Property-Testing in gradbeschränkten gerichteten Graphen unter Nichtsichtbarkeit eingehender Kanten

Dissertation

Zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Technischen Universität Dortmund
an der Fakultät für Informatik

von
Frank Hellweg

Dortmund, November 2013

Tag der mündlichen Prüfung: 14.11.2013

Dekan

Professor Dr. Gernot A. Fink

Gutachter

Erstes Gutachten: Professor Dr. Christian Sohler
Zweites Gutachten: Dr. habil. Matthias Westermann

Danksagung

Ich danke meinem Doktorvater Christian Sohler dafür, dass er mir die Möglichkeit gegeben hat, in diesem großartigen Forschungsgebiet mit ihm arbeiten zu dürfen – und der Deutschen Forschungsgemeinschaft und dem European Research Council dafür, dass sie dies als sinnvolles Vorhaben angesehen und finanziert haben[1] –, und dass er mich während der ganzen Promotionszeit mit viel Geduld betreut hat. Ich danke Matthias Westermann für die Begutachtung dieser Arbeit. Weiter danke ich meinen Kollegen und ehemaligen Kollegen in Dortmund wie Bonn sowohl für unzählige fachliche Gespräche – zuvorderst Melanie, mit der zusammen ich an dem schönen Ergebnis zu euklidischen Spannern forschen durfte – als auch für die schöne Zeit, die ich an den beiden Lehrstühlen erleben durfte. Und schließlich – und lediglich in Hinsicht schriftlicher Reihenfolge zuletzt – danke ich meinen Eltern, Verwandten, Freunden und allen anderen, die mir dabei geholfen haben, auch die eine oder andere Periode ohne erkennbaren Fortschritt mit Freude zu überwinden!

[1] DFG-Projekt SO 514/3; ERC Starting Grant 307696

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Ergebnisse der Arbeit 4

2 Grundlagen **7**
2.1 Mathematische Grundlagen 7
2.1.1 Graphen 7
2.1.2 Wahrscheinlichkeitsrechnung 11
2.1.3 Sonstige Abschätzungen 17
2.2 Property-Testing in Graphen 18
2.2.1 Techniken für Property-Testing in dünn besetzten Graphen 23
2.2.2 Techniken für Property-Testing in gerichteten Graphen 39
2.2.3 Techniken für untere Schranken 41
2.2.4 Techniken für Property-Testing in geometrischen Graphen 49
2.3 Sampling 55

3 Testen von Subgraphfreiheit **59**
3.1 Ein Property-Testing-Algorithmus für H-Freiheit 59
3.2 Ein Property-Testing-Algorithmus für 3-Stern-Freiheit 63
3.3 Untere Schranken für 3-Stern-Freiheit 78

4 Testen von starkem Zusammenhang **85**

5 Testen von gerichteten euklidischen Spannern **101**
5.1 Ein Property-Testing-Algorithmus für euklidische Spanner 102
5.2 Analyse des Algorithmus TesteSpanner 104
5.3 Eine untere Schranke für das Testen von euklidischen Spannern 119

6 Fazit und Ausblick **123**

Literaturverzeichnis **129**

1 Einleitung

Der Datenbestand der Menschheit hat sich in den vergangenen zehn Jahren vervielfacht. Dazu beigetragen hat vor allem das Internet: Durch die steigende Vernetztheit der Weltbevölkerung war es immer mehr Menschen möglich, Informationen im Internet zur Verfügung zu stellen, und durch das Aufkommen sozialer Netzwerke hat das Bestreben dazu breite Bevölkerungsschichten erreicht. Auch durch automatische Prozesse wie das Mitprotokollieren von Suchanfragen wurden große Datenbestände angelegt. Dies alles führt dazu, dass die Datenmenge recht unkontrolliert anwächst: Es entstehen Anwendungsfälle, in denen es schwierig ist, in einer großen Menge „unnützer“ Daten die gewünschte Information zu finden.

Dies zu ermöglichen ist die Aufgabe effizienter Algorithmen. Allerdings stellt die schiere Masse an Daten das traditionelle Verständnis von Effizienz in Frage: Bei Datenbeständen in einer Größe, wie sie die größten sozialen Netzwerke oder Google angehäuft haben, kann selbst ein Linearzeitalgorithmus zu langsam sein, um ihn in der Praxis einzusetzen. Es gibt verschiedene algorithmische Ansätze, um dieses Problems Herr zu werden:

Zur Zeit in der Praxis sehr beliebt ist die parallele Verarbeitung von Daten auf einer großen Anzahl von Computern; die Beliebtheit ist nicht zuletzt auf das von Google entwickelte *Map-Reduce*-Paradigma [32] und entsprechende Softwareumsetzungen wie Hadoop zurückzuführen: Map-Reduce erlaubt es, Datenverarbeitungsaufgaben sehr einfach zu parallelisieren, insbesondere ohne sich um die technischen Details kümmern zu müssen. Ein Problem dieses Ansatzes ist allerdings, dass er den insgesamten Rechenaufwand nicht verringert; dem Anwachsen der Datenmenge wird letztlich mit dem Anschaffen teurer Hardware begegnet.

Streaming (siehe z.B. [55] für eine Einführung) ist eine Methode, die einem Algorithmus Zugriff auf die gesamten Eingabedaten gibt; allerdings treffen diese Daten sukzessive und in nicht steuerbarer Reihenfolge ein, und ein Streamingalgorithmus darf nur sublinear viel Speicherplatz verwenden. Das bedeutet, dass ein komplettes Zwischenspeichern der Daten nicht möglich ist: Eine intelligente Strategie, eine problemspezifische Auswahl oder ein „Destillat“ der eingelesenen Daten zu speichern, ist ein wesentlicher Bestandteil eines Streamingalgorithmus. Streaming ist nicht zuletzt dann interessant, wenn Daten gleich bei ihrer Entstehung analysiert werden sollen, beispielsweise weitergeleitete Pakete an einem Router.

Eine neuere Technik für den Umgang mit großen Daten sind *Local-Computation*-Algorithmen [69, 7]. Solche Algorithmen können ein Problem für eine Teilmenge der Daten lösen, mit dem Ziel, eine Aussage über Elemente dieser Teilmenge zu treffen. Beispielsweise ist es so möglich, durch das Betrachten eines kleinen Teils eines Graphen festzustellen, ob eine bestimmte Kante in einem zufällig gewählten inklusionsweise maximalen Matching vorhanden ist. Wichtig ist hierbei, dass weitere Anfragen konsistent

mit den bisher getätigten sind, im Beispiel des Matchings also ein wiederholtes Anfragen für alle Kanten des Graphen tatsächlich ein inklusionsweise maximales Matching liefern würde.

Eine weitere – und die dieser Arbeit zugrunde liegende – Technik für das Lösen von Problemen auf großen Datenmengen stellen *Sublinearzeitalgorithmen* dar. Hierbei handelt es sich um Algorithmen, die nur einen kleinen, sublinearen Teil der Eingabedaten lesen, um ein Problem zu lösen; dadurch wird auch eine sublineare Laufzeit angestrebt. Stellt man keine besonderen Anforderungen an die Eingabe eines solchen Algorithmus – beispielsweise wird für eine in logarithmischer Laufzeit mögliche binäre Suche angenommen, dass die eingegebenen Daten sortiert sind –, so ist es in der Regel nicht möglich, ein exaktes Ergebnis zu garantieren. Zusätzlich können Probleme, deren Lösungen im Worst-Case eine lineare Größe haben, nicht direkt gelöst werden, da die Erzeugung einer Ausgabe linearer Länge mindestens lineare Laufzeit erfordern würde.

Es gibt zwei wesentliche Typen von Algorithmen in diesem Bereich, sublineare Approximationsalgorithmen und Property-Testing-Algorithmen. Algorithmen des ersten Typs liefern im Regelfall eine Approximation *des Werts* der optimalen Lösung des jeweiligen Optimierungsproblems, letztere lösen eine relaxierte Version eines Entscheidungsproblems. In beiden Fällen handelt es sich um randomisierte Algorithmen, so dass sie zusätzlich eine kleine, konstante Fehlerwahrscheinlichkeit haben dürfen – es handelt sich also um *Monte-Carlo*-Algorithmen.

Thema dieser Arbeit ist Property-Testing für Eigenschaften gerichteter Graphen, und zwar in einem Modell, das es Algorithmen nicht erlaubt, die eingehenden Kanten eines Knotens zu erfragen. Gerichtete Graphen sind gut geeignet, diverse Typen von Netzen zu modellieren, beispielsweise Straßen- und Versorgungsnetze, Linkstrukturen im Internet – der entsprechende Graph wird *Webgraph* genannt – oder Freundschaftsbeziehungen in manchen sozialen Netzen mit gerichteten Beziehungen – beispielsweise die „Folgen"-Beziehung in Twitter. Auch Funkverbindungen in Ad-Hoc- oder Sensornetzen können aufgrund unterschiedlicher Sende- und Empfangsleistungen der Teilnehmer gerichtete Beziehungen sein.

Fast alle genannten Anwendungsbeispiele haben gemein, dass dort sehr große Netze vorliegen können, die außerdem typischerweise *dünn besetzt* sind, das heißt die Anzahl der Kanten ist linear in der Anzahl der Knoten, nicht quadratisch. Einem einzelnen Knoten muss nicht bekannt sein, dass er eingehende Verbindungen von anderen Knoten hat: Eine Webseite kennt beispielsweise nicht die Links anderer Webseiten, die auf sie verweisen. Erreicht ein Crawler eine Webseite, dann sieht er also diese Links nicht, sondern muss dafür zunächst unabhängig die jeweilige verlinkende Webseite entdecken. Die Frage ist, ob man nun das ganze Internet erforschen muss, um Aussagen über seine Struktur zu treffen, oder ob dies schon mit wenigen lokal explorierten Bereichen möglich ist – oder allgemeiner: Wieviel Information kann man in einem gerichteten Graphen durch lokale Exploration erlangen, wenn die eingehenden Kanten der Knoten nicht sichtbar sind?

Property-Testing. Property-Testing ist grob gesagt eine Technik zum Lösen von Entscheidungsproblemen, die zu Gunsten einer sublinearen Laufzeit auf Genauigkeit ver-

zichtet: Zum einen handelt es sich um randomisierte Algorithmen, die eine kleine, beschränkte Fehlerwahrscheinlichkeit von zum Beispiel $\frac{1}{3}$ haben dürfen. Haben Property-Testing-Algorithmen außerdem die Eigenschaft, dass sie alle diejenigen Eingaben mit Wahrscheinlichkeit 1 akzeptieren, die die angefragte Eigenschaft besitzen, so sprechen wir von *einseitigem Fehler*, ansonsten von *zweiseitigem Fehler*.

Zum anderen müssen Property-Testing-Algorithmen nicht zwischen „ähnlichen" Eingaben unterscheiden können; als ähnlich werden Eingaben definiert, die sich nur in einem kleinen linearen Bruchteil ihrer Beschreibung unterscheiden. Diese Einschränkung bedeutet insbesondere, dass ein Property-Testing-Algorithmus eine Eingabe, die die angefragte Eigenschaft nicht hat, nicht ablehnen muss, falls sie einer Eingabe mit der Eigenschaft ähnlich ist; stattdessen darf er für eine solche Eingabe eine beliebige Antwort geben. Diese Ähnlichkeitsbeziehung wird durch einen Näheparameter ϵ parametrisiert: Im Beispiel dünn besetzter gerichteter Graphen sind zwei Graphen mit n Knoten und durch $D \in \mathbb{N}$ beschränktem Knotenausgangsgrad ähnlich, falls grob gesagt durch Einfügen oder Löschen von höchstens ϵDn Kanten der eine Graph in den anderen überführt werden kann.

Property-Testing-Algorithmen erhalten keinen direkten Zugriff auf die Eingabe, sondern können einzelne Elemente mit Anfragen an eine „Orakelfunktion" auslesen, die Zugriff auf die Eingabe hat. Im Fall dünn besetzter gerichteter Graphen kann ein Algorithmus so Adjazenzlisteneinträge beliebiger Knoten anfragen; im von uns untersuchten Modell sind in den Adjazenzlisten der Knoten ausschließlich die ausgehenden Kanten sichtbar. Das wichtigste Maß für die Effizienz eines Property-Testing-Algorithmus ist seine *Anfragekomplexität*, das heißt die Anzahl von Anfragen, die er im Worst-Case für eine Eingabe der Größe n benötigt.

Eine wesentliche Technik im Property-Testing für *ungerichtete* dünn besetzte Graphen ist, eine Anzahl von Knoten zufällig gleichverteilt zu wählen, ausgehend von diesen Knoten kleine Areale des Graphen zu erforschen und dann eine Entscheidung abhängig vom so explorierten Subgraphen zu treffen; zumeist ist das Ziel dabei, innerhalb der explorierten Areale einen Zeugen dafür zu finden, dass der Graph die angefragte Eigenschaft nicht hat und in diesem Fall abzulehnen. Diese Technik führt zu Property-Testing-Algorithmen mit einseitigem Fehler, denn falls kein solcher Zeuge gefunden wird, sollte der eingegebene Graph akzeptiert werden; somit werden nur Graphen abgelehnt, die die angefragte Eigenschaft nicht besitzen.

In gerichteten dünn besetzten Graphen ist diese Technik nicht so einfach umzusetzen, falls, wie oben angenommen, nur die eingehenden Kanten der Knoten sichtbar sind: Lokale Explorierungen können unter dieser Bedingung Knoten nicht direkt finden, wenn sie nur über eingehende Kanten der bereits erforschten Knoten mit diesen verbunden sind. Darüber hinaus kann für einen Knoten nicht einfach festgestellt werden, ob er komplett erforscht ist: Solange weniger eingehende Kanten des Knotens bekannt sind als die zulässige Maximalzahl, ist es möglich, dass es noch weitere, bisher unbekannte eingehende Kanten gibt. Somit sind die Informationen, die mit Hilfe einzelner lokaler Explorierungen gewonnen werden können, unvollständig und für viele Probleme nicht ausreichend. Es müssen also neue Techniken für Property-Testing-Algorithmen in diesem Modell entwickelt werden. Dies anhand einer Auswahl von Problemen zu tun ist das Hauptziel dieser Arbeit und der erste Schritt auf dem Weg zu einer Klassifizierung der

testbaren Eigenschaften in diesem Modell.

Die Organisation der Arbeit ist wie folgt: In Kapitel 2 folgt auf einige technische Grundlagen aus der Graphentheorie und der Stochastik ein knapper Überblick über den Forschungsbereich des Property-Testing in Graphen; dabei liegt der Fokus im Wesentlichen auf dem Modell dünn besetzter Graphen und den Techniken, die in diesem Modell Anwendung finden. Zudem wird die Frage diskutiert, inwieweit die Techniken für Property-Testing in dünn besetzten ungerichteten Graphen auf gerichtete Graphen übertragbar sind. Darauf folgt in den Kapiteln 3, 4 und 5 die Vorstellung der vom Verfasser im Rahmen der Dissertation erarbeiteten Ergebnisse. Die Arbeit schließt in Kapitel 6 mit einer Diskussion der Ergebnisse im Hinblick auf mögliche weitere Forschungsziele.

1.1 Ergebnisse der Arbeit

Die Ergebnisse der Kapitel 3, 4 und 5 sind den Veröffentlichungen [48] und [49] entnommen, deren erstere in Zusammenarbeit mit Melanie Schmidt und Christian Sohler entstanden ist und die zweitgenannte in Zusammenarbeit mit Christian Sohler. Der Beitrag des Autors dieser Schrift zum erstgenannten Ergebnis ist mit mindestens einem Drittel des Gesamtaufwands zu beziffern, der Beitrag zum zweiten Ergebnis mit mindestens der Hälfte. Die Arbeiten beschäftigen sich mit den folgenden drei Problemen in gerichteten gradbeschränkten Graphen:

Subgraphfreiheit. Zunächst beschäftigen wir uns mit dem Problem der Subgraphfreiheit. Ein Graph G heißt H-frei, falls der Graph H nicht als Subgraph von G vorkommt. Wir untersuchen dieses Problem zunächst für allgemeine verbotene Subgraphen und dann für Orientierungen von 3-*Sternen*, eine bestimmte Klasse von Subgraphen. Im allgemeinen Fall zeigen wir, dass für beliebige Subgraphen H ein Test auf H-Freiheit mit einseitigem Fehler und einer Anfragekomplexität von $\mathcal{O}(n^{1-1/k})$ möglich ist. Dabei ist k die Anzahl von „Quellkomponenten“ in H, das heißt die Anzahl von starken Zusammenhangskomponenten ohne eingehende Kante.

Eine spezielle Klasse von verbotenen Subgraphen ist die aller Orientierungen von 3-Sternen, das heißt Graphen, in denen drei Außenknoten mit einem Zentralknoten verbunden sind. Auf Freiheit aller Graphen dieser Klasse lässt sich in schwach zusammenhängenden Graphen effizienter testen: Wir zeigen, dass dies mit zweiseitigem Fehler und mit einer Anfragekomplexität von $\mathcal{O}(n^{1/2})$ möglich ist; direktes Anwenden des Property-Testers für allgemeine Subgraphfreiheit würde eine Komplexität von $\mathcal{O}(n^{2/3})$ ergeben, da eine der möglichen Orientierungen von 3-Sternen drei zum Zentrumsknoten gerichtete Kanten hat und damit drei Quellkomponenten. Der Algorithmus kombiniert eine Kollisionsstatistik auf den Zielknoten zufällig gewählter Kanten mit anderen statistischen Merkmalen, die durch die Problemstruktur motiviert sind; dadurch müssen nur einfache Kollisionen gezählt werden anstatt von Dreifachkollisionen, mit deren Hilfe 3-Sterne mit nach innen gerichteten Kanten direkt identifiziert werden könnten.

Wir zeigen für das Testen von 3-Stern-Freiheit außerdem zwei untere Schranken: Falls die Menge der möglichen Eingabegraphen auf schwach zusammenhängende Graphen be-

schränkt ist, benötigt jeder Property-Testing-Algorithmus $\Omega(n^{1/2})$ Anfragen, und falls die Eingabe ein beliebiger Graph sein kann, sind $\Omega(n^{2/3})$ Anfragen nötig. Das bedeutet, dass der vorgestellte Algorithmus asymptotisch optimal ist und dass die Voraussetzung schwach zusammenhängender Eingabegraphen nicht fallen gelassen werden kann.

Starker Zusammenhang. Das zweite Problem, mit dem wir uns beschäftigen, ist starker Zusammenhang. Ein gerichteter Graph ist stark zusammenhängend, wenn es von jedem seiner Knoten zu jedem anderen Knoten einen gerichteten Pfad gibt. Eine starke Zusammenhangskomponente ist daher ein Zeuge dafür, dass ein Graph diese Eigenschaft nicht besitzt, falls sie keine eingehenden oder keine ausgehenden Kanten hat. Die Schwierigkeit bei diesem Problem ist, dass die eingehenden Kanten einer starken Zusammenhangskomponente in unserem Modell nicht sichtbar sind und daher auch nicht direkt nachgewiesen werden kann, dass keine solchen Kanten existieren. Wir betrachten zunächst ein vereinfachtes Problem, in dem wir annehmen, dass es bei weit von Zusammenhang entfernten Graphen sehr viele Zusammenhangskomponenten gibt, die nur aus einem einzigen Knoten bestehen und keine eingehende Kante haben. Wir setzen dann eine Kollisionsstatistik ein, mit der wir Schätzwerte für die Anzahl von Knoten mit einer bestimmten Anzahl eingehender Kanten ermitteln; dies liefert auch eine Schätzung der Anzahl der Knoten ohne eingehende Kanten, und wenn diese Schätzung hoch genug ist, wird der Eingabegraph abgelehnt.

Wir zeigen schließlich für allgemeine Graphen, dass eine Reduktion existiert, die lokal konstruierbar ist und die kleine starke Zusammenhangskomponenten ohne eingehende Kanten in einzelne Knoten eines neuen Graphen überführt; wird also der oben beschriebene Schätzalgorithmus auf dem reduzierten Graphen ausgeführt, dann kann die Existenz solcher starken Zusammenhangskomponenten im Ausgangsgraphen statistisch nachgewiesen werden. Das führt zu einem Property-Testing-Algorithmus mit einer Anfragekomplexität von $\mathcal{O}(n^{1-\epsilon/(3+\alpha)})$ für einen beliebigen Parameter $\alpha > 0$; wegen der statistischen Auswertung handelt es sich um einen Algorithmus mit zweiseitigem Fehler.

Es sei bemerkt, dass Ähnliches zuvor schon von Oded Goldreich entdeckt wurde: Er gibt im Anhang A.3 seines Survey-Artikels *Introduction to Testing Graph Properties*, der in [39] enthalten ist, eine Beweisskizze dazu, in der er eine etwas andere Reduktion verwendet und zu einer Anfragekomplexität von $\mathcal{O}(n^{1-\epsilon^2/16D})$ kommt. Eine ausführliche Diskussion dieses Ergebnisses findet sich am Ende von Kapitel 4. Bei der Erarbeitung und Veröffentlichung der in Kapitel 4 vorgestellten Ergebnisse war dem Autor dieser Arbeit die Existenz von Goldreichs Ergebnis nicht bekannt.

Euklidische Spanner. Als drittes Problem untersuchen wir schließlich das der euklidischen Spanner. Ein Graph G, dessen Knoten auch Punkte im euklidischen Raum sind – also ein geometrischer Graph –, ist ein euklidischer $(1+\delta)$-Spanner, falls es zwischen allen Paaren von Knoten einen Pfad in G gibt, der maximal $(1+\delta)$ mal so lang ist wie die euklidische Distanz zwischen den Knoten. Dieses Problem hat eine gewisse Verwandtschaft mit starkem Zusammenhang: Anstatt die Existenz eines beliebigen Pfades zwischen allen Knotenpaaren einzufordern, darf hier der kürzeste Pfad außerdem eine bestimmte Länge

nicht überschreiten. Interessant ist an diesem Problem nicht zuletzt, in wiefern die zusätzlichen geometrischen Informationen algorithmisch genutzt werden können. Wir zeigen hier zunächst, dass es ausreicht, Verletzungen der Spanner-Eigenschaft in beschränkten geometrischen Arealen zu suchen; solche Areale können geschickt konstruiert werden, so dass sie auch eine beschränkte Anzahl von Knoten enthalten. Unser Algorithmus zieht also zufällig und gleichverteilt eine Anzahl von Knoten und prüft für alle Paare von nah beieinanderliegenden gezogenen Knoten, ob zwischen ihnen die Spanner-Eigenschaft gilt; dies geschieht mit Hilfe von Dijkstras Algorithmus [33], der nach Erreichen einer bestimmten Anzahl explorierter Knoten abgebrochen wird. Dies führt zu einem Property-Testing-Algorithmus mit einer Anfragekomplexität von $\mathcal{O}(n^{1/2})$. Hier ermöglicht uns die Problemstruktur, auch in einem Problem in gerichteten Graphen die gleichen lokalen Explorierungstechniken einzusetzen wie in dem entsprechenden Problem in ungerichteten Graphen.

Zusätzlich zeigen wir für das Testen von euklidischen Spannern eine untere Schranke von $\Omega(n^{1/3})$ für die Anfragekomplexität von Property-Testing-Algorithmen mit einseitigem Fehler.

2 Grundlagen

2.1 Mathematische Grundlagen

In diesem Kapitel werden im Laufe der Arbeit benötigte mathematische Begriffe definiert und einige grundlegende mathematische Techniken erwähnt, die Anwendung finden.

2.1.1 Graphen

Ein Graph $G = (V, E)$ ist definiert durch eine Menge V von Knoten und eine Menge E von Kanten. Jede Kante bildet dabei eine Beziehung zwischen zwei der Knoten ab. Diese Beziehungen können entweder gerichtet oder ungerichtet sein: Im ersten Fall ist die Kantenmenge E eine Teilmenge von $\{(u, v) | u, v \in V, u \neq v\}$, und wir sprechen von einem *gerichteten Graphen*. Die Richtung einer Kante (u, v) ist dabei von u nach v. Im zweiten Fall ist E eine Teilmenge von $\{\{u, v\} | u, v \in V, u \neq v\}$, und wir sprechen von einem *ungerichteten Graphen*. Insbesondere sind für alle in dieser Arbeit vorkommenden Graphen Selbstkanten ausgeschlossen, das heißt Kanten (v, v) beziehungsweise $\{v, v\}$ für einen Knoten $v \in V$. Für einen Graphen G gebe $V(G)$ die Menge seiner Knoten, $E(G)$ die Menge seiner Kanten an.

Definition 2.1 (Inzidenz, Adjazenz)**.** *Sei $G = (V, E)$ ein ungerichteter Graph. Wir nennen zwei Knoten $u, v \in V$* adjazent*, falls es eine Kante $\{u, v\} \in E$ gibt. Wir nennen eine Kante $e \in E$ und einen Knoten $v \in V$* inzident*, falls $v \in e$. Wir nennen zwei Kanten $e_1, e_2 \in E$ inzident, falls es einen Knoten $v \in e_1$ gibt mit $v \in e_2$.*

Sei $G' = (V', E')$ ein gerichteter Graph. Wir nennen zwei Knoten $u, v \in V$ adjazent*, falls $(u, v) \in E$ oder $(v, u) \in E$ gilt. Wir nennen eine Kante $e \in E$ und einen Knoten $v \in V$* inzident*, falls v an e beteiligt ist, das heißt es gilt $e = (u, v)$ oder $e = (v, u)$ für einen Knoten $u \in V$. Wir nennen zwei Kanten $e_1, e_2 \in E$ inzident, falls es einen Knoten v gibt, der sowohl an e_1 als auch an e_2 beteiligt ist.*

Ein Pfad ist eine Menge von aufeinander folgenden Kanten eines Graphen.

Definition 2.2 (Pfad)**.** *Sei $G = (V, E)$ ein ungerichteter Graph. Ein (einfacher) Pfad in G ist eine Menge $P \subseteq E$ von Kanten, für die es eine Ordnung gibt, so dass*

- *aufeinanderfolgende Kanten der Ordnung adjazent sind;*
- *jeder Knoten von G bis auf zwei Knoten $u, v \in V$ entweder nicht oder genau zweimal in Kanten von P enthalten ist, nämlich in zwei aufeinanderfolgenden Kanten der Ordnung;*

- *u und v genau einmal in Kanten von P enthalten sind, u in der ersten und v in der letzten Kante der Ordnung.*

u und v nennen wir Start- *beziehungsweise* Zielknoten *von P.*

In gerichteten Graphen nennen wir eine Kantenmenge P einen Pfad bzw. gerichteten Pfad, *wenn die obigen Voraussetzungen gelten und eine der gültigen Ordnungen der Kanten zusätzlich sicherstellt, dass alle Kanten in der Reihenfolge der Ordnung gerichtet sind; das heißt wenn für jede Kante $e = (v_1, v_2)$ des Pfades die e vorhergehende Kante – falls eine solche Kante existiert – eine zu v_1 eingehende Kante ist und die nachfolgende Kante – falls sie existiert – eine von v_2 ausgehende. Hat mindestens eine Kante eine andere Richtung, so nennen wir P einen ungerichteten Pfad.*

Den Kanten eines Graphen G können Kantengewichte zugeordnet werden; diese werden als Funktion $w_G : E(G) \to \mathbb{R}$ angegeben. Einen für diese Arbeit wichtigen Spezialfall stellen dabei geometrische Graphen dar, bei denen jeder Knoten gleichzeitig ein Punkt im euklidischen Raum ist; die Kantengewichte ergeben sich aus den euklidischen Distanzen der beteiligten Knoten. Formal lässt sich dies wie folgt definieren:

Definition 2.3. *Sei $G = (P, E)$ ein gerichteter Graph mit Knotenmenge $P \subset \mathbb{R}^d$ und Kantenmenge $E \subseteq P^2$ und einer Gewichtsfunktion $d_G : V^2 \to \mathbb{N}$. G ist ein* geometrischer Graph, *falls $d_G(p, q) = ||p - q||_2$ für alle $(p, q) \in E$ gilt.*

Für zwei nicht adjazente Knoten p, q definieren wir $d_G(p, q)$ als die Länge des kürzesten Pfades zwischen p und q in G, d.h. $d_G(p, q) = \arg\min_{Pfade\ \mathcal{P}\ von\ p\ nach\ q} \sum_{(u,v)\in\mathcal{P}} w_G(u, v)$ für alle $(p, q) \notin E$.

Ungerichtete geometrische Graphen sind analog definiert.

Definition 2.4 (Knotengrad)**.** *Sei $G = (V_G, E_G)$ ein ungerichteter Graph; der Knotengrad oder kurz Grad eines Knotens $v \in V_G$ von G ist definiert als* $\deg(v) = |\{u \in V_G | \{u, v\} \in E_G\}|$*, also als die Anzahl von Kanten, zu denen v inzident ist.*

Sei $H = (V_H, E_H)$ ein gerichteter Graph; der eingehende Knotengrad oder kurz Eingangsgrad eines Knotens $v \in V_H$ von H ist definiert als $\deg_{in}(v) = |\{u \in V_G | (u, v) \in E_G\}|$*; der ausgehende Knotengrad oder kurz Ausgangsgrad von v ist* $\deg_{out}(v) = |\{u \in V_G | (v, u) \in E_G\}|$*. Der* ungerichtete Knotengrad *oder einfach Grad von v ist* $\deg(v) = \deg_{in}(v) + \deg_{out}(v)$.

Es sei bemerkt, dass der Grad eines Knotens eines ungerichteten Graphen größer sein kann als die Anzahl zu ihm adjazenter Knoten; das kann dann geschehen, wenn es zu mindestens einem weiteren Knoten Kanten in beiden Richtungen gibt.

Definition 2.5 (Unter- bzw. Subgraph)**.** *Seien $G = (V_G, E_G)$ und $H = (V_H, E_H)$ gerichtete Graphen. H ist ein Unter- bzw. Subgraph von G, falls es eine injektive Abbildung $f : V_H \to V_G$ gibt, so dass $(f(u), f(v)) \in E_G$ für alle $(u, v) \in E_H$.*

Eine Funktion f wie in dieser Definition bezeichnet auch ein *Vorkommen* eines Graphen H in einem Graphen G: Solch ein Vorkommen ist der Graph, der aus der Knotenmenge $\{f(v) | v \in V(H)\}$ und der Kantenmenge $\{(f(u), f(v)) | (u, v) \in E(H)\}$ gebildet wird.

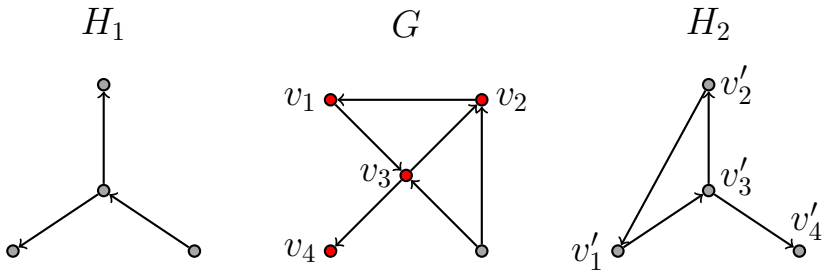

Abbildung 2.1: *Subgraphen: H_1 ist Subgraph von G, aber nicht induzierter Subgraph; H_2 ist der durch $v_1, \ldots, v_4$ induzierte Subgraph von G.*

Definition 2.6 (Induzierter Unter- bzw. Subgraph)**.** *Seien $G = (V_G, E_G)$ und $H = (V_H, E_H)$ gerichtete Graphen. H ist* induzierter *Unter- bzw. Subgraph von G, falls es eine injektive Abbildung $f : V_H \to V_G$ gibt, so dass $(f(u), f(v)) \in E_G$ genau dann, wenn $(u, v) \in E_H$ gilt.*

Ein *induziertes* Vorkommen eines Graphen H in einem Graphen G ist analog zum einfachen Vorkommen definiert. Außerdem sprechen wir für einen Graphen G und eine Knotenmenge $V \subseteq V(G)$ vom durch V induzierten Subgraphen von G, der aus der Knotenmenge V und der Kantenmenge $\{(u, v) \in V^2 | (u, v) \in E(G)\}$ besteht (siehe Abbildung 2.1). Wir nennen einen Graphen G *(induziert) H-frei*, falls H kein (induzierter) Subgraph von G ist.

Der duale Graph eines Graphen ersetzt die Knoten des Ausgangsgraphen durch Kanten und umgekehrt:

Definition 2.7 (Dualer Graph)**.** *Sei $G = (V, E)$ ein ungerichteter Graph. Der duale Graph von G ist definiert durch die Knotenmenge E und die Kantenmenge $\{\{e_1, e_2\} \subseteq E | \exists v \in e_1 \cap e_2\}$.*

Im Folgenden definieren wir Zusammenhang von Graphen und einige Begriffe, um die Zusammenhangskomponenten eines Graphen zu charakterisieren.

Definition 2.8 (Starker und schwacher Zusammenhang)**.** *Sei $G = (V, E)$ ein gerichteter Graph. Eine Knotenmenge $U \subseteq V$ nennen wir* stark zusammenhängend*, falls es für alle Knoten $u, v \in U$ einen gerichteten Pfad von u nach v gibt; wir nennen U* schwach zusammenhängend*, falls es für alle Knoten $u, v \in U$ einen ungerichteten Pfad von u nach v gibt.*

Wir nennen G stark bzw. schwach zusammenhängend, wenn V stark bzw. schwach zusammenhängend ist.

Wir bezeichnen einen gerichteten Graphen im Folgenden als *zusammenhängend*, falls er schwach zusammenhängend ist, und als *nicht zusammenhängend*, wenn er nicht schwach zusammenhängend ist; für Teilmengen seiner Knotenmenge verwenden wir die Begrifflichkeiten analog. Einen ungerichteten Graphen beziehungsweise eine Teilmenge seiner Knoten bezeichnen wir als zusammenhängend, wenn alle Paare von Knoten mit einem

Pfad verbunden sind und als nicht zusammenhängend, falls es ein Knotenpaar gibt, für das kein verbindender Pfad existiert.

Definition 2.9 (Starke und schwache Zusammenhangskomponenten). *Sei $G = (V, E)$ ein gerichteter Graph. Wir nennen eine Knotenmenge $U \subseteq V$* starke/schwache Zusammenhangskomponente*, falls folgende Bedingungen erfüllt sind:*

1. *U ist stark/schwach zusammenhängend.*
2. *Für alle nicht leeren Knotenmengen $\tilde{U} \subseteq V \backslash U$ ist $U \cup \tilde{U}$ nicht stark/schwach zusammenhängend.*

In ungerichteten Graphen sind Zusammenhangskomponenten analog definiert; dort muss nicht zwischen starken und schwachen Zusammenhangskomponenten unterschieden werden.

Definition 2.10 (Quell- und Senkenkomponenten). *Sei $G = (V, E)$ ein gerichter Graph. Eine starke Zusammenhangskomponente $U \subseteq V$ von G heißt* Quellkomponente*, falls es keine Kante von $V \backslash U$ nach V gibt; U heißt* Senkenkomponente*, falls es keine Kante von U nach $V \backslash U$ gibt.*

Wir werden solche Zusammenhangskomponenten auch kurz *Quelle* oder *Senke* nennen, falls aus dem Kontext ersichtlich ist, dass es sich nicht nur um einzelne Knoten handeln muss. Zusammenfassend für Quell- und Senkenkomponenten verwenden wir den Begriff *endständige Komponenten.*

Definition 2.11 (Orientierung von Graphen). *Sei $H = (V_H, E_H)$ ein ungerichteter Graph. Wir nennen einen gerichteten Graphen $G = (V_G, E_G)$ eine* Orientierung *von H, falls es eine bijektive Abbildung $f : V_H \to V_G$ gibt, so dass für alle $\{u, v\} \in E_H$ entweder $(u, v) \in \mathrm{E}_G$ oder $(v, u) \in \mathrm{E}_G$ gilt.*

Intuitiv gibt eine Orientierung eines ungerichteten Graphen jeder seiner Kanten eine Richtung.

Eine spezielle Klasse von Graphen, die wir in dieser Arbeit betrachten werden, ist die der k-Sterne beziehungsweise ihrer Orientierungen:

Definition 2.12 (k-Stern). *Wir nennen einen ungerichteten Graphen $H = (V, E)$ k-*Stern*, falls folgende Bedingungen erfüllt sind:*

1. *Es gibt genau einen Zentralknoten v und k Außenknoten $u \in V \backslash \{v\}$, so dass für alle Außenknoten u eine Kante $\{u, v\} \in E$ existiert.*
2. *H enthält keine weiteren Kanten.*

Die verbleibenden Knoten eines k-Sterns nennen wir *Außenknoten.* Grob gesagt handelt es sich bei einem k-Stern um einen ungerichteten Graphen, in dem ein Zentralknoten mit k äußeren Knoten verbunden ist, die wiederum keine weiteren Kanten untereinander haben. Wir werden uns insbesondere mit Orientierungen von k-Sternen beschäftigen, die als Subgraphen in gerichteten Graphen auftreten. Den Zentralknoten eines solchen Subgraphen nennen wir *k-Sternknoten*:

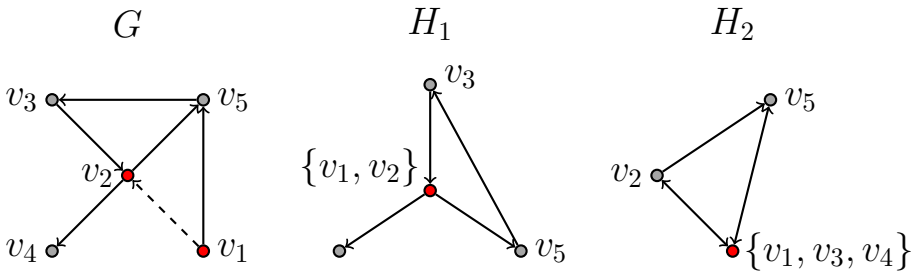

Abbildung 2.2: *Kontraktion: H_1 ist der Graph, der entsteht, wenn man in G die Knotenmenge $\{v_1, v_2\}$ kontrahiert. H_2 ist der Graph, der entsteht, wenn man in G die Knotenmenge $\{v_1, v_3, v_4\}$ kontrahiert.*

Definition 2.13 (k-Sternknoten)**.** *Sei G ein gerichteter Graph. Wir nennen $v \in V(G)$ k-*Sternknoten*, falls v der Zentralknoten einer Orientierung eines k-Sterns ist, der als Subgraph in G auftritt.*

Das bedeutet, dass ein Knoten ein k-Sternknoten ist, wenn er zu mindestens k weiteren Knoten adjazent ist.

Definition 2.14 (Kontraktion von Kanten und Subgraphen)**.** *Sei $G = (V, E)$ ein gerichteter Graph und sei $V' \subseteq V$ eine Knotenmenge. Der Graph $\mathcal{K}(G, V')$, der entsteht, wenn V' kontrahiert wird, ist definiert als der Graph, der die Knotenmenge V' durch einen einzelnen Knoten ersetzt; dieser erhält eine Kante zu jedem Knoten, zu dem einer der Teilknoten von V' in G eine Kante hatte und eine Kante von jedem Knoten, von dem aus in G eine Kante zu einem Knoten in V' bestand. Formal ist dies wie folgt definiert:*

$$\begin{aligned}\mathcal{K}(G, V') = \big(&\{v \in V \backslash V'\} \cup \{V'\},\\ &\{(u, v) \in E | u, v \in V \backslash V'\} \cup \{(u, V') | \exists v \in V', u \in V \backslash V' : (u, v) \in E\}\\ &\quad \cup \{(V', u) | \exists v \in V', u \in V \backslash V' : (v, u) \in E\}\big)\end{aligned}$$

Für eine Kante $e = (u, v)$ ist $\mathcal{K}(G, e) := \mathcal{K}(G, \{u, v\})$.

Kontraktion ist analog für ungerichtete Graphen definiert.

2.1.2 Wahrscheinlichkeitsrechnung

Für die grundlegenden Definitionen in diesem Unterkapitel folgen wir grob Bronstein et al. [20]. Sei E eine endliche, nichtleere Menge von Elementarereignissen, die sich gegenseitig ausschließen; dann nennen wir eine Teilmenge $A \subseteq E$ *Ereignis* und $\bar{A} = E \backslash A$ das *Gegenereignis* von A.

Ein Zufallsexperiment ist ein Vorgang, bei dem eines der Elementarereignisse eintritt, ohne dass der Ausgang vorhersagbar ist. Jedes Elementarereignis hat dabei die gleiche Wahrscheinlichkeit. Die Wahrscheinlichkeit, dass ein Ereignis $A \subseteq E$ eintritt, ist $\Pr[A] =$

$\frac{|A|}{|E|}$, und es gilt

$$\Pr[\bar{A}] = \frac{|\bar{A}|}{|E|} = \frac{|E \backslash A|}{|E|} = \frac{|E| - |A|}{|E|} = 1 - \Pr[A].$$

Für die elementaren Ereignisse E und $\emptyset$ gilt dementsprechend $\Pr[E] = 1 = 1 - \Pr[\emptyset]$.

Ein wichtiges Konzept ist die *bedingte Wahrscheinlichkeit*. Seien $A, B \subseteq E$ Ereignisse mit $B \neq \emptyset$; dann ist die bedingte Wahrscheinlichkeit $\Pr[A|B]$ von A unter B die Wahrscheinlichkeit, dass A eintritt unter der Voraussetzung, dass auch B eintritt; formell gilt $\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]} = \frac{|A \cap B|}{|B|}$; insbesondere gilt dadurch natürlich $\Pr[A|E] = \Pr[A]$.

Definition 2.15. *Zwei Ereignisse $A, B \subseteq E$ heißen* unvereinbar, *falls $A \cup B = \emptyset$. A und $B \neq \emptyset$ heißen* unabhängig, *falls $\Pr[A|B] = \Pr[A]$ gilt.*

Sind A und B unvereinbare Ereignisse, dann ist die Wahrscheinlichkeit ihrer Vereinigung die Summe der Einzelwahrscheinlichkeiten, denn es gilt dann

$$\Pr[A \cup B] = \frac{|A + B|}{|E|} = \frac{|A| + |B|}{|E|} = \Pr[A] + \Pr[B].$$

Sind A und B hingegen beliebige Ereignisse und damit nicht unbedingt unvereinbar, so gilt

$$\Pr[A \cup B] = \frac{|A| + |B| - |A \cap B|}{|E|} \leq \Pr[A] + \Pr[B].$$

Diese Abschätzung lässt sich auf eine beliebige Anzahl von Ereignissen erweitern und wird im Englischen häufig als *Union Bound* bezeichnet, ein Begriff, den wir für diese Arbeit übernehmen werden.

Sind A und B unabhängige Ereignisse, so ist die Wahrscheinlichkeit ihres Schnitts das Produkt ihrer Einzelwahrscheinlichkeiten, denn es gilt nach obiger Definition:

$$\Pr[A \cap B] = \frac{|A \cap B|}{|E|} = \frac{\Pr[A|B] \cdot |B|}{|E|} = \Pr[A] \cdot \frac{|B|}{|E|} = \Pr[A] \cdot \Pr[B].$$

Die Unabhängigkeitsbeziehung zwischen zwei nichtleeren Ereignissen A und B ist bijektiv, denn es gilt

$$\Pr[A|B] = \Pr[A] \Leftrightarrow \frac{|A \cap B|}{|B|} = \frac{|A|}{|E|} \Leftrightarrow \frac{|A \cap B|}{|A|} = \frac{|B|}{|E|} \Leftrightarrow \Pr[B|A] = \Pr[B].$$

Eine Zufallsvariable X ist eine Funktion, die jedem Ausgang eines Zufallsexperiments einen Wert zuordnet. Die Wahrscheinlichkeit $Pr[X = x]$, dass X einen Wert x annimmt, ist die Wahrscheinlichkeit, dass eines der Elementarereignisse eintritt, denen X den Wert x zuordnet. Hat eine Zufallsvariable X nur endlich viele oder abzählbar unendlich viele mögliche Werte, nennen wir sie *diskret*, ansonsten *stetig*. Falls die möglichen Werte 0 und 1 sind, nennen wir X *Indikatorzufallsvariable* für ein Ereignis A; A ist dabei die Menge aller Elementarereignisse, denen X den Wert 1 zuweist.

Zwei diskrete Zufallsvariablen X_1 und X_2 sind unabhängig voneinander, wenn

$$\Pr[X_1 = a \wedge X_2 = b] = \Pr[X_1 = a] \cdot \Pr[X_2 = b]$$

für alle Werte a, die X_1 annehmen kann und alle Werte b, die X_2 annehmen kann, gilt. Sind X_1 und X_2 Indikatorzufallsvariablen, dann folgt daraus, dass sie unabhängig sind, wenn die entsprechenden Ereignisse unabhängig voneinander sind. Für nicht diskrete Zufallsvariablen X_1 und X_2 muss für Unabhängigkeit die obige Beziehung für alle Teilmengen der Bildmengen von X_1 und X_2 gelten.

Der *Erwartungswert* und die *Varianz* von Zufallsvariablen sind wichtige Größen für die Vorhersage von Zufallsexperimenten:

Definition 2.16 (Erwartungswert, Varianz)**.** *Sei X eine Zufallsvariable, die Werte aus einer Menge M annehmen kann. Der Erwartungswert von X ist definiert als*

$$\mathrm{E}[X] = \sum_{x \in M} x \cdot \Pr[X = x].$$

Die Varianz von X ist definiert als $\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$.

Im Zusammenhang mit dem Erwartungswert sprechen wir auch von den *Momenten* einer Zufallsvariablen. Der Erwartungswert ist dabei das erste Moment. Das k-te Moment einer Zufallsvariable ist definiert als $E[X^k]$.

Eine wichtige Eigenschaft des Erwartungswerts zweier Zufallsvariablen X und Y ist seine *Linearität*: Es gilt $\mathrm{E}[X+Y] = E[X]+E[Y]$, und für eine Konstante d gilt außerdem $\mathrm{E}[dX] = d \cdot \mathrm{E}[X]$ und $\mathrm{E}[d + X] = d + \mathrm{E}[X]$.

Eine Wahrscheinlichkeitsverteilung ist, grob gesagt, eine Zuweisung von Wahrscheinlichkeiten zu den Werten, die eine Zufallsvariable annehmen kann; wir gehen hier also nicht von einem Zufallsexperiment aus, für dessen Ausgänge wir eine Zufallsvariable definieren, sondern wir definieren ein Zufallsexperiment so, dass eine Zufallsvariable die Eigenschaften der Verteilung hat. Grundsätzlich ist zu unterscheiden zwischen diskreten und stetigen Zufallsverteilungen: Erstere ordnen jedem der abzählbar vielen Werte, die die Zufallsvariable annehmen kann, eine Wahrscheinlichkeit zu; bei letzteren hat die Zufallsvariable überabzählbar viele mögliche Werte, und die Einzelwahrscheinlichkeit jedes dieser Werte ist 0. Daher ist hier die Wahrscheinlichkeit interessant, mit der die Zufallsvariable Werte aus einem gegebenen Intervall annimmt. Im Rahmen dieser Arbeit beschränken wir uns auf diskrete Zufallsverteilungen. Diese werden über eine Verteilungsfunktion $F(x) = \Pr[X \leq x]$ für eine Zufallsvariable X definiert; die Dichtefunktion $f(x)$ ist im diskreten Fall die Wahrscheinlichkeit $\Pr[X = x]$, dass X den Wert x annimmt; die Summe über alle möglichen Werte der Dichtefunktion ist dabei 1.

Eine Eigenschaft zweier Zufallsverteilungen ist ihre *statistische Distanz*. Um diese zu messen, gibt es einige unterschiedliche Ansätze; die in dieser Arbeit verwendete Form der statistischen Distanz wird im englischen auch als *total variation distance* bezeichnet:

Definition 2.17 (Statistische Distanz, siehe zum Beispiel [64], Anhang A)**.** *Seien D_F und D_G zwei diskrete Zufallsverteilungen mit Verteilungsfunktionen F und G mit gemeinsamem Definitionsbereich D. D_F und D_G haben eine statistische Distanz von δ, falls*

$$\max_{S \subseteq D} \left| \sum_{x \in S} F(x) - G(x) \right| = \delta$$

gilt. Zwei Zufallsvariablen X und Y, die den Verteilungsfunktionen D_F und D_G gehorchen, haben eine statistische Distanz von δ, falls D_F und D_G eine statistische Distanz von δ haben.

Wir werden im Laufe dieser Arbeit auf folgende Zufallsverteilungen zurückgreifen:

Definition 2.18. *Eine Zufallsvariable X ist* hypergeometrisch *verteilt mit Parametern $n, m, k \in \mathbb{N}$, falls für $x \in \{0, \ldots, k\}$*

$$\Pr[X = x] = \mathrm{HyG}(n, m, k)_x = \frac{\binom{m}{x} \cdot \binom{n-m}{k-x}}{\binom{n}{k}}.$$

Der Erwartungswert einer hypergeometrisch verteilten Zufallsvariablen X ist $\mathrm{E}[X] = k \cdot \frac{m}{n}$ (siehe zum Beispiel [20]). Anschaulich modelliert die hypergeometrische Verteilung, wieviele günstige Kugeln aus einer Urne mit insgesamt n Kugeln gezogen werden, wenn es $m \leq n$ günstige Kugeln in der Urne gibt und davon k Kugeln *ohne Zurücklegen* gezogen werden.

Die negative hypergeometrische Verteilung ist die der hypergeometrischen Verteilung zugehörige Laufzeitverteilung; das heißt, sie modelliert im obigen Experiment die Anzahl x der Kugeln, die gezogen werden müssen, bis man insgesamt k günstige Kugeln erhält. Ihre Dichtefunktion lässt sich leicht aus der der hypergeometrischen Verteilung konstruieren: Die Wahrscheinlichkeit, dass die x-te gezogene Kugel die k-te günstige ist, ist

$$\begin{aligned}\Pr[X = x] = &\Pr[k-1 \text{ günstige unter den ersten } x-1 \text{ Kugeln}] \\ &\cdot \Pr[x\text{-te Kugel ist günstig} \mid \text{es wurden schon } k-1 \text{ günstige gezogen}].\end{aligned}$$

Das erste Ereignis ist dabei dabei hypergeometrisch verteilt, während die Wahrscheinlichkeit des zweiten $\frac{m-k+1}{n-x+1}$ ist. Das ergibt folgende Definition der negativen hypergeometrischen Verteilung:

Definition 2.19. *Eine Zufallsvariable X ist* negativ hypergeometrisch *verteilt mit Parametern $n, m, k \in \mathbb{N}$, falls für $x \in \mathbb{N}_{\geq k}$*

$$\Pr[X = x] = \frac{\binom{m}{k-1} \cdot \binom{n-m}{x-k}}{\binom{n}{x-1}} \cdot \frac{m-k+1}{n-x+1} = \frac{m}{n} \cdot \frac{\binom{m-1}{k-1} \cdot \binom{n-m}{x-k}}{\binom{n-1}{x-1}}.$$

Der Erwartungswert einer negativ hypergeometrisch verteilten Zufallsvariablen X ist $\mathrm{E}[X] = k \cdot \frac{n+1}{m+1}$: Es gilt

$$\begin{aligned}\mathrm{E}[X] &= \sum_{x \geq 0} x \cdot \frac{m}{n} \cdot \frac{\binom{m-1}{k-1} \cdot \binom{n-m}{x-k}}{\binom{n-1}{x-1}} = k \cdot \sum_{x \geq 0} \frac{\frac{m}{k}\binom{m-1}{k-1} \cdot \binom{n-m}{x-k}}{\frac{n}{x}\binom{n-1}{x-1}} \\ &= k \cdot \frac{n+1}{m+1} \cdot \sum_{x \geq 0} \frac{m+1}{n+1} \cdot \frac{\binom{m}{k} \cdot \binom{n-m}{x-k}}{\binom{n}{x}},\end{aligned}$$

und die hintere Summe aggregiert über alle Werte der Dichtefunktion einer negativ hypergeometrisch verteilten Zufallsvariablen mit Parametern $n+1$, $m+1$ und k, ist also 1.

Wenn wir das obige Urnenexperiment dahingehend abwandeln, dass gezogene Kugeln wieder der Urne hinzugefügt werden, wir also *mit Zurücklegen* ziehen, dann ergibt sich für die Anzahl gezogener günstiger Kugeln die *Binomialverteilung*. Da in diesem Fall die Wahrscheinlichkeit, eine günstige Kugel zu ziehen, immer dieselbe ist, lässt sie sich durch eine Konstante $p \in [0,1]$ darstellen, anstatt die Anzahl n insgesamt vorhandener und die Anzahl m günstiger Kugeln anzugeben:

Definition 2.20 (Binomialverteilung)**.** *Eine Zufallsvariable X ist* binomialverteilt *mit Parametern $n \in \mathbb{N}$, $p \in [0,1]$, falls für $x \in \{0,\dots,k\}$*

$$\Pr[X = x] = \mathrm{Bi}(n,p)_x = \binom{n}{x} \cdot p^x (1-p)^{k-x}.$$

Der Erwartungswert einer binomialverteilten Zufallsvariablen X ist $\mathrm{E}[X] = np$ (siehe [20]).

Eine weitere diskrete Zufallsverteilung ist die Poissonverteilung:

Definition 2.21 (Poissonverteilung)**.** *Eine Zufallsvariable X ist* poissonverteilt *mit Parameter $\lambda \in \mathbb{R}$, falls für $x \in \mathbb{N}$*

$$\Pr[X = x] = \mathrm{Pois}(\lambda)_x = \frac{\lambda^x}{x!} \cdot e^{-\lambda}.$$

Für kleine p und große n ist die Poissonverteilung eine gute Näherung der Binomialverteilung; der Erwartungswert einer mit Parameter λ poissonverteilten Zufallsvariablen ist λ (siehe [20]).

Die geometrische Verteilung ist eine Laufzeitverteilung, die die Anzahl unabhängiger Wiederholungen eines Versuchs modelliert, die nötig sind, bis ein bestimmter, günstiger Ausgang eintritt. Der einzige nötige Parameter ist die Erfolgswahrscheinlichkeit p eines einzelnen Versuchs.

Definition 2.22. *Eine Zufallsvariable X ist* geometrisch *verteilt mit Parameter $p \in [0,1]$, falls für $x \in \mathbb{N}$*

$$\Pr[X = x] = \mathrm{Geo}(p)_x = p \cdot (1-p)^{x-1}.$$

Der Erwartungswert einer geometrisch verteilten Zufallsvariablen X ist $E[X] = \frac{1}{p}$ (siehe zum Beispiel [54]).

Bei der Analyse randomisierter Algorithmen ist es häufig nicht nötig, Wahrscheinlichkeiten exakt auszurechnen; stattdessen genügt es, obere oder untere Schranken zu verwenden. Um diese zu gewinnen, verwenden wir im Rahmen dieser Arbeit im Wesentlichen vier Techniken, nämlich die grundlegende Markow-Ungleichung, die Tschebyschow-Ungleichung, verschiedene Formen von Chernoff-Schranken und die Hoeffding-Ungleichung. Solche Ungleichungen nennen wir *Konzentrationsschranken* (englisch *Tail Bounds*), da sie Schranken für die Wahrscheinlichkeit geben, dass eine Zufallsvariable weit von ihrem Erwartungswert abweicht.

Theorem 2.1 (Markow-Ungleichung, nach [20]). *Sei X eine Zufallsvariable. Dann gilt* $\Pr[X \geq a] \leq \frac{\mathrm{E}[X]}{a}$.

Diese Ungleichung ist vor allem deswegen nützlich, weil sie auf jede beliebige Zufallsvariable angewendet werden kann, vorausgesetzt, ihr Erwartungswert ist endlich. Allerdings lässt sich durch direktes Anwenden der Markow-Ungleichung keine Schranke für die Wahrscheinlichkeit angeben, dass eine Zufallsvariable ihren Erwartungswert unterschreitet; außerdem nimmt die Wahrscheinlichkeit lediglich linear in a ab.

Ein in dieser Hinsicht besseres Ergebnis liefert die *Tschebyschow-Ungleichung*; sie gibt zudem eine Abschätzung der Wahrscheinlichkeit für große Abweichung vom Erwartungswert sowohl nach oben als auch nach unten. Um sie anzuwenden, muss allerdings die Varianz oder eine Abschätzung der Varianz bekannt sein:

Theorem 2.2 (Tschebyschow-Ungleichung, nach [20]). *Sei X eine Zufallsvariable mit endlicher Varianz. Dann gilt für alle $a > 0$*

$$\Pr[|X - \mathrm{E}[X]| \geq a] \leq \frac{\mathrm{Var}(X)}{a^2}.$$

Eine noch genauere Abschätzung liefern Chernoff-Schranken: Diese erreichen eine exponentielle Abnahme der Wahrscheinlichkeit und lassen sich, wie die Tschebyschow-Ungleichung, sowohl für Beschränkung der Abweichung vom Erwartungswert nach oben als auch nach unten einsetzen; wir verwenden verschiedene Formen von Chernoff-Schranken, um zu zeigen, dass Summen von Indikatorzufallsvariablen mit hoher Wahrscheinlichkeit nicht stark von ihrem Erwartungswert abweichen:

Theorem 2.3 (Multiplikative Chernoff-Schranken, nach [54]). *Seien $X_1, \ldots, X_n$ Indikatorzufallsvariablen für den Eintritt von n unabhängigen Ereignissen mit* $\Pr[X_i = 1] = p$ *für alle $i = 1, \ldots, n$. Sei $X := \sum_{1 \leq i \leq n} X_i$ und gelte $0 < \delta < 1$. Dann gilt:*

- $\Pr[X \geq (1+\delta)\mathrm{E}[X]] \leq e^{-\delta^2 \mathrm{E}[X]/2}$
- $\Pr[X \leq (1-\delta)\mathrm{E}[X]] \leq e^{-\delta^2 \mathrm{E}[X]/3}$
- $\Pr[|X - \mathrm{E}[X]| \geq \delta \cdot \mathrm{E}[X]] \leq 2e^{-\delta^2 \mathrm{E}[X]/3}$

Die dritte Schranke folgt dabei aufgrund der Union-Bound aus den ersten beiden. Diese Schranken machen sich zunutze, dass die Wahrscheinlichkeitsmasse in den „Enden“ (englisch *tails*) der Binomialverteilung exponentiell in der Entfernung vom Erwartungswert sinkt.

Die Hoeffding-Ungleichung ist dann nützlich, wenn eine Summe von Zufallsvariablen beschränkt werden soll, die andere Werte als 0 und 1 annehmen können.

Theorem 2.4 (Hoeffding-Ungleichung, nach [50]). *Seien $X_1, \ldots, X_n$ unabhängige gleichverteilte Zufallsvariablen mit $X_i \in [a_i, b_i]$ für alle $i = 1, \ldots n$. Sei $X := \sum_{1 \leq i \leq n} X_i$ und gelte $t > 0$. Dann gilt*

- $\Pr[X - \mathrm{E}[X] \geq t] \leq \exp\left(-\frac{2t^2}{\sum_i (b_i - a_i)^2}\right)$.

- $\Pr[\mathrm{E}[X] - X \geq t] \leq \exp\left(-\frac{2t^2}{\sum_i (b_i - a_i)^2}\right)$.

Falls $a_i = a$ und $b_i = b$ für alle $i = 1, \ldots n$, dann gilt

- $\Pr[X - \mathrm{E}[X] \geq t] \leq \exp\left(-\frac{2t^2}{n \cdot (b-a)^2}\right)$.
- $\Pr[\mathrm{E}[X] - X \geq t] \leq \exp\left(-\frac{2t^2}{n \cdot (b-a)^2}\right)$.

Aus dem Spezialfall $a = 0, b = 1$ der Hoeffding-Ungleichung ergeben sich die folgenden Abschätzungen, die auch als „additive" Chernoffschranken bekannt sind:

Theorem 2.5 (Additive Chernoff-Schranken, nach [50]). *Seien $X_1, \ldots, X_n$ Indikatorzufallsvariablen für den Eintritt von n unabhängigen Ereignissen mit $\Pr[X_i = 1] = p$ für alle $i = 1, \ldots, n$. Sei $X := \sum_{1 \leq i \leq n} X_i$ und gelte $t > 0$. Dann gilt:*

- $\Pr[X > \mathrm{E}[X] + t] \leq e^{-2t^2/n}$
- $\Pr[X < \mathrm{E}[X] - t] \leq e^{-2t^2/n}$

Ein Konzept, das an einigen Stellen dieser Arbeit auftritt, ist das sogenannte *Geburtstagsparadoxon*. Grundsätzlich sagt es aus, dass bei einer Grundmenge von n Elementen, von denen $m = \Omega(\sqrt{n})$ Elemente zufällig gleichverteilt mit Zurücklegen gezogen werden, mit konstant hoher Wahrscheinlichkeit mindestens ein Element zweimal gezogen wird.

Damit lässt sich ein Geburtstagsproblem wie folgt modellieren: Wie hoch ist die Wahrscheinlichkeit, dass in einer Gruppe von m Personen mindestens zwei davon den gleichen Geburtstag haben? Da für Geburtstage (ohne Schalttage) $n = 365$ Tage zur Verfügung stehen, reichen bei angenommener Gleichverteilung der Geburtstage $m = 23$ Personen, damit die Wahrscheinlichkeit eines doppelten Geburtstags mindestens

$$1 - \prod_{1 \leq i \leq 23} \frac{365 - i + 1}{365} > 0.5$$

ist; da dies eine deutlich geringere Anzahl an Personen ist als intuitiv angenommen, spricht man von einem Paradoxon.

Dies lässt sich auch auf häufigeres Ziehen des gleichen Elements erweitern: Wir sprechen von *k-Fach-Kollisionen*, falls ein Element der Stichprobe k mal gezogen wurde. Damit dies mit konstanter Wahrscheinlichkeit mindestens einmal passiert, muss die Samplegröße m mindestens $\Theta\left(n^{1-1/k}\right)$ sein.

2.1.3 Sonstige Abschätzungen

In diesem Abschnitt werden zwei gebräuchliche Abschätzungen erwähnt, die im Laufe der Arbeit angewendet werden. Die erste Abschätzung ergibt sich aus der Definition der Exponentialfunktion e^x und ist zum Beispiel nützlich, wenn die Wahrscheinlichkeit des Schnittes unabhäbgiger Ereignisse gleicher Wahrscheinlichkeit berechnet werden soll.

Lemma 2.1. *Für alle* $x \in \mathbb{R}\backslash\{0\}$ *gilt* $\left(1 - \frac{1}{x}\right)^x \leq e^{-1}$.

Um solche Terme in die andere Richtung abschätzen zu können, bietet sich die *Bernoulli-Ungleichung* an:

Lemma 2.2. *Für alle* $x \in \mathbb{R}_{\geq -1}$ *und* $k \in \mathbb{R}^+$ *gilt* $(1+x)^k \geq 1 + x \cdot k$.

2.2 Property-Testing in Graphen

Property-Testing ist eine recht junge Technik, die erst seit Mitte der 1990er Jahre erforscht wird. Erste Ergebnisse stammen von Rubinfeld und Sudan [68], die das Testen von Programmen auf Mitgliedschaft in Klassen von Funktionen untersuchen. Später wurde die Technik von Goldreich, Goldwasser und Ron auf das Testen von Grapheigenschaften übertragen [41].

Allgemein geht es beim Property-Testing darum, ein Entscheidungsproblem in sublinearer oder sogar konstanter Laufzeit zu lösen, nämlich ob die Eingabe eines Algorithmus eine vorher definierte Eigenschaft Π hat. Um das zu erreichen, wird Randomisierung eingesetzt: Nur ein kleiner, zufällig gewählter Teil der Eingabe wird von einem Property-Testing-Algorithmus stichprobenhaft untersucht, und die Entscheidung des Algorithmus gründet sich auf diese Stichproben. Natürlich muss das Entscheidungsproblem dafür relaxiert werden: Wenn sich zwei Eingaben nur an wenigen Stellen unterscheiden, eine der Eingaben aber die angefragte Eigenschaft Π besitzt und die andere nicht, dann ist die Wahrscheinlichkeit klein, dass mit Hilfe einer sublinearen oder konstanten Anzahl von Stichproben eine der wenigen unterschiedlichen Stellen entdeckt wird; ein Algorithmus, der seine Entscheidung auf wenige Stichproben gründet, wird demnach mit hoher Wahrscheinlichkeit für beide Eingaben die gleiche Antwort geben.

Daher wird im Property-Testing das Konzept der ϵ-*Entferntheit* eingesetzt: Zwei Eingaben a und b mit gleicher Länge n heißen ϵ-entfernt oder auch ϵ-fern voneinander, wenn mehr als ϵn Stellen der Codierung von a geändert werden müssen, um b zu erhalten; dabei gilt $0 < \epsilon < 1$. Im Gegenzug sind a und b ϵ-*nah* zueinander, wenn höchstens ϵn Stellen der Codierung von a geändert werden müssen, um b zu erhalten. Für eine Eigenschaft Π ist a ϵ-entfernt (bzw. ϵ-fern) von Π, wenn a ϵ-entfernt von allen $b \in \Pi$ ist, und ϵ-nah zu Π sonst.

Ein Property-Testing-Algorithmus muss nur dann zwischen zwei Eingaben a und b entscheiden können, wenn diese Eingaben entweder gleich sind oder ϵ-entfernt voneinander; ansonsten ist die Ausgabe des Algorithmus beliebig. Für eine Eingabe a und eine Eigenschaft Π sollte ein Property-Testing-Algorithmus also

- akzeptieren, wenn $a \in \Pi$ gilt;
- ablehnen, wenn a ϵ-entfernt von Π ist;
- eine beliebige der beiden Antworten geben, wenn $a \notin \Pi$ gilt und a ϵ-nah zu Π ist.

Als randomisierter Algorithmus hat ein Property-Testing-Algorithmus zudem eine bestimmte Fehlerwahrscheinlichkeit: Mit einer beschränkten Wahrscheinlichkeit dürfen auch

für von der angefragten Eigenschaft ϵ-entfernte Eingaben akzeptiert beziehungsweise Eingaben, die die Eigenschaft haben, abgelehnt werden. Geschieht dies beides mit einer Wahrscheinlichkeit größer 0, dann sprechen wir von einem Property-Testing-Algorithmus mit *zweiseitigem Fehler*; kann nur irrtümlich akzeptiert werden, aber nicht irrtümlich abgelehnt, dann hat der entsprechende Property-Testing-Algorithmus einen *einseitigem Fehler*; jede Eingabe, die die angefragte Eigenschaft hat, wird von einem solchen Algorithmus angenommen.

Aufgrund der Möglichkeit zur Wahrscheinlichkeitsverstärkung definieren wir, dass für keine Eingabe die Fehlerwahrscheinlichkeit eines Property-Testing-Algorithmus mehr als $\frac{1}{3}$ betragen darf.

Ein Property-Testing-Algorithmus erhält keinen direkten Zugriff auf seine Eingabe a, sondern „Orakelzugriff“ über eine Orakelfunktion f_a, mit der einzelne Elemente von a angefragt werden können; die genaue Definition dieser Funktion ist problemspezifisch. Die wichtigste Metrik für die Effizienz eines Property-Testing-Algorithmus ist seine *Anfragekomplexität*: die Anzahl der Aufrufe der Funktion f_a, die im Worst-Case für Eingaben a der Länge n vom Algorithmus durchgeführt wird. Daneben ist auch die Laufzeit für Property-Testing-Algorithmen eine wichtige Metrik.

Definition 2.23. *Ein Algorithmus $\mathcal{A}$ ist ein Property-Testing-Algorithmus für die Eigenschaft Π, falls er mit Orakelzugriff auf eine Eingabe a und einen Näheparameter $\epsilon < 1$*

- *mit Wahrscheinlichkeit mindestens $\frac{2}{3}$ akzeptiert, falls $a \in \Pi$ gilt;*
- *mit Wahrscheinlichkeit mindestens $\frac{2}{3}$ ablehnt, falls a ϵ-entfernt von Π ist.*

$\mathcal{A}$ ist Property-Testing-Algorithmus mit einseitigem Fehler für Π, falls $\mathcal{A}$ zusätzlich jede Eingabe $a \in \Pi$ mit Wahrscheinlichkeit 1 akzeptiert.

Mit dem Property-Testing eng verwandt ist der Bereich der *sublinearen Approximationsalgorithmen*. Dabei handelt es sich ebenfalls um zumeist randomisierte Algorithmen für Optimierungsprobleme. Da bei vielen Optimierungsproblemen die Größe der Lösung linear in der Eingabegröße ist, wird dabei nicht die Lösung selbst approximiert, sondern deren Wert; der bei der Approximation erlaubte Fehler ist entweder additiv oder multiplikativ oder beides. Auch hier ist eine beschränkte Fehlerwahrscheinlichkeit von zum Beispiel $\frac{1}{3}$ erlaubt, mit der der Algorithmus die Approximationsgarantie nicht einhält.

Die Beziehung zwischen Property-Testing und sublinearen Approximationsalgorithmen ist zweiseitig: In der einen Richtung können durch einen sublinearen Approximationsalgorithmus als Unterfunktion gemessene Werte einem Property-Testing-Algorithmus als Entscheidungsgrundlage dienen: Zum Beispiel könnte ein Property-Testing-Algorithmus für Zusammenhang von Graphen zunächst die Anzahl der Zusammenhangskomponenten mit einem additiven Fehler von $\frac{1}{4}\epsilon n$ approximieren und ablehnen, wenn der Wert der Approximation wesentlich größer als $\frac{1}{2}\epsilon n$ ist, ansonsten annehmen. Mit hoher Wahrscheinlichkeit würden dann Graphen mit vielen Zusammenhangskomponenten abgelehnt und Graphen mit wenigen Zusammenhangskomponenten angenommen; es bliebe dann noch der Zusammenhang zwischen der Anzahl der Zusammenhangskomponenten und der ϵ-Entferntheit von der Klasse der zusammenhängenden Graphen zu analysieren. Im

Ergebnisteil dieser Arbeit werden sublineare Approximationsalgorithmen an mehreren Stellen als Unterroutinen von Property-Testing-Algorithmen verwendet.

In der anderen Richtung kann ein Property-Testing-Algorithmus als Unterroutine eines sublinearen Approximationsalgorithmus fungieren; ein Beispiel hierfür, auf das Goldreich, Goldwasser und Ron hinweisen [41], ist ihr Property-Tester für die Eigenschaft, dass ein Graph einen Cut der Größe ρ hat: Durch Einteilen der möglichen Werte für die Größe des maximalen Cuts eines Graphen G in $\frac{1}{\epsilon}$ gleich große Intervalle und Ausführen des Property-Testing-Algorithmus für je einen Wert innerhalb jedes der Intervalle kann die Größe des maximalen Cuts von G additiv approximiert werden, wobei die Laufzeit nur um den Faktor $\frac{1}{\epsilon}$ größer ist als die Anfragekomplexität des Property-Testing-Algorithmus.[1]

Goldreich, Goldwasser und Ron haben in [41] auf die Verknüpfung zwischen Property-Testing und PAC-Lernen (siehe [70]) hingewiesen: In beiden Bereichen wird durch stichprobenhafte Anfragen von Funktionswerten einer dem Algorithmus unbekannten Funktion Wissen über diese erlangt. Dabei ist das Ziel beim Lernen der Erhalt einer Funktion, die die ursprüngliche Funktion für alle außer höchstens ϵn der möglichen Eingabewerte annähert; die also sozusagen ϵ-nah zu der unbekannten ursprünglichen Funktion ist. Beim PAC-Lernen kann dies mit hoher Wahrscheinlichkeit für Funktionen einer Klasse $\mathcal{F}$ garantiert werden, während für Funktionen außerhalb von $\mathcal{F}$ keine Gütegarantie besteht.

Gibt es für eine Klasse $\mathcal{F}$ von Funktionen einen PAC-Lernalgorithmus, der $f(n, \epsilon^{-1})$ Beispiele für eine Funktion der Größe n benötigt, und kann zusätzlich angenommen werden, dass dieser Algorithmus immer entweder eine Funktion in $\mathcal{F}$ ausgibt oder ohne Ausgabe terminiert, dann gibt es auch einen Property-Testing-Algorithmus für $\mathcal{F}$, der eine Anfragekomplexität von $\mathcal{O}(f(n, \epsilon^{-1}) + \epsilon^{-1})$ hat; Goldreich, Goldwasser und Ron zeigen dies in [41]: Ihr generischer Property-Testing-Algorithmus verwendet den Lernalgorithmus mit Näheparameter $\frac{\epsilon}{2}$ als Unterfunktion, um eine Hypothese h für f zu erhalten. Hält der Lernalgorithmus ohne Ausgabe, wird abgelehnt. Ist die Ausgabe hingegen eine Funktion $h \in \mathcal{F}$, so wird mit $\mathcal{O}(1/\epsilon)$ zusätzlichen Samples in f die Distanz von h und f mit einem additiven Fehler von $\frac{\epsilon}{4}n$ approximiert. Ist dieser Schätzwert größer als $\frac{3}{4}\epsilon n$, so wird abgelehnt, ansonsten akzeptiert.

Für $f \in \mathcal{F}$ gibt der Lernalgorithmus aufgrund seiner Gütegarantie mit hoher Wahrscheinlichkeit $h \in \mathcal{F}$ zurück, so dass h $\frac{\epsilon}{2}$-nah zu f ist. Damit ist die Schätzung der Entfernung zwischen h und f mit hoher Wahrscheinlichkeit höchstens $\frac{3}{4}\epsilon n$, und folglich akzeptiert der generische Algorithmus mit hoher Wahrscheinlichkeit.

Falls f ϵ-entfernt von $\mathcal{F}$ ist, hat der Lernalgorithmus mit hoher Wahrscheinlichkeit entweder keine Rückgabe – in diesem Fall wird richtigerweise abgelehnt –, oder es wird eine Funktion $h \in \mathcal{F}$ zurückgegeben. Da f ϵ-entfernt von $\mathcal{F}$ ist, ist f in diesem Fall aber auch ϵ-entfernt von h; der Schätzwert der Distanz zwischen f und h ist also mit hoher Wahrscheinlichkeit größer als $\frac{3}{4}\epsilon n$, und damit lehnt der generische Algorithmus mit hoher Wahrscheinlichkeit ab. Der Algorithmus ist demnach ein Property-Testing-Algorithmus für $\mathcal{F}$.

[1] Fischer und Newmann [38] konnten außerdem zeigen, dass sich für jede in dicht besetzten Graphen in konstanter Zeit testbare Eigenschaft auch die Entfernung eines Graphen zu dieser Eigenschaft in konstanter Zeit approximieren lässt.

Andere Verwandtschaften zu weiteren Forschungsbereichen sind spezifisch für einzelne Property-Testing-Modelle. An dieser Stelle sei die Verwandtschaft zwischen verteilten Algorithmen, zum Beispiel in Ad-Hoc- oder Sensornetzen, und Property-Testing in dünn besetzten Graphen erwähnt, die durch die Art und Weise entsteht, wie in diesem Property-Testing-Modell durch lokale Explorationen Information gewonnen wird. Insbesondere durch möglicherweise unterschiedliche Sende- und Empfangsstärken der einzelnen Teilnehmer von Ad-Hoc- oder Sensornetzen lassen sich diese zudem gut durch gerichtete Graphen modellieren. Sowohl das Modell dünn besetzter Graphen als auch die wesentlichen Explorationstechniken werden sogleich ausführlich vorgestellt.

Property-Testing von Grapheigenschaften wurde zunächst für ungerichtete Graphen erforscht. Dabei bildeten sich zwei grundsätzliche Modelle heraus, Property-Testing in *dicht besetzten* [41] und in *dünn besetzten Graphen* [43]. Die beiden Modelle unterscheiden sich in Bezug auf die üblichen algorithmischen Herangehensweisen deutlich, was auf eine unterschiedliche Art des Zugriffs zurückzuführen ist: Für dicht besetzte Graphen wird angenommen, dass sie als Adjazenzmatrix gespeichert sind, und daher liest eine Orakelfunktion f_G einen Adjazenzmatrixeintrag aus, das heißt ihre Signatur ist $f_G : V(G)^2 \to \{0, 1\}$; der zurückgegebene Wert ist 1 genau dann, wenn die angegebenen Knoten u und v verbunden sind.

Für dünn besetzte Graphen wird davon ausgegangen, dass sie in Adjazenzlistendarstellung vorliegen; dabei ist die Größe jeder Adajazenzliste durch eine Konstante $D \in \mathbb{N}$ beschränkt. Das bedeutet, dass das bei dicht besetzten Graphen verwendete Zugriffsmodell hier unnatürlich ist. Stattdessen liefert hier eine Zugriffsfunktion $f_G : V(G) \times \{1, \ldots, D\} \to V(G) \cup \{\#\}$ für einen gegebenen Knoten v und einen Adjazenzlistenplatz i den Eintrag des i-ten Adjazenzlistenplatzes von v zurück; dies ist entweder der dort gespeicherte Knoten, der also über eine Kante zu v adjazent ist, oder der Platzhalter $\#$, falls v weniger als i inzidente Kanten hat.

Die genaue Definition von ϵ-Entferntheit hängt natürlich auch von der Wahl der Graphdarstellung ab: Im Adjazenzlistenmodell mit durch D beschränkte Länge der Listen nennen wir Graphen G und H mit je n Knoten ϵ-entfernt voneinander, wenn mehr als ϵDn Adjazenzlisteneinträge von G geändert werden müssen, um H zu erhalten; dies ist ein ϵ-Bruchteil der maximalen Größe der Graphdarstellung eines Graphen mit n Knoten. Da in ungerichteten Graphen jede Kante in zwei Adjazenzlisten gespeichert ist, entspricht das dem Einfügen beziehungsweise Entfernen von $\epsilon Dn/2$ Kanten.

Im Adjazenzmatrixmodell ist die Darstellungsgröße eines Graphen mit n Knoten n^2; das bedeutet, dass Graphen G und H mit je n Knoten ϵ-entfernt voneinander sind, wenn mehr als ϵn^2 Adjazenzlisteneinträge von G geändert werden müssen, um H zu erhalten. Wiederum werden in der Adjazenzmatrix eines ungerichteten Graphen die Kanten doppelt gespeichert, und damit entspricht dies dem Hinzufügen beziehungsweise Entfernen von $\epsilon n^2/2$ Kanten.

Es sei noch kurz ein drittes Modell erwähnt, das in gewissem Sinne zwischen den beiden obigen Modellen steht: das Modell „allgemeiner Graphen“ (englisch *general graphs*), in dem Graphen in Adjazenzlisten gespeichert werden, aber keine Schranke für den Maximalgrad der Knoten existiert (siehe [62]). ϵ-Entferntheit wird in diesem Modell über die tatsächliche Beschreibungsgröße eines Graphen gemessen, nicht über die maximal durch

das Modell erreichbare wie bei den beiden anderen Modellen: Ein Graph mit m Kanten ist hier ϵ-entfernt von einer Eigenschaft, wenn mehr als ϵm Adjazenzlisteneinträge verändert werden müssen, um die Eigenschaft herzustellen. Das bedeutet auch, dass die Relation der ϵ-Entferntheit zwischen zwei Graphen nicht mehr symmetrisch ist, da diese Graphen unterschiedliche Kantenzahlen haben können. Ansonsten entspricht das Modell allgemeiner Graphen dem oben eingeführten Modell dünn besetzter Graphen.

Aus den besprochenen Modellierungen ergeben sich unterschiedliche algorithmische Herangehensweisen in den Modellen dicht und dünn besetzter Graphen. Die wesentliche Herangehensweise in dicht besetzten Graphen ist das zufällig gleichverteilte Sampeln einer Anzahl von Knoten; es wird dann der induzierte Subgraph der gesampelten Knoten angefragt, indem für alle Paare u, v von gesampelten Knoten $f_G(u, v)$ aufgerufen wird, und die Entscheidung des Algorithmus wird abhängig von der Struktur dieses Subgraphen getroffen.

Alon et al. [6] konnten zeigen, dass sich tatsächlich alle in diesem Modell testbaren Grapheigenschaften mit dieser Methode testen lassen. Die Idee ist, dass eine Eigenschaft genau dann testbar ist, wenn alle Graphen G, die die Eigenschaft haben, eine durch die Eigenschaft bestimmte globale Struktur haben, und diese Struktur lässt sich für jede Knotenzahl n von G beschreiben durch eine Menge von konstant vielen „Partitionierungsmustern“, deren Anzahl im Prinzip nur von ϵ abhängt. Wenn eins der Partitionierungsmuster auf G zumindest ungefähr passt, dann hat G die Eigenschaft oder ist zumindest ϵ-nah zu ihr; wenn G weit entfernt davon ist, dass ein beliebiges der Partitionierungsmuster passt, dann ist G auch ϵ-fern von der Eigenschaft.

Die Partitionierungsmuster definieren sich über Eigenschaften *regulärer Partitionierungen*: Dies sind Partitionierungen der Knotenmenge in k gleich große Partitionen $V_1, \ldots V_k$, wobei fast alle Paare U, V von Partitionen regulär sind, das heißt grob gesagt: für alle hinreichend großen Teilmengen von $U' \subseteq U$ und $V' \subseteq V$ entspricht die Kantendichte zwischen U' und V' ungefähr der zwischen U und V. Die Partitionierungsmuster geben nun sowohl k vor als auch die Kantendichten zwischen den einzelnen Partitionen als auch eine Menge $\bar{R}$ von Paaren von Partitionen, die nicht regulär sein müssen und für die es auch keine Vorgaben für die Kantendichte gibt.

Damit ein Partitionierungsmuster auf einen Graphen G passt, muss es eine Partitionierung von G in k gleich große reguläre Partitionen geben (beziehungsweise Partitionen, deren Größe höchstens um eins differiert), so dass man die Partitionen derart durchnumerieren kann, dass die durch das Partitionierungsmuster geforderten Eigenschaften bezüglich der Kantendichten gelten, bis auf die Paare von Partitionen in $\bar{R}$.

Es lässt sich zeigen, dass für eine geeignete, nur von $\frac{1}{\epsilon}$ abhängige Anzahl gesampelter Knoten eines Graphen G der durch diese Knoten induzierte Subgraph mit hoher Wahrscheinlichkeit ungefähr die gleichen regulären Partitionierungen hat wie der Ausgangsgraph. Das bedeutet, dass sich mit Hilfe des Subgraphen prüfen lässt, ob eins der durch die Knotenzahl, durch ϵ und durch die Eigenschaft bestimmten regulären Partitionierungsmuster annähernd auf G passt, oder ob G ϵ-entfernt von jedem der Muster ist. Durch dieses Vorgehen ergibt sich ein Property-Testing-Algorithmus für die Ausgangseigenschaft mit einer Anfragekomplexität, die nur von $\frac{1}{\epsilon}$ abhängt.

2.2.1 Techniken für Property-Testing in dünn besetzten Graphen

Interessanter im Kontext dieser Arbeit ist das zweite wesentliche Graphmodell im Property-Testing, das der gradbeschränkten dünn besetzten Graphen, die in Form von Adjazenzlisten vorliegen. Viele der in diesem Modell verwendeten Techniken lassen sich grundsätzlich auch in gradbeschränkten gerichteten Graphen einsetzen, und diese Techniken sollen hier deshalb relativ detailliert anhand einiger ausgewählter Ergebnisse vorgestellt werden. Der Fokus in diesem und in den folgenden Abschnitten liegt dabei auf den verwendeten Techniken: Daher werden wir grundlegende Ergebnisse in der Regel mit einem höheren Detailgrad besprechen als darauf aufbauende – bei letzteren werden wir uns mehr auf die dort neu eingeführten Konzepte konzentrieren und weniger auf die mit den Vorgängerarbeiten gemeinsamen technischen Grundlagen.

Grundsätzlich verspricht die im Adjazenzmatrixmodell gebräuchliche Technik, den induzierten Subgraphen einer Menge von gesampelten Knoten als Grundlage für die Entscheidungen eines Algorithmus zu verwenden, in gradbeschränkten Graphen keinen großen Informationsgewinn: Werden in einem gradbeschränkten Graphen mit n Knoten $o(\sqrt{n})$ Knoten gesampelt, so ist die Wahrscheinlichkeit, dass der induzierte Subgraph dieser Knoten überhaupt eine Kante enthält, gering (siehe [65]); und selbst bei größeren Samplemengen ist die Anzahl der im induzierten Subgraphen enthaltenen Kanten relativ klein. Deswegen ist diese Technik in dünn besetzten Graphen nicht gebräuchlich. Stattdessen werden zufällig Knoten gesampelt und von ihnen ausgehend durch lokale Explorationen Informationen gesammelt.

Dabei stellt das Sampeln von Knoten eines Graphen mit n Knoten im Prinzip nichts anderes dar als das zufällige Wählen einiger Knotennummern aus $\{1, \ldots, n\}$, und die sinnvollste Verteilung dafür ist die Gleichverteilung: Für andere Verteilungen gibt es immer Graphen und Knotennummerierungen, für die das Ergebnis des Algorithmus im Erwartungswert schlechter ist als für die Gleichverteilung – solche Knotennummerierungen lassen sich leicht erstellen, indem diejenigen Knoten, deren Erkundung viel relevante Information verspricht, mit Nummern versehen werden, deren Sampelwahrscheinlichkeit unter $\frac{1}{n}$ liegt. Dementsprechend stehen die Explorationsmethoden im Mittelpunkt dieses Kapitels, die Algorithmen nutzen, um ausgehend von den gesampelten Knoten Wissen über den Graphen zu erlangen. Es gibt zwei grundlegende Explorationstechniken: Breitensuche und damit verwandte Techniken und Random Walks.

Breitensuche und verwandte lokale Explorationstechniken

In dünn besetzten Graphen sind Breitensuchtechniken eine sehr natürliche Form der Exploration: Durch die Speicherung in Adjazenzlisten sind maximal D Anfragen nötig, um alle Nachbarn eines Knotens zu ermitteln, und dieser Prozess kann für die soeben neu explorierten Knoten fortgesetzt werden, bis eine festgelegte Anzahl von Knoten erkundet oder eine bestimmte Entfernung zum Startknoten erreicht ist.

Beispielhaft für viele Grapheigenschaften, die mit Hilfe von Breitensuchtechniken getestet werden können, sei der Property-Tester für Zusammenhang skizziert, den Goldreich und Ron in der Arbeit vorstellen, in der sie auch das Modell gradbeschränkter Graphen

TesteZusammenhang(G, ϵ)
  Sampele zufällig gleichverteilt $\Theta\left(\frac{1}{\epsilon}\right)$ Knoten
  Foreach gesampelter Knoten v **do**
    Führe eine Breitensuche von v ausgehend aus, bis die komplette Zusammenhangskomponente von v entdeckt ist oder mehr als $m = \Theta\left(\frac{1}{\epsilon}\right)$ Knoten erreicht wurden.
    If die Breitensuche exploriert die Zusammenhangskomponente von v vollständig **then return false**
  return true

Algorithmus 2.1: TesteZusammenhang*; Nach [43].*

ins Property-Testing einführen [43]; dieser arbeitet wir folgt: Zunächst werden $\Theta(1/\epsilon)$ Knoten zufällig gleichverteilt gesampelt, dann werden von jedem dieser Knoten aus die nächsten $\Theta(1/\epsilon)$ Knoten via Breitensuche erkundet. Falls bei einer der Breitensuchen eine Zusammenhangskomponente komplett exploriert wurde, lehnt der Algorithmus ab, ansonsten nimmt er an (siehe Algorithmus 2.1).

Der Algorithmus lehnt nur ab, wenn er eine isolierte Zusammenhangskomponente identifiziert; das heißt, ein Graph, der keine isolierten Zusammenhangskomponenten hat, also zusammenhängend ist, wird mit Wahrscheinlichkeit 1 angenommen – es handelt sich also um einen Property-Testing-Algorithmus mit einseitigem Fehler. Es bleibt zu zeigen, dass der Algorithmus mit hoher Wahrscheinlichkeit ablehnt, wenn der Eingabegraph G ϵ-fern von zusammenhängend ist. Dafür sind zwei Dinge zu zeigen: Erstens, dass G in diesem Fall viele kleine Zusammenhangskomponenten hat, die durch eine der Breitensuchen komplett exploriert werden können; zweitens, dass dann einer der im ersten Schritt des Algorithmus gesampelten Knoten mit hoher Wahrscheinlichkeit in einer dieser kleinen Zusammenhangskomponenten liegt.

Letzteres ist einfach zu zeigen: Wenn es $\Theta(\epsilon Dn)$ kleine Zusammenhangskomponenten gibt, dann gibt es auch $\Omega(\epsilon Dn)$ Knoten in diesen. Für jeden gesampelten Knoten ist die Wahrscheinlichkeit, dass er zu einer der kleinen Zusammenhangskomponenten gehört, also $\Omega(\epsilon D)$, und die Wahrscheinlichkeit, dass dies für keinen der gesampelten Knoten gilt, ist $(1 - \Omega(\epsilon D))^{\Theta(1/\epsilon)} = \Omega(1)$. Für geeignete Konstanten für die Größe des Samples und der explorierten Bereiche wird also eine der kleinen Zusammenhangskomponenten entdeckt und vollständig exploriert.

Es bleibt zu zeigen, dass G viele Zusammenhangskomponenten der Größe $\mathcal{O}(1/\epsilon)$ enthält, falls G ϵ-fern von zusammenhängend ist. Die Grundidee ist, dass mit $\epsilon Dn - 1$ Kanten ϵDn Zusammenhangskomponenten in einem Kreis verbunden werden können, und dass daher ein von Zusammenhang ϵ-entfernter Graph $\Omega(\epsilon Dn)$ Zusammenhangskomponenten haben muss[2]. Das Problem ist, dass dazu in jeder der Zusammenhangskomponenten zwei Knoten gefunden werden müssen, die weniger als D inzidente Kanten haben – nur an die-

[2] Goldreich und Ron definieren ϵ-Entferntheit im Gegensatz zu dieser Arbeit über mehr als ϵDn nötige Kantenoperationen, um Isomorphie zu einem Graphen mit der Eigenschaft herzustellen.

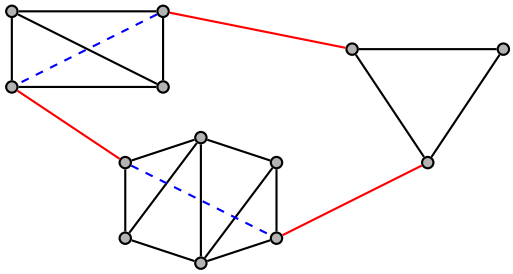

Abbildung 2.3: *Ein Graph wird durch Einfügen eines Kreises über seine Zusammenhangskomponenten (rote Kanten) verbunden; um die Gradschranke mit $D = 3$ einzuhalten, müssen dafür Kanten entfernt werden (blau, gestrichelt).*

se dürfen wegen der Gradbeschränkung neue Kanten angefügt werden. Allerdings haben alle Zusammenhangskomponenten, in denen höchstens ein Kantenplatz frei ist, mindestens D Knoten und eine relativ hohe Kantendichte; man kann zeigen, dass man unter diesen Bedingungen eine Kante entfernen kann, ohne dass die Komponente ihren Zusammenhang verliert (siehe Abbildung 2.3). Also kann jeder Graph G mit höchstens $\epsilon Dn/2$ Zusammenhangskomponenten durch höchstens ϵDn Kantenmodifikationen in einen zusammenhängenden Graphen überführt werden: Es wird pro Zusammenhangskomponente maximal eine Kante entfernt, um „Platz zu schaffen“; danach wird ein Kreis über alle Zusammenhangskomponenten eingefügt. In einem ϵ-entfernten Graphen muss es also mehr als $\epsilon Dn/2$ Zusammenhangskomponenten geben.

Der letzte Schritt im Beweis ist die Anwendung eines ebenfalls sehr typischen Zählarguments, das auch in den Ergebnissen dieser Arbeit wiederholt Anwendung findet: Es kann höchstens $\frac{n}{k}$ Zusammenhangskomponenten mit jeweils mindestens k Knoten geben; für $k := 4/\epsilon D$ folgt, dass mindestens $\epsilon Dn/4$ der $\epsilon Dn/2$ Zusammenhangskomponenten eines ϵ-entfernten Graphen weniger als $4/\epsilon D$ Knoten haben. Damit ist gezeigt, dass es viele kleine Zusammenhangskomponenten in einem ϵ-entfernten Graphen gibt, die per Breitensuche komplett exploriert werden können, und zusammen mit der obigen Argumentation ergibt sich, dass der skizzierte Algorithmus ein Property-Testing-Algorithmus für Zusammenhang in gradbeschränkten Graphen ist, der eine Anfragekomplexität von $\mathcal{O}(1/\epsilon^2)$ hat.

Das soeben skizzierte Vorgehen ist für Property-Testing in gradbeschränkten Graphen grundlegend, eine große Anzahl von Eigenschaften kann so oder ähnlich getestet werden: Sampele eine konstante Anzahl Knoten, exploriere via Breitensuche konstant große Bereiche um sie herum und lehne die Eingabe ab, wenn ein Zeuge gegen die Eigenschaft gefunden wurde; ansonsten akzeptiere.

Goldreich und Ron geben für den Algorithmus für Zusammenhang allerdings eine interessante Verfeinerungstechnik an, mit der sich die Anfragekomplexität weiter verringern lässt; die Analyse des einfachen Algorithmus ist nämlich nicht präzise: Falls zum Beispiel alle $\epsilon Dn/4$ kleinen Zusammenhangskomponenten eine Größe von 1 haben, dann müssen die Breitensuchen im Algorithmus nur eine einzige Kante anfragen, damit mit hoher Wahrscheinlichkeit eine kleine Zusammenhangskomponente identifiziert werden

TesteZusammenhangVerbessert(G, ϵ)
  For $i \leftarrow 1$ **to** $\lceil \log \frac{4}{\epsilon D} \rceil$ **do**
    Sampele zufällig gleichverteilt $\Theta\left(\frac{\log(4/\epsilon D)}{2^i \epsilon D}\right)$ Knoten
    Foreach gesampelter Knoten v **do**
      Führe eine Breitensuche von v ausgehend aus, bis die komplette Zusammenhangskomponente von v entdeckt ist oder 2^i Knoten erreicht wurden
      If die Breitensuche exploriert die Zusammenhangskomponente von v vollständig **then return false**
  return true

Algorithmus 2.2: TesteZusammenhangVerbessert*; Nach [43].*

kann; andererseits wird, falls alle kleinen Zusammenhangskomponenten eine Größe von nahezu $4/\epsilon D$ haben, die Wahrscheinlichkeit, einen ihrer Knoten zu sampeln, in der obigen Analyse stark unterschätzt.

Nehmen wir verallgemeinernd an, dass es mindestens $\frac{\epsilon D n}{4 \cdot \lceil \log(4/\epsilon D) \rceil}$ Zusammenhangskomponenten gibt, deren Größe zwischen 2^{i-1} und $2^i - 1$ liegt. Dann wird bei einer Samplegröße von $\Theta\left(\frac{\log(4/\epsilon D)}{2^i \epsilon D}\right)$ mit hoher Wahrscheinlichkeit einer dieser Knoten entdeckt, und eine Breitensuche muss maximal 2^i Knoten besuchen, um die Größe der Zusammenhangskomponente dieses Knotens zu verifizieren.

Es gibt $\lceil \log(4/\epsilon D) \rceil$ solche Intervalle $[2^{i-1}, 2^i - 1]$, in denen die Knotenzahl kleiner Zusammenhangskomponenten liegen kann; nach obiger Analyse existieren außerdem mindestens $\frac{\epsilon D n}{4}$ Komponenten, die jeweils weniger als $4/\epsilon D$ Knoten haben. Also gibt es mindestens ein Intervall, in dem die Knotenzahl von mindestens $\frac{\epsilon D n}{4 \cdot \lceil \log(4/\epsilon D) \rceil}$ der Komponenten enthalten ist; die diesem Intervall entsprechende Sampleprozedur findet daher mit hoher Wahrscheinlichkeit einen Knoten einer jener Komponenten.

Da das korrekte Intervall dem Algorithmus nicht bekannt ist, müssen alle $\mathcal{O}(\log 1/\epsilon)$ Intervalle durchprobiert werden: Für das i-te Intervall wird dabei ein Knotensample der Größe $\Theta(2^i \cdot \frac{\log 1/\epsilon}{\epsilon})$ gezogen und eine Breitensuche über maximal 2^i Knoten durchgeführt; wenn der Algorithmus dabei eine kleine Zusammenhangskomponente identifiziert, wird abgelehnt, sonst akzeptiert (siehe Algorithmus 2.2). Nach der obigen Argumentation findet der Algorithmus mit hoher Wahrscheinlichkeit einen Zeugen, falls der Eingabegraph ϵ-fern von zusammenhängend ist; ist er hingegen zusammenhängend, wird wie beim ersten Algorithmus immer akzeptiert. Die Anfragekomplexität von TesteZusammenhangVerbessert ist aber nur $\mathcal{O}\left(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon}\right)$ und damit deutlich besser als die von TesteZusammenhang.

Eine Möglichkeit Einfluss auf eine lokale Exploration zu nehmen ist, das Vorgehen von Zufallswürfen abhängig zu machen. Die beiden im Folgenden skizzierten Ergebnisse sind aus dem Bereich der sublinearen Approximationsalgorithmen, aber sowohl die Explorationstechniken als auch die Algorithmen sind, wie oben ausgeführt, auch für den Bereich

SchätzeAnzahlZusammenhangskomponenten(G, ϵ)
  Sampele zufällig gleichverteilt $\Theta\left(\frac{1}{\epsilon^2}\right)$ Knoten;
  Foreach gesampelte Knoten v_i **do**
    $k_i \leftarrow \deg(v_i)$
    Erkunde per Breitensuche die ersten $\deg(v_i)$ Kanten von v_i aus
    Do
      If Zusammenhangskomponente von v_i ist komplett erforscht
        then $\hat{c} \leftarrow \hat{c} + \frac{n}{s} \cdot 2^{\lceil \log_2 m_v / \deg(v) \rceil} \cdot \frac{1}{2} \cdot \frac{\deg(v)}{m_v}$
      Führe die Breitensuche weiter aus, bis k_i neue Kanten entdeckt sind
      $k_i \leftarrow 2 \cdot k_i$
    while Münzwurf mit W'keit $\frac{1}{2}$ ergibt „Kopf"
  return $\hat{c}$

Algorithmus 2.3: SchätzeAnzahlZusammenhangskomponenten*; Nach [23].*

des Property-Testing interessant.

Die erste Idee ist, die Tiefe, bis zu der einer Breitensuche durchgeführt wird, von Zufallswürfen abhängig zu machen. Chazelle et al. [23] setzen dies ein, um die Anzahl c der Zusammenhangskomponenten eines dünn besetzten Graphen $G = (V, E)$ mit n Knoten mit einem additiven Fehler von ϵn zu approximieren[3]. Die grundsätzliche Beobachtung, die ihr Algorithmus sich zunutze macht, ist, dass c die Summe der anteiligen Kantenbeiträge der Knoten zu ihrer jeweiligen Zusammenhangskomponente ist, also

$$\sum_{v \in V} \frac{\frac{1}{2}\deg(v)}{m_v} = \sum_{\text{ZHK } U \subseteq V} \frac{1}{m_U} \sum_{v \in U} \frac{1}{2} \deg(v) = \sum_{\text{ZHK } U \subseteq V} \frac{m_U}{m_U} = c,$$

wobei m_U die Anzahl der Kanten in der Zusammenhangskomponente U ist und $m_v = m_U$ für $v \in U$.

Da es höchstens $\frac{\epsilon}{2}n$ Zusammenhangskomponenten mit mehr als $\frac{2}{\epsilon}$ Knoten gibt, können diese ignoriert werden: dies verändert den Schätzwert um höchstens $\frac{\epsilon}{2}n$. Allgemein hat die Anzahl größerer Zusammenhangskomponenten weniger Einfluss auf c als die Anzahl kleiner Zusammenhangskomponenten, und das macht sich der Algorithmus von Chazelle et al. zunutze, indem er größere Zusammenhangskomponenten mit kleinerer Wahrscheinlichkeit komplett exploriert als kleinere.

Der Algorithmus sampelt grundsätzlich wieder Knoten von G und führt von ihnen ausgehend Breitensuchen aus. Für jeden der $s = \mathcal{O}(1/\epsilon^2)$ gesampelten Knoten wird die Anzahl der maximal von der Breitensuche erkundeten Kanten jetzt aber zufällig bestimmt: Die Wahrscheinlichkeit, dass für einen Knoten v die gesamte Komponente erforscht wird, ist dabei $2^{-\lceil \log_2 m_v / \deg(v) \rceil}$. Da m_v dem Algorithmus beim Zufallswurf unbekannt ist, wird das dadurch erreicht, dass im ersten Schritt die $\deg(v)$ inzidenten

[3] Der Algorithmus von Chazelle et al. funktioniert sogar für Graphen mit beschränktem Durchschnittsgrad, aber unbeschränktem Grad der Einzelknoten; die hier vorgestellte Variante ist dementsprechend leicht vereinfacht.

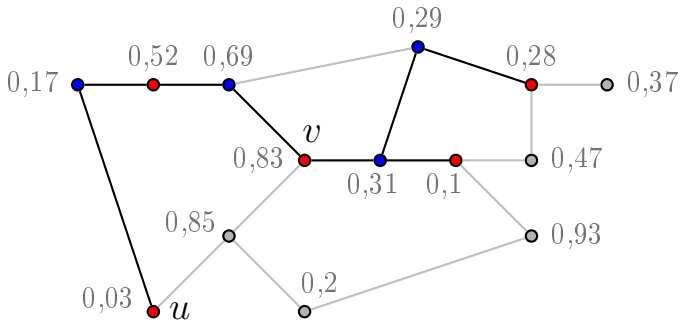

Abbildung 2.4: *Graph G mit den Knoten zugeordneten Zufallswerten $r(\cdot)$ und von* IndependentSetOrakel(G,v) *angefragten Knoten, die im Independent Set enthalten (rot) bzw. nicht enthalten sind (blau). Wäre $r(u) = 0{,}2$, dann wäre v nicht im Independent Set.*

Kanten von v gelesen werden und danach sukzessive durch Münzwürfe bestimmt wird, ob die Zahl gelesener Kanten verdoppelt werden soll (siehe Algorithmus 2.3).

Nur wenn die Zusammenhangskomponente von v komplett exploriert wurde, wird ein Schätzwert $\hat{c}$ erhöht, und zwar um $\beta_i := \frac{n}{s} \cdot 2^{\lceil \log_2 m_v / \deg(v) \rceil} \cdot \frac{1}{2} \cdot \frac{\deg(v)}{m_v}$. Der Faktor $2^{\lceil \log_2 m_v / \deg(v) \rceil}$ hebt dabei die geringere Wahrscheinlichkeit des Erkundens auf, so dass für den Erwartungswert $\mathrm{E}[\beta_i] = \frac{n}{2s} \cdot \frac{\deg(v_i)}{m_{v_i}}$ gilt, falls der i-te gesampelte Knoten v_i in einer Zusammenhangskomponente der Größe höchstens $\frac{2}{\epsilon}$ liegt. Da sich außerdem zeigen lässt, dass die Varianz von $\hat{c}$ relativ klein ist, folgt aus der Tschebyschow-Ungleichung, dass $\hat{c}$ mit hoher Wahrscheinlichkeit um höchstens $\frac{\epsilon}{2}n$ von $E[\hat{c}] = c$ abweicht, falls G keine Zusammenhangskomponenten der Größe mindestens $\frac{2}{\epsilon}$ enthält. Gibt es hingegen maximal $\frac{\epsilon}{2}n$ solcher Komponenten, tragen diese, falls Knoten aus ihnen gesampelt wurden, jeweils mit 0 zur Summe $\hat{c}$ bei: $\hat{c}$ unterschätzt c um höchstens weitere $\frac{\epsilon}{2}n$. Insgesamt gilt also mit hoher Wahrscheinlichkeit $c - \epsilon n \leq \hat{c} \leq c + \frac{1}{2}\epsilon n$.

Der Nutzen des Verfahrens zeigt sich bei der Analyse der Laufzeit des Algorithmus: Die erwartete pro Breitensuche benötigte Laufzeit ist logarithmisch in $\frac{1}{\epsilon}$, bei kompletter Exploration aller Komponenten mit höchstens $\frac{2}{\epsilon}$ Knoten wäre sie linear. Das führt zu einer Gesamtlaufzeit von $\mathcal{O}(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$.

Eine weitere Technik, die Zufallswürfe nutzt, um den Bereich einer lokalen Exploration zu beeinflussen, geben Nguyen und Onak an [57]. Ihr Ergebnis ist eine der wichtigsten Entwicklungen der letzten Jahre in Bezug auf lokale Exploration in gradbeschränkten Graphen und hat, obgleich zunächst für sublineare Approximationsalgorithmen angewendet, wesentliche Auswirkungen auf Property-Testing in gradbeschränkten Graphen; wir werden dies weiter unten sehen. Zentrales Element ihrer Technik ist ein „Orakel“, das zu angefragten Knoten eines Graphen zurückgibt, ob diese zu einem inklusionsmaximalen Independent Set gehören. Die Anfragen werden konsistent beantwortet: Würden nacheinander alle Knoten angefragt, dann würden die vom Orakel als solche ausgewiesenen Knoten tatsächlich ein inklusionsmaximales Independent Set bilden.

Im Prinzip simuliert das Orakel dabei den folgenden Greedy-Algorithmus: Betrachte

```
IndependentSetOrakel(G, v)
  If r(v) noch nicht gesetzt then wähle r(v) zuf. gleichverteilt aus [0,1]
  Foreach Nachbar u von v
    If r(u) noch nicht gesetzt then wähle r(v) zuf. gleichverteilt aus [0,1]
    If r(u) < r(v) then
      If IndependentSetOrakel(G, u) = true then return false
  return true
```

Algorithmus 2.4: IndependentSetOrakel*; Nach [57].*

die Knoten in einer beliebigen Reihenfolge; falls noch kein Nachbar des aktuellen Knotens als zugehörig zum Independet Set markiert wurde, markiere ihn als zugehörig.

Das Problem beim lokalen Simulieren dieses Algorithmus ist, die jeweilige Knotenreihenfolge einzuhalten. Dazu wird jedem Knoten v, der das erste Mal besucht wird, ein Zufallswert $r(v)$ zwischen 0 und 1 zugewiesen und gespeichert; die Reihenfolge, in der der Greedy-Algorithmus die Knoten besucht, wird durch die aufsteigende Reihenfolge der Zufallswerte $r(\cdot)$ abgebildet: Knoten mit niedrigeren Werten werden vom Greedy-Algorithmus vor Knoten mit höheren Werten besucht. Damit ist die Zugehörigkeit eines Knotens zum Independent Set ausschließlich von der Zugehörigkeit der Nachbarknoten mit niedrigeren Zufallswerten abhängig (siehe Abbildung 2.4); bei der lokalen Exploration müssen also nur diese Nachbarn berücksichtigt werden, die Nachbarknoten mit höherem Zufallswert gelten als nicht erreichbar.

Die Entscheidung, ob ein angefragter Knoten im Independent Set enthalten ist, erfolgt nun dadurch, dass alle seine nach obigem Prinzip erreichbaren Nachbarknoten selbst angefragt werden, was rekursiv fortgesetzt wird, bis kein weiterer Knoten mehr explorierbar ist; liefert das Orakel für alle Nachbarknoten *false* zurück, gehören diese also nicht zum Independent Set, so wird *true* zurückgeliefert, sonst *false* (siehe Algorithmus 2.4). Im Prinzip entspricht dies einer Tiefensuche mit der oben angegebenen Besonderheit bezüglich der Einschränkung explorierbarer Kanten.

Für die Analyse sei $Q(x)$ die erwartete Anzahl Knoten, die besucht werden, wenn ein Knoten v mit $r(v) = x$ angefragt wird; wir gehen dabei davon aus, dass sowohl v als auch jeder in der Rekursion besuchte Knoten einen Grad von D hat – ansonsten ist die erwartete Anzahl besuchter Knoten niedriger. Der Ansatz von Nguyen und Onak ist, das Intervall $[0,1]$ in $2D$ gleich große Abschnitte aufzuteilen und das Problem für die Analyse dadurch zu diskretisieren, dass für jeden Zufallswert $r(u)$ auf die nächstgrößere Teilintervallgrenze aufgerundet wird.

Sei v ein Knoten mit $r(v) \in [\frac{i-1}{2D}, \ldots, \frac{i}{2D})$, d.h. $r(v)$ ist im i-ten der Teilintervalle enthalten. Jeder der D zu v adjazenten Knoten liegt mit Wahrscheinlichkeit $\frac{1}{2D}$ im j-ten Teilintervall von $[0,1]$, und die erwartete Anzahl der Knoten, die in Folge einer Anfrage an diesen Knoten besucht werden, ist dann höchstens $Q(\frac{j}{2D})$. Das heißt, die erwartete Anzahl durch Anfrage an v ausgelöster Anfragen ist aufgrund der Linearität des Erwartungswerts höchstens

$$Q(r(v)) \leq Q\left(\frac{i}{2D}\right) \leq D + D \cdot \sum_{1 \leq j \leq i} \frac{1}{2D} \cdot Q\left(\frac{j}{2D}\right) \leq D + \frac{1}{2} \sum_{1 \leq j \leq i} Q\left(\frac{j}{2D}\right) :$$

D Anfragen an die Nachbarn von v werden bereits durch die Anfrage an v ausgelöst, und für die Nachbarn können nur dann weitere Anfragen erfolgen, wenn ihre Zufallswerte in den ersten i Intervallen liegen. Durch Auflösen nach $Q(\frac{i}{2D})$ ergibt sich $Q(\frac{i}{2D}) \leq 2D + \sum_{1 \leq j < i} Q(\frac{j}{2D})$, und mit vollständiger Induktion lässt sich damit $Q(\frac{i}{2D}) \leq 2D \cdot (2^i - 1)$ zeigen. Es folgt, dass jede Anfrage im Erwartungswert höchstens $2D \cdot (2^{2D} - 1) = 2^{\mathcal{O}(D)}$ Anfragen auslöst.

Yoshida et al. geben eine genauere Analyse an [73] und zeigen, dass die erwartete Anzahl von Anfragen, die durch eine Anfrage an das Orakel ausgelöst werden, höchstens $1 + \frac{D}{2}$ ist. Sie berücksichtigen dazu zwei Dinge, die in Nguyens und Onaks oben vorgestellter Analyse unberücksichtigt geblieben waren: Erstens reicht in IndependentSet-Orakel ein einziger gefundener Nachbarknoten, der im Independent Set ist, um dies für den aktuell betrachteten Knoten auszuschließen; ist ein solcher gefunden, müssen die weiteren Nachbarn nicht mehr angefragt werden. Zweitens ist es günstig, die Nachbarknoten in der aufsteigenden Reihenfolge ihrer Zufallswerte abzuarbeiten, denn für eine Orakelanfrage an einen Knoten mit kleinem Zufallswert ist die erwartete Anzahl der in der Folge dieser Anfrage besuchten Knoten relativ gering. Die Analyse argumentiert nicht direkt über die den Knoten zugeordneten Zufallswerte, sondern über die durch sie induzierte Ordnung der Knoten: Die Anzahl der in der Folge einer Anfrage besuchten Knoten ist abhängig von der Position des angefragten Knotens in dieser Ordnung.

Des Weiteren argumentieren Yoshida et al. nicht über erwartete Anfragenzahlen, sondern über die Summe der nötigen Anfragen über alle n möglichen Startknoten und über alle $n!$ möglichen Permutationen von Knotenordnungen; dieser Wert geteilt durch $n \cdot n!$ ergibt die erwartete Anzahl von für die Beantwortung einer Anfrage ausgeführten Anfragen. Durch vollständige Induktion nach der Nummer des angefragten Knotens i lässt sich zeigen, dass die Summe explorierter Knoten über alle Permutationen höchstens linear in i wächst: Sei π eine beliebige Ordnung, so dass der Knoten v an i-ter Stelle steht und ein Knoten u an der $(i+1)$-ten Stelle; sei π' die Ordnung, in der u und v vertauscht sind, also u an der i-ten und v an der $(i+1)$-ten Stelle steht – dadurch kann eine eventuelle Kante zwischen v und u mit der Ordnung π' begangen werden, mit π aber nicht. $\{u, v\}$ wird in einer Anfrage an v aber nicht mehr erkundet, wenn mindestens eine der Anfragen an zu v benachbarte Knoten mit kleinerer Ordnungsnummer als u *true* zurückliefert, was ausschließt, dass v im Independent Set enthalten ist.

Yoshida et al. zeigen, dass die Anzahl der Ordnungen π', für die $\{u, v\}$ aus diesem Grund nicht erkundet wird, gleich groß ist wie die Anzahl aller Tupel von möglichen Ordnungen π' und darin gültigen von v ausgehenden Explorationspfaden, in denen Knoten „hinter“ u angefragt werden. Intuitiv gesprochen bedeutet das, dass die erwartete Anzahl von Knoten, die bei Anfrage von v hinter der Kante $\{u, v\}$ entdeckt werden, aufgehoben wird durch die geringe Wahrscheinlichkeit, dass $\{u, v\}$ überhaupt exploriert werden muss. Insgesamt steigt die erwartete Anzahl von v aus erkundeter Knoten linear

in der Position i von v in der Ordnung, und durch Summieren über alle Ordnungen und Knoten ergibt sich die oben genannte erwartete Anzahl von $1 + \frac{D}{2}$ explorierten Knoten für eine Anfrage an IndependentSetOrakel.

Die Idee für die Nutzung der Technik von Nguyen und Onak für sublineare Approximationsalgorithmen ist nun, das Independent-Set-Orakel in einem geeigneten Metagraphen auszuführen, der für Anfragen lokal aus dem Ursprungsgraphen berechnet wird. Das führt beispielsweise zu einem sehr einfachen Approximationsalgorithmus für die Größe des maximalen Matchings in einem gegebenen Graphen G: Der Metagraph G' ist hier der zu G duale Graph, so dass eine Anfrage an einen Knoten in G' einer Anfrage an eine Kante in G gleichkommt, und das Orakel gibt *true* zurück, falls keine Nachbarkante mit niedrigerer Zufallszahl im Matching ist. Das inklusionsweise maximale Independent Set auf G', das vom Orakel berechnet wird, ist also auch ein inklusionsweise maximales Matching auf G und seine Größe damit eine 2-Approximation des maximalen Matchings in G.

Der Approximationsalgorithmus muss nun nur noch $\mathcal{O}(D^2/\epsilon^2)$ Kanten sampeln und für jede davon mit Hilfe des Orakels prüfen, ob der entsprechende Knoten in G' im Independent Set liegt; der Anteil der Knoten im Independent Set an der Gesamtzahl gesampelter Knoten kann für eine Approximation der Größe des Independent Set – und damit des entsprechenden Matchings in G – hochgerechnet werden. Mit Hilfe der Hoeffding-Ungleichung lässt sich zeigen, dass dieser Wert mit hoher Wahrscheinlichkeit nur um $\frac{\epsilon}{2}n$ vom tatsächlichen Wert abweicht, so dass sich insgesamt eine multiplikative 2-Approximation der Größe des maximalen Matchings mit einem zusätzlichen additiven Fehler von ϵn ergibt. Da der maximale Grad eines Knotens von G' höchstens $2D - 2$ ist – jede Kante in G ist inzident zu höchstens $2D - 2$ anderen Kanten –, ist die Laufzeit des Algorithmus $\mathcal{O}(2^{\mathcal{O}(D)}/\epsilon^2)$, oder $\mathcal{O}(D^4/\epsilon^2)$ mit Hilfe der verbesserten Analyse von Yoshida et al[4][5].

Die Technik lässt sich erweitern, indem man Metagraphen auf verschiedenen Ebenen angibt, wobei die Orakel hoher Ebenen ihre Antwort von den Orakeln niedrigerer Ebenen abhängig machen. Als Beispiel hierfür geben Nguyen und Onak unter anderem einen sublinearen $(1 + \epsilon)$-Approximationsalgorithmus für die Größe des maximalen Matchings an. Dieser simuliert den Algorithmus von Hopcroft und Karp [45], der sukzessive inklusionsmaximale Mengen von immer längeren *augmentierenden Pfaden* identifiziert und den Kantenstatus der Kanten dieser Pfade „invertiert“, das heißt: Kanten, die bisher im Matching waren, herausnimmt, und Kanten, die bisher nicht enthalten waren, hinzufügt.

In der lokal berechneten Umsetzung steht jede Ebene des Algorithmus für eine Zwischenlösung mit augmentierenden Pfaden der Länge maximal i. Der Metagraph G_i der i-ten Ebene enthält dabei für jeden Pfad in G, der ein augmentierender Pfad der Zwischenlösung der $(i - 1)$-ten Ebene ist, einen Knoten; Kanten in G_i verlaufen zwischen

[4] Der zusätzliche Faktor D kommt dadurch zustande, dass jeder rekursive Aufruf des Knotenorakels zunächst die den Nachbarknoten zugeordneten Zufallszahlen auslesen muss, was in Zeit $\mathcal{O}(D)$ geschehen kann.

[5] Onak et al. konnten für den Approximationsalgorithmus eine weitere Verbesserung angeben [60]: Durch eine geschickte Anfragestrategie und geschicktes Management der noch zu explorierenden Knoten erreichen sie eine Laufzeit von $\tilde{\mathcal{O}}(D \cdot \text{poly}(\epsilon^{-1}))$.

denjenigen Knoten, deren zugeordnete Pfade in G gemeinsame Knoten haben. Das Orakel der i-ten Ebene wählt nun analog zu IndependentSetOrakel eine unabhängige Menge von Knoten von G_i, das heißt eine Menge von augmentierenden Pfaden für die Zwischenlösung der $(i-1)$-ten Ebene, und aufgrund der Konstruktion von G_i haben diese Pfade keine gemeinsamen Knoten, ihre Kanten können also unabhängig voneinander dem Matching hinzugefügt beziehungsweise aus ihm entfernt werden. G_i lässt sich bei Bedarf lokal durch Zugriffe auf das Orakel der $(i-1)$-sten Ebene konstruieren.

Eine Anfrage, ob eine Kante e im Matching auf Ebene j enthalten ist, läuft nun wie folgt ab: Zunächst wird rekursiv geprüft, ob die Kante im Matching auf Ebene $j-1$ vorhanden ist; dann wird das Orakel von Ebene j angefragt, um zu prüfen, ob die Kante in einem der gewählten augmentierenden Pfade der Ebene j enthalten ist. Ist das der Fall, so wird das Ergebnis der rekursiven Anfrage invertiert, andernfalls wird es übernommen.

Zusammen mit der schon für die einfache Variante des Approximationsalgorithmus verwendeten Schätztechnik und einer geeigneten Wahl der maximalen Ebene k – an diese Ebene richtet sich die initiale Anfrage für jede gesampelte Kante – ergibt das einen $(1+\epsilon)$-Approximationsalgorithmus der Größe des maximalen Matchings mit einem zusätzlichen additiven Fehler von ϵn; durch die verbesserten Analysetechniken von Yoshida et al. hat dieser Algorithmus eine Laufzeit von $\mathcal{O}(D^{\mathcal{O}(1/\epsilon^2)} \cdot \epsilon^{-\mathcal{O}(1/\epsilon)})$. Wie wir später sehen werden, kann die Technik von Nguyen und Onak auch im Property-Testing eingesetzt werden.

Random Walks

Eine andere Explorationsmethode, die in dünn besetzten Graphen Anwendung findet, ist die der *Random Walks*. Ein Random Walk entsteht dadurch, dass, ausgehend von einem Startknoten, zufällig eine der inzidenten Kanten gewählt und der Zielknoten dieser Kante als nächster Knoten gewählt wird. Dies wird fortgesetzt, bis eine vorgegebene maximale Pfadlänge erreicht ist.

Ein Random Walk nutzt wie die Breitensuche die Lokalität eines gesampelten Knotens aus, kann aber schneller große Entfernungen zurücklegen. In dicht besetzten Graphen sind Random Walks weniger sinnvoll: Schlimmstenfalls hat hier ein besuchter Knoten $n-1$ Nachbarn, und das zufällige Wählen einer inzidenten Kante bringt dann nicht mehr Informationsgewinn als das zufällig gleichverteilte Sampeln eines Knotens des Graphen. Ein mathematisches Modell, mit dem sich das Verhalten von Random Walks für die Analyse gut erfassen lässt, sind Markowketten: Jeder Knoten stellt einen Zustand in einer Markowkette dar, und jede Kante hat eine bestimmte Wahrscheinlichkeit, einen Übergang zu dem entsprechenden Nachbarknoten zu bewirken. In der Regel wird für jeden Knoten implizit eine Selbstkante hinzugefügt, und für einen Knoten mit Grad k wird die Wahrscheinlichkeit, in einem Random Walk diese Selbstkante zu wählen, auf $1 - \frac{k}{2D}$ gesetzt, während alle „echten“ Kanten eine Übergangswahrscheinlichkeit von $\frac{1}{2D}$ erhalten – dadurch erhält die entsprechende Markowkette einige wünschenswerte Eigenschaften (siehe z.B. [30]).

Eine Eigenschaft, die sich sehr natürlich mit Random Walks testen lässt, ist die Expansion von Graphen. Wir nennen einen ungerichteten Graphen $G = (V, E)$ α-Expander, falls alle $U \subseteq V$, $|U| \leq \frac{1}{2}|V|$, mindestens $\alpha \cdot |U|$ Nachbarn in $U \backslash V$ haben; intuitiv hat G

TesteExpander(G, α, ϵ)
  Sampele zufällig gleichverteilt $\mathcal{O}(1/\epsilon)$ Knoten
  Foreach gesampelter Knoten v **do**
    Starte $m = \mathcal{O}(\sqrt{n}/\epsilon^2)$ der Länge $l = \mathcal{O}(\alpha^{-2} \cdot \log \frac{n}{\epsilon})$ von v
    $\hat{k} \leftarrow$ Anzahl paarweise gleicher Endpunkte der Random Walks
    If $\hat{k} > \frac{1+7\epsilon}{n} \cdot \binom{m}{2}$ **then return false**
  return true

Algorithmus 2.5: TesteExpander*; Nach [30].*

eine gleichmäßig hohe Kantendichte.

Die für Random Walks interessante Eigenschaft von Expandern ist, dass Random Walks sich in ihnen schnell – in $\mathcal{O}(\log n)$ Schritten – ihrer stationären Verteilung annähern, der Gleichverteilung auf allen Knoten. Das heißt, nach $\mathcal{O}(\log n)$ Schritten ist in einem guten Expander die Wahrscheinlichkeit, in einem bestimmten Knoten angelangt zu sein, ungefähr gleich für alle Knoten; in einem schlechten Expander wiederum ist die Wahrscheinlichkeit für manche Knoten wesentlich höher als für andere. Das lässt sich für einen Property-Testing-Algorithmus mit zweiseitigem Fehler ausnutzen, wie Czumaj und Sohler [30] auf Basis von Ergebnissen von Goldreich und Ron [44] zeigen konnten: Ihr Algorithmus sampelt $\mathcal{O}(1/\epsilon)$ Knoten zufällig und gleichverteilt und startet von jedem dieser Knoten aus $m = \mathcal{O}(\sqrt{n}/\epsilon^2)$ Random Walks der Länge $l = \mathcal{O}(\alpha^{-2} \cdot \log \frac{n}{\epsilon})$ (siehe Algorithmus 2.5). Die Anzahl der Random Walks und ihre Länge führt dazu, dass die erwartete Anzahl von Kollisionen, das heißt von Random Walks, die gleiche Start- und Zielknoten haben, $\Omega(1)$ ist, selbst wenn die Verteilung der Zielknoten sehr nahe der Gleichverteilung ist. Genauer gesagt konnten Goldreich und Ron in ihrer genannten Veröffentlichung zeigen, dass für jeden Startknoten $v \in V$ der Erwartungswert für die Anzahl der Kollisionen $\binom{m}{2} \cdot ||P_v^l||_2^2$ ist, wobei die P_v^l die Verteilung der Endknoten eines Random Walks der Länge l von v aus angeben; zudem ist die Varianz relativ klein. Für einen guten Expander gilt nach der obigen Diskussion $||P_v^l||_2^2 \approx \frac{1}{n}$, für einen schlechten Expander ist der Wert zumindest für viele Knoten deutlich größer. Czumaj und Sohler nutzen dies, um zu zeigen, dass ihr Algorithmus in einem α-Expander mit hoher Wahrscheinlichkeit für alle gesampelten Startknoten höchstens $\frac{1+7\epsilon}{n}\binom{m}{2}$ Kollisionen erreicht, während es in jedem Graphen, der ϵ-fern von einem $\Theta(\alpha^2/\log \frac{n}{\epsilon})$-Expander ist, mit hoher Wahrscheinlichkeit für mindestens einen Startknoten mehr Kollisionen sind[6].

[6] Zwei Nachfolgearbeiten haben dieses Ergebnis inzwischen verbessert: Kale und Seshadhri konnten die Schranke für die Expansion abzulehnender Graphen auf $\Theta(\alpha^2)$ erhöhen, in einer Vorversion von [51] zunächst allerdings nur unter der Bedingung, dass für die Feststellung der ϵ-Entferntheit auch ein Vergleich mit Graphen mit einem maximalen Knotengrad von $2D$ erlaubt ist. Nachmias und Shapira konnten zeigen, dass eine Schranke von $\Theta(\alpha^2)$ auch dann gilt, wenn die Vergleichsgraphen ebenfalls einen maximalen Knotengrad von D haben, wie üblich für Property-Testing in gradbeschränkten Graphen [58]. Beide Algorithmen haben in Abhängigkeit von n eine Anfragekomplexität von $\mathcal{O}(n^{\mu+1/2})$ für eine beliebig kleine Konstante $\mu > 0$, asymptotisch also etwas größer als der Algorithmus von Czumaj und Sohler.

TesteBipartitheit(G, ϵ)
  Sampele zufällig gleichverteilt $\mathcal{O}(1/\epsilon)$ Knoten
  Foreach gesampelter Knoten v **do**
    Starte $m = \mathcal{O}(\text{poly}(\epsilon^{-1} \log n) \cdot \sqrt{n})$ der Länge $l = \mathcal{O}(\text{poly}(\epsilon^{-1} \log n))$ von v
    If zwei der Random Walks berühren sich in $u \in V(G)$ derart, dass die entsprechenden Pfade von v nach u einen Kreis ungerader Länge ergeben
      then return false
  return true

Algorithmus 2.6: TesteBipartitheit*; Nach [42].*

Random Walks lassen sich für das Testen weiterer Grapheigenschaften einsetzen; die Schwierigkeit ist hier insbesondere, in der Analyse mit Graphen umzugehen, in denen die Random Walks verschieden schnell gegen die stationäre Verteilung konvergieren. Eine interessante Eigenschaft, für die ein Property-Testing-Algorithmus von Goldreich und Ron vorgestellt wurde [42], ist Bipartitheit. Ein Graph ist genau dann bipartit, wenn er keine Kreise ungerader Länge enthält, und der Algorithmus macht sich dies zunutze, indem er mit Hilfe von Random Walks versucht, einen Kreis ungerader Länge als Zeugen gegen die Eigenschaft zu finden: Für jeden der $\Theta(1/\epsilon)$ gesampelten Startknoten werden $m = \mathcal{O}(\text{poly}(\epsilon^{-1}, \log n) \cdot \sqrt{n})$ Random Walks der Länge $l = \mathcal{O}(\text{poly}(\epsilon^{-1}, \log n))$ gestartet. Die Idee ist, dass durch zwei der Random Walks mit gemeinsamem Ausgangsknoten v ein Kreis identifiziert wird, wenn die Random Walks sich auf einem beliebigen weiteren Knoten begegnen; hat genau einer der Random Walks bis dahin eine ungerade Anzahl von Kanten benutzt, dann ist dies ein Kreis ungerader Länge (siehe Abbildung 2.5). Falls für einen der Startknoten ein solcher Kreis gefunden wird, lehnt der Algorithmus ab, ansonsten akzeptiert er; es handelt sich demnach um einen Algorithmus mit einseitigem Fehler (siehe Algorithmus 2.6).

Es ist also zu zeigen, dass der Algorithmus für jeden von Bipartitheit ϵ-entfernten Graphen G einen Kreis ungerader Länge identifiziert. Wir vollziehen die Analyse von Goldreich und Ron relativ detailliert nach für Graphen G, in denen sich die Verteilungen der Zielknoten von Random Walks schnell der Gleichverteilung nähern, zum Beispiel weil G ein guter Expander ist. Goldreich und Ron nehmen für diesen Fall an, dass für Random Walks der Länge l von beliebigen Startknoten in G aus jeder Knoten von G mit Wahrscheinlichkeit mindestens $\frac{1}{2n}$ und höchstens $\frac{2}{n}$ Zielknoten ist.

Die Grundidee von Goldreich und Ron ist die Folgende: Zunächst werden nur Kollisionen auf dem letzten Knoten eines Random Walks mit l Schritten betrachtet. Solche Random Walks können Pfaden unterschiedlicher Länge entsprechen, da Selbstkanten bei der Pfadlänge nicht mitgezählt werden – wohl aber bei der Anzahl der Schritte des Random Walk. Uns interessiert vor allem die „Parität“ der Pfadlänge, das heißt ob es sich um einen Pfad gerader oder ungerader Länge handelt. Sei s ein beliebiger gesampelter Knoten, der Startknoten für m der Random Walks ist. Die erste Beobachtung ist, dass in einem von Bipartitheit ϵ-entfernten Graphen viele Paare von adjazenten Knoten u und v existieren, so dass beide Knoten mit einem von s ausgehenden Random Walk der Länge

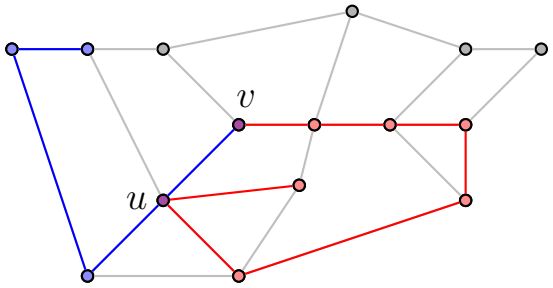

Abbildung 2.5: *Zwei von v ausgehende Random Walks (rote und blaue Kanten) begegnen sich in u und identifizieren dadurch einen Kreis ungerader Länge.*

l mit hoher Wahrscheinlichkeit auf einem Pfad gleicher Parität erreicht werden.

Die zweite Beobachtung ist, dass ein Knoten, der mit hoher Wahrscheinlichkeit nach l Schritten auf einem bestimmten Pfad erreicht wird, auch mit relativ hoher Wahrscheinlichkeit schon nach $l-1$ Schritten auf dem gleichen Pfad erreicht wird: Dazu muss entlang des Pfades eine Selbstkante weniger benutzt werden. Das lässt sich mit der ersten Beobachtung kombinieren, denn für ein Knotenpaar u, v, wie dort angenommen, ist nun die Wahrscheinlichkeit hoch, dass beide Knoten auf Pfaden der gleichen Parität erreicht werden, der eine Knoten nach l Random-Walk-Schritten, der andere nach $l-1$. Die Wahrscheinlichkeit, dass für letzteren Random-Walk der verbleibende Schritt die Kante zum ersten Knoten wählt, ist aber auch recht groß, nämlich $\frac{1}{2D}$, und das führt zu einer hohen Wahrscheinlichkeit für das Entdecken eines ungeraden Kreises an diesem Knoten.

Haben die beiden Pfade schon zuvor Kollisionen auf anderen Knoten, dann ist durch die unterschiedliche Parität der Pfade garantiert, dass zumindest einer der durch sie induzierten Kreise eine ungerade Länge hat.

Sei im Detail G ein von Bipartitheit ϵ-entfernter Graph und sei p_v^0 die Wahrscheinlichkeit, einen Knoten $v \in V$ auf einem Random Walk von s aus als Zielknoten zu erreichen, wobei der zurückgelegte Pfad gerade Länge hat; sei p_v^1 die Wahrscheinlichkeit, dass v Endknoten eines Random Walks von s aus ist und dabei ein Pfad ungerader Länge beschritten wurde. Es gilt $\max\{p_v^0, p_v^1\} \geq \frac{1}{4n}$, da wir angenommen haben, dass jeder Knoten mit Wahrscheinlichkeit mindestens $\frac{1}{2n}$ Zielknoten eines Random Walk ist.

Goldreich und Ron betrachten die folgende Partitionierung von V: V_1 enthält die Knoten v mit $p_v^0 \geq p_v^1$ und V_2 die mit $p_v^1 > p_v^0$. Sei v ein Knoten aus V_0: Das Produkt $p_v^0 \cdot p_v^1$ – und damit die Wahrscheinlichkeit, dass zwei Random Walks v auf Pfaden unterschiedlicher Länge erreichen – wird groß, wenn v viele Nachbarn in V_0 hat: Für jeden dieser Nachbarn gilt $p_u^0 \geq \frac{1}{4n}$, u ist also mit Wahrscheinlichkeit mindestens $\frac{1}{4n}$ Zielknoten eines Random Walk mit l Schritten, bei gerader Pfadlänge; damit ist u allerdings auch mit Wahrscheinlichkeit $\Theta(\frac{1}{n})$ Zielknoten eines Random Walk mit $l-1$ Schritten, bei ebenfalls gerader Pfadlänge. Da aber von u mit Wahrscheinlichkeit $\frac{1}{2D}$ v der nächste Knoten eines weiteren Random-Walk-Schritts ist, folgt aus dem Vorhandensein einer Kante zwischen u und v, dass die Wahrscheinlichkeit, mit einem Random Walk von l Schritten v über u auf einem Pfad dann ungerader Länge zu erreichen, mindestens $\frac{1}{2D} \cdot \Theta(\frac{1}{n}) = \Theta(\frac{1}{Dn})$ ist. Falls v D_0 Nachbarn in V_0 hat, folgt also $p_v^1 \geq \Theta(\frac{D_0}{Dn})$. Analog gilt $p_v^0 \geq \Theta(\frac{D_1}{Dn})$ für den Fall,

dass v in V_1 enthalten ist und D_1 Nachbarn in V_1 hat. Es folgt jeweils $p_v^0 \cdot p_v^1 \geq \Theta(\frac{D_0}{Dn^2})$.

Da G ϵ-fern von bipartit, ist die Summe $\sum_{v \in V} p_v^0 \cdot p_v^1$ damit relativ groß, mindestens $\Theta(\frac{\epsilon}{n})$, denn für jede Partitionierung von V in zwei Teilmengen muss G mehr als $\epsilon Dn/2$ Kanten zwischen Knoten der gleichen Mengen haben, also auch für die Partitionierung in V_1 und V_2. Damit ist die Wahrscheinlichkeit, dass zwei Random Walks von s aus den gleichen Zielknoten haben, einmal auf einem geraden, einmal auf einem ungeraden Pfad, mindestens $2 \cdot \sum_{v \in V} p_v^0 \cdot p_v^1 \geq \Theta(\frac{\epsilon}{n})$, und wegen $m = \Theta(\sqrt{n/\epsilon})$ gilt ähnlich dem Geburtstagsparadoxon, dass die erwartete Anzahl solcher Kollisionen $\Theta(1)$ ist. Da aufgrund der nach oben und unten beschränkten Wahrscheinlichkeiten p_v^i auch die Varianz klein ist, lässt sich mit Hilfe der Tschebyschow-Ungleichung beweisen, dass mit hoher Wahrscheinlichkeit mindestens eine solche Kollision auftritt.

Der zweite Fall ist wesentlich aufwändiger zu handhaben[7]: Das Verhalten von Random Walks ist in allgemeinen Graphen schwer zu analysieren; stattdessen analysieren Goldreich und Ron die Struktur derjenigen Graphen, in denen die Random-Walk-Technik für viele Startknoten schlecht funktioniert. Solche Knoten – Goldreich und Ron bezeichnen sie als *gute* Knoten – charakterisieren sie dadurch, dass die Wahrscheinlichkeit, dass m Random Walks der Länge l von einem dieser Knoten aus einen Kreis ungerader Länge entdecken, höchstens $\frac{1}{10}$ ist. Wir sind daran interessiert, dass TesteBipartitheit möglichst viele nicht gute Knoten sampelt, denn dann wird mit hoher Wahrscheinlichkeit ein Kreis ungerader Länge entdeckt.

Für einen Widerspruchsbeweis nehmen Goldreich und Ron dafür an, dass es höchstens $\frac{1}{16}\epsilon n$ nicht gute Knoten gibt. Aufgrund von Eigenschaften der guten Knoten und ihrer Umgebung lässt sich allerdings eine Partitionierung der Knoten von G in zwei Teilmengen konstruieren, die höchstens ϵDn Fehlerkanten aufweist, also Kanten zwischen Knoten der gleichen Teilmengen: ein Widerspruch zu der Annahme, dass G ϵ-entfernt von Bipartitheit ist[8]. Es muss also mehr als $\frac{1}{16}\epsilon n$ nicht gute Knoten geben, und demnach sampelt TesteBipartitheit(G, ϵ) mit hoher Wahrscheinlichkeit hinreichend viele nicht gute Knoten, so dass mit hoher Wahrscheinlichkeit für einen von ihnen ein Kreis ungerader Länge entdeckt wird.

Kategorisierung der testbaren Eigenschaften in gradbeschränkten Graphen

In den vergangenen Jahren hat eine Serie von Ergebnissen für eine große Klasse von Grapheigenschaften gezeigt, dass sie in dünn besetzten Graphen mit einer von der Knotenzahl unabhängigen Anzahl von Anfragen getestet werden kann [26, 15, 46, 59]. Im Gegensatz zu den dicht besetzten Graphen, bei denen wie oben gesehen die Testbarkeit durch globale Struktureigenschaften des Graphen definiert ist, sind es hier allerdings lokale Struktureigenschaften, die ausgenutzt werden.

Der zentrale Schritt in dieser Reihe von Ergebnissen ist das von Benjamini, Schramm und Shapira [15]. Sie zeigen, dass für alle monotonen hyperfiniten Grapheigenschaften Property-Testing-Algorithmen mit einer von der Eingabegröße unabhängigen Laufzeit

[7] Die Darstellung hier folgt dem in [66] gegebenen Überblick.

[8] Goldreich und Ron definieren ϵ-Entferntheit im Gegensatz zu dieser Arbeit über mehr als ϵDn nötige Kantenoperationen, um Isomorphie zu einem Graphen mit der Eigenschaft herzustellen.

existieren; ein Graph G ist (ϵ, k)-hyperfinit, wenn es möglich ist, $\epsilon|V(G)|$ Kanten aus G derart zu entfernen, dass der entstehende Graph nur noch Zusammenhangskomponenten der Größe maximal k enthält. Eine Grapheigenschaft ist hyperfinit, wenn alle Graphen mit der Eigenschaft sich für feste ϵ und k so partitionieren lassen. Eine Grapheigenschaft ist monoton, wenn sie unter Entfernen von Knoten und Kanten abgeschlossen ist.

Eine wichtige Unterklasse der monotonen hyperfiniten Grapheigenschaften ist die der durch verbotene Minoren definierten Grapheigenschaften (siehe [3]): Eine Grapheigenschaft $\mathcal{P}$ ist unter Minorenbildung abgeschlossen, falls für jeden Graphen G, der diese Eigenschaft besitzt, auch alle Graphen G' sie besitzen, die durch Löschen von Knoten und Kanten und Kontrahieren von Kanten aus G entstehen können – solche Graphen G' nennen wir Minoren von G. Planarität ist beispielsweise eine solche Eigenschaft, denn ein Graph ist genau dann planar, wenn die Graphen K_5 und $K_{3,3}$ nicht als Minoren vorkommen.

Benjamini et al. zeigen zunächst, dass sich Hyperfinitheit testen lässt. Dazu betrachten sie R-Nachbarschaften von Knoten v, das heißt Subgraphen, die durch die Knoten in Entfernung maximal R zu v induziert werden. Sei A eine Klasse von (ϵ, k)-hyperfiniten, auf Knotengrad D beschränkten Graphen und sei B die Klasse von auf Knotengrad D beschränkten Graphen, die weit entfernt davon sind, $(4\epsilon \log(4D/\epsilon), k)$-hyperfinit zu sein. Benjamini et al. beweisen, dass ein R existiert, das nur von d, k und ϵ abhängt, so dass zwischen allen Paaren von Graphen aus A und B signifikante Unterschiede in den Verteilungen der Isomorphieklassen der R-Nachbarschaften bestehen.

A und B sind dadurch durch einen Property-Testing-Algorithmus unterscheidbar: Ein solcher Algorithmus wählt zufällig Knoten aus dem Eingabegraphen G und bestimmt die Isomorphieklassen ihrer R-Nachbarschaften; die Anzahl möglicher Klassen hängt nur von R und damit nicht von n ab. Die ermittelten Häufigkeiten der einzelnen Isomorphieklassen werden dann mit den Häufigkeiten aller Graphen aus A verglichen (beziehungsweise einer konstant großen repräsentativen Auswahl der Graphen von A; Benjamini et al. zeigen, dass es eine solche Auswahl gibt). Ist der Unterschied signifikant kleiner als der minimale Unterschied zwischen zwei Graphen aus A und B sein kann, so wird der Graph in A vermutet, sonst in B. Dieser Property-Testing-Algorithmus hat natürlich einen zweiseitigen Fehler, da eine gewisse Wahrscheinlichkeit besteht, in G die R-Nachbarschaften einer nicht repräsentativen Stichprobe von Knoten zu untersuchen.

Die Grundidee des Testers für beliebige monotone hyperfinite Grapheigenschaften A ist nun wie folgt: Zunächst kann getestet werden, ob der eingegebene Graph (ϵ', k)-hyperfinit ist. Dieser Test akzeptiert nur dann mit hoher Wahrscheinlichkeit, wenn der Graph zumindest $(4\epsilon' \log(4D/\epsilon'), k)$-hyperfinit ist; ansonsten kann abgelehnt werden, denn ein nicht hyperfiniter Graph kann die Eigenschaft A nicht haben.

In der Folge kann also davon ausgegangen werden, dass eine Partitionierung des Eingabegraphen in konstant große Teilgraphen existiert, so dass für die Partitionierung nur $\tilde{\mathcal{O}}(\epsilon' n)$ Kanten gelöscht werden müssen; der entstehende Graph G' ist für geeignete Wahl von ϵ' also immer noch $\Theta(\epsilon)$-entfernt von A, wenn G ϵ-entfernt von A ist. Auch wenn also keine direkten Zeugen gegen die Eigenschaft in den Partitionen vorhanden sind, muss es dort strukturelle Merkmale geben, die sich nicht mit der Eigenschaft in Einklang bringen lassen. Das Problem ist, dass dem Property-Testing-Algorithmus von Benjamini

et al. die geeignete Partitionierung nicht bekannt ist; also untersucht er stattdessen die R-Nachbarschaften von in G' gesampelten Knoten – deren Verteilung ist hinreichend unterschiedlich zu der der Graphen in A, wenn G' $\Theta(\epsilon)$-entfernt von A ist. Weiterhin zeigen Benjamini et al., dass dies wiederum nur für eine konstante Anzahl von Graphen aus A geprüft werden muss: Die restlichen Graphen in A haben diesen Graphen sehr ähnliche Verteilungen der R-Nachbarschaften.

Zu zeigen, dass die Verteilungen von R-Nachbarschaften reichen, um hyperfinite von nicht hyperfiniten Graphen zu unterscheiden, ist die wichtigste Erkenntnis gegenüber der Vorgängerarbeit von Czumaj et al. [26]: Dort wird vorausgesetzt, dass die möglichen Eingabegraphen einer vererbbaren Klasse von Graphen mir schwacher Expansion angehören – vererbbar bedeutet, dass die Klasse gegenüber dem Entfernen von Knoten und Kanten abgeschlossen ist. Aus der Kombination dieser beiden Eigenschaften folgt insbesondere, dass alle Graphen der Klasse eine gute Partitionierung im obigen Sinne haben, was sich der Property-Testing-Algorithmus von Czumaj et al. zunutze macht, indem er ebenfalls die R-Nachbarschaften gesampelter Knoten untersucht. Dadurch, dass Hyperfinitheit selbst testbar ist, fällt die Einschränkung der möglichen Eingabegraphen durch das Ergebnis von Benjamini et al. weg.

Eine interessante aus der Arbeit von Benjamini et al. motivierte Frage ist, ob man eine gute Partitionierung eines hyperfiniten Graphen explizit konstruieren kann. Hassidim et al. [46] konnten dafür ein Partitionierungsorakel angeben, das mit Hilfe der oben vorgestellten Technik von Nguyen und Onak für einen Knoten dessen Partition in einer Approximation der optimalen Partitionierung zurückliefert; die Anfragekomplexität ist $\mathcal{O}(2^{\mathrm{poly}(\epsilon,d)})$. Dieses Orakel ermöglicht unter anderem einen einfachen Algorithmus zum Testen von Hyperfinitheit – es werden $\mathcal{O}(1/\epsilon^2)$ Knoten zufällig ausgewählt, deren Partitionen werden mittels des Orakels ermittelt und für einen zufälligen Nachbarn jedes der gesampelten Knoten wird geprüft, ob er ebenfalls in der jeweiligen Partition liegt. Dies liefert einen Schätzwert für die Anzahl der Knoten, die durch die Partitionierung aus dem Graphen entfernt werden, und wenn dieser zu groß ist, ist der Graph mit hoher Wahrscheinlichkeit nicht hyperfinit. Mit diesem Verfahren ist auch ein einfacher Tester für Minorenfreiheit möglich, der zunächst auf Hyperfinitheit testet und, falls dieser Test positiv ist – für hinreichend große Graphen ist Hyperfinitheit eine notwendige Voraussetzung für die Freiheit von beliebigen Minoren –, für weitere $\mathcal{O}(1/\epsilon)$ gesampelte Knoten deren Partitionen auf Vorkommen der gesuchten Minoren durchsucht. Da durch eine Partitionierung eines minorenfreien Graphen nur wenige Kanten gelöscht werden, muss es im Fall eines Graphen, der viele Vorkommen der gesuchten Minoren enthält, auch in den einzelnen Partitionen noch viele Vorkommen geben; in einem minorenfreien (und damit hyperfiniten) Graphen können mit hoher Wahrscheinlichkeit alle durch das Partitionierungsorakel zurückgelieferten Partitionen in konstanter Zeit exploriert werden, da sie konstante Größe haben.

Zuletzt konnten Newman und Sohler [59] aufbauend auf den genannten Ergebnissen zeigen, dass jede – also nicht nur jede monotone – hyperfinite Grapheigenschaft in konstanter Zeit testbar ist, ebenso wie jede Eigenschaft hyperfiniter Graphen; aus letzterem folgt insbesondere auch, dass in planaren Graphen jede Grapheigenschaft in konstanter Zeit getestet werden kann. Ihr zentrales Ergebnis ist, dass zwei hyperfinite Graphen,

deren Isomorphieklassen von R-Nachbarschaften eine ähnliche Verteilung haben, selbst ähnlich sind; damit lässt sich ein Property-Testing-Algorithmus für Graphisomorphie von hyperfiniten Graphen angeben, und daraus folgt, dass jede unter Graphisomorphie abgeschlossene Grapheigenschaft in konstanter Zeit testbar ist.

2.2.2 Techniken für Property-Testing in gerichteten Graphen

Grundsätzlich sollten sich die oben vorgestellten Explorationstechniken auch in gerichteten Graphen anwenden lassen. Für dicht besetzte Graphen legt ein Ergebnis von Alon und Shapira [8] nahe, dass dies tatsächlich gilt: Sie zeigen mit einer „gerichteten Version“ des Regularity Lemma, dass in gerichteten dicht besetzten Graphen Subgraphfreiheit für feste Subgraphen H durch das Sampeln einer konstanten Anzahl von Knoten und das Anfragen des induzierten Subgraphen dieser Knoten testbar ist.

In dünn besetzten Graphen sind die Techniken des Property-Testings ungerichteter Graphen je nach Modell nicht so einfach anwendbar: In dieser Arbeit nehmen wir wie bei ungerichteten dünn besetzten Graphen an, dass ein Graph G im Adjazenzlistenmodell gespeichert ist; dabei enthalten die Adjazenzlisten nur die ausgehenden Kanten des entsprechenden Knotens. Darum können mit Hilfe der Zugriffsfunktion f_G auch nur diese erfragt werden. Um eine eingehende Kante eines Knotens zu identifizieren, muss also der entsprechende Adjazenzlisteneintrag ihres Ursprungsknotens angefragt werden. Die Länge der Adjazenzlisten ist, wie im ungerichteten Modell, durch eine Konstante $D \in \mathbb{N}$ beschränkt. Für die meisten hier betrachteten Probleme werden wir zudem annehmen, dass die Anzahl eingehender Kanten jedes Knotens ebenfalls höchstens D ist.

Da die maximale Beschreibungsgröße eines so gespeicherten Graphen mit n Knoten Dn ist, ist ein Graph G ϵ-fern von einem Graphen H, wenn mehr als ϵDn Adjazenzlisteneinträge von G geändert werden müssen, um H zu erhalten; da jede Kante nur in einer Adjazenzliste gespeichert ist – der des Ausgangsknotens –, entspricht dies genau der Anzahl einzufügender beziehungsweise zu löschender Kanten. Den Modellen in ungerichteten Graphen entsprechend ist G ϵ-fern von einer Grapheigenschaft Π, wenn G ϵ-fern von allen Graphen mit dieser Eigenschaft ist.

Dies ist ein von Bender und Ron in [14] vorgeschlagenes Modell, wobei die beschränkte Anzahl eingehender Kanten pro Knoten von Bender und Ron explizit nur für einen Algorithmus in einem vereinfachten Modell gefordert wird, in dem Anfragen auf die eingehenden Kanten von Knoten möglich sind; die Konstruktionen unterer Schranken in derselben Veröffentlichung halten die Beschränkung der Anzahl eingehender Kanten allerdings ebenfalls ein. Wir werden von diesem Modell in den folgenden Kapiteln teilweise geringfügig abweichen; solche Abweichungen werden im jeweiligen Kapitel angegeben und diskutiert.

Grundsätzlich lassen sich Breitensuchen und Random Walks natürlich auch in diesem Modell einsetzen, und falls die eingehenden Kanten von Knoten für Algorithmen sichtbar sind, ergeben sich für viele Probleme Algorithmen, die ähnlich wie in ungerichteten Graphen funktionieren. Durch die Einschränkung, dass die eingehenden Kanten von Knoten nicht sichtbar sind, werden die Probleme schwieriger: Bender und Ron konnten beispielsweise zeigen, dass starker Zusammenhang bei sichtbaren eingehenden Kanten mit Hilfe

der Breitensuchtechnik mit Anfragekomplexität $\tilde{\mathcal{O}}(1/\epsilon)$ gestestet werden kann; für das Modell mit sichtbaren eingehenden Kanten geben sie allerdings eine untere Schranke von $\Omega(\sqrt{n})$ für die Anfragekomplexität jedes Property-Testing-Algorithmus. Bender und Ron geben zusätzlich noch eine untere Schranke von $\Omega(n^{1/3})$ für Kreisfreiheit an, aber keine Property-Testing-Algorithmen für das Modell, in dem eingehende Kanten nicht angefragt werden können.

Wir werden auf die beiden Schrankenkonstruktionen von Bender und Ron in Kapitel 2.2.3 genauer eingehen; eine wesentliche Erkenntnis für das Entwickeln von Algorithmen im hier betrachteten Modell sei jedoch vorweggenommen: Die durch einen Property-Testing-Algorithmus zu unterscheidenden Graphklassen wurden von Bender und Ron jeweils so gewählt, dass sie nicht unterschieden werden können, wenn der vom Algorithmus untersuchte Bereich keine „Kollision“ enthält. Als Kollision ist dabei zu verstehen, wenn der Zielknoten einer von einem Algorithmus angefragten Kante schon durch frühere Anfragen bekannt ist.

Das bedeutet, dass das Auffinden solcher Kollisionen Teil jeder Lösungsstrategie für diese Probleme sein muss. Hinter dieser Feststellung verbirgt sich letztlich die Notwendigkeit, eingehende Kanten von Knoten kennenzulernen: Um für einen gesampelten Knoten eine eingehende Kante zu sehen, muss diese Kante von ihrem Startknoten aus exploriert werden. Je nach Graphstruktur ist das nur dadurch zu erreichen, dass die Kante von einem anderen gesampelten Knoten aus entdeckt wird, zum Beispiel mit Hilfe einer Breitensuche oder eines Random Walk, oder auch dadurch, dass dieser Knoten selbst gesampelt wird.

Sämtliche im Rahmen dieser Arbeit vorgestellten Property-Testing-Algorithmen haben aus diesem Grund eine Anfragekomplexität von $\Omega(\sqrt{n})$ – einzige Ausnahme ist das Testen auf Subgraph-Freiheit, wenn die verbotenen Subgraphen stark zusammenhängend sind –, und für fast alle getesteten Grapheigenschaften gibt es untere Schranken, die zeigen, dass diese Schwelle asymptotisch nicht unterschritten werden kann.

Die interessante Frage ist, wieviel Wissen über die Verteilungen eingehender Kanten für das Testen verschiedener Grapheigenschaften nötig ist. Durch Sampeln von $\Theta(n^{1-1/D})$ Kanten lässt sich in einem durch D gradbeschränkten Graphen im Prinzip vollständige Information über diese Verteilung gewinnen, da so mit hoher Wahrscheinlichkeit auch D-Fach-Kollisionen gesampelt werden, also alle D eingehenden Kanten mindestens einer konstanten Anzahl von Knoten im Sample enthalten sind; dies ist also eine natürliche Schranke für die Anzahl von Anfragen für das Testen von Eigenschaften. Die Frage ist, für welche Eigenschaften diese vollständige Information nötig ist, und für welche Probleme es ausreicht, mit $\Theta(\sqrt{n})$ Samples überhaupt mit hoher Wahrscheinlichkeit Kollisionen zu erhalten. Wir werden in Kapitel 3 Beispiele für beide Typen von Eigenschaften sehen.

Es seien an dieser Stelle auch kurz dem Autor bekannte Ergebnisse im Modell gradbeschränkter Graphen mit Sichtbarkeit eingehender und ausgehender Kanten erwähnt; für dieses Modell geben Bender und Ron einen Property-Testing-Algorithmus mit Anfragekomplexität $\tilde{\mathcal{O}}(1/\epsilon)$ für starken Zusammenhang an, der versucht, durch Sampling von Knoten und anschließende Breitensuche sowohl über ein- als auch ausgehende Kanten kleine Quell- und Zielkomponenten zu identifizieren [14]. Yoshida und Ito konnten außerdem einen Property-Testing-Algorithmus für k-Kanten-Zusammenhang in diesem

Modell angeben, der eine Anfragekomplexität von $\tilde{\mathcal{O}}((ck/\epsilon)^k)$ hat [72]; $c > 1$ ist eine Konstante. Orenstein und Ron konnten zeigen, dass auch k-Knoten-Zusammenhang mit dieser Anfragekomplexität testbar ist; außerdem geben sie einen Algorithmus mit Anfragekomplexität $\tilde{\mathcal{O}}(1/\epsilon)$ an, der testet, ob ein Graph eulersch ist [61]. Sie untersuchen diese Eigenschaften außerdem im Adjazenzlistenmodell *ohne* beschränkten Knotengrad und stellen in diesem Modell für eulersche Graphen einen Algorithmus mit einer Anfragekomplexität von $\tilde{\mathcal{O}}(\sqrt{n}/\epsilon^{3/2})$ vor, zusammen mit einer unteren Schranke von $\Omega(\sqrt{n/\epsilon})$. Für k-Knoten-Zusammenhang und k-Kanten-Zusammenhang geben sie für dieses Modell Algorithmen mit einer Anfragekomplexität von $\tilde{\mathcal{O}}((kc/\epsilon)^{k+1})$ an, wiederum für eine Konstante $c > 1$.

2.2.3 Techniken für untere Schranken

Nicht alle Grapheigenschaften lassen sich mit der gleichen Anfragekomplexität testen. Dieser Abschnitt stellt Techniken vor, mit deren Hilfe sich für das Testen von Grapheigenschaften untere Schranken für die Anfragekomplexität zeigen lassen. Dabei kann auch hier zwischen Algorithmen mit einseitigem und mit zweiseitigem Fehler unterschieden werden.

Eine gebräuchliche Technik für das Herleiten unterer Schranken für die Güte randomisierter Algorithmen ist *Yaos Minimax-Prinzip* [71], das im Wesentlichen besagt, dass ein randomisierter Algorithmus im Erwartungswert auf einer Worst-Case-Eingabe nur so gut sein kann wie ein optimaler deterministischer Algorithmus im Erwartungswert auf einer beliebigen zufälligen Verteilung der Eingaben. Damit kann statt einer Schranke für randomisierte Algorithmen eine für deterministische Algorithmen auf zufälligen Eingaben hergeleitet werden, und daraus lässt sich die Schranke für erstere gewinnen. Die für untere Schranken für Property-Testing-Algorithmen gebräuchliche Formulierung dieses Prinzips ist die Folgende, die zum Beispiel Raskhodnikova und Smith angeben [65]:

Lemma 2.3 (siehe [65], Anhang A)**.** *Um für ein Entscheidungsproblem eine untere Schranke q für die Worst-Case-Anfragekomplexität jedes randomisierten Algorithmus zu zeigen, reicht es aus, Verteilungen über zwei Klassen $\mathcal{P}$ und $\mathcal{N}$ von Eingaben anzugeben, wobei*

- *$\mathcal{P}$ aus zu akzeptierenden Eingaben und*
- *$\mathcal{N}$ aus abzulehnenden Eingaben besteht*

und $\mathcal{P}$ und $\mathcal{N}$ für jeden deterministischen Algorithmus schwer zu unterscheiden sind, der höchstens q Anfragen an die Eingabe stellt.

Mit „schwer zu unterscheiden“ meinen Raskhodnikova und Smith, dass für jeden deterministischen Algorithmus mit höchstens q Anfragen die statistische Distanz zwischen den Verteilungen der von Eingaben aus $\mathcal{P}$ und $\mathcal{N}$ gelesenen Elemente kleiner als $\frac{1}{3}$ ist. Daraus folgt auch, dass dann die statistische Distanz zwischen den Rückgabewerten jedes solchen deterministischen Algorithmus kleiner als $\frac{1}{3}$ ist – das heißt, wenn der Algorithmus

mit Wahrscheinlichkeit $\frac{2}{3}$ Eingaben aus $\mathcal{P}$ akzeptiert, akzeptiert er auch Eingaben aus $\mathcal{N}$ mit Wahrscheinlichkeit mehr als $\frac{1}{3}$.

Im Kontext des Property-Testing von Graphen lässt sich $\mathcal{P}$ als Klasse von Graphen angeben, die die getestete Eigenschaft besitzen, und $\mathcal{N}$ als Klasse von Graphen, die ϵ-entfernt von ihr sind; falls die obige Bedingung bezüglich der statistischen Distanz des Ergebnisses für alle deterministischen Algorithmen gezeigt werden kann, bedeutet das, dass eine der beiden im Property-Testing geforderten Erfolgswahrscheinlichkeiten – die für Graphen mit der Eigenschaft oder die für ϵ-entfernte Graphen – für deterministische Algorithmen mit q Anfragen auf geeigneter Eingabeverteilung nicht eingehalten werden kann. Mit Hilfe des obigen Lemmas folgt dann, dass jeder Property-Testing-Algorithmus für die Eigenschaft mehr als q Anfragen benötigt.

Falls lediglich eine untere Schranke für Property-Testing-Algorithmen mit einseitigem Fehler angegeben werden soll, kann auf die Angabe der Klasse $\mathcal{P}$ der Graphen mit der entsprechenden Eigenschaft verzichtet werden: Solange sich der einem Algorithmus bekannte Teilgraph eines Graphen aus $\mathcal{N}$ zu einem Graphen mit der Eigenschaft vervollständigen lässt, darf der Algorithmus nicht ablehnen, denn der so vervollständigte Graph müsste mit Wahrscheinlichkeit 1 akzeptiert werden. Ein Algorithmus mit einseitigem Fehler versucht also, einen Beweis zu finden, dass der eingegebene Graph die getestete Eigenschaft nicht besitzt; wir werden einen solchen Beweis im Folgenden *Zeugen* gegen die Eigenschaft nennen, und um eine untere Schranke zu zeigen, genügt es zu zeigen, dass die Wahrscheinlichkeit, einen solchen Zeugen zu finden, für eine geeignete Verteilung über Eingabegraphen aus $\mathcal{N}$ gering ist.

Eine interessante Frage ist, ob auch vor dem Hintergrund dieser Argumentation deterministische Algorithmen auf einer Verteilung von Eingaben analysiert werden können, anstatt die randomisierten Algorithmen direkt zu analysieren. Dies lässt sich aus der ursprünglichen Formulierung von Yaos Minimax-Prinzip (siehe [71]) herleiten: Wir definieren dafür als Gedankenexperiment das Optimierungsproblem, in einem gegebenen Graphen Zeugen gegen die zu testende Eigenschaft zu finden; den zu maximierenden Wert X setzen wir auf 0, falls kein Zeuge gefunden wurde, und auf 1, falls mindestens ein Zeuge gefunden wurde. Wir zeigen dann, dass jeder deterministische Algorithmus mit q Anfragen auf einer zufällig gleichverteilt aus $\mathcal{N}$ gewählten Eingabe mit Wahrscheinlichkeit kleiner als $\frac{2}{3}$ einen Zeugen findet – damit ist der Erwartungswert von X ebenfalls kleiner als $\frac{2}{3}$. Das bedeutet nach Yaos Minimax-Prinzip, dass auch der Erwartetungswert von X für randomisierte Algorithmen mit q Anfragen im Worst-Case kleiner als $\frac{2}{3}$ ist; daher ist auch die Wahrscheinlichkeit, dass ein solcher Algorithmus einen Zeugen findet, kleiner als $\frac{2}{3}$. Also kann mit Wahrscheinlichkeit mehr als $\frac{1}{3}$ der einem solchen randomisierten Algorithmus bekannte Subgraph zu einem Graphen mit der entsprechenden Eigenschaft vervollständigt werden; der Algorithmus muss den Eingabegraphen in diesem Fall akzeptieren, obwohl er aus $\mathcal{N}$ ist. Nach Definition von Property-Testing-Algorithmen ist kein solcher Algorithmus ein Property-Testing-Algorithmus mit einseitigem Fehler.

Lemma 2.4. *Um eine untere Schranke q für die Anfragekomplexität jedes Property-Testing-Algorithmus mit einseitigem Fehler für eine bestimmte Grapheigenschaft Π herzuleiten, reicht es aus, eine Verteilung über eine Klasse $\mathcal{N}$ von Π ϵ-entfernter Graphen*

anzugeben, so dass kein deterministischer Algorithmus, der jeden Graphen aus Π *akzeptiert und nur* q *Anfragen an den Eingabegraphen stellt, auf einem nach der Verteilung zufällig gewählten Graphen mit Wahrscheinlichkeit mindestens* $\frac{2}{3}$ *einen Zeugen gegen* Π *findet.*

In beiden Fällen – für Schranken für Algorithmen mit ein- und mit zweiseitigem Fehler – bietet sich zudem folgende Vereinfachung der Analyse der deterministischen Algorithmen an: Es können den Klassen $\mathcal{P}$ und $\mathcal{N}$ möglicher Eingabegraphen jeweils alle Graphen hinzugefügt werden, die man durch Permutation der Knotennummern von Graphen dieser Klassen enthält; die deterministische Wahl eines Knotens aufgrund seiner Knotennmmer in einem zufällig gleichverteilt gewählten Eingabegraphen dieser Klassen ist dann gleichbedeutend mit der zufällig gleichverteilten Wahl eines Knotens in der „Grundversion" des jeweiligen Graphen – also dem Graphen mit derjenigen Knotenpermutation, die alle Knotennummern auf sich selbst abbildet. Wir werden dieses Argument in der Folge implizit benutzen, wenn wir argumentieren, dass ein Algorithmus zufällig gleichverteilt Knoten des Eingabegraphen sampelt.

Im Modell gradbeschränkter Graphen geben Goldreich und Ron erste Schranken für das Testen mit zweiseitigem Fehler an, nämlich für Bipartitheit und die Expander-Eigenschaft [43]; die grundsätzlich gleichen Techniken setzen auch Bender und Ron in [14] ein, um untere Schranken für zwei Probleme in gerichteten gradbeschränkten Graphen zu zeigen. Letztere Veröffentlichung ist insbesondere deswegen interessant für uns, weil den dort vorgestellten unteren Schranken für Kreisfreiheit und starken Zusammenhang das Property-Testing-Modell zugrunde liegt, das auch im Rahmen dieser Arbeit betrachtet werden soll. Deswegen werden wir hier detailliert auf diese beiden Schrankenkonstruktionen eingehen.

Bei beiden Konstruktionen geben Bender und Ron zwei Graphklassen $\mathcal{G}_1$ und $\mathcal{G}_2$ an, für die alle Graphen in $\mathcal{G}_1$ die zu testende Eigenschaft haben und alle oder zumindest fast alle Graphen in $\mathcal{G}_2$ ϵ-fern von ihr sind; die beiden Klassen enthalten für unendlich viele Werte für n je mindestens einen Graphen mit n Knoten und alle durch Permutationen der Knotennummern erzeugten dazu isomorphen Graphen.

Die Argumentation ist im Detail wie folgt: Sei $\mathcal{A}$ ein Algorithmus, der randomisiert s Kanten anfragt, um zu unterscheiden, ob der angefragte Graph G in $\mathcal{G}_1$ oder in $\mathcal{G}_2$ ist. Jede Anfrage besteht aus einem Knoten v und der Nummer $i \in \{1, ..., d\}$ der angefragten Kante (falls nur ausgehende Kanten angefragt werden können); die Rückgabe ist der Zielknoten der angefragten Kante.

Der angefragte Graph existiert jedoch nicht, sondern wird Schritt für Schritt konstruiert, nämlich entweder von einem Algorithmus $\mathcal{P}_1$, der Graphen aus $\mathcal{G}_1$ erzeugt, oder von einem Algorithmus $\mathcal{P}_2$, der Graphen aus $\mathcal{G}_2$ erzeugt. $\mathcal{P}_1$ und $\mathcal{P}_2$ fügen jeweils die Elemente randomisiert in G ein, die von $\mathcal{A}$ angefragt werden. Sei nun (v, i) eine Anfrage: Wurde v noch nicht angefragt, so wird v zunächst zufällig und gleichverteilt einem der noch freien „Knotenplätze" zugewiesen, die durch das Konstruktionsprinzip der jeweiligen Graphklassen $\mathcal{G}_1$ beziehungsweise $\mathcal{G}_2$ vorgegeben ist.

Danach muss der Zielknoten ermittelt werden: Dieser wird wieder zufällig ausgewählt unter den Knoten, die laut Konstruktionsprinzip der jeweiligen Graphklasse für eine Kan-

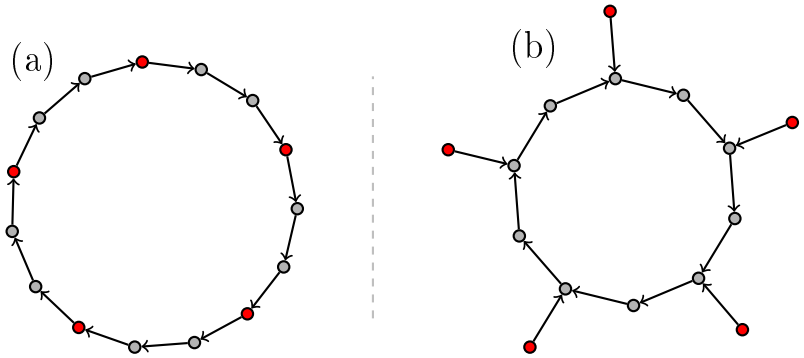

Abbildung 2.6: *Graphen der Klassen (a) $\mathcal{G}_1$ und (b) $\mathcal{G}_2$ für die untere Schranke für starken Zusammenhang; nach [14].*

te von v in Frage kommen; die Wahrscheinlichkeit, dass einer dieser Knoten gewählt wird, ist dabei proportional zu der Anzahl der noch nicht vergebenen Plätze seiner eingehenden Kanten. Es wird also sozusagen zufällig gleichverteilt ein noch freier Kantenplatz unter den Knoten ausgewählt, die eine Kante von v haben können. Werden durch Anfragen alle Knoten und Kanten von G erzeugt, dann erstellen $\mathcal{P}_1$ und $\mathcal{P}_2$ so zufällig gleichverteilt einen Graphen aus $\mathcal{G}_1$ beziehungsweise $\mathcal{G}_2$.

Wir interessieren uns nun für die Paare von Anfragen und Antworten, die entstehen, wenn $\mathcal{P}_1$ beziehungsweise $\mathcal{P}_2$ die Anfragen von $\mathcal{A}$ beantworten; die Idee ist, dass $\mathcal{A}$ nur bei unterschiedlichen Folgen unterschiedliche Ergebnisse zurückliefern kann. Die Wahrscheinlichkeit, dass $\mathcal{A}$ mit Zugriff auf $\mathcal{P}_1$ eine andere Antwort gibt als mit Zugriff auf $\mathcal{P}_2$, ist also durch die statistische Distanz der beiden Verteilungen von Anfrage-Antwort-Folgen beschränkt. Insbesondere bedeutet eine sehr kleine statistische Distanz, dass der Algorithmus $\mathcal{A}$, falls er mit Zugriff auf $\mathcal{P}_1$ mit Wahrscheinlichkeit $\frac{2}{3}$ akzeptiert, auch mit Zugriff auf $\mathcal{P}_2$ mit Wahrscheinlichkeit fast $\frac{2}{3}$ akzeptieren muss.

Da $\mathcal{P}_1$ und $\mathcal{P}_2$ zufällig gleichverteilt Graphen aus $\mathcal{G}_1$ beziehungsweise $\mathcal{G}_2$ konstruieren, folgt daraus, dass der Algorithmus $\mathcal{A}$, falls er jeden Graphen aus $\mathcal{G}_1$ mit Wahrscheinlichkeit $\frac{2}{3}$ akzeptiert, dies auch für jeden Graphen aus $\mathcal{G}_2$ mit fast ebenso hoher Wahrscheinlichkeit tut; damit hält $\mathcal{A}$ die für Property-Testing-Algorithmen geforderten Wahrscheinlichkeitsgarantien für das entsprechende Problem nicht ein, und nach Yaos Minimax-Prinzip existiert daher auch kein Property-Testing-Algorithmus für die Eigenschaft, der nur s Anfragen benötigt.

Dieses Beweismodell wenden Bender und Ron auf zwei Probleme an: Starken Zusammenhang und Kreisfreiheit. Die einfachere der beiden Schrankenkonstruktionen ist die für starken Zusammenhang. Hierfür werden die Graphklassen $\mathcal{G}_1$ und $\mathcal{G}_2$ wie folgt definiert: Für geeignet große n enthält $\mathcal{G}_1$ alle kreisförmigen Graphen mit n Knoten, bei denen die Kanten die gleiche Richtung haben. Natürlich sind diese Graphen stark zusammenhängend.

$\mathcal{G}_2$ enthält ebenfalls kreisförmige Graphen mit Kanten in gleicher Richtung, aber nur mit $n - \epsilon Dn - 1$ Knoten; die restlichen Knoten haben keine eingehende Kante, aber eine ausgehende Kante, deren Zielknoten einer der Kreisknoten ist. Jeder Kreisknoten

ist auf diese Weise höchstens mit einem Außenknoten verbunden (siehe Abbildung 2.6). Da für jeden der $\epsilon Dn + 1$ Außenknoten mindestens eine eingehende Kante hinzugefügt werden müsste, um ihn erreichbar zu machen, sind alle Graphen in $\mathcal{G}_2$ ϵ-fern von stark zusammenhängend.

Sei nun $\mathcal{A}$ ein Algorithmus, der maximal $\alpha\sqrt{n}$ Anfragen benötigt und seien $\mathcal{P}_1$ und $\mathcal{P}_2$ Prozesse, die wie oben beschrieben die Anfragen von $\mathcal{A}$ beantworten und dabei zufällig gleichverteilt einen Graphen aus $\mathcal{G}_1$ beziehungsweise $\mathcal{G}_2$ konstruieren; $\alpha < 1$ ist eine geeignete Konstante. Nach Beendigung von $\mathcal{A}$ haben diese beiden Prozesse Teilgraphen von Graphen aus $\mathcal{G}_1$ und $\mathcal{G}_2$ erstellt, die maximal $\alpha\sqrt{n}$ Kanten enthalten. Nach Konstruktion von $\mathcal{P}_1$ und $\mathcal{P}_2$ werden alle Anfragen von $\mathcal{A}$, die als Zielknoten einer Kante keinen schon zuvor erreichten oder angefragten Knoten enthalten, gleichverteilt unter den verbleibenden Knoten gewählt; falls $\mathcal{A}$ also in keiner Anfrage einen schon entdeckten Knoten erreicht, verhalten sich $\mathcal{P}_1$ und $\mathcal{P}_2$ gleich.

Es bleibt zu zeigen, dass die Wahrscheinlichkeit für das Erreichen eines schon entdeckten Knotens klein ist. $\mathcal{A}$ sind jederzeit höchstens $2\alpha\sqrt{n}$ Knoten als Start- oder Zielknoten von Kanten bekannt, und jeder davon besitzt höchstens zwei eingehende Kanten; beim Zuweisen eines Zielknotens für eine von $\mathcal{A}$ angefragte Kante ist daher sowohl für $\mathcal{P}_1$ als auch für $\mathcal{P}_2$ die Wahrscheinlichkeit, einen schon entdeckten Knoten zu wählen, höchstens $\frac{4\alpha\sqrt{n}}{n-2\alpha\sqrt{n}} \leq \frac{8\alpha}{\sqrt{n}}$. Die Wahrscheinlichkeit, dass dies in einer beliebigen der insgesamt $\alpha\sqrt{n}$ Anfragen geschieht, ist also aufgrund der Union-Bound höchstens $8\alpha^2$. Die statistische Distanz der durch $\mathcal{P}_1$ und $\mathcal{P}_2$ induzierten Verteilungen von Anfrage-Antwort-Folgen von $\mathcal{A}$ ist also höchstens $8\alpha^2$, und nach der obigen allgemeinen Argumentation ergibt sich, dass es keinen Property-Testing-Algorithmus für starken Zusammenhang gibt, der $o(n^{1/2})$ Anfragen benötigt.

Das zweite Problem, zu dem Bender und Ron eine untere Schranke geben, ist Kreisfreiheit. In beiden Graphklassen $\mathcal{G}_1$ und $\mathcal{G}_2$ sind die Knoten hier auf mehrere gleich große Layer aufgeteilt; die Layer haben eine Reihenfolge, und jeder Knoten hat ausschließlich Kanten zum jeweils nachfolgenden Layer (siehe Abbildung 2.7).

Die Graphen in $\mathcal{G}_1$ besitzen $n^{1/3}$ viele Layer $L_1, \ldots, L_{n^{1/3}} \subseteq V$, so dass jeder der Layer aus $n^{2/3}$ Knoten besteht. Die Kanten zwischen zwei Layern L_i und L_{i+1} werden durch d Matchings zwischen L_i und L_{i+1} definiert. Das stellt sicher, dass jeder Knoten in L_i d ausgehende Kanten hat und jeder in L_{i+1} d eingehende. Der erste Layer besitzt dabei keine eingehenden und der letzte keine ausgehenden Kanten. $\mathcal{G}_1$ besteht aus allen solchen Graphen mit n Knoten; diese sind offensichtlich kreisfrei.

Die Graphen in $\mathcal{G}_2$ haben nur zwei Layer S_1 und S_2, so dass jeder davon $n/2$ Knoten besitzt. Es gibt $Dn/2$ Kanten von S_1 nach S_2 und ebenfalls $Dn/2$ von S_2 nach S_1, analog zu $\mathcal{G}_1$ jeweils definiert durch d Matchings zwischen S_1 und S_2 beziehungsweise zwischen S_2 und S_1. Auch $\mathcal{G}_2$ besteht aus allen auf diese Weise konstruierbaren Graphen.

Dass die meisten Graphen in $\mathcal{G}_2$ ϵ-fern von kreisfrei sind, ist nicht offensichtlich, folgt aber aus der Dichte der die Kanten definierenden Matchings für geeignete ϵ und D: Für jeden kreisfreien gerichteten Graphen gibt es eine topologische Sortierung der Knoten, so dass kein Knoten eine Rückkante zu einem der vorhergehenden Knoten hat. Bender und Ron zeigen, dass ein zufällig gleichverteilt gewählter Graph aus $\mathcal{G}_1$ mit hoher Wahr-

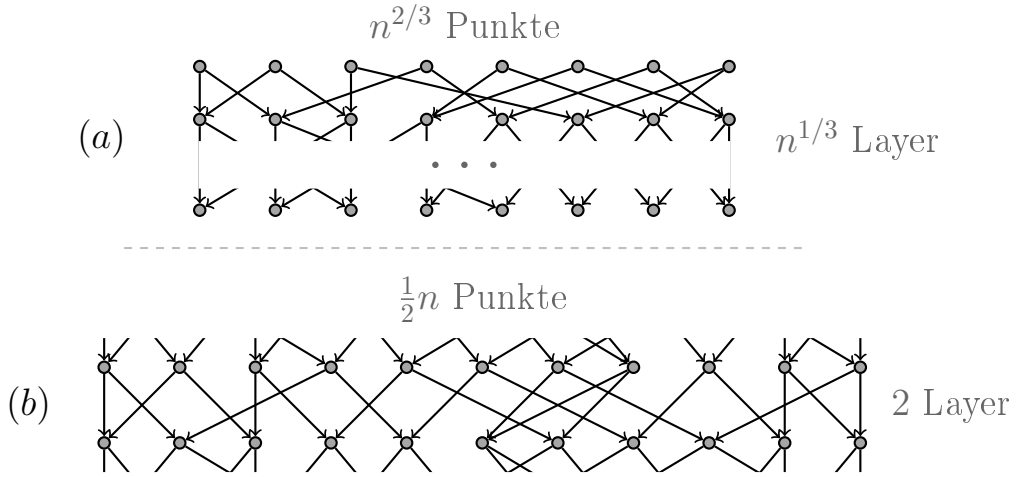

Abbildung 2.7: *Graphen der Klassen (a) $\mathcal{G}_1$ und (b) $\mathcal{G}_2$ für die untere Schranke für Kreisfreiheit; nach [14]*

scheinlichkeit für jede mögliche Ordnung der Knoten mehr als[9] ϵDn Rückkanten besitzt. Um Kreisfreiheit in einem Graphen herzustellen, müssen jedoch für eine der Ordnungen alle Rückkanten gelöscht werden, und damit ist ein zufällig gleichverteilt aus $\mathcal{G}$ gewählter Graph mit hoher Wahrscheinlichkeit ϵ-fern von kreisfrei.

Die Beweisidee ist wie folgt: Die Knotenmenge wird anhand der betrachteten Ordnung der Knoten derart in zwei gleich große Knotenmengen V^l und V^r partitioniert, dass V^l die $\frac{1}{2}n$ ersten Knoten der Ordnung enthält und V^r die $\frac{1}{2}n$ letzten.

Sei also (V^l, V^r) eine durch eine beliebige Ordnung von V induzierte Partitionierung von V und sei $V_i^l = V^l \cap S_i$ und $V_i^r = V^r \cap S_i$ für $i = 1, 2$ (siehe Abbildung 2.8). Es kann ohne Beschränkung der Allgemeinheit angenommen werden, dass $V_1^r \geq V_1^l$ gilt und damit auch $V_2^l \geq V_2^r$; V_1^r und V_2^l enthalten also jeweils mindestens $\frac{1}{4}n$ Knoten.

Sei $v \in V_1^l$ fest, aber beliebig. Sei G zufällig gleichverteilt gewählt unter den Graphen in $\mathcal{G}_2$, für die $v \in S_1$ gilt. Sei X eine Zufallsvariable für die Anzahl der Kanten von V_1^l nach V_2^r; es reicht zu zeigen, dass mit hoher Wahrscheinlichkeit X groß ist.

Da $|V_l^2| \geq \frac{1}{2}|S_2|$, ist für jede ausgehende Kante von v die Wahrscheinlichkeit, dass der Zielknoten in V_2^l ist, mindestens $\frac{1}{2}$ – die erwartete Anzahl von Kanten von V_1^l nach V_2^r ist also mindestens $E[X] \geq \frac{1}{2} \cdot |V_1^l| \cdot D \geq \frac{1}{8}Dn$. Die Ereignisse, dass für zwei beliebige von Knoten in V_1^l ausgehende Kanten die Zielknoten in V_2^r liegen, sind außerdem negativ korrelliert. Das heißt, X ist schärfer um seinen Erwartungswert konzentriert, als es bei Unabhängigkeit dieser Zufallsereignisse der Fall wäre. Die Wahrscheinlichkeit, dass X um mindestens $\frac{|V_1^l|}{4} \cdot D$ kleiner ist als sein Erwartungswert, lässt sich daher durch Anwendung einer Chernoff-Schranke für geeignetes D auf 2^{-2N} beschränken – das heißt, die

[9] Bender und Ron zeigen, dass es *mindestens* ϵDn Rückkanten gibt, aber der Beweis lässt sich leicht auf *mehr als* ϵDn Rückkanten erweitern.

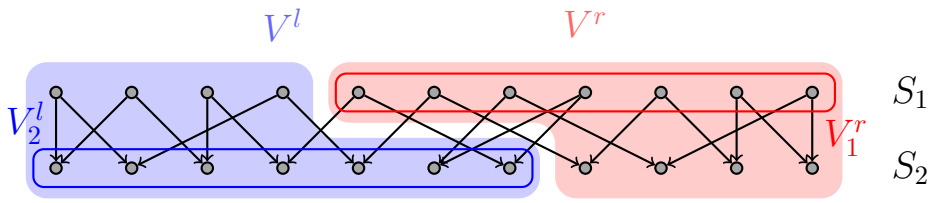

Abbildung 2.8: *Schematische Darstellung möglicher Partitionierungen V^l und V^r in Bezug auf die Punktlayer S_1 und S_2; die Kanten von S_2 nach S_1 sind ausgeblendet.*

Wahrscheinlichkeit, dass es mehr als $\mathrm{E}[X] - \frac{|V_1^l|}{4} \geq \frac{1}{16}Dn$ Kanten von V_1^r nach V_2^l gibt, ist mindestens $1 - 2^{-2n}$.

Also hat der gezogene Graph mit ebendieser Wahrscheinlichkeit bezüglich der betrachteten Knotenordnung mehr als $\frac{1}{16}Dn$ Rückkanten. Es gibt nur 2^n Möglichkeiten, Ordnungen so zu konstruieren, dass die resultierenden Partitionierungen V_r und V_l sich unterscheiden; daher ist die Wahrscheinlichkeit, dass für einen zufällig gezogenen Graphen eine der Orientierungen höchstens $\frac{1}{16}Dn$ Rückkanten hat, nach der Union-Bound höchstens $2^n \cdot 2^{-2n}$. Für $\epsilon \leq \frac{1}{16}$ folgt, dass ein zufällig gleichverteilt aus $\mathcal{G}_2$ gezogener Graph mit Wahrscheinlichkeit $1 - 2^{-n}$ ϵ-fern von kreisfrei ist.

Bender und Ron nehmen nun an, dass ein Algorithmus $\mathcal{A}$ höchstens $\alpha n^{1/3}$ Anfragen an P_1 beziehungsweise P_2 stellt. $\mathcal{P}_1$ und $\mathcal{P}_2$ sind dabei Prozesse, die nach der oben beschriebenen Vorgehensweise zufällig gleichverteilt Graphen aus $\mathcal{G}_1$ beziehungsweise $\mathcal{G}_2$ erstellen; $\alpha < 1$ ist eine geeignete Konstante. Dabei unterscheiden sich die Verteilungen der durch P_1 und P_2 bis zum Ende von $\mathcal{A}$ erstellten Teilgraphen nur in Anfragefolgen von $\mathcal{A}$, die (v, i) entweder für einen Knoten v ohne i-ten Nachfolger (bzw. Vorgänger) anfragen oder für die eine Anfrage einen schon in G eingefügten Knoten ergibt. Für letzteres Ereignis ist die Wahrscheinlichkeit höchstens α^2: Selbst wenn alle $t-1$ vorher angefragten Knoten dem gleichen Layer i zugewiesen wurden und der t-te Knoten dem Layer $i-1$, so ist die Wahrscheinlichkeit eines Treffers höchstens $\frac{t-1}{n^{2/3}}$ für P_1 und $\frac{t-1}{n/2} \leq \frac{t-1}{n^{2/3}}$ für P_2; Summieren über alle t und Auflösen mit der gaußschen Summenformel ergibt die Schranke von α^2.

Bender und Ron nehmen in der Folge an, dass keine solchen Kollisionen stattfinden. Die dadurch bedingte Wahrscheinlichkeit, dass durch Anfragen an $\mathcal{P}_1$ eine ausgehende Kante eines Knotens im letzten Layer angefragt wird, lässt sich auf α beschränken: Dazu kann angenommen werden, dass ein Algorithmus Knoten sampelt und von dort aus durch gezielte Anfragen versucht, möglichst viele Layer nach unten (oder oben) vorzustoßen, wobei die Gesamtzahl angefragter Kanten $\alpha n^{1/3}$ ist. Die Wahrscheinlichkeit, dass ein zufällig gleichverteilt gewählter Knoten sich in den letzten l der insgesamt K Layer befindet, ist $\frac{l}{K}$; das heißt, wenn der Algorithmus vom i-ten gesampelten Knoten v_i aus einen Pfad der Länge l_i exploriert, ist die Wahrscheinlichkeit, dass dabei der letzte Layer erreicht wird, $\frac{l_i}{K}$. Da $\sum_i l_i = \alpha n^{1/3}$ und $K = n^{1/3}$, folgt

$$\sum_i \Pr[\text{Pfad von } v_i \text{ erreicht letzten Layer}] = \sum_i \frac{l_i}{K} = \alpha.$$

Insgesamt ergibt dies eine Wahrscheinlichkeit von höchstens $\alpha^2 + \alpha$, dass $\mathcal{A}$ durch Anfragen auf $\mathcal{P}_1$ ein anderes Ergebnis erhält als durch Anfragen auf $\mathcal{P}_2$, und dies ist eine Schranke für die statistische Distanz zwischen den dadurch induzierten Verteilungen von Anfrage-Antwort-Folgen. Es folgt mit der oben gegebenen allgemeinen Begründung, dass es keinen Property-Testing-Algorithmus für Kreisfreiheit in diesem Modell gibt, der eine Anfragekomplexität von $o(n^{1/3})$ hat. Es sei noch erwähnt, dass diese Schranke auch gilt, wenn eingehende Kanten sichtbar sind; der Beweis ist analog.

Eine weitere für uns interessante Technik zur Konstruktion von unteren Schranken geben Raskhodnikova et al. [64]. Die oben vorgestellten Ergebnisse für starken Zusammenhang und Kreisfreiheit in gerichteten Graphen geben ebenso wie die erwähnten Ergebnisse für Bipartitheit und Expansion [41] in ungerichteten Graphen jeweils untere Schranken von maximal $\Omega(\sqrt{n})$. Dabei ist jeweils ein Argument, dass sich Verteilungen von Anfrage-Antwort-Folgen beim Zugriff auf unterschiedliche Graphklassen nur für diejenigen Folgen unterscheiden, in denen eine Kollision auftritt, das heißt für Folgen, in denen ein schon bekannter Knoten der Endknoten einer gesampelten Kante ist. Die Wahrscheinlichkeit dafür, dass ein solches Ereignis eintritt, ist nach dem Geburtstagsparadox hoch, wenn $\Theta(\sqrt{n})$ Samples gezogen werden; das bedeutet, dass für die Konstruktion einer unteren Schranke von $\omega(\sqrt{n})$ die Anfrage-Antwort-Folgen auch dann gleich sein müssen, wenn solche Kollisionen auftreten. Dies zu zeigen setzt erweiterte Techniken voraus.

Die Autoren von [64] zeigen eine untere Schranke für das Problem, die Anzahl unterschiedlicher Elemente in einem Array zu approximieren, ebenso wie für das damit verwandte Problem, die Kardinalität der Trägermenge einer Verteilung zu approximieren. Sie führen diese Probleme auf das Messen von Häufigkeiten gesampelter Elemente zurück und zeigen, dass jeder Approximationsalgorithmus für die beiden Probleme sich im Prinzip auf das Bewerten eines Histogramms gemessener Häufigkeiten reduzieren lässt. Die zugehörigen Zufallsvariable nennen sie *Frequenzvariable*, und sie ist definiert als die Häufigkeit, mit der ein zufällig gleichverteilt ausgewähltes Element in der Eingabe vorkommt.

Die Idee ist nun, dass die Verteilungen von Histogrammen, die durch das Sampeln von Elementen aus zwei unterschiedlichen Eingaben entstehen, nicht signifikant voneinander abweichen, wenn lediglich $o(n^{1-1/k})$ Samples gezogen werden und die zugehörigen Frequenzvariablen X und Y k proportionale Momente haben; das heißt, wenn $\frac{E[X]}{E[Y]} = \frac{E[X^i]}{E[Y^i]}$ für alle $i \in 1, \ldots, k$ gilt.

Dieser Umstand lässt sich für untere Schranken im Property-Testing von Graphen nutzen. Die Schwierigkeit ist dabei, die Graphklassen so zu wählen, dass die Häufigkeitsverteilung bestimmter Elemente das einzige Merkmal ist, anhand dessen sich die Graphen der beiden Klassen unterscheiden; schließlich modelliert ein Graph ein Beziehungsgeflecht, und das erlaubt im Allgemeinen das Messen einer Vielzahl von Parametern. Ein Beispiel, wie diese Technik trotzdem angewendet werden kann, und zwar auf Graphklassen, deren Graphen aus sehr kleinen isolierten schwachen Zusammenhangskomponenten bestehen, findet sich in Kapitel 3.3.

2.2.4 Techniken für Property-Testing in geometrischen Graphen

Da eins der Ergebnisse dieser Arbeit ein Property-Testing-Algorithmus für die euklidische Spannereigenschaft ist, soll an dieser Stelle kurz auf Techniken eingegangen werden, die für sublineare Algorithmen und insbesondere Property-Testing in geometrischen Graphen in der Literatur Verwendung finden. Allgemein ist die Anzahl der Ergebnisse für Property-Testing in geometrischen oder allgemein metrischen Graphen vergleichsweise gering; die meisten davon konzentrieren sich auf das Problem der minimalen Spannbäume.

Geometrische Graphen sind gegenüber einfachen Graphen dadurch gekennzeichnet, dass ihre Knoten zugleich Punkte im euklidischen Raum sind; durch diese doppelte Modellierung ergibt sich für ihre Kanten eine natürliche Gewichtsfunktion in Form der euklidischen Distanz zwischen den beteiligten Punkten. Geometrische Graphen sind für die Darstellung vielfältiger Sachverhalte geeignet: Beispielsweise lassen sich Versorgungs- und Straßennetze in guter Näherung als geometrische Graphen in der Ebene modellieren.

Auch bei geometrischen Graphen kann zwischen den Modellen dünn und dicht besetzter Graphen unterschieden werden: Für dicht besetzte geometrische Graphen nehmen wir auch hier an, dass sie als Adjazenzmatrix gespeichert sind, und ein dicht besetzter geometrische Graph ist ϵ-entfernt von einer Grapheigenschaft, wenn mehr als ϵn^2 Adjazenzmatrixeinträge des Graphen geändert werden müssen, um die Eigenschaft herzustellen. Dünn besetzte geometrische Graphen haben einen durch eine Konstante $D \in \mathbb{N}$ beschränkten Knotenausgangsgrad und wir nehmen an, dass sie in Adjazenzlistendarstellung gespeichert sind. Ein solcher Graph ist ϵ-fern von einer Eigenschaft Π, falls mehr als ϵDn Adjazenzlisteneinträge geändert werden müssten, um einen Graphen aus Π zu erhalten. Property-Testing-Algorithmen haben über Orakelfunktionen Zugriff auf die einzelnen Einträge der Adjazenzlisten bzw. Adjazenzmatrix und, über eine eigene Zugriffsfunktion, auf die Punktkoordinaten der Knoten.

Aus der Perspektive des Property-Testings drängt sich die Frage auf, in wieweit die zusätzlichen geometrischen Informationen beim Sampling und bei der Explorierung von Graphen hilfreich sind. In Bezug auf das Sampling hängt dies davon ab, ob Algorithmen auch geometrische Parameter für Samplinganfragen spezifizieren dürfen, zum Beispiel Regionen, in denen die gesampelten Knoten liegen müssen (siehe [28]). Die in Kapitel 5 vorgestellten Ergebnisse verzichten auf derartige Erweiterungen des Property-Testing-Modells und setzen das übliche zufällig gleichverteilte Sampling von Knoten ein. Das bedeutet, dass die geometrischen Informationen nur für die Explorierung des Graphen ausgehend von gesampelten Knoten genutzt werden können. Weiterhin ist auch für geometrische Graphen interessant, in wieweit Probleme in gerichteten Graphen schwerer sind als in ungerichteten.

Grundsätzlich bieten sich in geometrischen Graphen die gleichen Techniken an wie in den entsprechenden Modellen für ungewichtete Graphen üblich: In dicht besetzten Graphen verspricht der induzierte Subgraph einer gesampelten Knotenmenge Informationsgewinn, während in dünn besetzten Graphen lokale Explorationen interessant sind. Es konnte aber gezeigt werden, dass die geometrischen Informationen neue Herangehensweisen ermöglichen: So kann zum Beispiel der Approximationsalgorithmus für die Größe

eines metrischen minimalen Spannbaums von Czumaj und Sohler [27] eine der Breitensuche ähnliche Technik in dicht besetzten Graphen einsetzen, indem er eine Schwelle für die maximale zurückgelegte Distanz sukzessive um den Faktor $1 + \epsilon$ vergrößert und in jedem Schritt nur diejenigen Kanten liest, deren Länge den aktuellen Schwellwert nicht übersteigt – die (geo-)metrischen Informationen erlauben damit eine sinnvolle Auswahl der zu traversierenden Kanten.

Die zugrundeliegende Technik, die mit Hilfe dieses Tricks auf dicht besetzte Graphen übertragen wurde, stammt von Chazelle et al. [23]: Deren Algorithmus für das Approximieren der Größe des minimalen Spannbaums in dünn besetzten gewichteten, aber nicht notwendigerweise (geo-)metrischen Graphen schätzt sukzessive für wachsendes i die Anzahl c_i der Zusammenhangskomponenten in einem Hilfsgraphen $G^{(i)}$, die nur aus den Kanten mit Länge maximal i besteht; den zugehörigen Algorithmus für die Schätzung der Anzahl der Zusammenhangskomponenten eines Graphen haben wir schon oben besprochen. Aus diesen Schätzungen lässt sich auch ein Schätzer für die Größe des minimalen Spannbaums des Ausgangsgraphen G berechnen, denn diese ist $n - w + \sum_{1 \leq i \leq w-1} c_i$, wenn w die maximale Länge einer Kante in G ist und alle Kanten positive ganzzahlige Gewichte haben. Durch Multiplizieren der Gewichte mit $\frac{1}{\epsilon}$ und kaufmännisches Runden auf die nächste ganze Zahl liefert der Algorithmus auch für nicht ganzzahlige Kantenlängen eine gute Approximation, wie Chazelle et al. zeigen.

Im Kontext des Property-Testing wurde das Problem der minimalen Spannbäume ebenfalls erforscht: Czumaj und Sohler [29] haben gezeigt, dass sich für einen gegebenen geometrischen Graphen G in der Ebene mit $\mathcal{O}(\sqrt{n/\epsilon}\log(n/\epsilon))$ Anfragen testen lässt, ob er der minimale Spannbaum der zugrunde liegenden Punktmenge ist; der Graph wird als im „allgemeinen“ Graphmodell gespeichert angenommen, das oben schon kurz erwähnt wurde: Die Kanten sind in Adjazenzlisten gespeichert, es gibt aber keine Schranke für den Grad der Knoten. Wir werden allerdings sogleich sehen, dass aufgrund der Problemstruktur für weite Teile des Algorithmus eine solche Gradschranke angenommen werden kann. Da ein minimaler Spannbaum genau $n - 1$ Kanten hat, wird ein Graph als ϵ-fern vom minimalen Spannbaum[10] betrachtet, wenn mindestens ϵn Änderungen der Adjazenzlisten nötig sind, damit der resultierende Graph ein minimaler Spannbaum ist. Zusätzlich zu der Standardform des allgemeinen Graphmodells hat ein Algorithmus Orakelzugriff auf die Punktkoordinaten der Knoten. Wenn im Folgenden von Lagebeziehungen von Kanten die Rede ist, bezieht sich das immer auf diejenige Einbettung des Graphen in die Ebene, bei der die Kanten durch gerade Strecken zwischen den beteiligten Punkten repräsentiert werden. Ein wichtiges Konzept der Analyse des Algorithmus ist die MST-Vervollständigung eines geometrischen Graphen G: Das ist derjenige Graph, der zusätzlich zu den Kanten von G noch alle diejenigen Kanten enthält, die im minimalen Spannbaum der zugrundeliegenden Punktmenge enthalten sind, aber nicht in G.

Der Algorithmus von Czumaj und Sohler arbeitet zweistufig (siehe Algorithmus 2.7). In der ersten Stufe werden einige geometrische Eigenschaften geprüft, die jeder minimale Spannbaum in der euklidischen Ebene haben muss: der Graph ist zusammenhängend,

[10] Czumaj und Sohler gehen davon aus, dass die dem Eingabegraphen zugrundeliegende Punktmenge in allgemeiner Lage vorliegt; der minimale Spannbaum der Punktmenge ist also eindeutig.

TesteEuklidischenMST$(G = (P, E), \epsilon)$
  Teste, ob G wohlgeformt ist oder $\frac{1}{4}\epsilon$-entfernt von einem der Kriterien dafür
  If Zeuge gegen die Wohlgeformtheit von G gefunden **then return false**
  Sampele zufällig gleichverteilt $\mathcal{O}(\sqrt{n/\epsilon} + 1/\epsilon)$ Knoten von G
  Foreach gesampelter Knoten v
    Foreach Kante $\{v, u\} \in E$
      Traversiere von u aus nach der Linke-Hand-Regel die nächsten
        $\Theta(1/\epsilon)$ Kanten
      Traversiere von u aus nach der Rechte-Hand-Regel die nächsten
        $\Theta(1/\epsilon)$ Kanten
  $\tilde{G} \leftarrow$ MST-Vervollständigung des explorierten Teilgraphen von G
  If $\tilde{G}$ ist kreisfrei **then return true**
  else return false

Algorithmus 2.7: TesteEuklidischenMST*; Nach [29].*

jeder Knoten hat einen Grad von höchstens fünf und die MST-Vervollständigung des Graphen hat keine sich kreuzenden Kanten. Ein Graph, der weit entfernt von einer dieser Eigenschaften ist, ist auch weit entfernt vom minimalen Spannbaum. Einen Graphen, der alle diese Eigenschaften hat, nennen die Autoren *wohlgeformt*[11], und das Ziel der Testens der Eigenschaften ist, dass der Algorithmus im weiteren Verlauf davon ausgehen kann, dass der betrachtete Graph mindestens nahezu wohlgeformt ist.

Der Eingabegraph wird zunächst einzeln auf diese drei Eigenschaften getestet, in der folgenden Reihenfolge: Zunächst wird getestet, ob jeder Knoten maximal Grad fünf hat; das kann mit einem Knotensample der Größe $\Theta(\sqrt{n/\epsilon})$ und einfache Überprüfung der gesampelten Knoten geschehen. Wie Zusammenhang getestet werden kann, haben wir schon oben gesehen. Für das Testen sich kreuzender Kanten in G reicht es, ein Kantensample der Größe $\mathcal{O}(\sqrt{n/\epsilon})$ zu ziehen: dieses enthält mit hoher Wahrscheinlichkeit ein Paar sich kreuzender Kanten, falls der Graph ϵ-entfernt von kreuzungsfrei ist; das folgt aus dem Geburtstagsparadoxon verwandten Argumenten. Schwieriger ist das Identifizieren von Kreuzungen zwischen Kanten von G und zusätzlichen Kanten in der MST-Vervollständigung von G: Letztere können nicht gesampelt werden, da die zusätzlichen Kanten der MST-Vervollständigung nur implizit vorhanden sind. Allerdings lässt sich zeigen, dass solche Kreuzungen kleine Kreise in der MST-Vervollständigung von G implizieren, die sich durch geschicktes Sampling identifizieren lassen.

Alle diese Tests lehnen mit hoher Wahrscheinlichkeit ab, wenn der Graph $\frac{1}{8}\epsilon$-entfernt von der jeweiligen Eigenschaft ist[12]; ein vom minimalen Spannbaum ϵ-entfernter Graph $G = (P, E)$, der nicht mit hoher Wahrscheinlichkeit bei einem der drei Tests abgelehnt wird, wäre also immer noch $\frac{5}{8}\epsilon$-entfernt vom minimalen Spannbaum, wenn alle zur Her-

[11] englisch *well-shaped*

[12] In [29] wird jeweils auf $\frac{1}{4}\epsilon$-Entferntheit getestet; damit wäre dann aber im weiteren Verlauf nicht garantiert, dass die MST-Vervollständigung eines zusammenhängend $\frac{1}{4}\epsilon$-nahen Graphen viele Kreise enthält.

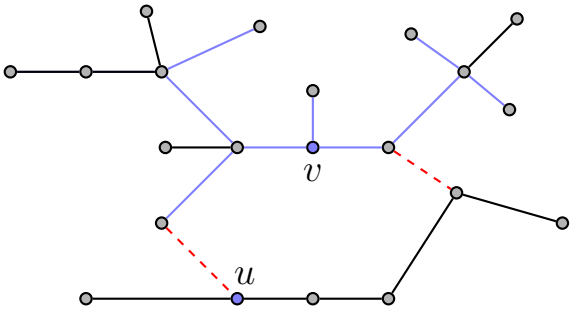

Abbildung 2.9: *Property-Testing-Algorithmus für den minimalen Spannbaum in der euklidischen Ebene: Ausschnitt des Eingabegraphen mit den von dem Knoten v aus erforschten Kanten (blau) und zusätzlichen Kanten der MST-Vervollständigung (rot). Würde zusätzlich noch u gesampelt und exploriert, dann enthielte die MST-Vervollständigung des induzierten Subgraphen der explorierten Knoten einen Kreis.*

stellung der drei Eigenschaften nötigen Kantenmodifikationen durchgeführt würden. Fügt man also in G alle Kanten des minimalen Spannbaums von P ein, die noch nicht enthalten sind, dann hat der entstehende Graph $\Theta(\epsilon n)$ Kanten mehr als G und enthält, da G (fast) zusammenhängend ist, Kreise.

Sei $G = (P, E)$ nun ein wohlgeformter Graph, der $\Theta(\epsilon)$-entfernt vom minimalen Spannbaum ist, und sei G' seine MST-Vervollständigung. Czumaj und Sohler zeigen, dass jede Kante, die in G' enthalten ist, auch in der MST-Vervollständigung jedes Teilgraphen von G enthalten ist, der die entsprechenden Knoten enthält; da nach den obigen Ausführungen G' Kreise enthält, enthalten auch die MST-Vervollständigungen von durch geeignete Teilmengen $S \subseteq P$ aufgespannten Subgraphen von G Kreise. Außerdem folgt daraus, dass die MST-Vervollständigung eines Teilgraphen von G die gleichen Kanten wie die MST-Vervollständigung von G enthält, letztere lässt sich also lokal konstruieren.

Das Ziel des Algorithmus ist nun, einen Teilgraphen von G derart zu explorieren, dass dessen MST-Vervollständigung einen Kreis enthält. Ein Kreis in G' definiert eine Innenfläche von G', wenn G' kreuzungsfrei ist, und die Idee von Czumaj und Sohler ist, dass man mit Hilfe der „Linke-Hand-Regel“ beziehungsweise der „Rechte-Hand-Regel“ entlang der Kanten einer Innenfläche traversieren kann: Dazu wählt man bei Erreichen eines Knotens die im Uhrzeigersinn folgende beziehungsweise die im Uhrzeigersinn vorhergehende Kante der soeben benutzten – hier werden die geometrischen Informationen algorithmisch genutzt. Solche Traversionen werden für alle inzidenten Kanten der gesampelten Knoten durchgeführt.

Eine zweite wichtige Beobachtung ist, dass zwei solche Explorationen für eine Innenfläche von G' ausreichen, falls ihr Rand höchstens zwei Kanten enthält, die in G fehlen: In diesem Fall zerfällt der die Innenfläche definierende Kreis in G in maximal zwei Teile, und wenn aus jedem der Teile ein Knoten bekannt ist, dann können nach der obigen Methode alle Kanten der Innenfläche bis auf die beiden fehlenden erforscht werden; die MST-Vervollständigung des explorierten Bereichs und damit die beiden fehlenden Kreiskanten lassen sich dann lokal berechnen, und dadurch wird der Kreis identifiziert. Dafür

muss nur aus jedem der beiden Einzelteile des Kreises in G ein Knoten gesampelt werden und von diesen Knoten aus die oben genannte Explorationstechnik mit hinreichender Tiefe durchgeführt werden.

Wenn es also $\Omega(\epsilon n)$ solche Kreise in G' gibt, die zudem einen Umfang von jeweils höchstens $\mathcal{O}(\frac{1}{\epsilon})$ Kanten haben, dann müssen nach dem Geburtstagsparadoxon höchstens $\mathcal{O}(\sqrt{n/\epsilon})$ Knoten gesampelt werden, um einen der Kreise vollständig explorieren zu können; die für die Explorationen nötige Tiefe ist dann $\mathcal{O}(\frac{1}{\epsilon})$. Dass es in G' viele Kreise gibt, folgt mit Hilfe der Eulerformel für die Anzahl der Kanten eines Graphen in Abhängigkeit von der Knoten- und Innenflächenzahl – die Kantenzahl ist, wie oben erwähnt, $n+\Theta(\epsilon n)$. Dass viele der Kreise klein sind und höchstens zwei in G nicht enthaltene Kanten haben, folgt aus Abzählargumenten. Damit[13] lehnt der Algorithmus TesteEuklidischenMST einen vom minimalen Spannbaum $\Theta(\epsilon)$-entfernten Graphen mit hoher Wahrscheinlichkeit ab. Falls der übergebene Graph G der minimale Spannbaum der zugrundeliegenden Punktmenge ist, findet andererseits keiner der Tests einen Zeugen gegen die Eigenschaft: Damit ist der Algorithmus ein Propery-Testing-Algorithmus mit einseitigem Fehler.

Für TesteEuklidischenMST kann übrigens eine bessere Abhängigkeit der Anfragekomplexität von $1/\epsilon$ erreicht werden, wenn die oben für das Testen von starkem Zusammenhang vorgestellte Technik zum zufälligen Beschränken der Größe explorierter Bereiche des Graphen angewendet wird.

Auch auf die Bestimmung von unteren Schranken haben zusätzliche geometrische Informationen Einfluss, denn es muss ausgeschlossen werden, dass ein Property-Testing-Algorithmus diese zum Brechen der angestrebten Schranke der Anfragekomplexität einsetzen kann; außerdem müssen natürlich auch die geometrischen Eigenschaften des Problems in der Anordnung der Punkte der Graphen der beiden Klassen berücksichtigt werden. Ein Beispiel ist die untere Schranke von $\Omega(n^{1/3})$ für die Anfragekomplexität jedes Property-Testing-Algorithmus für minimale Spannbäume, die Ben-Zwi et al. angeben [16]: Für die Graphen beider Klassen sind die Knoten dort auf einem $4 \times m$-Gitter angeordnet, wobei der horizontale Abstand zwischen den Gitterpunkten 3 ist und der vertikale Abstand 2. Jeder mögliche Knotenplatz ist gegenüber dem entsprechenden Gitterplatz außerdem um einen sehr kleinen zufälligen Wert in x-und y-Richtung verschoben, um die Punkte in allgemeine Lage zu bringen. In jeder der m Spalten des Gitters sind genau zwei Knotenplätze besetzt, und zwar durch einen Knoten auf einem der oberen beiden Plätze und einen weiteren Knoten auf einem der unteren beiden Plätze; insgesamt haben die Graphen dadurch $n = 2m$ Knoten. Die Knoten jeder der Reihen sind durch Kanten zum jeweiligen Vorgänger- und Nachfolgerknoten verbunden; zwischen den beiden Reihen gibt es nur eine einzige Verbindung (siehe Abbildung 2.10).

Die Klasse $\mathcal{P}$ minimaler Spannbäume besteht nun aus allen Graphen, in denen die Knoten in (fast) jeder Spalte einen Abstand von genau zwei Gitterpunkten haben, das heißt falls der obere Knoten in der ersten Reihe ist, ist der untere Knoten in der dritten Reihe und falls der obere Knoten in der zweiten Reihe ist, ist der untere Knoten in der vierten Reihe. Nur in einer einzelnen Spalte ist der obere Knoten auf der zweiten Position

[13] Czumaj und Sohler geben außerdem noch eine saubere Argumentation an für den Fall von Graphen G, die nicht wohlgeformt sind, aber ϵ-nah zu wohlgeformt.

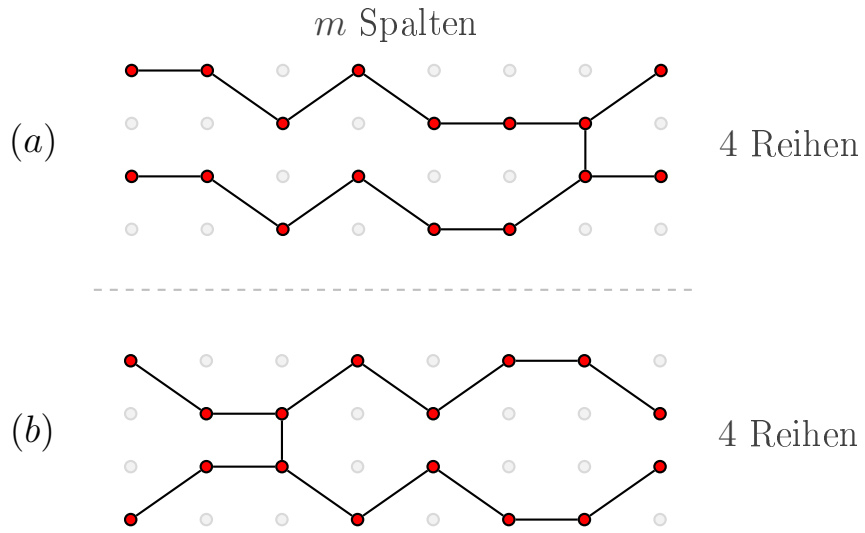

Abbildung 2.10: *Schematische Darstellung der Graphklassen (a) $\mathcal{P}$ und (b) $\mathcal{N}$ für die untere Schranke für das Testen von minimalen Spannbäumen in der euklidischen Ebene; nach [16].*

angeordnet und der untere Knoten auf der dritten. Diese beiden Knoten sind verbunden. Die Graphen dieser Klasse bestehen aus minimalen Spannbäumen: Die vertikale Distanz zwischen den beiden Knoten einer Spalte ist immer größer als die horizontale Distanz zweier benachbarter Knoten einer Reihe. Also bringt das Ersetzen einer horizontalen Kante durch eine vertikale keinen Gewinn.

Die Klasse $\mathcal{N}$ der von minimalen Spannbäumen weit entfernten Graphen besteht aus allen Graphen, bei denen die Knoten jeder Spalte einen Abstand von entweder einem oder drei Gitterpunkten haben, das heißt entweder an der ersten und vierten Position oder an der zweiten und dritten Position angeordnet sind. Auch hier wird das Knotenpaar einer zufälligen Spalte an der zweiten und dritten Position angeordnet und verbunden. Ein zufällig gleichverteilt gewählter Graph aus $\mathcal{N}$ ist, für geeignetes ϵ, ϵ-entfernt von jedem minimalen Spannbaum: Ben-Zwi et al. zeigen, dass es mit hoher Wahrscheinlichkeit mehr als ϵn Spalten gibt, für die eine kurze horizontale Kante der Länge ungefähr eins eingefügt und dafür an anderer Stelle eine lange vertikale Kante gelöscht werden kann.

Die beiden Klassen können von einem Algorithmus nur dann unterschieden werden, wenn er aus einer beliebigen Spalte außer der Verbindungsspalte beide Knoten kennt. Ist dies der Fall, dann kann er akzeptieren, wenn der Abstand der Knoten ungefähr 2 ist und ablehnen, wenn der Abstand ungefähr 1 oder 3 ist. Da die Knotennumerierung beliebig ist, kann der Algorithmus dies aber nicht durch gezielte Anfragen erreichen. Wenn der Property-Testing-Algorithmus nicht adaptiv ist, also insbesondere nicht aufgrund der geometrischen Informationen angepasste Explorationen vornimmt, dann benötigt er nach dem Geburtstagsparadoxon $\Omega(\sqrt{n})$ Anfragen, um mit hoher Wahrscheinlichkeit für eine Spalte beide Knoten zu sampeln; formal lässt sich das mit ähnlichen Argumenten zeigen wie für die oben gesehene untere Schranke für Kreisfreiheit in gerichteten Graphen von

Bender und Ron [14].

Interessanter an dieser Stelle sind adaptive Algorithmen, denn sie können sich die geometrischen Informationen der gesampelten Punkte zunutze machen; zum Beispiel könnte ein Algorithmus die beiden geometrisch nächsten gesampelten Punkte aus verschiedenen Reihen wählen und versuchen, durch lokale Exploration von einem der Punkte in Richtung des anderen dessen Spalte zu erreichen. Intuitiv gesprochen zeigen Ben-Zwi et al. hier, dass $q' = \Theta(\frac{m}{q^2}) = \Theta(\frac{n}{q^2})$ Punkte angefragt werden müssen, damit mit hoher Wahrscheinlichkeit höchstens q Spalten zwischen den nächsten beiden Punkten liegen, und nur dann kann ein adaptiver Algorithmus, der höchstens q Anfragen vornimmt, durch eine lokale Exploration die Spalte des nächsten Punktes erreichen, oder von beiden nahen Punkten Explorationen starten, die sich in einer Spalte treffen. Da der Algorithmus aber nur q Anfragen hat, muss $q' \leq q$ gelten und damit $q \geq \Theta(\frac{n}{q^2}) \Leftrightarrow q^3 \geq \Theta(n)$. Also sind mindestens $\Omega(n^{1/3})$ Anfragen für jeden adaptiven Tester für das Testen von euklidischen minimalen Spannbäumen in der Ebene erforderlich. Die Schranke ist für Algorithmen, die geometrische Informationen für ihre Anfragen nutzen, schwächer als für nicht adaptive Algorithmen: Das zeigt, dass untere Schranken für Property-Testing in geometrischen Graphen schwieriger zu konstruieren seien können als für Property-Testing für reine Graphprobleme.

2.3 Sampling

In den Ergebnissen der folgenden Kapitel wird Sampling auf eine von zwei Arten durchgeführt: Entweder es wird eine feste Anzahl s von Objekten – meist Knoten oder Kanten der Eingabegraphen – zufällig gleichverteilt gezogen, oder es wird jedes Objekt – also jeder Knoten oder jede Kante des Eingabegraphen – mit einer bestimmten Wahrscheinlichkeit $p < 1$ gezogen.

Die erste Samplingmethode ist in Graphen einfach umzusetzen, wenn man annimmt, dass ein hinreichend guter Zufallszahlengenerator zur Verfügung steht: Die Knoten eines Graphen $G = (V, E)$ werden als durchnummeriert von 1 bis $|V|$ angenommen, und das Sampeln eines Knotens besteht dann aus dem zufälligen Wählen einer Zahl zwischen 1 bis $|V|$. Das entspricht einem Ziehen eines Knotens *mit Zurücklegen*: Werden mehrere Knoten gezogen, so kann ein Knoten mehrmals im Sample vorhanden sein. Wir nehmen dabei an, dass das Generieren einer Zufallszahl mit $\lceil \log |V| \rceil$ Zufallsbits in konstanter Zeit möglich ist.

All dies gilt auch für das Sampeln von Kanten, außer dass dafür zwei Zufallszahlen generiert werden müssen: Eine von 1 bis $|V|$, um den Ausgangsknoten der Kante zu wählen, und eine von 1 bis zur Gradschranke D für den Adjazenzlistenplatz. Wird dabei eine leerer Adjazenzlistenplatz gesampelt, dann wird das Sample verworfen und mit einem neuen Knoten und Adjazenzlistenplatz wiederholt.

Es ist auch möglich, Elemente *ohne Zurücklegen* zu sampeln: Dabei muss für jedes gesampelte Element geprüft werden, ob es schon in der Menge der gesampelten Elemente vorhanden ist und, wenn dies zutrifft, das Sample neu gezogen werden. Für Samples sublinearer Größe – darunter fallen alle Samples von Algorithmen dieser Arbeit – ist die

Wahrscheinlichkeit, dass dabei viele Samples verworfen werden müssen, $o(1)$, denn in jedem Sampelschritt ist die Anzahl zuvor gezogener Elemente $o(n)$ und damit die Wahrscheinlichkeit, dass das Sample verworfen werden muss, $\frac{o(n)}{n} = o(1)$. Bei Verwendung geeigneter Datenstrukturen für das Prüfen schon gesampelter Elemente ist die Laufzeit, die für die Sampelprozedur benötigt wird, nur mit Wahrscheinlichkeit $o(1)$ größer als $\mathcal{O}(s)$, wenn s die Samplegröße ist. Deswegen werden wir im Folgenden davon ausgehen, dass $s = o(n)$ Elemente aus einer Grundmenge von n Elementen mit Zurücklegen gesampelt werden können.

Aufwändiger umzusetzen ist die zweite Samplingmethode, nämlich jedes der n Elemente unabhängig voneinander mit einer gegebenen Wahrscheinlichkeit p in die Stichprobe aufzunehmen. Die Idee dieser Methode ist es, Korrelationen aus dem Sample zu entfernen: Bei Verwendung der ersten Methode sind die Ereignisse, dass zwei Knoten u und v gezogen werden, negativ korreliert, das heißt es gilt $\Pr[u \text{ wird gezogen} | v \text{ wird gezogen}] < \Pr[u \text{ wird gezogen}]$. Allerdings ist es nicht wünschenswert, in einem sublinearen Algorithmus für eine lineare Anzahl von Elementen einen Zufallswurf durchzuführen, um für jedes Element einzeln zu entscheiden, ob es in das Sample aufgenommen wird. Das Verfahren lässt sich jedoch durch einfaches Ziehen ohne Zurücklegen simulieren, wenn die Anzahl der zu ziehenden Elemente nicht fest ist, sondern nach der Binomialverteilung mit Parametern n und p zufällig gewählt wird[14].

Dadurch ist sowohl gewährleistet, dass jedes Element mit Wahrscheinlichkeit p im Sample vorhanden ist, als auch dass dies für jedes Element unabhängig voneinander der Fall ist: Sei dazu X_v das Ereignis, dass ein Element v durch das Verfahren gezogen wird und sei Y eine mit Parametern n und p binomialverteilte Zufallsvariable für die Anzahl der gezogenen Elemente. Es gilt $\Pr[X_v | Y = i] = \frac{i}{n}$, da insgesamt i unterschiedliche Elemente gezogen werden und jedes der n Elemente die gleiche Wahrscheinlichkeit hat, im Sample enthalten zu sein. Es folgt also nach dem Satz der totalen Wahrscheinlichkeit

$$\begin{aligned}
\Pr[X_v] &= \sum_{1 \le i \le n} \Pr[X_v | Y = i] \cdot \Pr[X = i] = \sum_{1 \le i \le n} \frac{i}{n} \cdot \mathrm{Bi}(n,p)_i \\
&= \sum_{1 \le i \le n} \frac{i}{n} \cdot \binom{n}{i} p^i (1-p)^{n-i} = \sum_{1 \le i \le n} \frac{i}{n} \cdot \frac{n!}{(n-i)! \cdot i!} \cdot p^i (1-p)^{n-i} \\
&= p \cdot \sum_{1 \le i \le n} \frac{(n-1)!}{((n-1)-(i-1))! \cdot (i-1)!} \cdot p^{i-1} (1-p)^{(n-1)-(i-1)} \\
&= p \cdot \sum_{1 \le i \le n} \mathrm{Bi}(n-1,p)_{i-1} = p \cdot \sum_{0 \le j \le n-1} \mathrm{Bi}(n-1,p)_j = p,
\end{aligned}$$

[14] Batu et al. [11] merken an, dass dies in ähnlicher Form mit Hilfe der Poissonverteilung möglich ist, wenn danach die Elemente mit Zurücklegen gezogen werden; solche Algorithmen nennen Raskhodnikova et al. [64] *Poisson-Algorithmen*, beziehungsweise *Poisson-s-Algorithmen*, wenn die erwartete Zahl der Samples s ist. Da mit diesem Verfahren mehrfach gezogene Elemente möglich sind, ist die Wahrscheinlichkeit, mindestens einmal gezogen zu werden, für jedes Einzelelement kleiner als p; somit ist das Verfahren für unsere Zielsetzung nicht direkt geeignet. Batu et al. geben zudem weder eine Analyse noch eine Quelle für eine solche an.

da die Summe aller Wahrscheinlichkeiten der Biomialverteilung 1 ist. Weiter ist für zwei beliebige unterschiedliche Elemente u und v die Wahrscheinlichkeit, dass für fixes $Y = i$ beide Elemente gezogen werden, hypergeometrisch verteilt, das heißt es gilt

$$\begin{aligned}\Pr[X_v|X_u] = \frac{\Pr[X_v \cap X_u]}{\Pr[X_u]} = \frac{1}{p} \cdot \Pr[X_v \cap X_u] = \frac{1}{p} \cdot \sum_{2 \leq i \leq n} \Pr[X_v \cap X_u | Y = i] \cdot \Pr[Y = i] \\ = \frac{1}{p} \cdot \sum_{2 \leq i \leq n} \frac{\binom{2}{2}\binom{n-2}{i-2}}{\binom{n}{i}} \cdot \binom{n}{i} p^i (1-p)^{n-i} = \frac{p^2}{p} \cdot \sum_{2 \leq i \leq n} \binom{n-2}{i-2} p^{i-2} (1-p)^{n-i} \\ = p \cdot \sum_{0 \leq j \leq n-2} \mathrm{Bi}(n-2, p)_j = p\end{aligned}$$

und damit $\Pr[X_v] = p = \Pr[X_v|X_u]$; also sind die Ereignisse X_v und X_u unabhängig voneinander. Das Beweisprinzip lässt sich induktiv weiter anwenden, um zu zeigen, dass n-fache Unabhängigkeit der Sampelereignisse gilt.

Dieses Verfahren ist also äquivalent zu der Methode, jeden Knoten mit Wahrscheinlichkeit $p = \frac{s}{n}$ zu ziehen; hier ist die benötigte Laufzeit jedoch definiert durch das zufällige Wählen einer binomialverteilten Zufallsgröße Y mit Parametern n und p und Erwartungswert $s = np$ und das zufällig gleichverteilte Ziehen von Y Objekten.

Es bleibt zu diskutieren, wie verfahren wird, wenn Y sehr groß ist, denn eine Samplegröße von $\mathcal{O}(s)$ ist dann nicht mehr gegeben. Falls Y also um einen konstanten Faktor a größer ist als s, terminieren die in dieser Arbeit vorgestellten Algorithmen mit einem beliebigen Rückgabewert. Nach der Markowungleichung geschieht dies höchstens mit Wahrscheinlichkeit $\frac{1}{a}$, die auf die Fehlerwahrscheinlichkeit des Algorithmus addiert wird – dies ist für jeden der hier vorgestellten Algorithmen, der so verfährt, einzeln angegeben.

Dass bei diesem Vorgehen trotzdem nach der Sampleprozedur davon ausgegangen werden kann, dass jeder Knoten unabhängig mit Wahrscheinlichkeit p gezogen wurde, zeigt folgendes Gedankenexperiment: Sei $\mathcal{A}$ ein entsprechender Samplealgorithmus ohne das Abbruchkriterium, sei $\mathcal{A}_1$ der gleiche Algorithmus, aber mit dem zusätzlichen Kriterium, dass er die Eingabe bei $Y > as$ ablehnt, bevor eine weitere Verarbeitung geschieht. Sei $\mathcal{A}_2$ ein Algorithmus, der $\mathcal{A}$ aufruft und nach Beendigung von $\mathcal{A}$ prüft, wie viele Samples gezogen wurden: Wurden mehr als as Samples gezogen, dann lehnt $\mathcal{A}_2$ ab, ansonsten gibt $\mathcal{A}_2$ die Antwort von $\mathcal{A}$ zurück.

Es ist klar, dass $\mathcal{A}_1$ und $\mathcal{A}_2$ für gleiche Samples das gleiche Ergebnis zurückliefern; andererseits kann bei der Verarbeitung in $\mathcal{A}$ vorausgesetzt werden, dass jedes Objekt unabhängig mit Wahrscheinlichkeit p gesampelt wurde, auch wenn $\mathcal{A}$ von $\mathcal{A}_2$ aufgerufen wurde. Also kann dies auch für die Verarbeitung in $\mathcal{A}_1$ angenommen werden.

3 Testen von Subgraphfreiheit

Ein sehr grundlegendes Graphproblem ist das der Subgraphfreiheit. Seien G und H Graphen; dann ist G *H-frei*, falls H kein Subgraph von G ist.

Wir werden in diesem Kapitel zunächst einen einfachen allgemeinen Property-Testing-Algorithmus für Subgraphfreiheit und induzierte Subgraphfreiheit erarbeiten, der für Subgraphen mit k Quellkomponenten eine Anfragekomplexität von $\Theta(n^{1-1/k})$ erreicht; dieser Algorithmus hat einseitigen Fehler. Danach werden wir einen Algorithmus mit zweiseitigem Fehler entwickeln, der für eine bestimmte Klasse von Subgraphen – die aller Orientierungen von 3-Sternen – eine Anfragekomplexität von $\mathcal{O}(n^{1/2})$ hat. Die grundsätzliche Idee des Algorithmus ist, bestimmte Merkmale statistisch zu erfassen, die Graphen ohne 3-Sternknoten von Graphen mit vielen 3-Sternknoten unterscheiden; abhängig davon, ob die ermittelten Werte einen bestimmten Schwellwert überschreiten, wird die Eingabe akzeptiert oder abgelehnt. Dies führt dazu, dass der Algorithmus einen zweiseitigen Fehler hat. Die Anfragekomplexität ist geringer als die des einfachen Algorithmus: Dieser benötigt hier $\Theta(n^{2/3})$ Anfragen, da eine 3-Stern-Orientierung mit drei eingehenden Kanten ein Graph mit drei Quellkomponenten ist.

3.1 Ein Property-Testing-Algorithmus für H-Freiheit

Der Algorithmus TesteSubgraphFreiheit bekommt als Parameter zwei gerichtete gradbeschränkte Graphen G und H mit n beziehungsweise m Knoten und einen Näheparameter ϵ; k is die Anzahl der Quellkomponenten von H. Der Algorithmus zielt darauf ab, ein Vorkommen von H in G zu finden; dafür sampelt er jeden Knoten mit Wahrscheinlichkeit $p = \left(\frac{6m}{\epsilon n}\right)^{1/k}$, das heißt die erwartete Anzahl gesampelter Knoten ist $np = \mathcal{O}(\epsilon^{-1} m^{1/k} n^{1-1/k})$. Wie wir sehen werden, ist dies ausreichend dafür, in einem von H-Freiheit ϵ-entfernten Graphen G in mindestens einem Vorkommen von H einen Knoten in jeder Quellkomponente zu sampeln. Durch eine Breitensuche hinreichender Tiefe kann dann sichergestellt werden, dass dieses Vorkommen komplett exploriert wird (siehe Algorithmus 3.1).

Zunächst sei bemerkt, dass TesteSubgraphFreiheit die Eingabe nur ablehnt, falls tatsächlich ein Vorkommen von H entdeckt wird – der Algorithmus hat also einseitigen Fehler. Es bleibt zu zeigen, dass die Wahrscheinlichkeit, die Eingabe irrtümlich zu akzeptieren, gering ist. Dafür zeigen wir folgendes Lemma, das die Anzahl knotendisjunkter Vorkommen von H in G von unten beschränkt, falls G ϵ-fern von H-frei ist.

Lemma 3.1. *Seien $G = (V, E)$ und $H = (V_H, E_H)$ gerichtete Graphen mit jeweils durch $D \in \mathbb{N}$ beschränktem Knotenein- und -ausgangsgrad; sei $\epsilon < 1$ ein Näheparameter und sei*

TesteSubgraphFreiheit(G, H, ϵ)
  Sampele jeden Knoten von G mit einer Wahrscheinlichkeit von $p = \left(\frac{6m}{\epsilon n}\right)^{1/k}$;
    falls mehr als $4 \cdot \left(\frac{6m}{\epsilon}\right)^{1/k} n^{1-1/k}$ gesampelt wurden **then return true**
  Foreach gesampelte Knoten v **do**
    Führe Breitensuche von v ausgehend aus, mit einer maximalen Tiefe von m
    If die Breitensuche findet ein Vorkommen von H in G oder vervollständigt
      ein zuvor teilweise erforschtes Vorkommen **then return false**
  return true

Algorithmus 3.1: TesteSubgraphFreiheit

G ϵ-entfernt von H-frei. Es gelte $|V| =: n$ und $|V_H| =: m$. Dann enthält G mindestens $\frac{\epsilon n}{2m}$ knotendisjunkte Vorkommen von H.

Beweis. Es bestehe H aus l schwachen Zusammenhangskomponenten $H_1, \ldots, H_l$. Wir nehmen an, dass G weniger als $\frac{\epsilon n}{2m}$ knotendisjunkte Vorkommen von H enthält und führen dies zum Widerspruch. Sei M eine maximale Menge knotendisjunkter Vorkommen von H in G. Dann gibt es mindestens eine Zusammenhangskomponente H_i, für die kein weiteres Vorkommen in G existiert, das knotendisjunkt mit allen Vorkommen von H in M ist.

Insbesondere ist also mindestens eine Kante jedes Vorkommens von H_i zu einem Knoten inzident, der zu einem der Vorkommen von H in M gehört. Das Löschen aller Kanten, die zu Knoten von Vorkommen in M inzident sind, entfernt daher von jedem Vorkommen von H_i mindestens eine Kante; der entstehende Graph G' ist also H_i-frei und damit auch H-frei. Für jedes Vorkommen von H werden maximal $2Dm$ Kanten gelöscht.

Aufgrund der Annahme, es gebe weniger als $\frac{\epsilon n}{2m}$ knotendisjunkte Vorkommen von H in G, gilt $|M| < \frac{\epsilon n}{2m}$; das bedeutet, dass der oben beschriebene Löschvorgang weniger als $2Dm \cdot \frac{\epsilon n}{2m} = \epsilon Dn$ Kanten aus G löscht, um einen Graphen G' zu erhalten, der H-frei ist. Das ist ein Widerspruch zu der Annahme, dass G ϵ-fern von H-frei ist. □

Es bleibt zu zeigen, dass die Größe des Knotensamples ausreichend ist, um eins der Vorkommen von H komplett zu explorieren. Wir gehen hier zunächst davon aus, dass H schwach zusammenhängend ist.

Theorem 3.1. *Seien $G = (V, E)$ und $H = (V_H, E_H)$ gerichtete Graphen mit durch jeweils $D \in \mathbb{N}$ beschränktem Knotenein- und -ausgangsgrad, wobei H schwach zusammenhängend sei; sei $\epsilon < 1$ ein Näheparameter. Es gelte $|V| =: n$ und $|V_H| =: m$, und k sei die Anzahl der Quellkomponenten von H. Dann liefert* TesteSubgraphFreiheit(G, H, ϵ) true *zurück, falls G H-frei ist und mit Wahrscheinlichkeit mindestens* 2/3 false*, falls G ϵ-fern von H-frei ist; der Algorithmus hat eine Anfragekomplexität und Laufzeit von $\mathcal{O}\left(D^m \left(\frac{m}{\epsilon}\right)^{1/k} n^{1-1/k}\right)$.*

Beweis. Wie oben erörtert lehnt TesteSubgraphFreiheit nur ab, wenn ein Vorkommen von H in G nachgewiesen wird; also akzeptiert der Algorithmus, wenn G H-frei ist.

Wir nehmen im Weiteren an, dass G ϵ-fern von H-frei ist. Sei M eine maximale Menge von Vorkommen von H in G; Lemma 3.1 garantiert, dass die Kardinalität von M mindestens $\epsilon n/2m$ ist. Sei X_i das Ereignis, dass das nach einer festen aber beliebigen Ordnung i-te Vorkommen von H in M vollständig exploriert wird. Da die Breitensuchen in TesteSubgraphFreiheit eine Tiefe von m haben, geschieht dies, wenn in jeder Quellkomponente dieses Vorkommens mindestens ein Knoten gesampelt wurde; die Wahrscheinlichkeit hierfür ist mindestens p^k.

Da die Vorkommen von H in M knotendisjunkt sind, sind die Ereignisse X_i unabhängig voneinander. Es ist also

$$\begin{aligned}\Pr\left[\bigcap_{1\le i\le|M|}\bar{X}_i\right] &= \prod_{1\le i\le|M|}(1-\Pr[X_i]) \le \left(1-p^k\right)^{\epsilon n/2m}\\ &= \left(1-\frac{6m}{\epsilon n}\right)^{\epsilon n/2m} \le e^{-3} < \frac{1}{12}\end{aligned}$$

die Wahrscheinlichkeit, dass kein Vorkommen von H in M komplett exploriert wird und damit eine obere Schranke für die Wahrscheinlichkeit, dass kein Vorkommen von H in G identifiziert wird.

Die Wahrscheinlichkeit, dass TestSubgraphFreiheit in der ersten Zeile akzeptiert, weil zuviele Knoten gesampelt werden sollten, lässt sich durch die Markow-Ungleichung beschränken: Der Erwartungswert für die Anzahl gesampelter Knoten ist $np = (3m/\epsilon)^{1/k} \cdot n^{1-1/k}$, und die Wahrscheinlichkeit, mehr als viermal so viele Knoten zu sampeln, ist höchstens $1/4$.

Aufgrund der Union-Bound ist daher die Gesamtwahrscheinlichkeit, dass TesteSubgraphFreiheit irrtümlich akzeptiert, höchstens $1/4 + 1/12 = 1/3$.

Die Anzahl gesampelter Knoten ist höchstens $4 \cdot \left(\frac{3m}{\epsilon}\right)^{1/k} n^{1-1/k} + 1$; für maximal $4 \cdot \left(\frac{3m}{\epsilon}\right)^{1/k} n^{1-1/k}$ gesampelte Knoten wird eine Breitensuche der Tiefe m durchgeführt. Da G einen auf D beschränkten Ausgangsgrad hat, werden dabei höchstens D^m Knoten erkundet, das heißt ebenso viele Adjazenzlisteneinträge gelesen. Es ergibt sich also eine Anfragekomplexität von $\mathcal{O}\left(D^m \left(\frac{m}{\epsilon}\right)^{1/k} n^{1-1/k}\right)$. Da die Laufzeit einer Breitensuche linear in der Anzahl erforschter Kanten ist, ist dies auch die Laufzeit des Algorithmus. □

Wir nehmen nun an, dass H nicht zusammenhängend ist, sondern aus l schwachen Zusammenhangskomponenten $H_1, \ldots, H_l$ besteht. In diesem Fall reicht es aus, von jeder Zusammenhangskomponente H_i ein beliebiges Vorkommen zu finden, wenn die Vorkommen der einzelnen H_i knotendisjunkt sind; insbesondere können innerhalb einer maximalen Menge M von Vorkommen wie im obigen Beweis die Vorkommen der einzelnen H_i beliebig kombiniert werden, da alle dort enthaltenen Vorkommen aller Zusammenhangskomponenten paarweise knotendisjunkt sind.

Das bedeutet, dass der Algorithmus TesteSubgraphFreiheit für jede Zusammenhangskomponente H_i einzeln ausgeführt werden kann. Wir verwenden hierzu die Variante

```
TesteSubgraphFreiheitVerstärkt(G, H, ϵ, p)
  For i → 1 to log_3 ⌈1/p⌉
    If not TesteSubgraphFreiheit(G, H, ϵ, p) then return false
  return true
```

Algorithmus 3.2: TesteSubgraphFreiheitVerstärkt

TesteSubgraphFreiheitVerstärkt, die die Technik der Wahrscheinlichkeitsverstärkung einsetzt, um bessere Erfolgswahrscheinlichkeiten zu gewährleisten; der neue Parameter p ist die maximale erlaubte Fehlerwahrscheinlichkeit des Algorithmus.

Lemma 3.2. *Seien G und H gerichtete Graphen mit durch jeweils $D \in \mathbb{N}$ beschränktem Knotenein- und -ausgangsgrad, $\epsilon < 1$ ein Näheparameter und $p \leq 1/3$ eine Wahrscheinlichkeit. Dann gibt* TesteSubgraphFreiheitVerstärkt true *aus, falls G H-frei ist und mit Wahrscheinlichkeit mindestens $1 - p$* false*, falls G ϵ-fern von H-frei ist. Der Algorithmus hat eine Anfragekomplexität und Laufzeit von $\mathcal{O}\left(\log \frac{1}{p} \cdot D^m \left(\frac{m}{\epsilon}\right)^{1/k} n^{1-1/k}\right)$.*

Beweis. Die Korrektheit im Falle dass G H-frei ist folgt direkt aus der Korrektheit von TesteSubgraphFreiheit. Wir nehmen nun an, dass G ϵ-fern von H-frei ist. Nach Theorem 3.1 hat jeder Aufruf von TesteSubgraphFreiheit eine Wahrscheinlichkeit von $2/3$, *false* zurückzugeben. Die Wahrscheinlichkeit, dass keiner der Aufrufe *false* zurückgibt und TesteSubgraphFreiheitVerstärkt daher die Eingabe irrtümlich akzeptiert, ist also höchstens $(1/3)^{\log_3 \lceil 1/p \rceil} \leq 3^{-\log_3 1/p} = p$.

Wegen der in Theorem 3.1 gezeigten Anfragekomplexität und Laufzeit von TesteSubgraphfreiheit und da die *for*-Schleife von TesteSubgraphFreiheitVerstärkt $\log_3 \left\lceil \frac{1}{p} \right\rceil = \mathcal{O}\left(\log \frac{1}{p}\right)$ mal wiederholt wird, ist die Anfragekomplexität und Laufzeit des Algorithmus $\mathcal{O}\left(\log \frac{1}{p} \cdot D^m \left(\frac{m}{\epsilon}\right)^{1/k} n^{1-1/k}\right)$. □

Wir rufen TesteSubgraphFreiheitVerstärkt nun für jede schwache Zusammenhangskomponente H_i einzeln auf, wobei wir p auf $\frac{1}{3l}$ setzen. Wir nehmen an, dass G ϵ-fern von H-frei ist: Die Wahrscheinlichkeit, dass mindestens einer der Aufrufe von TesteSubgraphFreiheitVerstärkt fehlschlägt, ist nach der Union-Bound höchstens $l \cdot p = 1/3$. Daraus ergibt sich der folgende Korollar:

Korollar 3.1. *Sei $H = (V_H, E_H)$ ein gerichteter Graph mit m Knoten und durch jeweils $D \in \mathbb{N}$ beschränktem Knotenein- und -ausgangsgrad, der aus l schwachen Zusammenhangskomponenten besteht. Sei k_{max} die maximale Anzahl der Quellkomponenten einer dieser Zusammenhangskomponenten und k_{min} die minimale und sei $\epsilon < 1$ ein Näheparameter. Dann gibt es einen Property-Testing-Algorithmus für H-Freiheit, der in Graphen mit n Knoten und auf D beschränktem Eingangsgrad eine Anfragekomplexität und Laufzeit von $\mathcal{O}\left(l \log l \cdot D^m \left(\frac{m}{\epsilon}\right)^{1/k_{min}} n^{1-1/k_{max}}\right)$ hat.*

3.2 Ein Property-Testing-Algorithmus für 3-Stern-Freiheit

Wir werden nun das Problem der Subgraphfreiheit für eine bestimmte Klasse von Subgraphen betrachten, nämlich die Orientierungen von 3-Sternen. Dabei wollen wir einen Property-Testing-Algorithmus für die Frage entwickeln, ob ein beliebiger der Graphen dieser Klasse in G als Subgraph vorkommt. Im Folgenden nehmen wir zunächst an, dass die möglichen Eingabegraphen auf solche beschränkt sind, die keine Doppelkanten haben; das heißt, wenn zwischen zwei Knoten u und v eine Kante (u, v) existiert, dann existiert (v, u) nicht. Am Ende des Kapitels werden wir diese Einschränkung fallen lassen und zeigen, dass sich 3-Stern-Freiheit auch in Graphen testen lässt, die Doppelkanten aufweisen dürfen. Es sei bemerkt, dass für Graphen ohne Doppelkanten der ungerichtete Grad eines Knotens gleich der Anzahl der zu ihm adjazenten Knoten ist.

Da in gerichteten Graphen jeder k-Stern eine Orientierung haben muss, werden wir im Folgenden alle Orientierungen von k-Sternen kurz als k-Sterne bezeichnen. Wir werden trotzdem zwischen verschiedenen Typen von k-Sternen unterscheiden müssen, je nachdem, wieviele ein- beziehungsweise ausgehende Kanten sie haben:

Definition 3.1. *Sei G ein gerichteter k-Stern. Für $2 \leq m \leq k$ nennen wir G* eingehenden *m-Stern, falls mindestens m der Kanten von G zum Zentralknoten hin gerichtet sind. Wir nennen G* ausgehenden *m-Stern, falls mindestens m der Kanten zu den Außenknoten hin gerichtet sind. Analog bezeichnen wir die Zentralknoten solcher k-Sterne als* eingehende *beziehungsweise* ausgehende *m-Sternknoten.*

Grundsätzlich kann der Algorithmus TesteSubgraphFreiheit aus dem vorhergehenden Kapitel verwendet werden, um auf jede mögliche Orientierung von 3-Sternen einzeln zu testen; allerdings erreicht der Algorithmus für eingehende 3-Sterne ohne ausgehende Kanten lediglich eine Laufzeit von $\mathcal{O}(n^{2/3})$. Wir werden in diesem Kapitel hingegen einen Algorithmus erarbeiten, der in schwach zusammenhängenden Graphen G eine Laufzeit von $\mathcal{O}(n^{1/2})$ erreicht. Dabei muss ein zweiseitiger Fehler in Kauf genommen werden.

Durch die Beschränkung auf schwach zusammenhängende Graphen lässt sich dieses Problem auch anders beschreiben: als die Aufgabe, kreis- oder linienförmige gerichtete Graphen von solchen zu unterscheiden, die viele Knoten mit höherem Grad haben. Das ist nicht trivial, denn dadurch, dass die eingehenden Kanten von Knoten nicht sichtbar sind, kann der von einem Knoten aus durch Breitensuche explorierbare Teil eines Graphen sehr klein sein, bis hin zu Knoten, die nur eingehende Kanten besitzen.

Der Algorithmus nutzt grundsätzlich eine Eigenschaft aus, die schwach zusammenhängende Graphen haben, wenn sie viele eingehende 3-Sternknoten ohne ausgehende Kanten enthalten, aber wenige 3-Sternknoten mit ausgehenden Kanten: Die Anzahl der an eingehenden 2-Sternen beteiligten Kanten ist signifikant größer als die Anzahl der an ausgehenden 2-Sternen beteiligten Kanten (es sei hierzu noch einmal bemerkt, dass jeder 3-Sternknoten gleichzeitig auch ein 2-Sternknoten ist). Das bedeutet, dass in diesem Fall die Abschätzung der Anzahl ausgehender 2-Sterne über ein Knotensample signifikant andere Ergebnisse liefert als die Abschätzung der Anzahl eingehender 2-Sterne über eine

TESTE3STERNFREIHEIT(n, G, ϵ)
  if SCHÄTZEKANTENZAHL$(n, G, \frac{\epsilon}{16}) > n + \frac{\epsilon n}{16}$ **then return** false
  Foreach 3-Stern-Orientierung H mit mindestens einer ausgehenden Kante
    if not TESTESUBGRAPHFREIHEITVERSTÄRKT$(G, H, \frac{\epsilon}{192D}, \frac{1}{6})$
      then return false
  Sampele zufällig gleichverteilt $s_{3.3.1} = \frac{48}{\epsilon}$ Knoten $v_1, \ldots, v_{3.3.1}$
  $\hat{k} \leftarrow \frac{n}{s_{3.3.1}}$ mal die Anzahl von ausgehenden 2-Sternknoten in $\{v_1, \ldots, v_{3.3.1}\}$
  $\hat{k} \leftarrow \hat{k} + \frac{\epsilon n}{12}$
  Sampele jede Kante mit einer Wahrscheinlichkeit von $p = \frac{128D}{\epsilon^{3/2}\sqrt{n}}$
  if mehr als $s_{3.3.2} = \frac{2048 \cdot D\sqrt{n}}{\epsilon^{3/2}}$ Kanten wurden gesampelt **then return** false
  $\hat{c} \leftarrow \frac{1}{p^2}$ mal die Anzahl der Kollisionen von Zielknoten gesampelter Kanten
  if $\hat{c} < \frac{3}{2}\epsilon n$ **then return** true
  if $\hat{r} := \frac{\hat{c}}{\hat{k}} > 1 + \frac{\epsilon}{24}$ **then return** false
  else return true

Algorithmus 3.3: TESTE3STERNFREIHEIT

Kollisionsstatistik von gesampelten Kanten. Für eine solche Kollisionsstatistik reicht das Sampeln von $\mathcal{O}(n^{1/2})$ Kanten.

Enthält ein schwach zusammenhängender Graph hingegen keine 3-Sterne, so ist die Anzahl der ein- und ausgehenden 2-Sterne nahezu gleich und ebenso die Anzahl der an ihnen beteiligten Kanten, und damit unterscheiden sich die Ergebnisse der beiden Sampelmethoden dort nur geringfügig.

Falls eine beliebige andere Orientierung von 3-Sternen im getesteten Graphen häufig vorkommt, lässt sich dies in Laufzeit $\mathcal{O}(n^{1/2})$ mit Hilfe von TESTESUBGRAPHFREIHEIT nachweisen: Jede dieser Orientierungen verfügt über maximal zwei eingehende Kanten und damit höchstens zwei Quellkomponenten. Der Algorithmus TESTE3STERNFREIHEIT formalisiert diese Ideen.

Im Gegensatz zum allgemeinen Algorithmus zum Testen auf Subgraphfreiheit setzt TESTE3STERNFREIHEIT voraus, dass im übergebenen Graphen G nicht nur jeweils der Ein- und Ausgangsgrad jedes Knotens durch $D \in \mathbb{N}$ beschränkt ist, sondern der ungerichtete Knotengrad; wir können dabei $D \geq 3$ annehmen. Wir werden in der Analyse des Algorithmus zudem annehmen, dass $n = \omega\left(\frac{D^4}{\epsilon^3}\right)$ gilt; andernfalls könnte der Graph mit $\mathcal{O}(\epsilon^{-3/2}D^3\sqrt{n})$ Anfragen komplett exploriert und das Problem damit deterministisch gelöst werden.

Wir benötigen zunächst einige Hilfssätze zu den oben angesprochenen strukturellen Eigenschaften von schwach zusammenhängenden gerichteten Graphen. Um diese Eigenschaften zu formalisieren, führen wir den Begriff der *Balance* eines gerichteten Graphen ein:

Definition 3.2. *Sei $G = (V, E)$ ein gerichteter Graph. Die* Balance $\mathcal{B}(G)$ *von G ist definiert als die Differenz zwischen der Anzahl der eingehenden und der ausgehenden*

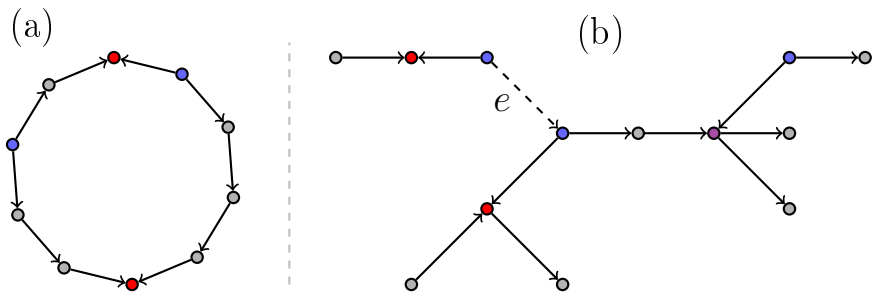

Abbildung 3.1: *Orientierungen von Graphen: (a) Kreisförmiger Graph; (b) Baum mit im Induktionsschritt gewählter Kante e. Ein- bzw. ausgehende 2-Sternknoten sind rot bzw. blau markiert.*

2-Sternknoten in G, das heißt

$$\mathcal{B}(G) = |\ \#\{v \in V | \deg_{out}(v) \geq 2\} - \#\{v \in V | \deg_{in}(v) \geq 2\}\ |$$

Wir zeigen zunächst für Orientierungen kreisförmiger Graphen eine obere Schranke für die Balance; danach verwenden wir dieses Ergebnis, um die Balance in komplexeren Graphen zu beschränken. Ein kreisförmiger Graph ist ein zusammenhängender ungerichteter Graph, in dem alle Knoten einen Grad von zwei besitzen.

Lemma 3.3. *Sei $G = (V, E)$ eine Orientierung eines kreisförmigen ungerichteten Graphen. Dann gilt $\mathcal{B}(G) = 0$.*

Beweis. Da jeder Knoten in G einen Grad von zwei hat, haben ausgehende 2-Sternknoten in G keine eingehenden Kanten und eingehende 2-Sternknoten keine ausgehenden Kanten; alle Knoten, die weder ein- noch ausgehende 2-Sternknoten sind, haben genau eine ein- und eine ausgehende Kante. Da die Gesamtzahl der von Knoten ausgehenden Kanten allerdings gleich der Gesamtzahl der zu Knoten eingehenden Kanten ist, ist die durchschnittliche Anzahl ausgehender Kanten pro Knoten 1, ebenso wie die durchschnittliche Anzahl eingehender Kanten. Daher muss die Anzahl eingehender 2-Sterne in G gleich der Anzahl ausgehender 2-Sterne sein und es gilt $\mathcal{B}(G) = 0$. □

Die folgenden Beweise werden mit dem Kontrahieren und Re-Expandieren oder Entfernen und wieder Einfügen von Kanten arbeiten. Dazu bemerken wir, dass die Balance eines Graphen sich maximal um eins ändert, wenn eine zusätzliche Kante eingefügt wird: Nur am Zielknoten der Kante kann ein neuer eingehender 2-Sternknoten entstehen, und nur am Ausgangsknoten ein neuer ausgehender 2-Sternknoten. Dies gilt natürlich „rückwärts“ auch für das Entfernen einer Kante.

Zunächst zeigen wir, dass Orientierungen linienförmiger Graphen eine Balance von höchstens 1 haben, und dass, wenn die Balance nicht 0 ist, die Kanten beider Außenknoten die gleiche Ausrichtung haben. Ein linienförmiger Graph ist ein zusammenhängender ungerichteter Graph, dessen Knoten einen Grad von je zwei haben, bis auf zwei Endknoten, deren Grad jeweils eins ist.

Lemma 3.4. *Sei $G = (V, E)$ eine Orientierung eines linienförmigen Graphen. Dann gilt $\mathcal{B}(G) \leq 1$. Falls $\mathcal{B}(G) \neq 0$, gilt zudem eine der beiden folgenden Aussagen:*

- *G hat einen ausgehenden 2-Sternknoten mehr als eingehende 2-Sternknoten und die beiden Endknoten haben eingehende Kanten.*
- *G hat einen eingehenden 2-Sternknoten mehr als ausgehende 2-Sternknoten und die beiden Endknoten haben ausgehende Kanten.*

Beweis. Für die Aussage $\mathcal{B}(G) \leq 1$ bemerken wir, dass das Einfügen einer zusätzlichen Kante e zwischen den beiden Endknoten u und v von G die Orientierung eines kreisförmigen Graphen erzeugt; sei G' dieser Graph. Wegen Lemma 3.3 gilt $\mathcal{B}(G') = 0$. Da das Einfügen von e die Balance nur um höchstens 1 geändert haben kann, kann die Balance von G nicht größer als 1 sein.

Falls $\mathcal{B}(G) > 1$ gilt, muss das Einfügen von e zudem einen ein- oder ausgehenden 2-Sternknoten erzeugt haben: einen eingehenden 2-Sternknoten, falls G zuvor mehr aus- als eingehende 2-Sternknoten hatte, einen ausgehenden 2-Sternknoten sonst. Dabei darf am jeweils anderen Endknoten kein 2-Sternknoten des anderen Typs entstehen, ansonsten wäre die Balance von G' nicht 0 – falls e also einen eingehenden 2-Sternknoten erzeugt, so haben sowohl u als auch v schon in G eine eingehende Kante besessen, und falls e einen ausgehenden 2-Sternknoten erzeugt, haben beide Knoten in G schon eine ausgehende Kante. Damit folgt die zweite Aussage des Lemmas. □

Wir werden nun zeigen, dass die maximale Balance einer Orientierung eines Baums abhängig von der Anzahl seiner Blätter ist.

Lemma 3.5. *Sei $G = (V, E)$ eine Orientierung eines Baums mit k Blättern. Dann gilt $\mathcal{B}(G) \leq k - 1$.*

Beweis. Wir beweisen dieses Lemma durch vollständige Induktion über die Anzahl k der Blätter des Baums. Als Induktionsanfang nehmen wir $k = 2$ an, das heißt G ist eine Orientierung eines linienförmigen Graphen. Nach Lemma 3.4 gilt dann $\mathcal{B}(G) \leq 1$.

Für den Induktionsschritt nehmen wir nun an, dass G eine Orientierung eines Baums mit $k > 2$ Blättern ist und dass alle Orientierungen von Bäumen mit $m < k$ Blättern eine Balance von höchstens $m - 1$ haben.

Da G mindestens drei Blätter hat, gibt es mindestens einen 3-Sternknoten v in G. Wir löschen eine zu v inzidente Kante e, und da G die Orientierung eines Baums ist, zerfällt G dadurch in zwei nicht verbundene Teilgraphen G_1 und G_2, wobei v in G_1 liegt. Insbesondere können wir v so wählen, dass G_2 nur zwei Blätter hat – also eine Orientierung eines linienförmigen Graphen ist – oder nur aus einem einzigen Knoten besteht (siehe Abbildung 3.1): Dies erreichen wir dadurch, dass wir e so wählen, dass einer der Blattknoten von G_2 auch in G Blattknoten ist.

G_1 hat $k - 1$ Blätter und daher wegen der Induktionsvoraussetzung eine Balance von höchstens $k - 2$; G_2 hat eine Balance von 0, falls er aus einem einzelnen Knoten besteht und ansonsten aufgrund der Induktionsvoraussetung eine Balance von höchstens 1.

Wir fügen nun e wieder ein. Es gilt $\mathcal{B}(G) \leq \mathcal{B}(G_1) + \mathcal{B}(G_2) + 1$; falls $\mathcal{B}(G_2) = 0$ gilt, folgt also direkt $\mathcal{B}(G) \leq k - 1$. Falls es in G_1 mehr eingehende als ausgehende 2-Sterne gibt und in G_2 mehr ausgehende als eingehende, folgt dies ebenfalls direkt. Wir gehen nun also davon aus, dass $\mathcal{B}(G_2) = 1$ gilt und dass in beiden Teilgraphen die Art der überzähligen 2-Sterne die gleiche ist; wir können dabei ohne Beschränkung der Allgemeinheit annehmen, dass sowohl G_1 als auch G_2 mehr eingehende als ausgehende 2-Sterne enthalten.

Wegen Lemma 3.4 haben dann aber beide Endknoten von G_2 ausgehende Kanten – das Zurückeinfügen von e kann also einen neuen eingehenden 2-Sternknoten nur bei v erzeugen. In diesem Fall ist e aber eine ausgehende Kante am entsprechenden Endknoten von G_2 und erzeugt dort einen neuen ausgehenden 2-Stern. Das bedeutet, dass das Einfügen von e die Balance nicht erhöht, es gilt also $\mathcal{B}(G) \leq \mathcal{B}(G_1)+\mathcal{B}(G_2) \leq k-2+1 = k-1$. □

Schlussendlich nutzen wir Lemma 3.5, um eine obere Schranke für die Balance beliebiger schwach zusammenhängender Graphen zu erhalten. Dazu benötigen wir die folgende Beobachtung:

Beobachtung 3.1. *Sei $G = (V, E)$ die Orientierung eines Baums mit 3-Sternknoten $C_3 \subseteq V$. Dann hat G genau $2 - 2|C_3| + \sum_{v \in C_3} \deg(v)$ Blätter.*

Beweis. Wir zeigen die Beobachtung durch vollständige Induktion über die Anzahl der 3-Sternknoten in G. Für den Induktionsanfang bemerken wir, dass G für $|C_3| = 0$ zwei Blätter hat und damit die Aussage gilt; wir nehmen nun $|C_3| = 1$ an, und dass der einzige 3-Sternknoten v einen ungerichteten Grad von k hat. G besteht dann aus einem zentralen Knoten v, von dem aus linienförmige Untergraphen ausgehen, deren Endknoten die Blätter von G sind. G hat also k Blätter und es gilt $2 - 2 \cdot |C_3| + \sum_{v \in C_3} = k$; damit gilt die Aussage auch für Graphen mit einem 3-Sternknoten.

Für den Induktionsschritt nehmen wir an, dass G ein Graph mit $|C_3| = i$ 3-Sternknoten ist und dass für alle Graphen G' mit $j < i$ 3-Sternknoten die Aussage gilt. Da G ein Baum ist und mindestens zwei 3-Sternknoten enthält, gibt es mindestens einen 3-Sternknoten v, von dessen inzidenten Kanten genau eine Kante e auf einem direkten Pfad zu einem weiteren 3-Sternknoten u liegt, alle weiteren inzidenten Kanten jedoch auf Pfaden zu Blättern. Wir entfernen alle Knoten und Kanten, die auf Pfaden von v zu Blättern liegen, erhalten jedoch v und e: Im entstehenden Graphen G' ist v ein Blatt. Sei C_3' die Menge der 3-Sternknoten von G': Es gilt $C_3' = C_3 \backslash \{v\}$, das heißt $|C_3'| < i$. Einsetzen der Induktionsvoraussetzung ergibt, das G' genau $2 - 2|C_3'| + \sum_{v \in C_3'} \deg(v)$ Blätter hat, wobei für alle $v \in C_3'$ der Grad von v in G' gleich dem Grad von v in G ist.

Wir betrachten nun den Subgraphen $\tilde{G}$ von G, der durch v und die gelöschten Knoten induziert wird. Dieser Graph hat k Blätter, wobei k der Grad von v in $\tilde{G}$ ist. Zusammenfügen von G' und $\tilde{G}$ ergibt G, wobei gegenüber G' v nun kein Blatt mehr ist, dafür aber $k = \deg(v) - 1$ neue Blätter von $\tilde{G}$ hinzugefügt werden; G enthält also $\deg(v) - 2$ mehr Blätter als G. Die Gesamtzahl der Blätter von G ist also

$$\deg(v) - 2 + 2 - 2|C_3'| + \sum_{v \in C_3'} \deg(v) = 2 - 2|C_3' \cup \{v\}| + \sum_{v \in C_3' \cup \{v\}} \deg(v)$$
$$= 2 - 2|C_3| + \sum_{v \in C_3} \deg(v).$$

□

Es sei explizit darauf hingewiesen, dass das folgende Lemma auch für Graphen gilt, die Doppelkanten enthalten:

Lemma 3.6. *Sei $G = (V, E)$ ein schwach zusammenhängender gerichteter Graph mit $(1 + \delta)|V|$ Kanten für ein beliebiges $\delta > 0$. Sei außerdem $C_3 \subseteq V$ die Menge der 3-Sternknoten in G. Dann ist die Balance von G höchstens $2+\delta|V|-2|C_3|+\sum_{v \in C_3} \deg(v)$.*

Beweis. Wir entfernen $\delta|V| + 1$ Kanten aus G derart, dass der entstehende Graph G' ebenfalls schwach zusammenhängend ist; insbesondere entfernen wir von allen Doppelkanten (u, v) und (v, u) eine der Kanten. Da G' genau $|V| - 1$ Kanten hat und schwach zusammenhängend ist, ist G' eine Orientierung eines Baums. Nach Lemma 3.5 ist die Balance von G' also kleiner als die Anzahl seiner Blätter. Diese ist nach Beobachtung 3.1 $2 - 2|C_3| + \sum_{v \in C_3} \deg(v)$; es gilt also $\mathcal{B}(G') \leq 1 - 2|C_3| + \sum_{v \in C_3} \deg(v)$.

Durch Zurückeinfügen der gelöschen Kanten ergibt sich wieder G; da jede eingefügte Kante die Balance um maximal eins verändert, folgt $\mathcal{B}(G) \leq \mathcal{B}(G_1) + \delta|V| + 1 \leq 2 + \delta|V| - 2|C_3| + \sum_{v \in C_3} \deg(v)$. □

Im Folgenden nehmen wir wieder an, dass in den betrachteten Graphen Doppelkanten ausgeschlossen sind; wir werden dies in den Prämissen der nachfolgenden Lemmas nicht eigens erwähnen. Das folgende Lemma garantiert, dass jeder schwach zusammenhängende Graph, der mehr Kanten als Knoten hat, mindestens einen 3-Sternknoten enthält; solche Graphen können also abgelehnt werden.

Lemma 3.7. *Sei $G = (V, E)$ ein schwach zusammenhängender Graph mit n Knoten und $m > n$ Kanten. Dann enthält G mindestens einen 3-Sternknoten.*

Beweis. Da jede Kante in G zwei inzidente Knoten hat, ist die Summe aller Knotengrade in G $2m$. Der durchschnittliche Knotengrad ist also $2m/n > 2$. Also gibt es mindestens einen Knoten mit einem Knotengrad größer zwei. □

Ein weiterer Mosaikstein für den Beweis der Korrektheit von TESTE3STERNFREIHEIT ist der Algorithmus SCHÄTZEKANTENZAHL, der die Anzahl der Kanten eines gegebenen Graphen G approximiert. Dazu sampelt er zufällig gleichverteilt $\mathcal{O}(D/\epsilon)$ Paare $(v, i) \in V(G) \times \{1, \ldots, D\}$, das heißt Kombinationen eines Knotens und der Nummer eines Adjazenzlistenplatzes. Für alle gesampelten Paare prüft er, ob der entsprechende Platz der Adjazenzliste mit einer Kante belegt ist und rechnet den Anteil gefundener

SchätzeKantenzahl(n, G, ϵ)
  $\hat{m} \leftarrow 0$
  Sampele zufällig gleichverteilt $s_{3.4} = \frac{2D}{\epsilon}$ Paare $(v_1, k_1), \ldots, (v_{s_{3.4}}, k_{s_{3.4}})$ eines Knotens und eines Adjazenzlistenplatzes, d.h. $(v_i, k_i) \in V \times \{1 \ldots D\}$
  for $i \leftarrow 1$ **to** $s_{3.4}$ **do**
    if v_i besitzt eine k_i-te Kante **then** $\hat{m} \leftarrow \hat{m} + \frac{Dn}{s_{3.4}}$
  return $\hat{m}$

Algorithmus 3.4: SchätzeKantenzahl

Kanten auf den gesamten Graphen hoch. Diese Schätzung erlaubt es, Graphen mit deutlich mehr als n Kanten direkt abzulehnen – nach Lemma 3.7 enthalten solche Graphen mindestens einen 3-Sternknoten.

Eine ähnliche Technik findet schon im Tester für Kreisfreiheit in ungerichteten gradbeschränkten Graphen von Goldreich und Ron Verwendung [43]: Dort wird der durchschnittliche Grad von Knoten in großen Zusammenhangskomponenten geschätzt, was Rückschlüsse auf das Vorhandensein von Kreisen erlaubt.

Der Korrektheitsbeweis von SchätzeKantenzahl nutzt im Bereich der sublinearen Approximationsalgorithmen übliche Techniken:

Lemma 3.8. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und m Kanten und einem durch $D \in \mathbb{N}$ beschränkten ungerichteten Knotengrad, der in Adjazenzlistendarstellung gespeichert ist, und sei $\epsilon < 1$ ein Näheparameter. Dann liefert* SchätzeKantenzahl *einen Wert $\hat{m}$ zurück, für den mit Wahrscheinlichkeit mindestens $1 - 2e^{-4}$*

$$m - \epsilon n \leq \hat{m} \leq m + \epsilon n$$

gilt. Die Laufzeit des Algorithmus ist $\mathcal{O}(D/\epsilon)$.

Beweis. Sei X_i eine Indikatorzufallsvariable für das Ereignis, dass v_i einen k_i-ten Nachbarn besitzt, $i = 1, \ldots, s_{3.4}$; sei $X = \sum_{1 \leq i \leq s_{3.4}} X_i$. Jede der m Kanten von G ist in genau einer der Adjazenzlisten als ausgehende Kante gespeichert, und es gibt insgesamt Dn Adjazenzlistenplätze. Daher gilt

$$\mathrm{E}[X_i] = \Pr[X_i = 1] = \frac{m}{Dn},$$

und für den Erwartungswert $\mathrm{E}[\hat{m}]$ von $\hat{m}$ folgt aufgrund der Linearität des Erwartungswerts

$$\begin{aligned}\mathrm{E}[\hat{m}] &= \mathrm{E}\left[\frac{Dn}{s_{3.4}} \cdot X\right] = \mathrm{E}\left[\frac{Dn}{s_{3.4}} \cdot \sum_{1 \leq i \leq s_{3.4}} X_i\right] \\ &= \frac{Dn}{s_{3.4}} \cdot \sum_{1 \leq i \leq s_{3.4}} E[X_i] = \frac{Dn}{s_{3.4}} \cdot \sum_{1 \leq i \leq s_{3.4}} \frac{m}{Dn} = m.\end{aligned}$$

Es bleibt zu zeigen, dass $\hat{m}$ mit Wahrscheinlichkeit höchstens $2e^{-4}$ um mehr als ϵn von seinem Erwartungswert abweicht. Dazu erhalten wir durch Verwenden einer additiven Chernoff-Schranke

$$\Pr\left[|\hat{m} - \mathrm{E}[\hat{m}]| > \epsilon n\right] = \Pr\left[|X - E[X]| > \frac{\epsilon s_{3.4}}{D}\right] \leq 2e^{-\epsilon^2 s_{3.4}^2/D^2} = 2e^{-4}.$$

Die Laufzeit von SchätzeKantenzahl ergibt sich direkt aus der Anzahl gezogener Samples. □

Analog zum vorangehenden Beweis lässt sich zeigen, dass der im Algorithmus Teste-3SternFreiheit ermittelte Wert $\hat{k}$ mit hoher Wahrscheinlichkeit eine gute Schätzung der Anzahl ausgehender 2-Sterne im Eingabegraphen G ist:

Lemma 3.9. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und durch $D \in \mathbb{N}$ beschränktem ungerichtetem Knotengrad und $k \leq n$ ausgehenden 2-Sternknoten; sei $\epsilon < 1$ ein Näheparameter. Für den in* Teste3SternFreiheit*(n, G, ϵ) berechneten Wert $\hat{k}$ gilt*

$$k \leq \hat{k} \leq k + \frac{\epsilon}{6}n$$

mit Wahrscheinlichkeit mindestens $1 - 2e^{-16}$.

Beweis. Wir betrachten das in der fünften Zeile des Algorithmus gezogene Knotensample; sei X_i eine Indikatorzufallsvariable für das Ereignis, dass der Knoten v_i ein ausgehender 2-Sternknoten ist. Die Wahrscheinlichkeit für dieses Ereignis und damit auch der Erwartungswert von X_i ist k/n.

Sei $X = \sum_{1 \leq i \leq s_{3.3.1}} X_i$; damit gilt $\hat{k} = \frac{\epsilon}{12}n + \frac{n}{s_{3.3.1}}X$. Mit Hilfe der Linearität des Erwartungswerts folgern wir:

$$\mathrm{E}\left[\hat{k}\right] = \frac{\epsilon}{12}n + \frac{n}{s_{3.3.1}}\mathrm{E}[X] = \frac{\epsilon}{12}n + \frac{n}{s_{3.3.1}}\left(\sum_{1 \leq i \leq s_{3.3.1}} \frac{k}{n}\right) = k + \frac{\epsilon}{12}n.$$

Wie beim vorangehenden Lemma benutzen wir eine additive Chernoff-Schranke, um die Wahrscheinlichkeit zu beschränken, dass $\hat{k}$ um mehr als $\epsilon n/12$ von seinem Erwartungswert abweicht:

$$\Pr\left[\left|\hat{k} - \mathrm{E}[\hat{k}]\right| > \epsilon n/12\right] = \Pr\left[|X - \mathrm{E}[X]| > \frac{\epsilon}{12}s_{3.3.1}\right] \leq 2e^{-\epsilon^2 s_{3.3.1}^2/144} = 2e^{-16}.$$

□

Den Korrektheitsbeweis von Teste3SternFreiheit teilen wir in vier Teile auf: Der erste Teil behandelt den Fall, dass der eingegebene Graph G 3-Stern-frei ist, die restlichen den, dass G ϵ-fern von 3-Stern-frei ist. Hier ergeben sich drei Unterfälle, die wir getrennt behandeln:

- G hat mehr als $n + \frac{\epsilon}{8}n$ Kanten;

- G hat höchstens $n + \frac{\epsilon}{8}n$ Kanten und enthält mindestens $\frac{\epsilon}{16D}n$ 3-Sternknoten mit jeweils mindestens einer ausgehenden Kante;
- G hat höchstens $n + \frac{\epsilon}{8}n$ Kanten und enthält weniger als $\frac{\epsilon}{16D}n$ 3-Sternknoten mit jeweils mindestens einer ausgehenden Kante.

Hierbei gibt n die Anzahl der Knoten von G an. Für jede mögliche Eingabe gilt einer der genannten Fälle. Wir beginnen die Analyse mit den drei Fällen, in denen wir davon ausgehen, dass G ϵ-fern von 3-Stern-frei ist und nehmen zunächst an, dass G mehr als $n + \frac{\epsilon}{8}n$ Kanten enthält; dass TESTE3STERNFREIHEIT dann mit einer Wahrscheinlichkeit von mindestens $2e^{-4}$ in der ersten Zeile ablehnt, folgt direkt aus Lemma 3.8:

Lemma 3.10. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und durch $D \in \mathbb{N}$ beschränktem ungerichtetem Knotengrad und $m > n + \frac{\epsilon}{8}n$ Kanten; sei $\epsilon < 1$ ein Näheparameter. G habe keine Doppelkanten. Dann liefert* TESTE3STERNFREIHEIT*(n, G, ϵ) mit Wahrscheinlichkeit mindestens $1 - 2e^{-4}$* false *zurück.*

Der nächste Fall ist, dass G höchstens $n + \frac{\epsilon}{8}n$ Kanten hat und mindestens $\frac{\epsilon}{16D}n$ 3-Sternknoten mit jeweils mindestens einer ausgehenden Kante enthält. Jeder dieser Sternknoten ist Zentralknoten mindestens eines Vorkommens einer 3-Stern-Orientierung mit mindestens einer eingehenden Kante. Abzüglich Isomorphie gibt es drei solcher Orientierungen, so dass mindestens eine davon mindestens $\frac{\epsilon}{48D}n$ Vorkommen in G hat. Das Entfernen einer Kante entfernt höchstens zwei Vorkommen dieser 3-Stern-Orientierung als Subgraphen aus G: Um alle Vorkommen zu entfernen, sind also mindestens $\frac{\epsilon}{96D}n \geq \frac{\epsilon}{192D}n + 1$ Kantenmodifikationen nötig[1], und damit ist G $\frac{\epsilon}{192D}$-fern von Freiheit von dieser 3-Stern-Orienierung.

Der entsprechende Aufruf von TESTESUBGRAPHFREIHEITVERSTÄRKT in der dritten Zeile des Algorithmus gibt also wegen Lemma 3.2 mit Wahrscheinlichkeit mindestens 5/6 *false* zurück und die Eingabe wird abgelehnt.

Aus diesen Überlegungen ergibt sich das folgende Lemma:

Lemma 3.11. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und durch $D \in \mathbb{N}$ beschränktem ungerichtetem Knotengrad und $m \leq n + \frac{\epsilon}{8}n$ Kanten und mindestens $\frac{\epsilon}{16D}n$ 3-Sternknoten mit jeweils mindestens einer ausgehenden Kante; sei $\epsilon < 1$ ein Näheparameter. G habe keine Doppelkanten. Dann liefert* TESTE3STERNFREIHEIT*(n, G, ϵ) mit Wahrscheinlichkeit mindestens* 5/6 false *zurück.*

Es verbleibt der Fall, dass G höchstens $n + \frac{\epsilon}{8}n$ Kanten hat und weniger als $\frac{\epsilon}{16D}n$ 3-Sternknoten mit ausgehenden Kanten enthält.

Lemma 3.12. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und durch $D \in \mathbb{N}$ beschränktem ungerichtetem Knotengrad und $m \leq n + \frac{\epsilon}{8}n$ Kanten und weniger als $\frac{\epsilon}{16D}n$ 3-Sternknoten mit ausgehenden Kanten; sei $\epsilon < 1$ ein Näheparameter. G sei ϵ-entfernt von 3-Stern-frei.* TESTE3STERNFREIHEIT*(n, G, ϵ) liefert dann mit Wahrscheinlichkeit mindestens $\frac{5}{6}$* false *zurück.*

[1] $n \geq \frac{192D}{\epsilon}$ kann hier wegen der oben getroffenen Annahme $n = \omega\left(\frac{D^4}{\epsilon^2}\right)$ angenommen werden.

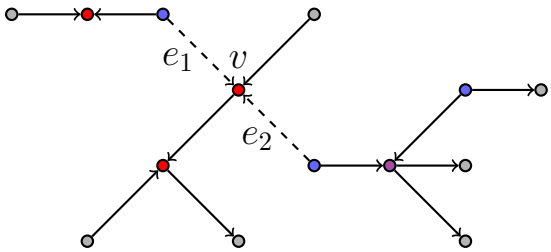

Abbildung 3.2: *Zweifachkollision an einem eingehenden 3-Sternknoten v; die gesampelten Kanten e_1 und e_2 sind hervorgehoben und die Knoten des Graphen sind farblich markiert wie in Abbildung 3.1.*

Beweis. Wir werden zeigen, dass unter den gegebenen Voraussetzungen in der vorletzten Zeile von TESTE3STERNFREIHEIT $\hat{r} > 1 + \frac{\epsilon}{24}$ mit Wahrscheinlichkeit mindestens $1 - e^{-6}$ gilt und damit der Algorithmus die Eingabe ablehnt. Die grundsätzliche Idee dieses Beweises ist die folgende: Nach Lemma 3.6 ist die maximale Balance von G linear abhängig von der Kantenzahl der 3-Sternknoten in G; die Wahrscheinlichkeit, bei dem Kantensample in TESTE3STERNFREIHEIT für einen Knoten zwei eingehende Kanten zu sampeln, steigt allerdings quadratisch in der Anzahl der eingehenden Kanten des Knotens. Da es mindestens ϵn 3-Sternknoten in G gibt und fast alle davon keine eingehenden Kanten haben, sollte der Schätzwert $\hat{c}$ im Erwartungswert die Anzahl der eingehenden 2-Sterne in G wesentlich deutlicher überschätzen, als die Anzahl der ausgehenden 2-Sterne aufgrund der Balance die Anzahl eingehender 2-Sterne übersteigen kann. Damit müsste $\hat{r}$ im Erwartungswert deutlich größer als eins sein.

Diese Idee muss noch in einigen Details konkretisiert werden. Wir benötigen dazu einige zusätzliche Definitionen: Sei C_2 die Menge der eingehenden 2-Sternknoten mit genau zwei eingehenden Kanten und sei C_3 die Menge der eingehenden 3-Sternknoten; sei außerdem $C_3' \subseteq C_3$ die Menge aller eingehenden 3-Sternknoten ohne ausgehende Kante und $\tilde{C}_3$ die Menge aller 3-Sternknoten in G. Wir definieren außerdem $c_2 := |C_2|$, $c_3 := |C_3|$, $c_3' := |C_3'|$ und $\tilde{c}_3 := |\tilde{C}_3|$. Die Menge C aller eingehenden 2-Sternknoten in G lässt sich dann als $C := C_2 + C_3$ darstellen und ihre Anzahl als $c := c_2 + c_3$.

Wir werden im Folgenden den Schätzwert $\hat{c}$ untersuchen; analog zu c_2 und c_3 teilen wir diesen in $\hat{c} = \hat{c}_2 + \hat{c}_3$ auf, wobei $\hat{c}_2$ und $\hat{c}_3$ den Beitrag von in C_2 beziehungsweise C_3 enthaltenen Knoten an $\hat{c}$ bezeichnen.

G enthält mindestens ϵn 3-Sternknoten: Enthielte G weniger, dann könnte man durch Löschen aller zu 3-Sternknoten inzidenten Kanten einen 3-Stern-freien Graphen erzeugen; da dabei weniger als $\epsilon D n$ Kanten gelöscht würden, widerspräche das der Voraussetzung, dass G ϵ-fern von 3-Stern-frei ist.

Laut Voraussetzung haben weniger als $\frac{\epsilon}{16D} n$ dieser 3-Sternknoten eingehende Kanten, das heißt es gibt wegen $D \geq 3$ $c_3' > \left(1 - \frac{1}{16D}\right) \epsilon n \geq \frac{47}{48} \epsilon n$ 3-Sternknoten ohne ausgehende Kanten. Da C_3' eine Teilmenge der Menge C_3 der eingehenden 3-Sternknoten ist, gilt außerdem

$$c_3 \geq c_3' > \left(1 - \frac{1}{16D}\right) \epsilon n \geq \frac{47}{48} \epsilon n. \tag{1}$$

Wir definieren

$$l := \sum_{v \in C_3} (\deg(v) - 2)$$

und werden im Folgenden eine obere Schranke für $\hat{k}$ und eine untere Schranke für $\hat{c}$ herleiten; beide Schranken beziehen sich auf l, was uns helfen wird, eine untere Schranke für $\hat{r} = \frac{\hat{c}}{\hat{k}}$ zu berechnen. Es gilt im Übrigen

$$l \geq c_3, \tag{2}$$

da jeder Knoten in C_3 einen Grad von mindestens drei hat.

Wir beginnen mit der oberen Schranke für $\hat{k}$. Da G höchstens $n + \frac{\epsilon}{8}n$ Kanten hat, ist die Balance von G nach Lemma 3.6 höchstens

$$\begin{aligned}
\mathcal{B}(G) &\leq \frac{\epsilon}{8}n + 2 + \sum_{v \in \tilde{C}_3} (\deg(v) - 2) \\
&= \frac{\epsilon}{8}n + 2 + \sum_{v \in C_3} (\deg(v) - 2) + \sum_{v \in \tilde{C}_3 \setminus C_3} (\deg(v) - 2) \\
&< \frac{\epsilon}{8}n + l + \frac{\epsilon}{16}n < l + \frac{\epsilon}{4}n,
\end{aligned}$$

da $\tilde{C}_3 \setminus C_3$ nur 3-Sternknoten enthält, die mindestens eine ausgehende Kante haben; dies sind nach Voraussetzung höchstens $\frac{\epsilon}{16D}n - 1$ viele, und zu ihnen sind höchstens $D\left(\frac{\epsilon}{16D}n - 1\right) < \frac{\epsilon}{16}n - 2$ Kanten inzident.

Da die Anzahl eingehender 2-Sternknoten c ist und die Balance von G höchstens $l + \frac{\epsilon}{4}n$, ist die Anzahl ausgehender 2-Sternknoten höchstens $c + l + \frac{\epsilon}{4}n$. Mit Hilfe von Lemma 3.9 lässt sich also folgern, dass mit Wahrscheinlichkeit mindestens $1 - 2e^{-16}$

$$\hat{k} \leq c + l + \frac{\epsilon}{4}n + \frac{\epsilon}{6}n < c_2 + c_3 + l + \frac{1}{2}c_3 \leq c_2 + \frac{5}{2}l$$

gilt; für die zweite Ungleichung benutzen wir dabei die Abschätzung $\frac{5}{12}\epsilon n < \frac{5}{12} \cdot \frac{48}{47}c_3 < \frac{1}{2}c_3$, die aus Ungleichung (1) folgt.

Wir werden nun $\hat{c}$ von oben beschränken. Für $v \in C$ sei X_v eine Indikatorzufallsvariable für das Ereignis, dass bei v eine Zweifachkollision festgestellt wird, also mindestens zwei eingehende Kanten von v gesampelt werden. Sei $X^{(2)} = \sum_{v \in C_2} X_v$ und $X^{(3)} = \sum_{v \in C_3} X_v$. Es gilt

$$\mathrm{E}[X_v] = \Pr[X_v = 1] = p^2(1-p)^{\deg_{\mathrm{in}}(v)-2}\binom{\deg_{\mathrm{in}}(v)}{2} \geq \alpha p^2 \binom{\deg_{\mathrm{in}}(v)}{2}$$

für $\alpha := \frac{47}{48}$ aufgrund der angenommenen Mindestknotenzahl von G und mit Hilfe der Bernoulli-Ungleichung: $(1-p)^{\deg_{\mathrm{in}}(v)-2} \geq (1-p)^D \geq 1 - pD = 1 - \frac{D^2}{\epsilon^{3/2}\sqrt{n}} \geq \frac{47}{48}$ für $n \geq \frac{2304D^4}{\epsilon^3}$.

Für den Beitrag der eingehenden 2-Sterne mit mindestens drei eingehenden Kanten zu $\hat{c}$ gilt daher

$$\begin{aligned}\mathrm{E}[\hat{c}_3] = \frac{1}{p^2}\mathrm{E}\left[X^{(3)}\right] = \frac{1}{p^2}\sum_{v\in C_3}\mathrm{E}[X_v] \geq \alpha\sum_{v\in C_3}\binom{\deg_{\text{in}}(v)}{2} \geq \alpha\sum_{v\in C_3'}\binom{\deg(v)}{2} \\ = \frac{\alpha}{2}\sum_{v\in C_3'}\deg(v)^2 - \deg(v) \geq 3\alpha\sum_{v\in C_3'}\deg(v) - 2\end{aligned}$$

aufgrund der Linearität des Erwartungswerts und weil für jeden 3-Sternknoten $\deg(v)^2 - \deg(v) \geq 6\cdot(\deg(v)-2)$ gilt. Wir wollen die letzte Summe nach l abschätzen, aber sie bezieht sich nur auf die Knoten in C_3'; es gibt nach Voraussetzung des Lemmas allerdings maximal $\frac{1}{16D}\epsilon n$ Knoten in $C_3\backslash C_3'$ – alle diese Knoten sind 3-Sterne mit mindestens einer ausgehenden Kante –, deren summierte Knotengrade höchstens $\frac{1}{16}\epsilon n \leq \frac{3}{47}l$ sind; letztere Ungleichung folgt aus

$$l \geq c_3 \geq \frac{47}{48}\epsilon n \tag{3}$$

mit Hilfe der Ungleichungen (2) und (1). Es folgt also

$$\sum_{v\in C_3'}\deg(v) - 2 > \sum_{v\in C_3}\deg(v) - 2 \quad - \sum_{v\in C_3\backslash C_3'}\deg(v) \geq l - \frac{3}{47}l = \frac{44}{47}l.$$

Diese Abschätzung setzen wir in die Abschätzung von $\mathrm{E}[\hat{c}_3]$ ein und erhalten $\mathrm{E}[\hat{c}_3] > 3\alpha l = \frac{11}{4}l \geq \frac{11}{4}\cdot\frac{47}{48}\epsilon n$ wegen $\alpha = \frac{47}{48}$ und Ungleichung (3). Eine multiplikative Chernoffschranke liefert eine obere Schranke für die Wahrscheinlichkeit, dass $\hat{c}_3$ diesen Wert um mehr als einen $(1-\epsilon/44)$-Faktor unterschreitet:

$$\begin{aligned}\Pr\left[\hat{c}_3 < (1-\epsilon/44)\mathrm{E}[\hat{c}_3]\right] = \Pr\left[X^{(3)} < (1-\epsilon/44)p^2\mathrm{E}\left[\hat{c}_3\right]\right] \leq \exp\left(-\frac{\epsilon^2p^2}{2\cdot 44^2}\mathrm{E}[\hat{c}_3]\right) \\ \leq \exp\left(-\frac{128^2\epsilon^2D^2}{2\cdot 44^2\epsilon^3 n}\cdot\frac{11}{4}\cdot\frac{47}{48}\epsilon n\right) \leq \exp\left(-\frac{8\cdot 47D^2}{33}\right) < e^{-100}.\end{aligned}$$

Analog gilt $\Pr\left[\hat{c}_3 < (1-\epsilon/24)\mathrm{E}[\hat{c}_3]\right] < e^{-600}$. Dabei benutzen wir jeweils $D \geq 3$.

Für die Bestimmung von $\hat{c}$ müssen wir nun noch die erwartete Anzahl an Zweifachkollisionen an Knoten in C_2 berechnen; diese Knoten haben genau zwei eingehende Kanten. Es gilt also

$$\mathrm{E}[\hat{c}_2] = \frac{1}{p^2}\sum_{v\in C_2}\mathrm{E}[X_v] = \frac{1}{p^2}\sum_{v\in C_2}p^2 = c_2.$$

Wir unterscheiden hier zwei Fälle: $c_2 \geq \frac{1}{16}l$ und $c_2 < \frac{1}{16}l$. Wenn c_2 groß ist, dann können wir mit Hilfe einer Chernoffschranke zeigen, dass $\hat{c}_2$ mit hoher Wahrscheinlichkeit nur wenig von c_2 abweicht, zusammen mit der obigen Schranke für $\hat{c}_3$ also $\hat{c}$ ein guter Schätzwert für c ist. Dann argumentieren wir, dass die Anzahl der eingehenden 3-Sternknoten in G mindestens ungefähr ein ϵ-Bruchteil der Anzahl aller Knoten und damit auch der Anzahl der eingehenden 2-Sternknoten ist; damit ist die durch die eingehenden 3-Sternknoten verursachte Erhöhung von $\hat{c}$ signifikant.

Falls c_2 klein ist, können wir die Abweichung zwischen $\hat{c}_2$ und c_2 nicht mit Hilfe einer Chernoffschranke abschätzen; allerdings ist dann der Beitrag von $\hat{c}_2$ zu $\hat{c}$ vernachlässigbar und kann in der Analyse ignoriert werden.

Wir beginnen mit dem ersten Fall: Es gilt $c_2 \geq \frac{1}{16}l \geq \frac{47}{16 \cdot 48}\epsilon n$ nach Abschätzung (3), und durch Anwendung einer multiplikativen Chernoffschranke erhalten wir

$$\begin{aligned}\Pr\left[\hat{c}_2 < (1-\epsilon/24)\mathrm{E}[\hat{c}_2]\right] &= \Pr\left[X^{(2)} < (1-\epsilon/24)\mathrm{E}\left[X^{(2)}\right]\right] \leq \exp\left(-\frac{\epsilon^2 p^2}{2 \cdot 24^2}c_2\right) \\ &\leq \exp\left(-\frac{128^2\epsilon^2 D^2}{2 \cdot 24^2 \epsilon^3 n} \cdot \frac{47\epsilon n}{16 \cdot 48}\right) < e^{-7}.\end{aligned}$$

Die Schranken für $\hat{c}_3$ und $\hat{c}_2$ ergeben zusammen eine Schranke für $\hat{c}$, so dass mit hoher Wahrscheinlichkeit

$$\begin{aligned}\hat{r} = \frac{\hat{c}}{\hat{k}} = \frac{\hat{c}_2 + \hat{c}_3}{\hat{k}} &\geq \frac{\left(1-\frac{\epsilon}{24}\right)\left(c_2 + \frac{11}{4}c_3\right)}{c_2 + \frac{5}{2}c_3} = \left(1-\frac{\epsilon}{24}\right)\left(1 + \frac{\frac{1}{4}c_3}{c_2 + \frac{5}{2}c_3}\right) \\ &= \left(1-\frac{\epsilon}{24}\right)\left(1 + \frac{1}{10}\left(\frac{\frac{5}{2}c_3}{c_2 + \frac{5}{2}c_3}\right)\right)\end{aligned}$$

gilt. Weiterhin gilt $c_2 \leq n - \left(1 - \frac{1}{16D}\right)\epsilon n$, da es mindestens $\left(1 - \frac{1}{16D}\right)\epsilon n$ 3-Sternknoten ohne eingehende Kanten gibt, und es gilt $c_3 \geq \left(1 - \frac{1}{16D}\right)\epsilon n$, da jeder dieser Knoten in C_3 enthalten ist (siehe Ungleichung (1)). Letztere Abschätzung dürfen wir in den Bruch $\frac{5c_3/2}{c_2+5c_3/2}$ einsetzen, da der Bruch kleiner eins ist und sich durch die Abschätzung Zähler und Nenner um den gleichen Betrag verringern: der entstehende Bruch hat einen kleineren Wert. Es gilt also weiter

$$\begin{aligned}\hat{r} &\geq \left(1-\frac{\epsilon}{24}\right)\left(1 + \frac{1}{10}\left(\frac{\frac{5}{2}\left(1-\frac{1}{16D}\right)\epsilon n}{n - \left(1-\frac{1}{16D}\right)\epsilon n + \frac{5}{2}\left(1-\frac{1}{16D}\right)\epsilon n}\right)\right) \\ &= \left(1-\frac{\epsilon}{24}\right)\left(1 + \frac{\frac{1}{4}\epsilon}{\frac{1}{1-\frac{1}{16D}} + \frac{3}{2}\epsilon}\right).\end{aligned}$$

Da $1 - \frac{1}{16D} \geq \frac{47}{48}$ und $\epsilon < 1$ – also insbesondere $\epsilon > \epsilon^2$ –, folgt schließlich

$$\begin{aligned}\hat{r} &\geq \left(1-\frac{\epsilon}{24}\right)\left(1 + \frac{\frac{1}{4}\epsilon}{\frac{48}{47} + \frac{3}{2}\epsilon}\right) \geq \left(1-\frac{\epsilon}{24}\right)\left(1 + \frac{\epsilon}{\frac{192}{47} + 6}\right) = \left(1-\frac{\epsilon}{24}\right)\left(1 + \frac{47\epsilon}{474}\right) \\ &= 1 + \frac{(47 \cdot 24 - 474)\epsilon - 47\epsilon^2}{24 \cdot 474} \geq 1 + \frac{607}{24 \cdot 474}\epsilon > 1 + \frac{1}{24}\epsilon.\end{aligned}$$

Die Wahrscheinlichkeit, dass die vorausgesetzten Schranken für $\hat{c}_2$ und $\hat{c}_3$ nicht eingehalten werden, ist dabei nach der Union-Bound höchstens $e^{-600} + e^{-7} < e^{-6}$.

Wir betrachten nun den Fall $c_2 < \frac{1}{16}l$. Da $c_3 \leq l$ nach Ungleichung (2), gilt

$$\hat{r} = \frac{\hat{c}}{\hat{k}} \geq \frac{\hat{c}_3}{\hat{k}} \geq \frac{\left(1 - \frac{\epsilon}{44}\right) \cdot \frac{11}{4}l}{c_2 + \frac{5}{2}c_3} = \left(1 - \frac{\epsilon}{44}\right) \cdot \frac{\frac{11}{4}l}{\frac{1}{16}l + \frac{5}{2}l} = \left(1 - \frac{\epsilon}{44}\right) \cdot \frac{44}{41}$$
$$= \frac{44 - \epsilon}{41} \geq \frac{41 + 2\epsilon}{41} > 1 + \frac{1}{24}\epsilon,$$

mit Wahrscheinlichkeit mindestens $e^{-100} < e^{-6}$.

Falls TESTE3STERNFREIHEIT die Überprüfung in der vorletzten Zeile erreicht, gilt also $\hat{r} > 1 + \frac{1}{24}\epsilon$ und der Algorithmus lehnt mit Wahrscheinlichkeit mindestens $1 - e^{-6}$ ab. Die einzige Möglichkeit für den Algorithmus, vorher mit *true* abzubrechen, ist, wenn in der drittletzten Zeile $\hat{c} \leq \frac{3}{2}\epsilon n$ gilt. Wir haben für die obigen Überlegungen allerdings schon den Eintritt des Ereignisses $\hat{c}_3 \geq \left(1 - \frac{\epsilon}{24}\right) \frac{11}{4}l$ vorausgesetzt; es gilt wegen Ungleichung (3)

$$\hat{c} \geq \hat{c}_3 \geq \left(1 - \frac{\epsilon}{24}\right) \frac{11}{4}l \geq \frac{47}{48} \cdot \frac{11}{4}\epsilon n \geq 2\epsilon n,$$

und damit akzeptiert TESTE3STERNFREIHEIT in der drittletzten Zeile nicht.

Damit ist die Gesamtwahrscheinlichkeit, G richtigerweise abzulehnen, mindestens $1 - e^{-6} > \frac{5}{6}$. □

Es bleibt zu zeigen, dass TESTE3STERNFREIHEIT auch für 3-Stern-freie Graphen G korrekt arbeitet.

Lemma 3.13. *Sei $G = (V, E)$ ein gerichteter 3-Stern-freier Graph mit n Knoten und durch $D \in \mathbb{N}$ beschränktem ungerichtetem Knotengrad; sei $\epsilon < 1$ ein Näheparameter. G habe keine Doppelkanten. Dann liefert* TESTE3STERNFREIHEIT*(n, G, ϵ) mit Wahrscheinlichkeit mindestens $\frac{3}{4}$* true *zurück.*

Beweis. TESTE3STERNFREIHEIT kann die Eingabe an drei Stellen irrtümlich ablehnen: Falls die Kantenzahl in der ersten Zeile deutlich überschätzt wird, falls das Kantensample mehr als $\frac{2048D\sqrt{n}}{\epsilon^{3/2}}$ Kanten enthält und falls der Schätzwert $\hat{r}$ größer als $1 + \frac{\epsilon}{24}$ ist. Die Tests auf Subgraphfreiheit in Zeile drei können nicht fehlschlagen, da TESTESUBGRAPHFREIHEITVERSTÄRKT ein Algorithmus mit einseitigem Fehler ist.

Die Kantenzahl von G ist höchstens n, da G zusammenhängend ist und keinen 3-Sternknoten enthält. Die Wahrscheinlichkeit, dass SCHÄTZEKANTENZAHL$(n, G, \epsilon/16)$ einen Wert größer als $n + \frac{\epsilon}{16}n$ zurückliefert, ist nach Lemma 3.8 höchstens $2e^{-4}$.

Die erwartete Anzahl von Kanten, die in der achten Zeile des Algorithmus gesampelt werden, ist $pn = \frac{128D\sqrt{n}}{\epsilon^{3/2}}$; mit Hilfe der Markowungleichung lässt sich die Wahrscheinlichkeit, dass das Kantensample mehr als $\frac{2048D\sqrt{n}}{\epsilon^{3/2}} = 16pn$ Kanten enthält, durch $\frac{1}{16}$ beschränken.

Es bleibt zu zeigen, dass der Schätzwert $\hat{r}$ mit hoher Wahrscheinlichkeit $1 + \frac{\epsilon}{24}$ nicht übersteigt. Sei k die Anzahl ausgehender 2-Sternknoten in G und c die Anzahl eingehender 2-Sternknoten. Nach Lemma 3.9 gilt $\hat{k} \geq k$ mit Wahrscheinlichkeit mindestens $1 - 2e^{-16}$, und da G die Orientierung eines kreis- oder linienförmigen Graphen ist, ist seine

Balance nach Lemma 3.3 beziehungsweise Lemma 3.5 höchstens 1; es gilt also $\hat{k} \geq c - 1$ mit hoher Wahrscheinlichkeit.

Sei C_2 die Menge eingehender 2-Sternknoten und sei X_v für alle diese Knoten v eine Indikatorzufallsvariable für das Ereignis, dass beide eingehenden Kanten gesampelt werden; sei $X := \sum_{v \in C_2} X_v$. Es gilt $\mathrm{E}[X_v] = \Pr[X_v = 1] = p^2$. Für den Erwartungswert von $\hat{c}$ folgt daher

$$\mathrm{E}[\hat{c}] = \frac{1}{p^2} X = \frac{1}{p^2} \sum_{v \in C_2} \mathrm{E}[X_v] = |C_2| = c.$$

Wir nehmen zunächst $c \geq \frac{\epsilon}{4} n$ an. Eine multiplikative Chernoffschranke liefert uns

$$\begin{aligned} \Pr\left[\hat{c} \geq \left(1 + \frac{\epsilon}{32}\right) c\right] &= \Pr\left[X \geq \left(1 + \frac{\epsilon}{32}\right) E[X]\right] \leq \exp\left(-\frac{\epsilon^2 p^2 c}{3 \cdot 1024}\right) \\ &= \exp\left(-\frac{128^2 \epsilon^3 D^2 n}{3 \cdot 4 \cdot 1024 \epsilon^3 n}\right) = e^{-2D^2/3} \leq e^{-6}. \end{aligned}$$

Damit gilt für $\hat{r}$

$$\hat{r} \leq \frac{\left(1 + \frac{\epsilon}{32}\right) c}{c - 1} = \left(1 + \frac{\epsilon}{24}\right) + \frac{\left(1 + \frac{\epsilon}{24}\right) - \frac{\epsilon}{96} c}{c - 1} \leq \left(1 + \frac{\epsilon}{24}\right) + \frac{\frac{25}{24} - \frac{\epsilon}{96 \cdot 4} \epsilon n}{c - 1},$$

und der hintere Summand des letzten Terms wird kleiner oder gleich null, wenn $n \geq \frac{400}{\epsilon^2}$ gilt, was aufgrund der Mindestgröße von n angenommen werden kann; damit gilt $\hat{r} \leq 1 + \frac{\epsilon}{24}$ mit Wahrscheinlichkeit mindestens $1 - e^{-6} - 2e^{-16} \geq \frac{31}{32}$.

Im Fall $c < \frac{\epsilon}{4}$ gilt aufgrund der Markow-Ungleichung $\Pr\left[\hat{c} > \frac{3}{2} \epsilon n\right] = \Pr\left[\hat{c} > 6\mathrm{E}[c]\right] \leq \frac{1}{6}$, und mit Wahrscheinlichkeit $\frac{5}{6}$ akzeptiert TESTE3STERNFREIHEIT in der drittletzten Zeile.

Damit ist die Wahrscheinlichkeit, dass TESTE3STERNFREIHEIT(n, G, ϵ) *false* zurückliefert, nach der Union-Bound höchstens $2e^{-4} + \frac{1}{32} + \frac{1}{6} < \frac{1}{4}$.

□

Aus den Lemmas 3.11, 3.12 und 3.13 folgt direkt die Korrektheit von TESTE3STERNFREIHEIT. Die Anfragekomplexität wird dominiert durch die Größe des Kantensamples, die maximal $\mathcal{O}\left(\frac{\sqrt{n}}{\epsilon^{3/2}}\right)$ ist. Alle weiteren Operationen im Algorithmus lassen sich in Laufzeit $\mathcal{O}\left(\frac{\sqrt{n}}{\epsilon^{3/2}}\right)$ durchführen.

Theorem 3.2. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und durch $D \in \mathbb{N}$ beschränktem ungerichtetem Knotengrad und sei $\epsilon < 1$ ein Näheparameter. G habe keine Doppelkanten.*

Dann gibt TESTE3STERNFREIHEIT*(n, G, ϵ) mit Wahrscheinlichkeit mindestens $\frac{3}{4}$* true *zurück, falls G 3-Stern-frei ist und mit Wahrscheinlichkeit mindestens $\frac{5}{6}$* false*, falls G ϵ-fern von 3-Stern-frei ist. Die Anfragekomplexität ist $\mathcal{O}\left(\frac{\sqrt{n}}{\epsilon^{3/2}}\right)$.*

Wir kommen zum Schluss wieder auf die eingangs besprochene Voraussetzung zurück, dass der eingegebene Graph G keine Doppelkanten haben darf. Nehmen wir nun an, G

habe Doppelkanten: Sei G' der Graph, der sich ergibt, wenn man in G von jeder Doppelkante diejenige Kante löscht, deren Ausgangsknoten die höhere Knotennummer hat als der Zielknoten – dies ist pro Doppelkante genau eine Kante, so dass G' keine Doppelkanten mehr enthält, aber gleichzeitig die Anzahl der Nachbarknoten jedes Knotens von G' gleich der des entsprechenden Knotens in G ist. Also ist G' genau dann 3-Stern-frei, wenn G 3-Stern-frei ist, und genau dann ϵ-entfernt von 3-Stern-frei, wenn auch G ϵ-entfernt von 3-Stern-frei ist. G' ist des Weiteren lokal aus G berechenbar: Für eine Kante (u, v) muss lediglich überprüft werden, ob es die entsprechende Kante in anderer Richtung ebenfalls gibt; ist dies der Fall und u hat eine größere Knotennummer als v, so wird die Kante als nicht existent angenommen. Falls die Kante gesampelt werden sollte, wird dann das Sample verworfen und stattdessen eine neue Kante gezogen.

Wir führen nun den Algorithmus TESTE3STERNFREIHEIT auf G' aus, indem wir diesen Graphen bei Bedarf lokal aus G konstruieren. Die erwartete Anzahl von Sampleversuchen für eine tatsächlich gezogene Kante ist dabei höchstens 2, und mit Hilfe der Markow-Ungleichung ergibt sich, dass die Gesamtzahl der zu sampelnden Kanten mit sehr hoher Wahrscheinlichkeit höchstens um einen konstanten Faktor erhöht – andernfalls kann der Algorithmus mit Rückgabe von *true* abbrechen, was eine geringe zusätzliche Fehlerwahrscheinlichkeit im Falle eines von 3-Stern-Freiheit ϵ-entfernten Eingabegraphen hinzufügt.

Korollar 3.2. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und durch $D \in \mathbb{N}$ beschränktem ungerichtetem Knotengrad und sei $\epsilon < 1$ ein Näheparameter.*

Dann gibt es einen Algorithmus, der mit Wahrscheinlichkeit mindestens $\frac{3}{4}$ true *zurückgibt, falls G 3-Stern-frei ist und mit Wahrscheinlichkeit mindestens $\frac{3}{4}$* false*, falls G ϵ-fern von 3-Stern-frei ist. Die Anfragekomplexität ist $\mathcal{O}\left(\frac{\sqrt{n}}{\epsilon^{3/2}}\right)$.*

3.3 Untere Schranken für 3-Stern-Freiheit

In diesem Abschnitt geben wir zwei untere Schranken für 3-Stern-Freiheit an: Eine Schranke von $\Omega(\sqrt{n})$ Anfragen für jeden Property-Testing-Algorithmus für 3-Stern-Freiheit in schwach zusammenhängenden Graphen, und eine Schranke von $\Omega(n^{2/3})$ Anfragen in Graphen, bei denen schwacher Zusammenhang nicht vorausgesetzt ist. Das bedeutet erstens, dass der im vorhergehenden Abschnitt angegebene Algorithmus für das Testen von 3-Stern-Freiheit in Bezug auf die Anzahl n der Knoten des Graphen asymptotisch optimal ist; zweitens bedeutet es, dass wir mit 3-Stern-Freiheit ein Problem identifiziert haben, das sich einfacher testen lässt, falls der Eingabegraph schwach zusammenhängend ist. Eine daraus resultierende interessante Frage ist, ob dies für weitere Probleme gilt.

Für die erste untere Schranke, die von $\Omega(\sqrt{n})$ für schwach zusammenhängende Graphen, verweisen wir auf [14] – die dort gegebene Schranke für starken Zusammenhang ist auch gleichzeitig eine für 3-Stern-Freiheit (siehe Kapitel 2.2.3 für eine detaillierte Diskussion des Originalergebnisses): Die Klasse stark zusammenhängender Graphen besteht aus gerichteten Kreisen, bei denen alle Kanten die gleiche Richtung haben; solche Graphen sind 3-Stern-frei. Die Klasse von Graphen, die ϵ-fern von stark zusammenhängend sind, besteht aus gerichteten Kreisen und zusätzlich mehr als ϵDn Außenknoten, die je eine Kante zu einem Kreisknoten haben; diese Graphen sind ϵ-fern von 3-Stern-frei: Jeder der

Kreisknoten, der eine zusätzliche Kante von einem der Außenknoten hat, ist ein 3-Stern, und es gibt mehr als ϵDn solche Knoten. Bender und Ron zeigen, dass diese beiden Graphklassen nicht mit $o(\sqrt{n})$ Anfragen zu unterscheiden sind, und daraus folgt direkt der folgende Korollar:

Korollar 3.3. *Im Adjazenzlistenmodell ohne Sichtbarkeit der eingehenden Kanten von Knoten benötigt jeder Property-Testing-Algorithmus für 3-Stern-Freiheit $\Omega(\sqrt{n})$ Anfragen, wobei n die Anzahl der Knoten des Eingabegraphen ist. Dies gilt auch, wenn die Menge der Eingabegraphen auf schwach zusammenhängende Graphen beschränkt ist.*

Im Folgenden werden wir eine untere Schranke von $\Omega(n^{2/3})$ für die Anfragekomplexität jedes Property-Testing-Algorithmus für 3-Stern-Freiheit im betrachteten Modell herleiten, die dann gilt, wenn die Menge möglicher Eingabegraphen nicht auf schwach zusammenhängende Graphen beschränkt ist. Im Prinzip wollen wir uns die Ergebnisse von [64] zunutze machen und zwei Probleminstanzen konstruieren – eine 3-Stern-freie, eine ϵ-entfernte –, für die die Verteilungen der eingehenden Knotengrade zwei proportionale Momente haben. Daraus würde dann folgen, dass jeder Algorithmus, der einen signifikanten Unterschied zwischen diesen Verteilungen feststellen kann, $\Omega(n^{2/3})$ Anfragen benötigt.

Das Problem dabei ist, zu zeigen, dass ein Algorithmus tatsächlich keine anderen Möglichkeiten hat, als diese Verteilungen zu analysieren. Da dies auch bei Graphen schwierig zu argumentieren ist, die nur aus isolierten k-Sternen, $k \leq 3$, bestehen, werden wir ein Zwischenproblem definieren, das stark an das Unterschiedliche-Elemente-Problem aus [64] angelehnt ist. Für dieses Problem werden wir die Techniken aus [64] anwenden; durch Reduktion des Zwischenproblems auf 3-Stern-Freiheit ergibt sich dann, dass die Schranke für die Anfragekomplexität auch für letzteres Problem gilt.

Das Zwischenproblem ist wie folgt definiert: Gegeben ist eine Folge A von m ganzen Zahlen $A_i \in \mathbb{N}, i = 1, \ldots m$. Dabei kommt jeder Wert bei maximal drei Gliedern der Folge vor. Sei l die Anzahl der unterschiedlichen Werte; wir nehmen an, dass die k verwendeten Werte aus $\{1, \ldots, l\}$ stammen. Die Frage ist nun, ob es mindestens einen Wert gibt, der bei drei Gliedern der Folge vorkommt oder nicht – in letzterem Fall nennen wir die Folge 3-Wert-frei. Es sei bemerkt, dass eine Probleminstanz von 3-Wert-Freiheit vollständig durch die zugehörige Folge A charakterisiert ist.

Im Kontext des Property-Testing nennen wir eine Folge ϵ-entfernt von 3-Wert-frei, falls mehr als ϵm Folgenglieder geändert werden müssen, um 3-Wert-Freiheit herzustellen; bei einer solchen Änderung dürfen den Folgegliedern auch Werte zugewiesen werden, die bisher nicht vergeben sind. Ein Algorithmus kennt die Länge m der Folge und kann den Wert des i-ten Folgenglieds anfragen, $i = 1, \ldots, m$.

Ein Poisson-s-Algorithmus ist ein Algorithmus, der zunächst mit Hilfe der Poisson-Verteilung mit Erwartungswert s zufällig bestimmt, wie viele Samples gezogen werden sollen. Die Analyse in [64] bezieht sich aus Poisson-s-Algorithmen, die nur die Histogramme ihrer zufälligen Stichproben erhalten, um eine Entscheidung zu treffen; letztere Einschränkung bedeutet, dass diese Algorithmen lediglich die Information erhalten, wie viele Werte in der Stichprobe einmal, zweimal etc. gezogen wurden, nicht aber die Nummern der Folgenelemente oder die Werte selbst:

Definition 3.3. *Sei S eine Multimenge. Das Histogramm $\mathcal{H}$ von S ist eine Funktion, die jeder natürlichen Zahl i die Anzahl der Elemente von S zuordnet, die genau i mal in S vorkommen; das heißt*

$$\mathcal{H}(i) := |\{s \in S | s \text{ ist genau } i \text{ mal in } S \text{ enthalten}\}| .$$

Wir benutzen zwei der in [64] verwendeten Hilfssätze, um zu zeigen, dass es auch für eine untere Schranke für das Testen von 3-Wert-Freiheit genügt, mit Pois-s-Algorithmen zu argumentieren, die nur das Histogramm ihres Samples erhalten. Das folgende Lemma ist ein direkter Korollar aus Lemma 3.1[2] und aus Lemma 5.3 a) und c) in [64]:

Lemma 3.14. *Sei $\mathcal{A}$ ein beliebiger samplingbasierter Algorithmus, der t Elemente der Folge A zieht; $\mathcal{A}$ akzeptiere mit Wahrscheinlichkeit mindestens $\frac{3}{4}$, falls A 3-Wert-frei ist und lehne mit Wahrscheinlichkeit mindestens $\frac{3}{4}$ ab, falls A ϵ-entfernt von 3-Wert-frei ist.*

Dann existiert ein Poisson-s-Algorithmus $\mathcal{A}'$, der im Erwartungswert $s = O(t)$ Elemente aus A zieht und nur das Histogramm dieser Samples erhält und der mit Wahrscheinlichkeit mindestens $\frac{3}{4} - o(1)$ akzeptiert, falls A 3-Wert-frei ist und mit Wahrscheinlichkeit mindestens $\frac{3}{4} - o(1)$ ablehnt, falls A ϵ-entfernt von 3-Wert-frei ist.

Es sei darauf hingewiesen, dass die Eigenschaften der 3-Wert-Freiheit einer Folge A sowie der ϵ-Entferntheit davon abgeschlossen unter Permutation der Folgenglieder und der möglichen Werte sind und damit den Prämissen den Lemmas 3.1 und 5.3 c) in [64] genügen. Die Mindestwahrscheinlichkeiten für $\mathcal{A}'$ folgen aus der von Lemma 5.3 a) in [64] garantierten, durch $\frac{4}{s} = o(1)$ beschränkten[3] statistischen Distanz zwischen den Verteilungen der Ergebnisse von $\mathcal{A}$ und $\mathcal{A}'$.

Sei nun analog zu [64] eine Frequenzvariable X_A definiert als eine Zufallsvariable für die Anzahl der Vorkommen eines zufällig gleichverteilt aus $\{1, \ldots, l\}$ gezogenen Wertes in A. Das zweite Lemma ist ebenfalls ein direkter Korollar aus der genannten Veröffentlichung von Raskhodnikova et al., nämlich aus Korollar 5.7:

Lemma 3.15. *Seien A und B Probleminstanzen von 3-Wert-Freiheit und seien X_A und X_B die zugehörigen Frequenzvariablen. Falls X_A und X_B $k-1$ proportionale Momente haben, dann gilt für jeden Poisson-s-Algorithmus $\mathcal{A}$ mit $s = o(n^{1-1/k})$, der nur die Histogramme seiner Samples sieht*

$$|\Pr[\mathcal{A}(A) = \text{true}] - \Pr[\mathcal{A}(B) = \text{true}]| = o(1).$$

Zwei Zufallsvariablen X_1 und X_2 haben $k-1$ proportionale Momente, wenn $\frac{\mathrm{E}[X_1]}{\mathrm{E}[X_2]} = \frac{\mathrm{E}[X_1^2]}{\mathrm{E}[X_2^2]} = \ldots = \frac{\mathrm{E}[X_1^{k-1}]}{\mathrm{E}[X_2^{k-1}]}$ gilt. Wir können nun beweisen, dass jeder Property-Testing-Algorithmus für 3-Wert-Freiheit mindestens $\Omega(n^{2/3})$ Samples benötigt.

Lemma 3.16. *Jeder Property-Testing-Algorithmus für 3-Wert-Freiheit benötigt $\Omega(n^{2/3})$ Samples, wobei n die Länge der Folge A ist.*

[2] Dieses Lemma zitieren Raskhodnikova et al. aus [10].

[3] Aufgrund von Korollar 3.3 können wir $s = \Omega(\sqrt{n})$ annehmen, daher gilt $\frac{4}{s} = o(1)$.

Beweis. Wir bemerken zunächst, dass aus Lemma 3.14 folgt, dass jeder Property-Testing-Algorithmus für 3-Wert-Freiheit durch einen asymptotisch gleich schnellen Poisson-s-Algorithmus simuliert werden kann, der nur Zugriff auf ein Histogramm seiner Samples erhält; für hinreichend große n ist die zusätzliche Versagenswahrscheinlichkeit gering. Also reicht es, zu zeigen, dass jeder solche Poisson-s-Algorithmus im Erwartungswert mindestens $s = \Omega(n^{2/3})$ Anfragen benötigt. Weiter folgt aus Lemma 3.15, dass jeder solche Poisson-s-Algorithmus mindestens $\Omega(n^{2/3})$ Anfragen benötigt, um zwei Probleminstanzen A und B zu unterscheiden, deren Frequenzvariablen X_A und X_A zwei proportionale Momente haben. Wir werden nun zwei Klassen $\mathcal{C}_A$ und $\mathcal{C}_B$ solcher Instanzen angeben, wobei alle $A \in \mathcal{C}_A$ 3-Wert-frei sind und alle $B \in \mathcal{C}_B$ ϵ-entfernt davon. Daraus folgt, dass jeder Property-Testing-Algorithmus für 3-Wert-Freiheit mindestens $\Omega(n^{2/3})$ Anfragen benötigt.

Wir definieren $\mathcal{C}_A$ als die Klasse aller Folgen A mit n Folgengliedern, so dass n ein Vielfaches von 32 ist und A $\frac{1}{2}n$ Werte enthält, jeden davon genau zweimal. Jede Folge aus $\mathcal{C}_A$ ist 3-Wert-frei.

$\mathcal{C}_B$ definieren wir als die Klasse aller Folgen B mit n Folgengliedern, so dass n ein Vielfaches von 32 ist und B folgende Verteilung von Werten hat: Von den insgesamt $\frac{17}{32}n$ vorkommenden Werten kommen $\frac{1}{32}n$ Werte dreimal vor, $\frac{13}{32}n$ Werte zweimal und $\frac{3}{32}n$ Werte einmal. Für $\epsilon < \frac{1}{32}$ sind alle Folgen in $\mathcal{C}_B$ ϵ-entfernt von 3-Wert-frei.

Es bleibt zu zeigen, dass die Frequenzvariablen gleich langer Folgen aus $\mathcal{C}_A$ und $\mathcal{C}_B$ zwei proportionale Momente haben. Seien für ein beliebiges, gültiges n $A \in \mathcal{C}_A$ und $B \in \mathcal{C}_B$ zwei feste aber beliebige Folgen der Länge n und seien X_A und X_B die entsprechenden Frequenzvariablen. Es gilt $\mathrm{E}[X_A] = 2$ und $\mathrm{E}[X_A^2] = 4$. Weiter gilt

$$\mathrm{E}[X_B] = \frac{1}{17} \cdot 3 + \frac{13}{17} \cdot 2 + \frac{3}{17} \cdot 1 = \frac{32}{17}$$

und

$$\mathrm{E}[X_B] = \frac{1}{17} \cdot 9 + \frac{13}{17} \cdot 4 + \frac{3}{17} \cdot 1 = \frac{64}{17}.$$

Es folgt

$$\frac{\mathrm{E}[X_A]}{\mathrm{E}[X_B]} = \frac{2 \cdot 17}{32} = \frac{4 \cdot 17}{64} = \frac{\mathrm{E}[X_A^2]}{\mathrm{E}[X_B^2]},$$

und damit haben X_A und X_B zwei proportionale Momente. □

Wir werden nun durch Reduktion zeigen, dass jeder Property-Testing-Algorithmus für 3-Stern-Freiheit $\Omega(n^{2/3})$ Anfragen benötigt, wenn dies auch eine untere Schranke für das Testen von 3-Wert-Freiheit ist.

Theorem 3.3. *Im Adjazenzlistenmodell ohne Sichtbarkeit der eingehenden Kanten von Knoten benötigt jeder Property-Testing-Algorithmus für 3-Stern-Freiheit $\Omega(n^{2/3})$ Anfragen, wobei n die Anzahl der Knoten des Eingabegraphen ist.*

Beweis. Wir nehmen für einen indirekten Beweis an, es gebe einen Property-Testing-Algorithmus $\mathcal{A}$ für 3-Sternfreiheit, der für Graphen mit n Knoten $o(n^{2/3})$ Anfragen benötigt. Wir konstruieren nun einen Algorithmus $\mathcal{A}'$, der als Eingabe eine Folge A der

Länge $n' = \frac{1}{2}n$ Werten enthält, die eine Probleminstanz von 3-Wert-Freiheit ist, und einen Näheparameter ϵ'.

$\mathcal{A}'$ ruft $\mathcal{A}$ auf und gibt den Rückgabewert von $\mathcal{A}$ zurück. $\mathcal{A}$ erhält den Näheparameter $\epsilon = 2\epsilon'$ und Orakelzugriff auf den folgenden Graphen $G = (V, E)$ mit $n = 2n'$ Knoten:

- V enthält n' Zentralknoten $v_1, \ldots, v_{n'}$ und n' Außenknoten $u_1, \ldots, u_{n'}$.
- Für alle $i = 1, \ldots, n'$, hat u_i eine Kante zu v_{A_i}; das heißt, die einem bestimmten Wert zugeordneten Außenknoten teilen sich in G einen gemeinsamen Zentralknoten.
- G enthält keine weiteren Knoten, so dass möglicherweise isolierte Knoten in G existieren.

Die Knoten u_i repräsentieren in G also die Folgenglieder von A, und über einen gemeinsamen Zentralknoten wird abgebildet, dass mehrere Folgenglieder den gleichen Wert haben.

Für eine Anfrage von $\mathcal{A}$ kann G lokal konstruiert werden[4]: Wird ein noch unbekannter Knoten angefragt, so wird zunächst per Münzwurf bestimmt, ob der angefragte Knoten ein Zentralknoten oder ein Außenknoten ist; die Wahrscheinlichkeit für einen Zentralknoten ist $\frac{n'-z}{2n'-z-a}$, wenn bisher z Zentral- und a Außenknoten bekannt sind. Ist der angefragte Knoten u ein Außenknoten, dann wird zufällig gleichverteilt eins der noch nicht bekannten Folgenglieder von A angefragt; die Wahrscheinlichkeit für jedes der verbliebenen Folgenglieder ist dabei $\frac{1}{n'-a}$, wenn a wiederum die Anzahl zuvor schon angefragter Außenknoten ist. Im Grunde wird also ohne Zurücklegen aus A gesampelt.

Sei nun i die Nummer des derart für einen neuen Außenknoten ermittelten Folgenglieds: Es wird eine Kante (u, v_{A_i}) in den Graphen eingefügt. Da $\mathcal{A}$ nur ausgehende Kanten von Knoten anfragen kann, keine eingehenden, kann so jede von $\mathcal{A}$ angefragte Kante konstruiert werden.

Da diese Prozedur pro Anfrage von $\mathcal{A}$ höchstens eine Anfrage an A durchführt, ist die Anzahl der Anfragen $o(n^{2/3}) = o((n')^{2/3})$. Des Weiteren erzeugt jeder Wert, der i mal in A vorkommt, einen i-Stern in G, und G enthält keine weiteren Sternknoten; also ist G genau dann 3-Stern-frei, wenn A 3-Wert-frei ist. Da $n = 2n'$, ist G zudem genau dann ϵ-entfernt von 3-Stern-frei, wenn A ϵ'-entfernt von 3-Wert-frei ist. Damit ist $\mathcal{A}'$ ein Property-Testing-Algorithmus für 3-Wert-Freiheit mit einer Anfragekomplexität von $o((n')^{2/3})$ – ein Widerspruch zu Lemma 3.16. Also existiert ein Algorithmus $\mathcal{A}$ wie oben angenommen nicht. □

Schlussendlich sei noch die nötige Anfragekomplexität für k-Stern-Freiheit mit $k > 3$ diskutiert. Das Problem der k-Wert-Freiheit lässt sich analog zu dem der 3-Wert-Freiheit für größere k definieren, und die Lemmas 3.14 und 3.15 gelten analog auch für dieses Problem; auch die Reduktion auf k-Stern-Freiheit lässt sich analog zum Fall $k = 3$ durchführen. Die Schwierigkeit besteht darin, den Beweis zu Lemma 3.16 anzupassen, genauer, für die Graphen der beiden Klassen $\mathcal{C}_A$ und $\mathcal{C}_B$ Häufigkeitsverteilungen so zu wählen, dass

[4] Dieses Verfahren erzeugt möglicherweise eine andere als die oben der Klarheit halber angegebene Permutation der Knoten; der Graph ist jedoch isomorph zu G.

die entsprechenden Frequenzvariablen $k-1$ proportionale Momente besitzen. Dies lässt sich für festes k leicht durchführen, allerdings ist die Frage nach einer geschlossenen, von k abhängigen Formel für solche Verteilungen offen. Das führt zu folgender Vermutung:

Vermutung 3.1. *Im Adjazenzlistenmodell ohne Sichtbarkeit der eingehenden Kanten von Knoten benötigt jeder Property-Testing-Algorithmus für k-Stern-Freiheit* $\Omega(n^{1-1/k})$ *Anfragen, wobei n die Anzahl der Knoten des Eingabegraphen ist.*

4 Testen von starkem Zusammenhang

Eine sehr grundlegende Eigenschaft gerichteter Graphen ist starker Zusammenhang, also die Frage, ob alle Paare von Knoten eines Graphen in beiden Richtungen mit Pfaden verbunden sind. Wie in Kapitel 2 schon erwähnt, wurde diese Eigenschaft im Modell dünn besetzter gerichteter Graphen zuerst untersucht von Bender und Ron [14]; desweiteren existiert, wie im Einleitungskapitel ausgeführt, eine Beweisskizze von Goldreich zu einem dem hier vorgestellten sehr ähnlichen Algorithmus [39]; Unterschiede zu und Gemeinsamkeiten mit dieser werden am Ende des Kapitels diskutiert.

Eins der Ergebnisse von Bender und Ron ist ein Property-Testing-Algorithmus mit konstanter Anfragekomplexität in einem einfacheren Modell, in dem auch die eingehenden Kanten von Knoten angefragt werden dürfen, das andere Ergebnis ist eine untere Schranke von $\Omega(\sqrt{n})$ für Property-Testing-Algorithmen im auch von uns untersuchten Modell nicht sichtbarer eingehender Kanten.

Ihre Konstruktion – genauer gesagt die Klasse der von starkem Zusammenhang ϵ-entfernten Graphen – lässt sich leicht für eine untere Schranke von $\Omega(n)$ Anfragen für Property-Testing-Algorithmen mit einseitigem Fehler verwenden: Für $n \in \mathbb{N}$ bestehe G aus einem Kreis von $n-\epsilon Dn-1$ Knoten, in dem die Kanten im Uhrzeigersinn angeordnet sind; ansonsten gebe es $\epsilon Dn+1$ weitere Knoten, die je eine Kante zu einem anderen der Kreisknoten haben (siehe Abbildung 4.1). G ist ϵ-fern von stark zusammenhängend, denn jeder der Außenknoten muss mit einer eigenen eingehenden Kante angebunden werden; für jeden Subgraphen H von G mit höchstens $n-\epsilon Dn-2$ Knoten gibt es aber einen stark zusammenhängenden Graphen G', so dass H auch Subgraph von G' ist: H enthält mindestens $\epsilon Dn+2$ Knoten von G nicht, so dass in G' $\epsilon Dn+1$ dieser Knoten je eine Kante zu einem der Außenknoten besitzen können; da mindestens ein Kreisknoten nicht entdeckt ist, kann mindestens eine der Kanten einen Kreis- mit einem Außenknoten verbinden, der so in den Kreis integriert wird. Sind keine Kreisknoten mehr frei, können die restlichen Außenknoten sukzessive an bereits angebundene Kreisknoten angebunden werden (siehe wieder Abbildung 4.1). Dadurch ist G' stark zusammenhängend, und kein Algorithmus, der höchstens $n-\epsilon Dn-2$ Knoten anfragt, kann ausschließen, dass G stark zusammenhängend ist; ein solcher Algorithmus mit einseitigem Fehler müsste G also akzeptieren.

Korollar 4.1. *Sei G ein gerichteter Graph mit n Knoten. Jeder Property-Testing-Algorithmus mit einseitigem Fehler, der nur ausgehende Kanten von Knoten von G anfragt, hat eine Anfragekomplexität von $\Omega(n)$.*

Wir werden in diesem Kapitel einen Property-Testing-Algorithmus mit zweiseitigem Fehler und einer Anfragekomplexität von $\mathcal{O}(n^{1-\epsilon/(3+\alpha)})$ erarbeiten. $\alpha > 0$ ist dabei eine beliebig kleine Konstante. Wir setzen dafür voraus, dass sowohl der Eingangs- als auch

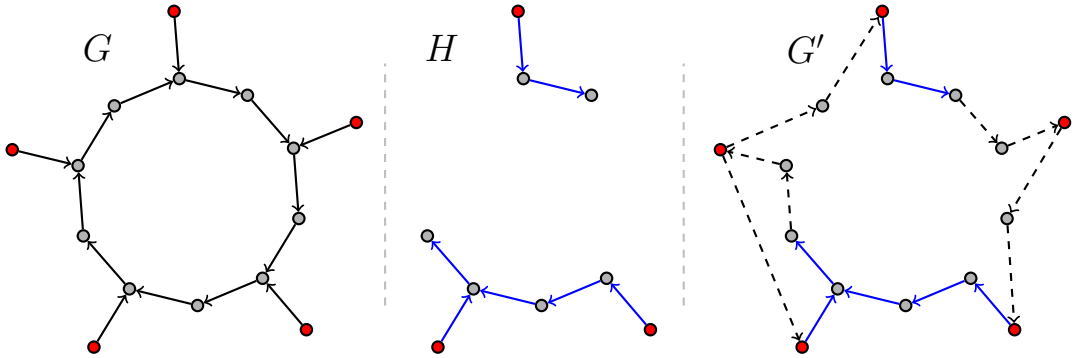

Abbildung 4.1: *Graph G, beispielhafter erforschter Graph H und daraus konstruierter stark zusammenhängender Graph G'*

der Ausgangsgrad jedes Knotens durch eine Konstante $D \in \mathbb{N}$ beschränkt ist. Ein Knoten kann also maximal $2D$ inzidente Kanten haben, von denen je maximal D eingehend und D ausgehend sind. Insgesamt kann es weiterhin höchstens ϵDn Kanten geben, so dass ein Graph weiterhin ϵ-fern von einem anderen ist, wenn mehr als ϵDn Adjazenzlisteneinträge des ersten geändert werden müssen, um den zweiten zu erhalten.

Wir können $D \geq 2$ annehmen, denn für $D = 1$ ist das Problem einfach zu lösen: Jeder Knoten kann in diesem Fall höchstens eine eingehende und eine ausgehende Kante haben; jede schwache Zusammenhangskomponente in einem solchen Graphen ist entweder eine Orientierung eines linien- oder eines kreisförmigen Graphen, und die Kanten innerhalb einer solchen Komponente haben jeweils die gleiche Orientierung. Der einzige stark zusammenhängende so aufgebaute Graph ist ein gerichteter Kreis über alle n Knoten. Jeder von starkem Zusammenhang ϵ-entfernte Graph besteht hingegen aus $\Omega(\epsilon n)$ schwachen Zusammenhangskomponenten, das heißt es gibt auch $\Omega(\epsilon n)$ schwache Zusammenhangskomponenten mit höchstens $\frac{1}{\epsilon}$ Knoten, und ein Knotensample der Größe $\Theta(\frac{1}{\epsilon})$ enthält daher mit hoher Wahrscheinlichkeit mindestens einen Knoten einer dieser Komponenten. Ist die betreffende Komponente ein gerichteter Kreis, so kann er in $\frac{1}{\epsilon}$ Anfragen komplett erkundet werden; handelt es sich um eine gerichtete Linie, so wird innerhalb von $\frac{1}{\epsilon}$ Anfragen eine Senke erreicht – beides folgt aus der Feststellung, dass alle Kanten der Komponente die gleiche Orientierung haben. Somit kann mit einer Anfragekomplexität von $\Theta(\frac{1}{\epsilon^2})$ für ϵ-entfernte Graphen mit hoher Wahrscheinlichkeit ein Zeuge gegen starken Zusammenhang identifiziert werden.

Korollar 4.2. *Im Fall $D = 1$ kann der starke Zusammenhang eines gerichteten Graphen mit durch jeweils D beschränktem Knotenein- und -ausgangsgrad in $\mathcal{O}(\frac{1}{\epsilon^2})$ Anfragen getestet werden.*

Für größere Werte von D ist das Testen auf starken Zusammenhang allerdings deutlich schwieriger – das Problem ist das Identifizieren von Zeugen: Ein Zeuge dafür, dass ein Graph G nicht stark zusammenhängend ist, ist eine starke Zusammenhangskomponente, die entweder nicht von anderen Knoten des Graphen aus erreicht werden kann oder keine ausgehenden Kanten hat; bei ersterem Typus handelt es sich um Quell-, bei zweiterem

```
TesteSenkenFreiheit(n, G, β)
    Sampele zufällig gleichverteilt s_4.1 = 4/(βD) Knoten in G
    Foreach gesampelte Knoten v do
        Führe Breitensuche von v ausgehend aus, die maximal 2/(βD) Knoten erkundet
        If die Breitensuche exploriert eine Senkenkomponente vollständig
                then return false
    return true
```

Algorithmus 4.1: TesteSenkenFreiheit

um Senkenkomponenten. Dass in einem Graphen, der ϵ-fern von stark zusammenhängend ist, die Gesamtzahl von endständigen Komponenten groß ist, haben bereits Bender und Ron bewiesen:

Lemma 4.1 ([14])**.** *Sei G ein gerichteter Graph mit jeweils durch $D \in \mathbb{N}$ beschränktem Ein- und Ausgangsgrad. Sei $\epsilon < 1$ ein Näheparameter. Falls G ϵ-fern von stark zusammenhängend ist, dann gibt es in G mehr als $\frac{1}{3}\epsilon Dn$ endständige Komponenten.*

Kleine Senkenkomponenten sind einfach zu identifizieren: Es genügt, einen Knoten in einer solchen Komponente zu sampeln und von dort aus via Breitensuche einen hinreichend großen Teil des Graphen zu explorieren; die Breitensuche wird dann mangels ausgehender Kanten abbrechen, wenn die komplette Senkenkomponente erforscht ist. Es lässt sich zeigen, dass, falls ein Graph G viele Senkenkomponenten hat, auch viele dieser Komponenten klein sind, und daher eine Samplingprozedur mit hoher Wahrscheinlichkeit eine solche kleine Senkenkomponente trifft; dies ist grundsätzlich die gleiche Vorgehensweise, die Bender und Ron für ihren Property-Testing-Algorithmus im Modell mit sichtbaren eingehenden Kanten verwenden:

Lemma 4.2. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und mit jeweils durch $D \in \mathbb{N}$ beschränktem Ein- und Ausgangsgrad. Sei $\beta < 1$ ein Näheparameter. Falls G keine Senkenkomponenten enthält, liefert* TesteSenkenFreiheit*(n, G, β)* true *zurück, falls G mindestens βDn Senkenkomponenten enthält, mit Wahrscheinlichkeit mindestens* $1 - e^{-2}$ false*; Laufzeit und Anfragekomplexität des Algorithmus sind jeweils in $\mathcal{O}\left(\frac{1}{\beta^2 D^2}\right)$.*

Beweis. Falls G keine Senkenkomponenten enthält, dann kann der Algorithmus nicht in der vorletzten Zeile *false* zurückgeben; damit ist der Rückgabewert richtigerweise *true*.

Wir nehmen nun an, dass G mindestens βDn Senkenkomponenten enthält. Da G n Knoten hat und jeder davon nur in einer Senkenkomponente liegen kann, haben mindestens $\frac{1}{2}\beta Dn$ Senkenkomponenten $\frac{2}{\beta D}$ oder weniger Knoten. Wird ein Knoten v in einer solchen Komponente gesampelt, dann exploriert die bei v startende Breitensuche die jeweilige Senkenkomponente vollständig und der Algorithmus liefert richtigerweise *false* zurück. Die Wahrscheinlichkeit, dass dies für keinen der gesampelten Knoten geschieht, ist höchstens $(1 - \frac{1}{2}\beta D)^{s_{4.1}} \leq e^{-2}$.

Die Laufzeit und Anfragekomplexität ergibt sich direkt aus der Anzahl der gezogenen Samples und der Knotenschranke der Breitensuche. □

SchätzeErreichbareKnoten(n, G, D, ϵ)
  for $i \leftarrow D$ **downto** 1 **do**
    Sampele jede Kante von G mit Wahrscheinlichkeit jeweils $p_i = a_i n^{-1/i}$,
    If mehr als $t_i = 16Da_i n^{1-1/i}$ Kanten gesampelt werden **then return** n
    $\hat{c}_i \leftarrow$ Anzahl der Knoten, von denen genau i eingehende Kanten gesampelt wurden
    $\hat{n}_i \leftarrow \frac{\hat{c}_i}{p_i^i} - \sum_{i<j\leq d} \binom{j}{i} \hat{n}_j (1-p_i)^{j-i}$
  return $\hat{m} = \frac{\epsilon D n}{16} + \sum_{1\leq i\leq D} \hat{n}_i$

Algorithmus 4.2: *SchätzeErreichbareKnoten*

Im Gegensatz zu Senkenkomponenten sind Quellkomponenten nicht einfach zu identifizieren: Eine in einer Quellkomponente gestartete Breitensuche kann die Komponente verlassen – außer, sie ist zugleich Quell- und Senkenkomponente – und beliebige weitere Bereiche des Graphen explorieren; mangels Sichtbarkeit eingehender Kanten kann mittels Breitensuche aber nicht nachgewiesen werden, dass eine starke Zusammenhangskomponente keine eingehenden Kanten hat, außer die übergeordnete schwache Zusammenhangskomponente – im schlimmsten Fall der gesamte Graph – wird komplett exploriert. Da in einem Graphen, der ϵ-fern von stark zusammenhängend ist, alle endständigen Komponenten Quellkomponenten sein können, muss ein Property-Testing-Algorithmus für starken Zusammenhang das Vorhandensein vieler Quellkomponenten jedoch nachweisen können.

Der hier vorgestellte Algorithmus wird diesen Nachweis indirekt führen: Wenn wir annehmen, dass ein Graph G viele Quellkomponenten enthält und jede davon aus genau einem Knoten besteht, dann gibt es in diesem Graphen viele Knoten ohne eingehende Kante. Diese Eigenschaft lässt sich statistisch nachweisen, indem zufällig Kanten gesampelt werden und für alle $2 \leq k \leq D$ die Anzahl der k-Fach-Kollisionen gemessen wird, also die Anzahl an Knoten, von denen genau k inzidente Kanten gesampelt wurden. Für ein hinreichend großes Kantensample lässt sich so für alle $k > 1$ abschätzen, wieviele Knoten von G einen Eingangsgrad von genau k haben; dieser Schätzwert von n subtrahiert ergibt wiederum eine Schätzung für die Anzahl der Knoten ohne eingehende Kanten. Überschreitet letztere einen bestimmten Schwellwert, so sollte der Graph als nicht stark zusammenhängend abgelehnt, ansonsten angenommen werden.

Dieses Vorgehen setzt voraus, dass alle – oder zumindest viele – Quellkomponenten mit einer Größe von einem Knoten existieren; wir werden eine Reduktion vorstellen, durch die jede hinreichend kleine Quellkomponente eines Graphen durch einen einzelnen Knoten ersetzt wird. Falls es im Ausgangsgraphen viele solcher kleinen Quellkomponenten gibt, enthält der entstehende Graph also viele Knoten mit Eingangsgrad 0, und das oben genannte Verfahren ist anwendbar. Zudem ist diese Reduktion lokal berechenbar: Wie wir zeigen werden, kann für einen beliebigen Knoten lokal berechnet werden, zu welchem Knoten im reduzierten Graphen er gehört, und es kann aus dem reduzierten Graphen nach üblichen Methoden gesampelt werden.

Wir reduzieren also das Problem, einen Graphen auf starken Zusammenhang zu testen,

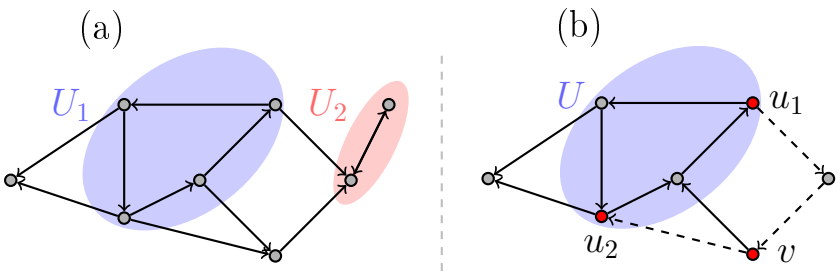

Abbildung 4.2: *(a) Kompakte Komponenten U_1, U_2; (b) für $\frac{3+\alpha}{\epsilon D} = 5$ ist U ist keine kompakte Komponente, der markierte Pfad verletzt die dritte Bedingung.*

auf das Problem, die Anzahl der Knoten ohne eingehende Kanten in einem Graphen zu approximieren. Das folgende Lemma geben wir zunächst nur an und beweisen es am Schluss des Kapitels[1]:

Lemma 4.3. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und mit jeweils durch $D \in \mathbb{N}$ beschränktem Ein- und Ausgangsgrad; G habe m Knoten mit Eingangsgrad größer oder gleich eins. Sei $\epsilon < 1$ ein Näheparameter.* SchätzeErreichbareKnoten*(n, G, ϵ) liefert einen Schätzwert $\hat{m}$ zurück, für den*

$$m \leq \hat{m} \leq m + \frac{\epsilon D n}{8}$$

mit Wahrscheinlichkeit mindestens $\frac{3}{4}$ gilt. Dafür sind höchstens $\mathcal{O}\left(\frac{D^{8D+15} \log D}{\epsilon^3} n^{1-1/D}\right)$ Anfragen an G nötig.

Die grundlegende Beweisstrategie ist, zunächst zu zeigen, dass für geeignete Wahl der a_i die Schätzer $\hat{n}_i$ mit hoher Wahrscheinlichkeit die Anzahl der Knoten mit i eingehenden Kanten approximieren (siehe Algorithmus 4.2). Mit Hilfe der Union-Bound lässt sich dann zeigen, dass $\hat{m}$ die im Lemma geforderten Schranken mit hoher Wahrscheinlichkeit einhält.

Wir werden nun die Reduktionsfunktion C vorstellen. Dafür benötigen wir den Begriff der *kompakten Komponente*; die Reduktion wird einen neuen Graphen erzeugen, in dem die kompakten Komponenten des Ausgangsgraphen kontrahiert sind:

Definition 4.1. *Sei $G = (V, E)$ ein gerichteter Graph mit jeweils durch D beschränktem Knoteneingangs- und -ausgangsgrad und sei ϵ ein Näheparameter; sei $\alpha > 0$ fest aber beliebig. Wir nennen eine Knotenmenge $U \subseteq V$ eine* kompakte Komponente*, falls folgende drei Bedingungen erfüllt sind:*

1. *$|U| \leq \frac{3+\alpha}{\epsilon D}$;*
2. *Der durch U induzierte Subgraph von G ist stark zusammenhängend;*

[1] Ein ähnliches Verfahren wie in SchätzeErreichbareKnoten für das Schätzen einer Verteilung aus mehreren i-Fach-Kollisionsstatistiken für alle möglichen i wird in [64] in Abschnitt 4, Fußnote 4 erwähnt, aber die Korrektheit wird dort nicht bewiesen.

3. *Es gibt keine Knoten in $v \in V \backslash U$ und $u_1, u_2 \in U$, so dass es Pfade von u_1 nach v und v nach u_2 gibt, deren Länge jeweils höchstens $\frac{3+\alpha}{\epsilon D}$ ist.*

Das folgende Lemma zeigt, dass die kompakte Komponente jedes Knotens eindeutig ist:

Lemma 4.4. *Seien $G = (V, E)$, D, ϵ und α wie in Definition 4.1. Dann gehört jeder Knoten von G zu höchstens einer kompakten Komponente.*

Beweis. Sei $v \in V$ ein beliebiger Knoten. Wir nehmen an, dass es zwei unterschiedliche kompakte Komponenten U, W gibt, so dass $v \in U$ und $v \in W$. Da $U \neq W$, gibt es mindestens einen Knoten, der in einer der Komponenten enthalten ist, in der anderen jedoch nicht – wir nehmen ohne Beschränkung der Allgemeinheit an, dass $u \in U \backslash W$. Da U eine kompakte Komponente ist, ist U stark zusammenhängend und enthält höchstens $\frac{3+\alpha}{\epsilon D}$ Knoten. Da $v, u \in U$, gibt es Pfade der Länge jeweils höchstens $\frac{3+\alpha}{\epsilon D}$ von v nach u und von u nach v. Da aber $v \in W$ und $u \notin W$, ist dies ein Widerspruch zur Annahme, dass W eine kompakte Komponente ist, denn die dritte Bedingung für kompakte Komponenten ist für W verletzt. □

Es ist möglich, dass Knoten eines Graphen nicht zu einer kompakten Komponente gehören; wir definieren, dass solche Knoten ihre eigene Komponente bilden. Zusammen mit dem obigen Lemma bedeutet das, dass jeder Knoten genau einer Komponente zugeordnet werden kann:

Definition 4.2. *Seien $G = (V, E)$, D, ϵ und α wie in Definition 4.1. Dann ist $C(v)$ für einen Knoten $v \in V$ die kompakte Komponente, in der v enthalten ist, bzw. es gilt $C(v) = \{v\}$, falls v in keiner kompakten Komponente gemäß Definition 4.1 enthalten ist.*

Wir definieren außerdem $C(G)$ als den Graphen, der entsteht, wenn jede kompakte Komponente von G kontrahiert wird.

Als nächstes zeigen wir, dass alle Knoten einer kompakten Komponente derselben kompakten Komponente zugeordnet sind.

Lemma 4.5. *Seien $G = (V, E)$, D, ϵ und α wie in Definition 4.1. Dann gilt $C(u) = C(v)$ für alle Knoten $v \in V$ und $u \in C(v)$.*

Beweis. Wir nehmen an, dass $|C(v)| > 1$ gilt, andernfalls gilt die Aussage wegen $C(v) = \{v\}$. Weiter nehmen wir an, dass es einen Knoten $u \in C(v)$ gibt mit $C(u) \neq C(v)$. Es gibt also einen Knoten, der entweder nur in $C(v)$ oder nur in $C(u)$ enthalten ist.

Wir nehmen zunächst an, dass es einen Knoten $w \in C(u) \backslash C(v)$ gibt: Dann können wir analog zu Lemma 4.4 die Existenz von Pfaden von u nach w und zurück folgern, die die dritte Bedingung kompakter Komponenten für $C(v)$ verletzen, und erhalten damit einen Widerspruch zur Annahme, $C(v)$ sei kompakte Komponente.

Wir nehmen nun an, dass es einen Knoten $w \in C(v) \backslash C(u)$ gibt. Analog zum ersten Fall gibt es dann Pfade von v nach w und von w nach v, die einen Widerspruch zur dritten Bedingung kompakter Komponenten für $C(u)$ darstellen. □

Damit ist nachgewiesen, dass die Abbildung C wohldefiniert ist. Die für uns wesentliche Eigenschaft dieser Abbildung ist, dass sie jeder kleinen Quellkomponente eines Graphen G eine eigene kompakte Komponente zuweist; in $C(G)$ wird jede dieser Komponenten also durch einen einzelnen Knoten ersetzt, und falls G viele kleine Quellkomponenten enthält, dann enthält $C(G)$ viele Knoten ohne eingehende Kanten.

Beobachtung 4.1. *Seien $G = (V, E)$, D, ϵ und α wie in Definition 4.1 und sei $U \subseteq V$ eine Quellkomponente mit höchstens $\frac{3+\alpha}{\epsilon D}$ Knoten. Dann ist U eine kompakte Komponente.*

Beweis. Nach Voraussetzung gilt $|U| \leq \frac{3+\alpha}{\epsilon D}$, und da U eine Quellkomponente ist, ist U nach Definition stark zusammenhängend; also sind die ersten beiden Bedingungen für kompakte Komponenten erfüllt. Außerdem gibt es keine Kante von $V \backslash U$ nach U, und damit kann auch die dritte Bedingung für kompakte Komponenten von U nicht verletzt werden. □

Für den Rest des Kapitels nehmen wir an, dass $\alpha > 0$ fest aber beliebig gewählt ist; immer, wenn für einen Graphen G die Rede von $C(G)$ ist, ist diese Wahl von α implizit mitgedacht, wie auch die Verwendung des jeweils angegebenen Näheparameters ϵ.

Das Ermitteln der kompakten Komponente eines Knotens v ist mit einer Anfragekomplexität und Laufzeit von $\mathcal{O}\left(\frac{1}{\epsilon} \cdot D^{\frac{6+2\alpha}{\epsilon D}-1}\right)$ möglich: Eine Breitensuche der Tiefe $\frac{3+\alpha}{\epsilon D}$ findet alle Knoten, die zu $C(v)$ gehören können; sei U die maximale Menge stark zusammenhängender Knoten mit $v \in U$, die dabei entdeckt wurde. Gilt $|U| > \frac{3+\alpha}{\epsilon D}$, so gilt $C(v) = \{v\}$, andernfalls ist U Kandidatenmenge für $C(v)$. Es muss noch die dritte Bedingung für kompakte Komponenten überprüft werden. Dies kann durch eine Breitensuche mit einer maximalen Tiefe von $\frac{6+2\alpha}{\epsilon D}$ geschehen, die als Startknoten die Knoten von U hat. Diese Breitensuche dominiert die Laufzeit und Anfragekomplexität der Knotenanfrage.

Es bleibt zu zeigen, dass in $C(G)$ mit den üblichen Methoden gesampelt werden kann. Wir realisieren Sampling in $C(G)$ wie folgt: Um eine Kante in $C(G)$ zu sampeln, sampeln wir zunächst eine Kante in G; sei (u, v) diese Kante. Daraufhin ermitteln wir $C(u)$ und $C(v)$ und prüfen, wie viele Kanten in G von Knoten in $C(u)$ zu Knoten in $C(v)$ verlaufen – dies ist die Anzahl k der Kanten, für die die Kante $(C(u), C(v))$ Repräsentant ist. $(C(u), C(v))$ wird nun mit Wahrscheinlichkeit $\frac{1}{k}$ als Sampelergebnis zurückgegeben, ansonsten wird das Sample verworfen. Dieses Vorgehen stellt sicher, dass die Sampelwahrscheinlichkeit für alle Kanten von $C(G)$ gleich hoch ist.

Sampling von Knoten funktioniert analog, nur dass nur die kompakte Komponente $C(v)$ des gesampelten Knotens v ermittelt werden muss; die Wahrscheinlichkeit, dass $C(v)$ nicht verworfen wird, ist dann $\frac{1}{|C(v)|}$.

Bei beiden Verfahren kann es für einen erfolgreichen Sampelvorgang in $C(G)$ nötig sein, mehrere Anfragen an G durchzuführen. Die maximale Anzahl der Kanten von G, die von einer Kante $(C(u), C(v))$ in $C(G)$ repräsentiert werden, ist $\frac{(3+\alpha)^2}{\epsilon^2 D^2}$, und die maximale Anzahl von Knoten von G, die von einem Knoten von $C(G)$ repräsentiert werden, ist $\frac{3+\alpha}{\epsilon D}$; damit ist die Wahrscheinlichkeit, dass eine gesampelte Kante beziehungsweise ein gesampelter Knoten angenommen wird, mindestens $\frac{\epsilon^2 D^2}{(3+\alpha)^2}$ beziehungsweise $\frac{\epsilon D}{3+\alpha}$.

SchätzeKnotenZahl(n, G, ϵ)
$x \leftarrow 0$
Sampele zufällig gleichverteilt $s_{4.3} = \frac{3}{\epsilon^2 d^2}$ Knoten von G
Foreach gesampelter Knoten v **do** $x \leftarrow x + \frac{1}{|C(v)|}$
return $\hat{n} = \frac{n}{t}x$

Algorithmus 4.3: SchätzeKnotenZahl

Die erwartete Anzahl tatsächlich gesampelter Objekte nach k unabhängigen Anfragen ist demnach $k \cdot \frac{\epsilon^2 D^2}{(3+\alpha)^2}$ für Kanten und $k \cdot \frac{\epsilon D}{3+\alpha}$ für Knoten, und nach der Markow-Ungleichung werden bei $\frac{k}{p} \cdot \frac{(3+\alpha)^2}{\epsilon^2 D^2}$ beziehungsweise $\frac{k}{p} \cdot \frac{(3+\alpha)^2}{\epsilon^2 D^2}$ Anfragen mit Wahrscheinlichkeit mindestens $(1-p)$ mindestens k Objekte gesampelt. Das führt zu folgender Beobachtung:

Beobachtung 4.2. *Seien $G = (V, E)$, D, ϵ und α wie in Definition 4.1 und sei $p \in (0, 1)$. Zufälliges gleichverteiltes Wählen von k Knoten von $C(G)$ mit Fehlerwahrscheinlichkeit höchstens p ist möglich mit Laufzeit und Anfragekomplexität $\mathcal{O}\left(\frac{k}{\epsilon^2 p} \cdot D^{\frac{6+2\alpha}{\epsilon D}-2}\right)$; zufälliges gleichverteiltes Wählen von l Kanten von $C(G)$ mit Fehlerwahrscheinlichkeit höchstens p ist möglich mit Laufzeit und Anfragekomplexität $\mathcal{O}\left(\frac{l}{\epsilon^3 p} \cdot D^{\frac{6+2\alpha}{\epsilon D}-3}\right)$.*

Es fehlt noch die Möglichkeit, die Anzahl der Knoten eines kontrahierten Graphen $C(G)$ zu ermitteln; diese wird zum Beispiel benötigt, um aus der Anzahl der Knoten mit eingehenden Kanten die Anzahl der Knoten ohne eingehende Kanten zu berechnen. Der Algorithmus SchätzeKnotenZahl schätzt für einen Graphen G mit n Knoten die Anzahl der Knoten in $C(G)$; im Gegensatz zum Approximationsalgorithmus für die Anzahl der Zusammenhangskomponenten in ungerichteten Graphen von Chazelle et al. ([23], siehe auch Kapitel 2.2.1) ist die Idee hier, einen Schätzwert um den Kehrwert der Größe der Komponente jedes gesampelten Knotens zu erhöhen – ein Knoten in einer großen kompakten Komponente steht nur anteilig für einen Knoten in $C(G)$. Außerdem kann das Verfahren der zufälligen Wahl der Explorationstiefe von Chazelle et al. hier nicht angewendet werden, da zum Identifizieren einer kompakten Komponente im Prinzip immer $\mathcal{O}\left(\frac{1}{\epsilon D} \cdot D^{(6+\alpha)/(\epsilon D)}\right)$ Knoten exploriert werden müssen (das gilt nur dann nicht, wenn der Ausgangsknoten in einer kleineren Senkenkomponente liegt).

Lemma 4.6. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und durch $D \in \mathbb{N}$ beschränktem Knotenein- und -ausgangsgrad; sei $\epsilon < 1$ ein Näheparameter und sei $\tilde{n}$ die Anzahl der Knoten in $C(G)$. Dann gilt für den von* SchätzeKnotenZahl(n, G, ϵ) *zurückgegebenen Wert $\hat{n}$*

$$\tilde{n} - \epsilon D n \leq \hat{n} \leq \tilde{n} + \epsilon D n$$

mit Wahrscheinlichkeit mindestens $1 - 2e^{-3}$. Die Anfragekomplexität und Laufzeit des Algorithmus ist $\mathcal{O}\left(\frac{1}{\epsilon^3 D^3} \cdot D^{(6+\alpha)/(\epsilon D)}\right)$.

Beweis. Für $i = 1, \ldots, \frac{3+\alpha}{\epsilon D}$ sei k_i die Anzahl der kompakten Komponenten in G, die exakt i Knoten haben. Sei weiter X_j für $j = 1, \ldots, s_{4.3}$ eine Zufallsvariable für den Beitrag des

TesteStarkenZusammenhang(n, G, ϵ)
  if $\epsilon D \geq 3 + \alpha$ **then return true**
  if not TesteSenkenFreiheit$(n, G, \frac{\alpha\epsilon}{6(3+\alpha)})$ **then return** false
  $\hat{n} \leftarrow$SchätzeKnotenzahl$(C(G), \frac{\alpha}{24(3+\alpha)}\epsilon)$
  $\hat{m} \leftarrow$SchätzeErreichbareKnoten$(n, C(G), \frac{3+\alpha}{\epsilon}, \frac{\alpha\epsilon^2 D}{6(3+\alpha)^2})$
  if $\tilde{m} < \hat{n} - \frac{\alpha}{12(3+\alpha)}\epsilon D n$ **then return false**
  else return true

Algorithmus 4.4: TesteStarkenZusammenhang

j-ten gesampelten Knotens v_j zu x, das heißt $X_j = \frac{1}{C(v_j)}$, und sei $X = \sum_{1 \leq j \leq s_{4.3}} X_j$. Für alle j gilt

$$E[X_j] = \sum_{1 \leq i \leq \frac{3+\alpha}{\epsilon D}} \frac{1}{i} \cdot \Pr[C(v_j) = i] = \sum_{1 \leq i \leq \frac{3+\alpha}{\epsilon D}} \frac{1}{i} \cdot \frac{k_i \cdot i}{n} = \frac{1}{n} \cdot \sum_{1 \leq i \leq \frac{3+\alpha}{\epsilon D}} k_i = \frac{\tilde{n}}{n},$$

da die Summe im vorletzten Schritt die Anzahl aller kompakten Komponenten in G und damit die Anzahl der Knoten von $C(G)$ ist. Weiter gilt

$$\mathrm{E}[\hat{n}] = \frac{n}{s_{4.3}} \cdot \mathrm{E}[X] = \frac{n}{s_{4.3}} \cdot \sum_{1 \leq j \leq s_{4.3}} \mathrm{E}[X_j] = \frac{n}{s_{4.3}} \cdot \sum_{1 \leq j \leq s_{4.3}} \frac{\tilde{n}}{n} = \tilde{n},$$

und da alle X_j zwischen 0 und 1 liegen, folgt aufgrund der Hoeffding-Ungleichung

$$\Pr[|\hat{n} - \mathrm{E}[\hat{n}]| > \epsilon d n] = \Pr[|X - \mathrm{E}[X]| > \epsilon d s_{4.3}] \leq 2 \exp\left(-\frac{\epsilon^2 d^2 s_{4.3}^2}{\sum_{1 \leq i \leq s_{4.3}} (1-0)^2}\right) = 2e^{-3}.$$

Nach den obigen Überlegungen benötigt das Ermitteln der kompakten Komponente eines Knotens von G höchstens $\mathcal{O}\left(\frac{1}{\epsilon D} \cdot D^{(6+\alpha)/(\epsilon D)}\right)$ Anfragen; da dies im Algorithmus für $s_{4.3} = \frac{3}{\epsilon^2 d^2}$ Knoten durchgeführt wird, ergibt sich insgesamt eine Anfragekomplexität von $\mathcal{O}\left(\frac{1}{\epsilon^3 D^3} \cdot D^{(6+\alpha)/(\epsilon D)}\right)$. Die Laufzeit verhält sich asymptotisch gleich. □

Wir können nun den Property-Testing-Algorithmus für starken Zusammenhang angeben. Er ruft TesteSenkenFreiheit auf, um zu testen, ob es viele Senkenkomponenten gibt; ist das nicht der Fall, dann schätzt er mittels SchätzeErreichbareKnoten die Anzahl der Knoten in $C(G)$ ohne eingehende Kante ab. Sollte dieser Schätzwert viel kleiner als die geschätzte Anzahl der Knoten von $C(G)$ sein, so wird die Eingabe abgelehnt, sonst angenommen.

Theorem 4.1. *Sei $G = (V, E)$ ein gerichteter Graph mit n Knoten und mit jeweils durch $D \in \mathbb{N}$ beschränktem Knoteneingangs- und -ausgangsgrad; sei ϵ ein Näheparameter. Dann liefert* TesteStarkenZusammenhang*(n, G, ϵ) mit Wahrscheinlichkeit jeweils mindestens $\frac{2}{3}$* false *zurück, falls G ϵ-fern von stark zusammenhängend ist, und* true*, falls G stark zusammenhängend ist. Die Anfragekomplexität des Algorithmus ist $\mathcal{O}\left(\alpha^{-4} \left(\frac{4}{\epsilon}\right)^{32/\epsilon+23} D^{144/(\alpha\epsilon^2)} n^{1-\epsilon/(3+\alpha)} \log \frac{1}{\epsilon}\right)$.*

Beweis. Sei $\tilde{n}$ die Anzahl der Knoten von $C(G)$ und sei $\tilde{m}$ die Anzahl der Knoten von $C(G)$, die mindestens eine eingehende Kante besitzen. Sei $\tilde{D}$ der maximale Knoteneinbeziehungsweise -ausgangsgrad in $C(G)$: Aufgrund der maximalen Größe von kompakten Komponenten gilt $\tilde{D} \leq \frac{3+\alpha}{\epsilon}$.

Wir nehmen zunächst an, dass G ϵ-fern von stark zusammenhängend ist, und zeigen, dass die Wahrscheinlichkeit, dass TesteStarkenZusammenhang trotzdem *true* zurückliefert, beschränkt ist. Zunächst bemerken wir, dass dies nicht aufgrund der Abfrage in der ersten Zeile geschehen kann: Für $\epsilon \geq 3 + \alpha$ gibt es keinen Graphen, der ϵ-fern von stark zusammenhängend ist, da in diesem Fall alle Kanten des Graphen entfernt werden könnten und dann ein Kreis über alle Knoten eingefügt werden könnte.

Da G ϵ-fern von stark zusammenhängend ist, enthält G nach Lemma 4.1 mehr als $\frac{1}{3}\epsilon Dn$ endständige Komponenten. Daraus folgt, dass es mindestens $\frac{\alpha}{6(3+\alpha)}\epsilon Dn$ Senkenkomponenten oder mindestens $\frac{1}{3}\epsilon Dn - \frac{\alpha}{6(3+\alpha)}\epsilon Dn = \frac{6+\alpha}{6(3+\alpha)}\epsilon Dn$ Quellkomponenten gibt. Im ersten Fall garantiert Lemma 4.2, dass TesteSenkenFeiheit mit Wahrscheinlichkeit höchstens $e^{-2} < \frac{1}{3}$ *true* zurückliefert.

Im zweiten Fall haben mindestens $\frac{\alpha}{6(3+\alpha)}\epsilon Dn$ der Quellkomponenten eine Größe von höchstens $\frac{3+\alpha}{\epsilon D}$, da die Quellkomponenten knotendisjunkt sind und insgesamt maximal n Knoten enthalten. Also ist auch die Anzahl von Knoten ohne eingehende Kanten in $C(G)$ mindestens $\frac{\alpha}{6(3+\alpha)}\epsilon Dn$, das heißt es gilt $\tilde{m} \leq \tilde{n} - \frac{\alpha}{6(3+\alpha)}\epsilon Dn$.

Lemma 4.3 garantiert, dass $\hat{m} \leq \tilde{m} + \frac{1}{8} \cdot \frac{\alpha\epsilon^2 D}{6(3+\alpha)^2} \cdot \tilde{D}n$ mit Wahrscheinlichkeit mindestens $\frac{3}{4}$ gilt; Lemma 4.6 garantiert $\hat{n} \geq \tilde{n} - \frac{\alpha}{24(3+\alpha)}\epsilon Dn$ mit Wahrscheinlichkeit mindestens $1 - 2e^{-3}$. Zusammen mit $\tilde{D} \leq D \cdot \frac{3+\alpha}{\epsilon D} = \frac{3+\alpha}{\epsilon}$ ergibt sich

$$\begin{aligned}\hat{m} &\leq \tilde{m} + \frac{1}{8} \cdot \frac{\alpha\epsilon^2 D}{6(3+\alpha)^2} \cdot \tilde{D}n \leq \tilde{n} - \frac{\alpha}{6(3+\alpha)}\epsilon Dn + \frac{1}{8} \cdot \frac{\alpha}{6(3+\alpha)}\epsilon Dn \\ &\leq \hat{n} + \frac{1}{4} \cdot \frac{\alpha}{6(3+\alpha)}\epsilon Dn - \frac{7}{8} \cdot \frac{\alpha}{6(3+\alpha)}\epsilon Dn < \hat{n} - \frac{\alpha}{12(3+\alpha)}\epsilon Dn,\end{aligned}$$

und damit liefert TesteStarkenZusammenhang(n, G, D, ϵ) in diesem Fall aufgrund der Union-Bound mit Wahrscheinlichkeit mindestens $1 - \frac{3}{4} - 2e^{-3} > \frac{2}{3}$ *false* zurück.

Zuletzt bleibt noch der Fall, dass G stark zusammenhängend ist; es gibt also keine endständigen Komponenten in G, und wegen Lemma 4.2 gibt der Aufruf TesteSenkenFreiheit$(n, G, \frac{\alpha\epsilon}{6(3+\alpha)})$ daher *true* zurück. Insbesondere gibt es außerdem keine Quellkomponenten in G, und damit gibt es keine Knoten ohne eingehende Kante in $C(G)$ – jeder solche Knoten in $C(G)$ würde das Vorhandensein einer Quellkomponente in G implizieren. Es gilt also $\tilde{m} = \tilde{n}$. Des Weiteren garantiert Lemma 4.3, dass $\hat{m} \geq \tilde{m}$ mit Wahrscheinlichkeit mindestens $\frac{3}{4}$ gilt, und nach Lemma 4.6 gilt $\hat{n} \leq \tilde{n} + \frac{\alpha}{24(3+\alpha)}\epsilon Dn$ mit Wahrscheinlichkeit mindestens $1 - 2e^{-3}$. Es ergibt sich insgesamt

$$\hat{m} \geq \tilde{m} = \tilde{n} \geq \hat{n} - \frac{\alpha}{24(3+\alpha)}\epsilon Dn > \hat{n} - \frac{\alpha}{12(3+\alpha)}\epsilon Dn,$$

und damit liefert TesteStarkenZusammenhang(n, G, D, ϵ) in diesem Fall aufgrund der Union-Bound mit Wahrscheinlichkeit mindestens $\frac{2}{3}$ *true* zurück.

Die Anfragekomplexität des Algorithmus wird bestimmt durch die Anfragekomplexität von SchätzeErreichbareKnoten$(n, C(G), \frac{3+\alpha}{\epsilon}, \frac{\alpha\epsilon^2 D}{6(3+\alpha)^2})$: Durch die Gradschranke von $\frac{3+\alpha}{\epsilon}$ von $C(G)$ werden dort für $\alpha < 1$

$$\mathcal{O}\left(\left(\frac{\alpha\epsilon^2 D}{6(3+\alpha)^2}\right)^{-3}\left(\frac{3+\alpha}{\epsilon}\right)^{8(3+\alpha)/\epsilon+15} n^{1-\epsilon/(3+\alpha)} \log\frac{3+\alpha}{\epsilon}\right)$$
$$= \mathcal{O}\left(\alpha^{-3}\left(\frac{4}{\epsilon}\right)^{32/\epsilon+21} n^{1-\epsilon/(3+\alpha)} \log\frac{1}{\epsilon}\right)$$

Anfragen an $C(G)$ gestellt. Da jede dieser Anfragen höchstens

$$\mathcal{O}\left(\frac{6(3+\alpha)^2}{\alpha\epsilon^2 D^2} \cdot D^{6(6+\alpha)(3+\alpha)^2/(\alpha\epsilon^2 D^2)}\right) = \mathcal{O}\left(\epsilon^{-2}\alpha^{-1} D^{144/(\alpha\epsilon^2)}\right)$$

Anfragen an G nach sich zieht, ist die Anfragekomplexität insgesamt

$$\mathcal{O}\left(\alpha^{-4}\left(\frac{4}{\epsilon}\right)^{32/\epsilon+23} D^{144/(\alpha\epsilon^2)} n^{1-\epsilon/(3+\alpha)} \log\frac{1}{\epsilon}\right).$$

□

Es bleibt noch zu beweisen, dass der Algorithmus SchätzeErreichbareKnoten die in Lemma 4.3 geforderte Gütegarantie hat, das heißt, dass $\tilde{m} \leq \hat{m} \leq \tilde{m} + \frac{1}{8}\epsilon D n$ für den Schätzwert $\hat{m}$ der Anzahl erreichbarer Knoten $\tilde{m}$ mit Wahrscheinlichkeit mindestens $\frac{3}{4}$ gilt.

Beweis (Lemma 4.3). Wir zeigen induktiv, dass die einzelnen Schätzer $\hat{n}_i$ für die Anzahl n_i der Knoten mit genau i eingehenden Kanten mit hoher Wahrscheinlichkeit gute Approximationen bieten; daraus folgern wir dann, dass auch $\hat{m}$ ein guter Schätzwert für $\tilde{m}$ ist. Zunächst sei bemerkt, dass nach der Markow-Ungleichung die Wahrscheinlichkeit, dass der Algorithmus in der Iteration mit Schleifenzähler i mehr als t_i Kanten sampelt und darum in der dritten Zeile einen möglicherweise falschen Wert zurückliefert, höchstens $\frac{1}{16D}$ ist; die Wahrscheinlichkeit, dass dies in einer beliebigen der D Iterationen geschieht, ist also nach der Union-Bound höchstens $\frac{1}{16}$. Falls der Algorithmus auf einem kontrahierten Graphen aufgerufen wird, können wir die in Beobachtung 4.2 geforderte Fehlerwahrscheinlichkeit für die Samplingprozedur auf $\frac{1}{16D}$ setzen; die Wahrscheinlichkeit, dass in mindestens einer Iteration die geforderte Anzahl von Samples nicht erreicht wird und deswegen der Algorithmus seine Gütegarantie nicht einhalten kann, ist wegen der Union-Bound höchstens $\frac{1}{16}$. Wir werden für den Rest des Beweises annehmen, dass keiner dieser Fälle eintritt.

Sei nun $a_i := a^{D/i}$ für $i = 1, \ldots, D$, wobei wir den konkreten Wert der Konstanten a später festlegen werden. Für alle diese Werte für i sei weiter V_i die Menge der Knoten von G, die genau i eingehende Kanten haben, und sei $n_i := |V_i|$.

Wir definieren die Zufallsvariable $Y_{i,j}$ als die Anzahl der i-Fach-Kollisionen auf Knoten in V_j in der Iteration der *for*-Schleife mit Schleifenzähler i; als i-Fach-Kollision bezeichnen

wir das Ereignis, dass für einen Knoten genau i seiner eingehenden Kanten zum Knotensample gehören. Sei außerdem $X_{v,i}$ eine Indikatorzufallsvariable für das Ereignis, dass in diesem Schleifendurchlauf an Knoten v eine i-Fach-Kollision stattfindet. Für Knoten $v \in V_j$ gilt

$$\begin{aligned}\mathrm{E}[X_{v,i}] = \Pr[X_{v,i} = 1] &= \binom{j}{i} p_i^i (1-p_i)^{j-i} \\ &= \binom{j}{i} (a_i n^{-1/i})^i (1-p_i)^{j-i} = \frac{1}{n}\binom{j}{i} a^D (1-p_i)^{j-i},\end{aligned}$$

da die Einzelereignisse, dass eingehende Kanten von v im Sample enthalten sind, für alle j eingehenden Kanten von v unabhängig sind; somit ist die Anzahl gezogener eingehender Kanten von v binomialverteilt. Es folgt

$$\mathrm{E}[Y_{i,j}] = \sum_{v \in V_j} \mathrm{E}[X_{v,i}] = \frac{n_j}{n}\binom{j}{i} a^D (1-p_i)^{j-i}.$$

Wir können nun die Wahrscheinlichkeit beschränken, dass $Y_{i,j}$ stark von seinem Erwartungswert abweicht. Die maximale Abweichung, die wir dabei erlauben wollen, ist $\delta_i := \frac{\epsilon a^D}{2^{i+4} D^{2i-1}}$. Da Chernoffschranken nur dann effektiv sind, wenn der Erwartungswert relativ groß ist, benutzen wir die Markow-Ungleichung, falls n_j klein ist und eine Chernoffschranke, falls n_j relativ groß ist.

Wir betrachten zunächst den Fall $n_j \leq \frac{\epsilon n (1-p)^{j-i}}{\binom{j}{i} 2^{i+7} D^{2i+1}}$, das heißt $\mathrm{E}[Y_{i,j}] \leq \frac{\epsilon a^D (1-p_i)^{2j-2i}}{2^{i+7} D^{2i+1}}$ und damit insbesondere $\mathrm{E}[Y_{i,j}] < \delta_i$. Das impliziert, dass $Y_{i,j}$ nur nach oben um mehr als δ_i von seinem Erwartungswert abweichen kann. Es gilt also

$$\Pr[|Y_{i,j} - \mathrm{E}[Y_{i,j}]| > \delta_i] = \Pr[Y_{i,j} - \mathrm{E}[Y_{i,j}] > \delta_i] < \Pr[Y_{i,j} > \delta_i] \leq \frac{\mathrm{E}[Y_{i,j}]}{\delta_i} \leq \frac{1}{8D^2}$$

aufgrund der Markow-Ungleichung.

Der zweite Fall ist, dass $n_j > \frac{\epsilon n (1-p)^{j-i}}{\binom{j}{i} 2^{i+7} D^{2i+1}}$ gilt, das heißt $\mathrm{E}[Y_{i,j}] > \frac{\epsilon a^D (1-p_i)^{2j-2i}}{2^{i+7} D^{2i+1}}$. Da $n_j \leq n$, gilt natürlich außerdem $\mathrm{E}[Y_{i,j}] \leq \binom{j}{i} a^D (1-p_i)^{j-i} \leq \frac{2^{i+4} D^{2i-1} \binom{j}{i} (1-p)^{j-i}}{\epsilon} \cdot \delta_i$. Da die Einzelereignisse $X_{v,i} = 1$ für alle Knoten v unabhängig sind, folgt mit Hilfe einer multipliativen Chernoffschranke

$$\begin{aligned}\Pr[|Y_{i,j} - \mathrm{E}[Y_{i,j}]| > \delta_i] &\leq \Pr\left[|Y_{i,j} - \mathrm{E}[Y_{i,j}]| > \frac{\epsilon}{2^{i+4} D^{2i-1} \binom{j}{i} (1-p)^{j-i}} \mathrm{E}[Y_{i,j}]\right] \\ &\leq 2\exp\left(-\frac{\epsilon^2}{3 \cdot 2^{2i+8} D^{4i-2} \binom{j}{i}^2 (1-p)^{2j-2i}} \mathrm{E}[Y_{i,j}]\right) \\ &< 2\exp\left(-\frac{\epsilon^3 a^D}{3 \cdot 2^{3i+15} D^{6i-1} \binom{j}{i}^2}\right),\end{aligned}$$

und für hinreichend großes $a = \mathcal{O}\left(\frac{D^{8+13/D}}{\epsilon^{3/D}} \log^{1/D} D\right)$ ist der letzte Term nicht größer als $\frac{1}{8D^2}$.

Grundsätzlich interessieren uns lediglich die i-Fach-Kollisionen auf Knoten aus V_i, denn die Größe dieser Menge wollen wir in der Iteration mit Schleifenzähler i abschätzen. Allerdings kann es auch für Knoten mit $j > i$ eingehenden Kanten i-Fach-Kollisionen geben, was den Schätzwert verfälschen kann. Wir müssen also die Anzahl der i-Fach-Kollisionen auf Knoten aus V_j, $j > i$, abschätzen, und diesen Schätzwert von der Anzahl der insgesamt in dieser Iteration gemessenen Kollisionen abziehen. Dadurch erhalten wir einen Schätzer für $Y_{i,i}$ und damit für n_i. Zum Schätzen der $Y_{i,j}$ benutzen wir dabei die Schätzwerte $\hat{n}_j$ aus den vorangegangenen Iterationen.

Wir nehmen im Folgenden an, dass $|Y_{i,j} - \mathrm{E}[Y_{i,j}]| \leq \delta_i$ für alle $i = 1, \dots D$ und $j \geq i$ gilt. Die Wahrscheinlichkeit hierfür ist nach der Union-Bound mindestens $\frac{7}{8}$, da die Wahrscheinlichkeit, dass ein $Y_{i,j}$ um mehr als δ_i von seinem Erwartungswert entfernt ist, wie oben gezeigt höchstens $\frac{1}{8D^2}$ ist. Wir zeigen nun durch vollständige Induktion über i, dass $\hat{n}_i$ unter dieser Annahme um höchstens $\frac{\epsilon n}{2^{i+3}D^{2i-2}}$ von n_i abweicht.

Als Induktionsanfang wählen wir die Iteration mit $i = D$ als Schleifenzähler. Es gilt

$$\mathrm{E}[\hat{n}_D] = \frac{\mathrm{E}[\hat{c}_D]}{p_D^D} = \frac{n}{a_D^D}\mathrm{E}[Y_{D,D}] = \frac{n}{a^D} \cdot \frac{n_D}{n}\binom{D}{D} a^D (1-p_D)^{D-D} = n_D.$$

Wir haben außerdem angenommen, dass $Y_{D,D}$ um maximal $\delta_D = \frac{\epsilon a^D}{2^{D+4}D^{2D-1}}$ von seinem Erwartungswert abweicht. Also weicht $\hat{n}_D$ nur um höchstens $\frac{\delta_D n}{a^D} = \frac{\epsilon n}{2^{D+4}D^{2D-1}} < \frac{\epsilon n}{2^{D+3}D^{2D-2}}$ von $\mathrm{E}[\hat{n}_D] = n_D$ ab.

Sei nun $i < D$ ein beliebiger Wert des Schleifenzählers. Als Induktionsvoraussetzung nehmen wir an, dass $\hat{n}_j$ für alle $j > i$ um höchstens $\frac{\epsilon n}{2^{j+3}D^{2j-2}}$ von n_j abweicht. Da $\mathrm{E}[Y_{i,j}] = \frac{n_j}{n}\binom{j}{i} a^D (1-p_i)^{j-i}$ für alle $j > i$, gilt

$$\begin{aligned}
\mathrm{E}[\hat{n}_i] =& \mathrm{E}\left[\frac{\hat{c}_i}{p_i^i} - \sum_{i<j\leq d}\binom{j}{i}\hat{n}_j(1-p_i)^{j-i}\right] = \frac{\mathrm{E}[\hat{c}_i]}{p_i^i} - \sum_{i<j\leq d}\binom{j}{i}\mathrm{E}[\hat{n}_j](1-p_i)^{j-i} \\
=& \frac{\mathrm{E}[\hat{c}_i]}{p_i^i} - \sum_{i<j\leq d}\binom{j}{i} n_j (1-p_i)^{j-i} = \frac{\mathrm{E}[\hat{c}_i]}{p_i^i} - \sum_{i<j\leq d}\frac{n}{a^D}\mathrm{E}[Y_{i,j}] \\
=& \frac{1}{p_i^i}\left(\mathrm{E}[\hat{c}_i] - \sum_{i<j\leq d} E[Y_{i,j}]\right) = \frac{E[Y_{i,i}]}{p_i^i} = n_i.
\end{aligned}$$

Es bleibt die maximale Abweichung von $\hat{n}_i$ zu seinem Erwartungswert zu beschränken. Es gibt dabei zwei Arten von Fehlerquellen: Die erste ist, dass $\hat{n}_j$ für $j > i$ von $\mathrm{E}[\hat{n}_j]$ abweichen kann; diese Abweichung ist aufgrund der Induktionsvoraussetzung maximal $\frac{\epsilon n}{2^{j+3}D^{2j-2}}$, und da $\hat{n}_j$ in der Berechnung von $\hat{n}_i$ mit $\binom{j}{i}(1-p_i)^{j-i}$ multipliziert wird, ist der Beitrag von $\hat{n}_j$ zur Gesamtabweichung höchstens

$$\alpha_j := \binom{j}{i} \cdot \frac{\epsilon n}{2^{j+3}D^{2j-2}} \leq D^{j-i} \cdot \frac{\epsilon n}{2^{j+3}D^{2j-2}} = \frac{\epsilon n}{2^{j+3}D^{j+i-2}} \leq \frac{\epsilon n}{2^{i+4}D^{2i-1}}.$$

Die zweite Fehlerquelle ergibt sich aus den Abweichungen von maximal δ_i zwischen $Y_{i,j}$ und $\mathrm{E}[Y_{i,j}]$, durch die $\hat{c}_i$ von $\mathrm{E}[\hat{c}_i]$ abweicht. Der Beitrag zur Gesamtabweichung für $j \geq i$ maximal

$$\beta_j := \frac{\delta_i}{p_i^i} = \frac{n}{a^D} \cdot \frac{\epsilon a^D}{2^{i+4} D^{2i-1}} = \frac{\epsilon n}{2^{i+4} D^{2i-1}}.$$

Die maximale Gesamtabweichung ist also

$$|\hat{n}_i - n_i| = |\hat{n}_i - \mathrm{E}[\hat{n}_i]| \leq \sum_{i<j\leq D} \alpha_j + \sum_{i\leq j\leq D} \beta_j < \sum_{i\leq j\leq D} \alpha_j + \beta_j \leq \frac{2D\epsilon n}{2^{i+4} D^{2i-1}} = \frac{\epsilon n}{2^{i+3} D^{2i-2}},$$

was den Induktionsschritt abschließt.

Insgesamt gilt also

$$\sum_{1\leq i\leq D} |\hat{n}_i - n_i| \leq \sum_{1\leq i\leq D} \frac{\epsilon n}{2^{i+3} D^{2i-2}} \leq \sum_{1\leq i\leq D} \frac{\epsilon n}{2^4} = \frac{\epsilon}{16} Dn,$$

und damit weicht $\hat{m}$ nur um maximal $\frac{\epsilon}{16}Dn$ von $\tilde{m} + \frac{\epsilon}{16}Dn$ ab, das heißt

$$\tilde{m} \leq \hat{m} \leq \tilde{m} + \frac{\epsilon}{8} Dn.$$

Die Erfolgswahrscheinlichkeit des Algorithmus ergibt sich aus den Wahrscheinlichkeiten für erfolgreiches Sampling und der Wahrscheinlichkeit, dass alle $Y_{i,j}$ die geforderte Nähe zu ihrem Erwartungswert haben; nach der Union-Bound ergibt sich damit eine Erfolgswahrscheinlichkeit von mindestens $1 - \left(\frac{1}{16} + \frac{1}{16} + \frac{1}{8}\right) = \frac{3}{4}$.

Die Anfragekomplexität des Algorithmus ist bestimmt durch die Anzahl gesampelter Kanten; diese ist maximal $\mathcal{O}(Da_i n^{1-1/i}) = \mathcal{O}(\epsilon^{-3} D^{8D+14} n^{1-1/i} \log D)$ in der Iteration mit Schleifenzähler i und damit insgesamt höchstens $\mathcal{O}(\epsilon^{-3} D^{8D+15} n^{1-1/D} \log D)$. □

Am Ende dieses Kapitels geben wir noch einen vergleichenden Überblick über die Beweisskizze zu einem Property-Testing-Algorithmus für starken Zusammenhang, die Oded Goldreich unabhängig und zeitlich vor dem hier vorgestellten Ergebnis veröffentlicht hat (siehe [39], Anhang des Survey-Artikels *Introduction to Testing Graph Properties*). Goldreich bemerkt dort zunächst, dass die Graphklassen, die Bender und Ron für die untere Schranke von $\Omega(\sqrt{n})$ Anfragen für das Testen von starkem Zusammenhang angeben, auch eine untere Schranke von $\Omega(n)$ für Algorithmen mit einseitigem Fehler implizieren. Der am Anfang des Kapitels angegebene Beweis für eine untere Schranke macht sich dies ebenfalls zunutze, wobei nur die von Bender und Ron angegebene Klasse ϵ-entfernter Graphen Anwendung findet; die für kleine Samplegrößen konstruierten stark zusammenhängenden Graphen gehören nicht notwendigerweise der von Bender und Ron angegebenen Klasse stark zusammenhängender Graphen an.

Für das Testen von starkem Zusammehang ist Goldreichs Herangehensweise bis auf die Reduktionsfunktion die gleiche wie die, die in diesem Kapitel gewählt wurde: Insbesondere schlägt er ebenfalls vor, das Problem zunächst für Quellkomponenten der Größe eins zu lösen, indem durch i-Fach-Kollisionsstatistiken der Zielknoten gesampelter Kanten die Anzahl der Knoten mit Eingangsgrad i geschätzt wird, für $i = 1, \ldots, D$. Die

Reduktionsfunktion selbst hat eine ähnliche Auswirkung wie die hier verwendete: Goldreich betrachtet gerichtete Kreise der Länge höchstens $s = \lceil \frac{4}{\epsilon D} \rceil$ und für jeden Knoten v die Menge C_v derjenigen Knoten, die zusammen mit v auf einem beliebigen solchen Kreis liegen; gilt $C_u = C_v$ für alle Knoten $u \in C_v$, dann wird die Knotenmenge kontrahiert. Damit ist garantiert, dass alle hinreichend kleinen Quellkomponenten zu einem einzelnen Knoten kontrahiert werden und die oben angegebene Vorgehensweise für den Spezialfall angewendet werden kann; zudem ist C_v für alle Knoten v lokal berechenbar. Goldreichs Reduktion führt im Gegensatz zu der hier angegebenen allerdings dazu, dass die maximale Knotenzahl kontrahierter Komponenten s^2 ist und damit der maximale Eingangsgrad des entstehenden Graphen $t = s^2 \cdot D \approx \frac{16}{\epsilon^2 D}$, was zu einer etwas schlechteren Anfragekomplexität führt[2].

Insgesamt skizziert Goldreich die Vorgehensweise lediglich und lässt alle technischen Details offen, insbesondere auch die konkrete Vorgehensweise beim Schätzen der Eingangsgradverteilung oder den Umstand, dass für die Ausführung des Algorithmus auf dem reduzierten Graphen mit einer Schätzung der Knotenzahl gearbeitet werden muss.

[2] Goldreich gibt $t = \lceil \frac{4}{\epsilon D} \rceil \cdot D$ an, obwohl er den maximalen Knotengrad des reduzierten Graphen mit $s \cdot s$ beziffert.

5 Testen von gerichteten euklidischen Spannern

Das Problem, anhand dessen wir uns in diesem Kapitel dem Testen von Eigenschaften geometrischer gerichteter Graphen nähern werden, ist, gute geometrische Spanner von anderen geometrischen Graphen zu unterscheiden. Ein Spanner ist ein geometrischer Graph, in dem es zwischen jedem Paar von Knoten einen „kurzen“ Pfad gibt, das heißt, einen Pfad, der die euklidische Distanz zwischen ihnen um maximal einen bestimmten, gegebenen Faktor überschreitet:

Definition 5.1. *Sei $G = (P, E)$ ein geometrischer Graph. G ist ein $(1+\delta)$-Spanner, falls $d_G(p,q) \leq (1+\delta)||p-q||_2$ für alle $p, q \in P$ gilt.*

Wir werden den Begriff $(1+\delta)$-Spanner-Eigenschaft sowohl für Graphen als auch für Knotenpaare verwenden; im Graphkontext haben diese Eigenschaft Graphen, die $(1+\delta)$-Spanner sind, im Knotenpaarkontext Paare von Knoten, deren Entfernung im Graphen höchstens $(1+\delta)$ mal so groß ist wie ihre euklidische Distanz.

In Bezug auf das Testen in gerichteten Graphen fällt zunächst eine grundlegende Verwandtschaft der $(1+\delta)$-Spanner-Eigenschaft mit der des starken Zusammenhangs auf, die wir im vorhergehenden Kapitel untersucht haben: Dort muss zwischen allen Paaren von Knoten ein Pfad beliebiger Länge vorhanden sein, um die Eigenschaft zu erfüllen, hier ein Pfad beschränkter geometrischer Länge. Wie sich zeigen wird, fällt dieser Unterschied algorithmisch gesehen schwer ins Gewicht: Während für starken Zusammenhang kein Property-Testing-Algorithmus mit einseitigem Fehler und einer Laufzeit von $o(n)$ existiert, kann die $(1+\delta)$-Spanner-Eigenschaft mit einseitigem Fehler in sublinearer Laufzeit getestet werden.

Eine für das Property-Testing unangenehme Eigenschaft ist, dass Graphen lokal $(1+\delta)$-Spanner sein können, während es trotzdem weit voneinander entfernte Knotenpaare gibt, deren kürzeste Verbindung die $(1+\delta)$-Spanner-Eigenschaft verletzt. Ein Beispiel hierfür ist ein Graph, in dem die Knoten auf einer Kurve angeordnet sind (siehe Abbildung 5.1). Diese Eigenschaft ist deshalb unangenehm, weil die Spanner-Eigenschaft zwischen zwei weit entfernten Knoten normalerweise nicht in sublinearer Zeit überprüft werden kann – hierzu müsste der kürzeste Pfad zwischen den Knoten exploriert werden, und dieser kann, wie in Abbildung 5.1, im schlimmsten Fall alle Knoten des Graphen umfassen. Wie wir sehen werden, ist die Anzahl solcher weit entfernten Knotenpaare, die die $(1+\delta)$-Spanner-Eigenschaft verletzen, klein, falls alle nah beieinanderliegenden Knoten diese Eigenschaft haben.

Wir setzen in diesem Kapitel lediglich eine konstante Schranke $D \in \mathbb{N}$ des Ausgangsgrads des Eingabegraphen voraus; der Eingangsgrad ist beliebig. Außerdem werden wir

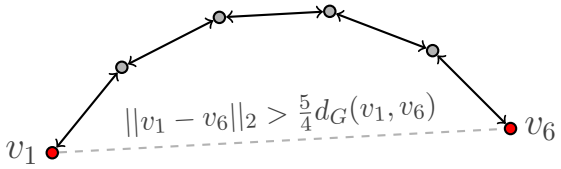

Abbildung 5.1: *Ein Graph G, der lokal ein $\frac{5}{4}$-Spanner ist; nur die weit voneinander entfernten Knoten v_1 und v_6 verletzen die $\frac{5}{4}$-Spanner-Eigenschaft.*

für das Testen der $(1+\delta)$-Spanner-Eigenschaft die Definition von ϵ-Entferntheit leicht abwandeln: Ein Graph ist ϵ-entfernt von der $(1+\delta)$-Spanner-Eigenschaft, wenn er ϵ-entfernt von allen Graphen mit der Eigenschaft ist, auch von denen, die die Gradschranke des Ausgangsgrads nicht einhalten; im Gegenzug parametrisieren wir die für ϵ-Entferntheit nötige Anzahl von Kantenänderungen nicht mit D, das heißt, zwei Graphen sind ϵ-entfernt voneinander, wenn mindestens ϵn Kantenoperationen nötig sind, um Isomorphie herzustellen – Kantenoperationen, die die Gradschranke verletzen dürfen.

Der Grund für diese Änderung des Modells ist, dass es Knotenmengen und Werte für D und δ gibt, für die kein Graph sowohl die $(1+\delta)$-Spanner-Eigenschaft erfüllt als auch einen auf D beschränkten Ausgangsgrad hat. Ein einfaches Beispiel hierfür sind vier Knoten, die die Ecken eines Quadrats bilden: Für $D = 2$ und $\delta < \sqrt{2}$ gibt es keine Möglichkeit, auf diesen Knoten einen $(1+\delta)$-Spanner zu konstruieren. Damit wäre ϵ-Entferntheit nicht wohldefiniert, falls die Klasse Π_δ der $(1+\delta)$-Spanner nur Graphen mit durch D beschränktem Ausgangsgrad enthielte.

Definition 5.2. *Sei $0 < \epsilon, \delta < 1$ und sei Π_δ die Klasse aller geometrischen Graphen, die die $(1+\delta)$-Spanner-Eigenschaft erfüllen. Sei G ein geometrischer Graph mit n Knoten und einem durch $D \in \mathbb{N}$ beschränkten Ausgangsgrad. Wir nennen G ϵ-fern von Π_δ (bzw. ϵ-fern von der $(1+\delta)$-Spanner-Eigenschaft), falls mehr als ϵn Kanten in G eingefügt werden müssen, um einen Graphen aus Π_δ zu erhalten.*

Wir werden noch eine weitere Vereinfachung vornehmen: Wir erwarten, dass alle Knoten auf einem Gitter mit einer Seitenlänge von $\Delta \in \mathbb{N}$ liegen, wobei die Zeitkomplexität unseres Algorithmus polylogarithmisch in Δ sein wird. Natürlich lässt sich für jede beliebige Knotenmenge ein geeignetes Gitter finden, so dass alle Knoten auf Gitterpunkten liegen; für hierfür ungünstige Knotenmengen steigt allerdings die Zeitkomplexität des Algorithmus. Der Einfachheit halber nehmen wir an, dass alle Knoten in $\{1, \ldots, \Delta\}^d$ liegen und Δ eine Zweierpotenz ist, was die Allgemeinheit nicht beschränkt.

5.1 Ein Property-Testing-Algorithmus für euklidische Spanner

Der Algorithmus TesteSpanner (siehe Algorithmus 5.1) versucht, einen Zeugen für die Verletzung der Spanner-Eigenschaft zu finden, also ein Paar (p, q) von Knoten, für die

TesteSpanner(n, G, δ, ϵ)
  Ziehe zufällig gleichverteilt $s_{5.1}$ Knoten $q_1, \ldots, q_{s_{5.1}}$ ohne Zurücklegen aus P
  For $i \leftarrow 1$ **to** $s_{5.1}$
    Führe Dijkstras Algorithmus von q_i startend aus, bis $s_{5.2}$ Knoten erreicht worden sind
    Sei R die Menge von erreichten Knoten und sei $W = \max_{q \in R} ||q_i - q||_2$
    Forall gezogene Knoten q_j mit $||q_i - q_j||_2 < \frac{1}{1+\delta} \cdot W$
      Führe Dijkstras Algorithmus von q_j startend auf dem durch R induzierten Subgraphen von G aus, bis q_i erreicht oder der Subgraph exploriert ist
      If $q_j \notin R$ **or** $d_G(q_i, q_j) \geq (1+\delta)\|q_i - q_j\|_2$ **or** $d_G(q_j, q_i) \geq (1+\delta)\|q_i - q_j\|_2$
        return false
  return true

Algorithmus 5.1: **TesteSpanner**.

es keinen gerichteten Pfad der Länge höchstens $(1+\delta) \cdot ||p-q||_2$ von p nach q gibt. Ein solches Knotenpaar nennen wir *verletzend*.

Definition 5.3. *Sei $G = (P, E)$ ein gerichteter geometrischer Graph und sei $\delta < 1$. Wir nennen $(u, v) \in P^2$ ein verletzendes Knotenpaar, falls $d_G(u, v) > (1+\delta)||u-v||_2$ oder $d_G(v, u) > (1+\delta)||u-v||_2$ gilt.*

Wie wir in der Analyse des von TesteSpanner sehen werden, lehnt der Algorithmus nur dann ab, wenn ein verletzendes Knotenpaar identifiziert wird; daher handelt es sich um einen Algorithmus mit einseitigem Fehler.

TesteSpanner zieht für einen gegebenen Graphen G ein zufällig gleichverteiltes Knotensample der Größe $s_{5.1} = \mathcal{O}(\sqrt{n})$ und exploriert für jeden der Knoten mit Hilfe von Dijkstras Algorithmus [33] die $s_{5.2}$ in Bezug auf die Graphdistanz nächsten Knoten in G. Sei v ein Knoten des Samples und sei R die Menge der in der zu v gehörigen Iteration der *For*-Schleife durch Dijkstras Algorithmus explorierten Knoten. Sei $u \in R$ der bezüglich der euklidischen Distanz am weitesten von v entfernte Knoten, der durch die Dijkstra-Suche gefunden wurde: Hat ein anderer gesampelter Knoten w eine Distanz von höchstens $\frac{1}{1+\delta}||u-v||_2$ zu v und wurde nicht durch die Dijkstra-Suche entdeckt, oder wurde w entdeckt, aber der gefundene Pfad von v nach w ist länger als $(1+\delta) \cdot ||v-w||_2$, dann lehnt TesteSpanner ab; außerdem wird für w durch eine weitere Dijkstra-Suche auf dem durch R induzierten Subgraphen von G getestet, ob auch ein kurzer Pfad von w nach v existiert, und falls dies nicht der Fall ist, ebenfalls abgelehnt. Falls für keinen der anfangs gesampelten Knoten abgelehnt wird, akzeptiert der Algorithmus die Eingabe.

Wir werden zeigen, dass es viele im obigen Sinn geometrisch nah beieinander liegende verletzende Knotenpaare v, w gibt, falls G ϵ-entfernt von jedem $(1+\delta)$-Spanner ist. Zudem folgt aus einem dem Geburtstagsparadox verwandten Argument, dass das von TesteSpanner gezogene Knotensample mit hoher Wahrscheinlichkeit eins dieser Paare enthält. Somit lehnt der Algorithmus in diesem Fall mit hoher Wahrscheinlichkeit ab,

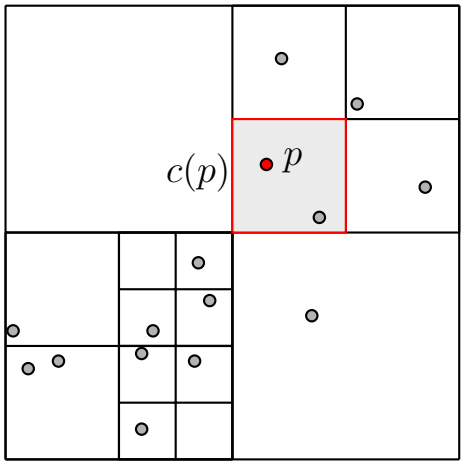

Abbildung 5.2: *Ein 2^d-Baum über eine Punktmenge im Zweidimensionalen (*Quadtree*); aufgeteilt wurde jeder Würfel mit mindestens drei Punkten. Ein Punkt p und die zugehörige Box $c(p)$ sind markiert.*

während er $(1+\delta)$-Spanner mit Wahrscheinlichkeit 1 annimmt, wie oben schon bemerkt.

Bezüglich der Laufzeit von TesteSpanner sei an dieser Stelle bemerkt, dass die **Forall**-Schleife in Zeile 5 mit Hilfe von Bereichsanfragen derart implementiert werden kann, dass nur nahe beieinander liegende Knotenpaare berücksichtigt werden und die Laufzeit des Algorithmus auf $\tilde{\mathcal{O}}(\sqrt{n})$ sinkt; eine detaillierte Analyse der Laufzeit und Anfragekomplexität von TesteSpanner folgt im nächsten Abschnitt.

5.2 Analyse des Algorithmus TesteSpanner

Sei $G = (P, E)$ ein beliebiger gerichteter geometrischer Eingabegraph mit durch $D \in \mathbb{N}$ beschränktem Ausgangsgrad und mit Knoten $P \subseteq \{1 \dots \Delta\}^d$; wir nehmen an, dass $d \in \mathbb{N}$ eine Konstante ist. Seien $\delta, \epsilon \in (0, 1)$.

Für die Analyse von TesteSpanner wollen wir den von G bedeckten Raum in eine Anzahl Würfel aufteilen, so dass jeder der Würfel höchstens $t = \mathcal{O}(\text{poly}(\log \Delta, \frac{1}{\epsilon}, \frac{1}{\delta}))$ Knoten enthält. Da die Knotendichte in verschiedenen Bereichen von G unterschiedlich hoch sein kann, verwenden wir hierfür einen 2^d-Baum[1].

Definition 5.4. *Seien $G = (P, E)$, ϵ und δ wie oben. Der 2^d-Baum H für G ergibt sich, indem, startend mit dem Würfel, der $\{1, \dots, \Delta\}^d$ bedeckt, rekursiv jeder Würfel in 2^d gleich große Unterwürfel aufgeteilt wird, falls er mehr als $t = \mathcal{O}(poly(\log \Delta, \frac{1}{\epsilon}, \frac{1}{\delta}))$ Knoten enthält.*

Den genauen Wert von t werden wir später festlegen. Die „Blatt“-Würfel des 2^d-Baums werden wir in der Folge als *Boxen* bezeichnen:

[1] Siehe [17] für eine Einführung für $d = 2$; die Datenstruktur heißt in diesem Fall *Quadtree*.

Definition 5.5. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G. Diejenigen der Unterwürfel von H, die nicht mehr aufgeteilt werden, nennen wir* Boxen*, und für einen Knoten $p \in P$ bezeichnen wir die eindeutige Box, die ihn enthält, mit $c(p)$. Für eine Box c bezeichnet $w(c)$ die Seitenlänge von c, gemessen in Gitterpunkten.*

Wir benötigen die Zuweisung der Knoten zu Boxen, um die Knotenmenge zu strukturieren: Wir werden, grob gesagt, unterscheiden zwischen Knoten, die in nahe beieinanderliegenden Boxen liegen, und Knoten, die in weit entfernten Boxen liegen; wie wir sehen werden, gibt es in von der $(1+\delta)$-Spanner-Eigenschaft ϵ-entfernten Graphen G viele – mindestens $\frac{3}{4}\epsilon n$ – verletzende Paare von Knoten in nahe beieinanderliegenden Boxen; dies entspricht der oben angesprochenen geometrischen Nähe verletzender Knotenpaare, die notwendig ist, damit die Dijkstra-Suche in TesteSpanner ein verletzendes Knotenpaar identifizieren kann. Formell definieren wir diese Nähebeziehung wie folgt:

Definition 5.6. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G. Für eine Box c ist die* Nachbarschaft $\Gamma(c)$ *definiert als die Menge aller Knoten, die in dem Würfel der Seitenlänge $3^{i^*} \cdot w(c)$, zentriert um den Mittelpunkt von c, liegen. Für Knoten $p, q \in P$ nennen wir q* geometrischen Nachbarn *von p, falls $q \in \Gamma(c(p))$. Wir nennen ein Paar p, q von Knoten geometrisch benachbart, wenn $p \in \Gamma(c(q))$ oder $q \in \Gamma(c(p))$.*

Dabei ist 3^{i^*} die kleinste Dreierpotenz, die größer oder gleich $\frac{17\sqrt{d}}{\delta}$ ist:

Definition 5.7. *Es gelte $i^* := \min\{i \in \mathbb{N} : 3^i \geq 17\sqrt{d}/\delta\}$.*

Falls ein Knoten p Nachbar eines Knotens q ist, muss q nicht zwangsläufig Nachbar von p sein: Die Seitenlänge des die Nachbarschaft um eine Box definierenden Würfels hängt von der Seitenlänge der Box ab, und die Seitenlängen von $c(p)$ und $c(q)$ können unterschiedlich sein.

Ziel der Definition der Nachbarschaft einer Box ist es, eine Beziehung zwischen der Anzahl der Knoten in einem Bereich und dessen räumlicher Ausdehnung herzustellen; dies benötigen wir, um zu zeigen, dass die Dijkstra-Suche in TesteSpanner für (fast) alle Boxen c einen Bereich bestimmter geometrischer Größe komplett exploriert.

Komplette Exploration bedeutet in diesem Fall nicht, dass alle Knoten in einem gegebenen Bereich – der Nachbarschaft $\Gamma(c)$ um eine Box c bzw. um einen Knoten $p \in c$ – erreicht werden, sondern dass alle Pfade zwischen Knoten $p \in c$ und $q \in \Gamma(c)$ gefunden werden, die eine Länge von höchstens $(1+\delta)||p-q||_2$ haben. Solche Pfade können auch Knoten außerhalb von $\Gamma(c)$ enthalten; formell definieren wir dafür folgende *erweiterte Nachbarschaft*:

Definition 5.8. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G. Für eine Box c ist die* erweiterte Nachbarschaft $\hat{\Gamma}(c)$ *definiert als die Menge aller Knoten, die in dem Würfel der Seitenlänge $2\sqrt{d} \cdot 3^{i^*+1} \cdot w(c)$, zentriert um den Mittelpunkt von c, liegen.*

Seien p und $q \in \Gamma(c(p))$; die Idee von Definition 5.8 ist, dass alle Knoten, die auf kurzen Pfaden von p nach q liegen, in $\hat{\Gamma}(c(p))$ enthalten sind; wir werden dies später noch formell zeigen.

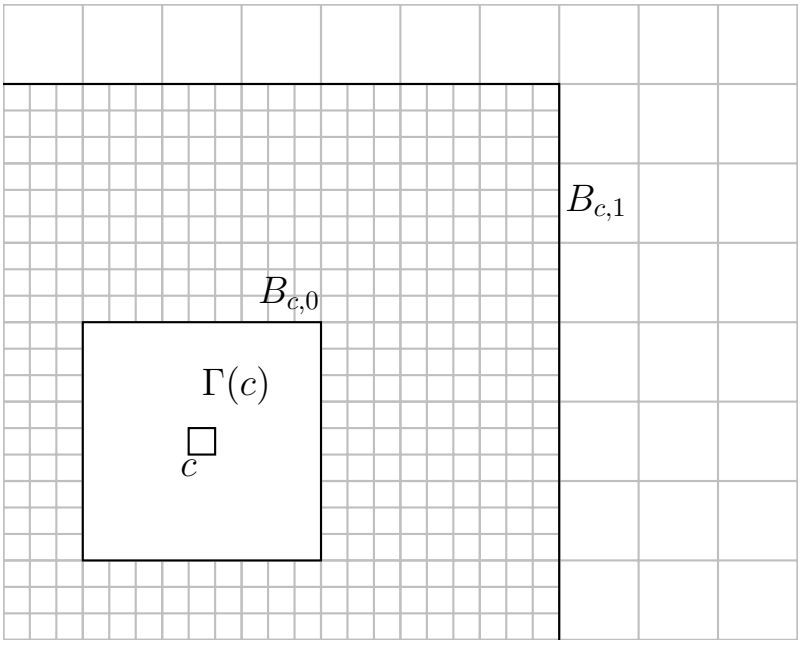

Abbildung 5.3: *Ausschnitt des exponentiellen Gitters um eine Box c; aus Platzgründen werden die Würfel $B_{c,i}$ um einen Faktor von 3^{i^*-2} verkleinert dargestellt.*

Nehmen wir an, dass $p \in c$ und $q \in \Gamma(c)$ ein verletzendes Knotenpaar bilden und die Dijkstra-Suche in TesteSpanner, startend bei p, jeden Knoten in $\hat{\Gamma}(c)$ erreicht; dann wird q entweder nicht entdeckt oder der kürzeste gefundene Pfad von p nach q ist zu lang. Das heißt, dass TesteSpanner ein solches verletzendes Paar (p, q) zumindest dann identifiziert, wenn sowohl p als auch q dem Knotensample angehören. Mit einer Argumentation ähnlich dem Geburtstagsparadoxon ergäbe sich dann, dass bei einer Samplegröße von ungefähr $\Theta(\sqrt{n})$ mindestens ein benachbartes verletzendes Paar identifiziert würde, vorausgesetzt, es gibt $\Theta(n)$ solcher Paare.

Um dies zu zeigen, werden wir eine obere Schranke für die Anzahl verletzender Knotenpaare herleiten, die keine Nachbarn sind oder die in Nachbarschaften liegen, die nicht von TesteSpanner exploriert werden können. Dafür gliedern wir für jede Box c die Knoten außerhalb von $\Gamma(c)$ durch ein *exponentielles Gitter* H_c.

Dieses ist ähnlich wie in [47] definiert (siehe Abbildung 5.3):

Definition 5.9 (Exponentielles Gitter um eine Box c)**.** *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G. Sei $B_{c,i}$ für alle $i \in \mathbb{N}$ ein Würfel, zentriert um den Mittelpunkt von c, der eine Seitenlänge von $3^{i^*+i} \cdot w(c)$ hat.*

Für $i \geq 1$ bedecken wir den Raum $B_{c,i} \backslash B_{c,i-1}$ mit Würfeln der Seitenlänge $w_i(c) = 3^{i-1} \cdot w(c)$. Diese Würfel nennen wir Zellen *im exponentiellen Gitter um c. Würfel, die außerhalb von $\{1, \ldots, \Delta\}^d$ liegen, werden dabei ignoriert – solche Würfel können keine Knoten aus P enthalten.*

Es sei bemerkt, dass $B_{c,0}$ der Würfel ist, der die Nachbarschaft $\Gamma(c)$ der Zelle c definiert.

Aufgrund der Definition der Würfel $B_{c,i}$ und $B_{c,i-1}$ werden zudem keine der $B_{c,i} \backslash B_{c,i-1}$ bedeckenden Zellen durch einen der beiden Würfel „zerschnitten“: Die Seitenlänge von $B_{c,i}$ ist ein ungerades Vielfaches der Seitenlänge von $B_{c,i-1}$, und $B_{c,i-1}$ ist ein ungerades Vielfaches von $w_i(c)$.

Lemma 5.1. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G, dessen Zellen jeweils maximal t Knoten beinhalten. Die Anzahl der Boxen von H ist höchstens $\frac{2^d n \log \Delta}{t}$.*

Beweis. Beim Aufteilen eines Würfels des 2^d-Baums haben die entstehenden Unterwürfel die Hälfte der Seitenlänge des Ausgangswürfels; das bedeutet, dass die maximale Tiefe des 2^d-Baums höchstens $\log \Delta$ ist. Würfel werden weiter aufgeteilt, wenn sie mehr als t Knoten haben: Für jede mögliche Würfelgröße (d.h. auf jeder Ebene des Baums) gibt es dabei maximal $\frac{n}{t+1}$ Würfel, für die das gilt, und die daher in insgesamt maximal $\frac{2^d n}{t+1}$ Unterwürfel auf der folgenden Ebene aufgeteilt werden. Das bedeutet, dass die Gesamtzahl der Boxen kleiner als $\frac{2^d n \log \Delta}{t}$ ist. □

Lemma 5.2. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G, dessen Zellen jeweils maximal t Knoten beinhalten. Für $t := \frac{2^{d+3} \log \Delta}{\epsilon} \cdot (3^{i^*+1})^d \cdot \lceil \log_3 \frac{2\Delta}{3^{i^*}} \rceil = \mathcal{O}\left(\frac{\log^2 \Delta}{\delta^d \epsilon}\right)$ ist die Gesamtzahl aller Zellen in den exponentiellen Gittern aller Boxen des 2^d-Baums höchstens $\epsilon n/8$.*

Beweis. Wir begrenzen die Anzahl der Zellen im exponentiellen Gitter um eine Box c dadurch, dass wir einen Würfel $B_{c,i}$ ermitteln, der das gesamte Gitter $\{1, \ldots, \Delta\}^d$ umfasst: Dies ist der Fall für $i = \lceil \log_3 \frac{2\Delta}{3^{i^*}} \rceil$, da $B_{c,i}$ eine Seitenlänge von $3^{i^*+i} \cdot w(c) \geq 2\Delta w(c) \geq 2\Delta$ hat. Für jeden Würfel $B_{c,j}$, $1 \leq j \leq i$, ist die Anzahl der Zellen, die $B_{c,j} - B_{c,j-1}$ bedecken, höchstens $(3^{i^*+j}/w_j(c))^d = (3^{i^*+1})^d$. Also ist die Anzahl von Zellen im exponentiellen Gitter H_c um jede beliebige Zelle c höchstens $(3^{i^*+1})^d \cdot \lceil \log_3 \frac{2\Delta}{3^{i^*}} \rceil$.

Da wegen Lemma 5.1 die Anzahl der Boxen höchstens $\frac{2^d n \log \Delta}{t}$ ist, ist die Gesamtzahl an Zellen in allen exponentiellen Gittern zusammen höchstens

$$\frac{2^d n \log \Delta}{t} \cdot (3^{i^*+1})^d \cdot \left\lceil \log_3 \frac{2\Delta}{3^{i^*}} \right\rceil < \frac{1}{8} \epsilon n$$

für den im Lemma festgesetzten Wert von t. □

Die Analyse von TesteSpanner im Fall, dass G ϵ-fern von der $(1 + \delta)$-Spanner-Eigenschaft ist, beruht grundsätzlich auf folgender Argumentation:

Für die Analyse verbinden wir jede Box c mit jeder Zelle z des exponentiellen Gitters um c: Wir fügen eine Hin- und eine Rückkante zwischen zwei Knoten in z und c ein; dies sind wegen Lemma 5.2 insgesamt maximal $\frac{1}{4}\epsilon n$ Kanten. Wie wir sehen werden, ist der dadurch entstehende Graph G' ein $(1 + \delta)$-Spanner, falls es keine Box c gibt, deren Nachbarschaft ein verletzendes Paar enthält. Da G aber ϵ-fern von jedem $(1 + \delta)$-Spanner ist, ist G' $\frac{3}{4}\epsilon$-fern von jedem $(1 + \delta)$-Spanner; also ist es nicht möglich, in G' mit weniger als $\frac{3}{4}\epsilon n$ Kantenänderungen alle verletzenden Paare in den Nachbarschaften aller

Boxen mit hinreichend kurzen Pfaden zu verbinden, denn der entstehende Graph wäre ein $(1+\delta)$-Spanner. Da jedes verletzende Paar durch Einfügen einer einzigen, direkten Kante verbunden werden kann, muss es mindestens $\frac{3}{4}\epsilon n$ verletzende Paare in Nachbarschaften von Zellen von G' und damit auch von G geben; es ist zu zeigen, dass TesteSpanner unter dieser Voraussetzung mindestens eins der Paare entdeckt. Dazu ist insbesondere auch eine obere Schranke für die Anzahl solcher verletzenden Paare nötig, zu denen ein einzelner Knoten gehören kann.

Ein Problem dieses Ansatzes besteht in der Annahme, dass die Nachbarschaft jeder Box c durch die Dijkstra-Suche in TesteSpanner hinreichend gut exploriert werden kann. Dies ist ein geometrisches Kriterium, während das Abbruchkriterium der Dijkstra-Suche das Überschreiten einer gegebenen Anzahl explorierter Knoten ist; wegen der Konstruktion des 2^d-Baums ist es aber möglich, dass die Nachbarschaft einer Box sehr viele kleinere Boxen und damit potentiell sehr viele Knoten enthält. Eine solche Nachbarschaft könnte nicht komplett exploriert werden.

Wir werden daher zeigen, dass es nur wenige solcher *schweren* Boxen gibt und dass die Gesamtzahl an Knotenpaaren in ihren Nachbarschaften höchstens $\frac{1}{8}\epsilon n$ ist. Durch Einfügen einer Hin- und einer Rückkante zwischen allen diesen Paaren werden natürlich auch alle verletzenden Knotenpaare miteinander verbunden, und wir werden zeigen, dass dabei insgesamt nur $\frac{1}{4}\epsilon n$ Kanten eingefügt werden.

Der obigen Argumentationslinie folgend bedeutet das, dass jeder Graph, der ϵ-fern von der $(1+\delta)$-Spanner-Eigenschaft ist, mindestens $\frac{1}{2}\epsilon n$ verletzende Knotenpaare in Nachbarschaften von nicht schweren Boxen enthält; für diese Boxen können wir die Anzahl der Knoten beschränken, die exploriert werden müssen, so dass TesteSpanner einen hinreichend großen geometrischen Bereich exploriert.

Um diesen Ansatz zu formalisieren, werden wir zunächst das Konzept der schweren Boxen formell definieren und eine obere Schranke für ihre Anzahl herleiten.

Definition 5.10. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G, dessen Zellen jeweils maximal t Knoten beinhalten. Eine Box c heißt* schwer*, falls $\hat{\Gamma}(c)$ mindestens*

$$k := \epsilon \cdot 2^{2d+3}(\sqrt{d})^d \cdot 3^{2di^*+d} t \log^2 \Delta = \mathcal{O}\left(\frac{\log^4 \Delta}{\delta^{3d}\epsilon^2}\right)$$

Boxen c' mit $w(c') < w(c)$ enthält. Andernfalls nennen wir c sampelbar*.*

Lemma 5.3. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G, dessen Zellen jeweils maximal t Knoten beinhalten. Es gibt höchstens $\frac{\epsilon n}{8 \cdot 3^{di^*} t^2}$ schwere Boxen in H.*

Beweis. Wir werden zunächst eine obere Schranke für die Anzahl größerer Boxen herleiten, in deren erweiterter Nachbarschaft eine Box liegen kann; sei dafür c eine beliebige Box. Sei $w > w(c)$ eine feste aber beliebige Seitenlänge von Würfeln des 2^d-Baums und sei q der eindeutige Würfel mit Seitenlänge w, der c enthält. Dann liegen laut Konstruktion alle Boxen c' mit $w(c') = w$ und $c \in \hat{\Gamma}(c')$ in einem Würfel der Seitenlänge $2\sqrt{d} \cdot 3^{i^*+1} \cdot w$ zentriert um q. Dies sind maximal $(2\sqrt{d} \cdot 3^{i^*+1})^d$ Boxen der Seitenlänge w.

Aufgrund der Konstruktion des 2^d-Baums gibt es höchstens $\log \Delta$ verschiedene Boxgrößen, d.h. die Anzahl größerer Boxen, in deren erweiterter Nachbarschaft c liegt, ist höchstens $2^d(\sqrt{d})^d \cdot 3^{di^*+d} \log \Delta$. Da wegen Lemma 5.1 die Gesamtzahl aller Boxen höchstens $\frac{2^d n \log \Delta}{t}$ ist, ist die Gesamtzahl derartiger *Enthalten-In*-Relationen maximal $\frac{2^{2d}(\sqrt{d})^d \cdot 3^{di^*+d} n \log^2 \Delta}{t}$. Weil jede schwere Box laut Definition 5.10 mindestens k solcher Relationen benötigt, begrenzt dies die Gesamtzahl schwerer Boxen auf

$$\frac{2^{2d}(\sqrt{d})^d \cdot 3^{di^*+d} n \log^2 \Delta}{tk} = \frac{\epsilon n}{8 \cdot 3^{di^*} t^2}.$$

□

Für das Hauptlemma der Analyse, Lemma 5.5, benötigen wir ein Hilfslemma, das grob gesagt eine Verbindung zwischen der Größe zweier Boxen und ihrer Nachbarschaftsbeziehung herstellt.

Definition 5.11. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G. Für zwei Knoten $p, q \in P$ sei $\varphi(p, q)$ die Nummer i des kleinsten Würfels $B_{c(p),i}$, der q enthält.*

Lemma 5.4. *Seien $G = (P, E)$, ϵ und δ wie oben und sei H der 2^d-Baum für G. Seien $p, q \in P$ mit $w(c(p)) \leq w(c(q))$. Dann gilt $\varphi(p, q) \geq \varphi(q, p)$. Falls $w(c(p)) = w(c(q))$, dann gilt auch $\varphi(p, q) = \varphi(q, p)$.*

Beweis. Wir unterscheiden zwei Fälle: $w(c(p)) = w(c(q))$ und $w(c(p)) < w(c(q))$. Im ersten Fall hat $B_{c(p),i}$ für alle i die gleiche Seitenlänge wie $B_{c(q),i}$, und nach Konstruktion des 2^d-Baums sind $c(p)$, $c(q)$, $B_{c(p),i}$ und $B_{c(q),i}$ an einem Gitter der Seitenlänge $w(c(p))$ ausgerichtet; für $B_{c(p),i}$ und $B_{c(q),i}$ gilt das, weil ihre Seitenlänge ein ungerades Vielfaches von $w(c(p))$ ist. Das bedeutet, dass $c(p)$ nicht von $B_{c(q),i}$ zerschnitten wird und umgekehrt, und daraus folgt $p \in B_{c(q),i} \Leftrightarrow q \in B_{c(p),i}$ und damit $\varphi(p, q) = \varphi(q, p)$.

Wir nehmen nun an, dass $w(c(p)) < w(c(q))$ gilt. Sei c der Würfel des 2^d-Baums mit $w(c) = w(c(q))$, der $c(p)$ enthält. Nehmen wir als Gedankenexperiment an, c würde im 2^d-Baum H nicht weiter unterteilt: Dann wäre c die Box, die p enthält, und es würde $w(c) = w(c(q))$ gelten. Analog zum ersten Fall würde daraus $\varphi(p, q) = \varphi(q, p) =: k$ folgen.

Da $c(p)$ aber ein kleinerer, in c enthaltener Würfel ist, haben die exponentiellen Würfel $B_{c(p),i}$ um $c(p)$ eine kleinere Seitenlänge als die entsprechenden exponentiellen Würfel $B_{c,i}$ um c. Da $c(p)$ in c enthalten ist, sind die exponentiellen Würfel um $c(p)$ in den entsprechenden Würfeln um c enthalten: Daher hat der kleinste Würfel $B_{c,j}$ um $c(p)$, der q enthält, einen Index von $j \geq k$, denn $B_{c,k}$ ist der kleinste Würfel um c, der q enthält. Also gilt $\varphi(p, q) = j \geq \text{k} = \varphi(q, p)$. □

Wir zeigen nun das zentrale Lemma dieser Analyse. Seine Kernaussage ist, dass in einen Graphen G $\frac{\epsilon}{2}n$ Kanten derart eingefügt werden können, dass fast alle verletzenden Knotenpaare in schweren Nachbarschaften durch direkte Kanten verbunden sind und alle verletzenden Knotenpaare, die nicht geometrisch benachbart sind, kurze Pfade in beiden

Richtungen besitzen. Für letztere Aussage ist allerdings die Voraussetzung, dass auch in den Nachbarschaften der entsprechenden Knoten kurze Pfade zwischen allen Knotenpaaren garantiert sind, so dass wir dies als Voraussetzung in das Lemma aufnehmen.

Lemma 5.5. *Sei $G = (P, E)$ ein geometrischer Graph wie oben mit einem 2^d-Baum H und einer entsprechenden Strukturierung wie oben erläutert; es gelte $0 < \epsilon, \delta < 1$ und wir nehmen an, dass die Zellen von H maximal t Knoten beinhalten. Wir nehmen an, dass für alle $p, q \in P$ mit $q \in \Gamma(c(p))$ und sampelbarer Nachbarschaft $\Gamma(c(p))$ die $(1+\delta)$-Spannereigenschaft in beiden Richtungen gilt, d.h. $d_G(p,q) \leq (1+\delta) \cdot ||p-q||_2$ und $d_G(q,p) \leq (1+\delta) \cdot ||p-q||_2$. Dann ist G $\frac{\epsilon}{2}$-nah an der $(1+\delta)$-Spannereigenschaft.*

Beweis. Wir werden zeigen, dass unter den obigen Voraussetzungen $\frac{\epsilon}{2}$ zusätzlich eingefügte Kanten genügen, um die $(1+\delta)$-Spannereigenschaft zwischen allen Paaren von Knoten in G herzustellen. Wir konstruieren aus G einen Graphen $G' = (V, E')$:

- Für alle Boxen c und Zellen z des exponentiellen Gitters um c füge eine Kante (p,q) und eine Rückkante (q,p) zwischen $p \in c$ und $q \in z$ ein; dabei ist p ein beliebiger Knoten in c und q ist ein beliebiger Knoten in der größten[2] Box c', die mindestens einen Knoten in z hat. Sei E_1 die Menge, die alle diese Kanten enthält: Wegen Lemma 5.2 gilt $|E_1| \leq \frac{1}{4}\epsilon n$.

- Für jede schwere Box c füge Kanten (p,q) und (q,p) für alle Paare von Knoten $p \in c$ und $q \in \Gamma(c) \backslash c$ hinzu; ignoriere hierbei Knoten q, die in Boxen mit $w(c(q)) < w(c)$ liegen. Sei E_2 die Menge aller dieser Kanten: Wegen Lemma 5.3 ist die Anzahl schwerer Boxen maximal $\frac{\epsilon n}{8 \cdot 3^{di^*} t^2}$, und die Nachbarschaft um eine Box c schneidet höchstens 3^{di^*} Boxen c' mit $w(c') \geq w(c)$; da jede dieser Boxen maximal t Knoten enthält – ebenso wie c selbst –, ist die Gesamtzahl der Kanten in E_2 höchstens $2 \cdot 3^{di^*} t^2 \frac{\epsilon n}{8 \cdot 3^{di^*} t^2} = \frac{1}{4}\epsilon n$.

Wir definieren nun $E' := E \cup E_1 \cup E_2$. Aufgrund der obigen Überlegungen gilt $|E' \oplus E| = |E' \backslash E| \leq \frac{1}{2}\epsilon n$. Wir zeigen, dass G' ein $(1+\delta)$-Spanner ist; diesen Beweis führen wir mittels vollständiger Induktion über $i := \varphi(p,q)$ für Knoten $p, q \in P$; dabei nehmen wir als Induktionsvoraussetzung an, dass für alle Knotenpaare $p', q' \in P$ mit $\varphi(q', p') < i$ und $\varphi(q', p') \leq \varphi(p', q')$ die $(1+\delta)$-Spannereigenschaft in beiden Richtungen gilt.

Als Induktionsanfang wählen wir beliebige Knoten p und q, so dass $i = \varphi(p,q) = \varphi(q,p) = 0$, das heißt $q \in \Gamma(p)$ und $p \in \Gamma(q)$. Falls $c(p)$ oder $c(q)$ sampelbar ist, gilt die $(1+\delta)$-Spannereigenschaft in beiden Richtungen aufgrund der Voraussetzung des Lemmas. Andernfalls sind sowohl $c(p)$ als auch $c(q)$ schwere Zellen. Sei ohne Beschränkung der Allgemeinheit $c(p)$ die Zelle mit $c(p) \leq c(q)$: Dann enthält E_2 Kanten (p,q) und (q,p), und damit gilt die $(1+\delta)$-Spannereigenschaft zwischen p und q auch in diesem Fall.

Für den Induktionsschritt sei $i \geq 1$ und p, q seien beliebige Knoten, so dass $\varphi(p,q) = i$ und $\varphi(q,p) \leq i$; dabei können wir ohne Beschränkung der Allgemeinheit annehmen, dass

[2] Falls die größte solche Box nicht eindeutig ist, kann eine beliebige der größten Boxen gewählt werden.

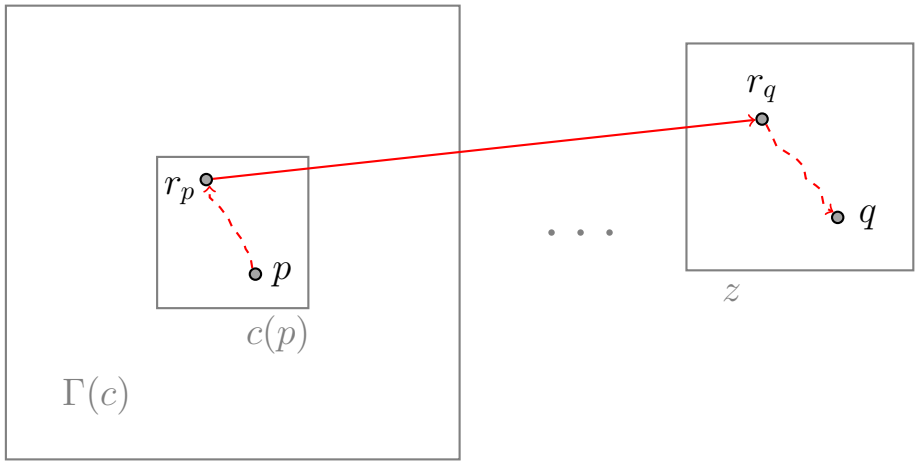

Abbildung 5.4: *Schematische Darstellung eines kurzen Pfades zwischen zwei Knoten p und q, wobei q in einer Zelle z des exponentiellen Gitters um $c(p)$ liegt.*

$w(c(p)) \leq w(c(q))$ gilt, ansonsten tauschen wir die Knotenbenennungen[3]. Da $\varphi(p,q) \geq 1$, liegt q in einer Zelle z des exponentiellen Gitters um $c(p)$; das heißt, es gibt Kanten $(r_p, r_q), (r_q, r_p) \in E_1$, so dass r_p ein Knoten in $c(p)$ ist und r_q in z liegt. Wir werden im Folgenden zeigen, dass es von p nach q und von q nach p Pfade gibt, die jeweils die entsprechende Verbindungskante zwischen r_p und r_q nutzen und eine Länge von höchstens $(1+\delta)||p-q||_2$ haben (siehe Abbildung 5.4). Dazu werden wir zunächst zeigen, dass es zwischen p und r_p in beiden Richtungen einen Pfad der Länge höchstens $(1+\delta)||p-r_p||_2$ gibt und für q und r_q in beiden Richtungen einen Pfad der Länge $(1+\delta)||q-r_q||_2$.

Ersteres folgt direkt aus der Induktionsvoraussetzung, denn es gilt $\varphi(p, r_p) = \varphi(r_p, p) = 0$. Für die Graphdistanz zwischen q und r_q wollen wir auch die Induktionsvoraussetzung anwenden; dafür müssen wir zeigen, dass $\varphi(q, r_q) < i$ und $\varphi(r_q, q) < i$ gilt. Zunächst bemerken wir, dass r_q laut Definition von E_1 in einer der größten Boxen liegt, die Knoten in z haben; insbesondere gilt $w(c(q)) \leq w(c(r_q))$, und wegen Lemma 5.4 folgt daraus $\varphi(r_q, q) \leq \varphi(q, r_q)$. Also müssen wir nur noch $\varphi(q, r_q) < i$ zeigen.

Dafür stellen wir zunächst fest, dass z eine Seitenlänge von $w_i(c(p)) = 3^{i-1}w(c(p)) \leq 3^{i-1}w(c(q))$ hat, wobei letztere Ungleichung aus der obigen Annahme $w(c(p)) \leq w(c(q))$ folgt. Da die euklidische Distanz zwischen q und r_q durch die Länge der Diagonalen von z beschränkt ist, gilt

$$\begin{aligned} ||q - r_q||_2 &\leq \sqrt{d} \cdot w_i(c(p)) \leq 3^{i-1}\sqrt{d} \cdot w(c(q)) \\ &= \frac{\sqrt{d}}{3^{i^*}} 3^{i^*+i-1} w(c(q)) \leq \frac{\delta}{17} 3^{i^*+i-1} w(c(q)); \end{aligned}$$

dabei folgt die letzte Ungleichung aus der Definition von i^*. Da $\frac{\delta}{17} \leq \frac{1}{17}$, ist die euklidische Distanz zwischen q und r_q höchstens so groß wie die ein Siebzehntel der Seitenlänge von

[3] Dies ist möglich, da wegen Lemma 5.4 und der Annahme $\varphi(q,p) \leq i$ nur dann $w(c(q)) < w(c(p))$ gelten kann, wenn auch $\varphi(q,p) = i = \varphi(p,q)$ gilt.

$B_{c(q),i-1}$ – diese ist $3^{i^*+i-1}w(c(q))$; also gilt $r_q \in B_{c(q),i-1}$ und damit $\varphi(q,r_q) \leq i-1$. Da, wie oben gezeigt, $\varphi(r_q,q) \leq \varphi(q,r_q)$, können wir mit Hilfe der Induktionsvoraussetzung folgern, dass $d_{G'}(q,r_q) \leq (1+\delta)||q-r_q||_2$ und $d_{G'}(r_q,q) \leq (1+\delta)||q-r_q||_2$ gilt.

Um den Induktionsschritt abzuschließen, werden wir nun zeigen, dass die aus den oben diskutierten Teilstücken zusammengesetzten Pfade von p nach q und von q nach p jeweils eine Länge von höchstens $(1+\delta)||p-q||_2$ haben. Wir zeigen dies für den Pfad von p nach q; der Beweis für den Rückpfad ist analog. Nach Konstruktion und aufgrund der obigen Überlegungen gilt

$$\begin{aligned} d_{G'}(p,q) &\leq d_{G'}(p,r_p) + d_{G'}(r_p,r_q) + d_{G'}(r_q,q) \\ &\leq (1+\delta)||p-r_p||_2 + ||r_p-r_q||_2 + (1+\delta)||r_q-q||_2. \end{aligned}$$

Die Idee ist, zu zeigen, dass der Umweg, der durch die Wahl von r_p und r_q als Knoten des Pfades entsteht, im Vergleich zu $||p-q||_2$ sehr klein ist; dies wird aus der Definition des exponentiellen Gitters um $c(p)$ folgen. Dafür stellen wir zunächst fest, dass die euklidischen Distanzen zwischen p und r_p und zwischen q und r_q durch die Länge der Diagonalen der jeweiligen Box bzw. Zelle begrenzt sind, d.h. es gilt

$$||p-r_p||_2 \leq \sqrt{d}\cdot w(c(p)) \leq \sqrt{d}\cdot w_i(c(p)) \text{ und } ||r_q-q||_2 \leq \sqrt{d}\cdot w_i(c(p)).$$

Durch zweimaliges Anwenden der Dreiecksungleichung erhalten wir des Weiteren

$$\begin{aligned} ||r_p-r_q||_2 &\leq ||r_p-p||_2 + ||p-r_q||_2 \\ &\leq ||r_p-p||_2 + ||p-q||_2 + ||q-r_q||_2 \\ &\leq ||p-q||_2 + 2\sqrt{d}\cdot w_i(c(p)). \end{aligned}$$

Wir können nun die Schranken für die Längen der Teilpfade addieren und erhalten

$$\begin{aligned} d_{G'}(p,q) &\leq 2(1+\delta)\sqrt{d}\cdot w_i(c(p)) + 2\sqrt{d}\cdot w_i(c(p)) + ||p-q||_2 \\ &\leq 4(1+\delta)\sqrt{d}\cdot w_i(c(p)) + ||p-q||_2. \end{aligned}$$

Da aber q außerhalb des $i-1$-ten Würfels $B_{c(p),i-1}$ um $c(p)$ liegt, ist die euklidische Distanz zwischen p und q mindestens der Abstand des Mittelpunkts von $c(p)$ zum Rand des Würfels $B_{c(p),i-1}$ minus die halbe Zellbreite von $c(p)$, also

$$\begin{aligned} ||p-q||_2 &\geq \frac{1}{2}\cdot 3^{i^*+i-1}w(c(p)) - \frac{1}{2}\cdot w(c(p)) = \frac{1}{2}\cdot 3^{i^*}w_i(c(p)) - \frac{1}{2}\cdot w(c(p)) \\ &\geq \frac{17}{2\delta}\sqrt{d}\cdot w_i(c(p)) - \frac{1}{2}\cdot w_i(c(p)) \geq \frac{8}{\delta}\sqrt{d}\cdot w_i(c(p)). \end{aligned}$$

Umstellen ergibt $\sqrt{d}\cdot w_i(c(p)) \leq \frac{\delta}{8}||p-q||_2$, was in die Abschätzung für $d_{G'}(p,q)$ eingesetzt werden kann; da $\delta \leq 2$, folgt

$$d_{G'}(p,q) \leq \frac{4\delta(1+\delta)}{8}\cdot||p-q||_2 + ||p-q||_2 \leq (1+\delta)||p-q||_2,$$

was den Induktionsschritt abschließt. □

Aus Lemma 5.5 ergibt sich der folgende Korollar:

Korollar 5.1. *Sei $G = (P, E)$ ein geometrischer Graph wie oben mit einem 2^d-Baum H und einer entsprechenden Strukturierung wie oben erläutert; es gelte $0 < \epsilon, \delta < 1$. Sei G außerdem ϵ-fern von der $(1+\delta)$-Spanner-Eigenschaft. Dann gibt es mindestens $\frac{1}{4}\epsilon n$ verletzende Knotenpaare (p, q), so dass q Nachbar von p ist und $c(p)$ sampelbar.*

Beweis. Wir nehmen an, dass G weniger als $\frac{1}{4}\epsilon n$ solcher verletzenden Knotenpaare enthält und führen diese Annahme zum Widerspruch. Als Gedankenexperiment fügen wir dazu für jedes solche Paar (p, q) Kanten (p, q) und (q, p) ein und erhalten einen Graphen G', in dem für alle sampelbaren Boxen c und Knotenpaare $p \in c$ und $q \in \Gamma(P)$ Pfade der Länge jeweils höchstens $(1+\delta)||p-q||_2$ von p nach q und von q nach p existieren. Wegen Lemma 5.5 ist G' $\frac{\epsilon}{2}$-nah an der $(1+\delta)$-Spanner-Eigenschaft; da G' sich von G aber nur durch weniger als $\frac{1}{2}\epsilon n$ eingefügte Kanten unterscheidet, folgt daraus, dass G ϵ-nah an der $(1+\delta)$-Spanner-Eigenschaft ist, ein Widerspruch zur Annahme, dass G ϵ-fern davon ist. □

Wir werden nun zeigen, dass die von Korollar 5.1 garantierte Anzahl verletzender Knotenpaare in sampelbaren Nachbarschaften ausreichend groß ist, so dass TesteSpanner mit hoher Wahrscheinlichkeit für mindestens eins dieser Paare beide beteiligten Knoten sampelt.

Lemma 5.6. *Sei $G = (P, E)$ ein gerichteter geometrischer Graph mit einem durch $D \in \mathbb{N}$ beschränkten Knotengrad; sei H ein 2^d-Baum über P und einer entsprechenden Strukturierung, wie oben erläutert, so dass jede Nachbarschaft höchstens ρ Knoten enthält und jeder Knoten geometrischer Nachbar von höchstens $\bar{\rho}$ Knoten ist. Falls G mindestens $\frac{1}{4}\epsilon n$ benachbarte verletzende Knotenpaare hat, dann enthält ein zufällig gleichverteilt ohne Zurücklegen gezogenes Knotensample der Größe $s_{5.1} = \mathcal{O}\left(\sqrt{\frac{\rho\cdot\bar{\rho}}{\epsilon}\cdot n}\right)$ mit Wahrscheinlichkeit mindestens $\frac{5}{6}$ mindestens eins davon.*

Beweis. Sei M die Menge der gesampelten Knoten. Wir unterteilen M in Teilmengen M_1 und M_2 der Größen s_1 und s_2, $s_{5.1} = s_1 + s_2$, und zeigen, dass mit Wahrscheinlichkeit je $\frac{11}{12}$ eins der benachbarten verletzenden Knotenpaare einen Knoten in M_1 und einen in M_2 hat.

Sei B die Menge der Knoten, die zusammen mit mindestens einem Nachbarn ein verletzendes Knotenpaar bilden. Da jeder Knoten maximal ρ Nachbarn hat, gilt $|B| \geq \frac{1}{4\rho}\epsilon n$. Da $|B \cap M_1|$ hypergeometrisch verteilt ist, ist die Anzahl X an gezogenen Samples, die nötig sind, um mindestens κ Elemente aus B zu erhalten, negativ hypergeometrisch verteilt. Daher gilt

$$\mathrm{E}[X] = \frac{\kappa(n+1)}{|B|+1} \leq \frac{\kappa(n+1)}{(\epsilon n/4\rho)+1} \leq \frac{2\kappa n}{\epsilon n/4\rho} = \frac{8\kappa\rho}{\epsilon};$$

Aufgrund der Markow-Ungleichung gilt für $s_1 := \frac{96\kappa\rho}{\epsilon}$

$$\Pr[X > s_1] \leq \Pr[X > 12 \cdot \mathrm{E}[X]] \leq \frac{1}{12}.$$

Wir nehmen nun an, dass $|B \cap M_1| \geq \kappa$ gilt. Da wir angenommen haben, dass jeder Knoten zu höchstens $\bar{\rho}$ Knoten benachbart ist, gibt es mindestens $\kappa/\bar{\rho}$ Knoten, die mit den Knoten in $B \cap M_1$ benachbarte verletzende Paare bilden; die Wahrscheinlichkeit, keinen davon zu sampeln, ist kleiner als $\left(1 - \frac{\kappa}{\bar{\rho}n}\right)^{s_2} \leq e^{-\ln 12} = \frac{1}{12}$ für $s_2 := \ln 12 \cdot \bar{\rho}n/\kappa$.

Aufgrund der Union-Bound ist die Wahrscheinlichkeit, dass beide Ereignisse eintreten, mindestens $1 - \frac{1}{12} - \frac{1}{12} = \frac{5}{6}$. Für $\kappa := \sqrt{\frac{\bar{\rho}}{\rho}\epsilon n}$ gilt außerdem $s_1 = 96\sqrt{\rho \cdot \bar{\rho}/\epsilon} \cdot \sqrt{n}$ und $s_2 = \ln 12 \cdot \sqrt{\bar{\rho} \cdot \rho/\epsilon} \cdot \sqrt{n}$ und damit $s_{5.1} = s_1 + s_2 \leq \frac{(96+\ln 12)\sqrt{\bar{\rho}\cdot\rho}}{\epsilon^{1/2}} \cdot \sqrt{n} = \mathcal{O}\left(\frac{\bar{\rho}^{0.5}\rho^{0.5}}{\epsilon^{0.5}} \cdot \sqrt{n}\right)$. □

Wir zeigen nun, dass für Punkte in sampelbaren Boxen das von TesteSpanner explorierte Areal tatsächlich groß genug ist, um alle Punkte in der Nachbarschaft des Ausgangspunkts zu enthalten. Dazu benötigen wir zunächst eine obere Schranke für die Anzahl der Punkte, die in Nachbarschaften und erweiterten Nachbarschaften von sampelbaren Boxen liegen können.

Lemma 5.7. *Sei $G = (P, E)$ ein geometrischer Graph wie oben mit n Knoten und einem 2^d-Baum H und einer entsprechenden Strukturierung wie oben erläutert; es gelte $0 < \epsilon, \delta < 1$. Sei c eine sampelbare Box: Dann enthält die Nachbarschaft von c höchstens $t \cdot 3^{di^*} + k$ Knoten; die erweiterte Nachbarschaft von c enthält höchstens $t \cdot ((2\sqrt{d} \cdot 3^{i^*+1})^d + k)$ Knoten. Damit enthält sowohl die Nachbarschaft als auch die erweiterte Nachbarschaft von c $\mathcal{O}(\delta^{-4d}\epsilon^{-3}\log^6 \Delta)$ Knoten.*

Beweis. Da die Nachbarschaft von c durch einen Würfel der Seitenlänge $3^{i^*} \cdot w(c)$ definiert ist, enthält sie maximal $(3^{i^*})^d + k$ Boxen des 2^d-Baums H – höchstens $(3^{i^*})^d$ Boxen der Größe $w(c)$ oder größer und, da $c(p)$ sampelbar ist, maximal k kleinere Boxen. Analog enthält die erweiterte Nachbarschaft von c höchstens $(2\sqrt{d} \cdot 3^{i^*+1})^d + k$ Boxen, da sie durch einen Würfel der Seitenlänge $2\sqrt{d} \cdot 3^{i^*+1}$ definiert ist.

Da laut Definitionvon H jede Box maximal t Knoten enthält, ist die Anzahl der Knoten in der Nachbarschaft von c also beschränkt durch $t \cdot 3^{di^*} + k$; die Anzahl der Knoten in der erweiterten Nachbarschaft von c ist höchstens $t \cdot ((2\sqrt{d} \cdot 3^{i^*+1})^d + k)$. Für die Abschätzung im $\mathcal{O}$-Kalkül bemerken wir, dass $(2\sqrt{d} \cdot 3^1)^d = \mathcal{O}(1)$ gilt und damit beide Knotenzahlen sich nach

$$
\begin{aligned}
& t \cdot (\mathcal{O}(1) \cdot 3^{di^*} + k) \\
&\quad = \frac{2^{d+3}\log \Delta}{\epsilon} \cdot (3^{i^*+1})^d \cdot \left\lceil \log_3 \frac{2\Delta}{3^{i^*}} \right\rceil \left(\mathcal{O}(1) \cdot 3^{di^*} + \epsilon \cdot 2^{2d+3}(\sqrt{d})^d \cdot 3^{2di^*+d} t \log^2 \Delta\right) \\
&\quad = \mathcal{O}\left(\frac{\log^2 \Delta}{\delta^d \epsilon}\right) \cdot \left(\mathcal{O}(\delta^{-d}) + \mathcal{O}\left(\frac{\log^4 \Delta}{\delta^{3d}\epsilon^2}\right)\right) = \mathcal{O}(\delta^{-4d}\epsilon^{-3}\log^6 \Delta)
\end{aligned}
$$

abschätzen. □

Wir benötigen auch eine obere Schranke für die Anzahl der Knoten, deren geometrischer Nachbar ein Knoten sein kann.

Lemma 5.8. *Sei $G = (P, E)$ ein geometrischer Graph wie oben mit n Knoten und einem 2^d-Baum H und einer entsprechenden Strukturierung wie oben erläutert; es gelte*

$0 < \epsilon, \delta < 1$. *Dann ist jeder Knoten geometrischer Nachbar von höchstens* $t \cdot 3^{di^*} \log \Delta = \mathcal{O}(\delta^{-4d}\epsilon^{-3} \log^7 \Delta)$ *anderen Knoten.*

Beweis. Sei p ein fester aber beliebiger Knoten. Es gibt maximal $\log \Delta$ verschiedene Größen von Boxen, und aufgrund der Definition der Nachbarschaft gibt es pro Boxgröße höchstens $(3^{i^*})^d$ Boxen, in deren Nachbarschaft p liegen kann, also höchstens $3^{di^*} \log \Delta$ Boxen insgesamt. Jede dieser Boxen enthält maximal t Knoten, so dass die Gesamtzahl an Knoten, deren geometrischer Nachbar p ist, maximal $t \cdot 3^{di^*} \log \Delta = \mathcal{O}(\delta^{-4d}\epsilon^{-3} \log^7 \Delta)$ ist; die Abschätzung im $\mathcal{O}$-Kalkül ist analog zu der im Beweis von Lemma 5.7. □

Das folgende Lemma garantiert, dass die Dijkstra-Suchen in der dritten Zeile von TesteSpanner mindestens einen Knoten entdecken, der mehr als $(1 + \delta)$ mal so weit vom jeweiligen Startknoten p entfernt ist wie jeder beliebige Punkt in $c(p)$; das ist wichtig aufgrund der Rolle des Wertes W im Algorithmus.

Lemma 5.9. *Sei $G = (P, E)$ ein geometrischer Graph wie oben mit n Knoten und einem 2^d-Baum H und einer entsprechenden Strukturierung wie oben erläutert; es gelte $0 < \epsilon, \delta < 1$. Sei p ein beliebiger von* TesteSpanner(n, G, δ, ϵ) *gesampelter Knoten, derart dass $c(p)$ eine sampelbare Box ist. Für $s_{5.2} := t \cdot ((2\sqrt{d} \cdot 3^{i^*+1})^d + k) + 1$ exploriert die von p ausgehende Dijkstra-Suche in* TesteSpanner *mindestens ein Knoten r, so dass $||p - r||_2 > (1 + \delta)||p - q||_2$ für alle $q \in \Gamma(c(p))$.*

Beweis. $c(p)$ ist sampelbar, die erweiterte Nachbarschaft $\hat{\Gamma}(c(p))$ enthält also laut Lemma 5.7 maximal $t \cdot ((2\sqrt{d} \cdot 3^{i^*+1})^d + k)$ Knoten, einen weniger als die Dijkstra-Suche exploriert. Es wird also mindestens ein Knoten r außerhalb von $\hat{\Gamma}(c(p))$ gefunden.

$\hat{\Gamma}(c(p))$ ist durch einen Würfel der Seitenlänge $2\sqrt{d} \cdot 3^{i^*+1} \cdot w(c(p))$ definiert, der um $c(p)$ zentriert ist. Da p in $c(p)$ liegt, folgt, dass die euklidische Distanz zwischen p und r mindestens

$$\begin{aligned}\sqrt{d} \cdot 3^{i^*+1} \cdot w(c(p)) - \frac{1}{2} w(c(p)) &= (3\sqrt{d} \cdot 3^{i^*} - \frac{1}{2}) \cdot w(c(p)) \\ &> 2\sqrt{d} \cdot 3^{i^*} \cdot w(c(p)) > (1 + \delta)\sqrt{d} \cdot 3^{i^*} \cdot w(c(p))\end{aligned}$$

ist. Letzerer Wert ist allerdings genau $(1 + \delta)$ mal die Länge der Diagonalen des $\Gamma(c(p))$ definierenden Würfels, also mehr als $(1 + \delta)$ mal die größte mögliche euklidische Distanz eines Punkts $q \in \Gamma(c(p))$ zu p. □

Wir fassen die obigen Ergebnisse nun im folgenden Theorem zusammen:

Theorem 5.1. *Sei G ein gerichteter geometrischer Graph im $\mathbb{R}^d$ mit einem durch $D \in \mathbb{N}$ beschränkten Knotenausgangsgrad, dessen Knoten auf einem Gitter der Weite Δ liegen. Dann testet der Algorithmus* TesteSpanner *mit einer Anfragekomplexität von $\mathcal{O}(\delta^{-8d}\epsilon^{-6.5} \log^{12.5} \Delta \cdot \sqrt{n})$, ob G ein $(1 + \delta)$-Spanner ist.*

Beweis. Wir nehmen zunächst an, dass G ein $(1 + \delta)$-Spanner ist und zeigen, dass TesteSpanner dann mit Wahrscheinlichkeit 1 akzeptiert. Sei p ein fester aber beliebiger

gesampelter Knoten: Dijkstras Algorithmus entdeckt von p aus die Knoten von G in aufsteigender Reihenfolge ihrer Graphdistanz zu p. Sei wie im Algorithmus R die Menge der durch diesen Dijkstra-Aufruf entdeckten Knoten, sei W die maximale euklidische Distanz eines entdeckten Knotens zu p und sei r dieser Knoten: Es gilt $d_G(p,r) \geq ||p-r||_2 = W$.

Sei q ein fester aber beliebiger in der ersten Zeile von TesteSpanner gesampelter Knoten mit $||p-q||_2 < \frac{1}{(1+\delta)} \cdot W$ – nur für solche Knoten wird die Spanner-Eigenschaft überprüft. Wir müssen zeigen, dass für q keine der Bedingungen gilt, die in der drittletzten Zeile geprüft werden, denn andernfalls würde TesteSpanner die Eingabe irrtümlich ablehnen.

Zunächst zeigen wir, dass q in R enthalten ist: Aufgrund der $(1+\delta)$-Spanner-Eigenschaft gilt $d_G(p,q) \leq (1+\delta)||p-q||_2 < (1+\delta)\frac{1}{(1+\delta)} \cdot W \leq d_G(p,r)$, und aufgrund der Eigenschaften von Dijkstras Algorithmus wurde q daher vor r erreicht, ist also in R enthalten. Insbesondere wurde dabei auch ein kürzester Pfad von p nach q gefunden, der die $(1+\delta)$-Spanner-Eigenschaft erfüllt.

Es bleibt zu zeigen, dass auch ein Rückpfad von q nach p gefunden wird, der die $(1+\delta)$-Spanner-Eigenschaft erfüllt – dass G $(1+\delta)$-Spanner ist, garantiert die Existenz eines solchen Pfades P. Alle Knoten auf dem Pfad P haben eine euklidische Distanz von höchstens $(1+\delta)||p-q||_2$ zu p: Gäbe es einen Knoten s mit $||p-r||_2 > (1+\delta)||p-q||_2$, dann würde auch $d_G(s,p) > (1+\delta)||p-q||_2$ gelten, und damit wäre schon der Teilpfad von P, der von s nach p führt, länger als $(1+\delta)||p-q||_2$ – ein Widerspruch zu der Annahme, dass P höchstens eine Länge von $(1+\delta)||p-q||_2$ hat. Damit liegen alle Knoten s von P in R, denn es gilt $d_G(s,p) \leq (1+\delta)||p-q||_2 < (1+\delta)\frac{1}{(1+\delta)}W$; alle diese Knoten werden also von der Dijkstra-Suche in Zeile 3 vor r entdeckt und gehören somit zum durch R induzierten Subgraphen von G. Die Dijkstra-Suche in Zeile 7 von TesteSpanner findet P also.

Wir nehmen nun an, dass G ϵ-fern von jedem $(1+\delta)$-Spanner ist und zeigen, dass TesteSpanner in diesem Fall mit hoher Wahrscheinlichkeit ablehnt. Sei (p,q) ein verletzendes Knotenpaar mit $q \in \Gamma(c(p))$, so dass sowohl p als auch q in Zeile 1 des Algorithmus gesampelt wurden. Da (p,q) ein verletzendes Knotenpaar ist, gibt es keinen kurzen Pfad von p nach q oder keinen kurzen Pfad von q nach p. Wir nehmen außerdem an, dass $c(p)$ sampelbar ist: Daher folgt aus Lemma 5.9, dass die von p ausgehende Dijkstra-Suche einen Knoten exploriert, dessen Distanz zu p mehr als $(1+\delta)||p-q||_2$ ist – es gilt also $||p-q||_2 < \frac{1}{(1+\delta)}W$ in Zeile 6 des Algorithmus, und damit gibt es einen Schleifendurchlauf der inneren *Foreach*-Schleife für q. Da allerdings $d_G(p,q) > (1+\delta)||p-q||_2$ oder $d_G(q,p) > (1+\delta)||p-q||_2$ gilt, wird q von p aus entweder nicht exploriert, oder einer der beiden gefundenen kürzesten Pfade ist zu lang: Eine der Überprüfungen in der drittletzten Zeile des Algorithmus trifft also zu, und die Eingabe wird daher abgelehnt.

Es bleibt zu zeigen, dass ein solches verletzendes Knotenpaar (p,q) tatsächlich mit hoher Wahrscheinlichkeit im in Zeile 1 gezogenen Sample vorhanden ist. Wegen Korollar 5.1 wissen wir, dass es in G mindestens $\frac{1}{4}\epsilon n$ solche benachbarte verletzende Knotenpaare in sampelbaren Boxen gibt. Zudem garantiert Lemma 5.7, dass jede Nachbarschaft höchstens $\rho := \mathcal{O}(\delta^{-4d} \cdot \epsilon^{-3} \log^6 \Delta)$ Knoten enthält. Nach Lemma 5.8 ist außerdem jeder Knoten Nachbar von höchstens $\bar{\rho} := t \cdot 3^{di^*} \log \Delta = \mathcal{O}(\delta^{-4d}\epsilon^{-3} \log^7 \Delta)$ weiteren Knoten.

Mit diesen Werten für ρ und $\bar{\rho}$ garantiert Lemma 5.6, dass ein Knotensample der Größe $s_{5.1} = \mathcal{O}(\delta^{-4d}\epsilon^{-2.5}\log^{6.5}\Delta\sqrt{n})$ mit Wahrscheinlichkeit mindestens $\frac{5}{6}$ ein benachbartes verletzendes Knotenpaar wie oben gefordert enthält; mit ebendieser Wahrscheinlichkeit wird also aufgrund der obigen Überlegungen die Eingabe abgelehnt.

Da für jeden gesampelten Knoten ein Dijkstra-Durchlauf gestartet wird, der maximal $\mathcal{O}(\delta^{-4d}\epsilon^{-3}\log^6\Delta)$ Knoten erkundet, ergibt sich insgesamt eine Anfragekomplexität von $\mathcal{O}(\delta^{-8d}\epsilon^{-6.5}\log^{12.5}\Delta\cdot\sqrt{n})$. Dazu sei bemerkt, dass die Dijkstra-Suche in der dritten Zeile den induzierten Subgraphen der erreichten Knotenmenge R komplett exploriert, ausgenommen der Kanten von und zu dem am weitesten vom Ausgangsknoten p entfernten Knoten $r \in R$; aufgrund von Lemma 5.9 kann aber kein kurzer Pfad von oder zu Knoten in $\Gamma(c(p))$ über r führen, und damit muss für die Dijkstra-Suchen in Zeile 7 keine neue Anfrage an den Graphen gestellt werden. □

Zuletzt bleibt noch die Laufzeitanalyse von TesteSpanner. Insbesondere wollen wir dabei zeigen, dass über die $\Theta(n)$ Paaren von Knoten, die sich aus dem Knotensample von TesteSpanner ergeben, derart iteriert weden kann, dass nur relevante Knotenpaare berücksichtigt werden und sich dadurch eine erwartete Anzahl von $\tilde{\mathcal{O}}(\sqrt{n})$ Schleifendurchläufen ergibt.

Theorem 5.2. *Der Algorithmus* TesteSpanner *lässt sich so implementieren, dass er eine Laufzeit von* $\tilde{\mathcal{O}}(\delta^{-12d}\epsilon^{-9.5}\log^{\max\{18.5,d+7.5\}}\Delta\cdot\log\log\Delta\cdot\sqrt{n})$ *erreicht.*

Beweis. Zur Organisation der in Zeile 1 von TesteSpanner gesampelten Knoten verwenden wir einen *Bereichsbaum* T (siehe z.B. [17]): Um dann in Zeile 6 des Algorithmus über möglichst wenige Knoten zu iterieren, wollen wir eine Kandidatenmenge durch Bereichsanfrage an T erhalten. Diese Kandidatenmenge soll alle Knoten aus T beinhalten, die in der Nachbarschaft des gerade in der äußeren Schleife behandelten Knotens p liegen, und möglichst wenige weitere Knoten.

Da die Gesamtzahl der in Zeile 1 gesampelten Knoten $s_{5.1} = \mathcal{O}(\delta^{-4d}\epsilon^{-2.5}\log^{6.5}\Delta\sqrt{n})$ ist, können wir T in Laufzeit $\mathcal{O}\left(\delta^{-4d}\epsilon^{-2.5}\log^{6.5}\Delta\sqrt{n}\cdot\log^d(\delta^{-4d}\epsilon^{-2.5}\log^{6.5}\Delta\sqrt{n})\right) = \mathcal{O}(\delta^{-4d}\epsilon^{-2.5}\cdot\log^{d+6.5}\Delta\cdot\sqrt{n}\log^d n)$ erstellen (siehe [17]).

Wir betrachten nun einen beliebigen Durchlauf der äußeren Schleife von TesteSpanner; sei p der entsprechende gesampelte Knoten, von dem aus der Graph in Zeile 3 mittels Dijkstras Algorithmus exploriert wurde, und sei R die größte euklidische Distanz eines dabei explorierten Knotens zu p.

Wie oben angedeutet, soll der an T angefragte Bereich ungefähr der Nachbarschaft $\Gamma(c(p))$ entsprechen; das Problem dabei ist, dass TesteSpanner die geometrische Größe von $\Gamma(c(p))$ nicht kennt. Die richtige Größe kann allerdings wie folgt ermittelt werden: Startend mit der kleinstmöglichen Seitenlänge einer Nachbarschaft, $3^{i^*}\cdot\lceil\sqrt[d]{t}\rceil$, werden alle Knoten in einem um p zentrierten Würfel dieser Seitenlänge im Bereichsbaum angefragt. Werden dabei höchstens $\hat{t} := t\cdot((2\sqrt{d}\cdot 3^{i^*+1})^d + k) = \mathcal{O}(\delta^{-4d}\epsilon^{-3}\log^6\Delta)$ Knoten gefunden, wird die Seitenlänge so lange verdoppelt, bis das erste Mal mehr als $\hat{t}$ Knoten gefunden werden. In diesem Fall wird die Bereichsanfrage abgebrochen, und die vorhergehende Bereichsanfrage liefert die geforderten Knoten. Nach höchstens $\mathcal{O}(\log\Delta)$ Wiederholungen

wird das Verfahren abgebrochen, da der angefragte Bereich dann den kompletten Graphen beinhaltet. Wir benennen die Würfel, die die angefragten Regionen definieren, mit $W_1, \ldots, W_{\log \Delta}$.

Um die Korrektheit dieses Verfahrens zu zeigen, nehmen wir zunächst an, dass p in einer sampelbaren Zelle liegt. Sei W_i der erste Würfel, der um p zentriert ist und für den die Anfrage an T mehr als $\hat{t}$ Knoten zurückliefert: W_i muss Knoten außerhalb von $\hat{\Gamma}(c(p))$ enthalten, denn $\hat{\Gamma}(c(p))$ enthält wegen Lemma 5.7 höchstens $\hat{t}$ Knoten, und da TesteSpanner ohne Zurücklegen sampelt, können daher auch nur $\hat{t}$ Knoten aus $\hat{\Gamma}(c(p))$ zum Sample aus Zeile 1 gehören.

W_i enthält also Bereiche außerhalb von $\hat{\Gamma}(c(p))$ und hat mindestens eine Seitenlänge von $(6\sqrt{d} \cdot 3^{i^*} - \frac{1}{2}) \cdot w(c(p)) > 5\sqrt{d} \cdot 3^{i^*} \cdot w(c(p))$: $\hat{\Gamma}(c(p))$ ist durch einen Würfel der Seitenlänge $2\sqrt{d} \cdot 3^{i^*+1}$ definiert. Die Verschiebung um maximal $\frac{1}{2}w(c(p))$ kann dabei entstehen, wenn p auf dem Rand von $c(p)$ liegt. $\Gamma(c(p))$ hat eine Seitenlänge von $3^{i^*} \cdot w(c(p))$ und ist daher komplett im nächstkleineren Würfel W_{i-1} enthalten: Dieser hat eine Seitenlänge von mindestens $\frac{5}{2}\sqrt{d} \cdot 3^{i^*} \cdot w(c(p))$. Also werden von der Anfrage an T mit Bereich W_{i-1} alle Knoten zurückgegeben, die in Zeile 1 von TesteSpanner gesampelt wurden und die in $\Gamma(c(p))$ liegen.

Wir nehmen nun an, dass p in einer schweren Zelle liegt: In diesem Fall liefert die Anfrage an den Bereichsbaum eventuell nicht die geforderte Knotenmenge zurück; dies spielt aber keine Rolle, da wir nicht fordern, dass der Algorithmus verletzende Knotenpaare in schweren Zellen entdeckt.

Jede Anfrage an den Bereichsbaum hat eine Laufzeit von höchstens $\mathcal{O}(\hat{t} + \log^d s_{5.1}) = \mathcal{O}(\delta^{-4d}\epsilon^{-3} \log^6 \Delta + \log^d \Delta n) = \mathcal{O}(\log^d \Delta n)$, da die Anfragen abgebrochen werden, sobald mehr als $\hat{t}$ Knoten zurückgeliefert wurden; pro gesampeltem Knoten werden höchstens $\mathcal{O}(\log \Delta)$ Anfragen benötigt. Zusammen mit der Anzahl $s_{5.1}$ gesampelter Knoten ergibt das für die Zugriffe auf den Bereichsbaum eine Laufzeit von $s_{5.1} \cdot \mathcal{O}(\log^d \Delta n \cdot \log \Delta) = \mathcal{O}(\delta^{-4d}\epsilon^{-2.5} \log^{d+7.5} \Delta \cdot \sqrt{n} \log^d n)$.

Es bleibt die für die Dijkstra-Explorierungen nötige Laufzeit zu beschränken. Jede der Explorierungen in Zeile 3 von TesteSpanner hat eine Laufzeit von $\mathcal{O}(s_{5.2} \cdot \log s_{5.2})$, ebenso jede der höchstens $\hat{t}$ Rückwärtsexplorierungen in Zeile 7. Es ergibt sich dadurch eine Gesamtlaufzeit von $\mathcal{O}(s_{5.1} \cdot \hat{t} \cdot s_{5.2} \log s_{5.2}) = \mathcal{O}(\delta^{-12d}\epsilon^{-9.5} \log^{18.5} \Delta \cdot \log \log \Delta)$.

Insgesamt ergibt sich eine Laufzeit von $\tilde{\mathcal{O}}(\delta^{-12d}\epsilon^{-9.5} \log^{\max\{18.5, d+7.5\}} \Delta \cdot \log \log \Delta \cdot \sqrt{n})$. □

Bisher sind wir in diesem Kapitel davon ausgegangen, dass wir die Spanner-Eigenschaft in gerichteten Graphen testen wollen; der Algorithmus TesteSpanner kann aber natürlich auch in ungerichteten Graphen eingesetzt werden. Die einzige Erleichterung gegenüber gerichteten Graphen ist, dass ein kürzester Pfad von einem Knoten p zu einem weiteren Knoten q automatisch auch ein kürzester Pfad von q nach p ist; die Dijkstra-Aufrufe in Zeile 7 von TesteSpanner können also entfallen. Die Anfragekomplexität bleibt dadurch gleich, während bei der Laufzeit ein Faktor von maximal $\mathcal{O}(\delta^{-4}\epsilon^{-3} \log^6 \Delta)$ eingespart werden kann. In Bezug auf die Abhängigkeit von der Knotenzahl n ist der Algorithmus TesteSpanner also in gerichteten Graphen gleich schnell wie in ungerichteten.

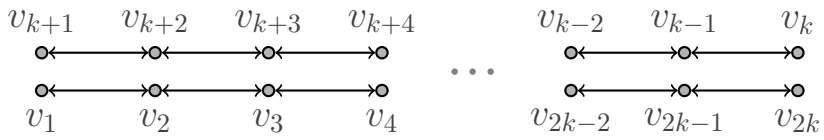

Abbildung 5.5: *Beispielhafter Graph der Klasse $\mathcal{C}_n$; Knoten mit gleichen y-Koordinaten werden hier der Übersichtlichkeit halber untereinander dargestellt.*

5.3 Eine untere Schranke für das Testen von euklidischen Spannern

In diesem Kapitel zeigen wir eine untere Schranke von $\Omega(n^{1/3})$ für die Anfragekomplexität jedes Property-Testing-Algorithmus mit einseitigem Fehler für die $(1+\delta)$-Spanner-Eigenschaft. Wir definieren dazu die folgende Klasse $\mathcal{C}$ von gerichteten geometrischen Graphen auf der eindimensionalen Linie:

Definition 5.12. *Sei $G_n(P_n, E_n)$ für $k \in \mathbb{N}$ und $n = 2k$ der folgende geometrische Graph: P_n bestehe aus Knoten $v_1, \ldots, v_n$, wobei $(i \mod k)$ die x-Koordinate von v_i ist. E_n enthalte für $i = 1, \ldots, k-1$ jeweils Kanten (v_i, v_i+1), (v_i+1, v_i), (v_{k+i}, v_{k+i+1}) und (v_{k+i+1}, v_{k+i}).*

Sei $\mathcal{C}_n$ die Klasse aller Graphen, die durch Permutation der Knotennummern von G_n (nicht aber ihrer Koordinaten) entstehen. Wir definieren $\mathcal{C} := \bigcup_{k\geq 3} \mathcal{C}_{2k}$.

Anschaulich besteht $\mathcal{C}$ aus Graphen, deren Knoten in zwei übereinander liegenden Linien angeordnet sind (siehe Abbildung 5.5); benachbarte Knoten der Linien sind jeweils durch Hin- und Rückkante miteinander verbunden, so dass jede der Linien einzeln ein $(1+\delta)$-Spanner für alle $\delta > 0$ ist. Andererseits sind die beiden Linien nicht verbunden, und um aus einem Graphen $G \in \mathcal{C}$ einen $(1+\delta)$-Spanner für ein beliebiges $\delta < \infty$ zu erzeugen, müssten zwischen allen Knotenpaaren v_i und v_{i+k} Hin- und Rückkanten eingefügt werden, insgesamt also $|V(G)|$ Kanten. Damit sind alle Graphen in $\mathcal{C}$ ϵ-fern von der $(1+\delta)$-Spanner-Eigenschaft für alle $\epsilon < 1$ und $\delta < \infty$.

Beobachtung 5.1. *Sei $G \in \mathcal{C}$ ein Graph mit n Knoten; seien $\epsilon < 1$ und $\delta < \infty$. Dann ist G ϵ-fern von jedem $(1+\delta)$-Spanner H.*

Wir zeigen nun die untere Schranke für das Testen von euklidischen Spannern mit einseitigem Fehler. Der Beweis ist an den der unteren Schranke für das Testen von minimalen Spannbäumen von Ben-Zwi et al. ([16], siehe auch Kapitel 2.2.4) angelehnt.

Theorem 5.3. *Jeder Property-Testing-Algorithmus mit einseitigem Fehler für die $(1+\delta)$-Spanner-Eigenschaft benötigt auf Graphen mit n Knoten $\Omega(n^{1/3})$ Anfragen.*

Beweis. Wir zeigen, dass jeder deterministische Algorithmus, der alle $(1+\delta)$-Spanner akzeptiert und höchstens $q := \frac{1}{4}n^{1/3}$ Anfragen an den Eingabegraphen richtet, mit Wahrscheinlichkeit kleiner als $\frac{2}{3}$ ablehnt, wenn ein zufällig gleichverteilt gewählter Graph mit

n Knoten aus $\mathcal{C}$ eingegeben wird. Wie in Kapitel 2 zur Anwendung von Yaos Minimax-Prinzip ausgeführt, ist dies hinreichend, um zu zeigen, dass auch kein Property-Testing-Algorithmus mit einseitigem Fehler für $(1+\delta)$-Spanner existiert, der nur q Anfragen benötigt und mit Wahrscheinlichkeit $\frac{2}{3}$ jeden Graphen ablehnt, der ϵ-entfernt von der $(1+\delta)$-Spanner-Eigenschaft ist.

Sei $\mathcal{A}$ ein beliebiger deterministischer Algorithmus, der q Anfragen an G stellt. Sei G ein zufällig gleichverteilt aus $\mathcal{C}$ gewählter Graph mit $n = 2k$ Knoten; seien $\epsilon < 1$ und $\delta < \infty$. Da $k \geq 3$, gilt $n > 3n^{1/3} \geq 12q$.

$\mathcal{A}$ greift auf G auf zwei Weisen zu: $\mathcal{A}$ kann unter Angabe der Knotennummer für einen Knoten seine Punktkoordinaten abfragen, und, ebenfalls unter Angabe der Knotennummer, Zugriff auf dessen Adjazenzlisteneinträge erhalten; werden diese angefragt, so wird ebenfalls die Knotennummer des adjazenten Knotens zurückgegeben. Dabei kann $\mathcal{A}$ sowohl Anfragen an bisher nicht bekannte Knoten richten – die Knotennummern dieser Knoten sind zu Beginn des Algorithmus vorgegeben, da $\mathcal{A}$ ein deterministischer Algorithmus ist – als auch adaptiv Anfragen an die Adjazenzlisten schon entdeckter Knoten.

Im Prinzip lässt sich $\mathcal{A}$ also als zweiphasig beschreiben: In der ersten Phase wird eine vorgegebene Menge M von Knoten angefragt; von jedem dieser Knoten ausgehend exploriert der Algorithmus möglicherweise weitere Knoten, indem sukzessive weitere Adjazenzlisteneinträge angefragt werden. Sei für $p \in M$ M_p die Menge der Knoten, die so von p aus entdeckt werden. Aufgrund der Struktur von G lässt sich der von $\mathcal{A}$ explorierte Bereich von G wie folgt beschreiben: Ausgehend von den Knoten $p \in M$ werden Pfade von p nach links und rechts erforscht. Da $\mathcal{A}$ nur q Anfragen an G stellt, gilt $|M| \leq q$ und $\sum_{p \in M} |M_p| \leq q$.

Da G ein zufällig gleichverteilt gewählter Graph aus $\mathcal{C}$ ist und die Knoten aus M sich durch Anfragen an Knoten mit vorgegebenen Nummern ergeben, lässt sich M als eine Menge zufällig gleichverteilt gewählter Knoten in Bezug auf den Graphen G_n (siehe Definition 5.12) sehen. Wir werden im weiteren Verlauf des Beweises annehmen, dass G_n der Eingabegraph ist und $\mathcal{A}$ die Menge M zufällig gleichverteilt aus den Knoten von G_n sampelt.

Da $\mathcal{A}$ einen einseitigen Fehler hat, lehnt $\mathcal{A}$ G nur dann ab, wenn ein Zeuge gegen die $(1+\delta)$-Spanner-Eigenschaft gefunden ist. Ein solcher Zeuge ist ein Paar p, q von Knoten mit gleichen Koordinaten, die nicht miteinander verbunden sind. Alle explorierten Teilgraphen von G, die keinen solchen Zeugen enthalten, lassen sich zu einem $(1+\delta)$-Spanner vervollständigen, und da wir vorausgesetzt haben, dass $\mathcal{A}$ jeden $(1+\delta)$-Spanner akzeptiert, muss $\mathcal{A}$ in diesem Fall auch G akzeptieren.

$\mathcal{A}$ kann also nur dann ein Zeugenpaar identifizieren, wenn zwei Knoten $p, q \in M$ den beiden unterschiedlichen Linien von G angehören und einen so geringen Abstand voneinander haben, dass M_p und M_q sich geometrisch überschneiden können; das heißt, wenn die Menge M der besuchten Knoten zwei Knoten v_i, v_{k+j} mit $|i-j| \leq q$ enthält. Da $\sum_{p \in M} |M_p| \leq q$ gilt, können sich die von v_i und v_{k+j} ausgehend explorierten Bereiche nur dann geometrisch überschneiden. Für einen Knoten v_i sei E_i das Ereignis, dass ein solcher Knoten v_{k+j} gesampelt wird; sei E das Ereignis, dass E_i für einen beliebigen Knoten v_i eintritt.

Die Wahrscheinlichkeit, dass in einem einzelnen Sampelschritt ein bestimmter Knoten

v_{k+j} gezogen wird, ist höchstens $\frac{1}{n-q}$; nach der Union-Bound ist die Wahrscheinlichkeit, dass v_{k+j} in einem der maximal q Sampelschritte gezogen wird, also kleiner als $\frac{q}{n-q}$. Die Wahrscheinlichkeit, dass ein beliebiger der $2q$ Knoten im entsprechenden Abstand von v_i gezogen wird, ist daher höchstens $\frac{2q^2}{n-q}$, wieder wegen der Union-Bound. Es gilt also $\Pr[E_i] < \frac{2q^2}{n-q} < \frac{24q^2}{11n}$ wegen $n > 12q$, und mit Hilfe der Union-Bound folgt

$$\Pr[E] = \Pr\left[\bigcup_{1\leq i\leq q} E_i\right] \leq \sum_{1\leq i\leq q} \Pr[E_i] \leq \sum_{1\leq i\leq q} \frac{24q^2}{11n} = \frac{24q^3}{11n} < \frac{6}{11} < \frac{2}{3};$$

Damit lehnt $\mathcal{A}$ G mit Wahrscheinlichkeit kleiner als $\frac{2}{3}$ ab; jeder Algorithmus $\mathcal{A}$, der einen zufällig gleichverteilten Graphen aus $\mathcal{C}$ mit Wahrscheinlichkeit mindestens $\frac{2}{3}$ ablehnt, benötigt also mehr als q Anfragen; da diese Wahrscheinlichkeitsschranke nach unserer Definition für Property-Testing-Algorithmen mit einseitigem Fehler gelten muss und nach Yaos Minimax-Prinzip benötigt also auch jeder Property-Testing-Algorithmus mit einseitigem Fehler für die euklidische Spanner-Eigenschaft mehr als q Anfragen. □

Die Eigenschaft, dass bei Graphen in $\mathcal{C}$ Knoten übereinander liegen können, lässt sich entfernen; wir definieren $\mathcal{C}'$ als die Klasse aller Graphen, die entstehen, wenn für $G \in \mathcal{C}$ für alle Knotenpaare v_i, v_{k+i} x-Koordinaten wie folgt gewählt werden: Einer der Knoten erhält die x-Koordinate i, der andere die x-Koordinate $i+\alpha$, wobei $\alpha < 1$ eine sehr kleine Konstante ist; für jedes δ lässt sich α hinreichend klein wählen, so dass das Einfügen von Hin- und Rückkanten zwischen allen Knotenpaaren v_i und v_{k+i} die $(1+\delta)$-Spanner-Eigenschaft herstellt. Da $\mathcal{C}'$ alle Graphen enthält, die sich so konstruieren lassen, kann kein Algorithmus die verschobenen Knotenpositionen ausnutzen.

6 Fazit und Ausblick

Thema dieser Arbeit war Property-Testing in dünn besetzten gerichteten Graphen, unter der Einschränkung, dass die eingehenden Kanten von Knoten für die Algorithmen nicht sichtbar sind – ein durch Anwendungsbeispiele wie Linkstrukturen im Internet oder Funkverbindungen in Ad-Hoc- oder Sensornetzen motiviertes, aber bisher wenig erforschtes Modell. Es konnten Property-Testing-Algorithmen für verschiedene Probleme in diesem Modell gezeigt werden, unter anderem für 3-Stern-Freiheit, starken Zusammenhang und euklidische Spanner.

Neben ihrer jeweiligen Einzelbedeutung haben diese Ergebnisse auch konzeptionelle Bedeutung für das Forschungsgebiet: Es konnte gezeigt werden, dass es überhaupt nichttriviale Grapheigenschaften gibt, die in diesem Modell in sublinearer Zeit getestet werden können. Eine grundlegende Technik dazu, die alle vorgestellten Algorithmen teilen, ist, dass „Kollisionen“ herbeigeführt werden: Eine Kollision tritt auf, wenn von k zusammengehörigen Elementen alle getrennt voneinander gesampelt werden und dadurch auch die gerichteten Kanten entdeckt werden, die diese Elemente verknüpfen. Im Fall von Subgraphfreiheit sind solche Elemente die k Quellkomponenten eines Vorkommens des verbotenen Subgraphen H, beim Tester für starken Zusammenhang sind es die $k \leq D$ eingehenden Kanten eines Knotens. Beim Property-Testing-Algorithmus für euklidische Spanner sind es $k = 2$ Knoten, die die Spannereigenschaft verletzen und gleichzeitig in einer bestimmten Art und Weise nahe beieinanderliegen: Hier ergibt sich die Notwendigkeit Kollisionen zu sampeln nicht direkt aus der gerichteten Natur der Graphen, sondern daraus, dass nicht alle nahe beieinanderliegenden Punkte auch durch kurze Pfade im Graphen miteinander verbunden sind und gerade diejenigen Paare von Punkten für den Algorithmus interessant sind, für die das nicht gilt. Die Gemeinsamkeit mit den anderen Problemen ist, dass auch hier ein Objekt möglicherweise von einem anderen aus nicht sichtbar ist und daher nur dann erkennbar ist, wenn beide Objekte gesampelt werden. Ansonsten bewirken die zusätzlichen geometrischen Eigenschaften des Spanner-Problems, dass der vorgestellte Algorithmus für gerichtete wie ungerichtete Graphen im Prinzip die gleiche Anfragekomplexität benötigt, obwohl die eingehenden Kanten von Knoten auch hier nicht sichtbar sind.

Um mit hoher Wahrscheinlichkeit k-Fach-Kollisionen herbeizuführen, sind $\Omega(n^{1-1/k})$ Anfragen an den Graphen erforderlich. Ein weiteres Ergebnis dieser Arbeit ist, dass diese Schranke durchbrochen werden kann, und in bestimmten Fällen auch Eigenschaften mit $o(n^{1-1/k})$ Anfragen getestet werden können, bei denen k zusammengehörige Elemente vorkommen können. Der in Kapitel 3 vorgestellte Property-Testing-Algorithmus für 3-Stern-Freiheit in schwach zusammenhängenden Graphen erreicht dies dadurch, dass er zusätzlich zum Erheben einer Kollisionsstatistik ein weiteres statistisches Merkmal des Graphen untersucht und mit den Ergebnissen der Kollisionsstatistik abgleicht. Dadurch

ist es möglich, statt einer 3-Fach-Kollisionsstatistik eine 2-Fach-Kollisionsstatistik zu verwenden und dadurch eine Anfragekomplexität von $\mathcal{O}(n^{1/2})$ statt $\mathcal{O}(n^{2/3})$ zu erreichen.

Nicht dem Kontext des Testens gerichteter Graphen zugehörig, aber trotzdem ein interessantes Ergebnis dieser Arbeit ist die Technik, mit der für den Property-Testing-Algorithmus für euklidische Spanner der geometrische Teil des Problems für die Analyse diskretisiert werden kann: Punktmengen werden durch 2^D-Bäume und exponentielle Gitter in gleich große, lokal abgeschlossene Bereiche eingeteilt, aufgrund derer eine Analyse möglich ist. Dabei steht insbesondere die logarithmisch beschränkte Zellenzahl eines exponentiellen Gitters in Verbindung mit dem Maß der ϵ-Entferntheit im Property-Testing, denn dadurch ist es möglich, für jede Gitterzelle eine Kante in einen von der Spannereigenschaft ϵ-entfernten Graphen einzufügen, so dass der entstehende Graph trotzdem noch $\Theta(\epsilon)$-entfernt von jedem Spanner ist und im Prinzip alle verbleibenden Verletzungen der Spannereigenschaft in den Basiszellen der exponentiellen Gitter liegen.

Im Folgenden werden einige offene Fragestellungen diskutiert, deren Beantwortung sich an die Ergebnisse dieser Arbeit anschließen könnte.

Testen und Klassifizieren von testbaren Eigenschaften. Ein seit Benders und Rons unterer Schranke [14] offenes Problem im Bereich des Property-Testing in dünn besetzten gerichteten Graphen ist Kreisfreiheit. Dieses Problem scheint von der Problemstruktur von starkem Zusammenhang deutlich abzuweichen: Zwar verlangt das Identifizieren eines Kreises als Zeugen gegen die Eigenschaft sozusagen eine Kollision – nämlich dass ein Knoten eines bereits erforschten Pfads als Zielknoten einer Kante des gleichen Pfads entdeckt wird –, und das spricht dafür, dass tatsächlich $\Omega(\sqrt{n})$ Anfragen eine untere Schranke für das Testen dieser Eigenschaft ist, wie von Bender und Ron vermutet; allerdings kann hier im Gegensatz zu starkem Zusammenhang überhaupt ein Zeuge gegen die Eigenschaft gefunden werden, ohne den kompletten Graphen zu lesen. Andererseits können die kleinsten Kreise in von Kreisfreiheit ϵ-entfernten Graphen sehr groß sein, zum Beispiel in der Knotenzahl logarithmische Länge haben. Reduktionsstrategien wie bei starkem Zusammenhang, die mit der Kontrahierung bestimmter Subgraphen arbeiten, versprechen daher keinen Erfolg; es erscheint nicht zuletzt deshalb unwahrscheinlich, dass das Approximieren der Verteilung eingehender Kanten für Kreisfreiheit hilfreich ist. Eventuell bieten Random-Walk-Strategien ähnlich wie beim Testen von Bipartitheit in ungerichteten Graphen ([42], siehe Kapitel 2.2.1) eine Lösung – dort werden mit Hilfe von Random Walks Kreise ungerader Länge identifiziert; allerdings entstehen einige neue Probleme beim Einsatz in gerichteten Graphen, so dass die Ansätze von Goldreich und Ron sich nicht einfach übertragen lassen.

Für starken Zusammenhang lässt sich eventuell ein effizienterer Algorithmus angeben als der in dieser Arbeit Vorgestellte. Es lassen sich zwar Graphklassen von stark zusammenhängenden und davon ϵ-entfernten Graphen konstruieren, so dass die Verteilung der Anzahl eingehender Kanten von Knoten $k-1$ proportionale Momente hat – und dies gilt auch schon, wenn alle Quellkomponenten der Graphen einzelne Knoten sind. Allerdings hat dies Auswirkungen auf die restliche Graphstruktur, die für unsere Betrachtungen im Gegensatz zu zum Beispiel 3-Stern-Freiheit nicht aus kleinen isolierten schwachen Zusam-

menhangskomponenten bestehen kann: Hat ein Graph mehr als $\Theta(\epsilon n)$ isolierte schwache Zusammenhangskomponenten, so enthält er viele kleine Senkenkomponenten und kann leicht in konstanter Zeit auf starken Zusammenhang getestet werden (siehe Kapitel 4).

Die Frage ist, ob man die restliche Graphstruktur algorithmisch nutzen kann, um, ähnlich wie bei 3-Stern-Freiheit in schwach zusammenhängenden Graphen gesehen, Messwerte zu gewinnen, die mit denen einer 2-Fach-Kollisionsstatistik abgeglichen werden können. Hilfreich dabei ist erstens die obige Beobachtung, dass angenommen werden kann, dass der Eingabegraph nur aus wenigen isolierten schwachen Zusammenhangskomponenten besteht; zweitens, dass die in Kapitel 4 vorgestellte Reduktionstechnik weiterhin Anwendung finden kann, das heißt angenommen werden kann, dass ϵ-entfernte Graphen viele Knoten mit Eingangsgrad 0 haben. Wenn der Knotengrad nicht im Exponenten von n vorkommt, bleibt jener trotz Reduktion der gleiche, und damit ist zum Beispiel ein Algorithmus mit Anfragekomplexität $\mathcal{O}(\sqrt{n})$ trotz Reduktion möglich.

Eine interessante Beobachtung ist zum Beispiel die Folgende: Wenn man in einem Graphen mit $\Theta(\epsilon n)$ Knoten ohne eingehende Kante eine Kante sampelt, dann gibt es nur $n - \Theta(\epsilon n)$ mögliche Zielknoten; sampelt man hingegen in einem stark zusammenhängenden Graphen eine Kante, dann gibt es n mögliche Zielknoten. Entspräche das Wählen des Zielknotens einer zufälligen Kante einem zufällig gleichverteilten Sample aller erreichbaren Knoten, dann ließe sich mit einer Samplegröße von $\mathcal{O}(\sqrt{n})$ die Anzahl solcher Knoten approximieren; damit ließen sich dann die obigen Graphklassen unterscheiden.

Allerdings handelt es sich nicht um ein zufällig gleichverteiltes Knotensample: Die Sampelwahrscheinlichkeit variiert je nach Eingangsgrad des Knotens um einen Faktor von maximal D. Die Frage ist, ob man ein annähernd gleichverteiltes Sample gewinnen kann, zum Beispiel durch geschicktes Wegwerfen von Samples oder durch das Wählen von Zielknoten von Random Walks.

In Bezug auf die Klassifizierung testbarer Grapheigenschaften ist eine weitere interessante Frage, ob es zusätzlich zu 3-Stern-Freiheit weitere Eigenschaften gibt, die sich in schwach zusammenhängenden Graphen einfacher testen lassen als in Graphen mit vielen schwachen Zusammenhangskomponenten: Wie in den Kapiteln 3.2 und 3.3 gesehen bieten schwach zusammenhängende Graphen zusätzliche Struktur, die von Algorithmen ausgenutzt werden kann.

Der Property-Tester für 3-Stern-Freiheit beispielsweise ließe sich zwar für Graphen mit nur wenigen schwachen Zusammenhangskomponenten adaptieren, aber bei mindestens $\Theta(\epsilon n)$ schwachen Zusammenhangskomponenten kann die Balance des Graphen sehr groß werden und das eigentlich zu messenden statistische Merkmal überlagern – jeder Property-Tester benötigt dann, wie gesehen, $\Omega(n^{2/3})$ Anfragen.

Die Methoden zum Testen von 3-Stern-Freiheit können übrigens nicht einfach auf das Testen von k-Stern-Freiheit übertragen werden: k-Stern-freie Graphen können viele $k-1$-Sterne enthalten, und das Problem dabei ist, dass die Argumente bezüglich der Balance dann nicht mehr anwendbar sind. Genauer gesagt kann nicht mehr davon ausgegangen werden, dass die Balance von k-Stern-freien Graphen ungefähr bekannt ist: Ein Graph mit $\Omega(n)$ $k-1$-Sternen kann eine Balance von $\Omega(kn)$ haben, aber auch eine Balance von 0. Gegenüber diesem Unterschied sind die messbaren statistischen Unterschiede bezüglich ein- und ausgehender 2-Sternknoten (oder auch $k-1$-Sternknoten) gering.

Schwacher Zusammenhang selbst kann vermutlich nicht mit einer Anfragekomplexität von $o(n^{1-\epsilon/4})$ getestet werden, eine Idee für zwei schwer unterscheidbare Graphklassen ist zum Beispiel die Folgende: Die Klasse schwach zusammenhängender Graphen enthält alle Graphen, die aus einem großen Kreis bestehen, dessen Kanten abwechselnd in unterschiedliche Richtungen gerichtet sind; die Klasse der davon ϵ-entfernten Graphen besteht aus ϵn kleinen solchen Kreisen mit jeweils $\frac{1}{\epsilon}$ Knoten. Da die Graphen lokal gleich aussehen – das größte lokal explorierbare Areal umfasst die drei Knoten eines ausgehenden 2-Sterns – und auch für eine Kollisionsstatistik der eingehenden Kanten einzelner Knoten gleich sind, ist die einzige messbare Statistik die Länge von schwach zusammenhängenden Graphstücken, die durch „Kollisionen“ exploriert wurden, zum Beispiel weil zwei ausgehende 2-Sternknoten gesampelt wurden, die sich einen adjazenten Knoten teilen. Dies ergäbe ein exploriertes Stück einer Länge von fünf Knoten, und bei $k-1$ gesampelten solcherart aufeinanderfolgenden ausgehenden 2-Sternknoten könnte man $2k+1$ Knoten erforschen. Ein statistisch signifikanter Unterschied zwischen den beiden Graphklassen würde sich erst ergeben, wenn hinreichend viele Knoten gesampelt würden, damit $\frac{1}{4\epsilon}$-Kollisionen entstehen: Erst dann wäre die Chance, in einem der kleinen Kreise eine solche Kollision herbeizuführen, signifikant größer als in einem großen Kreis, denn für zwei Knoten der Entfernung $\frac{1}{2\epsilon}$ gäbe es in einem Kreis der Größe $\frac{1}{\epsilon}$ dann zwei Möglichkeiten für die Erforschung einer Kette von $\frac{1}{2\epsilon}$ dazwischen liegenden aufeinanderfolgenden 2-Stern-Knoten; nämlich sozusagen in beiden Richtungen um den Kreis herum. Dass eine so große Anzahl von aufeinanderfolgenden ausgehenden 2-Sternknoten entdeckt wird, geschieht mit konstant hoher Wahrscheinlichkeit erst bei $\Omega(n^{1-\epsilon/4})$ Anfragen.

Untere Schranken. Eine wesentliche offene Frage für die Klassifizierung der testbaren Eigenschaften im Modell gradbeschränkter gerichteter Graphen ohne Sichtbarkeit eingehender Kanten ist die nach verbesserten unteren Schranken; interessant wäre hier, für weitere Probleme nachzuweisen, dass eine Schranke von $\Omega(n^{1-1/k})$ für die Anfragekomplexität nicht durchbrochen werden kann, wenn k Elemente zusammengehörige Elemente mangels gegenseitiger Sichtbarkeit einzeln gesampelt werden müssen.

Bei starkem Zusammenhang besteht das Problem beim Konstruieren einer stärkeren als der von Bender und Ron [14] gegebenen unteren Schranke von $\Omega(n^{1/2})$ beispielsweise darin, dass ausgeschlossen werden muss, dass Problemstruktur genutzt werden kann. Aufgrund der Ergebnisse von Rashkodnikova et al. ist zwar auszuschließen, dass eine 2-Fach-Kollisionsstatistik alleine zum Testen dieser Eigenschaft reicht – Graphklassen, deren Graphen für eine solche Statistik zwei proportionale Momente haben, lassen sich leicht angeben –; das Problem ist, die Klassen so zu wählen, dass das Messen anderer Parameter der Graphen ausgeschlossen werden kann.

In der in Kapitel 3 gezeigten unteren Schranke für das Testen von 3-Stern-Freiheit ist diese Idee mittelbar enthalten: Die in der Reduktion des Zwischenproblems 3-Wert-Freiheit auf 3-Sternfreiheit erzeugten Graphen bestehen aus einzelnen k-Sternen, die nicht miteinander verknüpft sind; dadurch ist beispielsweise das Explorieren längerer Kantenzüge ausgeschlossen, und im Prinzip reduzieren sich die möglichen Aktionen eines Algorithmus auf das Zählen der eingehenden Kanten der Zentralknoten. Solche Gra-

phen anzugeben ist allerdings nicht für jede Grapheigenschaft möglich, insbesondere dann nicht, wenn schwacher Zusammenhang der Eingabegraphen vorausgesetzt wird.

Testen von euklidischen Spannern. Eine wesentliche offene Frage hier ist, ob es einen Property-Testing-Algorithmus für euklidische Spanner gibt, der eine Anfragekomplexität von $\mathcal{O}(n^{1/3})$ erreicht. Der in Kapitel 5 vorgestellte Algorithmus hat eine Anfragekomplexität von $\mathcal{O}(n^{1/2})$, und dies ist im Lichte der Ergebnisse der anderen Kapitel die Laufzeit, die nötig ist, um mit hoher Wahrscheinlichkeit eine Zweifachkollision unter benachbarten Punkten zu erhalten. Die geometrischen Eigenschaften des Problems werden bei diesem Algorithmus im Prinzip nur in der Analyse genutzt, nämlich um zu zeigen, dass es in einem ϵ-entfernten Graphen linear viele benachbarte Punktpaare gibt, die Zeugen für die Verletzung der Spannereigenschaft sind. Der Algorithmus selbst verwendet geometrische Informationen nur als Kantengewichte für das Ausführen von Dijkstras Algorithmus und ignoriert völlig die Lagebeziehungen zwischen den Punkten, die sich aus ihren Positionen im euklidischen Raum ergeben.

Ein Algorithmus, der eine Anfragekomplexität von $o(n^{1/2})$ erreicht, müsste diese Informationen tatsächlich algorithmisch nutzen, zum Beispiel dadurch, dass das Verhalten der lokalen Explorierungen von den Lagebeziehungen zwischen den gesampelten Punkten abhängig gemacht wird. Diese Idee lässt sich leicht anhand der Graphen der Klasse $\mathcal{C}$ veranschaulichen, die in Kapitel 5.3 definiert worden ist; diese Klasse enthält Graphen, die aus zwei übereinanderliegenden, nicht miteinander verbundenen Punktlinien bestehen.

Ein Algorithmus, der die ϵ-Entferntheit dieser Graphen mit hoher Wahrscheinlichkeit nachweisen kann und eine Anfragekomplexität von $\mathcal{O}(n^{1/3})$ hat, arbeitet wir folgt[1]: Zunächst werden $\mathcal{O}(n^{1/3})$ Punkte zufällig und gleichverteilt gezogen. Es lässt sich zeigen, dass dabei mit hoher Wahrscheinlichkeit eine konstante Anzahl von Punktpaaren gesampelt wird, deren Abstand jeweils höchstens $\mathcal{O}(n^{1/3})$ ist, und dass sich unter diesen Punktpaaren auch Paare befinden, die jeweils einen Punkt aus jeder der Punktlinien erhalten. Der Algorithmus muss jetzt nur für alle diese Punktpaare prüfen, ob sie aus unterschiedlichen Linien stammen; das ist durch Explorierung jeweils von einem der Knoten in die Richtung des anderen in $\mathcal{O}(n^{1/3})$ Anfragen möglich. Insgesamt ist die Anfragekomplexität hierfür $\mathcal{O}(n^{1/3})$.

Die Schwierigkeit dieser Idee beim Übertragen auf allgemeinere Graphklassen liegt darin, dass es dichte und weniger dichte Bereiche in Graphen geben kann: Die euklidische Distanz zwischen zwei gesampelten Punkten p und q sagt dann wenig über die Anzahl der Punkte aus, die „zwischen“ ihnen liegen – die also auf Pfaden von p nach q liegen können, die die Spannereigenschaft erfüllen. Selbst wenn eine gleichmäßige Verteilung der Punkte angenommen wird, ist im Zweimensionalen die Kandidatenmenge aller Punkte, die auf einem geometrisch hinreichend kurzen Pfad zwischen p und q liegen können, für $\delta > 0$ $\Theta(n^{2/3})$, wenn die maximale Anzahl der Knoten auf dem Pfad auf $\Theta(n^{1/3})$ beschränkt ist.

[1] Ben-Zwi et al. [16] geben einen ähnlichen Algorithmus an, der die in ihrer unteren Schranke für euklidische minimale Spannbäume verwendeten Graphklassen in $\mathcal{O}(n^{1/3})$ Anfragen unterscheiden kann.

Andererseits ist es für das Zeigen einer unteren Schranke von $\Omega(n^{1/2})$ nötig zu zeigen, dass die Lagebeziehungen zwischen den Punkten tatsächlich nicht algorithmisch genutzt werden können. Aufgrund der obigen Überlegungen liegt nahe, dass für eine solche Schranke Graphklassen im Zwei- oder Höherdimensionalen untersucht werden müssen.

Eine weitere interessante Frage ist, ob der in Kapitel 5 angegebene Algorithmus auch ohne die Einschränkungen zur Lage der Punkte funktioniert – also auch für Mengen von Punkten, die nicht auf einem Gitter begrenzter Breite liegen; hierfür müsste für die Analyse eine völlig andere Vorgehensweise bei der Diskretisierung des Problems gewählt werden.

Literaturverzeichnis

[1] N. Alon. Testing subgraphs in large graphs. *Random Struct. Algorithms* 21(3-4), S. 359-370, 2002.

[2] N. Alon, S. Dar, M. Parnas, D. Ron. Testing of Clustering. *SIAM Journal on Discrete Mathematics* 16(3), S. 393-417, 2003.

[3] N. Alon, P. Seymour, R. Thomas. A Separator Theorem for Nonplanar Graphs. *Journal of the American Mathematical Society* 3(4), S. 801-808, 1990.

[4] H.-K. Ahn, M. Farshi, C. Knauer, M. Smid, Y. Wang. Dilation-Optimal Edge Deletion in Polygonal Cycles. *Algorithms and Computation*, S. 88-99, 2007.

[5] P.K. Agarwal, R. Klein, C. Knauer, S. Langerman, P. Morin, M Sharir, M. Soss. Computing the Detour and Spanning Ratio of Paths, Trees, and Cycles in 2D and 3D. *Discr. & Computational Geometry* 39(1-3), S. 17-37, 2007.

[6] N. Alon, E. Fischer, I. Newman, A. Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. *SIAM Journal on Computing* 39(1), S. 143-167, 2009.

[7] N. Alon, R. Rubinfeld, S. Vardi, N. Xie. Space-efficient local computation algorithms *In: Proceedings of the 23rd Annnual ACM-SIAM Symposium on Discr. Algorithms (SODA)*, S. 1132-1139, 2012

[8] N. Alon, A. Shapira. Testing subgraphs in directed graphs. *In: Proceedings of the 35th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 700-709, 2003.

[9] S. Arya, G. Das, M. Mount, J.S. Salowe, M. Smid. Euclidean spanners: short, thin, and lanky. *In: Proceedings of the 27th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 489-498, 1995.

[10] Z. Bar-Yossef, R. Kumar, D. Sivakumar. Sampling algorithms: Lower bounds and applications. *In: Proceedings of the 33rd Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 266-275, 2001.

[11] T. Batu, S. Dasgupta, R. Kumar, R. Rubinfeld. The complexity of approximating entropy. *In: Proceedings of the 34th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 678-687, 2002.

[12] T. Batu, L. Fortnow, E. Fischer, R. Kumar, R. Rubinfeld, P. White. Testing Random Variables for Independence and Identity. *In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, S. 442-451, 2001.

[13] T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, P. White. Testing that distributions are close. *In: Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, S. 259-269, 2000.

[14] M.A. Bender, D. Ron. Testing properties of directed graphs: acyclicity and connectivity. *Random Structures & Algorithms* 20(2), S. 184-205, 2002.

[15] I. Benjamini, O. Schramm, A. Shapira. Every minor-closed property of sparse graphs is testable. *In: Proceedings of the 40th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 393-402, 2008.

[16] O. Ben-Zwi, O. Lachish, I. Newman. Lower bounds for testing Euclidean Minimum Spanning Trees. *Information Processing Letters* 102(6), S. 219-225, 2007.

[17] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. Computational Geometry – Algorithms and Applications. *Springer*, 1997.

[18] E. Blais. Testing juntas nearly optimally. *In: Proceedings of the 41st Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 151-158, 2009.

[19] M. Blum, M. Luby, R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *In: Proceedings of the 22nd Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 73-83, 1990.

[20] I. N. Bronstein, K. A. Semendjajew. Taschenbuch der Mathematik. 25. Auflage, B. G. Teubner Verlagsgesellschaft, 1991.

[21] P.B. Callahan, S.R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. *In: Proceedings of the 4th Annnual ACM-SIAM Symposium on Discr. Algorithms (SODA)*, S. 291-300, 1993.

[22] B. Chazelle, D. Liu, A. Magen. Sublinear Geometric Algorithms. *SIAM Journal on Computing* 35(3), S. 627-646, 2006

[23] B. Chazelle, R. Rubinfeld, L. Trevisan. Approximating the Minimum Spanning Tree Weight in Sublinear Time *SIAM Journal on Computing* 34(6), S. 1370-1379, 2005

[24] K. L. Clarkson. Approximating algorithms for shortest path motion planning. *In: Proceedings of the 19th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 56-65, 1987.

[25] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, C. Sohler. Approximating the Weight of the Minimum Spanning Tree in Sublinear Time. *SIAM Journal on Comp.* 35(1), S. 91-109, 2005.

[26] A. Czumaj, A. Shapira, C. Sohler. Testing hereditary properties of non-expanding bounded-degree graphs. *SIAM Journal on Computing* 38(6), S. 2499-2510, 2009.

[27] A. Czumaj, C. Sohler. Estimating the Weight of Metric Minimum Spanning Trees in Sublinear Time. *SIAM Journal on Computing* 39(3), S. 904-922, 2009.

[28] A. Czumaj, C. Sohler. Property Testing with Geometric Queries. *In: Proceedings of the 9th Annnual European Symposium on Algorithms (ESA)*, S. 266-277, 2001.

[29] A. Czumaj, C. Sohler. Testing Euclidean minimum spanning trees in the plane. *ACM Transactions on Alg.* 4(3), 2008.

[30] A. Czumaj, C. Sohler. Testing expansion in bounded-degree graphs. *Combinatorics, Probability and Computing* 19(5-6), S. 693-709, 2010.

[31] A. Czumaj, C. Sohler, M. Ziegler. Property Testing in Computational Geometry. *In: Proceedings of the 8th Annnual European Symposium on Algorithms (ESA)*, S. 155-166, 2000.

[32] J. Dean, S. Ghemawat und Google Inc. MapReduce: Simplified Data Processing on Large Clusters. *In OSDI 04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*, S. 137-150, 2004

[33] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* 1, S. 269-271, 1959.

[34] D. Eppstein, K. A. Wortman. Minimum dilation stars. *Computational Geometry: Theory and Applications* 37(1), S. 27-37, 2007.

[35] F. Ergun, S. Kannan, R. Kumar, R. Rubinfeld, M. Viswanathan. Spot-Checkers. *Journal of Computer and System Sciences* 60(3), S. 717-751, 2000.

[36] M. Farshi, P. Giannopoulos, J. Gudmundsson. Finding the best shortcut in a geometric network. *In: Proceedings of the 21th Annnual ACM Symposium on Computational Geometry*, S. 327-335, 2005.

[37] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, A. Samorodnitsky. Monotonicity testing over general poset domains. *In: Proceedings of the 34th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 474-483, 2002.

[38] E. Fischer, I. Newman. Testing versus estimation of graph properties. *In: Proceedings of the 37th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 138-146, 2005.

[39] O. Goldreich. Property Testing: Current Research and Surveys. Springer 2010.

[40] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, A. Samorodnitsky. Testing Monotonicity. *Combinatorica* 20(3), S. 301-337, 2000.

[41] O. Goldreich, S. Goldwasser, D. Ron. Property Testing and its Connection to Learning and Approximation. *J. of the ACM* 45(4), S. 653-750, 1998.

[42] O. Goldreich, D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica Vol. 19*, pp. 289-298, 1999.

[43] O. Goldreich, D. Ron. Property Testing in Bounded Degree Graphs. *In: Proceedings of the 29th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 406-415, 1997.

[44] O. Goldreich, D. Ron. On Testing Expansion in Bounded-Degree Graphs. *Studies in Complexity and Cryptography*, S. 68-75, 2011.

[45] J. E. Hopcroft, R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Comp.* 2(4), S. 225-231, 1973.

[46] A. Hassidim, J.A. Kelner, H.N. Nguyen, K. Onak. Local graph partitions for approximation and testing. *In: Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, S. 22-31, 2009.

[47] S. Har-Peled, S. Mazumdar. On Coresets for k-Means and k-Median Clustering. *In: Proceedings of the 36th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 291-300, 2004.

[48] F. Hellweg, M. Schmidt, C. Sohler. Testing Euclidean Spanners. *In: Proceedings of the 18th Annnual European Symposium on Algorithms (ESA)*, S. 60-71, 2010.

[49] F. Hellweg, C. Sohler. Property Testing in Sparse Directed Graphs: Strong Connectivity and Subgraph Freeness. *In: Proceedings of the 20th Annnual European Symposium on Algorithms (ESA)*, S. 599-610, 2012.

[50] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association* 58(301), S. 13-30, 1963.

[51] S. Kale, C. Seshadhri. An Expansion Tester for Bounded Degree Graphs. *SIAM Journal on Computing* 40(3), S. 709-720, 2011.

[52] T. Kaufman, M. Sudan. Algebraic property testing: the role of invariance. *In: Proceedings of the 40th Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 403-412, 2008.

[53] I. Kleiner, D. Keren, I. Newman, O. Ben-Zwi. Applying Property Testing to an Image Partitioning Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 33, S. 256-265, 2011.

[54] M. Mitzenmacher, E. Upfal. Probability and Computing. Cambridge University Press, 2005.

[55] S. Muthukrishnan. Data Streams: Algorithms and Applications. Now Publishers, 2005.

[56] G. Narasimhan, M. Smid. Approximating the Stretch Factor of Euclidean Graphs. *SIAM Journal on Computing*, S. 978-989, 2000.

[57] H. N. Nguyen, K. Onak. Constant-Time Approximation Algorithms via Local Improvements. *In: Proceedings of the 49th IEEE Symposium on Foundations of Computer Science (FOCS)*, S. 327-336, 2008

[58] A. Nachmias, A. Shapira. Testing the expansion of a graph. *Information and Computation* 208(4), S. 309-314, 2010.

[59] I. Newman, C. Sohler. 2011. Every property of hyperfinite graphs is testable. *In: Proceedings of the 43rd Annnual ACM Symposium on the Theory of Computing (STOC)* S. 675-684, 2011.

[60] K. Onak, D. Ron, M. Rosen, R. Rubinfeld. A Near-Optimal Sublinear-Time Algorithm for Approximating the Minimum Vertex Cover Size. *In: Proceedings of the 23rd Annnual ACM-SIAM Symposium on Discr. Algorithms (SODA)*, S. 1123-1131, 2012.

[61] Y. Orenstein, D. Ron. Testing Eulerianity and connectivity in directed sparse graphs. *Theoretical Computer Science* 412(45), S. 6390-6408, 2011.

[62] M. Parnas, D. Ron. Testing the Diameter of Graphs. *Random Structures & Algorithms* 20(2), S. 165-183, 2002.

[63] L. Rademacher, S. Vempala. Testing Geometric Convexity. In: Proceedings of the 24th Foundations of Software Technology and Theoretical Computer Science (FSTTCS), S. 469-480, 2004.

[64] S. Raskhodnikova, D. Ron, A. Shpilka, A. Smith. Strong Lower Bounds for Approximating the Distribution Support Size and the Distinct Elements Problem. *SIAM Journal on Computing* 39(3), S. 813-842, 2009.

[65] S. Raskhodnikova, A. Smith. A Note on Adaptivity in Testing Properties of Bounded Degree Graphs. *Electronic Colloquium on Computational Complexity (ECCC)* 13(89), 2006.

[66] D. Ron. Algorithmic and Analysis Techniques in Property Testing. *Foundations and Trends in Theoretical Computer Science* 5(2), S. 73-205, 2010.

[67] D. Ron, G. Tsur. Testing Properties of Sparse Images. *In: Proceedings of the 51th IEEE Symposium on Foundations of Computer Science (FOCS)*, S. 468-477, 2010.

[68] R. Rubinfeld, M. Sudan. Robust Characterizations of Polynomials with Applications to Program Testing. *SIAM Journal on Computing* 25(2), S. 252-271, 1996.

[69] R. Rubinfeld, G. Tamir, S. Vardi, N. Xie. Fast Local Computation Algorithms. *In: Proceedings of the 2nd Symposium on Innovations in Computer Science*, S. 223-238, 2011

[70] L. G. Valiant. A Theory of the Learnable. *Communications of the ACM* 27(11), S. 1134-1142, 1984.

[71] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. *In: Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, S. 222-227, 1977.

[72] Y. Yoshida, H. Ito. Testing k-edge-connectivity of digraphs. *Journal of System Science and Complexity* 23(1), S. 91-101, 2010.

[73] Y. Yoshida, M. Yamamoto, H. Ito. An improved constant-time approximation algorithm for maximum matchings. *In: Proceedings of the 41st Annnual ACM Symposium on the Theory of Computing (STOC)*, S. 225-234, 2009.