Teil-)Ergebnis 0 zurückgegeben werden. Damit passt die Dekompositionsmethode in das in Bemerkung 3.1 angesprochene Schema „Enumeration mit geschicktem Zusammenfassen von Konstellationen zu Konstellationsklassen“: Die

Konstellationsklassen bestehen aus denjenigen Konstellationen, die sich im Dekompositionsbaum unterhalb einer Abbruchstelle befinden. Kann kein vorzeitiger Abbruch erreicht werden, besteht eine Bijektion zwischen Konstellationsklassen und Konstellationen, andernfalls ist die Zahl der Konstellationsklassen echt kleiner als die der Konstellationen.

Es besteht bei der Dekomposition keine freie Wahl der Reihenfolge der Berechnung der einzelnen Konstellationen mehr, allerdings gibt es immer noch die Freiheit der Wahl der Reihenfolge der Pivotelemente. Diese kann genutzt werden, um möglichst frühzeitig Situationen zu erzeugen, in denen die Rekursion wegen eines vorhersehbaren Ergebnisses abgebrochen werden kann, um so möglichst große Teile des vollen Dekompositionsbaumes nicht bearbeiten zu müssen. Für unser Beispiel bietet es sich an, bei der Pivotwahl eine zu einem minimalen Kantenschnitt gehörende Kante zu wählen. Damit tritt früh der Fall auf, dass der dem Löschen einer Kante e (also dem Fall $C(e) = 2$) entsprechende Ast im Dekompositionsbaum zu einem nichtzusammenhängenden Graphen gehört. Dessen Zusammenhangswahrscheinlichkeit 0 ist bekannt, ohne noch weitere Fälle unterscheiden zu müssen.

Alternativ können als Pivotelemente immer Kanten gewählt werden, deren Entfernung nicht unmittelbar zum Zerfall des Graphen führen, solange dies möglich ist. Unter diesen Umständen ist die Zahl der tatsächlich abzuarbeitenden Blätter im Dekompositionsbaum gleich der Zahl der Gerüste des Ausgangsgraphen. Diese Zahl wächst zwar im Allgemeinen immer noch exponentiell mit der Größe des Graphen, allerdings in einem geringeren Maße als die Zahl der Konstellationen. Vgl. dazu auch [Jon82].

Bemerkung 3.3. Üblicherweise wird die Dekompositionsmethode als direkte Umsetzung so genannter *Dekompositionsformeln* beschrieben. Diese Formeln setzen die zu berechnende Kenngröße in einem bestimmten Graphen G in Beziehung zu der gleichen oder einer anderen Graphenkenngröße in bestimmten Minoren $G_1, \ldots, G_r$:

$$\begin{aligned} R(G) &= f(R_1(G_1), \ldots, R_r(G_r)) \quad \forall G \in \mathcal{G} \\ R(\overline{K}_n) &= r(n) \end{aligned}$$

Ein Beispiel für eine derartige Beziehung ist die Formel

$$\begin{aligned} R(G) &= p_e\, R(G_e) + (1 - p_e)\, R(G_{-e}) \\ R(\overline{K}_n) &= [n \leq 1] \end{aligned}$$

zur Berechnung der Zusammenhangswahrscheinlichkeit eines stochastischen Netzes mit ausfallfreien Knoten und unabhängig voneinander ausfallenden Kanten. Ein weiteres Beispiel liefert

$$\begin{aligned} u(G) &= x\, u(G_{-v-N(v)}) + u(G_{-v}) \\ u(\overline{K}_n) &= (x+1)^n \end{aligned}$$

zur Bestimmung des Unabhängigkeitspolynoms $u(G) \in \mathbb{Z}[x]$ eines Graphen G.

Diese „übliche“ Form der Dekomposition könnte theoretisch eine nichtlineare Funktion f enthalten und ist damit etwas allgemeiner als die hier dargestellte „Dekomposition über Konstellationen“. Bislang ließ sich aber für alle Graphenkenngrößen, für die Dekompositionsformeln bekannt sind, ohne großen Aufwand eine formale Beschreibung in Grundform und darauf aufbauend einer der hier vorgestellten „linearen“ Dekompositionsalgorithmen finden. Insofern scheint zumindest für praktische Belange letztere Form ausreichend zu sein.

Bemerkung 3.4. Die bekannten Dekompositionsformeln sind entweder „kantenorientiert“ oder „knotenorientiert“. Im ersten Fall werden die Minoren in der Regel durch Kontraktion bzw. Löschen einer Kante, im zweiten durch Löschen eines Knotens bzw. Löschen und Kontraktion der Nachbarschaft eines Knotens gewonnen. Das ist insofern interessant, als dass sich in manchen Fällen daraus direkt eine der im Folgenden behandelten Splitting- bzw. Kompositionsformen ergibt und somit besonders wenig manueller Aufwand beim Einsatz der Splitting- bzw. Kompositionsmethode entsteht. Ein ausführliches Beispiel dafür ist im Abschnitt 5.1 zu finden.

Bemerkung 3.5. Bei der eben erwähnten Dekomposition über Minoren treten häufig Mengen isomorpher Untergraphen auf. Wenn solche Mengen erkannt werden können, so braucht jeweils nur ein Vertreter der Menge weiter zerlegt zu werden. Eine derartiger Algorithmus ist in Abbildung 3.1 dargestellt. Dabei handelt es sich um die Berechnung des in Abschnitt 5.6.2 behandelten Zuverlässigkeitspolynoms für den vollständigen Graphen auf drei Knoten K_3. Die Berechnung verläuft ähnlich wie bei der einfachen Dekomposition in drei Phasen:

1. Aufbau eines *Dekompositionsgraphen* (linker Teil des Schemas). Die Knoten des Dekompositionsgraphen sind Minoren des K_3, die (gerichteten) Kanten geben an, ob die Minoren durch Kontraktion (durchgezogene Pfeile) bzw. Löschen (gestrichelte Pfeile) von Kanten entstehen. Durch das Zusammenfassen isomorpher Minoren ist dieser Dekompositionsgraph im Allgemeinen kein Baum mehr.

2. Bewerten der Konstellationen, die den „Endknoten“ (also den zusammengefassten Blätter des entsprechenden Dekompositionsbaumes) des Dekompositionsgraphen entsprechen (gepunktete Pfeile).

3. *Rückwärtsrechnung* im Dekompositionsgraphen. Dies wird hier dargestellt durch Spiegelung des Dekompositionsgraphen, wobei die Minoren durch die entsprechenden Zwischenergebnisse ersetzt sind. Diese entstehen durch Multiplikation mit p für eine kontrahierte, und mit $1 - p$ für eine gelöschte Kante.

Der gesamte Vorgang bewertet im vorliegenden Beispiel nur drei Endknoten, wohingegen ein Verzicht auf das Zusammenfassen isomorpher Untergraphen zu acht Endknoten führt. Aufwendig ist natürlich die Erkennung der isomorphen Untergraphen, so dass eine naive Umsetzung dieser „verbesserten Dekomposition“ nicht notwendigerweise zu besseren Algorithmen führt. Die grundlegende Idee des Zusammenfassens in einem gewissen Sinne „gleichartiger“ Konstellationen wird jedoch später in der Kompositionsmethode aufgegriffen und dort effizient umgesetzt.

3.3 Splitting

Gegeben seien eine Indexform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ einer Graphenkenngröße R, ein Hypergraph $H = (V, E)$ sowie zwei Unterhypergraphen $H_1 = (V_1, E_1)$ und $H_2 = (V_2, E_2)$ von H und eine Knotenmenge $K \subseteq V$, so dass $H_1 \boxplus_K H_2 = H$ gilt.

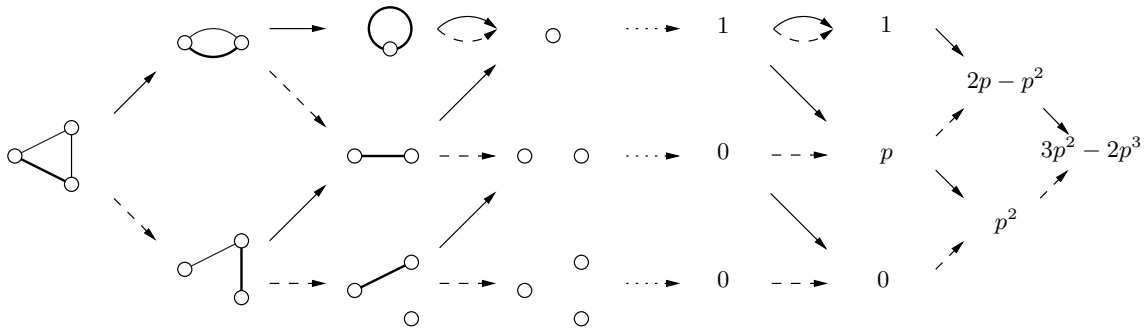


Abbildung 3.1: Berechnung des Zuverlässigkeitspolynoms durch Dekomposition mit Zusammenfassung isomorpher Zwischenstufen. Die Pivot-Kanten sind dicker eingezeichnet. Durchgezogene Pfeile entsprechen einer „funktionierenden“ Kante und damit dem Abstieg in den Kontraktionsast bzw. einer Multiplikation mit p; gestrichelte Pfeile entsprechen dagegen einer „ausgefallenen“ Kante und damit dem Abstieg in den Löschast des Dekompositionsbaums bzw. einer Multiplikation mit $1 - p$.

Die Methode des *Splittings* besteht in der Berechnung von Teilergebnissen in den beiden Unterhypergraphen und der anschließenden Kopplung der Teilergebnisse zum Gesamtergebnis. Für das Beispiel der Berechnung der Zusammenhangswahrscheinlichkeit sind entsprechende Ideen zum Beispiel aus [Ros77] bekannt.

Die Teilergebnisse werden in der vorliegenden Arbeit sowohl beim Splitting als auch in der nachfolgend behandelten Kompositionsmethode als Zustandsmenge dargestellt. Da $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \emptyset$, $V_1 \cup (V_2 - K) = V$, $V_1 \cap (V_2 - K) = \emptyset$ gilt, kann der $(c_\mathrm{V}, c_\mathrm{E})$-Konstellationsraum $\mathcal{C}_H$ als kartesisches Produkt des $(c_\mathrm{V}, c_\mathrm{E})$-Konstellationsraumes von $H_1 = (V_1, E_1)$ und des $(c_\mathrm{V}, c_\mathrm{E})$-Konstellationsraumes von $T_2 = (V_2, E_2, K)$ geschrieben werden. Konstellationen des Hypergraphen zerfallen also prinzipiell in Konstellationen auf H_1 und T_2. Die Kontaktzerlegung von H in H_1 und T_2 ist allerdings unsymmetrisch, was nicht nur unästhetisch ist, sondern in den Fällen, in denen $c_\mathrm{V} > 1$ ist, auch zu komplizierten Argumentationen beim Nachweis der Korrektheit führt. Im Folgenden soll deshalb eine symmetrische Formulierung des Splittings dargestellt werden.

Sei dazu wieder $H_1 = (V_1, E_2)$, $H_2 = (V_2, E_2)$ mit den oben genannten Eigenschaften sowie $H_K = (K, \emptyset)$ definiert. Zusätzlich sollen noch $\mathcal{C}_K = \mathcal{C}_{H_K}$ und für $l = 1, 2$ und $C_K \in \mathcal{C}_K$ die Abkürzungen

$$\begin{aligned}
\mathcal{I}_{H_l, C_K} &= \{\mathrm{ind}(C_l, K) : C_l \in \mathcal{C}_{H_l} \wedge C_l|_K = C_K\} \\
\mathrm{ev}_{H_l, C_K}(i) &= \bigoplus_{\substack{C_l \in \mathcal{C}_{H_l} \\ \mathrm{ind}(C_l, K) = i \\ C_l|_K = C_K}} \mathrm{val}(C_l) \qquad (3.1) \\
\mathcal{Z}_{H_l, C_K} &= \{(i, w) : i \in \mathcal{I}_{H_l, C_K}, w = \mathrm{ev}_{H_l, C_K}(i)\}
\end{aligned}$$

vereinbart sein.

Definition 3.6. Gegeben sei eine Indexform $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ einer Graphenkenngröße R. Bezeichne Γ die Menge

$$\bigcup_{H\in\mathcal{H}} \bigcup_{H_1 \boxplus_K H_2 = H} \{(C_K, (i_1, w_1), (i_2, w_2)) : C_K \in \mathcal{C}_K \wedge i_1 \in \mathcal{I}_{H_1,C_K} \wedge i_2 \in \mathcal{I}_{H_2,C_K} \wedge w_1, w_2 \in \mathcal{W}\}.$$

Existiert eine *Koppelfunktion* $\mathrm{glue} : \Gamma \to \mathcal{W}$, so dass für alle Hypergraphen $H \in \mathcal{H}$ und alle Zerlegungen $H_1 \boxplus_K H_2$ eines solchen Hypergraphen für jedes $C_K \in \mathcal{C}_K$, $i_1 \in \mathcal{I}_{H_1,C_K}$ und $i_2 \in \mathcal{I}_{H_2,C_K}$ gilt

$$\begin{aligned}&\mathrm{glue}(C_K, (i_1, \mathrm{ev}_{H_1,C_K}(i_1)), (i_2, \mathrm{ev}_{H_2,C_K}(i_2)))\\ &\qquad = \bigoplus_{\substack{C_1\in\mathcal{C}_{H_1}\\ \mathrm{ind}(C_1,K)=i_1\\ C_1|_K=C_K}} \bigoplus_{\substack{C_2\in\mathcal{C}_{H_2}\\ \mathrm{ind}(C_2,K)=i_2\\ C_2|_K=C_K}} \mathrm{val}(C_1 \cup C_2),\end{aligned} \tag{3.2}$$

so wird das Tupel $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{glue})$ eine *formale Beschreibung in Splittingform* (kurz *Splittingform*) von R genannt.

Mit einer Koppelfunktion glue gilt für einen beliebigen Hypergraphen $H \in \mathcal{H}$ mit einer Zerlegung $H = H_1 \boxplus_K H_2$:

$$\begin{aligned}&\bigoplus_{C_K\in\mathcal{C}_K} \bigoplus_{i_1\in\mathcal{I}_{H_1,C_K}} \bigoplus_{i_2\in\mathcal{I}_{H_2,C_K}} \mathrm{glue}(C_K, (i_1, \mathrm{ev}_{H_1,C_K}(i_1)), (i_2, \mathrm{ev}_{H_2,C_K}(i_2)))\\
&\overset{(3.2)}{=} \bigoplus_{C_K\in\mathcal{C}_K} \bigoplus_{i_1\in\mathcal{I}_{H_1,C_K}} \bigoplus_{i_2\in\mathcal{I}_{H_2,C_K}} \bigoplus_{\substack{C_1\in\mathcal{C}_{H_1}\\ \mathrm{ind}(C_1,K)=i_1\\ C_1|_K=C_K}} \bigoplus_{\substack{C_2\in\mathcal{C}_{H_2}\\ \mathrm{ind}(C_2,K)=i_2\\ C_2|_K=C_K}} \mathrm{val}(C_1 \cup C_2)\\
&= \bigoplus_{C_K\in\mathcal{C}_K} \bigoplus_{i_1\in\mathcal{I}_{H_1,C_K}} \bigoplus_{\substack{C_1\in\mathcal{C}_{H_1}\\ \mathrm{ind}(C_1,K)=i_1\\ C_1|_K=C_K}} \bigoplus_{i_2\in\mathcal{I}_{H_2,C_K}} \bigoplus_{\substack{C_2\in\mathcal{C}_{H_2}\\ \mathrm{ind}(C_2,K)=i_2\\ C_2|_K=C_K}} \mathrm{val}(C_1 \cup C_2)\\
&\overset{(2.5)}{=} \bigoplus_{C_K\in\mathcal{C}_K} \bigoplus_{\substack{C_1\in\mathcal{C}_{H_1}\\ C_1|_K=C_K}} \bigoplus_{\substack{C_2\in\mathcal{C}_{H_2}\\ C_2|_K=C_K}} \mathrm{val}(C_1 \cup C_2)\\
&= \bigoplus_{C\in\mathcal{C}_H} \mathrm{val}(C)\\
&= R(H).\end{aligned}$$

Die Funktion glue ermöglicht es also, die gewünschte Kenngröße allein aus Kenntnis der Zustandsmengen der beiden Teile zu bestimmen. Das führt unmittelbar zu folgendem *Rahmenalgorithmus der Splittingmethode*:

```
Algorithmus Splitting(Hypergraph H)
  w := O
  Für alle C_K in C_K:
    Bestimme Z_{H_1,C_K} := {(i, ev_{H_1,C_K}(i)) : i ∈ I_{H_1,C_K}}
```

```
Bestimme Z_{H2,C_K} := {(i, ev_{H2,C_K}(i)) : i ∈ I_{H2,C_K}}
Für alle z1 in Z_{H1,C_K}:
  Für alle z2 in Z_{H2,C_K}:
    w := w ⊕ glue(C_K, z1, z2)
Rückgabe w
```

Die Art der Berechnung der Zustandsmengen $\mathcal{Z}_{H_1,C_K}$ und $\mathcal{Z}_{H_2,C_K}$ wird durch diesen Rahmenalgorithmus nicht festgelegt. Entscheidend ist die Möglichkeit, diese „irgendwie“ (also zum Beispiel durch Enumeration, Reduktion, oder aber auch wiederholtes Splitting) zu berechnen und aus den Zustandsmengen direkt zum Ergebnis zu kommen, ohne noch einmal auf die Struktur des zu berechnenden Graphen zurückgreifen zu müssen.

Der Korrektheitsbeweis eines aus einer Splittingform $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{glue})$ konstruierten Splittingalgorithmus kann durch Nachweis der Beziehung (3.2) für jedes $C_K \in \mathcal{C}_K$, $i_1 \in \mathcal{I}_{H_1,C_K}$ und $i_2 \in \mathcal{I}_{H_2,C_K}$ erfolgen. Dieser Nachweis ist (falls es sich bei $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{glue})$ tatsächlich um eine Splittingform handelt) nicht schwer, aber „technisch“. In der Regel besteht jedoch die Möglichkeit, diesen Nachweis zu verkürzen, denn es gilt folgendes Lemma:

Lemma 3.7. *Eine Funktion* glue*, für die für alle $C_K \in \mathcal{C}_K$ und $C_1 \in \mathcal{C}_{H_1}$, $C_2 \in \mathcal{C}_{H_2}$ mit $C_1|_K = C_2|_K = C_K$*

$$\mathrm{glue}(C_K, (\mathrm{ind}(C_1, K), \mathrm{val}(C_1)), (\mathrm{ind}(C_2, K), \mathrm{val}(C_2))) \quad = \quad \mathrm{val}(C_1 \cup C_2) \tag{3.3}$$

und für alle $C_K \in \mathcal{C}_K$, $i_1 \in \mathcal{I}_{H_1,C_K}$, $i_2 \in \mathcal{I}_{H_2,C_K}$ und $w_1, w_2 \in \mathcal{W}$

$$\mathrm{glue}(C_K, (i_1, \bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ \mathrm{ind}(C_1,K)=i_1 \\ C_1|_K = C_K}} \mathrm{val}(C_1)), (i_2, w_2)) \quad = \quad \bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ \mathrm{ind}(C_1,K)=i_1 \\ C_1|_K = C_K}} \mathrm{glue}(C_K, (i_1, \mathrm{val}(C_1)), (i_2, w_2)) \tag{3.4}$$

$$\mathrm{glue}(C_K, (i_1, w_1), (i_2, \bigoplus_{\substack{C_2 \in \mathcal{C}_{H_2} \\ \mathrm{ind}(C_2,K)=i_2 \\ C_2|_K = C_K}} \mathrm{val}(C_2))) \quad = \quad \bigoplus_{\substack{C_2 \in \mathcal{C}_{H_2} \\ \mathrm{ind}(C_2,K)=i_2 \\ C_2|_K = C_K}} \mathrm{glue}(C_K, (i_1, w_1), (i_2, \mathrm{val}(C_2))) \tag{3.5}$$

gilt, erfüllt auch (3.2).

Beweis. Es gilt mit $i_l \in \mathcal{I}_{H_l,C_k}$ für $l = 1, 2$:

$$\begin{aligned}
&\mathrm{glue}(C_K, (i_1, \mathrm{ev}_{H_1,C_K}(i_1)), (i_2, \mathrm{ev}_{H_2,C_K}(i_2))) \\
&\overset{(3.1)}{=} \mathrm{glue}(C_K, (i_1, \bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ \mathrm{ind}(C_1,K)=i_1 \\ C_1|_K = C_K}} \mathrm{val}(C_1)), (i_2, \bigoplus_{\substack{C_2 \in \mathcal{C}_{H_2} \\ \mathrm{ind}(C_2,K)=i_2 \\ C_2|_K = C_K}} \mathrm{val}(C_2)) \\
&\overset{(3.4),(3.5)}{=} \bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ \mathrm{ind}(C_1,K)=i_1 \\ C_1|_K = C_K}} \bigoplus_{\substack{C_2 \in \mathcal{C}_{H_2} \\ \mathrm{ind}(C_2,K)=i_2 \\ C_2|_K = C_K}} \mathrm{glue}(C_K, (\mathrm{ind}(C_1, K), \mathrm{val}(C_1)), (\mathrm{ind}(C_2, K), \mathrm{val}(C_2))
\end{aligned}$$

$$\overset{(3.3)}{=} \bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ \mathrm{ind}(C_1,K)=i_1 \\ C_1|_K = C_K}} \bigoplus_{\substack{C_2 \in \mathcal{C}_{H_2} \\ \mathrm{ind}(C_2,K)=i_2 \\ C_2|_K = C_K}} \mathrm{val}(C_1 \cup C_2)$$

und damit (3.2). □

Zum Nachweis der Korrektheit von glue genügt es also (3.3), (3.4) und (3.5) zu zeigen.

Bemerkung 3.8. Nicht zu jeder Indexform einer Graphenkenngröße kann eine geeignete Funktion glue gefunden werden, um eine Splittingform zu vervollständigen. Im Gegenteil, je mehr Konstellationen durch die Funktion ind zu Konstellationsklassen zusammengefasst werden, desto schwieriger gestaltet sich die Suche nach der fehlenden Funktion.

Für die triviale Indexform mit $\mathrm{ind} : (C, K) \mapsto C$ kann dagegen für alle Graphenkenngrößen eine Splittingform gewonnen werden, indem man

$$\mathrm{glue} \quad : \quad (C_K, (i_1, w_1), (i_2, w_2)) \mapsto \mathrm{val}(i_1 \cup i_2) \tag{3.6}$$

wählt.

Für ein beliebiges $C_K \in \mathcal{C}_K$ und $C_1 \in \mathcal{C}_{H_1}$, $C_2 \in \mathcal{C}_{H_2}$ mit $C_1|_K = C_2|_K = C_K$ gilt nämlich

$$\begin{aligned} \mathrm{glue}(C_K, (\mathrm{ind}(C_1, K), \mathrm{val}(C_1)), (\mathrm{ind}(C_2, K), \mathrm{val}(C_2))) &\overset{(3.6)}{=} \mathrm{val}(\mathrm{ind}(C_1, K) \cup \mathrm{ind}(C_2, K)) \\ &= \mathrm{val}(C_1 \cup C_2) \end{aligned}$$

und damit (3.3). Außerdem gilt für alle $C_K \in \mathcal{C}_K$, $i_1 \in \mathcal{I}_{H_1,C_K}$, $i_2 \in \mathcal{I}_{H_2,\mathcal{C}_K}$, $w_2 \in \mathcal{W}$

$$\begin{aligned} \mathrm{glue}(C_K, (i_1, \bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ \mathrm{ind}(C_1,K)=i_1 \\ C_1|_K = C_K}} \mathrm{val}(C_1)), (i_2, w_2)) &= \mathrm{glue}(C_K, (i_1, \mathrm{val}(i_1)), (i_2, w_2)) \\ &= \bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ \mathrm{ind}(C_1,K)=i_1 \\ C_1|_K = C_K}} \mathrm{glue}(C_K, (i_1, \mathrm{val}(C_1)), (i_2, w_2)) \end{aligned}$$

und somit (3.4). Völlig analog folgt (3.5). Nach Lemma 3.7 folgt die Behauptung.

Diese so gewonnene *triviale Splittingform* hat aber ähnlich wie die triviale Indexform keine praktische Relevanz, da dabei die bereits berechneten Werte w_1 und w_2 ignoriert werden und der Wert des Ergebniszustands wieder mühevoll aus allen durch die Indizes repräsentierten Einzelkonstellationen berechnet wird. Ein solchermaßen implementierter Algorithmus ist also nicht besser als die Enumeration, im Gegenteil, es bestehen zusätzlich sogar noch erhebliche Speicheranforderungen für die zwischenzeitlich entstehenden Zustandsmengen $\mathcal{Z}_{H_1,C_K}$ bzw. $\mathcal{Z}_{H_2,C_K}$.

Bemerkung 3.9. Bei der Kopplung muss im allgemeinen Fall eine Dreifachsumme mit insgesamt $\sum_{C_K \in \mathcal{C}_K} |\mathcal{I}_{H_1,C_K}| \, |\mathcal{I}_{H_2,C_K}|$ Summanden berechnet werden. Für den Fall $c_\mathrm{V} = 1$ besteht jedoch $\mathcal{C}_K$ nur aus einem einzigen Element, so dass die äußere Summe im Algorithmus praktisch entfällt. Durch Ausnutzen weiterer problemspezifischer Zusatzinformation ist es in der

Regel möglich, diesen Aufwand weiter zu reduzieren. Das gilt auch für den Fall $c_{\mathrm{V}} > 1$. Besonders häufig kommt es zu derartigen Vereinfachungen, wenn die in Bemerkung 2.22 beschriebene Abschlussmethode, also der „Trick“ der mit $\mathbb{O}$ bewerteten nichtzulässigen Konstellationen, angewandt wird.

Das Problem bei der Suche nach einer (nichttrivialen) Splittingform für eine gegebene Graphenkenngröße $R = (c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val})$ besteht im Wesentlichen in der Bestimmung einer „günstigen“ Indizierung ind, die diese Grundform zu einer Indexform erweitert. Ziel der Suche ist dabei einerseits sicherzustellen, dass eine Funktion glue gefunden werden kann, die die Indexform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ zu einer Splittingform erweitert, und andererseits dabei die maximal zu erwartende Zahl von zu bearbeitenden Zustandskombinationen zu minimieren. Dieses Ziel kann zwar formal für eine fixierte Zerlegung eines Graphen als Optimierungaufgabe

$$\min_{\{(\mathrm{ind},\mathrm{glue}):\exists \mathrm{Splittingformel}\ (c_{\mathrm{V}},c_{\mathrm{E}},\mathcal{W},\oplus,\mathrm{val},\mathrm{ind},\mathrm{glue})\}} \sum_{C_K \in \mathcal{C}_K} |\mathcal{I}_{H_1,C_K}|\,|\mathcal{I}_{H_2,C_K}|$$

dargestellt werden, allerdings ist unklar, wie diese Aufgabe effizient gelöst bzw. wie eine evtl. gefundene Lösung auf beliebige Zerlegungen erweitert werden kann. Obwohl der Suchraum endlich ist, scheidet eine vollständige Durchmusterung aufgrund seiner Größe für alle praktischen Belange jedenfalls aus.

Da jede Splittingform prinzipiell in einen Splittingalgorithmus umsetzbar ist, wird man sich in der Praxis mit Heuristiken zur Einschränkung des Suchraumes zufrieden geben. Hilfreich ist dabei, dass in den bislang bearbeiteten Fällen für ähnliche Problemstellungen ähnlich strukturierte Indizierungen gefunden werden konnten. Diese Erkenntnis kann mit anderen Erfahrungen kombiniert werden, zum Beispiel dass induzierte Indizes oft als Teilmenge oder Mengenpartition der trennenden Knotenmenge oder als Kombination solcher Strukturen beschrieben werden können, für die sich meist direkt eine Interpretation in Bezug auf die zu berechnende Kenngröße finden lässt.

Bemerkung 3.10. Bei der Anwendung der Splittingmethode setzt sich der Gesamtaufwand zur Lösung eines Problems aus dem Aufwand zur Lösung der Teilprobleme und dem eigentlichen Kopplungsaufwand zusammen, wobei letzterer in der Regel exponentiell von der Mächtigkeit der Knotenmenge K abhängt. In sehr dichten Hypergraphen kann diese die Gesamtknotenzahl des Hypergraphen erreichen, so dass kein Gewinn gegenüber der Dekomposition zu verzeichnen ist. Handelt es sich um weniger dichte Strukturen, so sind dagegen Verbesserungen zu erwarten.

3.4 Reduktionen

Ein weiterer allgemeiner Ansatz zur Berechnung von Graphenkenngrößen ist die *Reduktionsmethode,* deren Grundidee es ist, die Berechnung einer Kenngröße in einem Hypergraphen H durch eine Folge von *Reduktionen* auf die eines anderen Hypergraphen zurückzuführen, für den das Ergebnis offensichtlich ist, oder aber mit einer anderen Methode berechnet werden kann.

Definition 3.11. Sei R eine Graphenkenngröße mit der Grundform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val})$ und seien $H_1 = (V_1, E_1)$ und $H_1' = (V_1', E_1')$ zwei Hypergraphen mit $V_1 \cap V_1' = K$ und $E_1 \cap E_1' = \emptyset$.

Bezeichne $\mathcal{C}_{H_1}$ und $\mathcal{C}_{H_1'}$ die zugehörigen $(c_\mathrm{V}, c_\mathrm{E})$-Konstellationsräume und $\mathcal{H}_2$ die Menge all derjenigen Hypergraphen $H_2 = (V_2, E_2)$, für die gilt $V_2 \cap V_1 = V_2 \cap V_1^{'} = K$ und $E_2 \cap E_1 = E_2 \cap E_1^{'} = \emptyset$. Wenn für alle Hypergraphen $H_2 \in \mathcal{H}_2$, alle Konstellationen $C_K \in \mathcal{C}_K = \mathcal{C}_{(K,\emptyset)}$ und alle $C_2 \in \mathcal{C}_{H_2}(c_\mathrm{V}, c_\mathrm{E})$ mit $C_2|_K = C_K$ gilt

$$\bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ C_1|_K = C_K}} \mathrm{val}(C_1 \cup C_2) \quad = \quad \bigoplus_{\substack{C_1' \in \mathcal{C}_{H_1'} \\ C_1'|_K = C_K}} \mathrm{val}(C_1' \cup C_2), \tag{3.7}$$

so bezeichnet man das Paar $(H_1, H_1^{'})$ als *einfache Reduktion* für die Graphenkenngröße R.

Lemma 3.12. *Mit den Bezeichnungen aus der Definition 3.11 gilt für beliebige Hypergraphen $H_2 \in \mathcal{H}_2$ und $H = H_1 \boxplus_K H_2$ und $H^{'} = H_1^{'} \boxplus_K H_2$:*

$$R(H) \quad = \quad R(H^{'})$$

Beweis. Es gilt:

$$\begin{aligned}
R(H) &\overset{(2.6)}{=} \bigoplus_{C \in \mathcal{C}_H} \mathrm{val}(C) \\
&= \bigoplus_{C_K \in \mathcal{C}_K} \bigoplus_{\substack{C_2 \in \mathcal{C}_{H_2} \\ C_2|_K = C_K}} \bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ C_1|_K = C_K}} \mathrm{val}(C_1 \cup C_2) \\
&\overset{(3.7)}{=} \bigoplus_{C_K \in \mathcal{C}_K} \bigoplus_{\substack{C_2 \in \mathcal{C}_{H_2} \\ C_2|_K = C_K}} \bigoplus_{\substack{C_1' \in \mathcal{C}_{H_1'} \\ C_1'|_K = C_K}} \mathrm{val}(C_1' \cup C_2) \\
&= \bigoplus_{C' \in \mathcal{C}_{H'}} \mathrm{val}(C') \\
&\overset{(2.6)}{=} R(H^{'})
\end{aligned}$$

□

Bemerkung 3.13. Die Beziehung (3.7) ist sehr restriktiv, so dass selbst für sehr spezielle Probleme selten derartige Paare $(H_1, H_1^{'})$ existieren. Deshalb wird versucht, mehr „Freiheitsgrade“ einzubauen. Handelt es sich bei der Wertemenge $\mathcal{W}$ zum Beispiel um einen Ring, kann in (3.7) ein zusätzlicher Faktor $f \in \mathcal{W}$ verwendet werden, indem nur noch

$$\bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ C_1|_K = C_K}} \mathrm{val}(C_1 \cup C_2) \quad = \quad f \bigoplus_{\substack{C_1' \in \mathcal{C}_{H_1'} \\ C_1'|_K = C_K}} \mathrm{val}(C_1' \cup C_2) \tag{3.8}$$

für alle Konstellationen $C_K \in \mathcal{C}_K$ und für alle $C_2 \in \mathcal{C}_{H_2}(c_\mathrm{V}, c_\mathrm{E})$ mit $C_2|_K = C_K$ gefordert wird – mit der Konsequenz, dass dann $R(H) = f\, R(H')$ gilt.

Definition 3.14. Wird in den Voraussetzungen der Definition 3.11 die Bedingung (3.7) für eine Funktion $r : \mathcal{W} \to \mathcal{W}$ durch

$$\bigoplus_{\substack{C_1 \in \mathcal{C}_{H_1} \\ C_1|_K = C_K}} \mathrm{val}(C_1 \cup C_2) \quad = \quad r \left(\bigoplus_{\substack{C_1' \in \mathcal{C}_{H_1'} \\ C_1'|_K = C_K}} \mathrm{val}(C_1' \cup C_2) \right) \tag{3.9}$$

ersetzt, so bezeichnet man das Tripel (H_1, H_1', r) als *Reduktion*. Dabei ist der Hypergraph $H_1 = (V_1, E_1)$ die *zu ersetzende Struktur* und $H_1' = (V_1', E_1')$ die *Ersatzstruktur*.

Eine Folge von Hypergraphen $(H_0 = H, \ldots, H_s)$ heißt *Reduktionskette* bezüglich einer Menge von Reduktionen $\mathcal{S} = \{S_1, \ldots, S_{|S|}\}$, wenn es Reduktionen $S_{p_i} \in \mathcal{S}$ gibt, so dass H_{i-1} für $i = 1, \ldots, s$ mittels S_{p_i} zu H_i reduzierbar ist. Wenn es eine Reduktionskette mit $H_s = (\emptyset, \emptyset)$ gibt, so heißt H *(in s Stufen) vollständig reduzierbar*. Für den Fall, dass ein Hypergraph H nicht mit Hilfe wenigstens einer Reduktion aus $\mathcal{S}$ reduziert werden kann, heißt er *irreduzibel* bezüglich $\mathcal{S}$.

Reduktionen können wie folgt algorithmisch umgesetzt werden:

1. Suche in H einen Untergraphen isomorph zu H_1, der nur an den Knoten von K mit dem Rest von H zusammenhängt.
2. Ersetze diesen Untergraphen durch H_1'. Der hierdurch entstehende Hypergraph sei H'.
3. Bestimme in diesem *reduzierten* Hypergraphen H' den Wert $w' = R(H')$.
4. Transformiere w' durch Anwendung von r auf die gesuchte Größe w im Ausgangshypergraphen H.

Bemerkung 3.15. Die Anwendung einer Reduktion ist natürlich nur sinnvoll, wenn die Bestimmung von w' einfacher ist als die Lösung des Ausgangsproblems, wenn also zum Beispiel H' bezüglich der Kanten- und Knotenzahl kleiner als H ist oder H' eine bestimmte Struktur aufweist, für die Lösungen besser bestimmt werden können.

Bemerkung 3.16. Reduktionen sind sehr vielfältig. Für das im Abschnitt 5.6.1 erwähnte Problem der Berechnung der Zusammenhangswahrscheinlichkeit stochastischer Graphen sind allein 17 verschiedene Reduktionen bekannt. Für das allgemeinere Problem der Bestimmung der K-Zusammenhangswahrscheinlichkeit erhöht sich diese Zahl auf ca. 25 (vgl. dazu insbesondere [PoS86, PoS90]). In fast allen dieser Fälle wird H' aus H erzeugt, indem Strukturen in der Nähe von Knoten geringen Grades (bis maximal vier) durch Ersatzstrukturen mit weniger Kanten oder Knoten ersetzt werden. Das Auffinden solcher Strukturen kann in polynomialer Zeit erfolgen, bei geeigneter Speicherung des Hypergraphen teilweise sogar in sublinearer Zeit. Die Transformation $w' \to w$ ist hier immer $\mathcal{O}(1)$, so dass die *Reduktionsmethode*, also das wiederholte Anwenden von Reduktionen, zu polynomialen Algorithmen führt, wenn der Graph vollständig reduzierbar ist.

Bemerkung 3.17. Die Reduktionsmethode hat zwei wesentliche Nachteile: Reduktionen sind sehr stark vom konkreten Problem abhängig und viele Graphen sind irreduzibel. Bei der Berechnung der Zusammenhangswahrscheinlichkeit stochastischer Graphen gibt es keine Reduktionen für schlichte Graphen, die ausschließlich aus Knoten vom Grad fünf und mehr bestehen. Selbst weniger dichte Strukturen können hier schon irreduzibel sein, so dass die Berechnung dieser Graphenkenngröße in beliebigen Graphen ausschließlich mit Hilfe der Reduktionsmethode nicht möglich ist.

Allerdings besteht häufig die Möglichkeit, die Reduktion mit der Dekomposition zu kombinieren und so zum Teil deutlich bessere Ergebnisse zu erreichen: Obwohl weder Dekomposition noch Reduktion als alleinige Methode zur Berechnung der Zusammenhangswahrscheinlichkeit von Graphen mit ca. 20 bis 30 Knoten und 50 bis 70 Kanten praktisch anwendbar sind, ergibt sich durch die Kombination der beiden Methoden sehr wohl ein Verfahren, mit dem Graphen dieser Größenordnung berechenbar sind. Dabei wird ein Graph zunächst durch Reduktionen so lange verkleinert, bis keine der bekannten Reduktionen mehr anwendbar ist. Danach wird ein einziger Dekompositionsschritt ausgeführt und die beiden resultierenden Untergraphen werden rekursiv mit dem gleichen Verfahren bearbeitet.

Dabei haben sowohl die Struktur des Ausgangsgraphen als auch die Wahl der Pivotelemente der Dekompositionsschritte einen wesentlichen Einfluss auf das Laufzeitverhalten des Verfahrens.

4 Die Kompositionsmethode

Eine konsequente Verfolgung des Splittinggedankens führt zu der Idee, die beim einfachen Splitting entstehenden Teilprobleme ebenfalls mit der (unsymmetrischen) Splittingmethode zu bearbeiten. Dabei zeigt sich erwartungsgemäß zunächst eine positive Korrelation der Leistung der Algorithmen mit der Zahl der Teile T_i in der Kontaktzerlegung. Problematisch sind aber die unregelmäßige Form der Teile und der erhebliche Koppelaufwand, der bei zunehmender Zahl der Teile bald die Rechenzeit dominiert. Der Aufwand der Verwaltung strukturell verschiedener Teile und mehrerer Kontaktknotenmengen sowie der komplizierter werdende Algorithmus relativieren den erwähnten Leistungsvorteil in jeder tatsächlich durchgeführten Berechnung weiter, so dass diese Methode des *Mehrfachsplittings* kaum praktische Bedeutung besitzt.

Die beiden genannten Grundprobleme sind jedoch zumindest für einige Problemstellungen mit dem gleichen Ansatz lösbar: Wählt man als Bestandteile einer Kontaktzerlegung jeweils strukturell gleichartige Untergraphen bzw. nur eine kleine Anzahl von Klassen solcher Untergraphen, so reduziert sich zunächst der Verwaltungsaufwand. Es zeigt sich außerdem, dass sich auf diesen speziell strukturierten Kontaktgraphen nicht nur oft die Teilprobleme explizit lösen lassen, sondern dass es in der Regel auch möglich ist, die normalerweise bei der Kopplung der Teile auftretende Dreifachsumme auf eine einfache Summe zu reduzieren. Die Umsetzung dieser Idee führt zu der in diesem Kapitel beschriebenen Kompositionsmethode und ihren Spezialfällen *Kantenkomposition* und *Knotenkomposition.*

4.1 Grundform der Komposition

Die *Kompositionsmethode* stellt eine Weiterentwicklung der Methode des Mehrfachsplittings dar. Der wesentliche Unterschied besteht dabei in der unterschiedlichen Abfolge der Berechnungen: Bei Algorithmen, die das Mehrfachsplitting implementieren, erfolgen die Berechnungen hierarchisch in einer Art Baumstruktur. Dabei findet in deren Blättern die Lösung der Teilprobleme und in den inneren Knoten die Verknüpfung der Teilergebnisse, die in Form von Zustandsmengen vorliegen, statt. Dagegen wird in Kompositionsalgorithmen diese Hierarchie in eine lineare Form gebracht, die dann in einem einzelnen Durchgang bearbeitet wird.

Sei R eine Graphenkenngröße mit der Indexform $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$, H ein fixierter Hypergraph und $(H_1, T_2, \ldots, T_t)$ eine Kontaktzerlegung von H, die gemäß Definition 2.2 aktive Knotenmengen $A_1, \ldots, A_t$ sowie die Mengen der zu deaktivierenden Knoten $D_1, \ldots, D_t$ induziert. Zunächst wird die Zustandsmenge $\mathcal{Z}_1 = \mathcal{Z}_{H_1,A_1}$ für den ersten Unterhypergraphen $H_1 = (V_1, E_1)$ bestimmt, wie sie auch notwendig wäre, wenn das Problem durch einmaliges Splitting an der ersten aktiven Knotenmenge A_1 in lediglich zwei Teile $H_1 = (V_1, E_1)$ und $H_1^{'} = ((V - V_1) \cup A_1), E - E_1)$ gelöst werden sollte. Es wird nun aber keine Zustandsmenge für $H_1^{'}$ bestimmt, um diese dann wie bei den Splittingmethoden mit $\mathcal{Z}_1$ zu koppeln.

Vielmehr wird $\mathcal{Z}_1$ so transformiert, dass das Ergebnis $\mathcal{Z}_2$ dieser *Transformation über* T_2 derjenigen Zustandsmenge $\mathcal{Z}_{H_1+T_2,A_2}$ entspricht, die für $H_1 + T_2$ zu bestimmen wäre, wenn das Problem durch einmaliges Splitting an der zweiten trennenden Knotenmenge A_2 in die Teile $H_1+T_2 = (V_1 \cup V_2, E_1 \cup E_2)$ und $((V - V_1 - V_2) \cup A_2, E - E_1 - E_2)$ gelöst werden sollte. Anschließend wird $\mathcal{Z}_2$ durch eine ähnliche Transformation in eine Zustandsmenge $\mathcal{Z}_3 = \mathcal{Z}_{H_1+T_2+T_3,A_3}$ für $H_1 + T_2 + T_3$ umgerechnet usw. Am Ende der Berechnungen entsteht schließlich eine Zustandsmenge $\mathcal{Z}_t$ für $H_1 + T_2 + \ldots + T_t = H$. Aus dieser kann durch die Projektion (2.10) direkt der gesuchte Wert $R(H)$ gewonnen werden.

Zur Vereinfachung der weiteren Ausführungen bietet es sich an, nicht mit $\mathcal{Z}_1$ zu starten, sondern mit einer den leeren Hypergraphen $H_0 = (\emptyset, \emptyset)$ repräsentierende Zustandsmenge $\mathcal{Z}_0 = \mathcal{Z}_{(\emptyset,\emptyset),\emptyset}$. Die Menge $\mathcal{Z}_1$ kann daraus völlig analog zu den danach folgenden Schritten durch Transformation über $T_1 = (V_1, E_1, \emptyset)$ gewonnen werden.

Definition 4.1. Sei $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ eine Indexform für eine Graphenkenngröße R. Gilt für alle Kontaktzerlegungen $(T_1, \ldots, T_t)$ aller Hypergraphen $H \in \mathcal{H}$ für $j = 1, \ldots, t-1$ für eine Funktion trs

$$\biguplus_{z \in \mathcal{Z}_j} \mathrm{trs}(z, T_{j+1}) \quad = \quad \mathcal{Z}_{j+1}, \tag{4.1}$$

so heißt das aus Elementen der Indexform sowie der Funktion trs bestehende Tupel $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{trs})$ *Kompositionsform* für die Graphenkenngröße R. Die Funktion trs heißt dabei *Transformationsfunktion.*

Kompositionsalgorithmen

Sei $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{trs})$ eine Kompositionsform. Dann gilt unter Verwendung der *Transformation*

$$\mathrm{tr}(\mathcal{Z}, T) \quad = \quad \biguplus_{z \in \mathcal{Z}} \mathrm{trs}(z, T) \tag{4.2}$$

wegen (2.10)

$$R(H) \quad = \quad \mathrm{pr}(\mathrm{tr}(\mathrm{tr}(\ldots \mathrm{tr}(\mathcal{Z}_0, T_1) \ldots, T_{t-1}), T_t)). \tag{4.3}$$

Eine Kompositionsform kann also wie folgt direkt in einen *Kompositionsalgorithmus* umgesetzt werden:

```
Funktion Projektion(Zustandsmenge 𝒵)
  w := O
  Für alle z in 𝒵:
    w := w ⊕ prs(z)
  Rückgabe w

Funktion Transformation(Zustandsmenge 𝒵, Kontaktgraph T)
  𝒵' := ∅
```

```
    Für alle z in 𝒵:
      𝒵' := 𝒵' ⊎ trs(z, T)
    Rückgabe 𝒵'

Algorithmus Komposition(Hypergraph H)
  Bestimme eine beliebige Kontaktzerlegung (T_1, ..., T_t) von H
  Bestimme 𝒵_0
  Für alle i von 1 bis t:
    𝒵_i := Transformation(𝒵_{i-1}, T_i)
  Rückgabe Projektion(𝒵_t)
```

Bemerkung 4.2. Im Verlaufe eines Kompositionsalgorithmus entsteht aus der Kontaktzerlegung $(T_1, \ldots, T_t)$ in Form von $T_1, T_1 + T_2, \ldots, T_1 + \cdots + T_t$ eine Folge von Unterhypergraphen $H_1, \ldots, H_t$ des Ausgangshypergraphen H. Dabei trennt die jeweils aktive Knotenmenge A_i, $i \in \{1, \ldots t\}$ (vgl. (2.1)) den Hypergraphen H in einen „vollständig berechneten Teil“ und in einen „ungesehenen Teil“. Die aktiven Knoten gehören zu keinem dieser Teile, sie sind vielmehr „bereits gesehen, aber noch nicht vollständig berechnet“. Eine Position i in der Kontaktzerlegung entspricht einem Zeitpunkt während des Ablaufs des Algorithmus. Sowohl die Folge der Hypergraphen H_i als auch die Folge der aktiven Knotenmengen A_i können also als eine Art Zeitachse angesehen werden. Dabei durchläuft jeder Knoten eine von zwei Laufbahnen: Entweder er durchläuft die Stadien *ungesehen*, *aktiv*, *berechnet* in genau dieser Reihenfolge, oder aber er wechselt direkt von *ungesehen* nach *berechnet*. Es hängt lediglich von der gewählten Kontaktzerlegung ab, zu welchem Zeitpunkt ein Knoten aktiviert bzw. berechnet wird.

Finden von Kompositionsformen

Auch bei der Suche nach Kompositionsformen ist unklar, wie im allgemeinen Fall aus einer Beschreibung einer Graphenkenngröße R in Grundform die zur Implementierung der Kompositionsalgorithmen notwendigen Funktionen ind und trs gewonnen werden können. Insbesondere ist das Finden einer geeigneten Indizierung nicht immer einfach. Dabei hilft oft der Vergleich mit ähnlichen Problemen sowie die folgende Heuristik:

> *Die Indizes der Zustände aus $\mathcal{Z}_i$ werden aus Teilmengen oder Partitionen von A_i oder Kombinationen solcher Objekte gebildet und hängen nur von den Knoten der jeweils aktuellen aktiven Knotenmenge A_i, nicht aber von den übrigen Knoten des Graphen ab.*

Um diese Regel einhalten zu können, ohne die Suche nach der Transformationsfunktion trs zu erschweren, bietet es sich an, die Aufgaben der Funktion trs auf zwei Funktionen $\mathrm{trs}^{\mathrm{L}}$ und $\mathrm{trs}^{\mathrm{D}}$ aufzuteilen und dafür die individuellen Anforderungen an $\mathrm{trs}^{\mathrm{L}}$ und $\mathrm{trs}^{\mathrm{D}}$ zu verringern. Es soll also gelten:

$$\mathrm{trs}(z, T) \quad = \quad \mathrm{trs}^{\mathrm{D}}(\mathrm{trs}^{\mathrm{L}}(z, T), D)$$

Dabei übernimmt $\mathrm{trs}^{\mathrm{L}}$ die eigentliche Transformation (also das „Einarbeiten der Information des Kontaktgraphen T in einen einzelnen Zustand z“), ohne dabei den Einschränkungen durch

die angegebene Heuristik zu unterliegen. Die *Deaktivierungsfunktion* $\mathrm{trs}^{\mathrm{D}}$ sorgt im Anschluss für das Einhalten der Heuristik, ohne noch einmal auf die Konstellationen von T zurückzugreifen. Die Menge D besteht aus den zu deaktivierenden Knoten, das heißt aus den Knoten, die zuvor aktiv waren bzw. im aktuellen Transformationsschritt hinzugekommen sind, nach diesem jedoch nicht (mehr) aktiv sein sollen.

Sind solche Funktionen $\mathrm{trs}^{\mathrm{L}}$ und $\mathrm{trs}^{\mathrm{D}}$ gefunden, kann die Berechnung von $R(H)$ durch die folgende modifizierte Form des Grundalgorithmus der Komposition erfolgen:

```
Funktion Projektion(Zustandsmenge 𝒵)
  w := O
  Für alle z in 𝒵:
    w := w ⊕ prs(z)
  Rückgabe w

Funktion Transformation(Zustandsmenge 𝒵, Kontaktgraph T)
  𝒵' := ∅
  Für alle z in 𝒵:
    𝒵' := 𝒵' ⊎ trs^L(z, T)
  Rückgabe 𝒵'

Funktion Deaktivierung(Zustandsmenge 𝒵, Knotenmenge D)
  𝒵' := ∅
  Für alle z in 𝒵:
    𝒵' := 𝒵' ⊎ trs^D(z, D)
  Rückgabe 𝒵'

Algorithmus ModifizierteKomposition(Hypergraph H)
  Bestimme eine beliebige Kontaktzerlegung (T_1, ..., T_t) von H
  Bestimme 𝒵_0
  Für alle i von 1 bis t:
    𝒵 := Transformation(𝒵_{i-1}, T_i)
    𝒵_i := Deaktivierung(𝒵, D_i)
  Rückgabe Projektion(𝒵_t)
```

Die Menge D_i bezeichnet dabei wie immer die durch (2.2) definierte Menge der zu deaktivierenden Knoten.

Korrektheit

Korrektheitsbeweise müssen prinzipiell für jeden Kompositionsalgorithmus separat geführt werden. Sie können aber einem einheitlichen Schema folgen: Zuerst wird ein Tupel $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{trs})$ angegeben, wobei $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ eine Indexform ist. Danach folgt ein Induktionsargument: Sei $(T_1, \ldots, T_t)$ eine beliebige Kontaktzerlegung eines Hypergraphen H. Wenn $\mathcal{Z}_0$ korrekt ist und trs eine Transformationsfunktion ist, so bestimmt der Algorithmus den Wert

$$\mathrm{pr}(\mathrm{tr}(\mathrm{tr}(\ldots \mathrm{tr}(\mathcal{Z}_0, T_1)\ldots), T_{t-1}), T_t)$$

und damit $R(H)$. Es genügt also, die beiden genannten Punkte zu überprüfen.

Die Überprüfung von $\mathcal{Z}_0$ ist in den Anwendungen trivial. Beim Nachweis, dass trs eine Transformationsfunktion ist, hilft folgendes Lemma:

Lemma 4.3. *Sei* $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ *eine Indexform für eine Graphenkenngröße* R *und seien ein Unterhypergraph* $H_1 = (W_1, F_1)$ *eines beliebigen Hypergraphen* $H_2 = (W_2, F_2) \in \mathcal{H}$ *und ein Kontaktgraph* $T_2 = (V_2, E_2, K_2) \in \mathcal{T}$ *so gegeben, dass* (H_1, T_2) *eine Kontaktzerlegung von* H_2 *ist. Bezeichne* $\mathcal{C}_1$ *den Konstellationsraum* $\mathcal{C}_{H_1}(c_\mathrm{V}, c_\mathrm{E})$, $\mathcal{C}_2$ *den Konstellationsraum* $C_{T_2}(c_\mathrm{V}, c_\mathrm{E})$ *und* $\mathcal{C}$ *den Konstellationsraum* $\mathcal{C}_{H_2}(c_\mathrm{V}, c_\mathrm{E})$. *Seien zwei Zielmengen* $A_1 \subseteq W_1$ *und* $A_2 \subseteq A_1 \cup V_2$ *gegeben und bezeichne*

$$\begin{aligned}
\mathrm{ev}_1(i) &= \bigoplus_{\substack{C_1 \in \mathcal{C}_1 \\ \mathrm{ind}(C_1, A_1) = i}} \mathrm{val}(C_1) \\
\mathrm{ev}(i) &= \bigoplus_{\substack{C \in \mathcal{C} \\ \mathrm{ind}(C, A_2) = i}} \mathrm{val}(C) \\
\mathcal{I}_1 &= \{\mathrm{ind}(C_1, A_1) : C_1 \in \mathcal{C}_1\} \\
\mathcal{I}_2 &= \{\mathrm{ind}(C, A_2) : C \in \mathcal{C}\} \\
\mathcal{Z}_1 &= \biguplus_{i_1 \in \mathcal{I}_1} (i_1, \mathrm{ev}_1(i_1)) \\
\mathcal{Z}_2 &= \biguplus_{C \in \mathcal{C}} (\mathrm{ind}(C, A_2), \mathrm{val}(C)).
\end{aligned} \tag{4.4}$$

Es sei trz^L *eine Funktion, die für einen beliebigen Index* $i \in \mathcal{I}_1$, *beliebige Werte* $w_1, w_2 \in \mathcal{W}$ *und Konstellationen* $C_1 \in \mathcal{C}_1$, $C_2 \in \mathcal{C}_2$

$$\mathrm{trz}^\mathrm{L}((\mathrm{ind}(C_1, A_1), \mathrm{val}(C_1)), C_2) = (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2), \mathrm{val}(C_1 \cup C_2)) \tag{4.5}$$

$$\mathrm{trz}^\mathrm{L}((i, w_1 \oplus w_2), C_2) = \mathrm{trz}^\mathrm{L}((i, w_1), C_2) \uplus \mathrm{trz}^\mathrm{L}((i, w_2), C_2) \tag{4.6}$$

erfüllt, und sei trz^D *eine Funktion, für die für beliebige Indizes* $i_1, i_2 \in \mathcal{I}_1$, *beliebige Werte* $w_1, w_2 \in \mathcal{W}$, *beliebige Konstellationen* $C \in \mathcal{C}$ *und* $C_2 \in \mathcal{C}_2$, *beliebiges* v *und* A *mit* $v \in A \subseteq A_1 \cup V_2$ *sowie* $D_1 = (V_2 \cup A_1) - A_2$ *gilt:*

$$\mathrm{trz}^\mathrm{D}((\mathrm{ind}(C, A), \mathrm{val}(C)), v) = (\mathrm{ind}(C, A - v), \mathrm{val}(C)) \tag{4.7}$$

$$\mathrm{trz}^\mathrm{D}((i_1, w_1) \uplus (i_2, w_2), v) = \mathrm{trz}^\mathrm{D}((i_1, w_1), v) \uplus \mathrm{trz}^\mathrm{D}((i_2, w_2), v) \tag{4.8}$$

Dann gilt für die durch

$$\mathrm{trs}(z, T) = \mathrm{trs}^\mathrm{D}(\mathrm{trs}^\mathrm{L}(z, T), D_1) \tag{4.9}$$

$$\mathrm{trs}^\mathrm{L}(z, T) = \biguplus_{C \in \mathcal{C}_T} \mathrm{trz}^\mathrm{L}(z, C) \tag{4.10}$$

$$\mathrm{trs}^\mathrm{D}(z, \{v_{d_1}, \ldots, v_{d_r}\}) = \mathrm{trz}^\mathrm{D}(\mathrm{trz}^\mathrm{D}(\ldots \mathrm{trz}^\mathrm{D}(z, v_{d_1}) \ldots, v_{d_{r-1}}), v_{d_r}) \tag{4.11}$$

gegebene Funktion trs *die Beziehung* $\biguplus_{z \in \mathcal{Z}_1} \mathrm{trs}(z, T_2) = \mathcal{Z}_2$.

Beweis. Es gilt mit $D_1 = \{v_{d_1}, \dots v_{d_r}\}$

$$
\begin{aligned}
&\biguplus_{z \in \mathcal{Z}_1} \operatorname{trs}(z, T_2) \\
&\quad = \quad \biguplus_{i \in \mathcal{I}_1} \operatorname{trs}\left((i,\, \operatorname{ev}_1(i)), T_2\right) \\
&\quad \overset{(4.9)}{=} \quad \biguplus_{i \in \mathcal{I}_1} \operatorname{trs}^{\mathrm{D}}\left(\operatorname{trs}^{\mathrm{L}}((i,\, \operatorname{ev}_1(i)), T_2), D_1\right) \\
&\quad \overset{(4.10)}{=} \quad \biguplus_{i \in \mathcal{I}_1} \operatorname{trs}^{\mathrm{D}}\left(\biguplus_{C_2 \in \mathcal{C}_2} \operatorname{trz}^{\mathrm{L}}((i,\, \operatorname{ev}_1(i)), C_2), D_1\right) \\
&\quad \overset{(4.11)}{=} \quad \biguplus_{i \in \mathcal{I}_1} \operatorname{trz}^{\mathrm{D}}\left(\dots \operatorname{trz}^{\mathrm{D}}\left(\biguplus_{C_2 \in \mathcal{C}_2} \operatorname{trz}^{\mathrm{L}}((i,\, \operatorname{ev}_1(i)), C_2), v_{d_1}\right) \dots v_{d_r}\right) \\
&\quad \overset{(4.4)}{=} \quad \biguplus_{i \in \mathcal{I}_1} \operatorname{trz}^{\mathrm{D}}\left(\dots \operatorname{trz}^{\mathrm{D}}\left(\biguplus_{C_2 \in \mathcal{C}_2} \operatorname{trz}^{\mathrm{L}}((i,\, \bigoplus_{\substack{C_1 \in \mathcal{C}_1 \\ \operatorname{ind}(C_1, A_1) = i}} \operatorname{val}(C_1)), C_2), v_{d_1}\right) \dots v_{d_r}\right) \\
&\quad \overset{(4.6)}{=} \quad \biguplus_{i \in \mathcal{I}_1} \operatorname{trz}^{\mathrm{D}}\left(\dots \operatorname{trz}^{\mathrm{D}}\left(\biguplus_{C_2 \in \mathcal{C}_2} \biguplus_{\substack{C_1 \in \mathcal{C}_1 \\ \operatorname{ind}(C_1, A_1) = i}} \operatorname{trz}^{\mathrm{L}}((i, \operatorname{val}(C_1)), C_2), v_{d_1}\right) \dots v_{d_r}\right) \\
&\quad \overset{(4.8)}{=} \quad \biguplus_{i \in \mathcal{I}_1} \biguplus_{C_2 \in \mathcal{C}_2} \biguplus_{\substack{C_1 \in \mathcal{C}_1 \\ \operatorname{ind}(C_1, A_1) = i}} \operatorname{trz}^{\mathrm{D}}\left(\dots \operatorname{trz}^{\mathrm{D}}\left(\operatorname{trz}^{\mathrm{L}}((i, \operatorname{val}(C_1)), C_2), v_{d_1}\right) \dots v_{d_r}\right) \\
&\quad = \quad \biguplus_{C_2 \in \mathcal{C}_2} \biguplus_{C_1 \in \mathcal{C}_1} \operatorname{trz}^{\mathrm{D}}\left(\dots \operatorname{trz}^{\mathrm{D}}\left(\operatorname{trz}^{\mathrm{L}}((\operatorname{ind}(C_1, A_1), \operatorname{val}(C_1)), C_2), v_{d_1}\right) \dots v_{d_r}\right) \\
&\quad \overset{(4.5)}{=} \quad \biguplus_{C_2 \in \mathcal{C}_2} \biguplus_{C_1 \in \mathcal{C}_1} \operatorname{trz}^{\mathrm{D}}\left(\dots \operatorname{trz}^{\mathrm{D}}\left((\operatorname{ind}(C_1 \cup C_2, A_1 \cup V_2),\, \operatorname{val}(C_1 \cup C_2)), v_{d_1}\right) \dots v_{d_r}\right) \\
&\quad = \quad \biguplus_{C \in \mathcal{C}} \operatorname{trz}^{\mathrm{D}}\left(\dots \operatorname{trz}^{\mathrm{D}}\left((\operatorname{ind}(C, A_1 \cup V_2),\, \operatorname{val}(C)), v_{d_1}\right) \dots v_{d_r}\right) \\
&\quad \overset{(4.7)}{=} \quad \biguplus_{C \in \mathcal{C}} (\operatorname{ind}(C, (A_1 \cup V_2) - D_1),\, \operatorname{val}(C)) \\
&\quad = \quad \biguplus_{C \in \mathcal{C}} (\operatorname{ind}(C, A_2),\, \operatorname{val}(C)) \\
&\quad = \quad \mathcal{Z}_2.
\end{aligned}
$$

□

Satz 4.4. *Es gelten die Voraussetzungen und Bezeichungen des vorangegangenen Lemma 4.3. Dann ist die durch (4.1) konstruierte Funktion* trs *eine Transformationfunktion und ergänzt somit die Indexform* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \operatorname{val}, \operatorname{ind})$ *zu einer Kompositionsform* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \operatorname{val}, \operatorname{ind}, \operatorname{trs})$.

Beweis. Sei $(H_1, T_2, \ldots, T_t)$ eine Kontaktzerlegung eines Hypergraphen H und bezeichne $H_j = H_1 + T_2 + \ldots + T_j$ sowie

$$\mathcal{Z}_r \quad = \quad \biguplus_{i \in \{\mathrm{ind}(C, A_r) : C \in \mathcal{C}_{H_r}\}} (i,\, \mathrm{ev}_{H_r, A_r}(i)).$$

Durch Verschiebung der Indizes in Lemma 4.3 erhält man die Gültigkeit von

$$\biguplus_{z \in \mathcal{Z}_r} \mathrm{trs}(z, T_{r+1}) = \mathcal{Z}_{r+1}$$

für alle $r = 1, \ldots, t-1$. Damit folgt die Behauptung direkt aus der Definition 4.1. □

Für den Nachweis, dass die Funktion trs eine Transformationsfunktion ist, genügt es also, die Teile $\mathrm{trs}^{\mathrm{L}}$ und $\mathrm{trz}^{\mathrm{D}}$ separat zu betrachten. Die Gültigkeit der „technischen" Voraussetzungen (4.6) und (4.8) ist meist offensichtlich, so dass im Wesentlichen der Nachweis von (4.5) und (4.7) verbleibt.

Bemerkung 4.5. Der Beweis von Satz 4.4 verwendet Lemma 4.3 nur für solche Zielmengen, die einer aus einer bestimmten Kontaktzerlegung des zu berechnenden Graphen stammenden aktiven Knotenmenge entsprechen. Es genügt also prinzipiell, das Lemma nur für derartige Zielmengen zu beweisen. Im vorliegenden allgemeinen Fall der Komposition ergeben sich aus dieser Erkenntnis keine Vereinfachungen, da die aktiven Mengen einer Kontaktzerlegung tatsächlich (fast) beliebig sein können. In den später behandelten Speziallfällen *Kanten-* und *Knotenkomposition* sind die verwendeten Kontaktzerlegungen allerdings strukturell sehr eingeschränkt und führen zu sehr speziellen aktiven Knotenmengen, deren besondere Struktur beim Nachweis der Beziehungen (4.5) und (4.7) helfen kann.

Komplexität

Es lassen sich kaum konkrete Aussagen bezüglich der Komplexität der Kompositionsalgorithmen treffen, die unabhängig von der zu berechnenden Graphenkenngröße sind. Die Anzahl der Zustände kann in jeder Zustandsmenge durch die Zahl aller möglichen Indizes nach oben begrenzt werden. Diese Zahl wiederum ist bei Einhaltung der angeführten Heuristik jedoch nur von der Mächtigkeit der trennenden Knotenmenge abhängig, insbesondere also zum Beispiel in Graphen beschränkter Wegweite bei geeigneter Wahl der Kontaktzerlegung durch eine Konstante beschränkt.

Die Komplexität des Algorithmus kann damit abgeschätzt werden durch die Summe (bzw. das Maximum) (a) des Aufwands für den Zerlegungsschritt, (b) des Bestimmens von $\mathcal{Z}_0$, (c) des Projektionsschritts und (d) des Produkts aus der Anzahl der Kontaktgraphen aus der Kontaktzerlegung, aus der maximal möglichen Zustandszahl und dem Aufwand zur Transformation eines einzelnen Zustandes.

In den im Kapitel 5 gezeigten Beispielen wird davon ausgegangen, dass bereits eine Kontaktzerlegung vorliegt, der Aufwand zur Bestimmung von $\mathcal{Z}_0$ vernachlässigbar ist, und dass der Aufwand für die Projektion vom Aufwand für alle Transformationen dominiert wird, so dass der Gesamtaufwand im Wesentlichen durch (d) bestimmt ist. Diese Annahme wird durch die numerischen Experimente bestätigt. Der tatsächliche Aufwand für das Finden einer Kontaktzerlegung wird in Anhang B diskutiert.

Implementation

Die Implementation von Kompositionsalgorithmen ist sehr einfach. Die Zustandsmengen können direkt als Feld von Zuständen implementiert werden, die Transformation bestimmt aus einem solchen Feld ein neues. Die meisten Algorithmen erlauben auch eine *in situ*-Transformation zumindest eines Teils der jeweiligen Zustandsmengen, so dass der Gesamtaufwand durch Vermeiden unnötiger Kopien von Zuständen gesenkt werden kann. Die formale Komplexität der Algorithmen ändert sich dabei zwar nicht, dennoch sind die dadurch zu erzielenden Leistungssteigerungen beträchtlich und können je nach Problemstellung und konkretem Graphen bis zu einem Faktor drei beim Speicherverbrauch und einem Faktor fünf bei der Rechenzeit ausmachen.

Problemspezifische Optimierungen sind insbesondere bei Kompositionsformen, die die in der Bemerkung 2.22 beschriebene Abschlussmethode benutzen, häufig möglich und aus praktischen Gründen meist notwendig. In diesem Fall können Zustände, die als unzulässig erkannt werden, unmittelbar aus der jeweils aktuellen Zustandsmenge entfernt werden, da sie am Ende nichts zum Ergebnis beitragen und bis dahin nur unnötigen Rechen- und Speicheraufwand bereiten würden.

Aus Gründen der Effizienz sollte dabei darauf geachtet werden, dass die unzulässigen Konstellationen möglichst frühzeitig erkannt werden. Wann das konkret geschehen kann, ist problemabhängig. In einigen Fällen ist es dabei hilfreich, im Index der Zustände mehr Information zu kodieren, als für das Aufstellen einer Index- bzw. Kompositionsform minimal notwendig ist. Solche Erweiterungen sind aber nur eine Optimierungen im Sinne der Verbesserung bestimmter die Laufzeit bestimmender konstanter Faktoren, die aber nichts am theoretischen asymptotischen Verhalten ändern.

4.2 Kantenkomposition

Algorithmus

Die *Kantenkomposition* stellt einen wichtigen Spezialfall der Komposition dar, da sie zu sehr einfachen Algorithmen führt, aber dennoch nur wenig an Flexibilität gegenüber der im vorangegangenen Abschnitt beschriebenen allgemeinen Form der Komposition verliert. Die Kantenkomposition zeichnet sich gegenüber der allgemeinen Form dadurch aus, dass die im Algorithmus verwendete Kontaktzerlegung des zu berechnenden Hypergraphen $H = (V, E)$ eine Kantenzerlegung sein muss (vgl. Abschnitt 2.4 für die entsprechende Definition).

Eine Kantenzerlegung $(H_0 = (\emptyset, \emptyset), T_1, \ldots, T_t)$ enthält nur zwei Typen von Kontaktgraphen, nämlich $(\{v\}, \emptyset, \emptyset)$ mit $v \in V$ und $(e, \{e\}, e)$ mit $e \in E$, so dass im Verlaufe des Kompositionsalgorithmus ebenfalls nur zwei Typen von Transformationen auftreten können. Eine Transformation, die zu einem Kontaktgraphen der Form $(\{v\}, \emptyset, \emptyset)$ gehört, soll dabei als *Knotenaktivierungsschritt für* v, die andere Form als *Kantenschritt für* e bezeichnet werden.

Es sei $\mathrm{trs}^{\mathrm{V}}(z, v)$ eine Kurzschreibweise für $\mathrm{trs}(z, (\{v\}, \emptyset, \emptyset))$ und $\mathrm{trs}^{\mathrm{E}}(z, e)$ für $\mathrm{trs}(z, (e, \{e\}, e))$. Durch die Spezialisierung auf Kantenzerlegungen kann der modifizierte Grundalgorithmus unter Verwendung der unveränderten Funktionen `Deaktivierung` und `Projektion` wie folgt umgeschrieben werden:

```
Funktion Knotenaktivierung(Zustandsmenge 𝒵, Knoten v)
  𝒵' := ∅
  Für alle z in 𝒵:
    𝒵' := 𝒵' ⊎ trs^V(z, v)
  Rückgabe 𝒵'

Funktion Kantenschritt(Zustandsmenge 𝒵, Kante e)
  𝒵' := ∅
  Für alle z in 𝒵:
    𝒵' := 𝒵' ⊎ trs^E(z, e)
  Rückgabe 𝒵'

Algorithmus Kantenkomposition(Hypergraph H = (V, E))
  Wähle eine (beliebige) Permutation π = (e_{k_1}, ..., e_{k_m}) von E
  Bestimme aus π eine Kantenzerlegung (H_0, T_1, ..., T_{|E|+|V|}) von H
  Bestimme 𝒵_0 = 𝒵_{H_0,∅}
  Für alle i von 1 bis |E| + |V|:
    Falls T_i von der Form ({v}, ∅, ∅) ist:
      𝒵 := Knotenaktivierung(𝒵_{i-1}, v)
    Sonst: (T_i ist von der Form (e, {e}, e))
      𝒵 := Kantenschritt(𝒵_{i-1}, e)
    𝒵_i := Deaktivierung(𝒵, D_i)
  Rückgabe Projektion(𝒵_{|E|+|V|})
```

Korrektheit

Die Funktionen $\mathrm{trs}^{\mathrm{V}}$, $\mathrm{trs}^{\mathrm{E}}$ und $\mathrm{trs}^{\mathrm{D}}$ beschreiben in diesem Fall trs vollständig, so dass im Zusammenhang mit der Kantenkomposition auch das Tripel $(\mathrm{trs}^{\mathrm{V}}, \mathrm{trs}^{\mathrm{E}}, \mathrm{trs}^{\mathrm{D}})$ anstelle von trs in der Kompositionsform verwendet werden kann.

In Anlehnung an den allgemeinen Fall genügt es zum Korrektheitsnachweis der *Kantenkompositionsform* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{ind}, \mathrm{prs}, (\mathrm{trs}^{\mathrm{V}}, \mathrm{trs}^{\mathrm{E}}, \mathrm{trs}^{\mathrm{D}}))$ aufgrund von Satz 4.4 mit den dort verwendeten Bezeichnungen, die Beziehungen

$$\begin{aligned}
\mathrm{trz}^{\mathrm{D}}((\mathrm{ind}(C, A), \mathrm{val}(C)), v) &= (\mathrm{ind}(C, A - v), \mathrm{val}(C)) \\
\mathrm{trs}^{\mathrm{V}}((\mathrm{ind}(C_1, A_1), \mathrm{val}(C_1)), v) &= \biguplus_{C_2 \in \mathcal{C}_{(\{v\},\emptyset,\emptyset)}} (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2), \mathrm{val}(C_1 \cup C_2)) \quad (4.12) \\
\mathrm{trs}^{\mathrm{E}}((\mathrm{ind}(C_1, A_1), \mathrm{val}(C_1)), e) &= \biguplus_{C_2 \in \mathcal{C}_{(e,\{e\},e)}} (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2), \mathrm{val}(C_1 \cup C_2))
\end{aligned}$$

sowie die „technischen“ Voraussetzungen

$$\begin{aligned}
\mathrm{trz}^{\mathrm{D}}(z_1 \uplus z_2, D) &= \mathrm{trz}^{\mathrm{D}}(z_1, D) \uplus \mathrm{trz}^{\mathrm{D}}(z_2, D) \\
\mathrm{trz}^{\mathrm{V}}((i, w_1 \oplus w_2), C_2) &= \mathrm{trz}^{\mathrm{V}}((i, w_1), C_2) \uplus \mathrm{trz}^{\mathrm{V}}((i, w_2), C_2) \\
\mathrm{trz}^{\mathrm{E}}((i, w_1 \oplus w_2), C_2) &= \mathrm{trz}^{\mathrm{E}}((i, w_1), C_2) \uplus \mathrm{trz}^{\mathrm{E}}((i, w_2), C_2)
\end{aligned}$$

zu überprüfen.

Falls bereits die Gültigkeit der durch $\mathrm{trs}^{\mathrm{L}}$ und $\mathrm{trs}^{\mathrm{D}}$ gegebenen Kompositionsform nachgewiesen wurde, genügt es sogar,

$$\begin{aligned} \mathrm{trs}^{\mathrm{E}}(z,e) &= \mathrm{trs}^{\mathrm{L}}(z,(e,\{e\},e)) \\ \mathrm{trs}^{\mathrm{V}}(z,v) &= \mathrm{trs}^{\mathrm{L}}(z,(\{v\},\emptyset,\emptyset)) \end{aligned} \tag{4.13}$$

für alle auftretenden Zustände z zu zeigen.

Bemerkung 4.6. Wir kommen noch einmal auf Bemerkung 4.5 zurück und nutzen die besondere Konstruktion der Kontaktzerlegung $(T_1,\ldots,T_t)$ mit $T_i=(V_i,E_i,K_i)$ zur Einschränkung derjenigen Mengen A_1 aus, für die Lemma 4.3 gezeigt werden muss. In einem zu einer Kante e gehörenden Transformationsschritt $\mathrm{trs}^{\mathrm{E}}$ sind alle zu e inzidenten Knoten (und damit alle Knoten des Kontaktgraphen) gemäß Konstruktion der speziellen Kontaktzerlegung Kontaktknoten. Es genügt also, Lemma 4.3 nur für solche A_1 zu zeigen, für die $V_2\subseteq A_1$ gilt. Aus dem gleichen Grund kann man bei der Behandlung von $\mathrm{trs}^{\mathrm{V}}$ vereinfachend $V_2\subseteq A_1+v$ annehmen, wobei v der in diesem Schritt zu aktivierende Knoten ist.

Bemerkung 4.7. Im Falle von Kantenkompositionsalgorithmen gibt es drei Szenarien für das globale Verhalten der Mächtigkeit der berechneten Zustandsmengen:

Typ A: $c_{\mathrm{V}}=1$, $c_{\mathrm{E}}>1$: Die Zahl der Zustände bleibt im Knotenaktivierungsschritt konstant und wächst im Kantenschritt.

Typ B: $c_{\mathrm{V}}>1$, $c_{\mathrm{E}}=1$: Die Zahl der Zustände wächst im Knotenaktivierungsschritt und bleibt konstant oder fällt im Kantenschritt.

Typ C: $c_{\mathrm{V}}>1$, $c_{\mathrm{E}}>1$: Die Zahl der Zustände wächst sowohl im Knotenaktivierungsschritt als auch im Kantenschritt.

In allen drei Fällen sinkt die Zahl der Zustände im Knotendeaktivierungsschritt. Das Anwachsen und Schrumpfen der Zustandsmenge gilt dabei für den allgemeinen Fall. In Ausnahmesituationen kann es auch vorkommen, dass zum Beispiel die Zahl der Zustände in einem Typ-A-Kantenschritt konstant bleibt.

Besonders für Algorithmen des Typs B bietet es sich an, den Knotenaktivierungsschritt direkt mit dem folgenden Kantenschritt zu verbinden und damit das kurzfristige Wachsen und Schrumpfen der Zustandsmenge zu vermeiden. Für Probleme vom Typ A kann es dagegen sinnvoll sein, Kantenschritte, die direkt vor einer Knotendeaktivierung liegen, mit diesem Deaktivierungsschritt zu einer Einheit zusammenzufassen.

Bemerkung 4.8. Die bei der Implementation von Problemen mit Hilfe der Abschlussmethode aus Bemerkung 2.22 auftretenden unzulässigen Konstellationen werden bei Kantenkompositionsalgorithmen in der Regel entweder im Kantenschritt oder im Knotendeaktivierungsschritt als unzulässig erkannt. Da die Kantenschritte vor den entsprechenden Knotendeaktivierungsschritten durchgeführt werden, sollte zur angesprochenen Vermeidung unnötigen Rechenaufwandes versucht werden, die Erkennung der unzulässigen Konstellationen bereits im Kantenschritt durchzuführen.

Ob das möglich ist, lässt sich häufig an der Formulierung des Problems erkennen. Werden zulässige Konstellationen dadurch identifiziert, dass eine bestimmte Eigenschaft *für alle Nachbarkanten* aller Knoten gilt, so kann eine Konstellation schon im Kantenschritt als nicht zulässig erkannt werden. Probleme dieser Art werden *positive Abschlussprobleme* genannt. Ein Beispiel dafür ist das klassische Knotenfärbungsproblem: Eine Knotenfärbung ist zulässig, wenn *für alle Kanten* gilt, dass die durch diese Kante verbundenen Knoten verschieden gefärbt sind. Diese Forderung an alle Kanten ist natürlich gleichzeitig eine Forderung an alle Nachbarkanten eines bestimmten Knotens.

Wenn dagegen eine zulässige Konstellation dadurch identifiziert wird, dass eine bestimmte Eigenschaft *für mindestens eine Nachbarkante* aller Knoten gilt (ein Knoten wird zum Beispiel durch eine Menge von Knoten dominiert, wenn er in dieser Menge ist oder durch *mindestens eine Kante* mit einem Knoten dieser Menge verbunden ist), so kann die Identifizierung der unzulässigen Konstellationen oft erst im Knotendeaktivierungsschritt stattfinden, da das der Zeitpunkt ist, zu dem bekannt wird, dass der zu deaktivierende Knoten keine weiteren unbearbeiteten Nachbarkanten hat. Diese Art von Problem wird als *negatives Abschlussproblem* bezeichnet.

4.3 Knotenkomposition

Algorithmus

Der zweite wichtige Spezialfall der Komposition ist die *Knotenkomposition*, die sich ähnlich wie die Kantenkomposition gegenüber der Grundform der Komposition durch zusätzliche Strukturforderungen an die verwendete Kontaktzerlegung auszeichnet: Knotenkompositionsalgorithmen benötigen Kontaktzerlegungen $(T_1, \ldots, T_t)$ der Form (2.3).

Die Kontaktgraphen bestehen in diesem Fall lediglich aus einem Nichtkontaktknoten, einigen Kontaktknoten und entsprechend vielen Kanten. Die Funktion $\mathrm{trs}^{\mathrm{L}}$ ist damit durch eine Funktion $\mathrm{trs}^{\mathrm{K}}$ bestimmt, wobei gilt

$$\mathrm{trs}^{\mathrm{L}}(z, T_i) = \mathrm{trs}^{\mathrm{K}}(z, v_{k_i}, \{e_{l_1}, \ldots, e_{l_p}\}).$$

Damit kann ein *Knotenkompositionsalgorithmus* wie folgt implementiert werden:

```
Funktion Knotenschritt(Zustandsmenge 𝒵, Knoten v, Kantenmenge F)
  𝒵' := ∅
  Für alle z in 𝒵:
    Setze 𝒵' := 𝒵' ⊎ trs^K(z, v, F)
  Rückgabe 𝒵'

Algorithmus Knotenkomposition(Hypergraph H = (V, E))
  Wähle eine (beliebige) Permutation π = (v_{k_1}, ..., v_{k_n}) von V
  Bestimme aus π eine Knotenzerlegung (T_1, ..., T_{|V|}) von H
  Bestimme 𝒵_0 = 𝒵_{(∅,∅),∅}
  Für alle i von 1 bis |V|:
    𝒵 := Knotenschritt(𝒵_{i-1}, v_{k_i}, E_i)
```

$\mathcal{Z}_i :=$ `Deaktivierung(`$\mathcal{Z}$`,` D_i`)`
`Rückgabe Projektion(`$\mathcal{Z}_{|V|}$`)`

Die Funktionen `Deaktivierung` und `Projektion` werden dabei unverändert vom modifizierten Grundalgorithmus (vgl. Seite 43) übernommen.

Korrektheit

Die Funktionen $\mathrm{trs}^{\mathrm{K}}$ und $\mathrm{trs}^{\mathrm{D}}$ beschreiben trs gemäß (4.9) vollständig, so dass im Zusammenhang mit der Knotenkomposition auch das Paar $(\mathrm{trs}^{\mathrm{K}}, \mathrm{trs}^{\mathrm{D}})$ anstelle von trs verwendet werden kann.

Auch hier genügt es in Anlehnung an den allgemeinen Fall, für

$$T_2 = \{\{v\} \cup \bigcup_{e \in F} e, F, \bigcup_{e \in F} e - v\}$$

die Beziehungen

$$\begin{aligned} \mathrm{trz}^{\mathrm{D}}((\mathrm{ind}(C, A), \mathrm{val}(C)), v) &= (\mathrm{ind}(C, A - v), \mathrm{val}(C)) \\ \mathrm{trs}^{\mathrm{K}}((\mathrm{ind}(C_1, A_1), \mathrm{val}(C_1)), v, F) &= \biguplus_{C_2 \in \mathcal{C}_{T_2}} (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2), \mathrm{val}(C_1 \cup C_2)), \end{aligned} \tag{4.14}$$

sowie die üblichen „technischen“ Voraussetzungen

$$\begin{aligned} \mathrm{trz}^{\mathrm{D}}(z_1 \uplus z_2, C) &= \mathrm{trz}^{\mathrm{D}}(z_1, C) \uplus \mathrm{trz}^{\mathrm{D}}(z_2, C) \\ \mathrm{trz}^{\mathrm{D}}(z_1 \uplus z_2, C_2) &= \mathrm{trz}^{\mathrm{D}}(z_1, C_2) \uplus \mathrm{trz}^{\mathrm{D}}(z_2, C_2) \\ \mathrm{trz}^{\mathrm{K}}((i, w_1 \oplus w_2), C_2) &= \mathrm{trz}^{\mathrm{K}}((i, w_1), C_2) \uplus \mathrm{trz}^{\mathrm{K}}((i, w_2), C_2) \end{aligned}$$

zu überprüfen, um die Korrektheit der Kompositionsform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{ind}, \mathrm{prs}, (\mathrm{trs}^{\mathrm{K}}, \mathrm{trs}^{\mathrm{D}}))$ nachzuweisen, die als *Knotenkompositionsform* bezeichnet wird.

Falls bereits die Gültigkeit einer durch $\mathrm{trs}^{\mathrm{L}}$ und $\mathrm{trs}^{\mathrm{D}}$ gegebenen Kompositionsform nachgewiesen wurde, genügt der Nachweis von

$$\mathrm{trs}^{\mathrm{K}}((\mathrm{ind}(C_1, A_1), \mathrm{val}(C_1)), v, F) = \mathrm{trs}^{\mathrm{L}}((\mathrm{ind}(C_1, A_1), \mathrm{val}(C_1)), T_2)$$

für alle $C_1 \in \mathcal{C}_1$.

Bemerkung 4.9. Knotenkompositionsalgorithmen sind vom Aufbau her einfacher als Kantenkompositionsalgorithmen, da es nur einen statt zwei Typen von Schritten gibt. Der „große“ Schritt bei der Behandlung eines Kontaktgraphen der Knotenkomposition (die Kontaktgraphen sind komplette Sterne) ist allerdings komplexer als die Schritte eines Kantenkompositionsalgorithmus, die jeweils nur einzelne Knoten oder Kanten verarbeiten. Die Frage, ob sich die Verwendung von Knotenkompositionsalgorithmen praktisch lohnt, kann nur für konkrete Problemstellungen beantwortet werden. Als Faustregel mag gelten, dass Knotenkompositionsalgorithmen oft günstig sind, wenn $c_{\mathrm{E}} = 1$, die einzelnen Kanten also keine „Individualität“ haben.

5 Anwendungen

Dieses Kapitel stellt einige Anwendungen der Kompositionsmethode vor. Am Anfang wird dabei eine Klasse von Aufgabenstellungen angesprochen, die durch einen gemeinsamen Algorithmus gelöst werden kann, danach folgt eine Reihe weniger allgemeiner Einzelfälle.

Obwohl das Kapitel recht umfangreich ist, werden nicht alle bislang bekannten Anwendungen detailliert beschrieben. Es soll vielmehr als Fundgrube für Tricks und Kniffe dienen und Beispiele zur Lösung von Routineproblemen wie dem Korrektheitsnachweis liefern.

Die Algorithmen werden in einem Schema beschrieben, das es erlaubt, schnell „Standardinformationen“ aus der Beschreibung zu extrahieren. Dafür werden die Ausführungen zu einer Anwendung wie folgt gegliedert:

Problem In einem solchen Abschnitt findet sich eine kurze Beschreibung der Aufgabenstellung sowie Verweise auf Literaturstellen. Besondere Eigenschaften wie zum Beispiel bekannte Dekompositionsformeln werden ebenfalls angeführt.

Indexform Dieser und die folgenden Abschnitte dienen dazu, das Problem in den vorgestellten Rahmen der Kompositionsalgorithmen einzupassen. Es wird eine formale Beschreibung der jeweiligen Graphenkenngröße in Indexform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ angeben und kurz deren Gültigkeit begründet. In einigen Fällen finden sich auch Hinweise, wieso gerade diese formale Beschreibung gewählt wurde.

Splitting, Komposition In derartig gekennzeichneten Abschnitten werden die entsprechenden formalen Beschreibungen angegeben bzw. kurz hergeleitet. Manchmal werden auch die Spezialformen für einen Kanten- bzw. Knotenkompositionsalgorithmus angegeben. Diese entstammen realen Implementierungen und enthalten unter Umständen bereits bestimmte Optimierungen, sind also nicht immer äquivalent zur allgemeinen Kompositionsform. In solchen Fällen wird auf die Optimierung besonders hingewiesen.

Implizit ist immer $H = (V, E)$ der gerade zu berechnende Hypergraph und A die aktuelle Zielmenge der Indizierung. Bei der Notation wurde aus Gründen der Übersichtlichkeit häufig auf aus dem Kontext ableitbare Details verzichtet. Falls nichts Gegenteiliges erwähnt wird, gelten beim Splitting die Bezeichnungen aus Lemma 3.7 und bei der Komposition diejenigen aus Lemma 4.3.

Die einzelnen Abschnitte innerhalb eines Beispiels bauen aufeinander auf. Dabei verwenden die Splitting- und Kompositionsformen die jeweils zuvor beschriebenen Indexformen, die wiederum die entsprechenden Grundformen erweitern.

Einige Übersichten mit konkreten Rechenergebnissen finden sich im Anhang.

5.1 Kantendekompositionsformeln

Problem

Für einige Graphenkenngrößen sind *Dekompositionsformeln* bekannt, die die in einem (gewöhnlichen) Graphen oder Hypergraphen zu berechnende Größe in eine Beziehung zur gleichen Größe in einer gewissen Anzahl von Minoren des Ausgangsgraphen setzt. In diesem Abschnitt sollen Kompositionsalgorithmen für den Fall bestimmt werden, dass es sich um eine lineare Beziehung und genau zwei Minoren handelt, die durch Kontraktion bzw. Löschen einer Kante entstehen. Die zu berechnende Größe soll also eine Beziehung der Art

$$\begin{aligned} R(H) &= a(e)\,R(H_e) + b(e)\,R(H_{-e}) \\ R(\overline{K}_n) &= r(n) \end{aligned} \tag{5.1}$$

erfüllen. Ein Beispiel für eine Graphenkenngröße, für die eine derartige *Kantendekompositionsformel* bekannt ist, ist die Zusammenhangswahrscheinlichkeit eines stochastischen Graphen, für die gilt $Z(G) = p_e\,Z(G_e) + (1-p_e)\,Z(G_{-e})$ und $Z(\overline{K}_n) = [n \leq 1]$. Weitere Beispiele liefern das von Negami in [Neg87] eingeführte Polynom für gewöhnliche Graphen mit der Kantendekompositionsformel $N(G) = x\,N(G_{-e}) + y\,N(G_e)$, $N(\overline{K}_n) = t^n$ und als Spezialfall davon das chromatische Polynom mit $\chi(G) = \chi(G_{-e}) - \chi(G_e)$, $\chi(\overline{K}_n) = x^n$.

Vereinfachend sei angenommen, dass die Menge

$$M = \bigcup_{H=(V,E)\in\mathcal{H}} \{R(H)\} \cup \{a(e) : e \in E\} \cup \{b(e) : e \in E\}$$

mit den Operationen $+$ und $\cdot$ bereits einen kommutativen Ring bildet bzw. durch Hinzunahme weiterer Elemente zu einem solchen, $\mathcal{M}$ genannten Ring erweitert werden kann.

Zur übersichtlicheren Darstellung sei die folgende Abkürzung eingeführt:

$$t(F,E) = \prod_{e\in F} a(e) \prod_{e\in E-F} b(e)$$

Indexform

Lemma 5.1. *Falls für eine Graphenkenngröße R die Beziehung (5.1) erfüllt ist, so gilt für alle $H = (V,E) \in \mathcal{H}$:*

$$R(H) = \sum_{F\subseteq E} \left(\prod_{e\in F} a(e) \prod_{e\in E-F} b(e)\,r(k(H{:}F)) \right) \tag{5.2}$$

Beweis.

1. Die direkte Umsetzung der Dekompositionsformel (5.1) führt wie bereits im Abschnitt 3.2 beschrieben zu einem Dekompositionsbaum, der im vorliegenden Fall $2^{|E|}$ Blätter hat. Zwischen diesen Blättern und der Menge 2^E besteht eine Bijektion wie folgt: Sei $F \subseteq E$ gegeben. Das zugeordnete Blatt findet man im Dekompositionsbaum, indem man bei der Wurzel startet und bei der Wahl eines Pivotelementes e mit $e \in F$ in den

„Kontraktionsast“ und bei $e \notin F$ in den „Löschast“ des Dekompositionsbaumes absteigt. Umgekehrt gelangt man von einem Blatt zu einer Teilmenge $F \subseteq E$, indem man mit einer leeren Menge F startend vom Blatt aus in Richtung Wurzel aufsteigt und all diejenigen Pivotelemente zu F hinzufügt, in deren „Kontraktionsast“ man sich befindet. Damit kann eine Summe über alle Blätter des Dekompositionsbaumes direkt als Summe über alle Teilmengen von E geschrieben werden.

2. Sei $F \subseteq E$ gegeben. Durch Kontraktion aller Kanten aus F und Löschen der Kanten $E - F$ entsteht aus H ein kantenloser Graph $\overline{K}_d$, dessen Knotenzahl d gleich $k(H{:}F)$ ist. Das Gewicht, mit dem das F entsprechende Blatt des Dekompositionsbaumes durch die Dekompositionsformel (5.1) in $R(H)$ eingeht, entspricht $r(k(H{:}F))$ multipliziert mit $a(e^{'})$ für alle kontrahierten Kanten $e^{'} \in F$ und $b(e^{''})$ für alle gelöschten Kanten $e^{''} \in E - F$.

3. Aus 1. und 2. folgt schließlich die Gültigkeit von (5.2). □

Bemerkung 5.2. Wir stellen fest, dass die ursprüngliche Aufgabe gelöst ist, wenn es gelingt, ein Polynom $R^{'}(H) = p_H(x) \in \mathcal{M}[x]$ so zu bestimmen, dass mit $n = |V|$ gilt:

$$R^{'}(H) \;=\; p_H(x) \;=\; \sum_{F \subseteq E} t(F,E)\, x^i \;=\; \sum_{i=0}^{n} \left(\sum_{\substack{F \subseteq E \\ k(H:F)=i}} t(F,E) \right) x^i \tag{5.3}$$

Dann gilt nämlich mit Hilfe der Funktion $\mathrm{proj} : \mathcal{M}[x] \to \mathcal{M}$, $\sum_{i=0}^{n} p_i\, x^i \mapsto \sum_{i=0}^{n} p_i\, r(i)$:

$$\begin{aligned}
\mathrm{proj}(R^{'}(H)) &= \mathrm{proj}(p_H(x)) \\
&= \mathrm{proj} \left(\sum_{i=0}^{n} \sum_{\substack{F \subseteq E \\ k(H:F)=i}} t(F,E)\, x^{k(H:F)} \right) \\
&= \sum_{i=0}^{n} \sum_{\substack{F \subseteq E \\ k(H:F)=i}} t(F,E)\, r(k(H{:}F)) \\
&= \sum_{F \subseteq E} t(F,E)\, r(k(H{:}F)) \\
&= \sum_{F \subseteq E} \prod_{e \in F} a(e) \prod_{e \in E-F} b(e)\, r(k(H{:}F)) \\
&= R(H)
\end{aligned}$$

Ziel der weiteren Abhandlung in diesem Abschnitt wird es sein, diese modifizierte Aufgabenstellung zu lösen und die Graphenkenngröße $R^{'}(H)$ zu bestimmen.

Lemma 5.3. *Sei $A \subseteq V$ eine zunächst beliebige Menge von Knoten eines Hypergraphen $H = (V,E)$. Das Tupel $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ mit*

$$c_{\mathrm{V}} \;=\; 1$$

$$\begin{aligned} c_{\mathrm{E}} &= 2 \\ \mathcal{W} &= \mathcal{M}[x] \\ O &= 0 \\ \oplus &: (w_1, w_2) \mapsto w_1 + w_2 \quad \forall w_1, w_2 \in \mathcal{W} \\ \mathrm{val} &: C \mapsto t(\hat{C}, E)\, x^{k(H:\hat{C})} \\ \mathrm{ind} &: (C, A) \mapsto \mathrm{part}(H{:}\hat{C}, A) \end{aligned} \tag{5.4}$$

stellt eine Indexform für das Problem der Bestimmung von R' dar. Die Definitionsbereiche der Funktionen entsprechen hier (und in vergleichbaren Fällen in den kommenden Abschnitten) den im Abschnitt 2.5 angegebenen. $\hat{C}$ bezeichnet die durch die Konstellation C nach Bemerkung 2.14 bestimmte Kantenmenge $\hat{C} = \{e : C(e) = 1\}$.

Beweis. Durch Überprüfen der Definition 2.23:

1. $(\mathcal{W}, \oplus)$ ist ein kommutatives Monoid, da $(\mathcal{M}, +, \cdot)$ ein Ring ist.

2. Sei $H = (V, E) \in \mathcal{H}$ beliebig. Die $(1,2)$-Konstellationen $C \in \mathcal{C}_H(c_{\mathrm{V}}, c_{\mathrm{E}})$ werden wieder mit Teilmengen $\hat{C}$ der Kantenmenge E identifiziert. Es gilt also insbesondere:

$$\begin{aligned} \bigoplus_{C \in \mathcal{C}_H(c_{\mathrm{V}}, c_{\mathrm{E}})} \mathrm{val}(C) &= \sum_{C \in \mathcal{C}_H(1,2)} \mathrm{val}(C) \\ &= \sum_{\hat{C} \subseteq E} t(\hat{C}, E)\, x^{k(H:\hat{C})} \\ &\overset{(5.3)}{=} R'(H) \end{aligned}$$

□

Splitting

Lemma 5.4. *Seien $H_1 = (V_1, E_1)$ und $H_2 = (V_2, E_2)$ Unterhypergraphen von $H = (V, E) \in \mathcal{H}$ mit $H_1 \boxplus_K H_2 = H$. Sei C eine $(1,2)$-Konstellation von H, die wie üblich mit einer Kantenmenge $\hat{C} \subseteq E$ identifiziert wird, und bezeichne $\hat{C}_1 = \hat{C} \cap E_1$ und $\hat{C}_2 = \hat{C} \cap E_2$. Dann gilt für die Zahl der Zusammenhangskomponenten*

$$\begin{aligned} k(H{:}\hat{C}) = {} & k(H_1{:}\hat{C}_1) - |\mathrm{part}(H_1{:}\hat{C}_1, K)| \\ & + k(H_2{:}\hat{C}_2) - |\mathrm{part}(H_2{:}\hat{C}_2, K)| \\ & + |\mathrm{part}(H_1{:}\hat{C}_1, K) \vee \mathrm{part}(H_2{:}\hat{C}_2, K)|. \end{aligned} \tag{5.5}$$

Beweis. Die Komponenten von $H{:}\hat{C}$ fallen in drei Klassen: (a) Komponenten von $H_1{:}\hat{C}_1$, die keine Knoten aus K enthalten, (b) Komponenten von $H_2{:}\hat{C}_2$, die keine Knoten aus K enthalten, und schließlich (c) Komponenten, die Knoten aus K enthalten. In der Klasse (a) sind $k(H_1{:}\hat{C}_1) - |\mathrm{part}(H_1{:}\hat{C}_1, K)|$ Elemente enthalten, in (b) $k(H_2{:}\hat{C}_2) - |\mathrm{part}(H_2{:}\hat{C}_2, K)|$ und Klasse (c) hat $|\mathrm{part}(H_1{:}\hat{C}_1, K) \vee \mathrm{part}(H_2{:}\hat{C}_2, K)|$ Elemente. □

Satz 5.5. *Sei $H_1 \boxplus_K H_2$ eine Zerlegung eines Hypergraphen H an einer Knotenmenge K, $C_K \in \mathcal{C}_K = C_{(K,\emptyset)}(1,2)$ und $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ die Indexform aus Lemma 5.3. Mit den üblichen Bezeichnungen, insbesondere $i_1 \vee i_2$ für die Verschmelzung zweier Partitionen $i_1, i_2 \in \mathcal{P}(K)$ und $|i|$ für die Zahl der Blöcke der Partition $i \in \mathcal{P}(K)$, sei* glue *die Funktion*

$$\mathrm{glue} : (C_K, (i_1, w_1), (i_2, w_2)) \mapsto w_1\, w_2\, x^{|i_1 \vee i_2| - |i_1| - |i_2|}. \tag{5.6}$$

Dann ist $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{glue})$ eine Splittingform für das Problem (5.3).

Beweis. Es gelten wieder die Abkürzungen (3.1). Seien $i_1 \in \mathcal{I}_1$ und $i_2 \in \mathcal{I}_2$ fixiert. Da $c_\mathrm{V} = 1$ ist, gilt $|\mathcal{C}_K| = 1$ und $C_1|_K = C_K$ bzw. $C_2|_K = C_K$ für alle $C_1 \in \mathcal{C}_{H_1}$ bzw. $C_2 \in \mathcal{C}_{H_2}$ und das einzige Element $C_K \in \mathcal{C}_K$. Es gilt

$$\begin{aligned}
&\mathrm{glue}(C_K, (\mathrm{ind}(C_1, K), \mathrm{val}(C_1)), (\mathrm{ind}(C_2, K), \mathrm{val}(C_2))) \\
&\quad\overset{(5.6)}{=}\; t(\hat{C}_1, E_1)\, x^{k(H_1:\hat{C}_1)}\, t(\hat{C}_2, E_2)\, x^{k(H_2:\hat{C}_2)} \\
&\qquad\qquad \cdot\, x^{|\mathrm{part}(H_1:\hat{C}_1,K) \vee \mathrm{part}(H_2:\hat{C}_2,K)| - |\mathrm{part}(H_1:\hat{C}_1,K)| - |\mathrm{part}(H_2:\hat{C}_2,K)|} \\
&\quad=\; t(\hat{C}_1 \cup \hat{C}_2, E_1 \cup E_2) \\
&\qquad\qquad \cdot\, x^{k(H_1:\hat{C}_1) + k(H_2:\hat{C}_2) + |\mathrm{part}(H_1:\hat{C}_1,K) \vee \mathrm{part}(H_2:\hat{C}_2,K)| - |\mathrm{part}(H_1:\hat{C}_1,K)| - |\mathrm{part}(H_2:\hat{C}_2,K)|} \\
&\quad\overset{(5.5)}{=}\; t(\hat{C}_1 \cup \hat{C}_2, E_1 \cup E_2)\, x^{k(H:(\hat{C}_1 \cup \hat{C}_2))} \\
&\quad=\; \mathrm{val}(C_1 \cup C_2).
\end{aligned}$$

Von der Gültigkeit von (3.4) und (3.5) überzeugt man sich leicht, so dass aus Lemma 3.7 die Behauptung folgt. □

Bemerkung 5.6. Die Rechnung im vorstehenden Beweis demonstriert sehr schön, wie „passende“ glue Funktionen mit mäßigem manuellen Aufwand aus einer gegebenen Indexform gewonnen werden können. Die Konstruktion erfolgt dabei „von unten nach oben“: Ausgehend von der rechten Seite von (3.3) wird zunächst der Wert der Funktion val durch seine aus der Grundform bekannte Definition ersetzt. Dann wird versucht, diesen Wert so zu zerlegen, dass er in zwei Teile zerfällt, die von gleicher Struktur sind und direkt den beiden Graphenteilen zugeordnet werden können. Eventuell verbleibende „Reste“ (hier der Wert $x^{|\mathrm{part}(H_1:\hat{C}_1,K) \vee \mathrm{part}(H_2:\hat{C}_2,K)| - |\mathrm{part}(H_1:\hat{C}_1,K)| - |\mathrm{part}(H_2:\hat{C}_2,K)|}$) dürfen nur von den Indizes, K und C_K, nicht aber explizit von Knoten aus $V - K$ oder Kanten abhängen. Gelingt dies, hat man eine Splittingformel gefunden. Gelingt dies nicht, so ist häufig ersichtlich, woran die Zerlegung scheitert. Wenn zum Beispiel der „Rest“ „zu groß“ ist, muss meist mehr Information in den Indizes untergebracht werden, also eine feinere Indizierung gewählt werden usw.

Reduktion

Wir beschränken uns in diesem Abschnitt über Reduktionen auf gewöhnliche Graphen, erlauben aber ausnahmsweise wieder parallele Kanten. Die Kantenmenge E eines Graphen $G = (V, E)$ ist also eine Multimenge, später im Abschnitt sind auch wieder Schlingen zulässig.

Lemma 5.7. *Seien $G_1 = (\{v_1, v_2\}, \{e, f\})$ und $G_1' = (\{v_1, v_2\}, \{g\})$ zwei Graphen mit parallelen Kanten e, f zwischen v_1 und v_2 in G_1 und $g = \{v_1, v_2\}$ in G_1' und Kantenbewertungen $a(e)$, $b(e)$, $a(f)$ und $b(f)$ für die Kanten von G_1. Mit der Wahl*

$$\begin{aligned} a(g) &= a(e)\,a(f) + a(e)\,b(f) + b(e)\,a(f) \\ b(g) &= b(e)\,b(f), \end{aligned} \tag{5.7}$$

der Kantenbewertungen von G_1' bildet das Paar (G_1, G_1') eine Reduktion.

Beweis. Seien $G_2 = (V_2, E_2) \in \mathcal{G}$ mit $V_2 \cap \{v_1, v_2\} = \{v_1, v_2\} \subseteq V_2$, $E_2 \cap \{e, f, g\} = \emptyset$ und $C_2 \in \mathcal{C}_{G_2}$ beliebig. Bezeichne $K = \{v_1, v_2\}$, $G = G_1 \boxplus_K G_2$ und $G' = G_1' \boxplus_K G_2$. Dann gilt für alle $\hat{C}_2 \in \mathcal{C}_{G_2}$

$$k(G{:}(\hat{C}_2 + e + f)) \;=\; k(G{:}(\hat{C}_2 + e)) \;=\; k(G{:}(\hat{C}_2 + f)) \;=\; k(G'{:}(\hat{C}_2 + g)) \tag{5.8}$$

und

$$k(G{:}\hat{C}_2) \;=\; k(G'{:}\hat{C}_2). \tag{5.9}$$

Da $c_{\mathrm{V}} = 1$ ist, besteht $\mathcal{C}_K$ nur aus einem einzigen Element C_K und es gilt

$$\begin{aligned} &\bigoplus_{C_1 \in \mathcal{C}_{G_1}} \mathrm{val}(C_1 \cup C_2) \\ &= \quad \mathrm{val}(\{(e,1),(f,1)\} \cup C_2) \;\oplus\; \mathrm{val}(\{(e,1),(f,2)\} \cup C_2) \;\oplus \\ &\qquad \mathrm{val}(\{(e,2),(f,1)\} \cup C_2) \;\oplus\; \mathrm{val}(\{(e,2),(f,2)\} \cup C_2) \\ &= \quad a(e)\,a(f)\,t(\hat{C}_2, E_2)\,x^{k(G:(\hat{C}_2+e+f))} \;+\; a(e)\,b(f)\,t(\hat{C}_2, E_2)\,x^{k(G:(\hat{C}_2+e))} \;+ \\ &\qquad b(e)\,a(f)\,t(\hat{C}_2, E_2)\,x^{k(G:(\hat{C}_2+f))} \;+\; b(e)\,b(f)\,t(\hat{C}_2, E_2)\,x^{k(G:\hat{C}_2)} \\ &\overset{(5.8),(5.9)}{=} \quad (a(e)\,a(f) + a(e)\,b(f) + b(e)\,a(f))\,t(\hat{C}_2, E_2)\,x^{k(G':(\hat{C}_2+g))} \;+ \\ &\qquad\qquad b(e)\,b(f)\,t(\hat{C}_2, E_2)\,x^{k(G':\hat{C}_2)} \\ &\overset{(5.7)}{=} \quad a(g)\,t(\hat{C}_2, E_2)\,x^{k(G':(\hat{C}_2+g))} \;+\; b(g)\,t(\hat{C}_2, E_2)\,x^{k(G':\hat{C}_2)} \\ &= \quad \mathrm{val}(\{(g,1)\} \cup C_2) \;\oplus\; \mathrm{val}(\{(g,2)\} \cup C_2) \\ &= \quad \bigoplus_{C_1' \in \mathcal{C}_{G_1'}} \mathrm{val}(C_1' \cup C_2). \end{aligned}$$

Damit sind die Voraussetzungen der Definition 3.11 erfüllt. □

Eine weitere Reduktion wird in folgendem Lemma beschrieben. Dazu seien wieder *Schlingen* erlaubt, das heißt, die Kanten eines Graphen dürfen die Form (vv) haben.

Lemma 5.8. *Seien $G_1 = (\{v\}, \{e\})$ und $G_1' = (\{v\}, \emptyset)$ zwei Graphen mit $e = \{v, v\}$ und Kantenbewertungen $a(e)$ und $b(e)$. Dann bildet das Tripel $(G_1, G_1' r)$ mit $r : \mathcal{W} \to \mathcal{W}$, $r(w) = (a(e) + b(e)) \cdot w$ eine Reduktion.*

Beweis. Seien $G_2 = (V_2, E_2) \in \mathcal{G}$ mit $v \in V_2$, $e \notin E_2$ und $C_2 \in \mathcal{C}_{G_2}$ beliebig. Bezeichne $K = \{v\}$, $G = G_1 \boxplus_K G_2$ und $G' = G'_1 \boxplus_K G_2$. Es gilt zunächst

$$k(G{:}(\hat{C}_2 + e)) \;=\; k(G{:}\hat{C}_2) \;=\; k(G'{:}\hat{C}_2).$$

Da $c_V = 1$ ist, besteht $\mathcal{C}_{(K,\emptyset)}$ nur aus einem einzigen Element $C_K = \{(v,1)\}$ und es gilt:

$$\begin{aligned}
\bigoplus_{C_1 \in \mathcal{C}_{G_1}} \mathrm{val}(C_1 \cup C_2) &= \mathrm{val}(\{(e,1)\} \cup C_2) \;\oplus\; \mathrm{val}(\{(e,2)\} \cup C_2) \\
&= a(e)\, t(\hat{C}_2, E_2)\, x^{k(G:(\hat{C}_2+e))} \;+\; b(e)\, t(\hat{C}_2, E_2)\, x^{k(G:\hat{C}_2)} \\
&= (a(e) + b(e))\, t(\hat{C}_2, E_2)\, x^{k(G':\hat{C}_2)} \\
&= (a(e) + b(e)) \bigoplus_{C'_1 \in \mathcal{C}_{G'_1}} \mathrm{val}(C'_1 \cup C_2)
\end{aligned}$$

Damit sind die Voraussetzungen der Definition 3.14 erfüllt. □

Bemerkung 5.9. Die Möglichkeit, in einer vergleichsweise großen Gruppe von Aufgabenstellungen Reduktionen der eben genannten Art (*Parallel-* und *Schlingenreduktionen)* anzuwenden und damit parallele Kanten bzw. Schlingen zu beseitigen, ist einer der Gründe, warum sich in der restlichen Arbeit auf schlichte Graphen beschränkt wird.

Komposition

Lemma 5.10. *Seien $H_1 = (V_1, E_1)$ und $T_2 = (V_2, E_2, K_2)$ die zwei Teile einer Kontaktzerlegung (H_1, T_2) eines Hypergraphen $H = (V, E) = (V_1 \cup V_2, E_1 \cup E_2)$. Sei $A_1 \subseteq V_1$ eine zunächst beliebige Menge. Gegeben seien weiterhin zwei Konstellationen $C_1 \in \mathcal{C}_{H_1}$ und $C_2 \in \mathcal{C}_{T_2}$. Bezeichne*

$$\mathrm{ex}(\mathrm{part}(H_1{:}\hat{C}_1, A_1), C_2) \;=\; \mathrm{part}(H_1{:}\hat{C}_1, A_1)_{+(V_2 - A_1)} \vee \mathrm{part}(T_2{:}\hat{C}_2, V_2)_{+(A_1 - V_2)}. \quad (5.10)$$

Dann gilt

$$\mathrm{ex}(\mathrm{part}(H_1{:}\hat{C}_1, A_1), C_2) \;=\; \mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2) \;\in \mathcal{P}(A_1 \cup V_2) \quad (5.11)$$

sowie

$$k((H_1{+}T_2){:}(\hat{C}_1 \cup \hat{C}_2)) \;=\; k(H_1{:}\hat{C}_1) - |\mathrm{part}(H_1{:}\hat{C}_1, A_1)| + |\mathrm{ind}(C_1{\cup}C_2, A_1{\cup}V_2)|. \quad (5.12)$$

Beweis.

1. Durch die Erweiterung um die Knoten $V_2 - A_1$ entsteht aus $\pi = \mathrm{part}(H_1{:}\hat{C}_1, A)$ die Partition $\pi_{+(V_2-A_1)} \in \mathcal{P}(A_1 \cup (V_2 - A_1)) = \mathcal{P}(A_1 \cup V_2)$ von $A_1 \cup V_2$. Sei $C' \in \mathcal{C}_{H_1+T_2}$ die Konstellation, für die gilt $C'(e) = C_1(e)$ für $e \in E_1$ und $C'(e) = 2$ sonst. Dann gilt $\mathrm{part}((H_1 + T_2){:}\hat{C}', A_1 \cup V_2) = \pi_{+(V_2-A_1)}$. Die Partition $\pi_{+(V_2-A_1)}$ ist also die induzierte Partition von C' mit der Zielmenge $A_1 \cup V_2$. Andererseits entsteht aus $\mathrm{part}(T_2{:}\hat{C}_2, V_2)$ durch Erweiterung um die Knoten aus $A_1 - V_2$ eine zweite Partition von $A_1 \cup V_2$, nämlich diejenige, die von der Konstellation C'' induziert wird, für die gilt $C''(e) = C_2(e)$ für $e \in E_2$ und $C''(e) = 2$ sonst. Schließlich ist die Verschmelzung $\pi_{+(V_2-A_1)} \vee \mathrm{part}(T_2{:}\hat{C}_2, V_2)_{+(A_1-V_2)}$ der von C' und C'' induzierten Partitionen gleich der induzierten Partition der Konstellation $C = C_1 \cup C_2 \in \mathcal{C}_{H_1+T_2}$.

2. Die Zahl der Komponenten ergibt sich als die Summe aus der Zahl $k(H_1{:}\hat{C}_1) - |\pi|$ derjenigen Komponenten, die keine Knoten aus $A_1 \cup V_2$ enthalten, und der Zahl derjenigen Komponenten, die Knoten aus $A_1 \cup V_2$ enthalten. □

Satz 5.11. *Es gelten die gleichen Bezeichnungen wie im vorangegangenen Lemma. Dann ergänzt die aus den Funktionen*

$$\begin{aligned} \mathrm{trz}^{\mathrm{L}} &: ((i,w),C_2) \mapsto (\mathrm{ex}(i,C_2), w\, t(\hat{C}_2,E_2)\, x^{|\mathrm{ex}(i,C_2)|-|i|}) \\ \mathrm{trz}^{\mathrm{D}} &: ((i,w),v) \mapsto (i_{-v},w) \end{aligned} \tag{5.13}$$

gemäß (4.9) konstruierte Funktion trs *die Indexform* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ *aus Lemma 5.3 zu einer Kompositionsform.*

Beweis. Seien $C_1 \in \mathcal{C}_{H_1}$, $C_2 \in \mathcal{C}_{T_2}$ beliebige Konstellationen, $C = C_1 \cup C_2$ und $A_1 \subseteq V_1$ eine fixierte Knotenmenge. Es gilt

$$\begin{aligned} &\mathrm{trz}^{\mathrm{L}}((\mathrm{ind}(C_1,A_1),\mathrm{val}(C_1)),C_2) \\ &\quad\overset{(5.13)}{=}\ (\mathrm{ex}(\mathrm{ind}(C_1,A_1),C_2),\ \mathrm{val}(C_1)\, t(\hat{C}_2,E_2)\, x^{|\mathrm{ex}(\mathrm{ind}(C_1,A_1),C_2)|-|\mathrm{ind}(C_1,A_1)|}) \\ &\quad\overset{(5.4)}{=}\ (\mathrm{ex}(\mathrm{part}(H_1{:}\hat{C}_1,A_1),C_2), \\ &\qquad\qquad t(\hat{C}_1,E_1)\, t(\hat{C}_2,E_2)\, x^{k(H_1:\hat{C}_1)}\, x^{|\mathrm{ex}(\mathrm{part}(H_1:\hat{C}_1,A_1),C_2)|-|\mathrm{part}(H_1:\hat{C}_1,A_1)|}) \\ &\quad\overset{(5.11)}{=}\ (\mathrm{ind}(C,A_1\cup V_2),\ t(\hat{C},E_1\cup E_2)\, x^{k(H_1:\hat{C}_1)+|\mathrm{ind}(C,A_1\cup V_2)|-|\mathrm{part}(H_1:\hat{C}_1,A_1)|}) \\ &\quad\overset{(5.12)}{=}\ (\mathrm{ind}(C,A_1\cup V_2),\ t(\hat{C},E_1\cup E_2)\, x^{k((H_1+T_2):\hat{C})}) \\ &\quad\overset{(5.4)}{=}\ (\mathrm{ind}(C,A_1\cup V_2),\ \mathrm{val}(C)). \end{aligned}$$

Außerdem gilt für eine beliebige Menge $A \subseteq V$, einen beliebigen Knoten $v \in A$ und eine beliebige Konstellation $C \in \mathcal{C}$:

$$\begin{aligned} \mathrm{trs}^{\mathrm{D}}((\mathrm{ind}(C,A),\mathrm{val}(C)),v) &\overset{(5.4)}{=} \mathrm{trs}^{\mathrm{D}}((\mathrm{part}(H{:}\hat{C},A),\mathrm{val}(C)),v) \\ &\overset{(5.13)}{=} (\mathrm{part}(H{:}\hat{C},A)_{-v},\mathrm{val}(C)) \\ &= (\mathrm{part}(H{:}\hat{C},A-v),\mathrm{val}(C)) \\ &\overset{(5.4)}{=} (\mathrm{ind}(C,A-v),\mathrm{val}(C)) \end{aligned}$$

Damit gelten (4.5) und (4.7). Von der Gültigkeit der verbleibenden technischen Voraussetzungen von Lemma 3.7 überzeugt man sich leicht, so dass aufgrund von Satz 4.4 die gemäß (4.9) aus den hier angegebenen $\mathrm{trs}^{\mathrm{D}}$ und $\mathrm{trs}^{\mathrm{L}}$ konstruierte Funktion trs die Indexform aus Lemma 5.3 zu einer Kompositionsform ergänzt. □

Satz 5.12. *Die Kompositionsform (5.13) kann durch die Wahl von*

$$\begin{aligned} \mathrm{trs}^{\mathrm{V}} &: ((i,w),v) \mapsto (i_{+v}, w\,x) \\ \mathrm{trs}^{\mathrm{E}} &: ((i,w),e) \mapsto (i_e, w\,a(e)\,x^{|i_e|-|i|}) \ \uplus\ (i, w\,b(e)) \end{aligned} \tag{5.14}$$

zu einer Kantenkompositionsform vom Typ A spezialisiert werden.

Beweis.

1. Im Knotenaktivierungsschritt ist $\mathcal{C}_2 = \{\{(v,1)\}\}$ und damit $t(\hat{C}_2, E_2) = 1$ für das einzige Element $C_2 \in \mathcal{C}_2$. Der neue Knoten v ist isoliert in $H_1 + T_2$. Damit hat $\operatorname{ind}(C_1 \cup C_2, A_1 \cup V_2)$ einen Block (den Block $\{v\}$) mehr als $\operatorname{ind}(C_1, A_1)$ für beliebiges $C_1 \in \mathcal{C}_{H_1}$ und es gilt $\operatorname{part}(T_2{:}\hat{C}_2, V_2)_{+(A_1 - V_2)} = (\emptyset_{+v})_{+(A_1 - v)} = \emptyset_{+A_1+v}$. Folglich ist

$$\begin{aligned}
\operatorname{trs}^{\mathrm{L}}((i,w), T_2) &\overset{(4.10)}{=} \operatorname{trz}^{\mathrm{L}}((i,w), \{(v,1)\}) \\
&\overset{(5.13)}{=} (\operatorname{ex}(i, C_2),\ w\, t(\hat{C}_2, E_2)\, x^{|\operatorname{ex}(i,C_2)| - |i|}) \\
&\overset{(5.10)}{=} (i_{+v} \vee \emptyset_{+A_1+v},\ w\, 1\, x^{|i_{+v} \vee \emptyset_{+A_1+v}| - |i|}) \\
&= (i_{+v},\ w\, x^{|i_{+v}| - |i|}) \\
&\overset{(5.14)}{=} \operatorname{trs}^{\mathrm{V}}((i,w), v).
\end{aligned}$$

2. Im Kantenschritt ist $\mathcal{C}_2 = \{\{(e,1)\}, \{(e,2)\}\}$. Bezeichne $C_{21} = \{(e,1)\}$ und $C_{22} = \{(e,2)\}$. Dann gilt

   $$\operatorname{part}(T_2{:}\hat{C}_{21}, V_2)_{+(A_1 - V_2)} = \operatorname{part}(T_2{:}\{e\}, V_2)_{+(A_1 - V_2)} = \{e\}_{+(V_2 - e)_{+(A_1 - V_2)}} = \{e\}_{+(A_1 - e)}$$

   und

   $$\operatorname{part}(T_2{:}\hat{C}_{22}, V_2)_{+(A_1 - V_2)} = \operatorname{part}(T_2{:}\emptyset, V_2)_{+(A_1 - V_2)} = (\emptyset_{+V_2})_{+(A_1 - V_2)} = \emptyset_{+A_1}.$$

   Außerdem kann wegen Bemerkung 4.6 $V_2 \subseteq A_1$ und damit $V_2 - A_1 = \emptyset$ angenommen werden. Für beliebiges $i \in \mathcal{P}(A_1)$ gilt somit

   $$\operatorname{ex}(i, \hat{C}_{21}) = i \vee \operatorname{part}(T_2{:}\hat{C}_{21}, V_2)_{+(A_1 - V_2)} = i \vee \{e\}_{+(A_1 - e)} = i_e$$

   und

   $$\operatorname{ex}(i, \hat{C}_{22}) = i \vee \operatorname{part}(T_2{:}\hat{C}_{22}, V_2)_{+(A_1 - V_2)} = i \vee \emptyset_{+A_1} = i.$$

   Folglich ist

   $$\begin{aligned}
   \operatorname{trs}^{\mathrm{L}}((i,w), T_2) &\overset{(4.10)}{=} \operatorname{trz}^{\mathrm{L}}((i,w), C_{21}) \uplus \operatorname{trz}^{\mathrm{L}}((i,w), C_{22}) \\
   &\overset{(5.13)}{=} (\operatorname{ex}(i, C_{21}), w\, t(\hat{C}_{21}, E_2)\, x^{|\operatorname{ex}(i,C_{21})| - |i|}) \uplus \\
   &\qquad (\operatorname{ex}(i, C_{21}), w\, t(\hat{C}_{22}, E_2)\, x^{|\operatorname{ex}(i,C_{22})| - |i|}) \\
   &= (i_e, w\, a(e)\, x^{|i_e| - |i|}) \uplus (i, w\, b(e)\, x^{|i| - |i|}) \\
   &\overset{(5.14)}{=} \operatorname{trs}^{\mathrm{E}}((i,w), e).
   \end{aligned}$$

3. Aus 1. und 2. folgt wegen (4.13) die Behauptung. □

Komplexität

Satz 5.13. *Sei $H = (V, E)$ ein Hypergraph, R eine Graphenkenngröße mit einer Kantendekompositionsformel der Form (5.1) und eine Kantenkompositionsfolge für H mit einer Weite von höchstens d gegeben. Die Zeit für eine Operation (Vergleich, Kopie, Summe, ...) auf Indizes und Werten sei durch $\mathcal{O}(\omega)$ beschränkt. Dann existiert ein Kantenkompositionsalgorithmus zur Berechnung von $R(H)$ mit einer Komplexität von $\mathcal{O}(\omega\,(n+m))$.*

Beweis. Wir schätzen den Aufwand für den vorgestellten Kantenkompositionsalgorithmus nach oben ab. Für einen einzelnen Schritt gilt: Die Partitionen sind in jedem Fall auf die aktive Knotenmenge beschränkt, da Knoten erst bei ihrer Aktivierung in den Index aufgenommen und bei ihrer Deaktivierung wieder entfernt werden. Die Anzahl der verschiedenen Partitionen kann also nicht größer als die d-te Bell-Zahl $B(d)$ sein, die die Gesamtzahl der Mengenpartitionen einer d-elementigen Grundmenge angibt. Die Zahl d ist nach Voraussetzung fixiert, also unabhängig von n. Damit gibt es höchstens $B(d) = \mathcal{O}(1)$ verschiedene Indizes. Ein Transformationsschritt kann durch sequentielle Bearbeitung der Zustandsmenge und Zwischenspeichern aller Folgezustände in einer temporären Liste erfolgen, die nachfolgend durch Zusammenfassen von Zuständen mit gleichem Index und Entfernen von Zuständen mit Wert O normalisiert werden kann. Die temporäre Liste hat höchstens doppelt so viele Elemente wie die ursprüngliche Zustandsmenge, da die speziellen Transformationsfunktionen aus einem Zustand lediglich einen (Knotenaktivierung) oder zwei (Kantenschritt) Nachfolger produzieren. Die Normalisierung der Zustandsliste kann durch Sortieren der Liste mit maximal $\mathcal{O}(1)$ Elementen in $\mathcal{O}(\omega)$ Zeit und anschließendem Zusammenfassen von Zuständen mit gleichem Index in $\mathcal{O}(\omega)$ Zeit, insgesamt also in $\mathcal{O}(\omega)$ Zeit erfolgen. Zusammen mit der Zahl $2\,n+m$ der Transformationsschritte ergibt sich daraus eine obere Schranke $\mathcal{O}(\omega\,(n+m))$ für die Komplexität des gesamten Algorithmus. □

Bemerkung 5.14. Die vorstehende Aussage ist nur von theoretischem Interesse. Der tatsächliche Aufwand einer Rechnung verbirgt sich hinter der Anzahl $B(d)$ von Zuständen, die nur als $\mathcal{O}(1)$ in die Abschätzung eingeht. In realistischen Beispielen ($n \approx 100$, $d \approx 10$) ist $B(d)$ bereits um mehrere Zehnerpotenzen größer als n bzw. m und damit dominierend.

Bemerkung 5.15. Der vorgeschlagene Weg der Konstruktion von Kantenkompositionsalgorithmen aus Dekompositionsformeln ist sehr einfach zu implementieren, liefert aber nicht immer einen optimalen Algorithmus. Falls die Leistung unzureichend ist, können die konstruierten Algorithmen jedoch häufig als Grundlage für „handverbesserte“ Algorithmen dienen. In einigen Fällen lassen sich zum Beispiel die als Werte benutzten Polynome durch eine fixierte Anzahl von Koeffizienten ersetzen und dadurch Speicherplatz und Rechenzeit sparen.

5.2 Chromatische Invarianten

Definition 5.16. Eine *zulässige Färbung* eines Hypergraphen $H = (V, E)$ ist eine Funktion $\mathrm{col} : V \to \{1, \ldots, |V|\}$, so dass für alle Kanten $e \in E$ und für alle Knotenpaare $v_1, v_2 \in e$ gilt $\mathrm{col}(v_1) \neq \mathrm{col}(v_2)$. Die Elemente des Wertebereiches von col heißen dabei *Farben*. Eine *Minimalfärbung* ist eine zulässige Färbung des Hypergraphen mit der Eigenschaft, dass es keine zulässige Färbung von H mit weniger Farben gibt.

Eine Einschränkung dieser Definition auf die Menge der gewöhnlichen Graphen ist problemlos möglich und wird häufig verwendet. Da sich durch die Betrachtung dieses speziellen Problems keine algorithmischen Vorteile ergeben, soll hier das allgemeine Problem behandelt werden. Hauptziel ist dabei nicht die Erzeugung besonders effizienter Algorithmen sondern die Darstellung der Kompositionsmethode an wohlbekannten Beispielen.

Einige Berechnungen von chromatischen Invarianten wie dem chromatischen Polynom, der chromatischen Zahl oder aber die Suche nach einer Minimalfärbung können mit fast identischen Kompositionsalgorithmen gelöst werden. Die grundlegenden Ideen sollen hier dargestellt werden.

5.2.1 Minimalfärbung

Problem

Gesucht ist eine Färbung der Knoten eines Hypergraphen, so dass keine zwei benachbarten Knoten die gleiche Farbe erhalten und die Zahl der dabei verwendeten Farben minimal ist.

Indexform

Sei $H = (V, E)$ ein Hypergraph mit $n = |V|$ Knoten. Eine Färbung von H kann direkt als eine $(n, 1)$-Konstellation betrachtet werden. Dabei wird zur Vereinfachung angenommen, dass die Farben direkt mit den Labels $1, \ldots, n$ identifiziert werden. Ist C eine derartige Konstellation, so wird die Menge der benötigten Farben durch $\text{colours}(C) = \{C(v_1), \ldots, C(v_n)\}$ bestimmt. Sind zwei zulässige Färbungen gegeben, sind wir an derjenigen interessiert, die weniger Farben benötigt. Die Färbungen können also direkt als Werte $\mathcal{W}$ dienen. Die Minimumsbildung beim Zusammenfassen von Konstellationen muss mit Hilfe der Operation $\oplus$ realisiert werden. Um die Kommutativität von $\oplus$ zu sichern, ist ein „tie breaker“ für Mengen gleicher Mächtigkeit notwendig. Dazu bietet es sich an, die Färbungen mit Hilfe einer beliebigen linearen Ordnung $\prec$ (zum Beispiel der lexikographischen) zu ordnen und zu definieren:

$$\oplus : (C_1, C_2) \mapsto \begin{cases} C_1 & \text{falls } |\text{colours}(C_1)| < |\text{colours}(C_2)| \text{ oder} \\ & \quad |\text{colours}(C_1)| = |\text{colours}(C_2)| \wedge \text{colours}(C_1) \prec \text{colours}(C_2) \\ C_2 & \text{sonst} \end{cases} \tag{5.15}$$

Nicht alle $(n, 1)$-Konstellationen repräsentieren zulässige Färbungen. Um dennoch über den kompletten Konstellationsraum summieren zu können, ohne das Ergebnis zu verfälschen, können diese nichtzulässigen Konstellationen wie bereits in Bemerkung 2.22 angedeutet auf ein neues Element $\mathbb{O}$ von $\mathcal{W}$ abgebildet werden, das sich neutral bezüglich der Operation $\oplus$ verhält.

Lemma 5.17. *Das Tupel* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ *mit*

$$\begin{aligned} c_{\mathrm{V}} &= n \\ c_{\mathrm{E}} &= 1 \\ \mathcal{W} &= \mathcal{C} \cup \{\mathbb{O}\} \\ O &= \mathbb{O} \\ \oplus &: \textit{wie in } (5.15) \\ \mathrm{val} &: C \mapsto \begin{cases} C & \textit{falls } C \textit{ zulässig} \\ \mathbb{O} & \textit{sonst} \end{cases} \end{aligned} \tag{5.16}$$

$$\text{ind} \quad : \quad (C, A) \mapsto C|_A$$

ist eine formale Beschreibung für das vorliegende Problem in Indexform.

Beweis. Durch Überprüfen der Definition 2.23. Die verwendete Ordnung der Färbungen sorgt dafür, dass die Operation $\oplus$ kommutativ ist. □

Splitting

Lemma 5.18. *Sei* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ *die Indexform aus Lemma 5.17. Das Tupel* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{glue})$ *mit*

$$\text{glue} \quad : \quad (C_K, (i_1, w_1), (i_2, w_2)) \mapsto \begin{cases} w_1 \cup w_2 & \textit{falls} \;\; w_1 \neq \mathbb{O} \;\; \textit{und} \;\; w_2 \neq \mathbb{O} \\ \mathbb{O} & \textit{sonst} \end{cases} \tag{5.17}$$

stellt eine Splittingform für das Problem der Bestimmung einer Minimalfärbung dar.

Beweis. Seien $H_1 = (V_1, E_1)$ und $H_2 = (V_2, E_2)$ zwei Unterhypergraphen eines Hypergraphen $H = (V, E)$, für die gilt $H_1 \boxplus_K H_2 = H$. Sei $C_K \in \mathcal{C}_{(K,\emptyset)}$ fixiert sowie $C_1 \in \mathcal{C}_{H_1}$ und $C_2 \in \mathcal{C}_{H_2}$ mit $C_1|_K = C_2|_K = C_K$ gegeben. Wir unterscheiden zwei Fälle:

1. Es gilt $w_1 \neq \mathbb{O}$ und $w_2 \neq \mathbb{O}$. Dann repräsentiert $C_1 \cup C_2$ sowohl eingeschränkt auf H_1 als auch eingeschränkt auf H_2 eine zulässige Färbung und damit wegen der Übereinstimmung von C_1 und C_2 auf K eine zulässige Färbung von H. Insbesondere gilt:

$$\begin{aligned} &\mathrm{glue}(C_K, (\mathrm{ind}(C_1, K), \mathrm{val}(C_1)), (\mathrm{ind}(C_2, K), \mathrm{val}(C_2)) \\ &\qquad = \;\; \mathrm{val}(C_1) \cup \mathrm{val}(C_2) \;\; = \;\; \mathrm{val}(C_1 \cup C_2) \end{aligned}$$

2. Es gilt $w_1 = \mathbb{O}$ oder $w_2 = \mathbb{O}$, das heißt, $\mathrm{val}(C_1)$ oder $\mathrm{val}(C_2)$ ist gleich $\mathbb{O}$. Das heißt aber, C_1 oder C_2 repräsentiert eine nichtzulässige Färbung im jeweiligen Unterhypergraphen, so dass auch $C_1 \cup C_2$ nicht zulässig in H ist. Somit ist hier $\mathrm{val}(C_1 \cup C_2) = \mathbb{O}$.

In beiden Fällen ist (3.3) erfüllt. Von der Gültigkeit von (3.4) und (3.5) kann man sich leicht überzeugen, so dass die Behauptung aus Lemma 3.7 folgt. □

Komposition

Satz 5.19. *Es gelten wieder die Bezeichnungen aus Lemma 4.3. Die durch die Funktionen*

$$\begin{aligned} \mathrm{trz}^{\mathrm{L}} \quad &: \quad ((i, w), C_2) \mapsto \begin{cases} (i \cup C_2|_{V_2}, w \cup C_2) & \textit{falls} \;\; w \neq \mathbb{O} \;\; \textit{und} \;\; i \cup C_2 \;\textit{zul. in}\; T_2 \\ (i \cup C_2|_{V_2}, \mathbb{O}) & \textit{sonst} \end{cases} \\ \mathrm{trz}^{\mathrm{D}} \quad &: \quad ((i, w), v) \mapsto (i|_{A-v}, w) \end{aligned} \tag{5.18}$$

gemäß (4.9), (4.10) und (4.11) konstruierte Funktion trs *ergänzt die Indexform aus Lemma 5.17 zu einer Kompositionsform.*

Beweis. Sei $C_1 \in \mathcal{C}_{H_1}$, $C_2 \in \mathcal{C}_{T_2}$. Es werden zwei Fälle unterschieden:

1. Es ist $\mathrm{val}(C_1) \neq \mathbb{O}$ und $\mathrm{ind}(C_1, A_1) \cup C_2$ repräsentiert eine zulässige Färbung von T_2. Damit ist $C_1 \cup C_2$ zulässig in $H_1 + T_2$ und es gilt:

$$\begin{aligned} & \mathrm{trz}^{\mathrm{L}}((\mathrm{ind}(C_1, A_1), \mathrm{val}(C_1)),\, C_2) \\ \overset{(5.18)}{=}\ & (C_1|_{A_1} \cup C_2|_{V_2},\, \mathrm{val}(C_1) \cup \mathrm{val}(C_2)) \\ =\ & ((C_1 \cup C_2)|_{A_1 \cup V_2},\, C_1 \cup C_2) \\ =\ & (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2),\, \mathrm{val}(C_1 \cup C_2)) \end{aligned}$$

2. Andernfalls ist entweder C_1 nicht zulässig in H_1, oder aber $\mathrm{ind}(C_1, A_1) \cup C_2$ ist nicht zulässig in T_2. Damit ist $C_1 \cup C_2$ nicht zulässig in $H_1 + T_2$.

Sei $A \subseteq V$ eine Menge, $v \in A$ ein Knoten, $w \in \mathcal{W}$ ein Wert und $C \in \mathcal{C}_H$ eine Konstellation. Dann gilt:

$$\mathrm{trz}^{\mathrm{D}}((\mathrm{ind}(C, A), w), v) \;=\; \mathrm{trz}^{\mathrm{D}}((C|_A, w), v) \;=\; (C|_{A-v}, w) \;=\; (\mathrm{ind}(C, A - v), w)$$

Damit gelten (4.5) und (4.7). Die übrigen Voraussetzungen von Lemma 4.3 sind leicht nachprüfbar, so dass mit Hilfe von Satz 4.4 die Behauptung folgt. □

Bemerkung 5.20. Bei der Implementierung besteht keine Notwendigkeit, Zustände zu erzeugen, die unzulässige Färbungen repräsentieren, da diese keinen Einfluss auf das Endergebnis haben werden. Wenn so sichergestellt ist, dass keine „unzulässigen Zustände“ erzeugt werden, kann auch der Test $w \neq \mathbb{O}$ entfallen. Anstelle von (5.18) sollte also

$$\begin{aligned} \mathrm{trz}^{\mathrm{L}} &: ((i, w), C_2) \mapsto (i \cup C_2|_{V_2}, w \cup C_2) \cdot [i \cup C_2 \text{ zulässig}] \\ \mathrm{trz}^{\mathrm{D}} &: ((i, w), v) \mapsto (i|_{A-v}, w) \end{aligned} \tag{5.19}$$

umgesetzt werden. Derartige Vereinfachungen sind typisch für Kanten- und Knotenkompositionsalgorithmen, wenn die Abschlussmethode zum Einsatz kommt.

Satz 5.21. *Durch die Wahl von*

$$\begin{aligned} \mathrm{trs}^{\mathrm{V}} &: ((i, w), v) \mapsto \biguplus_{j=1}^{n} (i \cup \{(v, j)\}, w \cup \{(v, j)\}) \\ \mathrm{trs}^{\mathrm{E}} &: ((i, w), e) \mapsto (i, w \cup \{(e, 1)\}) \cdot [i(v_1) \neq i(v_2)\, \forall v_1, v_2 \in e : v_1 \neq v_2] \end{aligned} \tag{5.20}$$

entsteht aus der in Satz 5.19 angegebenen Kompositionsform eine Kantenkompositionsform für einen Kantenkompositionsalgorithmus vom Typ B.

Beweis. Wir benutzen die verbesserte Form (5.19), bei der keine Zustände erzeugt werden, die unzulässigen Konstellationen entsprechen und zeigen (4.13):

1. Im Knotenaktivierungsschritt ist $\mathcal{C}_2 = \{\{(v,1)\},\ldots,\{(v,n)\}\}$ und für $C_2 \in \mathcal{C}_2$ gilt $C_2|_{V_2} = C_2$. Der zu aktivierende Knoten v ist isoliert in $H_2 = H_1 + T_2 = (V_1 \cup \{v\}, E_1)$. Jede beliebige Färbung des Knotens erweitert damit eine vorherige zulässige Färbung zulässig. Es gilt also:

$$\begin{aligned}
\mathrm{trs}^{\mathrm{L}}((i,w),T_2) &= \biguplus_{C_2\in\mathcal{C}_{T_2}} \mathrm{trz}^{\mathrm{L}}((i,w),C_2)\\
&\overset{(5.19)}{=} \biguplus_{j=1}^{n}(i\cup\{(v,j)\},w\cup\{(v,j)\})\cdot[i\cup\{(v,j)\}\text{ zulässig}]\\
&= \biguplus_{j=1}^{n}(i\cup\{(v,j)\},w\cup\{(v,j)\})\\
&= \mathrm{trs}^{\mathrm{V}}((i,w),v)
\end{aligned}$$

2. Im Kantenschritt ist $\mathcal{C}_2 = \{\{(e,1)\}\}$ mit dem einzigen Element $C_2 = \{(e,1)\}$. Damit gilt:

$$\begin{aligned}
\mathrm{trs}^{\mathrm{L}}((i,w),T_2) &= \mathrm{trz}^{\mathrm{L}}((i,w),\{(e,1)\})\\
&\overset{(5.19)}{=} (i,w\cup\{(e,1)\})\cdot[i\cup\{(e,1)\}\text{ zulässig}]\\
&= (i,w\cup\{(e,1)\})\cdot[i(v_1)\neq i(v_2)\,\forall v_1,v_2\in e: v_1\neq v_2]\\
&= \mathrm{trs}^{\mathrm{E}}((i,w),e)
\end{aligned}$$

3. Aus 1. und 2. folgt wegen (4.13) die Behauptung. □

Bemerkung 5.22. Mit einer leichten Abwandlung des Algorithmus lässt sich das Entscheidungsproblem lösen, ob ein Graph mit einer gegebenen Anzahl d von Farben zulässig färbbar ist. Dazu wird

$$\mathrm{trs}^{\mathrm{V}} : ((i,w),v) \mapsto \biguplus_{j=1}^{d}(i\cup\{(v,j)\},w\cup\{(v,j)\})$$

verwendet. Es werden also nur Zustände erzeugt, die Konstellationen repräsentieren, die die Farben $1,\ldots,d$ verwenden. Ist die Zustandsmenge im letzten Schritt leer, so gibt es keine Färbung mit der geforderten Eigenschaft, bleibt dagegen ein Element übrig, ist eine Färbung gefunden, die höchstens d Farben verwendet. Diese Färbung ist sogar eine Minimalfärbung.

Bemerkung 5.23. Durch aufeinanderfolgende Versuche, den Graphen wie in der vorstehenden Bemerkung beschrieben mit maximal 1, 2, 3, usw. Farben zu färben, ist es unter Umständen möglich, eine Minimalfärbung schneller zu bestimmen als mit einer direkten Rechnung ohne Beschränkung der Farbzahl, da durch diese Beschränkung die Zahl der Zustände erheblich eingeschränkt wird. Dieses Verfahren ist insbesondere hilfreich, wenn die chromatische Zahl klein im Vergleich zur Zahl der Knoten im Graphen ist. Die Zahl der notwendigen Iterationen entspricht dabei der chromatischen Zahl. Diese kann *a priori* zum Beispiel durch den Satz von Brooks abgeschätzt werden.

Bemerkung 5.24. Es ist möglich, die Anzahl der zu berechnenden Zustände sowohl beim Splitting als auch in der Kantenkomposition noch weiter erheblich zu verringern. Dazu wird zunächst die Aufgabenstellung leicht modifiziert: Anstelle einer beliebigen Minimalfärbung soll diejenige gefunden werden, die unter Minimalfärbungen am kleinsten bezüglich der lexikographischen Ordnung ist. Eine so gefundene Färbung ist offensichtlich auch eine Lösung des ursprünglichen Problems. Weiterhin ist damit die Farbe des ersten bearbeiteten Knotens fixiert. Der erste Knotenschritt des Kantenkompositionsalgorithmus liefert also nur noch einen statt $n = |V|$ Zustände. Für den zweiten Knoten gibt es dann nur zwei mögliche Farben, also insgesamt zwei statt n^2 Zustände. Später kann die Zahl der Zustände auf $\mathcal{O}(d^d)$ mit $d = \mathrm{pw}(H) + 1$ beschränkt werden. (Es gibt immer maximal d aktive Knoten. Die Farben dieser Knoten können immer aus der Menge $\{1, \ldots, d\}$ gewählt werden, indem ggf. Farben global ausgetauscht werden.) Dieser Wert ist zwar noch immer exponentiell in der Wegweite des Hypergraphen, aber nicht mehr von seiner Knotenzahl abhängig.

Auf die Details der Umsetzung soll hier verzichtet werden. Eine Implementation aller erwähnten Verbesserungen ist in [Poe03] zu finden.

5.2.2 Chromatische Zahl

Problem

Im vorliegenden Abschnitt soll eine Variation der gerade bearbeiteten Problemstellung behandelt werden. Diese wird häufig als vollkommen eigenständige Aufgabe betrachtet, führt jedoch bei der Anwendung der Kompositionsmethode zu einem nahezu identischen Algorithmus: Es soll die *chromatische Zahl* $\chi(H)$ eines Hypergraphen berechnet werden. Diese ist gleich der Zahl der in einer Minimalfärbung des Hypergraphen verwendeten Farben.

Lösung

Ist eine zulässige Färbung mit minimaler Farbzahl eines Hypergraphen H gegeben, so ist die Bestimmung von $\chi(H)$ trivial. Insofern können die Algorithmen aus dem vorangegangenen Abschnitt 5.2.1 direkt verwendet werden.

Reduktion

Für das Entscheidungsproblem, ob ein bestimmter Graph $G = (V, E)$ mit einer vorgegebenen Zahl von d Farben zulässig färbbar ist, existiert eine Reduktion, die besonders in wenig dichten Graphen (bis hin zu $|E| \approx \frac{1}{2}d|V|$) wirkungsvoll eingesetzt werden kann: Ein Knoten v vom Grad $d-1$ oder kleiner kann entfernt werden, da eine zulässige Färbung mit d Farben des verbleibenden Rests nur $d-1$ Farben für die Nachbarn von v verwendet und somit mindestens eine weitere Farbe für v verbleibt. Diese Reduktion kann solange angewandt werden, bis der verbleibende Rest nur Knoten vom Grad d oder mehr hat. Verbleibt ein leerer Restgraph, so heißt der ursprüngliche Graph $d-1$-*degeneriert* und kann mit d Farben zulässig gefärbt werden.

Eine zweite Reduktion existiert für nichtadjazente Knoten u und v von G, für die $N_G(u) \subseteq N_G(v)$ gilt. In diesem Fall kann eine zulässige k-Färbung von $G-u$ auf G erweitert werden, da

u mit der selben Farbe wie v gefärbt werden kann. Auch hier kann also ein kompletter Knoten samt inzidenter Kanten entfernt und damit die Größe des zu lösenden Problems verringert werden.

Beide Reduktionen können in beliebiger Reihenfolge angewendet werden. Verbleibt nach den Reduktionen ein nichtleerer Restgraph, kann die Frage nach der d-Färbbarkeit noch nicht beantwortet werden. Das verbleibende Restproblem muss mit einer anderen Methode gelöst werden.

5.2.3 Chromatisches Polynom

Problem

Der Wert des *chromatischen Polynoms* $\chi_G(x) \in \mathbb{Z}[x]$ an der Stelle $x \in \mathbb{N}$ gibt an, auf wieviel verschiedene Weisen der Graph G mit x Farben zulässig gefärbt werden kann.

Komposition

Die für die Berechnung des chromatischen Polynoms bekannte Kantenkompositionsformel

$$\begin{aligned} \chi_G(x) &= \chi_{G_{-e}}(x) - \chi_{G_e}(x) \\ \chi_{\bar{K}_n}(x) &= x^n \end{aligned}$$

passt in das im Abschnitt 5.1 behandelte Schema mit $a(e) = -1$, $b(e) = 1$ und $r(n) = x^n$.

5.3 Matching

Auch dieser Abschnitt behandelt zum Zweck der Demonstration der Kompositionsmethode wohlbekannte Problemstellungen. Die dabei entstehenden Kompositionsalgorithmen sind exponentiell von der Wegweite der Graphen abhängig und damit zumindest in den beiden ersten Beispielen „Maximales Matching“ und „Matchingzahl“ asymptotisch schlechter als bekannte polynomiale Algorithmen (vgl. zum Beispiel [GoM95]). Letztere lassen sich allerdings nicht für das im dritten Beispiel behandelte „Matchingpolynom“ verallgemeinern, wohl aber die angegebenen Kompositionsalgorithmen.

5.3.1 Maximales Matching

Problem

Ein *Matching* ist eine Teilmenge M der Kantenmenge E eines Hypergraphen $H = (V, E)$, so dass jeder Knoten des Hypergraphen zu maximal einer der Kanten aus M inzident ist. Es soll ein Matching maximaler Mächtigkeit in einem Hypergraphen gefunden werden.

Indexform

Die Definition eines Matchings deutet mit ihrem direkten Bezug zu bestimmten Kantenmengen darauf hin, dass der $(1,2)$-Konstellationsraum verwendet werden sollte. Eine Konstellation C kann dann als Teilmenge $\hat{C}$ der Kantenmenge des Hypergraphen und damit als potentielles Matching interpretiert werden. Die Operation $\oplus$ übernimmt von zwei Matchings das „bessere“, also das mit mehr Elementen. Die Kommutativität dieser Operation wird, wie bei der Konstruktion einer zulässigen Färbung im Abschnitt 5.2.1 beschrieben, durch Einführung einer (beliebigen, also zum Beispiel wieder der lexikographischen) Ordnung $\prec$ auf der Menge der Matchings garantiert.

Da nicht alle Kantenmengen Matchings sind, muss die Bewertung einer Kantenmenge, die kein Matching ist, so erfolgen, dass sie keinen Einfluss auf das Ergebnis hat. Dafür wird wieder die in Bemerkung 2.22 beschriebene Abschlussmethode eingesetzt und ein besonderer Wert $\mathbb{O}$ eingeführt, für den $\mathbb{O} \oplus M = M \oplus \mathbb{O} = M$ für alle Matchings M gilt. Damit nimmt $\oplus$ die folgende Form an:

$$\oplus : (w_1, w_2) \mapsto \begin{cases} w_1 & \text{falls } \; w_2 = \mathbb{O} \\ w_2 & \text{falls } \; w_1 = \mathbb{O} \\ w_1 & \text{falls } \; w_1, w_2 \neq \mathbb{O} \;\text{ und }\; (|w_1| \geq |w_2| \text{ oder } |w_1| = |w_2| \wedge w_1 \prec w_2) \\ w_2 & \text{sonst} \end{cases}$$

Lemma 5.25. *Eine formale Beschreibung des Problems in Indexform ist das Tupel* $(c_\mathrm{V}, c_\mathrm{E}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$*, für das gilt:*

$$\begin{aligned} c_\mathrm{V} &= 1 \\ c_\mathrm{E} &= 2 \\ \mathcal{W} &= 2^E \cup \{\mathbb{O}\} \\ O &= \mathbb{O} \\ \oplus &: \textit{wie oben} \\ \mathrm{val} &: C \mapsto \begin{cases} \hat{C} & \textit{falls } \; \mathrm{valid}(\hat{C}) \\ \mathbb{O} & \textit{sonst} \end{cases} \\ \mathrm{ind} &: (C, A) \mapsto A \cap \bigcup_{e \in \hat{C}} e \end{aligned}$$

Dabei bezeichnet $\mathrm{valid}(\hat{C})$ *die Zulässigkeit der Kantenmenge* $\hat{C}$ *als Matching, also den Wert* $[e_1 \cap e_2 = \emptyset \, \forall e_1, e_2 \in \hat{C} : e_1 \neq e_2]$.

Beweis. Durch Überprüfen der Definition 2.23. Die eingeführte Ordnung stellt die Kommutativität der Operation $\oplus$ sicher. □

Komposition

Satz 5.26. *Die Indexform* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ *aus Lemma 5.25 kann durch*

$$\begin{aligned} \mathrm{trz}^{\mathrm{L}} &: ((i,w),C_2) \mapsto \begin{cases} (i \cup \bigcup_{e\in\hat{C}_2} e, w \cup \hat{C}_2) & \textit{falls } \; w \neq \mathbb{O} \wedge \mathrm{valid}(\hat{C}_2) \wedge i \cap \bigcup_{e\in\hat{C}_2} e = \emptyset \\ (i \cup \bigcup_{e\in\hat{C}_2} e, \mathbb{O}) & \textit{sonst} \end{cases} \\ \mathrm{trz}^{\mathrm{D}} &: ((i,w),v) \mapsto (i - v, w) \end{aligned} \tag{5.21}$$

zu einer Kompositionsform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{trs})$ *ergänzt werden.*

Beweis. Sei $C_1 \in \mathcal{C}_{H_1}$, $C_2 \in \mathcal{C}_{T_2}$ beliebig gewählt. Wir unterscheiden zwei Fälle:

1. Es gelte $\mathrm{val}(C_1) \neq \mathbb{O}$, $\mathrm{ind}(C_1, A_1) \cap \bigcup_{e\in\hat{C}_2} e = \emptyset$ und $\hat{C}_2$ zulässig auf T_2. Dann ist $\hat{C}_1 \cup \hat{C}_2$ sowohl eingeschränkt auf H_1 als auch auf T_2 ein Matching und die beiden eingeschränkten Matchings überdecken disjunkte Knotenmengen. Damit ist $\hat{C}_1 \cup \hat{C}_2$ ein Matching in $H_1 + T_2$ und es gilt:

$$\begin{aligned} \mathrm{trz}^{\mathrm{L}}((\mathrm{ind}(C_1,A_1), \mathrm{val}(C_1)), C_2) &= (\mathrm{ind}(C_1,A_1) \cup \bigcup_{e\in\hat{C}_2} e,\ \mathrm{val}(C_1) \cup \hat{C}_2) \\ &= (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2),\ \mathrm{val}(C_1 \cup C_2)) \end{aligned}$$

2. Im anderen Fall ist entweder $\hat{C}_1$ bereits kein Matching in H_1, oder aber die Erweiterung um $\hat{C}_2$ liefert kein Matching in $H_1 + T_2$ mehr. Auch in diesem Fall ist

$$\mathrm{trz}^{\mathrm{L}}((\mathrm{ind}(C_1, A), \mathrm{val}(C_1)), C_2) = (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2),\ \mathrm{val}(C_1 \cup C_2)).$$

In beiden Fällen ist also (4.5) erfüllt. Die Gültigkeit von (4.7) sieht man genauso leicht wie die der restlichen Voraussetzungen von Lemma 4.3, so dass nach 4.4 die Behauptung gilt. □

Bemerkung 5.27. Verzichtet man auf die Erzeugung von Zuständen mit einem Wert $\mathbb{O}$, so vereinfacht sich (5.21) zu

$$\begin{aligned} \mathrm{trz}^{\mathrm{L}} &: ((i,w),C_2) \mapsto (i \cup \bigcup_{e\in\hat{C}_2} e, w \cup \hat{C}_2) \cdot [i \cap \bigcup_{e\in\hat{C}_2} e = \emptyset \wedge \mathrm{valid}(\hat{C}_2)] \\ \mathrm{trz}^{\mathrm{D}} &: ((i,w),v) \mapsto (i - v, w). \end{aligned} \tag{5.22}$$

Dies kann bei einer Implementierung zur Vereinfachung und Beschleunigung der Berechnungen eingesetzt werden.

Satz 5.28. *Die Transformationsfunktionen aus (5.22) lassen sich wie folgt für eine Kantenkompositionsform vom Typ A spezialisieren:*

$$\begin{aligned} \mathrm{trs}^{\mathrm{V}} &: ((i,w),v) \mapsto (i,w) \\ \mathrm{trs}^{\mathrm{E}} &: ((i,w),e) \mapsto (i \cup e, w \cup \{e\}) \cdot [i \cap e = \emptyset] \ \uplus \ (i,w) \end{aligned}$$

Beweis.

1. In einem Knotenaktivierungsschritt ist $\mathcal{C}_2 = \{\{(v,1)\}\}$. Für das einzige $C_2 \in \mathcal{C}_2$ gilt $\hat{C}_2 = \emptyset$ ($\hat{C}_2$ ist im Falle der vorliegenden (1,2)-Konstellationen eine Kanten-, keine Knotenmenge) und damit

$$\begin{aligned}
\mathrm{trs}^{\mathrm{L}}((i,w),T_2) &= \mathrm{trz}^{\mathrm{L}}((i,w),\{(v,1)\}) \\
&= (i \cup \bigcup_{e\in\emptyset} e, w \cup \emptyset) \cdot [i \cap \bigcup_{e\in\emptyset} e = \emptyset] \\
&= (i,w) \\
&= \mathrm{trs}^{\mathrm{V}}((i,w),v).
\end{aligned}$$

2. In einem Kantenschritt ist $\mathcal{C}_2 = \{\{(e,1),(e,2)\}\}$ und damit

$$\begin{aligned}
\mathrm{trs}^{\mathrm{L}}((i,w),T_2) &= \mathrm{trz}^{\mathrm{L}}((i,w),\{(e,1)\}) \ \uplus\ \mathrm{trz}^{\mathrm{L}}((i,w),\{(e,2)\}) \\
&= (i \cup \bigcup_{f\in\{e\}} f, w \cup \{e\}) \cdot [i \cap \bigcup_{f\in\{e\}} e = \emptyset] \ \uplus \\
&\qquad (i \cup \bigcup_{f\in\emptyset} f, w \cup \emptyset) \cdot [i \cap \bigcup_{f\in\emptyset} e = \emptyset] \\
&= (i \cup e, w \cup \{e\}) \cdot [i \cap e = \emptyset] \ \uplus\ (i,w) \\
&= \mathrm{trs}^{\mathrm{E}}((i,w),e).
\end{aligned}$$

□

5.3.2 Matchingzahl

Problem

Es soll die Anzahl der Kanten eines Matchings maximaler Mächtigkeit in einem Hypergraphen gefunden werden, das heißt die *Matchingzahl*

$$\mu(G) = \max_{\substack{F\subseteq E \\ F \text{ ist Matching}}} |F| = \max_{\substack{F\subseteq E: \\ e_1\cap e_2=\emptyset \\ \forall e_1,e_2\in F: e_1\neq e_2}} |F|$$

bestimmt werden.

Lösung

Die Matchingzahl kann aus einem der im vorangegangenen Abschnitt konstruierten Matchings durch simples Abzählen der Kanten gewonnen werden. Dabei ist eine Verbesserung der Algorithmen möglich, indem als Werte keine Kantenmengen sondern nur noch ihre Mächtigkeiten benutzt werden, so dass die verwendete Indexform sich nur durch

$$\mathcal{W} = \mathbb{Z} \cup \{\mathbb{O}\}$$

$$\oplus \quad : \quad (w_1, w_2) \mapsto \begin{cases} w_1 & \text{falls } \ w_2 = \mathbb{O} \\ w_2 & \text{falls } \ w_1 = \mathbb{O} \\ \max(w_1, w_2) & \text{sonst} \end{cases}$$

$$\text{val} \quad : \quad C \mapsto \begin{cases} |\hat{C}| & \text{falls valid}(\hat{C}) \\ \mathbb{O} & \text{sonst} \end{cases}$$

von der aus Lemma 5.25 unterscheidet.

Sämtliche formalen Beschreibungen und Nachweise können völlig analog angegeben bzw. geführt werden, so dass hier auf deren Wiederholung verzichtet wird. Es sei lediglich bemerkt, dass als spezielle Transformationsfunktionen für einen Kantenkompositionsalgorithmus vom Typ A die Funktionen

$$\begin{aligned} \text{trs}^{\text{V}} \quad &: \quad ((i, w), v) \mapsto (i, w) \\ \text{trs}^{\text{E}} \quad &: \quad ((i, w), e) \mapsto (i \cup e, w + 1) \cdot [i \cap e = \emptyset] \ \uplus \ (i, w) \end{aligned}$$

verwendet werden können.

5.3.3 Matching-Polynom

Problem

Das *Matchingpolynom* eines Graphen G ist ein univariates Polynom $M_G(x) \in \mathbb{Z}[x]$. Mit der Bezeichnung $p_G(k)$ für die Zahl der Matchings der Mächtigkeit k in G gilt nach [CDGT88]:

$$M_G(x) \quad = \quad \sum_{k=0}^{m} (-1)^k p_G(k)\, x^{n-2k}$$

Wir werden hier ein etwas verändertes Polynom $m_G(x) \in \mathbb{Z}[x]$ berechnen, für dessen Koeffizienten $[x^k] m_G(x) = p_G(k)$ gilt. Beide Polynome sind direkt ineinander transformierbar.

Lösung

Auch dieses Problem kann völlig analog zu Abschnitt 5.3.1 behandelt werden. Die Unterschiede bestehen wieder im Monoid der Werte, für das hier

$$\begin{aligned} \mathcal{W} \quad &= \quad \mathbb{Z}[x] \cup \{\mathbb{O}\} \\ \oplus \quad &: \quad (w_1, w_2) \mapsto \begin{cases} w_1 & \text{falls } \ w_2 = \mathbb{O} \\ w_2 & \text{falls } \ w_1 = \mathbb{O} \\ w_1 + w_2 & \text{sonst} \end{cases} \end{aligned}$$

gewählt wird, und der Bewertung

$$\text{val} \quad : \quad C \mapsto \begin{cases} x^{|\hat{C}|} & \text{falls valid}(\hat{C}) \\ \mathbb{O} & \text{sonst.} \end{cases}$$

Als spezielle Transformationsfunktionen für einen Kantenkompositionsalgorithmus vom Typ A werden nach einer analogen Herleitung die Funktionen

$$\begin{aligned} \mathrm{trs}^{\mathrm{V}} &: ((i,w),v) \mapsto (i,w) \\ \mathrm{trs}^{\mathrm{E}} &: ((i,w),e) \mapsto (i \cup e, x\,w) \cdot [i \cap e = \emptyset] \ \uplus \ (i,w) \end{aligned}$$

gefunden. Auch hier kann auf die Einführung des Elementes $\mathbb{O}$ verzichtet und an seiner Stelle die gewöhnliche Null aus $\mathbb{Z}[x]$ verwendet werden. Damit vereinfacht sich die vollständige Beschreibung zu:

$$\begin{aligned} c_{\mathrm{V}} &= 1 \\ c_{\mathrm{E}} &= 2 \\ \mathcal{W} &= \mathbb{Z}[x] \\ \oplus &: (w_1,w_2) \mapsto w_1 + w_2 \\ \mathrm{val} &: C \mapsto x^{|\hat{C}|} \cdot [\mathrm{valid}(\hat{C})] \\ \mathrm{ind} &: (C,A) \mapsto A \cap \bigcup_{e \in \hat{C}} e \\ \mathrm{trs}^{\mathrm{V}} &: ((i,w),v) \mapsto (i,w) \\ \mathrm{trs}^{\mathrm{E}} &: ((i,w),e) \mapsto (i \cup e, x\,w) \cdot [e \cap i = \emptyset] \ \uplus \ (i,w) \\ \mathrm{trz}^{\mathrm{D}} &: ((i,w),v) \mapsto (i - v, w) \end{aligned}$$

5.4 Negami und Tutte

5.4.1 Negami-Polynom

Problem

Für gewöhnliche Graphen wurde von Seiya Negami in [Neg87] eine polynomiale Grapheninvariante in drei Unbestimmten eingeführt, für die folgende Dekompositionsformel gilt:

$$\begin{aligned} N(G) &= x\,N(G_e) + y\,N(G_{-e}) \\ N(\overline{K}_n) &= t^n \end{aligned} \tag{5.23}$$

Komposition

Satz 5.29. *Die Wahl von*

$$\begin{aligned} \mathrm{trs}^{\mathrm{V}} &: ((i,w),v) \mapsto (i_{+v}, w\,t) \\ \mathrm{trs}^{\mathrm{E}} &: ((i,w),e) \mapsto (i_e, w\,x\,t^{|i_e|-|i|}) \ \uplus \ (i, w\,y) \\ \mathrm{trs}^{\mathrm{D}} &: ((i,w),v) \mapsto (i_{-v}, w) \end{aligned} \tag{5.24}$$

liefert einen Kantenkompositionsalgorithmus vom Typ A.

Beweis. Die Dekompositionsformel (5.23) hat die im Abschnitt 5.1 behandelte Form. Die angegebenen Transformationsformeln entsprechen der Wahl von $a(e) = x$, $b(e) = y$ und $r(n) = t^n$. □

5.4.2 Tutte-Polynom

Problem

William Tutte hat für die nach ihm benannte bivariate Polynominvariante mit ganzzahligen Koeffizienten [Tut74] folgende Dekompositionsformel angegeben:

$$\begin{aligned} T_G(x,y) &= \begin{cases} y\,T_{G_{-e}}(x,y) & \text{falls } e \text{ eine Schlinge in } G \text{ ist} \\ x\,T_{G_{-e}}(x,y) & \text{falls } e \text{ eine Brücke in } G \text{ ist} \\ T_{G_{\setminus e}}(x,y) + T_{G_{-e}}(x,y) & \text{sonst} \end{cases} \\ T_{\bar{K}_n}(x,y) &= 1 \end{aligned}$$

Dabei sind Schlingen in Graphen zulässig und $G_{\setminus e}$ bezeichnet eine Operation ähnlich zur sonst verwendeten Kontraktion, bei der aber eventuell entstehende Schlingen *nicht* entfernt werden.

Komposition

Die angegebene Dekompositionsformel passt leider nicht direkt in das im Abschnitt 5.1 behandelte Schema. Es ist allerdings aus [Neg87] bekannt, dass Tutte- und Negami-Polynom für jeden Graphen $G = (V, E)$ durch

$$N_G((x-1)(y-1), 1, y-1) = (y-1)^{|V|}(x-1)^{k(G)}T_G(x,y) \tag{5.25}$$

ineinander überführt werden können, also in einem gewissen Sinne äquivalent sind. Dadurch kann auch das Tutte-Polynom mit einem Kantenkompositionsalgorithmus vom Typ A berechnet werden, indem der im vorangegangenen Abschnitt beschriebene Algorithmus zur Berechnung des Negami-Polynoms entsprechend der Formel (5.25) modifiziert wird:

Satz 5.30. *Die Transformation*

$$\begin{aligned} \mathrm{trs}^{\mathrm{V}} &: ((i,w),v) \mapsto (i_{+v}, w\,(x-1)) \\ \mathrm{trs}^{\mathrm{E}} &: ((i,w),e) \mapsto (i,w) \uplus (i_e, w\,(i_e = i\,?\,y-1 : \frac{1}{x-1})) \\ \mathrm{trs}^{\mathrm{D}} &: ((i,w),v) \mapsto (i_{-v}, w) \end{aligned}$$

kann zur Konstruktion eines Kantenkompositionsalgorithmus vom Typ A verwendet werden. Dabei bezeichnet die Abkürzung $a\,?\,b : c$ den Wert b falls a erfüllt ist und c sonst.

Beweis. Folgt aus 5.24 und 5.25. □

Dieses Verfahren ist wesentlich einfacher als der direkte Weg über die von Tutte angegebene Dekompositionsformel, da hier die „ungewöhnliche“ Unterscheidung der Kanten in Brücken und Schlingen nicht notwendig ist.

Bemerkung 5.31. Durch Substitution von $x-1$ durch x ergeben sich die speziellen Transformationsfunktionen

$$\begin{aligned}
\mathrm{trs}^{\mathrm{V}} &: ((i,w),v) \mapsto (i_{+v}, w\,x) \\
\mathrm{trs}^{\mathrm{E}} &: ((i,w),e) \mapsto (i,w) \uplus (i_e, w\,(i_e = i\,?\,y-1 : \frac{1}{x})) \\
\mathrm{trs}^{\mathrm{D}} &: ((i,w),v) \mapsto (i_{-v}, w),
\end{aligned}$$

die natürlich auch nur ein entsprechend modifiziertes Polynom liefern. Der Vorteil besteht im Verzicht auf die relativ teure Polynomdivision durch $x-1$. Durch weitere Modifikationen ist es auch möglich, ganz auf die Division zu verzichten. Man kann zum Beispiel die Multiplikationen mit x im Knotenschritt mit entsprechenden Divisionen im Kantenschritt kompensieren, wodurch ein Algorithmus entsteht, der lediglich Polynomadditionen und -subtraktionen sowie Verschiebungen der Exponenten verwendet.

Bemerkung 5.32. Es war in den Beispielrechnungen interessanterweise zu beobachten, dass die Koeffizienten des verschobenen Polynoms absolut größer als die des richtigen Tutte-Polynoms sind. Das gilt auch für Verschiebungen um einen anderen Betrag als 1. Eine Begründung dafür konnte allerdings nicht gefunden werden.

Folgerung 5.33. *Alle Invarianten, die aus dem Tutte-Polynom abgeleitet werden können, also zum Beispiel das chromatische Polynom, das Zuverlässigkeitspolynom, die Anzahl der Nirgendsnullflüsse, die Anzahl der azyklischen Orientierungen u.v.a.m., sind mit Hilfe von Kompositionsalgorithmen in Graphen beschränkter Wegweite bei geeigneter gegebener Wegzerlegung in polynomialer Zeit berechenbar.*

Beweis. Das Tutte-Polynom hat durch die Äquivalenz (5.25) und wegen (5.23) eine Darstellung durch eine Kantendekompositionsformel der Art (5.1). Die Behauptung folgt unmittelbar aus Satz 5.13. □

5.5 Unabhängige Mengen

Eine Teilmenge I der Knotenmenge V eines (gewöhnlichen) Graphen $G = (V, E)$ heißt *unabhängig*, wenn keine zwei Knoten dieser Teilmenge zueinander benachbart sind. Drei Problemstellungen im Zusammenhang mit derartigen unabhängigen Mengen (das Bestimmen der Unabhängigkeitszahl, das Bestimmen einer unabhängigen Menge maximaler Mächtigkeit sowie die Berechnung des Unabhängigkeitspolynoms) können durch einander sehr ähnliche Kantenkompositionsalgorithmen gelöst werden.

Da auch hier eine direkte Übertragung sowohl der Definition der Unabhängigkeit als auch der resultierenden Algorithmen von gewöhnlichen Graphen auf Hypergraphen möglich ist, erfolgt die Darstellung wieder für den allgemeinen Fall.

5.5.1 Unabhängigkeitszahl

Problem

Es soll das Maximum der Mächtigkeiten aller unabhängigen Knotenmengen eines Hypergraphen $H = (V, E)$ bestimmt werden. Diese Zahl wird Unabhängigkeitszahl genannt und mit $\alpha(H)$ bezeichnet.

Indexform

Knoten können entweder in einer unabhängigen Menge enthalten sein oder nicht, Kanten sind „uninteressant“, insofern bietet sich der $(2, 1)$-Konstellationsraum zur Beschreibung des Problems an. Eine Konstellation kann damit direkt als „Kandidat für eine unabhängige Knotenmenge“ interpretiert werden, dessen Zulässigkeit in die Bewertung mit einfließen muss. Dabei kann wieder die Abschlussmethode aus Bemerkung 2.22 eingesetzt werden.

Es verbleibt die Bestimmung der Wertemenge und der Operation $\oplus$. Als Werte für zulässige Konstellationen dienen die Mächtigkeiten der durch sie repräsentierten unabhängigen Mengen, also ganze Zahlen zwischen 0 und $|V|$. Von zwei Mächtigkeiten unabhängiger Mengen soll die größere Mächtigkeit erhalten werden, die Addition im Monoid wird also durch die Maximumbildung in den ganzen Zahlen realisiert. Der Wert der unzulässigen Konstellationen soll wieder mit $\mathbb{O}$ bezeichnet werden.

Damit ist bereits eine formale Beschreibung in Indexform gefunden:

Lemma 5.34. *Das Tupel* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind})$ *mit*

$$
\begin{aligned}
c_{\mathrm{V}} &= 2 \\
c_{\mathrm{E}} &= 1 \\
\mathcal{W} &= \{0, \ldots, |V|\} \cup \{\mathbb{O}\} \\
O &= \mathbb{O} \\
\oplus \quad &: \quad (w_1, w_2) \mapsto \begin{cases} w_1 & \textit{falls} \quad w_2 = \mathbb{O} \\ w_2 & \textit{falls} \quad w_1 = \mathbb{O} \\ \max(w_1, w_2) & \textit{sonst} \end{cases} \\
\mathrm{val} \quad &: \quad C \mapsto \begin{cases} |\hat{C}| & \textit{falls} \quad \hat{C} \quad \textit{unabh.} \\ \mathbb{O} & \textit{sonst} \end{cases} \\
\mathrm{ind} \quad &: \quad (C, A) \mapsto \hat{C}|_A
\end{aligned}
$$

bildet eine Indexform für das Problem.

Beweis. Direkt durch Überprüfen der Definition 2.23. □

Splitting

Satz 5.35. *Mit den Bezeichnungen aus Lemma 3.7 und unter Verwendung der in Lemma 5.34 angegebenen Indexform sowie von*

$$\begin{aligned}&\operatorname{glue}(C_K,(i_1,w_1),(i_2,w_2))\\ &\quad=\begin{cases} w_1+w_2-|C_K| & \textit{falls}\ \ w_1\neq\mathbb{O}\wedge w_2\neq\mathbb{O}\\ \mathbb{O} & \textit{sonst}\end{cases}\end{aligned}\tag{5.26}$$

ist das Tupel $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \operatorname{val}, \operatorname{ind}, \operatorname{glue})$ *eine Splittingform für die Berechnung der Unabhängigkeitszahl.*

Beweis. Sei für eine Zerlegung $H = H_1 \boxplus_K H_2$ eines Hypergraphen in Teile H_1 und H_2 eine Konstellation $C_K \in \mathcal{C}_{(K,\emptyset)}$ fixiert und $C_1 \in \mathcal{C}_{H_1}$, $C_2 \in \mathcal{C}_{H_2}$ mit $C_1|_K = C_2|_K = C_K$ beliebig gewählt. Wir betrachten zwei Fälle:

1. Es gilt $\operatorname{val}(C_1) \neq \mathbb{O} \wedge \operatorname{val}(C_2) \neq \mathbb{O}$. Dann ist die Knotenmenge $\hat{C}_1 \cup \hat{C}_2$ sowohl eingeschränkt auf H_1 als auch eingeschränkt auf H_2 eine unabhängige Menge, wobei diese eingeschränkten Mengen aufgrund der Definition von C_1 und C_2 auf K übereinstimmen. Damit ist $\hat{C}_1 \cup \hat{C}_2$ eine unabhängige Menge auf H und es gilt:

$$\begin{aligned}&\operatorname{glue}(C_K,(\operatorname{ind}(C_1,K),\operatorname{val}(C_1)),(\operatorname{ind}(C_2,K),\operatorname{val}(C_2)))\\ &\overset{(5.26)}{=}\ \operatorname{val}(C_1)+\operatorname{val}(C_2)-|C_K|\\ &=\ |\hat{C}_1|+|\hat{C}_2|-|\hat{C}_K|\\ &=\ |\hat{C}_1\cup\hat{C}_2|\\ &=\ \operatorname{val}(C_1\cup C_2)\end{aligned}$$

2. Im anderen Fall muss $\operatorname{val}(C_1) = \mathbb{O}$ oder $\operatorname{val}(C_2) = \mathbb{O}$ sein. Mindestens eine der Einschränkungen von $\hat{C}_1 \cup \hat{C}_2$ auf H_1 bzw. H_2 ist also keine unabhängige Menge, mithin ist auch $\hat{C}_1 \cup \hat{C}_2$ keine unabhängige Menge in H und es gilt $\operatorname{val}(\hat{C}_1 \cup \hat{C}_2) = \mathbb{O}$.

In beiden Fällen ist also $\operatorname{glue}(C_K, (\operatorname{ind}(C_1, K), \operatorname{val}(C_1)), (\operatorname{ind}(C_2, K), \operatorname{val}(C_2))) = \operatorname{val}(C_1 \cup C_2)$ und damit (3.3) erfüllt. Von der Gültigkeit von (3.4) und (3.5) überzeugt man sich leicht. Somit bildet die Funktion glue wegen Lemma 3.7 eine mögliche Ergänzung der Indexform aus Lemma 5.34 zu einer Splittingform. □

Komposition

Satz 5.36. *Es gelten die Bezeichungen aus Lemma 4.3. Die aus*

$$\begin{aligned}\operatorname{trz}^{\mathrm{L}} &: ((i,w),C_2)\mapsto\begin{cases}(i\cup\hat{C}_2, w+|\hat{C}_2|) & \textit{falls}\ \ w\neq\mathbb{O}\ \ \textit{und}\ \ i\cup\hat{C}_2\ \textit{unabh. in}\ T_2\\ (i\cup\hat{C}_2,\mathbb{O}) & \textit{sonst}\end{cases}\\ \operatorname{trz}^{\mathrm{D}} &: ((i,w),v)\mapsto(i-v,w)\end{aligned}$$

gemäß (4.9), (4.11) und (4.10) konstruierte Transformationsfunktion trs *bildet zusammen mit der oben in Lemma 5.34 angeführten Indexform eine Kompositionsform.*

Beweis. Sei $C_1 \in \mathcal{C}_{H_1}$, $C_2 \in \mathcal{C}_{T_2}$ beliebig gewählt. Wir unterscheiden wieder zwei Fälle:

1. Es gilt $\mathrm{val}(C_1) \neq \mathbb{O}$ und $\mathrm{ind}(C_1, A_1) \cup \hat{C}_2$ ist unabhängig in T_2. Dann ist $\hat{C}_1 \cup \hat{C}_2$ auf $H_1 + T_2$ eine unabhängige Menge und es gilt

$$\begin{aligned} \mathrm{trz}^{\mathrm{L}}((\mathrm{ind}(C_1, A_1), \mathrm{val}(C_1)), C_2) &= (\mathrm{ind}(C_1, A_1) \cup \hat{C}_2,\ \mathrm{val}(C_1) + |\hat{C}_2|) \\ &= (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2),\ \mathrm{val}(C_1 \cup C_2)). \end{aligned}$$

2. Im anderen Fall ist bereits $\hat{C}_1$ nicht unabhängig in H_1, oder aber die Einschränkung $\hat{C}_1|_{A_1}$ wird durch $\hat{C}_2$ nicht zu einer unabhängigen Menge in T_2 ergänzt. In beiden Fällen ist $\hat{C}_1 \cup \hat{C}_2$ keine unabhängige Menge in $H_1 + T_2$, das heißt, es gilt $\mathrm{val}(\hat{C}_1 \cup \hat{C}_2) = \mathbb{O}$.

Damit ist in beiden Fällen Gleichung (4.5) erfüllt. Von der Gültigkeit von (4.7) sowie der übrigen technischen Voraussetzungen von Lemma 4.3 überzeugt man sich leicht, so dass nach Satz 4.4 die gemäß (4.9), (4.11) und (4.10) aus $\mathrm{trz}^{\mathrm{L}}$ und $\mathrm{trz}^{\mathrm{D}}$ konstruierte Funktion trs die verwendete Indexform zu einer Kompositionsform ergänzt. □

Bemerkung 5.37. Auf die Erzeugung von Zuständen, die unzulässige Konstellationen erzeugen, sollte wieder verzichtet werden, da diese keinen Einfluss auf das Endergebnis haben werden. Die im letzten Satz verwendeten Transformationsfunktionen vereinfachen sich damit zu

$$\begin{aligned} \mathrm{trz}^{\mathrm{L}} &: ((i, w), C_2) \mapsto (i \cup \hat{C}_2, w + |\hat{C}_2|) \cdot [i \cup \hat{C}_2 \text{ unabh. in } T_2] \\ \mathrm{trz}^{\mathrm{D}} &: ((i, w), v) \mapsto (i - v, w). \end{aligned}$$

Der Ausdruck $z \cdot [p]$ bezeichnet die leere Menge, falls die Aussage p falsch ist, und eine einelementige Zustandsmenge mit dem Zustand z, falls die Aussage wahr ist. Die Bewertung $\mathbb{O}$ für die unzulässigen Konstellationen wird in diesem modifizierten Algorithmus gar nicht mehr erzeugt, so dass darauf auch in den Transformationsschritten keine Rücksicht genommen werden muss.

Satz 5.38. *Die speziellen Transformationen für einen Kantenkompositionsalgorithmus vom Typ B können wie folgt angegeben werden:*

$$\begin{aligned} \mathrm{trs}^{\mathrm{V}} &: ((i, w), v) \mapsto (i \cup \{v\}, w + 1) \uplus (i, w) \\ \mathrm{trs}^{\mathrm{E}} &: ((i, w), e) \mapsto (i, w) \cdot [|i \cap e| < 2] \end{aligned}$$

Beweis. Es muss (4.13) gezeigt werden:

1. Im Falle eines Knotenschrittes ist $\mathcal{C}_2 = \{\{(v, 1)\}, \{(v, 2)\}\}$. Da wir keine Zustände mit Index $\mathbb{O}$ mehr erzeugen, repräsentiert i zulässige Konstellationen, das heißt unabhängige Mengen. In diesem Fall ist sowohl i als auch $i \cup \{v\}$ unabhängig in T_2, da v isoliert ist, und es gilt:

$$\begin{aligned} \mathrm{trs}^{\mathrm{L}}((i, w), T_2) &= \mathrm{trz}^{\mathrm{L}}((i, w), \{(v, 1)\}) \uplus \mathrm{trz}^{\mathrm{L}}((i, w), \{(v, 2)\}) \\ &= (i \cup \{v\}, w + |\{v\}|) \uplus (i \cup \emptyset, w + |\emptyset|) \\ &= (i \cup \{v\}, w + 1) \uplus (i, w) \\ &= \mathrm{trs}^{\mathrm{V}}((i, w), v) \end{aligned}$$

2. Im Fall eines Kantenschrittes hat C_2 die Form $\{(e,1)\}$ und es gilt wegen $\hat{C}_2 = \emptyset$:

$$\begin{aligned}\mathrm{trs}^{\mathrm{L}}((i,w),T_2) &= \mathrm{trz}^{\mathrm{L}}((i,w),\{(e,1)\}) \\ &= (i, w+|\emptyset|)\cdot[i \cup \emptyset \text{ unabh. in } T_2] \\ &= (i,w)\cdot[|i\cap e|<2] \\ &= \mathrm{trs}^{\mathrm{E}}((i,w),e)\end{aligned}$$

□

5.5.2 Unabhängige Knotenmenge maximaler Mächtigkeit

Problem

Es soll eine unabhängige Knotenmenge maximaler Mächtigkeit bestimmt werden. Diese Mächtigkeit ist gleich der Unabhängigkeitszahl.

Komposition

Der Ansatz ist fast identisch zu dem bei der Bestimmung der Unabhängigkeitszahl verwendeten. Der Wertebereich muss angepasst werden, indem dessen Elemente statt der Mächtigkeit der repräsentierten Knotenmengen einen konkreten Repräsentanten darstellen. Ähnlich wie bei der Konstruktion von zulässigen Färbungen wird die Kommutativität der Operation $\oplus$ durch Verwendung einer beliebigen, also zum Beispiel der lexikographischen Ordnung $\prec$ der Elemente von $\mathcal{W}$ sichergestellt.

Wir interessieren uns diesmal von vornherein nur für eine Umsetzung in einen Kantenkompositionsalgorithmus und verzichten deshalb darauf, zunächst einen besonderen Wert $\mathbb{O}$ für unzulässige Konstellationen einzuführen, nur um ihn dann zum Schluss in Analogie zu Bemerkung 5.37 zu ignorieren.

Analog zu den entsprechenden Aussagen bei der Unabhängigkeitszahl kann man sich davon überzeugen, dass das aus

$$\begin{aligned}
c_{\mathrm{V}} &= 2 \\
c_{\mathrm{E}} &= 1 \\
\mathcal{W} &= 2^V \\
\oplus &: (w_1,w_2) \mapsto \begin{cases} w_1 & \text{falls} \quad |w_1|>|w_2| \quad \text{oder} \quad |w_1|=|w_2| \wedge w_1 \prec w_2 \\ w_2 & \text{sonst} \end{cases} \\
O &= \emptyset \\
\mathrm{val} &: C \mapsto \begin{cases} \hat{C} & \text{falls } \hat{C} \text{ unabh.} \\ \emptyset & \text{sonst} \end{cases} \\
\mathrm{ind} &: (C,A) \mapsto \hat{C}|_A \\
\mathrm{trs}^{\mathrm{V}} &: ((i,w),v) \mapsto (i+v, w+v) \uplus (i,w) \\
\mathrm{trs}^{\mathrm{E}} &: ((i,w),e) \mapsto (i,w)\cdot[|e\cap i|<2] \\
\mathrm{trs}^{\mathrm{D}} &: ((i,w),v) \mapsto (i-v,w)
\end{aligned}$$

konstruierte Tupel $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{trs})$ eine Kompositionsform zur Berechnung einer maximalen unabhängigen Menge in einem Hypergraphen darstellt.

5.5.3 Unabhängigkeitspolynom

Problem

Das Polynom $u_H(x) \in \mathbb{Z}[x]$, bei dem der Koeffizient $[x^k]u_H(x)$ die Zahl der unabhängigen Mengen der Mächtigkeit k im Hypergraphen H angibt, wird als *Unabhängigkeitspolynom* bezeichnet.

Komposition

Das Unabhängigkeitspolynom eines Hypergraphen kann völlig analog zur Unabhängigkeitszahl bzw. zur Bestimmung einer maximalen unabhängigen Menge durch einen Kantenkompositionsalgorithmus vom Typ B berechnet werden. Der Index besteht wie bei diesen aus Teilmengen der Knotenmenge. Werte sind hier jedoch univariate Polynome mit ganzzahligen Koeffizienten. Die Addition von Werten ist die normale Polynomaddition.

Wie im vorangegangenen Abschnitt sollen hier nur die zur Konstruktion einer Kantenkompositionsform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val}, \mathrm{ind}, \mathrm{trs})$ notwendigen Elemente angegeben werden:

$$\begin{aligned}
\mathcal{W} &= \mathbb{Z}[x] \\
\oplus &: (w_1, w_2) \mapsto w_1 + w_2 \\
O &= 0 \\
\mathrm{val} &: C \mapsto x^{|\hat{C}|} \cdot [\hat{C} \text{ unabh.}] \\
\mathrm{ind} &: (C, A) \mapsto \hat{C}|_A \\
\mathrm{trs}^{\mathrm{V}} &: ((i, w), v) \mapsto (i + v, w\,x) \uplus (i, w) \\
\mathrm{trs}^{\mathrm{E}} &: ((i, w), e) \mapsto (i, w) \cdot [|e \cap i| < 2] \\
\mathrm{trs}^{\mathrm{D}} &: ((i, w), v) \mapsto (i - v, w)
\end{aligned}$$

Die notwendigen Nachweise erfolgen völlig analog zum Abschnitt 5.5.1. Zur Begründung der Korrektheit der Funktionen $\mathrm{trs}^{\mathrm{V}}$ und $\mathrm{trs}^{\mathrm{E}}$ sei lediglich angemerkt, dass im Knotenaktivierungsschritt aus einem Zustand wie gehabt zwei Nachfolgezustände entstehen. Im ersten Fall wird der Knoten in die unabhängige Knotenmenge aufgenommen. Damit erhöht sich für jede vom Zustand repräsentierte Konstellation die Zahl der Knoten in der unabhängigen Menge um 1, was einer Multiplikation des Polynoms mit x entspricht. Im zweiten Fall kommt der Knoten nicht in die unabhängige Knotenmenge, an der Zahl der unabhängigen Knoten ändert sich nichts, der Wert wird also unverändert übernommen.

5.6 Zuverlässigkeitskenngrößen

Die Definition des Begriffs „Zuverlässigkeit“ sowie verschiedener Zuverlässigkeitskenngrößen in netzartigen Strukturen war und ist der Gegenstand vieler Arbeiten (vgl. dazu [AlF91, Bal77,

PoS86, PoS90, PST92, Ros77, SaW82, Val77] u.v.a.m.). Im vorliegenden Abschnitt wird ein netzartiges System durch einen stochastischen Graphen beschrieben. Dieser ist ein gewöhnlicher Graph, dessen Kanten und Knoten bestimmte Bewertungen im Bereich $[0, 1]$ haben. Diese wird gewöhnlich als die *Funktionswahrscheinlichkeit* des Elements interpretiert, also die Wahrscheinlichkeit dafür, dass sich das Element in einem Zustand befindet, in dem es die ihm zugedachte Funktion fehlerfrei ausführen kann. Ihr Komplement heißt *Ausfallwahrscheinlichkeit*. Diese vage Formulierung des „Funktionierens“ hat eine ausreichende Flexibilität des Modells bei der Anpassung an eine konkrete Situation zur Folge; die Einfachheit des Modells gestattet andererseits die tatsächliche Durchführung von Berechnungen in Netzen mit bis zu mehreren Tausend Elementen. Die Zuverlässigkeit des Gesamtsystems wird in diesem Modell durch die *Zusammenhangswahrscheinlichkeit* des stochastischen Graphen und verwandte Größen beschrieben.

In diesem Abschnitt werden einige Anwendungen der Kompositionsmethode in diesem Bereich vorgestellt. Obwohl die resultierenden Algorithmen insbesondere zum Ende des Abschnitts hin etwas komplexer als die bislang behandelten Beispiele werden, bleiben sie immer noch überschaubar und sind dabei trotz zum Teil erheblich besserer Leistung keinesfalls komplizierter als die in der Literatur vorgeschlagenen (vgl. zum Beispiel [CaL96, YLK02, LMC00]). Das Gleiche gilt im Übrigen auch für verschiedene Verallgemeinerungen und ähnliche Aufgabenstellungen, wie zum Beispiel die Berechnung von Rekonstruktionswahrscheinlichkeiten ([LYCK99]) oder der Restzusammenhangswahrscheinlichkeit ([BoS91]), auf die hier nicht weiter eingegangen werden kann.

5.6.1 Zusammenhangswahrscheinlichkeit

Problem

Gegeben sei ein Graph $G = (V, E) \neq (\emptyset, \emptyset)$ und eine Funktion $p : E \to [0, 1] \subset \mathbb{R}$, $e \mapsto p_e$. Die Größe

$$R(G) \quad = \quad \sum_{F \subseteq E} \left(\prod_{e \in F} p_e \cdot \prod_{e \in E - F} (1 - p_e) \right) [k(G{:}F) = 1] \tag{5.27}$$

wird als *Zusammenhangswahrscheinlichkeit* von G bezeichnet. Sie kann wie folgt interpretiert werden: Bezeichne p_e die Wahrscheinlichkeit dafür, dass die Kante e in einem funktionsfähigen Zustand ist. Unter der Annahme, dass Kantenausfälle stochastisch unabhängig voneinander sind, ist $R(G)$ die Wahrscheinlichkeit dafür, dass es zwischen jedem Knotenpaar im Graphen einen Weg gibt, der vollständig aus funktionstüchtigen Kanten besteht.

Umgekehrt kann die Zusammenhangswahrscheinlichkeit auch als die letztgenannte Wahrscheinlichkeit definiert werden. Die Formel (5.27) ergibt sich dann über den Satz von der totalen Wahrscheinlichkeit aus der Summe der Auftrittswahrscheinlichkeiten derjenigen Fälle, die zu einem zusammenhängenden Graphen, das heißt zu einem „Restgraphen“ mit einer einzigen Komponente führen. Diese Auftrittswahrscheinlichkeiten ergeben sich als Produkt des Produktes der Ausfallwahrscheinlichkeiten der im konkreten Einzelfall ausgefallenen Kanten und des Produktes der Funktionswahrscheinlichkeiten der nicht ausgefallenen Kanten.

Für die Zusammenhangswahrscheinlichkeit ist die Kantendekompositionsformel

$$\begin{aligned} R(G) &= p_e\, R(G_e) + (1 - p_e)\, R(G_{-e}) \\ R(\overline{K}_n) &= [n \leq 1], \end{aligned} \tag{5.28}$$

bekannt. Desweiteren gibt es für $G = G_1 \boxplus_K G_2$ die Formeln

$$R(G) = \sum_{\pi_1, \pi_2 \in \mathcal{P}(K)} a(\pi_1, \pi_2)\, R(G_1, \pi_1)\, R(G_2, \pi_2) \tag{5.29}$$

mit der zur Maximumsbildung im Partitionsverband (Möbius-)inversen Funktion a und

$$R(G) = \sum_{\pi \in \mathcal{P}(K)} P(G_1, \pi)\, R(G_2, \pi), \tag{5.30}$$

die auch als *Splittingformeln* bezeichnet werden. In diesen Formeln stehen $R(G_1, \pi_1)$ bzw. $R(G_2, \pi_2)$ für die Zusammenhangswahrscheinlichkeit der stochastischen Graphen, die durch Kontraktion der Partition π_1 in G_1 bzw. π_2 in G_2 gewonnen wird, und $P(G_1, \pi)$ für die Wahrscheinlichkeit dafür, dass G_1 die Partition π in K induziert.

Splitting und Komposition

Die Kantendekompositionsformel (5.28) passt mit $a(e) = p_e$, $b(e) = 1 - p_e$, $r(n) = [n \leq 1]$ in das Schema (5.1), so dass die in Abschnitt 5.1 beschriebenen Splitting- und Kompositionsformen verwendet werden können.

5.6.2 Zuverlässigkeitspolynom

Problem

Das *Zuverlässigkeitspolynom* (*in P-Form*) kann als eine Variante der Berechnung der Zusammenhangswahrscheinlichkeit angesehen werden. Anstelle möglicherweise individuell unterschiedlicher, aber numerisch bekannter Ausfallwahrscheinlichkeiten, finden hier Kantenausfälle mit identischer, aber unbekannter Wahrscheinlichkeit x statt. Knotenausfälle werden nicht berücksichtigt. Das Ergebnis ist dementsprechend ein univariates Polynom $p \in \mathbb{R}[x]$.

Splitting und Komposition

Das Zuverlässigkeitspolynom in P-Form hat die Dekompositionsformel

$$\begin{aligned} R(G) &= x\, R(G_e) + (1 - x)\, R(G_{-e}) \\ R(\overline{K}_n) &= [n \leq 1]. \end{aligned} \tag{5.31}$$

Diese Dekompositionsformel ist eine Spezialisierung des in Abschnitt 5.4 beschriebenen Negami-Polynoms und kann wie dieses mit Hilfe des in Abschnitt 5.1 beschriebenen Schemas behandelt werden. Dabei ist $a(e) = x$, $b(e) = 1 - x$ und $r(n) = [n \leq 1]$ zu setzen.

Die so genannte *N-Form* des Zuverlässigkeitspolynoms hat die Dekompositionsformel

$$\begin{aligned} R^{\mathrm{N}}(G) &= x\, R^{\mathrm{N}}(G_e) + R^{\mathrm{N}}(G_{-e}) \\ R^{\mathrm{N}}(\overline{K}_n) &= [n \leq 1]. \end{aligned}$$

Auch dieses Problem passt in das Schema (5.1). Durch die weggefallene Multiplikation mit $1 - x$ reduziert sich der Rechenaufwand in der Praxis auf etwa die Hälfte.

5.6.3 K-Zusammenhangswahrscheinlichkeit

Problem

Die Berechnung der K-Zusammenhangswahrscheinlichkeit $R^{\mathrm{K}}(G)$ ist eine Verallgemeinerung der Bestimmung der Zusammenhangswahrscheinlichkeit. Während bei letzterer der „Erfolgsfall“ durch funktionierende Verbindungen zwischen allen Netzknoten charakterisiert wird, muss bei der K-Zusammenhangswahrscheinlichkeit lediglich eine bestimmte Teilmenge $K \subseteq V$ der Knoten verbunden sein. Der Fall $K = V$ ist dabei explizit erlaubt und entspricht der Zusammenhangswahrscheinlichkeit. Für praktische Anwendungen ist jedoch häufig der Spezialfall $|K| = 2$ interessant, insbesondere wenn zusätzlich noch Knotenausfälle möglich sind. Knotenausfälle sollen dabei untereinander stochastisch unabhängig sein, allerdings soll ein Knotenausfall zur Folge haben, dass alle inzidenten Kanten ebenfalls als ausgefallen zählen.

Die Knoten aus K werden als *Terminalknoten* bezeichnet. Sie haben trotz der gleichen Bezeichnung „K“ nichts mit den Kontaktknoten von Kontaktgraphen zu tun.

Bemerkung 5.39. Es ist üblich, sich bei der Behandlung des Problems auf ausfallfreie Terminalknoten zu beschränken, also $p_v = 1$ für alle $v \in K$ zu fordern. Das ist keine wesentliche Einschränkung, da für beliebige stochastische Graphen G mit Terminalknoten K gilt:

$$R^{\mathrm{K}}(G) = R^{\mathrm{K}}(G^*) \prod_{v \in K} p_v$$

Dabei bezeichnet G^* denjenigen stochastischen Graphen, der aus G gewonnen wird, indem die Funktionswahrscheinlichkeiten der Terminalknoten auf 1 gesetzt werden.

Die Berechnung der K-Zusammenhangswahrscheinlichkeit ist NP-schwierig, selbst wenn man sich auf die Spezialfälle $|K| = 2$ und $K = V$ in planaren Graphen beschränkt [Bal77]. Für einige wenige, in der Regel sehr dünne Strukturen sind polynomiale Algorithmen bekannt, die auf Reduktionen beruhen (vgl. dazu [PoS86, PoS90, PST92] und andere). Die bislang bekannten Reduktionen sind allerdings für dichte Graphen kaum anwendbar. Insbesondere sind bereits kleine Graphen mit einem vollständigen Graphen auf sechs Knoten als Minor mit keinem dieser Reduktionsalgorithmen ohne weitere Hilfsmittel berechenbar.

Es gibt zahlreiche praktisch interessante Modifikationen des Problems, zu deren Lösung aber auch in aktuellen Arbeiten (vgl. zum Beispiel [CaP01]) immer noch Methoden ähnlich der im Abschnitt 3.2 beschriebenen Dekomposition verwendet werden.

Indexform

Anders als in den vorangegangenen beiden Beispielen dieses Abschnitts über Zuverlässigkeitskenngrößen kann das Problem der K-Zusammenhangswahrscheinlichkeit nicht auf das in Abschnitt 5.1 beschriebene Schema zurückgeführt werden. Der Grund dafür sind die Knoten, die nicht mehr „anonym“ sind, sondern in den Varianten „intakt“ (Label „1“) und „ausgefallen“ (Label „2“) vorkommen. Die Knotenausfälle legen zusammen mit den ebenfalls „binären“ Kantenausfällen die Verwendung des $(2,2)$-Konstellationsraumes nahe. Die Terminal/Nichtterminal-Eigenschaft der Knoten ist nicht beeinflussbar, braucht also nicht durch die Konstellationen berücksichtigt zu werden.

Definition 5.40. Sei A eine Menge. Eine *markierte ausfallbehaftete Partition* ist entweder ein bestimmtes Element $\mathbb{O}$ oder ein Paar (π, m), bei dem $\pi \in \mathcal{P}(B)$, $B \subseteq A$ eine gewöhnliche Mengenpartition der Menge B und $m \subseteq \pi$ eine Teilmenge der Blöcke von π bezeichnet. Diejenigen Blöcke von π, die in m enthalten sind, heißen *markiert*, die anderen *unmarkiert*. Die Knoten aus $B - A$ sollen *ausgefallen* genannt werden. Die Menge aller markierten ausfallbehafteten Partitionen einer Menge A wird mit $\bar{\mathcal{P}}(A)$ bezeichnet.

Die im Abschnitt 2.1 beschriebenen Operationen können auf markierte ausfallbehaftete Partitionen erweitert werden, indem die entsprechende Operation auf π durchgeführt und m angepasst wird. Das gilt auch für die Verschmelzung zweier markierter ausfallbehafteter Partitionen. In diesem Fall werden die markierten Blöcke des Ergebnisses von denjenigen Blöcken gebildet, die Knoten enthalten, die in wenigstens einem der beiden Operanden bereits zu markierten Blöcken gehörten. Das besondere Element $\mathbb{O}$ soll sich neutral bezüglich der Verschmelzung verhalten.

Die folgende Abhandlung setzt $G = (V, E) \neq (\emptyset, \emptyset)$, $|K| > 1$ und $p_v = 1$ für alle $v \in K$ voraus. Diese Einschränkungen sind nicht wesentlich. Zur übersichtlicheren Darstellung sei die folgende Abkürzung für eine Konstellation $C \in \mathcal{C}_G(2,2)$ eines stochastischen Graphen $G = (V, E)$ mit der Terminalmenge K eingeführt:

$$t(C) \quad = \quad \prod_{\substack{v \in V-K \\ C(v)=1}} p_v \prod_{\substack{v \in V-K \\ C(v)=2}} (1-p_v) \prod_{\substack{e \in E \\ C(e)=1}} p_e \prod_{\substack{e \in E \\ C(e)=2}} (1-p_e)$$

Definition 5.41. Sei $G = (V, E)$ ein stochastischer Graph mit den Terminalknoten K. Bezeichne $A \subseteq V$ eine Teilmenge seiner Knotenmenge, die sowohl Terminal- als auch Nichtterminalknoten enthalten kann, und $C \in \mathcal{C}_G(2,2)$ eine $(2,2)$-Konstellation. Die Abkürzung

$$\mathrm{rem}(C) \quad = \quad G[\{v \in V : C(v) = 1\} \cup K]{:}\{e \in E : C(e) = 1 \wedge (C(u) = 1 \forall u \in e)\}$$

soll denjenigen Graphen bezeichnen, der aus den intakten Knoten und Kanten von G besteht, und

$$k^{'}(C) \quad = \quad |\{(V_j, E_j) \in \mathrm{comp}(\mathrm{rem}(C)) : V_j \cap K \neq \emptyset\}|$$

die Zahl der Komponenten von $\mathrm{rem}(C)$, die Terminalknoten enthalten.

Bezeichne $\{(V_j, E_j)\}_{j=1}^{k(\mathrm{rem}(C))}$ die Zusammenhangskomponenten von $\mathrm{rem}(C)$ und $B_j = V_j \cap A$. Dann stellt

$$\mathrm{part}'(C, A) \quad = \quad \Big(\bigcup_{\substack{j \in \{1,\ldots,k(\mathrm{rem}(C))\} \\ B_j \neq \emptyset}} \{B_j\}, \bigcup_{\substack{j \in \{1,\ldots,k(\mathrm{rem}(C))\} \\ B_j \neq \emptyset \\ V_j \cap K \neq \emptyset}} \{B_j\} \Big)$$

diejenige ausfallbehaftete Partition (π, m) dar, bei der $\pi = \mathrm{part}(\mathrm{rem}(C), A)$ ist und m diejenigen Blöcke umfasst, die aus Komponenten von $\mathrm{rem}(C)$ entstanden sind, die Terminalknoten enthalten.

Lemma 5.42. *Das Tupel* $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \mathrm{val})$ *mit*

$$\begin{aligned} c_{\mathrm{V}} &= 2 \\ c_{\mathrm{E}} &= 2 \\ \mathcal{W} &= \mathbb{R} \\ O &= 0 \\ \oplus &: (w_1, w_2) \mapsto w_1 + w_2 \\ \mathrm{val} &: t(C)\,[k'(C) = 1] \end{aligned}$$

stellt eine Grundform zur Berechnung der K-Zusammenhangswahrscheinlichkeit dar.

Beweis. Jede Konstellation entspricht einem Ereignis bei der Beobachtung des Ausfallverhaltens des Graphen, das aus den Elementarereignissen „Kantenausfälle“ und „Knotenausfälle“ zusammengesetzt ist. Ein Label „1“ entspricht dabei den funktionierenden Kanten bzw. Knoten, das Label „2“ den ausgefallenen. Da die Kanten- und Knotenausfälle stochastisch unabhängig sind, tritt eine Konstellation $C \in \mathcal{C}_G(2,2)$ mit der Wahrscheinlichkeit $t(C)$ auf. Dabei wird $p_v = 1$ für $v \in K$ berücksichtigt. Durch die Ausfälle entsteht aus dem stochastischen Graphen ein (gewöhnlicher) Untergraph, der wegen $p_v = 1$ alle Terminalknoten enthält. Die Zahl der Komponenten des Restgraphen, die Terminalknoten enthalten, ist durch $k'(C)$ gegeben. Damit ist

$$\bigoplus_{C \in \mathcal{C}_G(2,2)} \mathrm{val}(C) \quad = \quad \sum_{C \in \mathcal{C}_G(2,2)} t(C)\,[k'(C) = 1] \quad = \quad R^{\mathrm{K}}(G)$$

gleich der Wahrscheinlichkeit dafür, dass alle Terminalknoten im Restgraphen in der gleichen Komponente liegen, der Restgraph also K-zusammenhängend ist. □

Komposition

In diesem Abschnitt wird eine Reihe von Kantenkompositionsformen für zunächst leicht veränderte Problemstellungen angegeben, um schließlich zu einer Kantenkompositionsform für die Berechnung der K-Zusammenhangswahrscheinlichkeit zu kommen.

Wir stellen fest, dass die ursprüngliche Aufgabe gelöst werden kann, wenn es gelingt, ein Polynom $p_G(x) \in \mathbb{R}[x]$ so zu bestimmen, dass gilt:

$$p_G(x) \quad = \quad \sum_{C \in \mathcal{C}_G(2,2)} t(C)\, x^{k'(C)}$$

Dann gilt nämlich unter Zuhilfenahme der Funktion $\text{proj} : \mathbb{R}[x] \to \mathbb{R}$, $p \mapsto [x^1]p$:

$$\begin{aligned} \text{proj}(p_G(x)) &= \text{proj}\left(\sum_{C \in \mathcal{C}_G(2,2)} t(C)\, x^{k'(C)}\right) \\ &= \sum_{C \in \mathcal{C}_G(2,2)} t(C)\, [k'(C) = 1] \\ &= R^{\mathrm{K}}(G) \end{aligned}$$

Für die Berechnung von $p_G(x)$ kann eine Grundform gefunden werden, die sich nur wenig von der aus Lemma 5.42 unterscheidet:

Lemma 5.43. *Sei $G = (V, E)$ ein Graph, $C \in \mathcal{C}_G(2,2)$ eine $(2,2)$-Konstellation. Dann stellt das aus*

$$\begin{aligned} c_{\mathrm{V}} &= 2 \\ c_{\mathrm{E}} &= 2 \\ \mathcal{W} &= \mathbb{R}[x] \\ O &= 0 \\ \oplus &: (w_1, w_2) \mapsto w_1 + w_2 \\ \text{val} &: t(C)\, x^{k'(C)} \end{aligned}$$

konstruierte Tupel $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \text{val})$ eine Grundform zur Bestimmung von $p_G(x)$ dar.

Beweis. Analog zum Beweis von 5.42. Der Koeffizient $[x^d]p_G(x)$ gibt die Wahrscheinlichkeit dafür an, dass der Restgraph d Komponenten mit Terminalknoten enthält. □

Satz 5.44. *Die Grundform aus Lemma 5.43 zur Lösung des modifizierten Problems „Berechnung von $p_G(x)$“ kann durch*

$$\begin{aligned} \text{ind} &: (C, A) \mapsto \text{part}'(C, A) \\ \text{trs}^{\mathrm{V}} &: (((\pi, m), w), v) \mapsto \begin{cases} ((\pi + \{v\}, m + \{v\}), w\, x) & \text{falls } v \in K \\ ((\pi + \{v\}, m), w\, p_v) \uplus ((\pi, m), w\,(1 - p_v)) & \text{falls } v \notin K \end{cases} \\ \text{trs}^{\mathrm{E}} &: (((\pi, m), w), (uv)) \mapsto ((\pi, m)_{(uv)}, w\, p_{(uv)}\, x^{-[\pi_u \neq \pi_v \wedge \pi_u \in m \wedge \pi_v \in m]}) \uplus \\ &\qquad\qquad ((\pi, m), w\,(1 - p_{(uv)})) \\ \text{trz}^{\mathrm{D}} &: ((\sigma, w), v) \mapsto (\sigma - v, w) \end{aligned} \tag{5.32}$$

zu einer Kantenkompositionsform $(c_{\mathrm{V}}, c_{\mathrm{E}}, \mathcal{W}, \oplus, \text{val}, \text{ind}, \text{trs})$ erweitert werden.

Beweis. Sei $H_1 + T_2$ eine Kontaktzerlegung eines Graphen H_2, $C_1 \in \mathcal{C}_{H_1}$.

1. Im Knotenaktivierungsschritt ist $T_2 = \{\{v\}, \emptyset, \emptyset\}$ und somit $\mathcal{C}_{T_2} = \{\{(v,1)\}, \{(v,2)\}\}$. Der Knoten v ist isoliert in $H_1 + T_2$. Wir unterscheiden zwei Fälle:

a) v ist Terminalknoten. Für diesen gilt nach Bemerkung 5.39 $p_v = 1$. Die Konstellation $\{(v, 2)\}$ tritt also mit Wahrscheinlichkeit 0 auf. Sie hat demzufolge keinen Einfluss auf das Endergebnis und kann deshalb ignoriert werden. Die Konstellation $\{(v, 1)\}$ entspricht einem intakten Terminalknoten. Da dieser isoliert in $H_1 + T_2$ ist, bildet er in $\text{part}'(C_1 \cup C_2, A_1 + v)$ einen einelementigen Block. Dieser ist markiert, da er einen Terminalknoten (v selbst) enthält. Mit Ausnahme des neuen Blockes stimmt $\text{part}'(C_1 \cup C_2, A_1 + v)$ mit $\text{part}'(C_1, A_1)$ überein.
Da der Definitionsbereich von C_2 die Menge $\{v\}$ ist, unterscheiden sich die Werte $t(C_1)$ und $t(C_1 \cup C_2)$ nur durch den Faktor $p_v = 1$. Dazu kommt ein Faktor x, da sich die Zahl der Blöcke, die Terminalknoten enthalten, durch den neuen markierten Einerblock um eins erhöht.
Es gilt also für eine beliebige Konstellation $C_1 \in \mathcal{C}_{H_1}$ mit $\text{ind}(C_1, A_1) = \text{part}'(C_1, A_1) = (\pi, m)$ und $\text{val}(C_1) = w$:

$$\begin{aligned}
&\biguplus_{C_2 \in \mathcal{C}_{(\{v\},\emptyset,\emptyset)}} (\text{ind}(C_1 \cup C_2, A_1 \cup V_2),\ \text{val}(C_1 \cup C_2)) \\
&\qquad = (\text{ind}(C_1 \cup \{(v,1)\}, A_1 + V_2),\ \text{val}(C_1 \cup \{(v,1)\})) \\
&\qquad = (\text{part}'(C_1 \cup \{(v,1)\}, A_1 + v),\ \text{val}(C_1)\, x) \\
&\qquad = ((\pi + \{v\}, m + \{v\}),\ w\, x) \\
&\qquad = \text{trs}^{\text{V}}(z, v)
\end{aligned}$$

b) v ist Nichtterminalknoten. Sei $C_1 \in \mathcal{C}_{H_1}$ eine beliebige Konstellation mit $\text{ind}(C_1, A_1) = \text{part}'(C_1, A_1) = (\pi, m)$ und $\text{val}(C_1) = w$. Die beiden Konstellationen aus $\mathcal{C}_{T_2}$ werden getrennt behandelt:

i. Die Konstellation $C_{21} = \{(v, 1)\}$ entspricht einem intakten Nichtterminalknoten. Da dieser isoliert in $H_1 + T_2$ ist, bildet er in $\text{part}'(C_1 \cup C_{21}, A + v)$ einen Einerblock. Dieser ist nicht markiert, da er keinen Terminalknoten enthält. Mit Ausnahme des neuen Blockes stimmt $\text{part}'(C_1 \cup C_{21}, A + v)$ mit $\text{part}'(C_1, A)$ überein.
Da der Definitionsbereich von C_{21} die Menge $\{v\}$ ist, unterscheiden sich die Werte $t(C_1)$ und $t(C_1 \cup C_{21})$ nur durch den Faktor, der durch v erzeugt wird, also um p_v, da der Knoten intakt ist. Die Zahl der markieren Blöcke und damit die Potenz von x ändert sich nicht.
Es gilt also:

$$(\text{ind}(C_1 \cup C_{21}, A_1 + V_2),\ \text{val}(C_1 \cup C_{21})) \quad = \quad ((\pi + \{v\}, m),\ w\, p_v)$$

ii. Die Konstellation $C_{22} = \{(v, 2)\}$ entspricht einem ausgefallenen Nichtterminalknoten. Ähnlich wie im vorangegangenen Fall erhält man:

$$(\text{ind}(C_1 \cup C_{22}, A_1 + V_2),\ \text{val}(C_1 \cup C_{22})) \quad = \quad ((\pi, m),\ w\,(1 - p_v))$$

Der Knoten v erscheint dabei nicht mehr in den Blöcken der ausfallbehafteten Partition $\pi \in \mathcal{P}(A_1 \cup V_2)$, ist also ein „ausgefallener“ Knoten von π.

Wegen i. und ii. gilt auch in diesem Fall:

$$\biguplus_{C_2 \in \mathcal{C}_{T_2}} (\text{ind}(C_1 \cup C_2, A_1 \cup V_2),\ \text{val}(C_1 \cup C_2))$$

$$\begin{aligned} &= ((\pi + \{v\}, m), w\, p_v) \uplus ((\pi, m), w\,(1 - p_v)) \\ &= \mathrm{trs}^{\mathrm{V}}(z, v) \end{aligned}$$

2. Im Kantenschritt ist $T_2 = \{e, \{e\}, e\}$ und demzufolge $\mathcal{C}_{T_2} = \{\{(e,1)\}, \{(e,2)\}\}$. Wir unterscheiden zwei Fälle bezüglich der Konstellation $C_2 \in \mathcal{C}_2$:

   a) Die Konstellation $\{(e,2)\}$ entspricht einer ausgefallenen Kante. An der Struktur der induzierten Partition ändert sich nichts, es gilt also $\mathrm{part}'(C_1 \cup C_2, A) = \mathrm{part}'(C_1, A)$. Im Wert ergibt sich ähnlich wie in den Knotenschritten ein Faktor von $(1 - p_e)$ und damit ein Folgezustand

   $$(((\pi, m),\ w\,(1 - p_e)).$$

   b) Die Konstellation $\{(e,1)\}$ entspricht einer intakten Kante. Die induzierte Partition $\mathrm{part}'(C_1 \cup C_2, A)$ entsteht aus $\mathrm{part}'(C_1, A)$ durch Kontraktion von e. Bei dieser Kontraktion kann es vorkommen, dass zwei markierte Blöcke verschmelzen und sich somit die Zahl der markierten Blöcke um eins verringert. Dieser Fall tritt genau dann ein, wenn die beiden Blöcke der Partition, die die Endknoten der Kante enthalten, voneinander verschieden und beide markiert sind. Im Wert kommt analog zu den vorherigen Fällen ein Faktor p_e hinzu, so dass sich als Folgezustand mit $e = (uv)$ ergibt:

   $$((\pi, m)_e,\ w\, p_e\, x^{-[\pi_u \neq \pi_v \wedge \pi_u \in m \wedge \pi_v \in m]})$$

   Aus a) und b) folgt:

$$\begin{aligned} &\biguplus_{C_2 \in \mathcal{C}_{T_2}} (\mathrm{ind}(C_1 \cup C_2, A_1 \cup V_2),\ \mathrm{val}(C_1 \cup C_2)) \\ &\quad = \quad ((\pi, m)_{(uv)},\ w\, p_{(uv)}\, x^{-[\pi_u \neq \pi_v \wedge \pi_u \in m \wedge \pi_v \in m]}) \uplus ((\pi, m), w\,(1 - p_{(uv)})) \\ &\quad = \quad \mathrm{trs}^{\mathrm{E}}(z, e) \end{aligned}$$

3. Es gilt für beliebige Konstellationen $C \in \mathcal{C}$, beliebiges v und A mit $v \in A \subseteq A_1 \cup V_2$:

$$\mathrm{trz}^{\mathrm{D}}((\mathrm{ind}(C, A), \mathrm{val}(C)), v) \quad = \quad (\mathrm{ind}(C, A - v), \mathrm{val}(C))$$

Wegen 1. bis 3. gilt (4.12). Von der Gültigkeit der übrigen, technischen Voraussetzungen des Lemmas 4.3 überzeugt man sich leicht. □

Bemerkung 5.45. Wenn die Endknoten der zu behandelnden Kante im Kantenschritt im gleichen Block liegen, kann einfach ein einziger unveränderter Nachfolgezustand produziert werden, anstelle zweier Zustände mit gleichem Index, deren Werte einmal das p_e-fache und einmal das $(1 - p_e)$-fache des Ausgangswertes sind.

Wir modifizieren das Problem weiter: Dazu wird festgestellt, dass die ursprüngliche Aufgabe gelöst werden kann, wenn es gelingt, ein Polynom $p'_G(x) \in \mathbb{R}[x]$ so zu bestimmen, dass mit der bereits angeführten Funktion $\mathrm{proj} : \mathbb{R}[x] \to \mathbb{R},\ p \mapsto [x^1]p$ gilt:

$$\mathrm{proj}(p'_G(x)) \quad = \quad \mathrm{proj}(p_G(x))$$

Anstelle $p_G(x)$ zu bestimmen, genügt es also, ein Polynom $p'_G(x)$ zu finden, das mit $p_G(x)$ im Koeffizienten vor x^1 übereinstimmt. Ein solches Polynom wird durch folgende Überlegung gefunden:

Die Koeffizienten aller auftretenden Polynome sind nichtnegativ. Beim Zusammenfassen zweier Zustände mit gleichem Index wird bei der Berechnung von $p_G(x)$ die gewöhnliche Polynomaddition verwendet. Es soll der Fall betrachtet werden, dass dabei ein Polynom mit mindestens zwei positiven Koeffizienten $a_k = [x^k]p_G(x)$ und $a_l = [x^l]p_G(x)$ mit $k < l$ entsteht. Im Laufe der weiteren Rechnung wird dieses Polynom (möglicherweise) mit positiven Skalaren multipliziert, mit einer Potenz von x multipliziert oder mit weiteren Polynomen mit nichtnegativen Koeffizienten summiert. Andere Operationen werden nicht verwendet. In jedem Fall entsteht zum Schluss wieder ein Polynom, das mindestens zwei positive Koeffizienten vor x^{k_1} und x^{l_1} mit $l - k = l_1 - k_1$ besitzt. Nun gilt $[x^0]p_G(x) = 0$, da jeder Restgraph noch alle Terminalknoten und damit mindestens eine Komponente mit Terminalknoten enthält. Damit kann k_1 nicht gleich 0 sein, da sonst der positive Koeffizient vor $x^{k_1} = x^0$ zusammen mit anderen nichtnegativen Werten zum Absolutglied von $p_G(x)$ aufsummiert würde, welches aber gleich 0 ist. Demzufolge muss $k_1 \geq 1$ und damit $l_1 \geq 2$ sein. Aufgrund der Definition von proj liefert der Koeffizient von x^{l_1} aber keinen Beitrag für die Berechnung von $R^{\mathrm{K}}(G)$. Es ist somit nicht notwendig, diesen Koeffizienten korrekt zu berechnen, insbesondere wenn ein solcher „Fehler“ zur Vereinfachung der Rechnung beitragen könnte.

Eine Variante eines solchen „produktiven Fehlers“ besteht darin, den Koeffizienten a_l auf 0 zu setzen bzw. gar nicht erst zu erzeugen. Realisiert wird das, indem für die Operation $\oplus$ anstelle der bislang verwendeten gewöhnlichen Polynomaddition die Funktion

$$\oplus \quad : \quad (a_1\, x^{d_1}, a_2\, x^{d_2}) \mapsto \begin{cases} a_1\, x^{d_1} & \text{falls } d_1 < d_2 \\ a_2\, x^{d_2} & \text{falls } d_1 > d_2 \\ (a_1 + a_2)\, x^{d} & \text{falls } d_1 = d_2 = d \end{cases}$$

verwendet wird.

Man überzeugt sich, dass ein solchermaßen modifizierter Kantenkompositionsalgorithmus als Zustandswerte nur Polynome mit höchstens einem positiven Koeffizienten produziert. Damit verringert sich der Aufwand für den Algorithmus beträchtlich: Einerseits braucht pro Wert nur ein reeller Koeffizient und ein ganzzahliger Exponent gespeichert zu werden und andererseits sind die Operationen auf diesen Werten erheblich schneller als Polynomadditionen auf Polynomen mit bis zu $n = |V|$ positiven Koeffizienten.

5.6.4 Approximation der K-Zusammenhangswahrscheinlichkeit

Ein allgemeingültiger Ansatz zur Berechnung von Approximationen soll in diesem Abschnitt speziell auf die Approximation der K-Zusammenhangswahrscheinlichkeit zugeschnitten vorgestellt werden. Die in diesem Abschnitt eingeführten Bezeichnungen sollen durchgängig für den gesamten Abschnitt gelten.

Sei $G = (V, E)$ ein stochastischer Graph, für den mit Hilfe des zuletzt beschriebenen modifizierten Kantenkompositionsalgorithmus ein Polynom $p'_G(x)$ mit $\mathrm{proj}(p'_G(x)) = R^{\mathrm{K}}(G)$ berechnet wurde. Bezeichne $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$ die Menge der Indizes *aller* dabei auftretenden Zustände, unabhängig davon, in welchem Transformationsschritt dieser Zustand benutzt

wurde. Sei index die (bijektive) Funktion

$$\text{index} \;:\; \mathcal{I} \to \{1,\ldots,|\mathcal{I}|\},\; i_j \to j.$$

Bezeichne $a_{k,j}\, x^{d_{k,j}}$ den Wert desjenigen Zustands nach dem k-ten Schritt, der den Index i_j besitzt. Falls kein derartiger Zustand existiert, soll $a_{k,j} = 0$ gelten. Sei a_k der Vektor $(a_{k,j})_{j=1}^{|\mathcal{I}|}$ für $k = 1,\ldots,t$. Die Elemente von a_k sind nach Konstruktion nichtnegativ. Als a_0 wählen wir den Vektor $(a_{0,j})_{j=1}^{|\mathcal{I}|}$ mit dem Einträgen $a_{0,\text{index}(i_\emptyset)} = 1$ für den Index $i_\emptyset = (\pi_\emptyset, m_\emptyset) = (\emptyset, \emptyset)$ und 0 sonst.

Man überzeugt sich wie folgt, dass für $k = 0,\ldots,t$ der Vektor a_k genau die gleiche Information beinhaltet wie die Zustandsmenge $\mathcal{Z}_k$ des modifizierten Algorithmus: Die Position eines Eintrags in a_k kodiert den Index eines Zustands (i, w), der Eintrag selbst liefert den entsprechenden Koeffizienten. Der Exponent des Monoms w ist implizit durch den Zeitpunkt k gegeben, da zu jedem beliebigen Zeitpunkt für einen bestimmten Index höchstens ein Exponent möglich ist.

Bezeichne s_k für $k = 1,\ldots,t$ die bijektive Funktion, die $\mathcal{Z}_k$ in a_k überführt und $\text{tr}_k^{'} : \mathbb{R}_+^{|\mathcal{I}|} \to \mathbb{R}_+^{|\mathcal{I}|}$, $\text{pr}^{'} : \mathbb{R}_+^{|\mathcal{I}|} \to \mathbb{R}_+$ die Funktionen

$$\begin{aligned} \text{tr}_k^{'} &= s_k \circ \text{tr}_k \circ s_{k-1}^{-1} \\ \text{pr}^{'} &= \text{proj} \circ \text{pr} \circ s_t^{-1}. \end{aligned}$$

wobei $\text{tr}_k(\mathcal{Z}) = \text{tr}(\mathcal{Z}, T_k)$ ist. Dann gilt:

$$\begin{aligned} R^{\text{K}}(G) &= \text{proj}(\text{pr}(\text{tr}(\ldots \text{tr}(\mathcal{Z}_0, T_1)\ldots), T_t)) \\ &= (\text{proj} \circ \text{pr} \circ s_t^{-1}) \left((s_t \circ \text{tr}_k \circ s_{t-1}^{-1}) \left(\ldots (s_1 \circ \text{tr}_k \circ s_0^{-1})\,(s_0(\mathcal{Z}_0))\ldots\right)\right) \\ &= \text{pr}^{'}(\text{tr}_t^{'}(\ldots \text{tr}_1^{'}(a_0)\ldots)) \end{aligned} \tag{5.33}$$

Lemma 5.46. *Bezeichne $\|.\|$ die Spaltensummennorm im $\mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ und gleichzeitig die Summennorm im $\mathbb{R}^{|\mathcal{I}|}$. Die Funktionen $\text{pr}^{'}$ und $\text{tr}_k^{'}$ sind lineare Abbildungen, die durch*

$$\text{tr}_k^{'}(a_{k-1}) = Q_k\, a_{k-1} \tag{5.34}$$

und

$$\text{pr}^{'}(a_t) = p^{\text{T}}\, a_t \tag{5.35}$$

für geeignete Matrizen $Q_k \in \mathbb{R}_+^{|\mathcal{I}|\times|\mathcal{I}|}$ und einen Vektor $p = (p_j)_{j=1}^{|\mathcal{I}|} \in \{0,1\}^{|\mathcal{I}|}$ mit

$$\|p\| = 1 \tag{5.36}$$

dargestellt werden können.

Beweis.

1. Die Linearität der Funktionen $\text{tr}_k^{'}$ folgt aus der Kompositionsform aus Satz 5.44 sowie der Tatsache, dass s bijektiv ist. Damit kann $\text{tr}_k^{'}$ als Matrix geschrieben werden.

2. Die letzte der bei der Komposition entstehenden aktiven Knotenmengen ist immer leer. Damit gibt es nur den Index $i_\emptyset$ in $\mathcal{I}_{H_t,\emptyset}$ und somit höchstens einen einzigen Zustand mit positivem Wert. Die Funktion pr' kann also als $\mathrm{pr}'(a_t) = p^{\mathrm{T}}\, a_t$ mit $p = ([j = \mathrm{index}(i_\emptyset)])_{j=1}^{|\mathcal{I}|} \in \{0,1\}^{|\mathcal{I}|}$ geschrieben werden. Damit ist pr' offensichtlich linear. Die Darstellung (5.36) folgt sofort. □

Folgerung 5.47. *Es gilt*

$$R^{\mathrm{K}}(G) \;=\; p^{\mathrm{T}} \cdot Q_t \cdot \ldots \cdot Q_1 \cdot a_0. \tag{5.37}$$

Beweis. Folgt direkt aus (5.33), (5.34) und (5.35). □

Lemma 5.48. *Mit den Bezeichnungen aus Lemma 5.46 gilt*

$$\|Q_k\| \le 1 \tag{5.38}$$

für alle $k = 1, \ldots, t$.

Beweis. Seien $i \in \mathcal{I}$ und $k \in \{1, \ldots, t\}$ fixiert und bezeichne $j = \mathrm{index}(i)$. Die Spalte $q_{k,j} = (q_{k,j,l})_{l=1}^{|\mathcal{I}|}$ von Q_k enthält Einträge, die der Transformation des Wertes $a_{k,j}\, x^{d_{k,j}}$ gemäß Satz 5.44 entsprechen. Die Transformation selbst hängt vom konkreten Schritt ab, in jedem Fall ist aber

$$\|q_{k,j}\| \;=\; \sum_{l=1}^{|\mathcal{I}|} |q_{k,j,l}| \;\le\; 1,$$

da es maximal zwei Nichtnull-Einträge in $q_{k,j}$ gibt: Die Kombinationen $(p_e, 1-p_e)$ (in einem Kantenschritt), $(p_v, 1-p_v)$ (in einem Knotenschritt mit Nichtterminalknoten) oder ein einziger Nichtnull-Eintrag 1 (in einem Knotenschritt mit Terminalknoten). Gleichung (5.38) folgt damit direkt aus der Definition der Spaltensummennorm. □

Satz 5.49. *Sei* $\varepsilon > 0$ *gegeben. Definiere für* $k \in \{1, \ldots, t\}$ *und* $f \in \mathbb{R}_+^{|\mathcal{I}|}$ *mit* $\|f\| \le \varepsilon$*:*

$$R^{\mathrm{K}}_{k,f}(G) \;=\; p^{\mathrm{T}} \cdot Q_t \cdot \ldots \cdot Q_{k+1} \cdot (Q_k \cdot \ldots \cdot Q_1 \cdot a_0 - f)$$

Dann gilt

$$R^{\mathrm{K}}_{k,f}(G) \;\le\; R^{\mathrm{K}}(G) \;\le\; R^{\mathrm{K}}_{k,f}(G) + \varepsilon. \tag{5.39}$$

Beweis. Es gilt:

$$\begin{aligned} R^{\mathrm{K}}(G) - R^{\mathrm{K}}_{k,f}(G) \;&=\; p^{\mathrm{T}} \cdot Q_t \cdot \ldots \cdot Q_1 \cdot a_0 \;-\; p^{\mathrm{T}} \cdot Q_t \cdot \ldots \cdot Q_{k+1} \cdot (Q_k \cdot \ldots \cdot Q_1 \cdot a_0 - f) \\ &=\; p^{\mathrm{T}} \cdot Q_t \cdot \ldots \cdot Q_{l+1} \cdot f \end{aligned}$$

Insbesondere sind die Einträge aller beteiligten Matrizen und Vektoren nichtnegativ. Es gilt also $R^{\mathrm{K}}(G) - R^{\mathrm{K}}_{k,f}(G) \ge 0$ und damit

$$R^{\mathrm{K}}_{k,f}(G) \;\le\; R^{\mathrm{K}}(G). \tag{5.40}$$

Außerdem gilt

$$\begin{aligned} R^{\mathrm{K}}(G) - R^{\mathrm{K}}_{k,f}(G) &= |R^{\mathrm{K}}(G) - R^{\mathrm{K}}_{k,f}(G)| \\ &= |p^{\mathrm{T}} \cdot Q_t \cdot \ldots \cdot Q_{k+1} \cdot f| \\ &\leq \|p^{\mathrm{T}}\| \cdot \|Q_t\| \cdot \ldots \cdot \|Q_{k+1}\| \cdot \|f\| \\ &\overset{(5.38)}{\leq} \|p^{\mathrm{T}}\| \cdot 1 \cdot \ldots \cdot 1 \cdot \varepsilon \\ &\overset{(5.36)}{=} \varepsilon \end{aligned}$$

und somit

$$R^{\mathrm{K}}(G) \leq R^{\mathrm{K}}_{k,f}(G) + \varepsilon. \tag{5.41}$$

Aus (5.40) und (5.41) folgt schließlich die Behauptung. □

Im Folgenden wird angenommen, dass die in der Regel aus Messungen stammenden Eingangsdaten p_e fehlerbehaftet sind und demzufolge auch nur ein Endergebnis mit einer bestimmten Genauigkeit erwartet werden kann. Insbesondere soll der Algorithmus selbst einen systematischen Fehler bis zu einer vorgegebenen Schranke ε verursachen dürfen.

Das Entfernen eines Zustandes $z = (i, c\,x^d) \in \mathcal{Z}_k$ aus der nach dem k-ten Transformationsschritt erzeugten Zustandsmenge $\mathcal{Z}_k$ hat die gleichen Auswirkungen auf den Wert des Ergebnisses wie das Ersetzen dieses Zustandes durch den Zustand $z' = (i, 0)$ mit dem gleichen Index und dem Wert 0. Diese Ersetzung $z \to z'$ entspricht in der „Matrixformulierung“ (5.37) der Subtraktion eines Vektors $f = (f_l)_{l=1}^{|\mathcal{I}|}$, für dessen Einträge $f_l = c \cdot [l = \mathrm{index}(i)]$ gilt.

Satz 5.49 sagt aus, dass die Subtraktion dieses Vektors dazu führt, dass das Endergebnis $R^{\mathrm{K}}_{k,f}(G)$ der ansonsten unverändert durchgeführten Rechnung sich um höchstens c vom „richtigen“ (das heißt, dem mit dem nichtmodifizierten Algorithmus bei gleichermaßen fehlerhaften Eingangsdaten und Rundungen erhaltenen) Ergebnis unterscheidet.

Da ein aus $\mathcal{Z}_k$ entfernter Zustand im folgenden Transformationschritt $k+1$ nicht mehr vorhanden ist, verringert sich der Aufwand für diesen Schritt. Damit kann das Entfernen von Zuständen in der Praxis zur gezielten Beschleunigung des Kompositionsalgorithmus zur Berechnung der K-Zusammenhangswahrscheinlichkeit eingesetzt werden. Ziel ist es, den Gesamtaufwand zu minimieren.

Um ein Ergebnis zu erhalten, das um höchstens ε vom exakten Ergebnis abweicht, kann jede beliebige Menge $Z = \{z_1, \ldots, z_{|Z|}\}$ von Zuständen aus (möglicherweise verschiedenen) Zustandsmengen $\mathcal{Z}_k$ entfernt werden, solange nur $\sum_{(i,c\,x^d)\in Z} c \leq \varepsilon$ ist. Die im Anhang C.2 dargestellten Beispielrechnungen verwenden alle folgende Heuristik: Der zulässige Fehler ε wird gleichmäßig auf die t Transformationen aufgeteilt. Nach jeder Transformation k wird die Zustandsmenge zu $\mathcal{Z}_k = \{(i_1, w_1 x^{d_1}), \ldots, (i_{|\mathcal{Z}_k|}, w_{|\mathcal{Z}_k|} x^{d_{|\mathcal{Z}_k|}})\}$ mit $w_i \leq w_j$ für $i \leq j$ geordnet. Danach werden die Zustände $(i_1, w_1), \ldots, (i_j, w_j)$ mit $j = \max\{j : \sum_{l=1}^{j} w_l \leq \frac{\varepsilon}{t}\}$ entfernt. Diese Wahl ist sicher nicht optimal, bringt jedoch bereits erheblichen Gewinn.

A Einige Zahlenfolgen

Die folgende Tabelle enthält einige Zahlenfolgen, die häufig als obere Schranken für die Mächtigkeiten von Zustandsmengen und damit zur Abschätzung der Komplexität von Kompositionsalgorithmen verwendet werden können.

Die Bell-Zahlen $B(n)$ geben (unter anderem) die Anzahl der Mengenpartitionen einer Menge der Mächtigkeit n an. In Kanten- und Knotenkompositionsalgorithmen, bei denen die Indizes der Zustände aus Mengenpartitionen der aktiven Mengen bestehen, liefert die Bell-Zahl $B(n)$ eine obere Schranke für die Zahl der gleichzeitig auftretenden Zustände bei der Berechnung von Graphen der Wegweite $n-1$ bei geeigneter Wahl der Kontaktzerlegung.

Ist bei solchen Problemen zusätzlich noch bekannt, dass nur *nichtkreuzende* Partitionen auftreten, wie es zum Beispiel häufig bei planaren Graphen der Fall ist, so kann die Catalan-Zahl $C(n)$ als obere Schranke verwendet werden.

Die Folgen e^n bzw. $n!$ wurden ebenfalls in der Tabelle dargestellt, um eine Vorstellung von der Größenordnung der Zahlen zu vermitteln.

n	1	2	3	4	5	6	7	8	9	10
2^n	2	4	8	16	32	64	128	256	512	1 024
$\lceil e^n \rceil$	3	8	21	55	149	404	1 097	2 981	8 104	22 027
$C(n)$	1	2	5	14	42	132	429	1 430	4 862	16 796
$B(n)$	1	2	5	15	52	203	877	4 140	21 147	115 975
$n!$	1	2	6	24	120	720	5 040	40 320	362 880	3 628 800

11	12	13	14	15
2 048	4 096	8 192	16 384	32 768
59 875	162 755	442 414	1 202 605	3 269 018
58 786	208 012	742 900	2 674 440	9 694 845
678 570	4 213 597	27 644 437	190 899 322	1 382 958 545
39 916 800	479 001 600	479 001 600	87 178 291 200	1 307 674 368 000

20	25	30
1 048 576	33 554 432	1 073 741 824
485 165 196	72 004 899 338	10 686 474 581 525
6 564 120 420	4 861 946 401 452	3 814 986 502 092 304
51 724 158 235 372	4 638 590 332 229 999 353	846 749 014 511 809 332 450 147
$\approx 2.4 \cdot 10^{18}$	$\approx 1.3 \cdot 10^{25}$	$\approx 2.7 \cdot 10^{32}$

B Bestimmen von Kontaktzerlegungen

In der bisherigen Beschreibung der Kompositionsmethode wurde der Schritt der Bestimmung einer Kontaktzerlegung bewusst ausgeschlossen. Das soll in diesem Abschnitt nachgeholt werden.

Die Kompositionsalgorithmen arbeiten mit jeder Kontaktzerlegung des zu berechnenden Hypergraphen korrekt. Allerdings hängt das Laufzeitverhalten stark von der gewählten Kontaktzerlegung ab, so dass es sich anbietet, die Freiheit bei der Wahl der Kontaktzerlegung zur Beschleunigung der Algorithmen zu verwenden. Die Wahl folgt also rein praktischen Gesichtspunkten, so dass es hier ausreichen soll, Kontaktzerlegungen für die tatsächlich eingesetzten Knoten- bzw. Kantenkompositionen in gewöhnlichen Graphen, nicht aber für den allgemeinen Fall zu behandeln.

Optimierungsziele

Eigentliches Ziel der Optimierung bei der Wahl der Kontaktzerlegung ist ein minimaler Gesamtaufwand des resultierenden Algorithmus für einen gegebenen Graphen G. Dieser Gesamtaufwand ist allerdings nicht nur von der Wahl der Kontaktzerlegung, sondern auch von der konkreten zu berechnenden Graphenkenngröße abhängig. Darüber hinaus ist er häufig nicht exakt bestimmbar, ohne die Rechnung tatsächlich durchzuführen. Damit bleibt nur die Verwendung von Optimierungszielen zweiter Ordnung. Das einzige neben der Zahl der Teile (die aber für Kanten- und Knotenkomposition fixiert ist) in die Abschätzung des Gesamtaufwandes eingehende von der Kontaktzerlegung abhängige Kriterium ist die Weite der Kontaktzerlegung. Es bietet sich also zunächst an, diese zu minimieren.

Das Minimum der Weiten aller Kanten- bzw. Knotenkompositionsfolgen eines Graphen $G = (V, E)$ ist gleich seiner Wegweite $\mathrm{pw}(G)$. Allerdings ist das Bestimmen der Wegweite eines Graphen selbst ein NP-schwieriges Problem. Ist bekannt, dass ein Graph die Wegweite d hat, kann eine Wegzerlegung der Weite d im allgemeinen Fall in $\mathcal{O}(|V|^{d+2})$ Operationen (vgl. [ACP87]) bzw. $\mathcal{O}(|V|)$ mit enormen konstanten Faktor, der exponentiell in d ist, (vgl. [Bod96]) exakt gelöst werden kann. Auch für leichte Modifikationen der Aufgabenstellung wie zum Beispiel der Bestimmung der Schnittweite sind nur vergleichbar „gute“ Ergebnisse bekannt (vgl. [Thi00, TSB01]): Der Aufwand ist allen diesen Fällen für praktische Zwecke viel zu hoch, so dass weitere Abstriche am Ziel der Optimierung gemacht werden müssen: Das Ziel lautet nun, mit akzeptablem Aufwand eine Wegzerlegung des Graphen zu bestimmen, deren Weite nicht wesentlich von der Wegweite des Graphen abweicht – mit anderen Worten: wir suchen eine schnelle Heuristik.

Die verwendeten Methoden zur Bestimmung einer Kontaktzerlegung können in zwei Klassen eingeteilt werden: „globale“ und „lokale“. In beiden Fällen können konkrete Aussagen weder bezüglich der Qualität der erreichten Approximation der Wegweite noch bezüglich der tatsächlichen Laufzeit der Verfahren gemacht werden. Keine der genannten Methoden garantiert,

optimale Lösungen zu finden. Im Gegenteil, es ist in allen Fällen leicht möglich, Graphen zu konstruieren, in denen die gefundene Näherung beliebig weit vom Optimum entfernt ist. Unter Annahme von gewissen Regularitätsforderungen (zum Beispiel Planarität und beschränkter Knotengrad, Existenz von Minoren bestimmter Struktur), die in der Praxis oft schon aus technischen Gründen gegeben sein müssen (Einbettung des Netzes in die Erdoberfläche, beschränkte Anzahl von Schnittstellen der Geräte, bestimmte Vorschriften zum Netzaufbau, usw.), können allerdings in manchen Fällen Fehlerabschätzungen gefunden werden.

Zum Teil können zur Lösung Ansätze zur Verteilung von Aufgaben auf Parallelrechnern verwendet werden, ein Überblick dazu ist in [Els97] zu finden. Zwei dieser Methoden sollen hier kurz angesprochen werden.

Globale Methoden - Spektrale Bisektion

Fiedler hat in [Fie98] einen approximativen Algorithmus zur Bisektion eines Graphen vorgeschlagen, der den Graphen in zwei gleich große Teile (bezüglich der Knotenzahl) teilt und dabei versucht, die Zahl der verbindenden Kanten zu minimieren.

Dieses Problem lässt sich als diskrete Optimierungsaufgabe formulieren, deren Relaxation auf reelle Variablen direkt gelöst werden kann. Dazu wird zunächst angenommen, dass es sich um einen Graphen $G = (V, E)$ mit einer geraden Anzahl von Knoten $n = |V|$ handelt, wobei die Knoten mit den Zahlen $\{1, \ldots, n\}$ identifiziert werden. Einträge in einem *Indexvektor* $x \in \{-1, 1\}^n$ kennzeichnen die Zugehörigkeit eines Knotens zu einem der beiden zu erzeugenden Teile. Die Funktion

$$f(x) = \frac{1}{4} \sum_{(uv)\in E} (x_u - x_v)^2$$

liefert die Zahl der geschnittenen Kanten, da $|x_u - x_v| = 0$ für Kanten (uv) mit Endknoten im gleichen Teil und $|x_u - x_v| = 2$ für Kanten mit Endknoten in verschiedenen Teilen gilt.

Durch eine Reihe von Umformungen lässt sich daraus unter Zuhilfenahme der Admittanzmatrix L des Graphen und eines Vektors $e = (1, \ldots, 1)^T \in \mathbb{Z}^n$ das Ziel der Minimierung der geschnittenen Kanten als

$$\begin{aligned} f(x) &= \frac{1}{4} x^T L x \stackrel{!}{=} \min \\ x &\in \{\pm 1\}^n \\ x^T e &= 0 \end{aligned}$$

schreiben. Durch Relaxation $x \to z \in \mathbb{R}^n$ mit $\|x\| = \|z\|$ lässt sich unter Ausnutzung der Orthogonalität der Eigenvektoren von L ein Minimum von $f^*(z) = \lambda_2\, n$ für $z = \sqrt{n}\, u_2$ finden, wobei λ_2 der zweitkleinste Eigenwert von L und u_2 ein zugehöriger normierter Eigenvektor ist, der für $\lambda_3 > \lambda_2$ sogar noch bis auf einen Faktor ± 1 eindeutig bestimmt ist. Für die Details der Rechnung sei auf die sehr ausführliche Darstellung in [Els97] verwiesen.

Die Frage, wie aus der angegebenen Lösung des relaxierten Problems eine gute Bisektion des Graphen gefunden werden kann, wird in der Regel durch $x_i = -1$, falls $z_i < z'$, und $x_i = 1$, falls $z_i > z'$, mit dem Median z' der z_i beantwortet. Vgl. dazu auch [CCS97].

Da diese Methode aus einer linearen Ordnung (der Einträge im besagten Eigenvektor) durch „Halbierung“ eine „Halbierung“ des Graphen erzeugt, liegt es nahe, direkt die gesamte Reihenfolge der Knoten entsprechend der Werte z_i aus dem Eigenvektor zu übernehmen.

In der Praxis haben sich durch diesen Ansatz brauchbare Ergebnisse ergeben, insbesondere wenn eine „lokale Nachbesserung“ zum Beispiel mit der im folgenden Abschnitt beschriebenen Methode von Kernigan und Lin durchgeführt wurde. Die Nachbesserung scheint vor allem deshalb notwendig zu sein, weil die spektrale Bisektion eigentlich versucht, Kantenschnitte zu minimieren. Die Kompositionsmethode benötigt aber kleine Knotenschnitte, die nicht notwendigerweise mit kleinen Kantenschnitten übereinstimmen, auch wenn sie häufig „in der Nähe“ liegen.

Lokale Methoden - Kernigan/Lin

Kernigan und Lin haben bereits 1970 in [KeL70] eine heuristische Methode beschrieben, die versucht, eine vorhandene Bisektion $G = G_1 \boxplus_K G_2$ mit Untergraphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ bzw. eine allgemeinere Partitionierung eines Graphen $G = (V, E)$ zu verbessern. Dabei werden bei der Suche nach einer guten Bisektion benachbarte Knoten zwischen den beiden Teilen ausgetauscht. Dazu wird zunächst ein so genannter diff-Wert aller Knoten bestimmt. Dieser entspricht dem Wert, um den sich die Größe des gewählten Kantenschnittes verringern würde, wenn der Knoten $v \in V_1$ von V_1 nach V_2 verlegt würde:

$$\mathrm{diff}(v, V_1, V_2) = |N_G(v) \cap V_2| - |N_G(v) \cap V_1|$$

Eine zweite Größe, der gain-Wert eines Knotenpaares (u, v) mit $u \in V_1$ und $v \in V_2$, gibt die Veränderung der Schnittgröße an, wenn beide Knoten auf die jeweilig andere Seite des Schnitts wechseln:

$$\mathrm{gain}(u, v, V_1, V_2) = \mathrm{diff}(u, V_1, V_2) + \mathrm{diff}(v, V_2, V_1) - 2 \cdot [(uv) \in E]$$

Mit dieser Notation nimmt der Algorithmus von Kernigan und Lin folgende Form an:

```
Gegeben sei eine Zerlegung der Knotenmenge V = V_1 + V_2
  s := 0
  D_0 := |{(uv) : u ∈ V_1, v ∈ V_2}|
  Wiederhole
    s := s + 1
    D_s := |{(uv) ∈ E : u ∈ V_1 ∧ v ∈ V_2}|
    Berechne die diff-Werte aller Knoten
    M := V
    d_0 := 0
    Für i von 1 bis min(|V_1|, |V_2|):
      Finde (u_i, v_i) := argmax_{(u,v):u∈V_1∩M, v∈V_2∩M} gain(u, v, V_1, V_2)
      M := M − {u_i, v_i}
      Für alle Nachbarn w von u_i und v_i:
        t := 2 · [(wu_i) ∈ E] − 2 · [(wv_i) ∈ E]
        t := −t falls w ∈ V_2
```

$$
\begin{aligned}
\mathrm{diff}(w) &= \mathrm{diff}(w) + t \\
d_i &= d_{i-1} + \mathrm{gain}(u_i, w_i, V_1, V_2) \\
&\texttt{Finde } j := \mathrm{argmax}_i\, d_i \\
V_1 &:= (V_1 - \{u_1, \ldots, u_i\}) \cup \{v_1, \ldots, v_i\} \\
V_2 &:= (V_2 - \{v_1, \ldots, v_i\}) \cup \{u_1, \ldots, u_i\} \\
&\texttt{Bis } |D_s| \geq |D_{s-1}| \\
&\texttt{Rückgabe } (V_1, V_2)
\end{aligned}
$$

Abgewandelte Implementationen können $\mathcal{O}(|E|)$ für eine Iteration erreichen (vgl. [FiM82]), so dass der Algorithmus sicher nach $\mathcal{O}(|E| \cdot |V|)$ Operationen terminiert.

Lokale Methoden - Dynamische Suche

In einem „hausgemachten“ Verfahren wird zunächst einer von zwei im Graphen diametral entgegengesetzt liegenden Knoten als Startknoten festgelegt. Danach beginnt von diesem Startknoten aus eine Breitensuche mit beschränkter Suchtiefe, in deren Verlauf alle möglichen Fortsetzungen der bislang festgelegten Knotenfolge um eine bestimmte Anzahl von Knoten untersucht werden. Der erste Knoten derjenigen Fortsetzung, die die minimalen Gesamtkosten verspricht, wird tatsächlich Bestandteil der endgültigen Knotenfolge, und das Verfahren beginnt von vorn mit dieser erweiterten endgültigen Knotenfolge. Das Verfahren ermöglicht zahlreiche Variationen der Parameter, insbesondere der Suchtiefe und der Kostenfunktion.

Eine Weiterentwicklung dieser Methode, für die in der Praxis allerdings bislang noch kein Bedarf bestand, besteht darin, die Knoten- bzw. Kantenfolge erst während des Ablaufs des eigentlichen Kompositionsalgorithmus zu erzeugen. Damit ist es prinzipiell möglich, mit der Bearbeitung von Graphen zu beginnen, wenn diese noch gar nicht vollständig bekannt sind. Das könnte insbesondere im Zusammenhang mit dynamisch erzeugten Graphen (zum Beispiel Suchbäume), oder bei sehr großen Graphen, die nicht „im Stück“ in den Arbeitsspeicher geladen werden können, von Interesse sein.

C Experimentelle Ergebnisse

C.1 Enumeration und Polynome

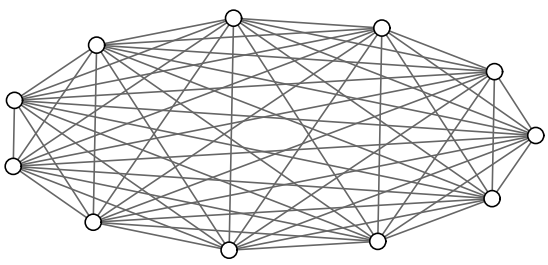
In diesem Abschnitt sollen für einige der im Kapitel 5 beschriebenen Kompositionsalgorithmen praktische Rechenergebnisse angegeben werden. Die verwendeten Testgraphen sind vollständige Graphen K_n auf n Knoten (vgl. Abb.C.1) , $n \times n$ Gitter G_n (Abb. C.2), Springergraphen S_n (Abb. C.3) sowie einige unregelmäßige Graphen (C.4). Diese Auswahl von Beispielen deckt typische Fälle dichter bzw. planarer Graphen ab.

In den Tabellen sind die Ergebnisse für jeden Algorithmus in jeweils vier Zeilen zusammengefasst. In der ersten Zeile steht die maximale Mächtigkeit der während der Abarbeitung der Algorithmen entstehenden Zustandsmengen, in der zweiten Zeile die Gesamtzahl der bearbeiteten Zustände. Diese Daten sind nur von der Implementation der Algorithmen, nicht aber vom verwendeten Rechner abhängig. Eine Markierung „-“ bedeutet, dass die Durchführung der Berechnung aufgrund des begrenzten Hauptspeichers scheiterte.

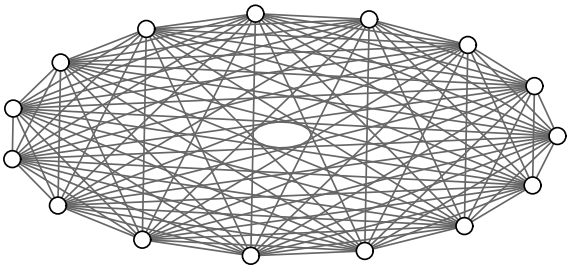
In der dritten und vierten Zeile folgen gemessene Laufzeiten und der Speicherplatzbedarf aus einer C++-Implementation der Algorithmen auf einem AMD Athlon^{TM} XP1700+ (Stepping 2) mit 1 GByte RAM (256 kByte Prozessor-Cache) unter Linux 2.4.19 unter Verwendung des g++ 3.2. Diese Werte sind stark architektur- und implementationsabhängig.

Vollständige Graphen

Graph	K_{10}	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{20}	K_{25}
Knoten	10	11	12	13	14	15	20	25
Kanten	45	55	66	78	91	120	190	300
Wegweite	9	10	11	12	13	14	19	24
	42 294	231 590	-	-	-	-	-	-
Zuverl.-	450 594	$2.64 \cdot 10^6$	-	-	-	-	-	-
polynom	33 s	1 h	-	-	-	-	-	-
	72 MB	334 MB	-	-	-	-	-	-
K-Zus.-	42 294	231 590	$1.36 \cdot 10^6$	$8.43 \cdot 10^7$	-	-	-	-
hangs-	450 594	$2.64 \cdot 10^6$	$1.64 \cdot 10^7$	$1.08 \cdot 10^8$	-	-	-	-
wahrsch.	0.7 s	5.4 s	43.4 s	353 s	-	-	-	-
(exakt)	10 MB	20 MB	92 MB	670 MB	-	-	-	-
	19	21	23	25	27	29	39	49
Unabh.-	651	848	1 081	1 353	1 667	2 026	4 601	8 751
polynom	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s
	1 MB	1 MB	1 MB	1 MB	1 MB	1 MB	1 MB	1 MB
Chromat.	42 294	231 930	$1.4 \cdot 10^6$	-	-	-	-	-
Polynom	410 097	$2.4 \cdot 10^6$	$1.5 \cdot 10^7$	-	-	-	-	-
(Typ A)	1.0 s	7.0 s	58 s	-	-	-	-	-
	6 MB	20 MB	138 MB	-	-	-	-	-
Chromat.	9	10	11	12	13	14	19	24
Polynom	257	332	420	522	639	772	1 712	3 202
(Typ B)	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s	0.2 s
	1 MB	1 MB	1 MB	1 MB	1 MB	1 MB	1 MB	1 MB
	42 294	231 950	$1.4 \cdot 10^6$	-	-	-	-	-
Tutte-	460 029	$2.7 \cdot 10^6$	$1.6 \cdot 10^7$	-	-	-	-	-
Polynom	7.4 s	67 s	3400 s	-	-	-	-	-
	10 MB	28 MB	223 MB	-	-	-	-	-
	16 322	49 108	147 502	442 724	$1.3 \cdot 10^6$	$4.0 \cdot 10^6$	-	-
Kreisüber-	176 310	596 123	$2.0 \cdot 10^6$	$6.5 \cdot 10^6$	$2.1 \cdot 10^7$	$6.9 \cdot 10^7$	-	-
deckungen	0.1 s	0.7 s	3.4 s	14 s	52 s	220 s	-	-
	1 MB	2 MB	4 MB	16 MB	62 MB	150 MB	-	-



(a) Vollst. Graph K_{11}: 11 Knoten, 55 Kanten.

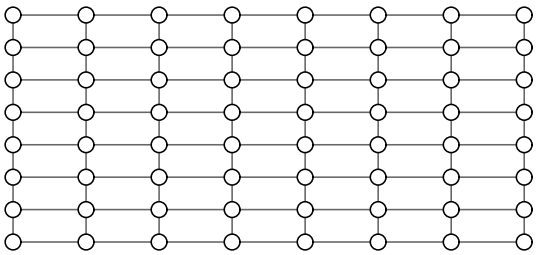


(b) Vollst. Graph K_{15}: 15 Knoten, 105 Kanten.

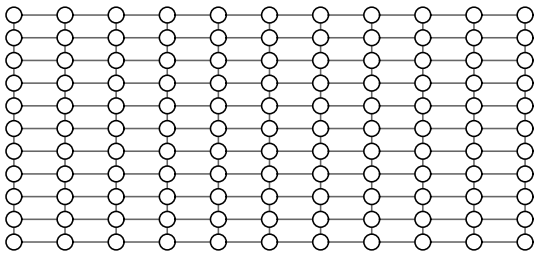
Abbildung C.1: Vollständige Graphen

Gittergraphen

Graph	G_8	G_9	G_{10}	G_{11}	G_{12}	G_{13}	G_{15}
Knoten	64	81	100	121	144	169	225
Kanten	112	144	180	220	262	306	420
Wegweite	8	9	10	11	12	13	15
	2 860	9 724	33 592	117 572	-	-	-
Zuverl.-	175 009	682 587	$2.7 \cdot 10^6$	$1.0 \cdot 10^7$	-	-	-
polynom	0.4 s	1.7 s	13 s	227 s	-	-	-
	1 MB	2 MB	4 MB	88 MB	-	-	-
K-Zus.-	2 860	9 724	33 592	$1.2 \cdot 10^5$	$4.2 \cdot 10^5$	$1.5 \cdot 10^6$	-
hangs-	175 009	682 587	$2.7 \cdot 10^6$	$1.0 \cdot 10^7$	$4.0 \cdot 10^7$	$1.6 \cdot 10^8$	-
wahrsch.	0.2 s	0.8 s	3.9 s	21 s	93 s	414 s	-
(exakt)	1 MB	2 MB	4 MB	9 MB	22 MB	93 MB	-
	384	768	1 536	3 072	6 144	12 288	49 152
Unabh.-	26 660	61 988	141 348	317 476	704 548	$1.6 \cdot 10^6$	$7.3 \cdot 10^6$
polynom	0.2 s	0.3 s	0.6 s	1.3 s	3.1 s	7.8 s	103 s
	1 MB	2 MB	2 MB	3 MB	6 MB	16 MB	65 MB
Chromat.	2 860	9 724	33 592	117 572	-	-	-
Polynom	168 719	657 775	$2.6 \cdot 10^6$	$1.0 \cdot 10^7$	-	-	-
(Typ A)	1.4 s	7.3 s	66 s	2500 s	-	-	-
	2 MB	9 MB	47 MB	196 MB	-	-	-
Chromat.	17 007	94 828	-	-	-	-	-
Polynom	597 002	$3.7 \cdot 10^6$	-	-	-	-	-
(Typ B)	16 s	870 s	-	-	-	-	-
	17 MB	150 MB	-	-	-	-	-
	5 720	-	-	-	-	-	-
Tutte-	214 003	-	-	-	-	-	-
Polynom	150 s	-	-	-	-	-	-
	86 MB	-	-	-	-	-	-
	1 138	3 242	8 588	24 858	67 630	195 736	$1.6 \cdot 10^6$
Kreisüber-	43 767	132 967	437 573	$1.3 \cdot 10^6$	$4.3 \cdot 10^6$	$1.3 \cdot 10^7$	$1.2 \cdot 10^8$
deckungen	0.1 s	0.1 s	0.3 s	0.9 s	4.0 s	16 s	230 s
	1 MB	1 MB	1 MB	2 MB	3 MB	5 MB	65 MB



(a) Gitter G_8: 64 Knoten, 112 Kanten.



(b) Gitter G_{11}: 121 Knoten, 220 Kanten.

Abbildung C.2: Gittergraphen

Springergraphen

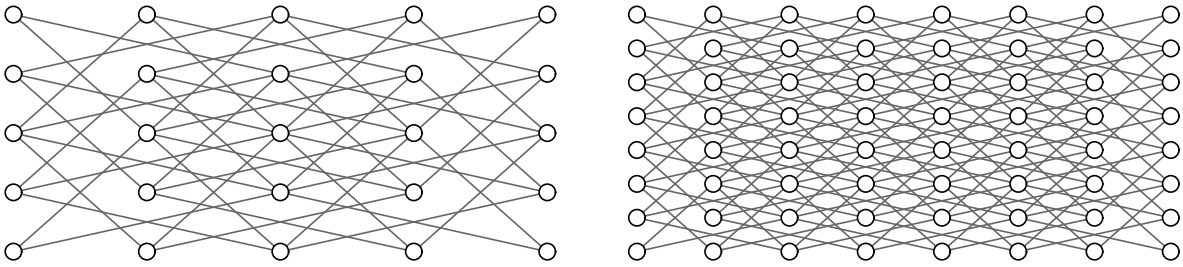
Der Springergraph S_n besteht aus Knoten $\{v_{ij} : 1 \leq i, j \leq n\}$ und Kanten $\{(v_{ij}v_{kl}) : |i-k| + |j-l| = 3 \wedge i \neq k \wedge j \neq l\}$. Für $n \geq 4$ ist S_n zusammenhängend, nichtplanar und dichter als ein Gittergraph gleicher Knotenzahl.

In den Berechnungen wurden Knotenkontaktzerlegungen verwendet, die aus den Knotenfolgen

$$(v_{12}, v_{14}, \ldots, v_{1t},\ v_{1s}, \ldots, v_{13}, v_{11},\ v_{22}, v_{24}, \ldots, v_{2t},\ v_{2s}, \ldots, v_{23}, v_{21},\ \ldots,\ v_{n1})$$

mit $t = n$ und $s = n-1$ für gerades n und $t = n-1$ und $s = n$ für ungerades n konstruiert wurden. In beiden Fällen beträgt die Weite der gewählten Kontaktzerlegung $2\,n$. Für kleines n liegt dieser Wert über der jeweiligen Wegweite ($\mathrm{pw}(S_3) = 2$, $\mathrm{pw}(S_4) = 4$, $\mathrm{pw}(S_5) \leq 9$), asymptotisch gilt allerdings $\mathrm{pw}(S_n) = 2\,n$ für $n \geq 6$.

Graph	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
Knoten	16	25	36	49	64	81	100	121
Kanten	24	48	80	120	168	224	288	360
Weite	8	10	12	14	16	18	20	22
Unabh.-polynom	160	440	1 456	3 920	11 968	32 384	103 936	$2.8 \cdot 10^7$
	267	14 288	77 758	307 264	$1.2 \cdot 10^6$	$4.4 \cdot 10^6$	$1.7 \cdot 10^7$	$5.7 \cdot 10^7$
	0.1 s	0.1 s	0.3 s	0.9 s	4 s	20 s	220 s	1600 s
	1 MB	1 MB	1 MB	3 MB	11 MB	36 MB	140 MB	440 MB
Tutte-Polynom	1.716	-	-	-	-	-	-	-
	12.562	-	-	-	-	-	-	-
	0.9 s	-	-	-	-	-	-	-
	1 MB	-	-	-	-	-	-	-
Kreisüberdeckungen	98	1 867	93 675	932 591	$8.4 \cdot 10^6$	-	-	-
	1 020	20 019	$1.3 \cdot 10^6$	$3.0 \cdot 10^7$	$5.2 \cdot 10^8$	-	-	-
	0.1 s	0.2 s	1.7 s	52 s	2300 s	-	-	-
	1 MB	1 MB	4 MB	42 MB		-	-	-



(a) Springergraph S_5: 25 Knoten, 48 Kanten.

(b) Springergraph S_8: 64 Knoten, 168 Kanten.

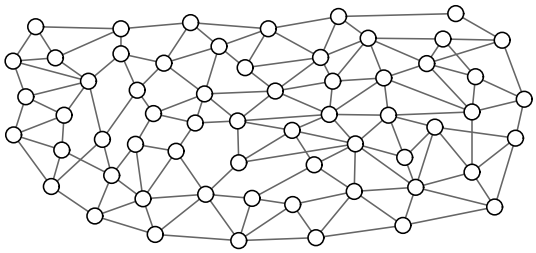
Abbildung C.3: Springergraphen

Unregelmäßige Graphen

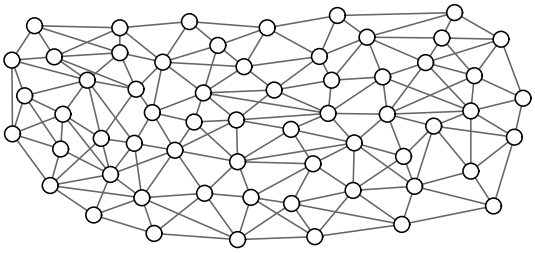
Die in der folgenden Tabelle dargestellten Ergebnisse für die in Abbildung C.4 dargestellten Graphen zeigen, dass vergleichsweise geringfügige Änderungen an der Struktur des Graphen (zum Beispiel das Erhöhen der Kantendichte um 10%) erheblichen Einfluss auf das Laufzeitverhalten der Kompositionsalgorithmen haben können (hier am Beispiel des Zuverlässigkeitspolynoms), aber nicht haben müssen (hier am Beispiel des Unabhängigkeitspolynoms). Das bestätigt die Erfahrung, dass es so gut wie unmöglich ist, das konkrete Verhalten in unregelmäßigen Strukturen *a priori* aus einfachen Eigenschaften wie Knoten- oder Kantenzahl abzuschätzen.

Graph	GGR1	GGR2	GGR3	GGR4	GGR5	GGR6	G250
Knoten	60	60	60	60	60	60	250
Kanten	141	177	210	240	255	275	500
Wegweite	≤ 8	≤ 9	≤ 9	≤ 10	≤ 11	≤ 12	≤ 11
	6 536	29 152	40 986	-	-	-	701 108
Zuverl.-	245 006	732 077	$1.8 \cdot 10^6$	-	-	-	$4.8 \cdot 10^7$
polynom	13 s	90 s	680 s	-	-	-	1500 s
	27 MB	99 MB	170 MB	-	-	-	230 MB
K-Zus.-	6 536	29 152	40 986	231 542	$1.4 \cdot 10^6$	-	701 108
hangs-	245 006	732 077	$1.8 \cdot 10^6$	$1.7 \cdot 10^7$	$8.6 \cdot 10^7$	-	$4.8 \cdot 10^7$
wahrsch.	3 s	8 s	16 s	200 s	1200 s	-	650 s
(exakt)	13 MB	15 MB	20 MB	50 MB	330 MB	-	170 MB
	129	131	131	157	192	252	2 304
Unabh.-	9 090	10 777	12 921	18 784	21 340	26 727	361 275
polynom	1.0 s	1.1 s	1.1 s	1.2 s	1.2 s	1.4 s	13 s
	10 MB	10 MB	10 MB	10 MB	10 MB	10 MB	13 MB
Chrom.	6 097	25 964	40 554	-	-	-	-
Polynom	228 438	685 936	$1.5 \cdot 10^6$	-	-	-	-
(Typ A)	6 s	13 s	30 s	-	-	-	-
	18 MB	33 MB	45 MB	-	-	-	-
Chrom.	6 097	25 964	17 007	52 922	192 713	-	-
Polynom	140 791	244 066	367 483	$1.5 \cdot 10^6$	$4.4 \cdot 10^6$	-	-
(Typ B)	6 s	8 s	10 s	50 s	700 s	-	-
	22 MB	45 MB	42 MB	67 MB	202 MB	-	-
	6 536	-	-	-	-	-	-
Tutte-	254 556	-	-	-	-	-	-
Polynom	700 s	-	-	-	-	-	-
	240 MB	-	-	-	-	-	-
	3 517	14 581	16 131	48 936	145 560	440 440	43 200
Kreisüber-	13 900	424 205	843 219	$3.7 \cdot 10^6$	$9.8 \cdot 10^6$	$2.7 \cdot 10^7$	$2.9 \cdot 10^6$
deckungen	0.7 s	1.8 s	2.7 s	4.3 s	14 s	46 s	4 s
	2 MB	3 MB	5 MB	7 MB	14 MB	22 MB	10 MB

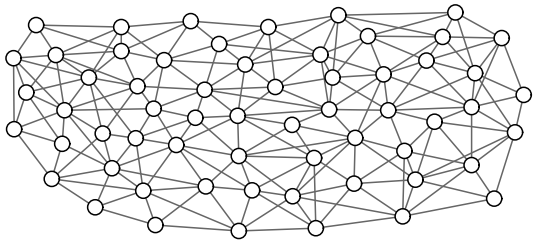
Tabelle C.1: Unregelmäßige Graphen



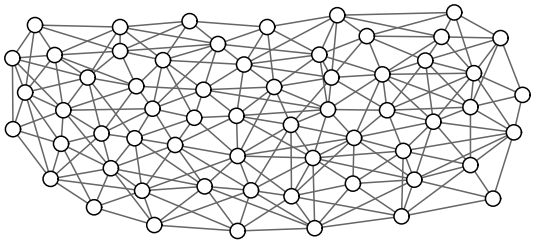
(a) Graph GGR1: 60 Knoten, 141 Kanten.



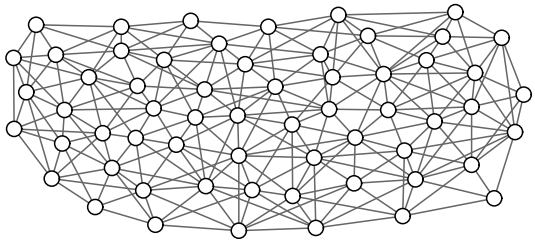
(b) Graph GGR2: 60 Knoten, 177 Kanten.



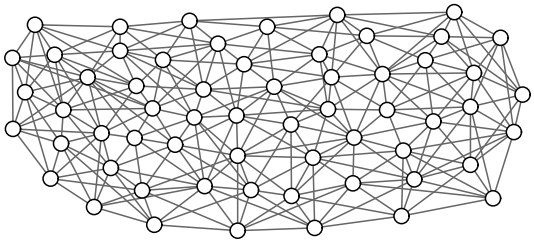
(c) Graph GGR3: 60 Knoten, 210 Kanten.



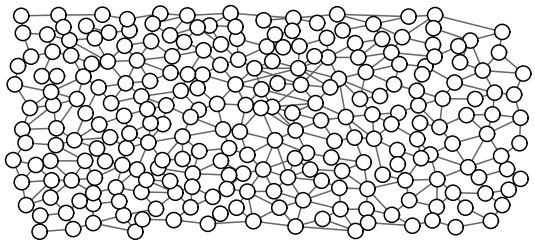
(d) Graph GGR4: 60 Knoten, 240 Kanten.



(e) Graph GGR5: 60 Knoten, 255 Kanten.



(f) Graph GGR6: 60 Knoten, 275 Kanten.



(g) Graph G250: 250 Knoten, 500 Kanten.

Abbildung C.4: Einige unregelmäßige Graphen

C.2 K-Zusammenhangswahrscheinlichkeit

In diesem Abschnitt werden in zwei Tabellen Ergebnisse der approximativen Bestimmung von K-Zusammenhangswahrscheinlichkeiten in Gittergraphen unterschiedlicher Größe für verschiedene Kantenausfallwahrscheinlichkeiten dargestellt. Als Terminalknoten wurden jeweils zwei gegenüberliegende Ecken des Gitters gewählt.

Graph	G_{10}	G_{15}	G_{20}	G_{25}	G_{30}	G_{50}	G_{120}
Knoten	100	225	400	625	900	2 500	14 400
Kanten	180	420	760	1 200	1 740	4 900	28 560
$p = 0.1$	24 990	-	-	-	-	-	-
$R(G_{10}) \in [0.0000000,$	581 597	-	-	-	-	-	-
$0.0000001]$	1 s	-	-	-	-	-	-
$p = 0.3$	96 988	-	-	-	-	-	-
$R(G_{10}) \in [0.0000215$	$5.9 \cdot 10^6$	-	-	-	-	-	-
0.0000216	10 s	-	-	-	-	-	-
$p = 0.5$	147 376	-	-	-	-	-	-
$R(G_{10}) \in [0.0642228,$	$6.8 \cdot 10^6$	-	-	-	-	-	-
$0.0642239]$	17 s	-	-	-	-	-	-
$p = 0.7$	136 236	-	-	-	-	-	-
$R(G_{10}) \in [0.6791793,$	$6.3 \cdot 10^6$	-	-	-	-	-	-
$0.6791804]$	16 s	-	-	-	-	-	-
$p = 0.9$	26 240	363 900	$3.4 \cdot 10^6$	-	-	-	-
$R(G_{10}) \in [0.9756607,$	$1.3 \cdot 10^6$	$3.4 \cdot 10^7$	$5.0 \cdot 10^8$	-	-	-	-
$0.9756618]$	2.2 s	100 s	2000 s	-	-	-	-
$p = 0.95$	5 540	36 848	155 476	672 648	$2.6 \cdot 10^6$	-	-
$R(G_{10}) \in [0.9944790,$	322.542	$3.9 \cdot 10^6$	$2.7 \cdot 10^7$	$1.5 \cdot 10^8$	$7.7 \cdot 10^8$	-	-
$0.9944801]$	0.4 s	6.6 s	62 s	420 s	4500 s	-	-
$p = 0.99$	464	1 732	3 368	5 512	9 440	110 616	-
$R(G_{10}) \in [0.9997953,$	36 062	234 871	806 040	$2.0 \cdot 10^6$	$4.3 \cdot 10^7$	$1.5 \cdot 10^8$	-
$0.9997964]$	0.1 s	0.3 s	0.9 s	2.0 s	4.3 s	680 s	-
$p = 0.999$	100	204	320	416	496	816	23 200
$R(G_{10}) \in [0.9999972,$	7 708	36 601	95 015	195 976	333 311	$1.5 \cdot 10^6$	$2.0 \cdot 10^8$
$0.9999983]$	0.1 s	0.2 s	0.3 s	0.5 s	0.7 s	3 s	770 s

Tabelle C.2: Vorgegebene Genauigkeit $1.1 \cdot 10^{-6}$

Graph	G_{10}	G_{15}	G_{20}	G_{25}	G_{30}	G_{120}
Knoten	100	225	400	625	900	14 400
Kanten	180	420	760	1200	1740	28 560
$p = 0.1$	14 260	-	-	-	-	-
$R(G_{10}) \in [0.00000000000,$	884 093	-	-	-	-	-
$0.00000000001]$	3 s	-	-	-	-	-
$p = 0.3$	149 192	-	-	-	-	-
$R(G_{10}) \in [0.00002151970,$	$8.9 \cdot 10^6$	-	-	-	-	-
0.00002151981	51 s	-	-	-	-	-
$p = 0.5$	164 052	-	-	-	-	-
$R(G_{10}) \in [0.06422230013,$	$7.6 \cdot 10^6$	-	-	-	-	-
$0.06422230024]$	21 s	-	-	-	-	-
$p = 0.7$	161 044	-	-	-	-	-
$R(G_{10}) \in [0.67917988404,$	$7.5 \cdot 10^6$	-	-	-	-	-
$0.67917884115]$	21 s	-	-	-	-	-
$p = 0.9$	87 584	$3.8 \cdot 10^6$	-	-	-	-
$R(G_{10}) \in [0.97566162371,$	$4.2 \cdot 10^6$	$3.2 \cdot 10^8$	-	-	-	-
$0.97566162382]$	10.1 s	1380 s	-	-	-	-
$p = 0.95$	30 988	387 068	$2.9 \cdot 10^6$	-	-	-
$R(G_{10}) \in [0.99448002667,$	$1.6 \cdot 10^6$	$3.7 \cdot 10^8$	$4.4 \cdot 10^8$	-	-	-
$0.99448002678]$	2.8 s	105 s	1650 s	-	-	-
$p = 0.99$	3 356	11 952	39 100	113 132	262 216	-
$R(G_{10}) \in [0.99979596952,$	198.974	$1.5 \cdot 10^6$	$7.0 \cdot 10^6$	$3.1 \cdot 10^7$	$1.1 \cdot 10^8$	-
$0.99979596963]$	0.2 s	1.8 s	12 s	108 s	551 s	-
$p = 0.999$	460	840	1 792	3 796	6 148	264 380
$R(G_{10}) \in [0.99999799598,$	35 871	147 656	436 261	$1.2 \cdot 10^6$	$3.0 \cdot 10^6$	$1.2 \cdot 10^9$
$0.99999799609]$	0.1 s	0.2 s	0.5 s	1.3 s	6.2 s	2.8 h

Tabelle C.3: Vorgegebene Genauigkeit $1.1 \cdot 10^{-10}$

C.3 Neil Sloanes Ganzzahl-Sequenzen

Ein objektiver Vergleich der Kompositionsalgorithmen mit den anderen Algorithmen zur Lösung gleicher Aufgabenstellungen ist kaum durchzuführen. Die üblichen technischen Probleme eines *benchmarkings* wie zum Beispiel identische Testumgebung sowohl auf der Hardwareseite (CPU, RAM, Cache, Bus, ...) als auch auf der Softwareseite (Betriebssystem, Compiler, Programmiersprache, Bibliotheken, ...) treten prominent in den Vordergrund, da kaum eine Publikation diesbezüglich genügend Details enthält. Noch seltener sind konkrete Implementationen zu finden, so dass ein nicht zu vernachlässigender subjektiver Aspekt hinzukommt: Eine Neuimplementation von oft nur skizzenhaft beschriebenen Algorithmen wird kaum alle vom ursprünglichen Autor verwendeten Tricks enthalten, so dass ein solches Vorgehen mit großer Wahrscheinlichkeit zu einem systematischen Fehler führt, der die „Fremdalgorithmen" benachteiligt.

Auf der Suche nach „objektiven" Kriterien fällt Neil Sloanes *On-Line Encyclopedia of Integer Sequences* [Slo03] ins Auge, die mehrere zehntausend Folgen nichtnegativer ganzer Zahlen enthält. Einige dieser Folgen stammen aus Abzählproblemen in Graphen und sind damit prinzipiell für den Test von Kompositionsalgorithmen geeignet. Im Zuge dieser Tests konnten folgende, als „schwer berechenbar" klassifizierte Folgen erweitert werden:

- A003763: 1, 6, 1 072, 4 638 576, 467 260 456 608, 1 076 226 888 605 605 706, 56 126 499 620 491 437 281 263 608 – die Zahl der Hamilton-Kreise auf einem $2\,n \times 2\,n$-Gitter. Die ersten fünf Glieder der Folge wurden von J. Shallit im Februar 2002 eingereicht. Mit Hilfe eines Kantenkompositionsalgorithmus konnten die beiden folgenden Glieder im März 2003 ermittelt werden.

- A080690: 1, 14, 2 398, 5 015 972, 128 091 434 266, 39 931 856 138 212 664, 151 966 368 274 993 474 937 668, 7 059 965 159 455 454 640 307 807 067 492, 4 003 910 412 343 921 295 679 925 280 332 950 062 686 – die Zahl der azyklischen Orientierungen eines $n \times n$-Gitters. Der Artikel von Calkin, Merino, Noble und Noy [CMN03] aus dem Jahr 2003 erwähnt die ersten sieben Glieder dieser Folge, während die beiden letzten aus einem Kantenkompositionsalgorithmus stammen.

- A080691: 1, 15, 3 102, 8 790 016, 341 008 617 408, 181 075 508 242 067 552, 1 315 927 389 374 152 034 113 856, 130 877 523 274 817 580 209 987 036 404 864, 178 135 975 585 132 088 643 635 627 145 305 047 963 624 – die Zahl der spannenden Bäume eines $n \times n$-Gitters. Die ersten Glieder stammen ebenfalls aus [CMN03] und die beiden letzten wieder aus einem Kantenkompositionsalgorithmus.

- A028420 entspricht der Bestimmung der Zahl aller Matchings im $n \times n$-Gitter. Die angegebenen Folge kann mit Hilfe eines Kompositionsalgorithmus auf 22 Glieder erweitert werden. Der bei Sloane angegeben Algorithmus von Zeilberger berechnet bei ähnlichem Speicher- und Zeitbedarf nur 15 Glieder.

C.4 Praktische Grenzen

Die folgende Tabelle gibt einen Überblick über die Größe von Aufgaben, die sich mit Kompositionsalgorithmen lösen lassen. Die Zahlen im rechten Teil der Aufstellung geben an, bis zu welcher Größe n die jeweiligen Graphenkenngrößen innerhalb von drei Minuten auf einem System der im Abschnitt C.1 genannten Art in vollständigen Graphen K_n, in $n \times n$-Gittern G_n sowie im $n \times n$-Springergraphen S_n (vgl. Abb. C.3) berechnet werden können.

Trägt ein Eintrag eine Markierung „+“, so ist das ein Indikator dafür, dass die spezielle Struktur der gewählten Graphen zu besonders guter Leistung des Algorithmus führt, das Ergebnis aber nicht auf Graphen ähnlicher Struktur und Größe ($n = 100 \ldots 1000$) verallgemeinert werden kann. Die hochgestellten Zahlen verweisen auf Bemerkungen auf der folgenden Seite, die die hier gezeigten Resultate mit bekannten „externen“ Ergebnissen vergleichen.

Problem	K_n	G_n	S_n
Zulässige Färbung	+	10	5
Chromatische Zahl	+	+	+
Chromatisches Polynom [4.]	+	8	5
Zahl azyklischer Orientierungen [5.]	+	11	5
Maximale Clique	25	+	+
Zahl der Kreisüberdeckungen	14	14	6
Verallg. chrom. Polynom	14	7	4
Dominationszahl	26	14	9
Dominationspolynom	21	10	7
Dominierende Knotenmenge	22	13	8
Zahl der Hamilton-Kreise	11	13	6
Zahl der Hamilton-Wege	11	13	6
Kürzeste Rundreise	13	12	6
Unabhängigkeitspolynom	+	16	11
Maximale unabhängige Menge	+	18	12
Zahl unabhängiger Mengen	+	20	15
Unabhängigkeitszahl	+	20	15
Zahl perfekter Matchings	24	24	13
Matchingpolynom	23	15	9
Negami-Polynom	11	8	5
Tutte-Polynom [1. 3. 4.]	11	8	5
Zuverlässigkeitspolynom (P-Form) [2. 3. 4.]	11	9	5
Zuverlässigkeitspolynom (N-Form)	11	10	5
Flusspolynom [4.]	11	10	5
Zusammenhangswahrscheinlichkeit	12	11	5
K-Zusammenhangswahrscheinlichkeit	11	10	5
Konstruktion eines Steinerbaums	12	15	5
Zahl spannender Wälder [5.]	12	12	6

Tabelle C.4: Innerhalb von drei Minuten berechenbare Strukturen

1. In [SIT98] wird das Tutte-Polynom in dichten Graphen mit bis zu 14 Knoten berechnet, allerdings nicht innerhalb von drei Minuten.

2. Das in [Mon] angegebene Programm ist in der Lage, das Zuverlässigkeitspolynom von vollständigen Graphen bis zum K_{12} und von Gittergraphen bis zum G_5 zu berechnen.

3. Das Zuverlässigkeitspolynom kann nach der in [KyH01] vorgestellten Methoden in Graphen mit bis zu 100 Kanten berechnet werden.

4. Die Funktionen aus dem *network*-Paket von Maple 9 berechnen diese Polynome bis zum K_7 bzw. G_4.

5. In [CMN03] werden azyklische Orientierungen und spannende Wälder bis zum G_7 angegeben.

Symbolverzeichnis

p	logische Aussage	
$[p]$	Indikatorfunktion	9
$z \cdot [p]$	Selektion	9
A, J, M	Mengen	
$M + v$	Element zu Menge hinzufügen	7
$M - v$	Element aus Menge löschen	7
$M - J$	Menge von Elementen löschen	7
2^Ω	Menge aller Teilmengen von Ω	7
Ω^*	Menge aller Multimengen mit Grundmenge Ω	8
π	Mengenpartition	
$\mathcal{P}(\mathcal{A})$	Menge aller Mengenpartitionen der Menge A	8
$\pi_1 \vee \pi_2$	Verschmelzung von Partitionen	9
π_M	Kontraktion einer Menge in einer Partition	9
π_e	Kontraktion einer Kante in einer Partition	9
π_{+S}	Erweitern einer Partition um Einzelblöcke	8
$\pi_{+v,B}$	Einfügen eines Knotens in einen Block	8
π_{+v}	Einfügen eines neuen Blocks in eine Partition	8
G	Graph	
H	Hypergraph	
T	Kontaktgraph	
n	Zahl der Knoten eines Hypergraphen	
m	Zahl der Kanten eines Hypergraphen	
e, f	Kanten eines Hypergraphen	
u, v, w	Knoten eines Hypergraphen	

E, F	Mengen von Kanten	
V, U, N, J	Mengen von Knoten	
$\mathcal{G}$	Menge aller Graphen	12
$\mathcal{H}$	Menge aller Hypergraphen	10
$\mathcal{T}$	Menge aller Kontaktgraphen	13
$N_H(v)$	Nachbarschaft eines Knotens	11
$N_H(J)$	Nachbarschaft einer Knotenmenge	11
$\bar{G}$	Komplement eines Graphen	12
$\mathrm{comp}(H)$	Menge der Komponenten eines Hypergraphen	10
$k(H)$	Zahl der Komponenten eines Hypergraphen	10
$I_H(v)$	Inzidente Kanten zu einem Knoten	11
$I_H(J)$	Inzidente Kanten zu einer Knotenmenge	11
$H{:}F$	Unterhypergraph mit Kantenmenge F	11
H_{-e}	Löschen einer Kante in einem Hypergraphen	11
H_{-v}	Löschen eines Knotens in einem Hypergraphen	11
H_{-J}	Löschen einer Knotenmenge in einem Hypergraphen	11
H_J	Kontraktion einer Knotenmenge in einem Hypergraphen	11
H_e	Kontraktion einer Kante in einem Hypergraphen	11
$H_{+v,J}$	Kontraktion einer Knotenmenge zu einem Knoten	12
$H_1 \boxplus_K H_2$	Zerlegung eines Hypergraphen an der Knotenmenge K	14
$H_1 + T_2$	Kontaktzerlegung eines Hypergraphen	13
$\mathrm{pw}(H)$	Wegweite eines Hypergraphen	19
$\mathrm{tw}(H)$	Baumweite eines Hypergraphen	18

i	Index	
w	Wert	
z	Zustand	
R	Graphenkenngröße	
$\mathcal{C}_H(c_E, c_V)$	Menge aller (c_E, c_V)-Konstellationen eines Hypergraphen	20
$\mathcal{W}$	Wertebereich einer Graphenkenngröße	22
$\oplus$	Binäre Operation auf $\mathcal{W}$	5
$\mathcal{I}$	Indexbereich einer Graphenkenngröße bzgl. einer Indizierung	21
$\uplus$	Zusammenfassen von Zuständen bzw. Zustandsmengen	25

Index

Literaturverzeichnis

[AlF91] Alexopoulos, C.; Fishman, G. S.: *Characterizing stochastic flow networks using the Monte Carlo method.* Networks 21 (1991), 775–798.

[And95] Andrzejak, A.: *A polynomial-time algorithm for computation of the Tutte polynomials of graphs of bounded treewidth.* Freie Universität Berlin, Institut für Informatik, Technical Report B 95-16, 1995.

[ACP87] Arnborg, S.; Corneil, D. G.; Proskurowski, A.: *Complexity of finding embeddings in a k-tree.* SIAM J. Alg. Disc. Meth. 8 (1987), 277–284.

[ArP94] Arnborg, S.; Proskurowski, A.: *Linear time algorithms for NP-hard problems on graphs embedded in k-trees.* Discrete Appl. Math. 23 (1994), 97–291.

[Bal77] Ball, M. O.: *Network reliability and analysis: Algorithms and complexity.* Cornell University Ithaca, OR Department, Doctoral Thesis, 1977.

[BiL46] Birkhoff, G. D.; Lewia, D.: *Chromatic polynomials.* Trans. Amer. Math. Soc. 60 (1946), 355–451.

[Bod95] Bodlaender, H. L.: *Treewidth: Algorithmic techniques and results.* Proceedings 22nd Intern. Symposium on Math. Foundations of Computer Science, MFCS'97, Lecture Notes in Computer Science 12, 1995.

[Bod96] Bodlaender, H. L.: *A linear time algorithm for finding tree-decompositions of small treewidth.* SIAM Journal on Computing 25 (1996), 1305–1317.

[Bod98] Bodlaender, H. L.: *A partial k-arboretum of graphs with bounded treewidth.* J. Theoretical Computer Science 209 (1998), 1–45.

[BoF00] Bodlaender, H. L.; Fomin, F. V.: *Approximation of pathwidth of outerplanar graphs.* Utrecht University, Dept. of Computer Science, Technical Report UU-CS-2000-23, 2000.

[BGH95] Bodlaender, H. L; Gilbert, J. R.; Hafsteinsson, H.; Kloks, T.: *Approximating treewidth, pathwidth, frontsize and shortest elimination tree.* J. Algorithms 18 (1995), 238–255.

[BoS91] Boesch, F.; Satyanarayana, A: *On residual connectedness network reliability.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science 5 (1991), 51–59.

[BPT92] Borie, R. B.; Parker, R. G.; Tovey, C. A.: *Automatic generation of linear-time algorithms from predicate calculus description of problems on recursively constructed graph families.* Algorithmica 7, 5&6 (1992).

[CMN03] Calkin, N.; Merino, C.; Noble, S.; Noy, M.: *Improved Bounds for the Number of Forests and Acyclic Orientations in the Square Lattice.* The Electronic Journal of Combinatorics, 10, 1 #R4 (2003).

[CaP01] Cancela, H.; Petingi, L.: *Diameter constrained network reliability: exact evaluation by factorization and bounds.* In: ICIL'2001 - International Conference on Industrial Logistics, Okinawa, Japan (2001), 359–366.

[CaL96] Carlier, J.; Lucet, C.: A *decomposition algorithm for network reliability evaluation.* J. Discrete Appl. Math. 65 (1996), 141–156.

[CCS97] Chan, T. F.; Ciarlet Jr, J. P.; Szeto, W. K.: *On the optimality of the spectral bisection graph partition method.* SIAM Journal on Scientific Computing 18, 3 (1997), 43–948.

[CHW94] Chandrasekharan, N.; Hedetniemi, S. T.; Wimer, T. V.: *Enumeration techniques for certain k-terminal families of graphs.* Journal of Combinatorics, Information and System Science 19, 3-4 (1994), 121–138.

[Cou90] Courcelle, B.: *The monadic second-order logic of graphs I: Recognizable sets of finite graphs.* Information and Computation 85 (1990), 12–75.

[CDGT88] Cvetkovic, D. M.; Doob, M.; Gutman, I.; Torgasev, A.: *Recent results in the theory of graph spectra.* Annals Disc. Math. 36 (1988).

[DPT03] Dohmen, K.; Pönitz, A.; Tittmann, P.: *A new two-variable generalization of the chromatic polynomial.* J. Discrete Math. Theoret. Comput. Sci. 6 (2003), 69–90.

[Els97] Elsner, U.: *Graph partitioning – a survey.* SFB 393 preprint (1997), http://www.mathematik.tu-chemnitz.de/preprint/1997/SFB393_27.html.

[Eva76] Evans, J. R.: *Maximum flow in probabilistic graphs - the discrete case.* Networks 6 (1976), 161–183.

[FiM82] Fiduccia, C. M.; Mattheyses, R M.: *A linear time heuristic for improving network partitions.* General Electric Co., Technical Report 82CRD130, 1982.

[Fie98] Fiedler, M.: *Laplacian of graphs and algebraic connectivity, combinatorics and graph theory.* PWN-Polish Scientific Publishers, Warsaw, Banach Center Publications 25 (1989).

[Flu97] deFluiter, B.: *Algorithms for graphs of small treewidth.* University of Utrecht, PhD Thesis, 1997.

[GaJ79] Garey, M. R.; Johnson, D. S.: *Computers and intractability.* W. H. Freeman and Co., New York, ISBN 0-7167-1045, 1979.

[GoM95] Gondran, M; Minoux, M.: *Graphs and algorithms.* Wiley series in discrete mathematics, John Wiley & Sons, ISBN 0-471-10374-8, 1995.

[Jon82] Johnson, R.: *Some combinatorial aspects of network reliability.* University of California, Berkely, PhD Thesis, 1982.

[JuT02] Junghans, U.; Tittmann, P.: *A Comparison of Heuristic Methods for Path Decompositions of Graphs.* Operations Research 2002, Universität Klagenfurt, 2002.

[Kar72] Karp, R. M.: *Reducibility among combinatorial problems.* Plenum Press, Complexity of Computer Computations (1972), 85–103.

[KeL70] Kernigan, B. W.; Lin, S.: *An efficient heuristic procedure for partitioning graphs.* The Bell System Technical Journal 29, 2 (1970), 291–307.

[KBH01] Koster, A.; Bodlaender, H. L.; van Hoesel, S.: *Treewidth: Computational Experiments.* Elsevier Science Publishers, Electronic Notes in Discrete Mathematics 8 (2001).

[KyH01] Kyoko, S.; Hiroshi, I.: *Computation of the Tutte Polynomial and Network Reliability.* IPSJ SIGNotes ALgorithms 047-006 (2001).

[LYCK99] Lee, C. Y.; Yeh, Y. S.; Chen, D. J.; Ku, K. L.: *A probability model for reconstructing secret sharing under the internet environment.* Information Science 116 (1999), 109–127.

[Lee80] Lee, S. H.: *Reliability evalution of a flow network.* IEEE Trans. Rel. 29 (1980), 24–26.

[LMC00] Lucet, C.; Manouvrier, J.-F.; Carlier, J.: *Evaluating network reliability and 2-edge-connected reliability in linear time for bounded pathwidth graphs.* Algorithmica 27 (2000), 316–336.

[Mon] Monagan, M.: http://www.mapleapps.com/categories/mathematics/combinatorics/html/relpoly.html.

[Neg87] Negami, S.: *Polynomial invariants of graphs.* Transactions of the American Mathematical Society 299 (1987), 601–622.

[PoS86] Politof, T.; Satyanarayana, A.: *Efficient algorithms for reliability analysis of planar networks - a survey.* IEEE Trans. Rel. 35 (1986), 252–259.

[PoS90] Politof, T.; Satyanarayana, A.: *A Linear-Time Algorithm to Compute the Reliability of Planar Cube-Free Networks.* IEEE Trans. Rel. 39, 5 (1990), 557–563.

[PST92] Politof, T; Satyanarayana, A.; Tung, L.: *An $\mathcal{O}(n \log n)$ algorithm to compute the all-terminal reliability of $K_5, K_{2,2,2}$-free networks.* IEEE Trans. Rel. 41, 4 (1992), 512–517.

[Poe99] Pönitz, A.: *Computing invariants in graphs of small bandwidth.* Mathematics and Computers in Simulation 49 (1999), 179–191.

[Poe02] Pönitz, A.: *Graph editor.* http://mathematik.htwm.de/software/gedit.

[Poe03] Pönitz, A.: *From edge decomposition formulae to composition algorithms.* Springer-Verlag Berlin Heidelberg NewYork, ISBN 3-540-00387-8, Operation Research Proceedings (2003), 383–388.

[SaP01] Sans, O.; Pönitz, A.: *Computation of the reconstruction probability of secret sharing schemes based on the K-terminal reliability.* Poster contribution to the Euro-Conference on Combinatorics, Graph Theory and Applications, Comb01, 2001.

[SaW82] Satyanarayana, A.; Wood, R. K.: *Polygon-to-chain reductions and network reliability.* University of California, Research Report ORC 82-4 (1982).

[Sch89] Scheffler, P.: *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme.* PhD thesis, Akademie der Wissenschaften der DDR, Berlin, 1989.

[See85] Seese, D.: *Tree-partite graphs and the complexity of algorithms.* Springer-Verlag Berlin, FCT'85, L. Budach (ed.): LNCS 199 (1985), 412–421.

[SIT98] Sekine, K.; Imai, H.; Tani, S.: *Computing the Tutte polynomial of a graph and the Jones polynomial of a knot of moderate size,* Extended Abstract, http://naomi.is.s.u-tokyo.ac.jp/~sekine/, 1998.

[Slo03] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences.* Published electronically at http://www.research.att.com/~njas/sequences, 2003.

[StC97] Strayer, H. J.; Colbourn, C. J.: *Bounding flow performance in probabilistic weighted networks.* IEEE Trans. Rel. 46, 1 (1997), 3–10.

[RoS86] Robertson N.; Seymour, P. D.: *Graph minors. II. Algorithmic aspects of tree-width.* J. Algorithms 7 (1986), 309–322.

[Ros77] Rosenthal, A.: *Computing the reliability of complex networks.* SIAM J. Appl. Math. 32 (1977), 384–393.

[TeP97] Telle, J. A.; Proskurowski, A.: *Algorithms for vertex partitioning problems on partial k-trees.* SIAM J. Disc. Math. 10, 4 (1997), 259–555.

[Thi00] Thilikos, D. M.: *Algorithms and obstructions for linear-width and related search parameters.* Discrete Applied Mathematics 105, 1-3 (2000), 239–271.

[TSB01] Thilikos, D. M.; Serna, M. J.; Bodlaender, H. L.: *A Polynomial Time Algorithm for the Cutwidth of Bounded Degree Graphs with Small Treewidth.* Lecture Notes in Computer Science 2161 (2001), 380+.

[Tut74] Tutte, W.: *Codichromatic graphs.* J. Combin. Theory Ser. B 16 (1974), 355–451.

[Val77] Valiant, L. G.: *The complexity of enumeration and reliability problems.* SIAM Journal of Computing 8, 3 (1979), 410–421.

[YLK02] Yeh, Fu-Min; Lu, Shyue-Kung; Kuo, Sy-Yen: *OBDD-based evaluation of k-terminal network reliability.* IEEE Trans. Rel. 51, 4 (2002), 443– 451.

Thesen

- Die in der vorliegenden Arbeit beschriebenen *Kompositionsalgorithmen* demonstrieren, dass bekannte theoretische Ergebnisse auf dem Gebiet der Bestimmung von polynomialen Grapheninvarianten und dem Lösen von Entscheidungs- und Zählproblemen in Graphen und Hypergraphen beschränkter Wegweite in einfache Algorithmen umgesetzt werden können.
- Eine Modifizierung der Algorithmen zur Lösung bestimmter Optimierungsprobleme in Graphen sowie zur Durchführung von Berechnungen in stochastischen Netzstrukturen ist möglich. Besonders im letztgenannten Gebiet können Kompositionsalgorithmen die Leistung bekannter Spezialalgorithmen zum Teil deutlich übertreffen.
- Zur Erzeugung neuer Kompositionsalgorithmen zur Lösung weiterer Probleme kann die vorgestellte *Kompositionsmethode* eingesetzt werden. Diese Methode kann ausgehend von einer geeigneten formalen Beschreibung des zu lösenden Problems vollkommen automatisch ablaufen. Die notwendigen formalen Beschreibung können dabei aus bekannten Dekompositions- oder Splittingformeln abgeleitet, oder aber manuell erstellt werden. Trotz des evtl. verbleibenden manuellen Anteils ist die Methode vergleichsweise einfach anzuwenden und führt in vielen Fällen zu – zumindest aus praktischer Sicht – akzeptablen Ergebnissen.
- Kompositionsalgorithmen können prinzipiell für beliebige endliche ungerichtete Graphen und Hypergraphen eingesetzt werden. Wenn für eine bestimmte Problemstellung polynomiale Algorithmen für Graphen beschränkter Baumweite bekannt sind, erzeugt die Kompositionsmethode in der Regel polynomiale Algorithmen für Graphen beschränkter Wegweite.
- Die Schließung der „theoretischen Lücke“ zwischen Graphen beschränkter Baumweite und Graphen beschränkter Wegweite ist durch Modifikation der Kompositionsalgorithmen möglich. Die experimentellen Ergebnisse legen jedoch nahe, dass sich durch eine solche Erweiterung die Kompliziertheit sowohl der notwendigen formalen Beschreibungen als auch der Algorithmen selbst erheblich erhöhen. Dadurch werden zwei wesentliche Vorteile der unmodifizierten Kompositionsmethode aufgegeben, so dass diese Erweiterung praktisch nicht sinnvoll erscheint.
- Die in den formalen Beschreibungen verwendeten *Konstellationen* und *Bewertungen* stellen eine Alternative zu bekannten allgemeinen Beschreibungssprachen von Graphenproblemen wie zum Beispiel der monadischen Logik zweiter Ordnung dar. Sie zeichnen sich durch hohe Flexibilität aus, so dass die formale Beschreibung „anwendungsnah“ erfolgen kann, sind aber andererseits durch die direkte Kopplung an die Kompositionsmethode auch „algorithmennah“. Damit ist die Kompositionsmethode sehr gut für die Lösung praktischer Aufgabenstellungen und Spezialanwendungen geeignet.
- Die einheitliche Beschreibung von Problemstellungen und Lösungsverfahren durch Konstellationen ermöglicht die problemlose Kombination von Kompositions-, Splitting- und Reduktionsansätzen und damit die Verwendung problemspezifischer Optimierungen als Vorbereitungsschritt.