Diplomarbeit

Das Problem Multipoint Relay – Komplexität und Algorithmen

Alexander Fanghänel

Betreuer: PD Dr. Rolf Niedermeier
Dipl.-Inform. Michael Dom
Dipl.-Inform. Falk Hüffner

eingereicht am: 26.07.2006

Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik
Theoretische Informatik I: Komplexitätstheorie

Zusammenfassung. Für die Pfadsuche von Routing-Algorithmen sind in drahtlosen, mobilen Netzwerken Broadcasts erforderlich. Um die Redundanz und die Netzbelastung durch Broadcasts im Vergleich zu naiven Algorithmen zu verringern, kann ein Netzknoten eine Teilmenge seiner Nachbarn berechnen, sogenannte *Multipoint Relays*, die dann alle Knoten der Distanz zwei zum Sender benachrichtigen.

Im Allgemeinen stellt die Berechnung einer kleinstmöglichen Multipoint-Relay-Menge ein NP-hartes Problem dar, was bedeutet, dass optimale Lösungen vermutlich nicht effizient berechenbar sind. Da jedoch effizient berechenbare optimale Lösungen von unzweifelhaftem praktischen Interesse sind, setzen wir uns in dieser Arbeit mit der Multipoint-Relay-Berechnung auf speziellen Graphklassen, die für die Modellierung mobiler Netzwerke genutzt werden, auseinander. Genauer stellen wir für planare Graphen zwei Polynomialzeitalgorithmen vor, die eine optimale Lösung liefern. Der erste Algorithmus berechnet eine gesuchte Lösung durch Datenreduktion in $O(n^3)$ Schritten. Der zweite vorgestellte Algorithmus stellt eine dynamische Programmierung auf einer sogenannten *Baumzerlegung* dar, die in $O(2^\omega n)$ Schritten durchführbar ist. Dabei gibt ω die *Baumweite* eines Graphen – seine Ähnlichkeit zu einem Baum – an und wir können zeigen, dass bei diesem Problem auftretende planare Graphen eine Baumweite von $\omega \leq 5$ vorweisen. Im Anschluss untersuchen wir, wie sich das Problem verhält, wenn ein Knoten einen größeren lokalen Auschnitt des Netzes kennt, statt lediglich seine Nachbarn und deren Nachbarn. Wir zeigen, dass die beiden für planare Graphen präsentierten Algorithmen auch hier eine effiziente Berechnung einer optimalen Lösung zulassen und sich die Anzahl der sendenden Knoten im gesamten Netzwerk reduziert. Weiterhin betrachten wir die Klasse der Unit-Disk-Graphen und geben bei einer leichten Problemmodifikation einen Approximiationsalgorithmus mit einer additiven Approximationskonstante an, der eine Laufzeit von $O(n)$ besitzt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ob zu Hause oder in öffentlichen Bereichen wie Flughäfen und Universitäten, schon längst sind drahtlose Netzwerke in unser tägliches Leben integriert. Doch die momentanen Anwendungsgebiete scheinen erst die Spitze des Eisbergs darzustellen. Auch in Anwendungen, die sich erst in der Entwicklung befinden, spielen mobile Netzwerke eine tragende Rolle. So werden zur Zeit Ideen verfolgt, den Straßenverkehr mittels sogenannter Car-to-Car-Kommunikation (C2CC) sicherer zu gestalten [XMKS04, OKD06]. Um so wichtiger ist es, sich mit der Funktionsweise dieser Netzwerke vertraut zu machen und effiziente Algorithmen dafür zu entwickeln. Diese Ad-Hoc-Netzwerke bestehen zumeist aus mobilen Stationen mit geringem Senderadius. Jede dieser Stationen ist in der Lage, empfangene Nachrichten in einem kleinen Umkreis wieder zu senden und somit an weitere Teilnehmer zu übermitteln. So ist es möglich, dass eine Kommunikation zwischen zwei Teilnehmern möglich wird, obwohl die Distanz zwischen ihnen ihre Sendeleistung um ein Vielfaches übersteigt.

Mit dem verstärkten Auftreten dieser Netze wird es auch nötig, neue Strategien für die Nachrichtenübermittlung zu studieren, da es einige wesentliche Aspekte gibt, die mobile Ad-Hoc-Netze von ihren (verdrahteten) Vorgängern unterscheiden. Während fest installierte Netze eine sich nur langsam ändernde oder sogar statische Topologie aufweisen, die jedem Teilnehmer nur einmal bekannt gemacht werden muss, unterliegt in einem mobilen Netzwerk die Netzstruktur ständigen Änderungen. Da solche mobilen Netze im Allgemeinen keine zentrale Netzwerkverwaltung aufweisen, würde eine ständige Kommunikation zur Aktualisierung der Topologieinformationen das Netzwerk stark be- oder sogar überlasten. Daher benutzt man für die Kommunikation zwischen zwei Netzteilnehmern eine reaktive Strategie.

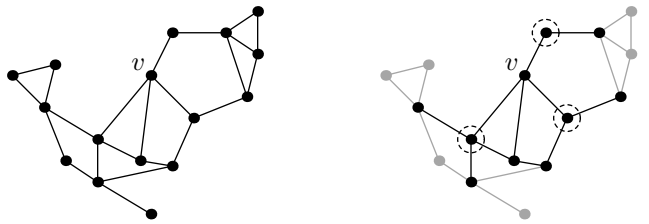
Sobald ein Verbindungsaufbau gewünscht wird, initiiert der Sender eine Suche nach dem Empfänger. Dazu schickt er an alle Stationen in seiner Sendereichweite die Nachricht, dass er auf der Suche nach dem Empfänger ist, diese wiederum schicken diese Information an alle in ihrer Reichweite liegenden Teilnehmer bis die Nachricht schließlich den Empfänger erreicht. Dieser sendet auf dem nun gefundenen Weg eine Bestätigungsmeldung an den ursprünglichen Sender und ein gezielter Verbindungsaufbau ist nun durchführbar.

Dieses „Senden an alle“, auch als broadcasten bezeichnet, führt bei einer naiven Durchführung trotz kurzer Kontrollnachrichten zu einer stetigen Belastung des Netzes. Um diese Belastung möglichst gering zu halten, wurden verschiedene Broadcasting-Strategien entwickelt. Zum Einen existiert ein probabilistischer Ansatz, bei dem eine Station eine erhaltene Broadcastnachricht mit einer festgelegten Wahrscheinlichkeit weitersendet [SCS02, ZA05]. Diese Strategie hat allerdings den entscheidenden Nachteil, dass nicht sichergestellt werden kann, dass tatsächlich alle Teilnehmer die Nachricht erhalten. Zum Anderen existiert ein Modell basierend auf der Kenntnis der Nachbarschaften, bei denen eine broadcastende Station festlegt, welche der von ihr erreichten Netzteilnehmer die Nachricht weitersenden sollen. Die so ausgewählten Teilnehmer bezeichnet man auch als Multipoint Relays. Wir wollen uns damit beschäftigen, wie eine broadcastende Station eine möglichst kleine Anzahl solcher Multipoint-Relay-Knoten bestimmen kann.

Mit dieser Berechnung, welche wir als das Problem Multipoint Relay bezeichnen, beschäftigt sich die vorliegende Arbeit. Wir modellieren die Netzwerke als Graphen, bei denen jeder Teilnehmer als ein Knoten und eine direkte Verbindung zwischen zwei Teilnehmern als Kante des Graphen dargestellt wird. Ausgehend von dieser Modellierung formulieren wir folgende Fragen: Welche direkten Nachbarn des Senders müssen ihrerseits mindestens senden, um alle Knoten, die den Abstand zwei zum Sender haben, zu benachrichtigen? Welche direkten Nachbarn des Senders stellen also Multipoint Relays dar?

In Abb. 1 (i) ist ein Netzwerk dargestellt, in welchem der Knoten v einen Broadcast initiieren möchte. Die in Abb. 1 (ii) dunkel gefärbten Knoten und Kanten kennt v nach zweimaligem Austausch der Nachbarschaftsinformationen. Mit diesen Informationen berechnet er die drei markierten Knoten als Multipoint Relays, sobald diese Knoten senden, werden alle Knoten, die v kennt, erreicht.

Wir werden in dieser Arbeit untersuchen, wie ein Knoten eine optimale Multipoint-Relay-Menge bestimmen kann. Doch wie läuft der Vorgang des Broadcastens im gesamten Netz ab, also wann muss ein Knoten eine



(i) (ii)

Abbildung 1.1: Im Bild ist ein Netzwerk (i) und ein broadcastender Knoten v mit seinem lokalen Blick auf das Netzwerk und den berechneten Multipoint Relays (ii) dargestellt.

Multipoint-Relay-Menge berechnen? Ein Knoten initiiert einen Broadcast, zum Beispiel weil er wie bereits beschrieben einen bestimmten Kommunikationspartner sucht. Dann berechnet er seine Multipoint Relays, sendet die Nachricht an alle Knoten, die er erreicht und fordert die Multipoint Relays auf, ihrerseits wieder den Broadcast zu veranlassen. Es kann passieren, dass ein Knoten im Laufe dieses Vorgangs dieselbe Nachricht auf unterschiedlichen Wegen mehrmals zugestellt bekommt. Tritt dieser Fall ein, so sendet der Knoten die Nachricht nur einmal weiter, genau dann wenn er beim ersten Empfangen der Nachricht die Aufforderung dazu bekommen hat. Ist er beim ersten Mal von einem Graphknoten v nicht als Multipoint-Relay-Knoten ermittelt worden, so hat v ja bereits sichergestellt, dass alle Nachbarn dieses Knotens erreicht werden. Ein späteres Senden auf Grund einer Aufforderung wäre also redundant.

Mit diesem einfachen Kommunikationsschema müssen wir nur noch sicherstellen, dass die berechneten Multipoint-Relay-Mengen möglichst klein sind, um die netzspezifische Kommunikationskapazität so wenig wie möglich zu belasten und somit für den eigentlichen Datenaustausch nutzbar zu belassen. Wir werden zeigen, dass die Berechnung einer kleinstmöglichen Menge von Multipoint Relays ein NP-hartes Problem darstellt. Das hat zur Folge, dass es nach derzeitigem Stand der Forschung[1] wohl nicht möglich ist, einen Algorithmus anzugeben, der in polynomialer Zeit in Abhängigkeit der Eingabegröße eine optimale Lösung liefert. Lediglich eine exponenzielle Laufzeit wäre denkbar, was jedoch sehr schnell zu Rechenzeiten führt, die nicht mehr vertretbar sind.

[1] sofern $P \neq NP$

Da wir nicht erwarten können, effiziente Algorithmen für NP-harte Probleme zu finden, müssen andere Lösungsansätze gesucht werden. Wir werden deshalb in dieser Arbeit Graphklassen für dieses Problem zugrunde legen, die zur Modellierung mobiler Netze verwendet werden und untersuchen, ob und wie sich für Graphen dieser Klassen optimale Lösungen berechnen lassen.

1.1 Literaturüberblick

Ein Broadcasten mittels Multipoint Relays wurde erstmalig vorgeschlagen in [Com96]. Quayyum et al. [QVL02] untersuchen das Problem genauer, beweisen die NP-Vollständigkeit von Multipoint Relay auf allgemeinen Graphen und stellen eine Heuristik zur Problemlösung vor.

Calinescu et al. [CMWZ04] bezeichnen dieses Problem auch als Minimum Forwarding Set und untersuchen es speziell für Unit-Disk-Graphen. Sie führen an, dass die Komplexität dieses Problems auf Unit-Disk-Graphen noch nicht bekannt ist und stellen einen 3-Approximations-Algorithmus mit einer Laufzeit von $O(n \log^2 n)$ vor.

Wu, Lou und Dai [WLD06] betrachten Multipoint Relay zeigen, wie man durch Verwenden von Multipoint-Relay-Algorithmen für mobile Ad-Hoc-Netzwerke ein sogenanntes Backbone des Netzwerkes, eine zusammenhängende Menge sendender Knoten, berechnen kann. Dazu zeigen sie zwei Lösungsansätze für Multipoint Relay. Erstens einen senderabhängigen, bei dem die konkrete Ausprägung einer globalen Menge von Multipoint Relays abhängig von broadcastenden Netzwerkknoten ist, also die eingangs beschriebene Idee. Und als zweites präsentieren sie einen senderunabhängigen Algorithmus, bei welchem unabhängig vom sendenden Knoten anhand von Identifikationsnummern immer dieselben Stationen als Multipoint Relays ausgewählt werden. Sie beschreiben dann, wie man mit Hilfe der beiden Ansätze in einem mobilen Netzwerk ein Backbone des Netzwerkes berechnen kann.

1.2 Gliederung und Ergebnisse dieser Arbeit

Im 2. Kapitel führen wir zunächst die verwendeten Begriffe aus der Graphentheorie ein. Im Anschluss daran geben wir eine exakte Problembeschreibung an und untersuchen die Komplexität von Multipoint Relay. Die folgenden beiden Kapitel beschäftigen sich mit zwei Algorithmen, die eine optimale Menge von Multipoint Relays auf planaren Graphen in Polynomialzeit be-

rechnen. Zum Einen zeigen wir, wie die Berechnung über Reduktionsregeln gelingt. Zum Anderen stellen wir einen Algorithmus vor, der mit der Technik des dynamischen Programmierens auf einer Baumzerlegung arbeitet. Diese Ergebnisse liefern uns einen exakte Algorithmus mit einer Laufzeit von $O(n)$.

Bis zu diesem Punkt sind wir davon ausgegangen, dass ein Netzknoten seine Nachbarn und deren Nachbarn kennt. Wie sich das Problem verändert, wenn ein Knoten mehr lokale Informationen besitzt, untersuchen wir im Kapitel 5. In diesem Modell soll sichergestellt werden, dass alle Knoten benachrichtigt werden, die der Sender kennt. Wir stellen fest, dass sich optimale Lösungen für diese Problemvariation auf planaren Graphen noch immer effizient berechnen lassen und wir auf diese Weise die Anzahl der broadcastenden Knoten im gesamten Netzwerk weiter verringern können.

Disk-Graphen stellen ein geeignetes Modell für drahtlose Netzwerke dar. Dabei interpretieren wir einen Knoten als Netzteilnehmer und einen Kreis mit dem Knoten als Mittelpunkt als seinen Senderadius. Ein Knoten ist dann Nachbar all derjenigen Knoten, in deren Kreis er sich befindet. Eine Einschränkung dieser Graphklasse stellen die Unit-Disk-Graphen dar, bei denen der Radius aller dieser Kreise gleich groß ist. Im Kapitel 6 stellen wir einen Ansatz vor, wie sich auf Unit-Disk-Graphen mit einem geringfügig erweiterten Multipoint-Relay-Modell ein additiver Approximations-Algorithmus in polynomialer Zeit realisieren lässt. Dieses Ergebnis lässt vermuten, dass die Modifikation dieses Modells auf Unit-Disk-Graphen kein NP-hartes Problem darstellt.

Zusammenfassend wollen wir einen Überblick über die Ergebnisse unserer Arbeit geben:

- Wir stellen einen exakten Algorithmus mit einer Laufzeit von $O(n^3)$ zur Berechnung von Multipoint-Relay-Mengen auf planaren Graphen mittels Datenreduktion vor.

- Wir präsentieren einen Baumzerlegungsalgorithmus, der bei einer fester Baumweite ω eines Graphen in $O(2^\omega n)$ eine optimale Multipoint-Relay-Menge berechnet und zeigen, dass planare MPR-Instanzen eine Baumweite $\omega \leq 5$ besitzen.

- Wir zeigen, wie sich eine MPR-Berechnung durch Kenntnis der Distanz-d Nachbarn verändert und dass sich auf planaren Graphen die Menge der insgesamt sendenden Knoten in einem Netzwerk reduzieren lässt.

- Wir zeigen, dass sich auf Unit-Disk-Graphen bei Kenntnis der Nachbarschaften der Distanz-2-Knoten untereinander eine Multipoint-Relay-Menge berechnen lässt, die höchstens 18 Knoten mehr enthält als eine

optimale Multipoint-Relay-Menge. Dabei lassen wir zu, dass Distanz-2-Nachbarn erst im dritten Sendeschritt benachrichtigt werden.

Kapitel 2

Bezeichnungen und Grundlagen

Wir wollen zunächst die in dieser Arbeit verwendeten Bezeichnungen definieren und einige grundlegende Aussagen festhalten. Dazu beschäftigen wir uns in den folgenden drei Abschnitten mit Grundbegriffen der Graphentheorie, einigen ausgewählten Graphklassen und Baumzerlegungen von Graphen. Im Anschluss daran legen wir die Begriffe des zu untersuchenden Problems Multipoint Relay fest und führen erste allgemeine Komplexitätsbetrachtungen durch.

2.1 Grundlagen der Graphentheorie

Die in dieser Arbeit verwendeten Bezeichnungen orientieren sich weitestgehend am Standard der Graphentheorie [Die06]. So wird ein *Graph* G durch zwei disjunkte Mengen V und E spezifiziert, wir schreiben auch $G = (V, E)$. Die Menge V bezeichnen wir als *Knotenmenge*, ihre Elemente als *Knoten*. Für einen Knoten $v \in V$ schreiben wir auch $v \in G$. Die Menge $E = \{\{v_1, v_2\} \mid v_1, v_2 \in V\}$ bezeichnen wir als *Kantenmenge*, ihre Elemente entsprechend als *Kanten* des Graphen. Wir wollen in dieser Arbeit davon ausgehen, dass wir stets endliche Graphen vorliegen haben, also Graphen, deren Knoten- und Kantenmengen endlich sind. Eine visuelle Darstellung eines Graphen können wir erhalten, wenn wir seine Knoten als Punkte in der Ebene und seine Kanten entsprechend als knotenverbindende Linien zeichnen. Eine konkrete Darstellung eines Graphen in der euklidischen Ebene bezeichnen wir auch als *Einbettung*. Da diese Art der Darstellung für einen Graphen unzählige Möglichkeiten bietet ihn unterschiedlich zu zeichnen soll

darauf hingewiesen werden, dass die formale Definition eines Graphen unabhängig von seiner bildlichen Darstellung ist.

Zwei Knoten v_1 und v_2 aus G heißen *benachbart* oder *adjazent* in G, wenn $\{v_1, v_2\} \in E$ ist. Diese Knoten v_1 und v_2 bezeichnen wir dann auch als *Nachbarn* in G. Für einen Knoten $v_1 \in V$ bezeichne

$$N_G(v_1) := \{v_2 \in V \mid \{v_1, v_2\} \in E\}$$

die Menge seiner Nachbarknoten. Weiterhin sei

$$N_G^2(v_1) := \{v_3 \in V \mid \exists v_2 \in N_G(v_1) : v_3 \in N(v_2) \setminus N(v_1) \wedge v_3 \neq v_1\}$$

die Menge derjenigen Knoten des Abstandes zwei zum Knoten v_1. Analog dazu definieren wir die Menge der Nachbarn einer Knotenmenge $V' \subseteq V$ als

$$N_G(V') := \{v \in V \setminus V' \mid \exists v' \in V' : v \in N(v')\}.$$

Ist der Bezug zum Graphen G aus dem Kontext ersichtlich schreiben wir an Stelle von $N_G(v_1)$ bzw. $N_G^2(v_1)$ auch nur $N(v_1)$ bzw. $N^2(v_1)$. Der *Knotengrad* eines Knotens v wird mit $\deg_G(v)$ oder auch einfach mit $\deg(v)$ bezeichtet und gibt die Anzahl der Nachbarn von v in G an, also $\deg(v) := |N_G(v)|$. Der *Grad* eines Graphen sei $\deg(G) = \max_{v \in V}\{\deg(v)\}$. Wir sagen, eine Knotenmenge V_1 *dominiert* eine Knotenmenge V_2, wenn jeder Knoten $v \in V_2$ entweder in V_1 liegt oder zu V_1 benachbart ist.

Ein Graph $G' = (V', E')$ heißt *Teilgraph* von $G = (V, E)$, $G' \subseteq G$, wenn $V' \subseteq V$ und $E' = \{\{v_1, v_2\} \mid v_1, v_2 \in V' \wedge \{v_1, v_2\} \in E\}$ gilt. Wir nennen einen Teilgraphen $G' = (V', E')$ von $G = (V, E)$ *induziert*, wenn er alle Kanten $\{v_1, v_2\} \in E$ enthält, für die $v_1, v_2 \in V'$ gilt und bezeichnen diesen auch mit $G[V']$. Außerdem sei $G - V' := G[V \setminus V']$. Wir nennen einen Graphen G' einen *Minor* von G, wenn G' aus einem Teilgraph von G durch Knotenverschmelzungen entsteht. Knotenverschmelzung bedeuted dabei, dass zwei in G benachbarte Knoten v_1, v_2 in G' zu einem Knoten v zusammengefasst werden, wobei v in G' zu allen Knoten benachbart ist, die in G zu v_1 oder v_2 adjazent sind. In Abb. 2.1 findet sich je ein Beispiel für einen Teilgraphen, einen induzierten Teilgraphen und einen Minor.

Ein *Pfad* der Länge k ist ein Graph $P = (V, E)$ mit

$$V = \{v_0, v_1, \ldots, v_k\}, E = \{\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}\}$$

wobei alle Elemente aus V paarweise verschieden sind. Wir schreiben auch kürzer $P = (v_0, v_1, \ldots, v_k)$. Einen Graph $C = (V, E)$ mit

$$V = \{v_0, v_1, \ldots, v_k\}, E = \{\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, v_0\}\}$$

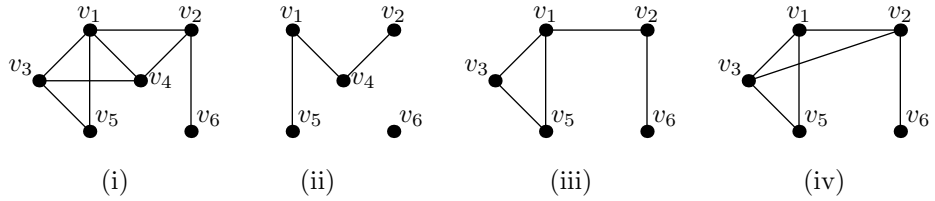


Abbildung 2.1: Ein Graph (i) mit einem Teilgraphen (ii), einem induzierten Teilgraphen (iii) und einem Minor (iv).

nennen wir einen *Kreis* der Länge $k+1$. Hierfür schreiben wir auch kürzer $C = (v_0, v_1, \ldots, v_k, v_0)$.

Einen Graphen G nennen wir *zusammenhängend*, wenn für je zwei beliebig gewählte Knoten v_1 und v_2 aus G ein Pfad zwischen v_1 und v_2 existiert.

Einen kreisfreien Graphen bezeichnen wir als *Wald*, einen zusammenhängenden kreisfreien Graphen als *Baum*.

2.2 Ausgewählte Graphklassen

Üblicherweise wird bei einem Netzwerk nicht nur die zugrunde liegende Struktur mit einem Graphen modelliert, man versucht auch den Graphen geeignet zu beeinflussen [BB06]. Konkreter bedeutet dies, dass man versucht, die Kantenanzahl in einem gegebenen Graphen zu verringern, da dies Routingalgorithmen im allgemeinen schneller und leichter implementierbar macht.

In diesem Abschnitt wollen wir die Graphklassen, mit denen wir uns im Laufe der Arbeit beschäftigen und die als netzwerkmodellierende Graphen verwendet werden, definieren und im Falle der planaren Graphen einige später verwendete Eigenschaften zeigen.

2.2.1 Planare Graphen

Wir nennen einen gegebenen Graphen G *planar*, wenn G eine Einbettung in der Ebene besitzt, in der sich keine Kanten kreuzen. Durch eine konkrete Einbettung in die Ebene wird diese in Gebiete aufgeteilt, die wir als *Flächen* bezeichnen. Die den Graph umgebende Fläche bezeichnen wir als *äußere Fläche*. Über den Zusammenhang zwischen Knoten, Kanten und Flächen in einem planaren Graphen lässt sich folgendes festhalten.

Satz 2.1 (Eulerscher Polyedersatz)**.** *Sei $G = (V, E)$ ein eingebetteter pla-*

narer Graph mit $|V|$ Knoten, $|E|$ Kanten und f_G Flächen. Dann gilt:

$$|V| - |E| + f_G = 2.$$

Ein Beweis dieses Satzes lässt sich leicht über Induktion führen, hier sei auf die Literatur verwiesen [Die06]. Dieser Satz liefert zugleich das Wissen, dass die Kantenanzahl in planaren Graphen linear abhängig von der Knotenzahl ist. Dieses Wissen ist an einigen Stellen für Laufzeitbetrachtungen unentbehrlich.

Eine kreuzungsfreie Einbettung eines planaren Graphen bezeichnen wir als *außerplanar* oder auch *1-außerplanar*, wenn alle Knoten des Graphen an der äußeren Fläche liegen. Entsprechend bezeichnen wir eine Einbettung eines planaren Graphen als *r-außerplanar*, wenn diejenige Einbettung, die durch Entfernung aller Knoten an der äußeren Fläche entsteht, $(r-1)$-außerplanar ist. Wir nennen einen planaren Graphen r-außerplanar, wenn er eine r-außerplanare Einbettung besitzt.

Zu einer gegebenen planaren Einbettung eines Graphen konstruieren wir den *dualen Graphen*, indem wir in jede Fläche der Einbettung einen Knoten platzieren, die Knoten benachbarter Flächen verbinden und die ursprünglichen Knoten löschen.

Lemma 2.2. *Die Konstruktion des dualen Graphen zu einer gegebenen Einbettung ist in $O(|V|)$ Schritten möglich.*

Dieses Lemma ist leicht einsichtig, wenn wir uns überlegen, dass wir lediglich für jede Kante des Originalgraphen prüfen müssen, welche zwei Flächen er voneinander trennt, wir also jede Kante genau einmal betrachten müssen.

Wir werden in einem späteren Kapitel Datenreduktionen auf gegebenen planaren Graphen durchführen. Für die grundlegende Funktionalität dieser Reduktionen benötigen wir noch folgendes Lemma.

Lemma 2.3. *Sei G ein planarer Graph. Dann ist jeder Teilgraph von G planar.*

Diese Erkenntnis bedeutet, dass wir in einem planaren Graphen Knoten und Kanten löschen können und dabei die Planarität erhalten bleibt, die Klasse der planaren Graphen also invariant gegenüber Kanten- und Knotenlöschungen ist.

Die Idee, ein gegebenes Netzwerk nicht einfach nur durch Graphen darzustellen, sondern zum Beispiel durch Anpassung der Sendeleistung der einzelnen Knoten den Graphen gezielt zu beeinflussen, wurde erstmalig von Takagi

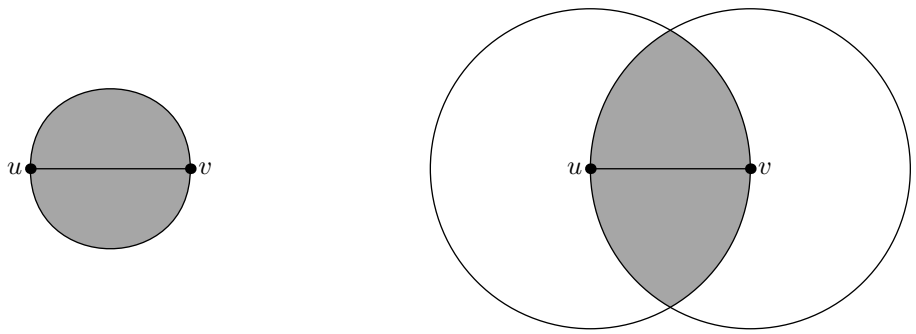


Abbildung 2.2: Die markierten Flächen zeigen die leeren Nachbarschaften einer Kante im Gabriel-Graphen (i) und im Relative-Neighbourhood-Graphen (ii).

und Kleinrock vorgestellt [TK84]. Durch derartige gezielte Graphänderungen wird es möglich, Netzwerken eine Graphstruktur zugrunde zu legen, die effiziente Algorithmen ermöglicht. Um diese Form der Topologiekontrolle zu nutzen, sollte für ein beliebiges Netzwerk der Topologiegraph lokal berechenbar sein, also muss jeder Knoten in der Lage sein, die Kanten des Topologiegraphen in seiner unmittelbaren Umgebung zu berechnen.

Zwei dieser Graphklassen wollen wir im Folgenden vorstellen. Diese beiden Graphen sind Graphen die über eine sogenannte *leere Nachbarschaft* definiert werden, d.h. eine Kante ist genau dann im Graph enthalten, wenn eine bestimmte Umgebung der Kante keine weiteren Knoten enthält.

Definition 2.4 (Gabriel-Graph)**.** Ein Graph G heißt Gabriel-Graph, kurz GG, wenn für jede Kante $\{u, v\}$ gilt: Der Kreis, der $\{u, v\}$ als Durchmesser hat, enthält keine weiteren Punkte als u und v.

Definition 2.5 (Relative-Neighbourhood-Graph)**.** Ein Graph G heißt Relative-Neighbourhood-Graph, kurz RNG, wenn für jede Kante $\{u, v\}$ gilt: Der Schnitt der Kreise, die u und v als Mittelpunkt und die Kante $\{u, v\}$ als Radius haben enthält keine weiteren Knoten als u und v.

Beispiele für die leeren Nachbarschaften dieser Graphen finden sich in Abb. 2.2. Betrachten wir eine Einbettung einer gegebenen Punktmenge. Dann können wir folgendes festhalten.

Lemma 2.6. *Bei fester Einbettung einer gegebenen Punktmenge ist der Relative-Neighbourhood-Graph immer ein Teilgraph des Gabriel-Graphs.*

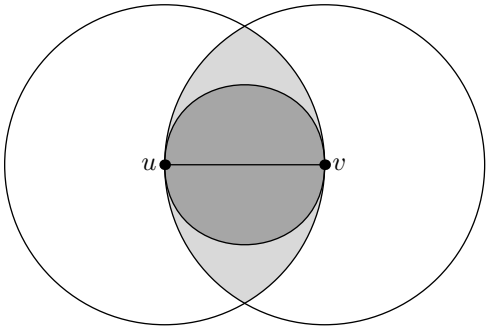


Abbildung 2.3: Diese Abbildung zeigt, dass die Nachbarschft einer Kante in einem Gabriel-Graph vollständig in der Nachbarschaft einer Kante in einem RNG enthalten ist.

Beweis. Um Lemma 2.6 zu zeigen, genügt es zu beweisen, dass ein RNG höchstens die Kanten eines Gabriel-Graphs besitzt. Klar wird das, wenn wir uns die Nachbarschaften der Kanten einmal betrachten. Abbildung 2.3 zeigt, dass die leere Nachbarschaft einer Kante eines Gabriel-Graphs vollständig in der leeren Nachbarschaft einer Kante des RNG enthalten ist. Das bedeutet aber, dass ein Gabriel-Graph auf einer Knotenmenge wenigstens die Kanten des RNG auf der selben Knotenmenge enthält. □

2.2.2 Unit-Disk-Graphen

Wir bezeichnen einen Graphen als *Unit-Disk-Graphen* oder kurz *UDG*, wenn für diesen Graphen eine Einbettung in der Ebene existiert, sodass zwischen je zwei Knoten genau dann eine Kante existiert, wenn der Abstand zwischen den Knoten ≤ 1 ist. Dieses Graphenmodell eignet sich gut um mobile Netzwerke darzustellen. Dazu interpretieren wir jeden Knoten als einen Netzwerkteilnehmer mit einem für alle Knoten gleichen Senderadius. Ein Teilnehmer kann dann einen anderen erreichen, wenn sich dieser in seinem Senderadius befindet.

Eine naheliegende Erweiterung der Unit-Disk-Graphen stellen die *Disk-Graphen* dar. Bei diesen Graphen stellt jeder Knoten v den Mittelpunkt eines Kreises C_v in der Ebene dar. Es existiert eine *gerichtete Kante* $e = (v, w)$ von v nach w dann, wenn die Fläche des Kreises C_v den Knoten e enthält. Diese Graphklasse stellt eine natürlicheres Modell für drahtlose Netzwerke dar, da nicht jeder Knoten den selben Senderadius besitzen muss. Jedoch weisen diese Graphen eine schwierigere Struktur auf, weshalb wir uns erst einmal mit der Einschränkung auf Unit-Disk-Graphen zufrieden geben wollen.

2.3 Baumweite und Baumzerlegungen

Viele Probleme der Graphentheorie lassen sich leicht lösen, wenn die zugrundeliegenden Graphen Bäume sind. Diese Beobachtung führte dazu, dass man zu untersuchen begann, wie baumähnlich ein gegebener Graph ist. Erstmalig vorgestellt wurden die Begriffe von Robertson und Seymour [RS86], eine ausführliche Zusammenfassung der Begriffe und Ideen finden sich unter anderem bei Bodlaender [Bod93]. Die in dieser Arbeit verwendeten Definitionen wollen wir hier angeben.

Definition 2.7 ([Bod93])**.** Sei $G = (V, E)$ ein Graph. Dann heißt ein Paar $\langle \{X_i \mid i \in I\}, T\rangle$ *Baumzerlegung* von G, wobei $\forall i \in I : X_i \subseteq V$ und T ein Baum mit der Knotenmenge I ist, so dass gilt:

1. $\bigcup_{i \in I} X_i = V$,
2. für jede Kante $\{v, w\} \in E$ existiert ein $i \in I$ so, dass $\{v, w\} \subseteq X_i$,
3. $\forall i, j, k \in I$: Falls j auf dem Pfad zwischen i und k in T liegt, so gilt $X_i \cap X_k \subseteq X_j$.

Die Mengen $X_i, i \in I$, werden auch als Bags bezeichnet. Die *Weite* ω einer Baumzerlegung $\langle \{X_i \mid i \in I\}, T\rangle$ ist

$$\omega := \max_{i \in I}\{|X_i|\} - 1.$$

Die *Baumweite* eines Graphen G bezeichnet den kleinsten Wert k, für den eine Baumzerlegung von G der Weite k existiert.

Es existiert eine speziellere Art der Baumzerlegung, die sich algorithmisch besser bearbeiten lässt, da an die inneren Baumknoten spezielle Forderungen gestellt werden.

Definition 2.8 ([ABF$^+$02])**.** Eine Baumzerlegung $\langle \{X_i \mid i \in I\}, T\rangle$ heißt *schöne Baumzerlegung* gdw. folgendes gilt:

1. Jeder Knoten in T hat höchstens zwei Kinder.
2. Wenn ein Knoten i zwei Kinder j und k besitzt, so gilt $X_i = X_j = X_k$. Einen solchen Knoten bezeichnen wir als *Join Node*.
3. Wenn ein Knoten i nur ein Kind j hat, so muss eine der folgenden Aussagen zutreffen:

- $|X_i| = |X_j| + 1$ und $X_j \subsetneq X_i$. Einen solchen Knoten i nennen wir *Introduce Node*.
- $|X_j| = |X_i| + 1$ und $X_i \subsetneq X_j$. Einen solchen Knoten i nennen wir *Forget Node*.

Wie das folgende Lemma zeigt, ist es leicht, aus einer gegebenen Baumzerlegung eine schöne Baumzerlegung zu konstruieren.

Lemma 2.9. *Bei einer gegebenen Baumzerlegung der Weite k mit n Baumknoten für einen Graphen G lässt sich eine schöne Baumzerlegung der Weite k mit $O(|V|)$ Baumknoten für den Graphen G in $O(|V|)$ Schritten finden.*

2.4 Das Problem Multipoint Relay

Wie anfangs bereits erläutert, wollen wir in einem mobilen Netzwerk eine Nachricht an alle Teilnehmer senden. Diese Aufgabe kommt sehr häufig zur Anwendung, da Routing-Algorithmen ohne globale Kenntnis der Netzstruktur solche sogenannten Broadcasts nutzen, um einen Sendeweg zwischen zwei Teilnehmern zu finden. Da die Knoten des Netzwerks mobil sind, kann im Allgemeinen ein Teilnehmer nicht die ganze Topologie des Netzwerkes kennen, da dazu ein viel zu großer Kommunikationsaufwand nötig wäre. Leichter ist es, Kenntnis über einen lokalen Bereich des Netzwerkes zu besitzen. Deshalb gehen wir davon aus, dass ein Netzwerkteilnehmer, die wir in Graphen durch Knoten symbolisieren wollen, seine Nachbarn und deren Nachbarn kennt. Dies ist durch zweimaligen Austausch von Kontrollinformationen möglich. In einem ersten Schritt erhält jeder Knoten die Informationen über seine Nachbarschaft, in einem zweiten Schritt sendet jeder Knoten die Menge seiner Nachbarn an die zu ihm adjazenten Knoten. Um im Falle eines Broadcasts trotz der eingeschränkten Informationen nun einerseits möglichst wenige Sendungen zu benötigen und andererseits jeden Knoten des Netzwerkes zu erreichen, wählt der Sender einige seiner Nachbarn aus, die aufgefordert werden, diese Nachricht weiterzusenden. Diese ausgewählten Nachbarn bezeichnen wir als *Multipoint Relays* eines Knotens, die Menge aller Multipoint Relays entsprechend als *Multipoint-Relay-Menge*.

Definition 2.10. Sei $G = (V, E)$ ein Graph und $x \in V$. Dann nennen wir eine Menge MPR_x Multipoint-Relay-Menge für x in G gdw. $\mathrm{MPR}_x \subseteq N(x)$ und $N^2(x) \subseteq N(\mathrm{MPR}_x)$. Wir nennen MPR_x optimal, wenn für x in G keine kleinere Multipoint-Relay-Menge existiert.

Diese Definition ermöglicht uns nun eine exakte Problembeschreibung.

Multipoint Relay

Eingabe: Ein Graph, ein Knoten x des Graphen.

Problem: Angabe einer optimalen Multipoint-Relay-Menge MPR_x.

Im Folgenden treffen wir einige Vereinbarungen, die den Umgang mit dem Problem erleichtern und die Verständlichkeit der Arbeit erhöhen sollen.

Wir betrachten bei einer Eingabeinstanz lediglich die Komponenten, die für die Berechnung einer optimalen Multipoint-Relay-Menge relevant sind. Demzufolge bestehe eine Eingabeinstanz des Problems genau aus einem ausgezeichneten Knoten eines Graphen, seinen Nachbarn und deren Nachbarn. Um die Lesbarkeit zu erhöhen, bezeichnen wir den ausgezeichneten Knoten, für den es eine Multipoint-Relay-Menge zu berechnen gilt, mit x, Knoten aus seiner Nachbarschaft als y-Knoten oder mit den Variablen $y_i, 1 \leq i \leq |N(x)|$, und Knoten aus $N^2(x)$ als z-Knoten oder mit den Variablen $z_j, 1 \leq j \leq \left|N^2(x)\right|$. Während der Berechnung von Multipoint-Relay-Mengen für einen Knoten x gehen wir o.B.d.A. davon aus, dass sowohl zwischen je zwei Knoten $y_1, y_2 \in N(x)$ als auch zwischen je zwei Knoten $z_1, z_2 \in N^2(x)$ keine Kante existiert, da für die Berechnung eines optimalen Multipoint Relays lediglich die Beziehungen zwischen Knoten aus $N(x)$ und $N^2(x)$ interessant sind. Weiterhin genügt es, wenn wir jede Zusammenhangskomponente aus $G - \{x\}$ einzeln betrachten; wir können demzufolge fordern, dass $G - \{x\}$ zusammenhängend sei. Um in späteren Argumentationen keine entarteten Sonderfälle betrachten zu müssen habe jeder Nachbar von x auch wenigstens eine Kante nach $N^2(x)$, es gelte also $\forall y \in N(x) \exists z \in N^2(x) : z \in N(y)$.

Wie wir im Folgenden zeigen ist die Berechnung einer optimalen Multipoint-Relay-Menge nicht nur nicht trivial, sondern sogar ein NP-schweres Problem.

2.5 Die Komplexität des Problems

Wir setzen als bekannt voraus, dass für NP-schwere Probleme im Allgemeinen keine effizienten Algorithmen existieren, sofern $\mathrm{P} \neq \mathrm{NP}$. Da eine Lösung für solche Probleme jedoch wünschenswert ist, hat man sich in den letzten Jahren differenzierter mit der Struktur dieser Probleme und der vermutlich unvermeidbaren kombinatorischen Explosion der Berechnung auseinandergesetzt und versucht, *Parameter* zu finden, durch die sich die Schwierigkeit der Berechnung ausdrücken und begrenzen lässt. Detaillierte Ausführungen zur sogenannten *parametrisierten Komplexität* kann man in den Büchern

von Downey und Fellows [DF99] oder Niedermeier [Nie06] finden. Wir wollen an dieser Stelle lediglich die für diese Arbeit wichtigen Grundgedanken skizzieren.

Ein *parametrisiertes Problem* ist eine Sprache $L \subseteq \Sigma^* \times \Sigma^*$ über einem endlichen Alphabet Σ. Für eine Probleminstanz $(x, k) \in L$ bezeichnet man die zweite Komponente auch als *Parameter*. Ist bei einem Problem der Parameter in praktischen Anwendungen typischerweise klein, so ist die Frage interessant, ob Algorithmen für diese Probleme existieren, deren exponenzieller (oder noch größerer) Aufwand lediglich vom Parameter und nicht von der Eingabegröße abhängt. Ein parametrisiertes Problem nennen wir *fixed-parameter tractable*, wenn für eine beliebige, berechenbare Funktion f in $f(k) \cdot |x|^{O(1)}$ Schritten berechnet werden kann, ob $(x, k) \in L$ gilt. Die Klasse dieser Probleme bezeichnet man mit FPT.

Ähnlich wie in der klassischen Komplexitätstheorie lassen sich parametrisierte Probleme aufeinander reduzieren. Analog zu Reduktionen der klassischen Komplexitätstheorie ist die parametrisierte Reduktion so beschaffen, dass eine Reduktion eines Problems L auf ein Problem $L' \in \text{FPT}$ und eine anschließende Lösung von L' gerade einen FPT-Algorithmus für L darstellen.

Definition 2.11. Wir bezeichnen eine Funktion als *parametrisierte Reduktion* von einem parametrisierten Problem L auf ein anderes parametrisiertes Problem L', wenn sie für eine gegebene Instanz (x, k) in $f(k) \cdot |x|^{O(1)}$ Schritten eine Instanz (x', k') so berechnet, dass gilt:

1. $(x, k) \in L \leftrightarrow (x', k') \in L'$,
2. k' hängt nur von k ab,

wobei f eine beliebige, berechenbare Funktion ist, die nur von k abhängt.

Es sind parametrisierte Probleme bekannt, die nicht in FPT liegen. Eine Klasse von parametrisierten Problemen, die FPT (vermutlich echt) enthält, ist die Klasse W[2][1], für die das Entscheidungsproblem Dominating Set ein hartes Problem darstellt [DF99]:

Dominating Set
Eingabe: Ein ungerichteter Graph $G = (V, E)$, ein ganzzahliger Wert k.

[1] Wir überspringen hier eine Klasse der sogenannten W-Hierarchie. Vermutlich sehen die Inklusionen der Klassen so aus: $\text{FPT} \subsetneq \text{W[1]} \subsetneq \text{W[2]} \subsetneq \ldots$

Problem: Existiert eine Teilmenge $V' \subseteq V$ mit $|V'| \leq k$ derart, dass für jeden Knoten $v \in V$ wenigstens ein Nachbarknoten oder v selbst in V' liegt?

Wir wollen ausgehend von diesem Problem zeigen, dass das Problem Multipoint Relay ebenfalls ein W[2]-hartes Problem ist. So wie in der klassischen Komplexitätstheorie vermutet wird, dass P $\neq$ NP gilt, geht man davon aus, dass FPT $\neq$ W[2] ist und somit W[2]-harte Probleme nicht in FPT liegen. Betrachten wir dazu die folgende Formulierung des Multipoint-Relay-Problems als Entscheidungsproblem:

Multipoint Relay
Eingabe: Ein Graph, ein Knoten x des Graphen, ein ganzzahliger Wert k.
Problem: Gibt es eine Multipoint-Relay-Menge MPR_x mit $|\text{MPR}_x| \leq k$?

In dieser Variante des Problems wird also nicht nach einer optimalen Lösung gesucht, sondern lediglich gefragt, ob eine Lösung der Größe k existiert. Dieses so parametrisierte Problem ist offensichtlich nicht schwerer als das ursprüngliche Problem: Angenommen wir haben eine optimale Lösung für Multipoint Relay, so genügt es die Größe der Lösung zu bestimmen und mit dem Parameter k zu vergleichen. Mit diesem Einstieg können wir folgenden Satz zeigen.

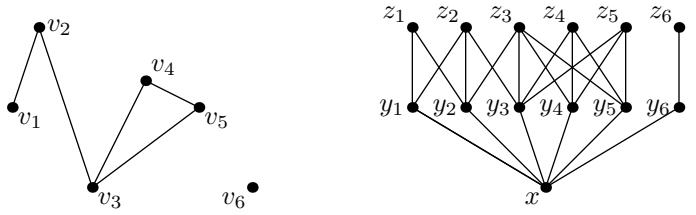
Satz 2.12. Multipoint Relay *ist W[2]-hart.*

Beweis. Im Folgenden wollen wir eine Reduktion des oben angegebenen Entscheidungsproblems Dominating Set auf das Entscheidungsproblem Multipoint Relay angeben. Sei $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ eine gegebene Dominating-Set-Instanz. Wir konstruieren dazu eine MPR-Instanz $G' = (V', E')$. Die Knotenmenge V' besteht dabei aus zwei Kopien $Y = \{y_i \mid v_i \in V\}$ und $Z = \{z_i \mid v_i \in V\}$ von V sowie einem zusätzlichen Knoten x. Die Kanten von G' beschreiben wir über die Nachbarschaften der Knoten:

$$N_{G'}(x) = Y,$$
$$N_{G'}(y_i) = \{z_i\} \cup \{z_j \mid v_j \in N_G(v_i)\} \cup \{x\}.$$

In Abb. 2.4 ist ein Beispiel einer solchen Reduktion zu sehen. Die beschriebene Reduktion lässt sich offensichtlich in polynomialer Zeit durchführen.

Wir zeigen im Folgenden, dass die Antwort auf die Frage nach einer Multipoint-Relay-Menge für den Knoten x der Größe höchstens k im soeben konstruierten Graphen auch eine Antwort auf die Frage nach einer



(i) (ii)

Abbildung 2.4: In (i) ist der Graph G und in (ii) der durch die Reduktion entstandene Graph G' zu sehen.

dominierenden Knotenmenge gleicher Größe im Ausgangsgraphen G liefert. Sei MPR_x eine solche Menge. Offensichtlich ist MPR_x eine Teilmenge von $N(x) = Y$. Gleichzeitig ist $DS = \{v_i \mid y_i \in \mathrm{MPR}_x\}$ eine dominierende Menge für G: Betrachten wir einen beliebigen Knoten $v_i \in V$ und die entsprechenden Kopien y_i und z_i. Da MPR_x eine Multipoint-Relay-Menge ist, ist z_i Nachbar eines Knotens $y_l \in \mathrm{MPR}_x$. Nach Definition der Nachbarschaftsbeziehungen in unserer Reduktion können zwei Fälle auftreten. Entweder gilt $z_i = z_l$, dann ist $v_i \in DS$. Oder es gilt $z_i \in \{z_j \mid v_j \in N_G(v_l)\}$, dann ist $v_l \in DS$ und es existiert eine Kante $\{v_i, v_l\} \in E$. Somit ist also DS eine G dominierende Knotenmenge der Größe k.

Andererseits ist G' gerade so konstruiert, dass sich aus einer dominierenden Punktmenge DS für G eine Multipoint-Relay-Menge gleicher Größe für x in G' gemäß $\mathrm{MPR}_x = \{y_i \mid v_i \in DS\}$ angeben lässt. Da für jeden Knoten $z_i \in N^2_{G'}(x)$ gerade die Kanten $\{z_i, y_j\}$ in G' enthalten sind, für die entweder gilt $i = j$ oder $\{v_i, v_j\} \in E$, ist jedes z_i Nachbar eines Knotens $y_j \in \mathrm{MPR}_x$. □

Die in diesem Beweis dargestellte Reduktion ist nicht nur eine parametrisierte Reduktion, wir können sie gleichzeitig als Polynomialzeitreduktion auffassen. Deshalb lässt sich direkt aus dem Beweis noch folgender Fakt ableiten:

Korollar 2.13. Multipoint Relay *ist NP-hart.*

Sowohl Satz 2.12 als auch Korollar 2.13 lassen vermuten, dass für Multipoint Relay auf allgemeinen Graphen keine effizient berechenbaren Lösungen existieren. Dies soll uns jedoch als Motivation dafür dienen, für spezielle Graphklassen nach effizienten Algorithmen zu suchen.

Kapitel 3

Problemlösungen durch Datenreduktionsregeln

Es gestaltet sich im Allgemeinen schwierig, durch einen globalen Blick auf einen gegebenen Graphen eine optimale Lösung für unser Problem zu finden, da der Aufwand mit zunehmender Größe der Eingabeinstanz exponenziell wächst. Durch Lokalisierung kleinerer lokaler Strukturen, die eine eindeutige Handhabung bezüglich des Problems zulassen, wird es jedoch möglich Regeln anzugeben, die den gegebenen Graphen Schritt für Schritt verkleinern und während dieses Prozesses Knoten in unsere gesuchte Lösung aufnehmen. In diesem Kapitel wollen wir einen solchen Lösungsansatz verfolgen und untersuchen, wie ein solcher Reduktionsalgorithmus beschaffen ist. Dazu zeigen wir zuerst drei Reduktionsregeln, die auf alle Klassen von Graphen anwendbar sind und somit eine beliebige Eingabeinstanz auch im Vorfeld einer Bearbeitung durch ein anderes Verfahren sinnvoll verkleinern können. Im Anschluss daran zeigen wir weitere Regeln mit denen sich planare Probleminstanzen vollständig reduzieren lassen und somit die Berechnung einer optimalen Multipoint-Relay-Menge auf diesem Wege möglich machen: Eine Berechnung einer optimalen Multipoint-Relay-Menge stellt dann eine sukzessive Anwendung der Regeln und eine damit verbundene Verkleinerung der Knotenmenge dar. Sobald eine Probleminstanz in diesem Prozess keine z-Knoten mehr enthält, terminiert der Algorithmus und liefert, wie wir zeigen werden, eine optimale Lösung.

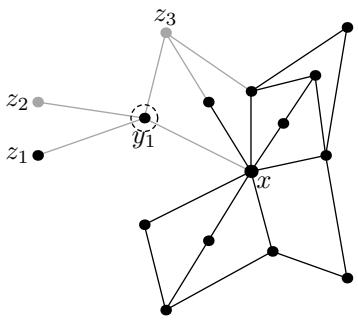


Abbildung 3.1: Der Knoten z_1 hat als einzigen Nachbarn den Knoten y_1, also fügen wir y_1 der Menge MPR_x hinzu und löschen y_1 und seine Nachbarn in G.

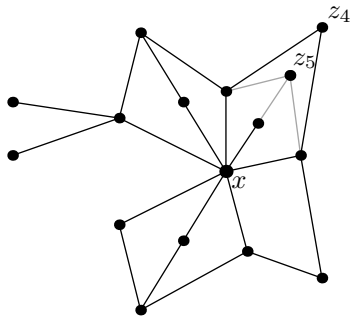


Abbildung 3.2: Der Knoten z_5 hat als Nachbarn wenigstens die Knoten, die z_4 auch hat, also können wir z_5 in G löschen.

3.1 Allgemeine Reduktionsregeln

In diesem Abschnitt wollen wir zunächst einmal drei Reduktionsregeln angeben, die wir auf beliebige Graphen anwenden können und untersuchen anschließend, wie die dadurch reduzierten Graphen strukturiert sind.

In einer ersten Regel fügen wir die y-Knoten zu MPR_x hinzu, die zwingend in einer optimalen Multipoint-Relay-Menge enthalten sein müssen.

Regel 1. *Existiert ein z-Knoten, der zu genau einem $y \in N(x)$ benachbart ist, so fügen wir y zu MPR_x hinzu und löschen y und alle zu y benachbarten z-Knoten.*

Ein Beispiel dieser Regel ist in Abb. 3.1 zu sehen.

Korrektheit: Hat ein Knoten $z_1 \in N^2(x)$ nur einen Nachbarn $y \in N(x)$, so muss dieser Knoten y in der Multipoint-Relay-Menge enthalten sein, da sonst z_1 nicht durch MPR_x dominiert werden könnte. Dann können wir diejenigen Knoten $z \in N^2(x)$ löschen, für die eine Kante $\{y, z\} \in E$ existiert, da solche z nach diesem Schritt bereits zu einem Knoten aus MPR_x benachbart sind.

Laufzeit: Wir müssen einen solchen Knoten z in G finden und anschließend alle zu y benachbarten Knoten löschen. Um einen solchen Knoten z zu finden durchsuchen wir E genau einmal. Jeder z-Knoten, der nur an

einer Kante beteiligt ist, ist ein solcher z-Knoten. Demzufolge dauert die Anwendung dieser Regel also $O(|E|)$ Schritte.

Die beiden nächsten Reduktionsregeln dienen dazu Knoten, die für die Berechnung keine Informationen liefern, aus $N^2(x)$ bzw. $N(x)$ zu löschen. Konkreter entfernt die zweite Regel diejenigen Knoten aus $N^2(x)$, die für die Bestimmung von MPR_x nicht benötigt werden.

Regel 2. *Existiert ein Knoten $z_1 \in N^2(x)$ und ein Knoten $z_2 \in N^2(x)$ in G so, dass $N(z_2) \subseteq N(z_1)$ ist, dann lösche z_1 in G.*

Abb. 3.2 zeigt ein Beispiel für die Anwendung dieser Regel.

Korrektheit: Im Laufe der Berechnung muss der Knoten z_2, dessen y-Nachbarn auch alle Nachbarn von z_1 sind, von MPR_x dominiert werden. Dadurch wird aber auch z_1 durch unsere Lösungsmenge dominiert und wir können ihn für unsere Berechnung löschen.

Laufzeit: Um festzustellen, ob für einen beliebigen Knoten z_1 ein Knoten z_2 existiert, so dass $N(z_2) \subseteq N(z_1)$ gilt, können wir alle Nachbarn von z_1 betrachten und prüfen, ob all diese noch einen weiteren gemeinsamen Nachbarn z_2 haben. Dazu fertigen wir eine Liste an, die alle Knoten eines Nachbars von z_1 enthält. Nun betrachten wir sukzessive alle weiteren Nachbarn von z_1 und streichen alle Knoten der Liste, die nicht zu jedem dieser Nachbarn adjanzent sind. Enthält nach der Betrachtung aller Nachbarn von z_1 die Liste noch mindestens einen Knoten z_2, so gilt $N(z_2) \subseteq N(z_1)$ und wir können z_1 löschen. Führen wir diesen Vorgang nun für jeden Knoten durch, so benötigen wir dafür $O(|E|^2)$ Schritte.

Wir verfahren für die y-Knoten nun genau so, löschen also in der dritten Reduktionsregel solche Knoten aus $N(x)$, die auf jeden Fall nicht in MPR_x aufgenommen werden müssen.

Regel 3. *Existiert ein Knoten $y_1 \in N(x)$ und ein Knoten $y_2 \in N(x)$ derart, dass $N(y_1) \subseteq N(y_2)$ ist, so lösche y_1 aus G.*

Zu dieser Regel findet sich ein Beispiel in Abb 3.3.

Korrektheit: Wieso können wir den Knoten y_1 ohne weiteres löschen? Angenommen y_1 ist in einer optimalen Multipoint-Relay-Menge MPR_x enthalten. Dann ist $(\mathrm{MPR}_x \setminus \{y_1\}) \cup \{y_2\}$ ebenfalls eine optimale Multipoint-Relay-Menge, da jeder Knoten, der durch y_1 dominiert wurde, auch zu y_2 benachbart ist.

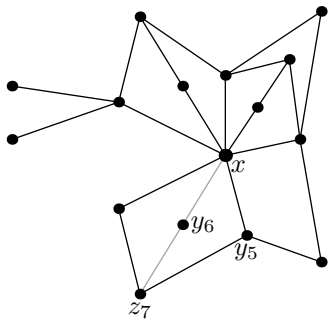


Abbildung 3.3: Die Nachbarschaft von y_6 ist eine Teilmenge der Nachbarschaft von y_5, also können wir y_6 aus G löschen.

Laufzeit: Wir verfahren analog zu Regel 3, fertigen also für einen Knoten y_1 eine Liste L an, in der die gemeinsamen Nachbarn seiner Nachbarn vermerkt werden. Ist in der Liste nach Betrachtung aller Nachbarn von y_1 noch ein Knoten y_2 enthalten, so ist $N(y_1) \subseteq N(y_2)$. Somit verhält sich der Aufwand wie in Regel 3, also $O(|E|^2)$.

Bevor wir uns im nächsten Abschnitt mit den Regeln für planare Graphen auseinandersetzen, wollen wir eine offensichtliche Eigenschaft einer durch die ersten drei Regeln vollstandig reduzierten Probleminstanz festhalten.

Lemma 3.1. *Sei G durch die Regeln 1 bis 3 vollständig reduziert. Dann besitzt jeder Knoten v in $G - \{x\}$ wenigstens den Knotengrad 2.*

Beweis. Würde ein z-Knoten z mit $\deg_{G-\{x\}}(z) = 1$ existieren ließe sich Regel 1 anwenden. Ist dies nicht der Fall und es existiert ein $y \in N(x)$ mit $\deg_{G-\{x\}}(y) = 1$, dann könnte man aber Regel 3 anwenden. Also muss jeder Knoten in $G - \{x\}$ wenigstens den Grad zwei besitzen. □

3.2 MPR mit Reduktionsregeln auf planaren Graphen

Nachdem wir im vorigen Abschnitt Reduktionsregeln für allgemeine Graphen untersucht haben, wollen wir uns nun ausführlich mit planaren Graphen auseinandersetzen und eine Menge von Regeln angeben die es uns ermöglichen, für Graphen dieser Klasse optimale Multipoint-Relay-Mengen zu berechnen.

Die Aufgabe besteht also darin, die nach Anwendung der ersten drei Regeln noch bestehenden Strukturen optimal aufzulösen. Offensichtlich existiert in $G' := G - \{x\}$ wenigstens ein Kreis, da nach Lemma 3.1 jeder Knoten in diesem Graph wenigstens den Grad 2 besitzt. Bei einer näheren Betrachtung des Problems werden wir sehen, dass es sinnvoll ist, von innen nach außen vorzugehen, da weiter innen liegende Kreise eine optimale Auflösung im Sinne von Multipoint Relay zulassen. Dazu müssen wir uns zuerst darüber klar werden, was „innen“ bedeutet und wie solche „inneren“ Kreise beschrieben werden können. Dazu wollen wir zuerst einmal die etwas technische Definition angeben und direkt im Anschluss an Beispielen verdeutlichen.

Definition 3.2. Sei G eine planare Probleminstanz mit gegebener planarer Einbettung ϕ. Wir nennen eine Kante e eine *innere Kante* in ϕ, wenn e an einer Fläche mit x liegt. Entsprechend bezeichnen wir jede Kante, die an keiner Fläche mit x liegt als *äußere Kante*. Einen Kreis C in G nennen wir einen *inneren Kreis* in ϕ, wenn eine der folgenden Aussagen zutrifft:

- Der Kreis C enthält keine oder genau eine äußere Kante.
- Falls C zwei äußere Kanten besitzt, so besitzen diese Kanten einen gemeinsamen z-Knoten.

Wir bezeichnen einen Kreis C als *leeren Kreis* in ϕ, wenn in der durch C umschlossenen Fläche keine Kante liegt.

Die in Abb. 3.2 dargestellten Graphen mit den hervorgehobenen Kreisen sollen die Definition nochmals verdeutlichen. In (i) und (ii) werden zwei Kreise gezeigt, die innere Kreise sind, einmal mit keiner und einmal mit einer äußeren Kante. Der in (i) dargestellte Fall ist der Sonderfall, dass $G - \{x\}$ selbst ein Kreis ist, der Graph also nur aus inneren Kanten besteht. Der in (iii) hervorgehobene Kreis ist ein innerer Kreis mit zwei äußeren Kanten. Solche Kreise können nur auftreten, wenn es einen z-Knoten gibt, der an keiner Fläche mit x liegt. In (iv) ist ein Kreis markiert, der kein innerer Kreis ist.

Lemma 3.3. *Sei G gemäß Regeln 1 bis 3 reduziert und ϕ eine planare Einbettung von G. Dann existiert in ϕ ein innerer, leerer Kreis.*

Beweis. Zuerst machen wir uns klar, dass es in jeder gemäß Regeln 1 bis 3 reduzierten MPR-Instanz einen inneren Kreis geben muss. Betrachten wir dazu eine äußere Kante $e = \{y_1, z_1\}$. Einerseits kann der Knoten z der Kante

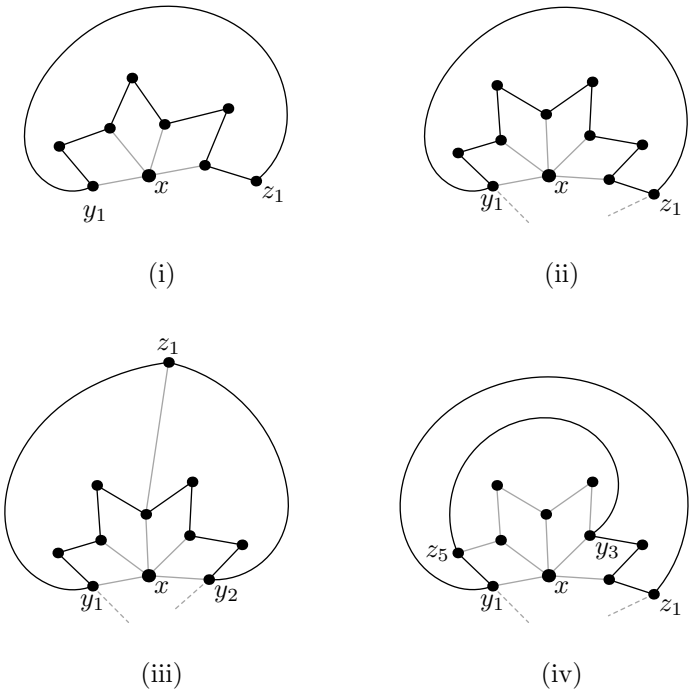


Abbildung 3.4: Wir gehen in den Bildern o.B.d.A. davon aus, dass die durch die gepunkteten Linien angedeuten Teilgraphen untereinander verbunden sind. Der in (i) dunkel markierte Kreis stellt einen inneren, leeren Kreis mit keiner äußeren Kante dar. In (ii) ist der dunkel markierte Kreis ein innerer, leerer Kreis mit der äußeren Kante $\{y_1, z_1\}$. Der in (iii) dunkel dargestellte Kreis stellt einen inneren Kreis mit zwei äußeren Kanten $\{y_1, z_1\}$ und $\{z_1, y_2\}$ dar, der nicht leer ist. Und in (iv) ist der dunkel marierte Kreis ein leerer Kreis, der kein innerer Kreis ist, da er zwei nicht benachbarte äußere Kanten $\{y_1, z_1\}$ und $\{y_3, z_5\}$ enthält.

an einer gemeinsamen Fläche mit x liegen, wie z.B. in Abb. 3.2 (ii). Dann muss aber ein Pfad $(z_1, \ldots, z_i, y_1)$ existieren, der nur aus inneren Kanten besteht, da die äußere Kante e von jeder Fläche, an der x liegt, getrennt ist. Andererseits kann es sein, dass der Knoten z_1 an keiner Fläche mit x liegt, ein Beispiel dazu ist in Abb. 3.2 (iii) zu sehen. Da z_1 wegen Lemma 3.1 mindestens den Knotengrad 2 besitzt, muss eine weitere äußere Kante $e' = \{z_1, y_2\}$ existieren und dann auch ein Pfad $(y_2, z_2, \ldots, z_i, y_1)$, der nur aus inneren Kanten besteht.

Nun wollen wir noch zeigen, dass auch mindestens ein innerer, leerer Kreis existieren muss. Dazu betrachten wir einen inneren Kreis. Ist dieser nicht leer, so muss er mindestens eine äußere Kante enthalten. Mit einer Fallunterscheidung analog zum ersten Teil des Beweises muss auch hier wieder ein Pfad existieren, der nur aus inneren Kanten besteht. Wir können diesen so wählen, dass er nur aus inneren Kanten des betrachteten inneren Kreises besteht. Mit diesem Pfad und der (oder den) äußeren Kanten haben wir einen inneren Kreis gefunden, der weniger innere Kanten enthält. Ist dieser Kreis kein innerer, leerer Kreis, so können wir den beschriebenen Vorgang wiederholt anwenden, bis ein leerer innerer Kreis gefunden wurde. □

Da wir die folgenden Reduktionsregeln auf die eben beschriebenen leeren, inneren Kreise anwenden wollen, muss es uns gelingen einen solchen Kreis effizient zu finden. Damit beschäftigt sich das nächste Lemma.

Lemma 3.4. *In einer bereits durch die Regeln 1 bis 3 reduzierten Multipoint-Relay-Instanz $G = (V, E)$ mit gegebener planarer Einbettung lässt sich ein innerer, leerer Kreis in $O(n)$ Schritten finden.*

Beweis. Wir betrachten zunächst einmal den Graphen $G - \{x\}$ mit einer planaren Einbettung ϕ. Von dieser Einbettung wollen wir fordern, dass die Fläche f_x, die durch Löschen des Knotens x entstanden ist, und die äußere Fläche f_{ext} verschieden sind. Eine solche Einbettung können wir immer finden, da jede durch die ersten drei Regeln reduzierte planare MPR-Instanz einen Kreis bestehend nur aus inneren Kanten haben muss. Diesen können wir so einbetten, dass er x komplett von der Fläche f_{ext} trennt.

Wir wollen uns nun einen Hilfsgraphen $H = (V_H, E_H)$ wie folgt konstruieren. Sei $(G-\{x\})^*$ der zu $G-\{x\}$ duale Graph. Der Graph H ist der Graph, der aus $(G - \{x\})^*$ entsteht, wenn wir den Knoten v_x, der die Fläche f_x aus $G - \{x\}$ repräsentiert, löschen. Weiterhin sei $v_{ext} \in V_H$ derjenige Knoten, der die äußere Fläche f_{ext} repräsentiert.

Wir wollen nun zeigen, dass jeder Kreis in G, der einen weitestmöglich vom Knoten v_{ext} entfernten Knoten $v \in V_H$ umschließt, ein innerer, leerer

Kreis ist. Klar ist, dass jeder Kreis, der einen Knoten aus H umschließt, ein leerer Kreis ist.

Betrachten wir nun einen Knoten $v_1 \in V_H$ mit größtmöglichem Abstand zu v_{ext}. Angenommen dieser Kreis ist kein innerer Kreis. Dann existieren mindestens zwei äußere Kanten in der Begrenzung des Kreises, die nicht benachbart sind. Eine dieser Kanten trennt f_1 in G von einer Fläche, deren Abstand zu f_{ext} eins geringer ist als der von f_1. Die zweite äußere Kante trennt f_1 von einer weiteren Fläche f_2. Da f_1 und f_x die Fläche f_2 jedoch komplett umschließen, muss nun in H ein Pfad von v_{ext} zu v_2 über v_1 führen, da v_x nicht in H enthalten ist. Demzufolge hat v_2 einen größeren Abstand zu v_{ext} als der Knoten v_1. Demzufolge ist jeder so gefundene Kreis ein innerer, leerer Kreis.

Wir haben in Lemma 2.2 gesehen, dass sich der duale Graph in $O(n)$ Schritten finden lässt. In dem wie oben beschrieben konstuierten Graphen H lässt sich ein von v_{ext} weitestmöglich entfernter Knoten in $O(n)$ Schritten finden, indem wir jeden Knoten bei v_{ext} beginnend mit dem Abstand d zu v_{ext} markieren, d.h. die Nachbarn von v_{ext} werden mit $d = 1$ markiert, deren bis dahin unmarierte Nachbarn mit $d = 2$ usw.

Insgesamt lässt sich ein innerer, leererKreis in $O(n)$ Schritten finden.

□

Auf den folgenden Seiten werden wir betrachten, welche Eigenschaften die auftretenden inneren Kreise haben können und für diese Varianten einen Reduktionsschritt zur Berechnung einer Multipoint-Relay-Menge angeben. Da sich alle nun folgenden Datenreduktionen auf die beschriebenen inneren, leeren Kreise beziehen, werden wir im direkten Anschluss an die Regeln nun deren Korrektheit betrachten. Die Laufzeit einer vollständigen Datenreduktion untersuchen wir nach Angabe aller Reduktionsregeln.

Zuerst betrachten wir die einfachste Variante, einen Kreis in dem jeder z-Knoten zu genau zwei y-Knoten benachbart ist und umgekehrt.

Regel 4. *Sei G ein Graph mit planarer Einbettung ϕ und entsprechend Regeln 1 bis 3 reduziert. Weiterhin sei $C = (y_1, z_1, y_2, z_2, \ldots, y_k, z_k, y_1)$ ein innerer, leerer Kreis in ϕ derart, dass $\deg(y_i) = 3$ für jeden Knoten $y_i \in C, 1 \leq i \leq k$ und $\deg(z_i) = 2$ für jeden Knoten $z_i \in C, 1 \leq i \leq k$ gilt. Dann füge y_1 zu MPR hinzu und lösche y_1, z_1 und z_k aus G.*

Ein Beispiel für die Anwendung dieser Regel findet sich in Abb. 3.5.

Korrektheit: Gilt $\deg(y_i) = 3$ für jeden Knoten $y_i \in C, 1 \leq i \leq k$, und $\deg(z_i) = 2$ für jeden Knoten $z_i \in C, 1 \leq i \leq k$, so ist $G - \{x\}$ ein Kreis von

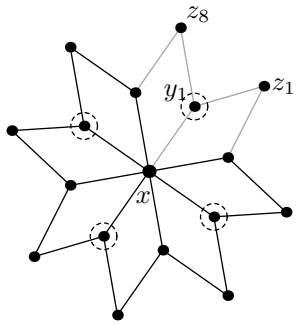


Abbildung 3.5: Die Anwendung der Regel 4 führt dazu, dass die Knoten y_1, z_1 und z_8 sowie die hell markierten Kanten gelöscht werden. Nach Ausführen der Regel lässt sich der Kreis mit den ersten drei Regeln auflösen. Im Bild würden dadurch die markierten y-Knoten in MPR$'$ aufgenommen werden.

dem keine weitere Kante wegführt. Um die Knoten $z_i, 1 \leq i \leq k$, mit unserer Multipoint-Relay-Menge zu dominieren benötigen wir mindestens $\lceil k/2 \rceil$ y-Knoten des Kreises, da jeder y-Knoten höchstens zwei noch nicht dominierte z-Knoten dominieren kann. Wir können eine Lösung mit $\lceil k/2 \rceil$ Knoten finden, die demzufolge optimal sein muss. Dazu fügen wir einen beliebigen Knoten, o.B.d.A. y_1, der Menge MPR$'$ hinzu. Dies führt dazu, dass wir nach Regel 3 weitere Knoten löschen können um dann wieder Regel 1 zu benutzen um weitere Knoten zu unserer Multipoint-Relay-Menge hinzuzufügen. Als Konsequenz dessen werden $\lceil k/2 \rceil$ Knoten zu MPR$'$ hinzugefügt.

In den nächsten Schritten werden wir die verschiedenen inneren, leeren Kreise untersuchen, die möglich sind. Die folgende Regel dient dazu innere, leere Kreise aufzulösen, bei denen ein y-Knoten noch zu z-Knoten benachbart ist, die nicht auf dem Kreis liegen.

Regel 5. *Sei G ein Graph mit einer planaren Einbettung ϕ und entsprechend Regeln 1 bis 3 reduziert. Weiterhin sei $C = (y_1, z_1, y_2, z_2, \ldots, y_k, z_k, y_1)$ ein innerer, leerer Kreis in ϕ derart, dass* $\deg(y_1) > 3$*, für jeden y-Knoten $y_i, 2 \leq i \leq k$,* $\deg(y_i) = 3$ *und* $\deg(z_i) = 2$ *für jeden Knoten $z_i \in C, 1 \leq i \leq k$. Dann füge y_1 zu MPR$'$ hinzu und lösche y_1 und jeden z-Knoten in G, der zu y_1 benachbart ist.*

In Abb. 3.6 ist ein Anwendungsbeispiel dieser Regel zu sehen.

Korrektheit: Auch hier benötigen wir mit dem selben Argument wie oben wieder $\lceil k/2 \rceil$ y-Knoten, um die Menge $\{z_1, z_2, \ldots, z_k\}$ zu dominieren. Sobald

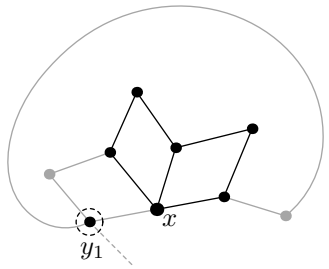


Abbildung 3.6: Der Knoten y_1 wird in die Menge MPR$'$ aufgenommen. Dadurch werden die ersten drei Reduktionsregeln wieder anwendbar und der Kreis optimal aufgelöst.

wir einen y-Knoten dieses Kreises wählen und diesen in unsere Multipoint-Relay-Menge aufnehmen, lässt sich der Kreis mit den bekannten Regeln auflösen. Im Unterschied zum letzten Fall existiert aber ein ausgezeichneter Knoten y_1, der z-Knoten dominiert, die nicht auf C liegen. Deshalb nehmen wir genau dieses y_1 in MPR$'$ auf, da wir so mit einer für C minimal nötigen Anzahl von y-Knoten eine maximale Anzahl z-Knoten dominieren.

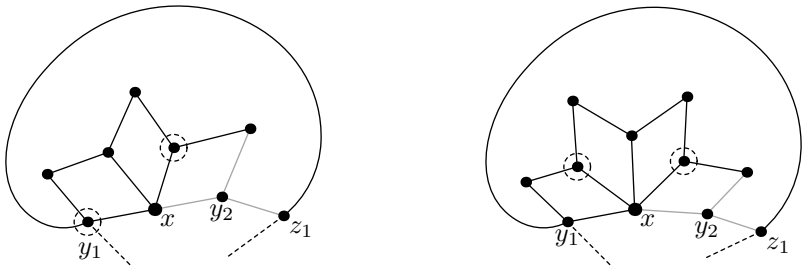
Die im Folgenden aufgeführte Regel dient dazu innere, leere Kreise aufzulösen, bei denen ein z-Knoten einen Knotengrad größer zwei und höchstens ein y-Knoten den Grad größer drei besitzt.

Es trägt wesentlich zum Verständnis der Regel bei, wenn wir uns vorher klarmachen, dass in einem solchen Kreis die beiden gerade beschriebenen Knoten direkt durch eine äußere Kante verbunden sind. Denn nur so ist es möglich, dass die Kanten an y_1 und z_1, die nicht zum Kreis gehören weder in die Kreisfläche noch in von x und dem Kreis begrenzten Gebiete hineinführen, d.h. der Kreis also tatsächlich einen inneren, leeren Kreis darstellt.

Regel 6. *Sei G ein Graph mit einer planaren Einbettung ϕ und entsprechend Regeln 1 bis 3 reduziert. Weiterhin sei $C = (y_1, z_1, y_2, z_2, \ldots, y_k, z_k, y_1)$ ein innerer, leerer Kreis in ϕ derart, dass* $\deg(y_1) \geq 3$, $\deg(z_1) > 2$, *für jeden weiteren y-Knoten $y_i, 2 \leq i \leq k$,* $\deg(y_i) = 3$ *und für jeden weiteren z-Knoten $z_i, 2 \leq i \leq k$,* $\deg(z_i) = 2$. *Dann lösche y_2 aus G.*

Zwei Beispiele für die Anwendung dieser Regel finden sich in Abb. 3.7.

Korrektheit: Bei dieser Regel benötigen wir $\lceil k/2 \rceil$ Knoten, um die Menge $\{z_1, z_2, \ldots, z_k\}$ durch MPR$'$ zu dominieren. Wir können aber nur von



(i) (ii)

Abbildung 3.7: Nach Löschen des Knotens y_2 und den hell markierten Kanten lässt sich der Kreis mit den ersten drei Regeln auflösen. Im Bild würden danach jeweils die markierten y-Knoten in MPR$'$ aufgenommen werden. Abhängig von der Anzahl der im Kreis vorkommenden y-Knoten muss y_1 entweder in eine optimale Lösungsmenge aufgenommen (i) oder im weiteren Verlauf des Algorithmus' untersucht (ii) werden.

$\{z_2, z_3, \ldots, z_k\}$ mit Sicherheit sagen, dass sie von y-Knoten aus C dominiert werden müssen, da sie ja zu keinen weiteren y-Knoten benachbart sind. Nun können zwei Fälle auftreten: Im ersten Fall ist k ungerade. Dann gelingt es uns die Menge $\{z_2, z_3, \ldots, z_k\}$ mit $\lceil k/2 \rceil - 1$ Knoten zu dominieren, indem wir $\{y_3, y_5, \ldots, y_k\}$ durch Anwendung der ersten drei Regeln MPR$'$ hinzufügen. Das ergibt sich aber, wenn wir y_2 aus G löschen. Welcher y-Knoten zu MPR$'$ hinzugefügt wird um z_1 zu dominieren wird später entschieden. Ein Beispiel zu diesem Fall findet sich in Abb. 3.7 (ii). Im zweiten Fall ist k gerade, dann benötigen wir sowohl für $\{z_1, z_2, \ldots, z_k\}$ als auch für $\{z_2, z_3, \ldots, z_k\}$ genau $\lceil k/2 \rceil$ dominierende Knoten. Dann ist es aber günstig, y_1 in die zu konstruierende Multipoint-Relay-Menge aufzunehmen, da dieser Knoten weitere z-Knoten dominiert. Wenn wir nun y_2 in G löschen, wird durch mehrfache Anwendung der ersten drei Regeln wieder jeder zweite y-Knoten in MPR$'$ aufgenommen, nämlich genau y_3, y_5, …, y_{k-1} und y_1. In Abb. 3.7 (i) wird dieser Fall dargestellt. In beiden Fällen ist es also korrekt, den Knoten y_2 nicht in die Multipoint-Relay-Menge aufzunehmen.

Nun bleiben nur noch innere, leere Kreise übrig, bei denen ein höchstens z-Knoten und genau zwei y-Knoten Verbindungen zu Knoten besitzen, die nicht zum Kreis gehören. Wir überlegen wir uns leicht, dass diese Art von Kreisen die letzte sind, die wir betrachten müssen. Würde ein weiterer y- oder z-Knoten zusätzliche Kanten besitzen, so müssten diese entweder an-

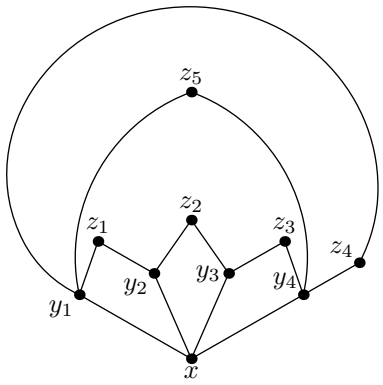


Abbildung 3.8: Der Kreis $(y_1, z_1, \ldots, z_3, y_4, z_5, y_1)$ stellt einen inneren, leeren Kreis mit zwei benachbarten äußeren Kanten dar. Dabei liegt der z-Knoten, der zu beiden äußeren Kanten inzident ist, an keiner Fläche mit x.

dere Kanten kreuzen oder zu Knoten innerhalb des Kreises führen. Ersteres würde die Planarität des Graphen zerstören, zweiteres stünde im Widerspruch dazu, dass wir einen leeren, inneren Kreis betrachten.

In Abb. 3.8 stellt der Kreis $(y_1, z_1, \ldots, z_3, y_4, z_5, y_1)$ ein Beispiel dar. Hier sieht man auch, dass der z-Knoten mit Grad ≥ 3 zu beiden y-Knoten mit Grad ≥ 4 benachbart sein muss.

Ein Beispiel soll zunächst einmal verdeutlichen, dass es für diese Kreise allerdings keine eindeutige Zuordnung einer optimalen Multipoint-Relay-Menge gibt. Betrachten wir dazu den Kreis $(y_1, z_1, \ldots, z_3, y_4, z_5, y_1)$, der in Abb. 3.8 dargestellt ist. Eine optimale Multipoint-Relay-Menge für diesen Kreis stellen sowohl die Knoten y_1 und y_3 als auch die Knoten y_2 und y_4 dar. Nun kann aber der Fall eintreten, dass entweder y_1 oder y_4 durch eine benachbarte Struktur in eine optimale Lösung erzwungen wird. Dann ist aber nur eine der beiden für diesen Kreis optimalen Teillösungen im Sinne der globalen Lösung optimal.

Mit dieser Erkenntnis müssen wir uns aber folgende Frage stellen. Wieso versuchen wir dann nicht erst, die benachbarten Strukturen aufzulösen? Diese Frage lässt sich leicht beantworten, denn es lässt sich ein Graph konstruieren, der eine Reduktion eines solchen Kreises erfordert, bevor weitere Regeln anwendbar sind. Auch hier dient uns nochmals der in Abb. 3.8 dargestellte Graph als Beispiel. Hier ist keine der uns bisher bekannten Regeln anwendbar, wir müssen also einen inneren, leeren Kreis mit zwei äußeren Kanten auflösen. Am Beispiel wird noch eine Eigenschaft offensichtlich: Die z-Knoten, die durch die bisherigen Reduktionsregeln nicht bearbeitet wer-

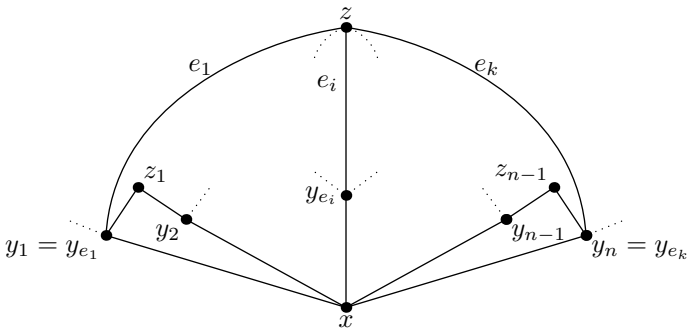


Abbildung 3.9: In diesem Bild sind die Bezeichnungen für den Teilgraphen H dargestellt, die wir bei Regel 7 für z-Knoten, die an keiner Fläche mit x liegen, verwenden.

den können und die eben beschriebenen Strukturen verursachen, sind genau die z-Knoten, die nur an Flächen liegen, an denen x nicht liegt.

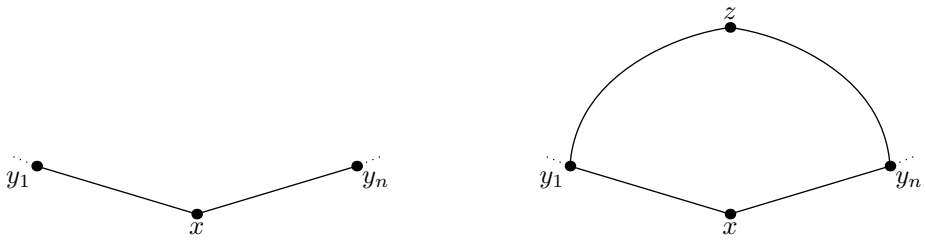
Betrachten wir nun einen solchen Knoten z. O.B.d.A. können wir alle Kanten von z in einer gegebenen planaren Einbettung der Reihe nach mit e_1 bis e_k bezeichnen, wie es in Abb. 3.9 dargestellt ist. Jede der Kanten $e_1, \ldots, e_k$ ist eine äußere Kante, da sonst Regel 6 anwendbar wäre. Weiterhin bezeichne H den Teilgraphen, der den Kreis (z, y_1, x, y_n, z) und alle in dieser Fläche liegenden Knoten enthält.

Regel 7. *Sei G ein Graph mit einer planaren Einbettung ϕ und entsprechend Regeln 1 bis 3 reduziert. Weiterhin sei z ein Knoten wie eben beschrieben und seien alle Kreise $(z, y_{e_i}, z_j, \ldots, y_{e_{i+1}}, z), 1 \leq i \leq k-1$, innere, leere Kreise. Dann können zwei Fälle auftreten.*

- *Ist n ungerade und $\{y_2, y_4, \ldots, y_{n-1}\}$ dominieren $\{z, z_1, z_2, \ldots, z_{n-1}\}$, dann ersetze H wie in Abb. 3.10 (i) gezeigt.*
- *Sonst ersetze H wie in Abb. 3.10 (ii) gezeigt.*

Sobald in G sowohl y_1 als auch y_n reduziert wurden, bestimme für H eine optilmale Lösung wie folgt: Falls y_1 in MPRenthalten ist, so löschen ihn und alle seine z-Nachbarn in H. Falls y_1 nicht in MPRenthalten ist, so lösche y_1 in H. Mit y_n verfahre analog. Anschließend lässt sich H mit den Regeln 1 bis 5 auflösen.

Korrektheit: Betrachten wir zunächt den Pfad $(y_1, z_1, y_2, z_2, \ldots, z_{n-1}, y_n)$. Offensichtlich benötigen wir mindestens in beiden Fällen $\lceil n/2 \rceil$ Knoten um



(i) (ii)

Abbildung 3.10: In Abhängigkeit davon, wie sich die z-Knoten in Abb. 3.9 dominieren lassen, ersetzen wir den Teilgraphen H durch (i) oder (ii). Die beiden Fälle sind ausführlich in Regel 7 dargestellt.

die z-Knoten des Pfades zu dominieren. Zusätzlich müssen wir noch sicherstellen, dass auch der Knoten z dominiert wird.

Im ersten Fall der Regel führt ein wiederholtes Anwenden der bereits bekannten Regeln auf den Teilgraphen H zu der in der Regel genannten Menge $\{y_2, y_4, \ldots, y_{n-1}\}$, die sowohl alle z-Knoten des Teilgraphs dominiert als auch eine minimale Größe von $(n+1)/2$ hat. Wieso nehmen wir sie also nicht direkt als optimale Lösung? Es kann in G der Fall auftreten, dass sowohl y_1 als auch y_n in eine optimale Lösung aufgenommen werden müssen. Dann sind aber sowohl z als auch z_1 und z_{n-1} bereits dominiert und wir können für den Pfad $(y_2, z_2, \ldots, z_{n-2}, y_{n-1})$ eine optimale dominierende Menge mit $(n-1)/2$ Elementen finden, nämlich $\{y_3, y_5, \ldots, y_{n-2}\}$, genau die Menge, die eine Anwendung der bekannten Regeln auf den Graphen H nach Übertragen der Informationen für die Knoten y_1 und y_n liefert.

Betrachten wir den zweiten Fall der Regel. Hier müssen wir nochmals in zwei Fälle untergliedern. Ist n ungerade, so dominieren $\{y_2, y_4, \ldots, y_{n-1}\}$ den Knoten z nicht, da wir uns im zweiten Fall der Regel befinden. Wir müssen also einen weiteren Knoten zu unserer optimalen Lösung dazunehmen, um z zu dominieren. Sinnvollerweise ist dies entweder y_1 oder y_n, da diese auch weitere Knoten in G dominieren könnten. Die Entscheidung, welcher der beiden Knoten tatsächlich dafür ausgewählt wird, wird im Verlaufe der Berechnung einer optimalen Multipoint-Relay-Menge getroffen. Sollten auch hier wieder beide in eine optimale Lösungsmenge aufgenommen werden, so verfahren wir analog zum ersten Fall der Regel. Ist n gerade, so wären entweder $\{y_1, y_3, \ldots, y_{n-1}\}$ oder $\{y_2, y_4, \ldots, y_n\}$ optimale dominierende Mengen für H, die auch wieder durch Anwendung der bereits bekannten Regeln gefunden werden. Da wir in H aber nicht entscheiden können, welcher der

Knoten y_1 oder y_n gut für eine optimale Lösung ist, überlassen wir diese Entscheidung dem Algorithmus für G.

Satz 3.5. *Ein Algorithmus, der eine Multipoint-Relay-Menge auf einer planaren Probleminstanz berechnet, liefert eine optimale Lösung in höchstens $O(n^3)$ Schritten.*

Beweis. Wir wollen zuerst untersuchen, wie lange es dauert, bis mindestens ein Knoten des Graphen $G = (V, E)$ reduziert wurde. Wir prüfen dazu zuerst, ob eine der Regeln 1 bis 3 anwendbar ist. Dies ist in $O(|E|^2) = O(n^2)$ Schritten möglich. Ist keine der Regeln anwendbar, so bestimmen wir einen inneren, leeren Kreis mittels Lemma 3.4 in $O(n)$ Schritten und reduzieren diesen entsprechend der Regeln 4 bis 7. Insgesamt benötigen wir also $O(n^2)$ Schritte, mindestens einen Knoten zu reduzieren. Da wir diesen Vorgang für jeden Knoten durchführen müssen, beträgt der Aufwand eine gegebene planare Probleminstanz zu vollständig zu reduzieren $O(n \cdot n^2) = O(n^3)$ Schritte. □

Wir haben gesehen, wie sich für planare Graphen mittels Datenreduktion eine optimale Multipoint-Relay-Menge finden lässt. Wir wollen untersuchen, welche der gezeigten Reduktionsregeln für die beiden vorgestellten Routing-Graphen relevant sind.

3.2.1 Gabriel-Graphen

Da Gabriel-Graphen eine Unterklasse der planaren Graphen darstellen, genügen offensichtlich die in diesem Kapitel vorgestellten Reduktionsregeln, um für diese Graphen eine optimale Multipoint-Relay-Menge zu bestimmen. Es stellt sich die Frage, ob wir die Menge der Regeln für diese Graphen weiter einschränken können. Dass dies tatsächlich der Fall ist, zeigt folgender Satz.

Satz 3.6. *In der Klasse der Gabriel-Graphen lässt sich eine optimale Multipoint-Relay-Menge mit den Reduktionsregeln 1 bis 4 berechnen.*

Beweis. Sei G eine Multipoint-Relay-Instanz, die aus einem Gabriel Graph konstruiert wurde. Nach Anwendung der ersten drei Reduktionsregeln gelte für alle Knoten $v \in V$ im Folgenden $\deg_G(v) \geq 2$, wie in Lemma 3.1 gezeigt. Wir betrachten vier Knoten $v_1, \ldots, v_4 \in V$ mit $\{v_1, v_2\}$, $\{v_2, v_3\}$, $\{v_3, v_4\}$, $\{v_4, v_1\} \in E$. Wenn wir nun eine Gerade g durch v_1 und v_3 konstruieren und dazu die leeren Nachbarschaften konstruieren (Abb. 3.11) sehen wir Folgendes: Die von den vier Knoten umschlossene Fläche darf keine weiteren Knoten enthalten. Die beiden leeren Nachbarschaften von $\{v_1, v_2\}$ und

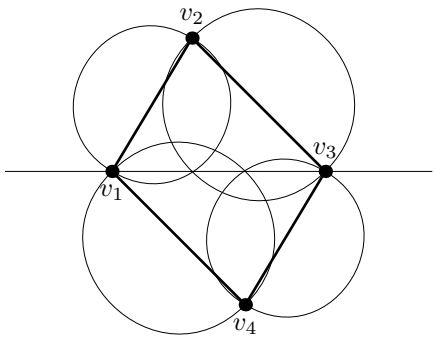


Abbildung 3.11: Die vier Knoten v_1, v_2, v_3, v_4 und die von ihnen eingeschloßene Fläche zusammen mit den Nachbarschaften der Kanten.

$\{v_2, v_3\}$ decken den Teil der Fläche ab, der auf der Seite der Geraden liegt, auf der sich auch v_2 befindet. Dass dies immer so sein muss, machen wir uns klar, indem wir die beiden Schnittpunkte der Kreise betrachten. Einer der Schnittpunkte ist der Knoten v_2 und der andere Schnittpunkt s liegt genau auf g, da sowohl v_1, v_2, s als auch v_2, v_3, s Dreiecke mit einem rechten Winkel an s darstellen und die Kreise gerade die Umkreise der Dreiecke sind. Das bedeutet aber, dass es auf der Seite von g, auf der v_2 liegt, keinen weiteren Knoten geben kann, der sowohl zu v_1 als auch zu v_3 benachbart ist. Analog kann man nun für die andere Seite der Geraden g und v_4 argumentieren. Wir wollen nun untersuchen, was diese Beobachtung für unser Problem bedeutet:

- Angenommen die Knoten v_1 und v_3 sind y-Knoten. Dann ist aber o.B.d.A. v_2 unser Knoten x, der ja zu beiden y-Knoten benachbart sein muss. Dann existiert aber höchstens ein z-Knoten, nämlich v_4, der zu beiden y-Knoten benachbart ist.

- Angenommen die Knoten v_2 und v_4 sind y-Knoten. Dann ist o.B.d.A. v_1 unser Knoten x und v_3 ein z-Knoten. Daran macht man sich aber leicht klar, dass jeder y-Knoten zu höchstens zwei z-Knoten vom Grad 2 benachbart sein kann, da jeder y-Knoten nur an zwei derartig konstruierten x-z-Geraden liegen kann.

So beschaffenen Graphen lassen sich aber gerade mit den Regeln 1 bis 4 auflösen. □

Der Satz 3.6 zeigt, dass wir uns auf Gabriel-Graphen bis auf den Sonderfall, der durch Regel 4 abgedeckt wird, nicht mit inneren Kreisen zu

beschäftigen brauchen. Das führt dazu, dass wir uns während einer Berechnung auf Gabriel-Graphen auch nicht dafür interessieren, ob eine Kante eine innere oder äußere Kante ist. Weiterhin lässt sich beobachten, dass sich die Regeln 1 bis 4 auch ohne eine gegebene Einbettung auf einen Graphen anwenden lassen.

3.2.2 Relative-Neighbourhood-Graph

Satz 3.7. *In einem Relative-Neighbourhood-Graph lässt sich eine optimale Multipoint-Relay-Menge mit den Reduktionsregeln 1 bis 4 berechnen.*

Wir haben in Lemma 2.6 bereits gezeigt, dass der RNG einen Teilgraph des Gabriel-Graphen darstellt. Damit ist die Gültigkeit des Satzes 3.7 offensichtlich.

Kapitel 4

Dynamisches Programmieren und Baumzerlegungen

In diesem Kapitel wollen wir uns zunächst mit einem Algorithmus auseinandersetzen, der über den Ansatz des dynamischen Programmierens bei einer gegebnenen Baumzerlegung einer Multipoint-Relay-Instanz eine optimale Lösung liefert. Im Anschluss dieser ausführlichen Betrachtung wollen wir für r-außerplanare Graphen zeigen, wie eine Baumzerlegung mit konstanter Baumweite schnell konstruiert werden kann und dass wir so für unsere Probleminstanz eine Baumzerlegung mit konstanter Baumweite berechnen können. Der hier vorgestellte Baumzerlegungsalgorithmus ist angelehnt an einen Algorithmus für Dominating Set [Nie06]. Die Vorgehensweise für Multipoint Relay unterscheidet sich von diesem dahingehend, dass wir bei einer gegebenen Multipoint-Relay-Instanz die Knoten in potenziell dominierende Knoten, die y-Knoten, und in zu dominierende Knoten, die z-Knoten, unterteilen können. Im Wesentlichen führt diese Tatsache im Algorithmus dazu, dass wir während des dynamischen Programmierens auf der Baumzerlegung eine detaillierte Fallunterscheidung durchführen müssen, um festzustellen, ob der hinzugefügte oder entfernte Knoten in einem Bag ein y- oder ein z-Knoten ist.

4.1 Lösungen für Multipoint Relay mittels dynamischen Programmierens

Wir werden im Folgenden einen Algorithmus angeben, der bei einer gegebenen Baumzerlegung eine optimale Multipoint-Relay-Menge berechnet. Wir wissen bereits, dass jeder Knoten verschiedene Zustände annehmen kann: ein

y-Knoten kann in MPR_x enthalten sein oder nicht, ein z-Knoten kann durch MPR_x dominiert sein oder nicht. Der Algorithmus verwaltet für jeden Knoten i der Baumzerlegung eine Tabelle. In dieser Tabelle werden für die Knoten des zugehörigen Bags X_i alle möglichen Kombinationen von Zuständen gespeichert und ihre Auswirkungen auf die Qualität einer dadurch festgelegten Multipoint-Relay-Menge berechnet. Dazu werden wir jedem möglichen Knotenzustand eine Farbe zuordnen und für jede mögliche Färbung aller Knoten eines Bags eine Bewertung berechnen, die unter Annahme dieser Färbung angibt, wieviele der bis zu diesem Zeitpunkt des Algorithmus' betrachteten Knoten in eine optimale Multipoint-Relay-Menge aufgenommen werden müssen. Nach Abschluss der Berechnungen lässt sich aus der Wurzel der Baumzerlegung die Größe einer optimalen Multipoint-Relay-Menge angeben und bei einer geeigneten Speicherung des Berechnungsweges diese Menge explizit angeben.

Für den Algorithmus werden einige Vorbetrachtungen nötig. Sei $G' = (V', E')$ unsere Multipoint-Relay-Instanz ohne den Knoten x und $\langle \{X_i \mid i \in I\}, T\rangle$ eine schöne Baumzerlegung für G'. Wir gehen im Folgenden stets davon aus, dass die Elemente in den Bags X_i derart geordnet sind, dass

$$X_i = \left(y_{i_1}, y_{i_2}, \ldots, y_{i_{n_i}}, z_{i_1}, z_{i_2}, \ldots, z_{i_{m_i}}\right) \tag{4.1}$$

für $i_1 \leq \cdots \leq i_{n_i}$ und $i_1 \leq \cdots \leq i_{m_i}$. Als weiteres Hilfsmittel wollen wir uns eine Verknüpfungsfunktion „$\cdot$“ zweier Tupel $(a_1, a_2, \ldots, a_r)$ und $(b_1, b_2, \ldots, b_s)$ folgendermaßen definieren:

$$(a_1, a_2, \ldots, a_r) \cdot (b_1, b_2, \ldots, b_s) := (a_1, a_2, \ldots, a_r, b_1, b_2, \ldots, b_s). \tag{4.2}$$

In Analogie soll diese Funktion auch für Mengen von Tupeln gelten, so dass für zwei beliebige Mengen A und B

$$A^r \cdot B^s := \{(a_1, \ldots, a_r, b_1, \ldots, b_s) \mid a_1, \ldots, a_r \in A \text{ und } b_1, \ldots, b_s \in B\}. \tag{4.3}$$

Färbung. Zunächst färben wir die Knoten in Abhängigkeit von ihrem Zustand. Sei $v \in V'$. Ist v ein y-Knoten, so kann er wie folgt gefärbt werden:

- „weiß“, repräsentiert durch 0, bedeutet, dass v nicht in MPR_x enthalten ist;
- „schwarz“, repräsentiert durch D, bedeutet, dass v zu MPR_x gehört, also ein dominierender Knoten ist.

Ist v ein z-Knoten, so sollen folgende Färbungen zulässig sein:

- „weiß“, repräsentiert durch 0, zeigt an, dass v noch nicht durch MPR_x dominiert wird;
- „grau“, repräsentiert durch d, gibt an, dass $v \in N(\mathrm{MPR}_x)$ liegt, also von MPR_x dominiert wird.

In den Worten der beschriebenen Knotenfarben suchen wir also für unser Problem eine Lösung, bei der alle z-Knoten grau und gleichzeitig möglichst wenige y-Knoten schwarz gefärbt sind. Im Algorithmus werden für jedes Bag X_i alle möglichen Knotenfärbungen betrachten. Einen Vektor

$$c = (c_1, c_2, \ldots, c_{n_i}, c_{n_i+1}, \ldots, c_{n_i+m_i}) \in \{0, D\}^{n_i} \cdot \{0, d\}^{m_i} \tag{4.4}$$

nennen wir eine *Färbung* für X_i. Entsprechend wird einem Knoten $y_{i_t} \in X_i$ die Farbe c_t und einem Knoten $z_{i_t} \in X_i$ die Farbe c_{n_i+t} zugewiesen. In der Färbung eines Bags kann der Fall auftreten, dass ein z-Knoten $z_{i_t} \in X_i$ grau gefärbt ist, obwohl keiner seiner y-Nachbarn in diesem Bag schwarz gefärbt ist. Eine solche Färbung nennen wir *lokal ungültig*, entsprechend alle anderen Färbungen lokal gültig. Eine lokal ungültige Färbung kann jedoch trotzdem zu einer optimalen Multipoint-Relay-Menge führen, da z_{i_t} von einem y-Knoten dominiert werden kann, der nicht in X_i liegt. Im Algorithmus müssen wir also sicherstellen, dass eine lokal ungültige Färbung nicht dazu führt, dass ein z-Knoten nicht durch MPR_x dominiert wird. Weiterhin verwenden wir für eine Färbung c die Bezeichnung

$$\#_D(c) := |\{t \in \{1, 2, \ldots, n_i\} \mid c_t = D\}| \,. \tag{4.5}$$

Die Funktion $\#_D(c)$ zählt also diejenigen y-Knoten einer Färbung c, die schwarz gefärbt sind, also in eine Multipoint-Relay-Menge aufgenommen werden, falls c eine Färbung ist, die zu einer optimalen Lösung führt.

Bewertungsfunktion. Um eine Aussage treffen zu können, wie die Färbung eines Bags die Gesamtlösung beeinflusst, führen wir für eine Färbung c eines Bags X_i eine Bewertungsfunktion

$$a_i : \{0, D\}^{n_i} \cdot \{0, d\}^{m_i} \to \mathbb{N} \cup \{+\infty\} \tag{4.6}$$

ein. Der Wert $a_i(c)$ soll dabei angeben, wieviele y-Knoten unter Berücksichtigung der Färbung c für ein Bag X_i in MPR_x aufgenommen werden. Die Funktion a_i soll dabei aber nicht nur die Knoten des aktuellen Bags

betrachten, sondern auch diejenigen, die in allen Bags des Teilbaumes mit der Wurzel i vorkommen. Dabei kann es vorkommen, dass unter Berücksichtigung einer Färbung c keine korrekte Lösung möglich ist. Ein Beispiel dafür wäre eine Färbung, in der der einzige y-Nachbar eines z-Knotens weiß gefärbt ist. Ist keine korrekte Lösung für eine Färbung c möglich, so soll $a_i(c) = +\infty$ gelten.

Monotonie. Im Algorithmus werden wir in einem Join-Knoten i unserer Baumzerlegung die Lösungen der Kinder von i zusammenführen. Um die Anzahl der Vergleiche niedrig zu halten, die bei der Berechnung der Bewertungsfunktion einer Färbung von X_i nötig werden, führen wir an dieser Stelle den Begriff der Monotonie auf unseren Färbungen ein. Dazu definieren wir uns eine Halbordnungsrelation $\preceq$ auf der Menge der Farben $\{0, d, D\}$, derart, dass $0 \preceq d$ und $f \preceq f$ für alle $f \in \{0, d, D\}$ gelte. Diese Halbordnungsrelation lässt sich auf offensichtliche Art und Weise auf die Menge der Färbungen ausdehnen. Seien

$$c = (c_1, \ldots, c_{n_i}, c_{n_i+1}, \ldots, c_{n_i+m_i}) \tag{4.7}$$

und

$$\hat{c} = (\hat{c}_1, \ldots, \hat{c}_{n_i}, \hat{c}_{n_i+1}, \ldots, \hat{c}_{n_i+m_i}) \tag{4.8}$$

zwei Färbungen. Dann gilt $c \preceq \hat{c}$ genau dann, wenn $c_t \preceq \hat{c}_t$ für alle $c_t \in c, \hat{c}_t \in \hat{c}$ mit $t \in \{1, \ldots, n_i + m_i\}$ gilt. Bei genauerer Betrachtung zeigt sich, dass $c \preceq \hat{c}$ genau dann gilt, wenn die y-Knoten durch beide Färbungen gleich gefärbt werden und in c höchstens diejenigen z-Knoten die Farbe grau erhalten, die auch in $\hat{c}$ grau gefärbt werden.

Eine Abbildung a_i von $\{0, D\}^{n_i} \cdot \{0, d\}^{m_i}$ auf $\mathbb{N} \cup \{+\infty\}$ nennen wir *monoton*, wenn aus $c, \hat{c} \in \{0, D\}^{n_i} \cdot \{0, d\}^{m_i}$ mit $c \preceq \hat{c}$ folgt, dass für die zugehörigen Funktionswerte $a_i(c) \leq a_i(\hat{c})$ gilt. Mit anderen Worten bedeutet das, dass eine monotone Bewertungsfunktion a_i eine Färbung c nicht schlechter als eine Färbung $\hat{c}$ bewertet, wenn in beiden Färbungen dieselben y-Knoten schwarz gefärbt sind und in c höchstens die z-Knoten grau gefärbt sind, die auch in $\hat{c}$ die Farbe grau zugewiesen bekommen. Im Algorithmus soll die Funktion a_i den geringsten getätigten Aufwand für eine Färbung wiedergeben. In unserem Algorithmus werden wir zeigen, dass alle Abbildungen a_i monoton sind und wir dadurch tatsächlich eine Laufzeitverbesserung erhalten.

Schritt 1 – Initialisierung. Zu Beginn legen wir für jeden Blattknoten i unserer Baumzerlegung eine Tabelle an, in der alle möglichen Färbungen des

zugehörigen Bags eingetragen werden. Anschließend berechnen wir für jede Färbung c den Wert $a_i(c)$. Dabei müssen wir beachten, dass wir lokal ungültige Färbungen mit $+\infty$ bewerten. Ausgehend von den Bewertungsfunktionen der Blätter berechnen wir später fortlaufend die Bewertungsfunktion der jeweiligen Elternknoten. In den Blattknoten entspricht der Wert einer lokal gültigen Färbung genau der Anzahl der schwarz gefärbten y-Knoten, da genau diese Knoten in eine optimale Multipoint-Relay-Menge aufgenommen werden müssen:

$$a_i(c) = \begin{cases} +\infty & \text{falls } c \text{ für } X_i \text{ lokal ungültig ist} \\ \#_D(c) & \text{sonst} \end{cases} \tag{4.9}$$

Es ist leicht einzusehen, dass die so definierten Funktionen a_i monoton sind und die Berechnungen für einen Baumknoten in $O(2^{n_i+m_i})$ Schritten durchgeführt werden können.

Schritt 2 – Dynamisches Programmieren. Wir betrachten nun Elternknoten unserer Baumzerlegung, für die bereits alle Kindknoten evaluiert wurden. Für solche Knoten i legen wir eine Tabelle an, die alle Färbungen des zugehörigen Bags X_i enthält und berechnen dann für jede Färbung c den Funktionswert $a_i(c)$. Da wir von einer schönen Baumzerlegung ausgehen, können solche Elternknoten drei verschiedene Knotentypen sein:
Forget Nodes: Ist der Elternknoten ein Forget Node, so können zwei Fälle auftreten. Zum Einen kann im Vater ein y-Knoten weniger vorkommen, und zum Anderen ein z-Knoten weniger.

- Sei i ein Forget Node mit Kind j, die Werte der Funktion a_j bereits bekannt und sei

  $$X_i = \left(y_{i_1}, y_{i_2}, \ldots, y_{i_{n_i}}, z_{i_1}, z_{i_2}, \ldots, z_{i_{m_i}}\right). \tag{4.10}$$

  O.B.d.A. sei eventuell nach einer Umsortierung der Knoten

  $$X_j = \left(y, y_{i_1}, y_{i_2}, \ldots, y_{i_{n_i}}, z_{i_1}, z_{i_2}, \ldots, z_{i_{m_i}}\right). \tag{4.11}$$

  Der Knoten y kommt also in X_j vor und in X_i nicht. Wir betrachten für die Berechnung des Funktionswertes $a_i(c)$ diejenigen Färbungen des Bags X_j, die an allen Stellen mit der Färbung c von X_i übereinstimmen und nehmen von diesen die mit dem kleinsten Funktionswert. Im Bag X_i wird unsere Multipoint-Relay-Menge weder vergrößert noch verkleinert und dementsprechend ändert sich der Funktionswert unserer Bewertungsfunktion nicht. Der Wert $a_i(c)$ ergibt sich also wie folgt:

  $$a_i(c) := \min_{f \in \{0,D\}} \{a_j((f) \cdot c)\} \tag{4.12}$$

Da hier die Färbung der Knoten und die damit verbundene Bewertung nicht verändert wird, ist a_i monoton, wenn a_j monoton ist. Die angegebene Berechnung lässt sich für einen Baumknoten leicht in $O(2^{n_i+m_i})$ ausführen.

- Sei i ein Forget Node mit Kind j, der Verlauf der Funktion a_j bekannt und sei
$$X_i = \left(y_{i_1}, y_{i_2}, \ldots, y_{i_{n_i}}, z_{i_1}, z_{i_2}, \ldots, z_{i_{m_i}}\right). \tag{4.13}$$
O.B.d.A. sei entsprechend
$$X_j = \left(y_{i_1}, y_{i_2}, \ldots, y_{i_{n_i}}, z_{i_1}, z_{i_2}, \ldots, z_{i_{m_i}}, z\right). \tag{4.14}$$
Der Knoten z kommt also in X_j vor und in X_i nicht. Damit wissen wir auf Grund der Definition der Baumzerlegung, dass z in keinem Bag oberhalb des Baumknotens i mehr auftaucht. An dieser Stelle müssen wir also sichergehen, dass z durch MPR_x dominiert wird. Das erreichen wir, indem wir für die Berechnung der Funktion a_i nur auf solche Färbungen des Bags X_j zurückgreifen, in denen der Knoten z auch tatsächlich grau, d.h. mit d, gefärbt ist. Also ergibt sich $a_i(c)$ wie folgt:
$$a_i(c) := a_j(c \cdot (d)) \tag{4.15}$$
Auch hier sieht man leicht, dass a_i monoton ist, wenn a_j monoton ist und die Berechnungen für einen Baumknoten in $O(2^{n_i+m_i})$ ausgeführt werden können.

Introduce Nodes: Auch hier können wie bei Forget Nodes wieder zwei Fälle auftreten. Entweder ist der neu auftretende Knoten ein y- oder ein z-Knoten.

- Sei i ein Introduce Node mit Kind j, die Funktionswerte der Funktion a_j bekannt und sei
$$X_j = \left(y_{j_1}, y_{j_2}, \ldots, y_{j_{n_j}}, z_{j_1}, z_{j_2}, \ldots, z_{j_{m_j}}\right). \tag{4.16}$$
O.B.d.A können wir sagen, dass
$$X_i = \left(y, y_{j_1}, y_{j_2}, \ldots, y_{j_{n_j}}, z_{j_1}, z_{j_2}, \ldots, z_{j_{m_j}}\right). \tag{4.17}$$
Weiterhin sei $N(y) \cap X_i = \{z_{p_1}, z_{p_2}, \ldots, z_{p_s}\}$ die Menge der Nachbarknoten von y in X_i. Berechnen wir nun $a_i(c)$ für eine Färbung c, in der y schwarz gefärbt ist, so wollen wir auf eine Färbung von X_j

zurückgreifen, in der alle Knoten $z_{p_1}, z_{p_2}, \ldots, z_{p_s}$ weiß gefärbt sind. Da y ohnehin die genannten z-Knoten dominiert, stellen wir mit einer derartigen Färbung aufgrund der Monotonie sicher, aus dem Bag X_j die mit dem kleinsten zugehörigen Funktionswert zu wählen. Dazu definieren wir uns für eine Färbung

$$c = \left(c_1, \ldots, c_{n_j}, c_{n_j+1}, \ldots, c_{n_j+m_j}\right) \tag{4.18}$$

des Bags X_j folgende Funktion ϕ mit

$$\phi(c) := \left(c_1, \ldots, c_{n_j}, \hat{c}_{n_j+1}, \ldots, \hat{c}_{n_j+m_j}\right) \tag{4.19}$$

derart, dass

$$\hat{c}_t := \begin{cases} 0 & \text{falls } t \in \{p_1, \ldots, p_s\} \\ c_t & \text{sonst.} \end{cases} \tag{4.20}$$

Wie oben beschrieben berechnet die Funktion ϕ für ein Bag X_i mit dem neuen Knoten y und der Färbung c' diejenige Färbung c von X_j, bei der alle in X_i zu y benachbarten Knoten noch nicht dominiert werden. Nach der Monotonie der Bewertungsfunktion ist das gerade die günstigste Färbung, die wir betrachten müssen, um $a_i(c')$ für eine Färbung c' für X_i zu berechnen:

$$a_i((0) \cdot c) := a_j(c) \tag{4.21}$$

$$a_i((D) \cdot c) := a_j(\phi(c)) + 1. \tag{4.22}$$

Ist der neue Knoten y für eine bestimmte Färbung kein Knoten aus unserer Multipoint-Relay-Menge, so verändert sich in der Bewertung nichts, denn es wird MPR_x kein weiterer Knoten hinzugefügt. Ist dieser Knoten hingegen ein dominierender Knoten – also aus MPR_x – so addieren wir zur günstigsten dazu passenden Färbung eins dazu, nämlich den neu hinzugekommenen Knoten y, der schwarz gefärbt wurde und somit die Menge MPR_x um eins vergrößert. Diese günstigste dazu passende Färbung berechnet uns auf Grund der Monotonie aber gerade die Funktion ϕ.

Dadurch sieht man auch leicht, dass a_i monton ist, wenn a_j die Monotonieeigenschaft besitzt. Weiterhin lassen sich diese Berechnungen für einen solchen Baumknoten in $O(2^{n_i+m_i})$ durchführen.

- Sei i ein Introduce Node mit Kind j, der Verlauf der Funktion a_j bekannt und sei

$$X_j = \left(y_{j_1}, y_{j_2}, \ldots, y_{j_{n_j}}, z_{j_1}, z_{j_2}, \ldots, z_{j_{m_j}}\right). \tag{4.23}$$

O.B.d.A können wir sagen, dass

$$X_i = \left(y_{j_1}, y_{j_2}, \ldots, y_{j_{n_j}}, z_{j_1}, z_{j_2}, \ldots, z_{j_{m_j}}, z\right). \tag{4.24}$$

Ist z in der neuen Färbung nicht dominiert, so verhält sich die Bewertung dieser Färbung durch a_i genauso wie in X_j. Ist z jedoch in der Färbung für X_i dominiert, so können zwei Fälle auftreten. Hat z in X_i einen y-Knoten als Nachbarn, der schwarz gefärbt ist, so wird z bereits durch MPR_x dominiert und wir haben keinen weiteren Aufwand zu tätigen. Besitzt z jedoch keinen schwarz gefärbten Nachbarn, so könnte eine unkorrekte Lösung entstehen, wenn wir mit dieser Färbung weiterrechnen. Deshalb bewerten wir sie mit $+\infty$. Ist nun c eine Färbung für das Bag X_j, dann ergibt sich a_i so:

$$a_i(c \cdot (0)) := a_j(c) \tag{4.25}$$

$$a_i(c \cdot (d)) := \begin{cases} a_j(c) & \text{falls } z \text{ einen Nachbarn} \\ & y_{i_t} \in X_i \text{ mit } c_t = D \text{ hat;} \\ +\infty & \text{sonst.} \end{cases} \tag{4.26}$$

Hier sieht man leicht, dass a_i monoton ist. Die Berechnungen für einen solchen Baumknoten können auch hier wieder in $O(2^{n_i+m_i})$ ausgeführt werden.

Join Nodes: Sei i ein Join Node mit den Kindern j und k, die Funktionswerte ihrer Bewertungsfunktionen a_j und a_k bekannt und nach Definition

$$X_i = X_j = X_k = \left(y_{i_1}, y_{i_2}, \ldots, y_{i_{n_i}}, z_{i_1}, z_{i_2}, \ldots, z_{i_{m_i}}\right). \tag{4.27}$$

Wie berechnet sich der Wert $a_i(c)$ für eine Färbung c eines Join-Knotens? Wir wollen insgesamt erreichen, dass möglichst wenige y-Knoten schwarz gefärbt sind, während alle z-Knoten grau gefärbt sind. Dazu betrachten wir bei einer Färbung c die schwarz gefärbten y-Knoten und prüfen, welche z-Knoten durch diese im rechten und linken Teilbaum dominiert werden. Wir müssen also für eine Färbung c zwei Färbungen der Kinderknoten finden, die zu dieser Färbung „passen“. Dies führt zunächst zu folgender Definition.

Definition 4.1. Sei c gemäß (4.4) eine Färbung für X_i und analog $\hat{c}$ eine Färbung für X_j und $\tilde{c}$ eine Färbung für X_k. Wir sagen, zwei Färbungen $\hat{c}$ und $\tilde{c}$ *rechtfertigen* c, wenn gilt:

1. $\forall t \in \{1, \ldots, n_i\} : c_t = \hat{c}_t = \tilde{c}_t$ und

2. $\forall t \in \{n_i + 1, \ldots, n_i + m_i\} : (c_t = d) \rightarrow ((\hat{c}_t = d) \vee (\tilde{c}_t = d))$.

Mit anderen Worten bedeutet das, dass zwei Färbungen der Kinderknoten eine Färbung c von X_i rechtfertigen, wenn die Farben der y-Knoten in c, $\hat{c}$ und $\tilde{c}$ übereinstimmen und ein z-Knoten in c höchstens dann die Farbe grau erhält, wenn dieser Knoten auch in wenigstens einer der Färbungen $\hat{c}$ oder $\tilde{c}$ grau gefärbt wird. Betrachten wir nun zwei Färbungen $\hat{c}$ und $\tilde{c}$, die c rechtfertigen. O.B.d.A. sei $\hat{c}$ eine Färbung von X_j und $\tilde{c}$ eine Färbung von X_k. Der Wert $a_j(\hat{c})$ ist in dem Teilbaum mit der Wurzel j bereits durch den Algorithmus berechnet und gibt die bis dahin schwarz gefärbten y-Knoten an. Analoges gilt für den Wert $a_k(\tilde{c})$. Aufgrund der Definition einer schönen Baumzerlegung müssen genau alle diejenigen Knoten, die in den Bags beider Teilbäume vorkommen, in den Bags X_j und X_k enthalten sein. Wenn wir die Werte $a_j(\hat{c})$ und $a_k(\tilde{c})$ addieren, so zählen wir daher genau die schwarz gefärbten y-Knoten doppelt, die in den beiden Bags X_j und X_k vorkommen, müssen diese also noch einmal abziehen. Da es viele Paare von Färbungen geben kann, die eine Färbung c für X_i rechtfertigen, müssen wir das Paar finden, dass die kleinste Gesamtsumme aufweist. Demzufolge berechnet sich $a_i(c)$ wie folgt:

$$a_i(c) = \min\{a_j(\hat{c}) + a_k(\tilde{c}) \mid \hat{c} \text{ und } \tilde{c} \text{ rechtfertigen } c\} - \#_D(c). \qquad (4.28)$$

Mit dieser Definition ist a_i monoton, wenn a_j und a_k monoton sind.

Bei oberflächlicher Betrachtung kann in diesem Fall größerer Aufwand als in den vorher betrachteten Knotentypen entstehen. Im schlimmsten Fall müssen $O(2^{n_i}4^{m_i})$ Vergleiche getätigt werden, da wir alle möglichen Färbungen von z-Knoten miteinander kombinieren und diese Ergebnisse prüfen müssen. An dieser Stelle können wir aber das Wissen um die Monotonie ausnutzen, um die Laufzeit dieses Schrittes zu verbessern. Viele dieser Vergleiche können nämlich gespart werden, wenn man für zwei Färbungen $\hat{c}$ und $\tilde{c}$, die c rechtfertigen, statt in der obigen Definition den zweiten Punkt durch folgende Forderung ersetzt:

2'. $\forall t \in \{1, \ldots, m_i\} : (c^z_t = d) \rightarrow (((\hat{c}^y_t = d) \vee (\tilde{c}^y_t = d)) \wedge (\hat{c}^y_t \neq \tilde{c}^y_t))$.

Da wir in der Berechnung des Funktionswertes $a_i(c)$ ohnehin minimieren, können wir das Wissen um die Monotonie der Färbungen ausnutzen und zu einer Färbung $\hat{c}$ durch die neue Forderung direkt die günstigste passende Färbung $\tilde{c}$ so angeben, dass $\hat{c}$ und $\tilde{c}$ die Färbung c rechtfertigen. Dies bedeutet aber, dass für eine Färbung c und einem darin grau gefärbten z-Knoten z_t lediglich entschieden werden muss, in welchem der beiden Färbungen $\hat{c}$ und $\tilde{c}$

der Knoten z_t grau gefärbt wird, in der anderen Färbung entsprechend weiß. So existieren noch 2^{m_i} zu testende Möglichkeiten und der Schritt lässt sich für einen derartigen Baumknoten in $O(2^{n_i+m_i})$ ausführen.

Schritt 3 – Auswertung. Sei nun r die Wurzel unserer Baumzerlegung T mit n_r y-Knoten und m_r z-Knoten. Wir können die Größe der kleinsten Multipoint-Relay-Menge aus der Kenntnis von a_r angeben:

$$|\mathrm{MPR}_x| = \min\{a_r(c) \mid c \in \{0,1\}^{n_r} \cdot \{d\}^{m_r}\}. \tag{4.29}$$

Wir betrachten dazu also nur noch die Färbungen, in denen jeder z-Knoten grau gefärbt ist. So wird sichergestellt, dass auch jeder z-Knoten von MPR_x dominiert wird. Wurden zusätzlich noch die Informationen gespeichert, aus welchen Färbungen der Nachkommen sich die jeweiligen Bewertungsfunktionen zusammensetzen, lässt sich durch einfaches Zurückverfolgen der Berechnung eine optimale Multipoint-Relay-Menge angeben.

Lemma 4.2 (Korrektheit)**.** *Der angegebene Algorithmus berechnet eine optimale Multipoint-Relay-Menge.*

Beweis. Wir betrachten dazu noch einmal die Definition einer Baumzerlegung. Da jeder Knoten unseres Graphen in wenigstens einem Bag der Baumzerlegung vorkommt, wird jedem Knoten eine Farbe zugewiesen. Dann stellen wir sowohl während der Initialisierung als auch während des Prozesses des dynamischen Programmierens sicher, dass ein z-Knoten nur dann die Farbe d zugewiesen bekommt, wenn er im aktuellen oder einem schon besuchten Bag einen mit D gefärbten y-Nachbarn besitzt. Und aus der dritten Eigenschaft einer Baumzerlegung und den Kommentaren in Schritt 2 können wir folgern, dass die Berechnungen der Bewertungsfunktion konsistent in Bezug auf die bereits früher berechneten Funktionen sind. □

Lemma 4.3 (Laufzeit)**.** *Der angegebene Algorithmus arbeitet für eine Baumzerlegung $\langle\{X_i \mid i \in I\}, T\rangle$ der Weite ω in $O(2^\omega\,|I|)$ Schritten.*

Beweis. Wir haben bereits für jeden Schritt einzeln eine Laufzeitbetrachtung durchgeführt. Mit dem Wissen, dass die Anzahl der Knoten $m_i + n_i$ eines jeden Bags X_i immer kleiner gleich der Baumweite ω ist, wissen wir also, dass wir für jeden Knoten i unseres Baumes T höchstens $O(2^\omega)$ Schritte benötigen, um die zugehörige Bewertungsfunktion a_i zu berechnen. Für den gesamten Baum benötigen wir demzufolge $O(2^\omega\,|I|)$ Schritte. □

Satz 4.4. *Sei G eine Instanz von* Multipoint Relay *und für $G' := G - \{x\}$ eine schöne Baumzerlegung $\langle \{X_i \mid i \in I\}, T\rangle$ mit der Weite ω gegeben. Dann lässt sich eine optimale Multipoint-Relay-Menge MPR_x in $O(2^\omega |I|)$ berechnen.*

Die Lemmas 4.2 und 4.3 garantieren die Gültigkeit des Satzes 4.4.

4.2 Baumzerlegung einer planaren Multipoint-Relay-Instanz

Im letzten Abschnitt haben wir einen effizienten Algorithmus angegeben, der bei gegebener Baumzerlegung einer Multipoint-Relay-Instanz eine optimale Lösung liefert. Dies wirft jedoch die Frage auf, wie sich eine solche Baumzerlegung mit kleiner Baumweite finden lässt.

Wir werden später zeigen, dass die Außerplanarität einer Multipoint-Relay-Instanz beschränkt ist. Deshalb geben wir im Folgenden einen Algorithmus mit Ergebnissen aus [ABF+02] und [Bod98] an, der für einen r-außerplanaren Graphen eine Baumzerlegung berechnet und nutzen diesen dann für unser Problem.

Satz 4.5. *Sei $G = (V, E)$ ein r-außerplanarer Graph mit einer r-außerplanaren Einbettung. Dann lässt sich eine Baumzerlegung $\langle \{X_i \mid i \in I\}, T\rangle$ mit einer Baumweite $\omega \leq 3r - 1$ in $O(r |V|)$ finden.*

Wir beweisen den Satz, indem wir in den folgenden Abschnitten einen Algorithmus angeben, wie für r-außerplanare Graphen eine Baumzerlegung effizient gefunden werden kann. Die Konstruktion der gewünschten Baumzerlegung findet dabei in mehreren Schritten statt. Zuerst bestimmen wir die Schichten des gegebenen Graphen G. Als zweites berechnen wir einen Graphen H, der G als Minor enthält und dessen größter Knotengrad 3 ist. Von diesem Graphen H bestimmen wir einen für uns geeigneten aufspannenden Wald, dessen sogenannte „Edge-Remember"- und „Vertex-Remember"-Zahlen klein sind. Mit einem derartigen aufspannenden Wald ist es uns möglich eine Baumzerlegung von H mit beschränkter Baumweite zu konstruieren, die wir im Anschluss in eine Baumzerlegung von G transformieren.

Schritt 1 – Bestimmen der Schichten. Im ersten Schritt wollen wir vorbereitend die Knoten und Flächen der gegebenen Einbettung in Schichten einordnen. Sei G ein r-außerplanarer Graph mit einer r-außerplanaren

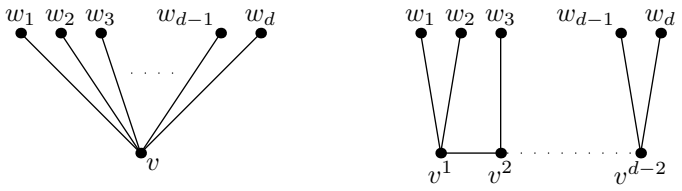


Abbildung 4.1: Wir ersetzen jeden Knoten v mit Grad $d \geq 4$ in G, wie v in (i), durch einen Pfad mit $d-2$ neuen Knoten in H, wie in (ii) zu sehen.

Einbettung. Zuerst betrachten wir die Flächen der Einbettung unseres Graphen. Wir konstruieren den dualen Graph G^* und sagen dann, dass eine Fläche f in die Schicht L^*_{i+1} gehört, wenn ihr minimaler Abstand im dualen Graphen zur äußeren Fläche i Kanten beträgt. Offensichtlich kann es Flächen geben, die in L^*_{r+1} liegen. Es können jedoch keine Schichten L^*_s mit $s > r+1$ existieren.

Weiterhin gehört ein Knoten $v \in V$ aus dem Graphen G in die Schicht L_i, wenn i der kleinste Index derart ist, dass v zu einer Fläche f_v aus L^*_i benachbart ist.

Schritt 2 – Einbetten des Graphs in einen Graph mit Grad 3. In diesem Schritt wollen wir einen r'-außerplanaren Graphen $H = (V_H, E_H)$ mit $r' \leq r$ konstruieren, dessen größter Knotengrad 3 ist und der G als Minor enthält. Dazu ersetzen wir jeden Knoten $v \in V$ mit $\deg_G(v) \geq 4$ durch einen Pfad mit $d-2$ Knoten vom Grad 3, wie in Abb. 4.1 gezeigt. Wir müssen diese Konstruktion so durchführen, dass der auf diese Weise entstehende Graph H wie gefordert r'-außerplanar bleibt. Dazu bestimmen wir für einen solchen Knoten v aus der Schicht L_i eine zugehörige Fläche f_v aus L^*_i. Da v auf dem Kreis liegt, der f_v begrenzt, finden wir zwei Kanten $\{v, x\}$ und $\{v, y\}$, die an f_v grenzen. Dann konstruieren wir den eben beschriebenen Pfad so, dass x in Abb. 4.1 die Rolle von w_1 und y die Rolle von w_d übernimmt. So ist sichergestellt, dass jeder der $d-2$ Knoten v^i des neuen Pfades an $f_v \in L^*_i$ liegt. Nun sei H der Graph, den wir aus G erhalten, wenn wir alle Knoten mit einem Grad $d \geq 4$ in G auf die beschriebene Art und Weise ersetzt haben.

Da wir durch diese Pfadkonstruktion keinerlei neue Flächen erzeugen, haben G und H dieselbe Menge von Flächen, mit anderen Worten besitzen die dualen Graphen G^* und H^* dieselbe Knotenmenge. Weiterhin sind

Flächen, die in G benachbart sind, auch in H benachbart, der Graph H^* kann also aus G^* erzeugt werden, indem höchstens einige Kanten zu G^* hinzugefügt werden. Das bedeutet aber, dass G^* ein Teilgraph von H^* ist. Daraus folgt, dass die Entfernung eines Knotens in H^* zur äußeren Fläche keinesfalls zunehmen kann. Da so aber die Schichten der Flächen und damit die Schichten der Knoten definiert wurden, ist r' durch r nach oben beschränkt.

Schritt 3 – Passenden aufspannenden Wald des Grad-3-Graphs konstruieren. Wir wollen nun für den eben konstruierten Graphen H einen geeigneten aufspannenden Wald konstruieren, aus dem wir später die Baumzerlegung berechnen. In diesem Schritt gehen wir induktiv vor, indem wir bei der innersten Schicht beginnend unseren aufspannenden Wald geeignet vergrößern. Offensichtlich gelten die folgenden Fakten: Wenn wir für ein beliebiges $s > 1$ von einem s-außerplanaren Graphen mit Knotengrad höchstens 3 alle diejenigen Kanten entfernen, die an die äußere Fläche grenzen, erhalten wir einen $(s-1)$-außerplanaren Graphen. Wenn wir von einem außerplanaren Graphen mit Knotengrad nicht größer als 3 alle Kanten an der äußeren Fläche entfernen, so erhalten wir einen Wald.

Wir können so auch die Kanten in Schichten $E_1, \ldots, E_{r'+1}$ einteilen, wobei E_1 die Kanten an der äußeren Fläche enthält und E_i diejenigen Kanten, die an der äußeren Fläche liegen, sobald $E_1 \cup \cdots \cup E_{i-1}$ aus H entfernt wurden. Auch diese Mengen lassen sich mit dem dualen Graphen in $O(|V_H|)$ berechnen. Den Teilgraphen, der durch $E_i \cup \cdots \cup E_{r'+1}$ definiert ist, bezeichnen wir auch mit H_i.

Nun können wir unseren aufspannenden Wald konstruieren. Wir beginnen mit dem Wald $T_{r'+1}$, der aus den Kanten $E_{r'+1}$ besteht. Da diese die inneren Kanten eines außerplanaren Graphen mit dem Höchstknotengrad 3 sind, enthält $T_{r'+1}$ keine Kreise.

Aus einem gegebenen $T_i, 1 < i \leq r' + 1$, konstruieren wir T_{i-1} folgendermaßen: Wir fügen zu T_i so lange Kanten aus E_{i-1} hinzu, bis einerseits keine Kreise entstehen und andererseits das Hinzunehmen jeder beliebigen weiteren Kante aus E_{i-1} einen Kreis in T_i erzeugen würde. Auf diese Weise entsteht für jedes i, $1 \leq i \leq r' + 1$, ein aufspannender Wald für H_i. Ein solcher Vorgang benötigt hierbei höchstens $O(|V_H|)$ Schritte. Entsprechend benötigen wir für die Konstruktion von T_1 nicht mehr als $O(r\,|V_H|)$ Schritte.

Anzumerken ist hierbei, dass sich ein aufspannender Wald für einen beliebigen Graphen in $O(|V|)$ leicht mit einer Tiefensuche finden lässt. Wir benötigen jedoch einen auf diese spezielle Weise konstruierten Wald, da die-

ser kleine sogenannte Vertex-Remember- und Edge-Remember-Zahlen besitzt, die wir zunächst einmal definieren wollen.

Definition 4.6. Sei $G = (V, E)$ ein Graph und $T = (V, F)$ ein aufspannender Wald von G. Dann können wir jeder Kante $e = \{v, w\} \in E \setminus F$ einen eindeutig bestimmten *Basiskreis* zuordnen, der aus e und dem Pfad von v nach w in T besteht.

Damit sei die *Vertex-Remember-Zahl* $\mathrm{vr}(G, T, v)$ eines Knotens $v \in V$ bezüglich T die Anzahl aller Basiskreise, die v enthalten. Die Vertex-Remember-Zahl $\mathrm{vr}(G, T)$ von G bezüglich T sei $\mathrm{vr}(G, T) = \max_{v \in V}\{\mathrm{vr}(G, T, v)\}$.

Analog sei die *Edge-Remember-Zahl* $\mathrm{er}(G, T, e)$ einer Kante $e \in F$ bezüglich T die Anzahl aller Basiskreise, die e enthalten. Die Edge-Remember-Zahl $\mathrm{er}(G, T)$ von G bezüglich T sei dann $\mathrm{er}(G, T) = \max_{e \in F}\{\mathrm{er}(G, T, e)\}$.

Direkt im Anschluss an den Algorithmus werden wir zeigen, dass unser aufspannender Wald gerade so konstruiert wurde, dass die daraus zu konstruierende Baumzerlegung eine Baumweite ω vorweist, die durch

$$\max\{\mathrm{vr}(H, T_1), \mathrm{er}(H, T_1) + 1\} \leq 3r' - 1 \leq 3r - 1 \tag{4.30}$$

beschränkt ist. Da dieses Wissen nicht zum eigentlichen Algorithmus beiträgt, sei hier auf das Ende dieses Kapitels verwiesen, wo wir einen Beweis für diesen Fakt darlegen.

Schritt 4 – Baumzerlegung aus dem Spannbaum konstruieren. In diesem Schritt wollen wir zeigen, wie sich aus einem gegebenen aufspannenden Wald eine Baumzerlegung für H konstruieren lässt.

Satz 4.7. *Sei $T = (V, F)$ ein aufspannender Wald für den Graphen $G = (V, E)$. Dann lässt sich eine Baumzerlegung mit $O(|V|)$ Knoten und der Baumweite höchstens $\max\{\mathrm{vr}(G, T), \mathrm{er}(G, T)+1\}$ in $O(\mathrm{vr}(G, T)\cdot|V|)$ Schritten bestimmen.*

Beweis. Wir wollen nun für G eine Baumzerlegung $\langle\{X_i \mid i \in I\}, T'\rangle$ konstruieren. Dazu sei $T' = (I, F')$ mit $I = V \cup F$ und $F' = \{\{v, e\} \mid v \in V, e \in F, \exists w \in V : e = \{v, w\}\}$. Die Bags X_i bestimmen sich wie folgt. Für jeden Knoten $v \in V$ fügen wir einen der Knoten v und w, o.B.d.A. v, der Menge X_v hinzu. Für jede Kante $e = \{v, w\} \in F$ fügen wir v und w dem Bag X_e hinzu. Nun fügen wir noch für jede Kante $e = \{v, w\} \in E \setminus F$ den Knoten v zu allen Mengen X_u und X_f hinzu, für die $u \in V$ oder $f \in F$ auf dem Pfad von v nach w in T liegt.

Mit üblichen Graphenalgorithmen können wir den Pfad zwischen zwei Knoten in einem Baum in polynomialer Zeit in der Länge des Pfades finden. Da jeder Knoten des Baumes T zu höchstens $\mathrm{vr}(G,T)$ solchen Pfaden gehört, ist die Laufzeit durch $O(\mathrm{vr}(G,T)\cdot|V|)$ beschränkt.

Wir überprüfen leicht, dass wir so tatsächlich eine Baumzerlegung konstruiert haben:

1. $\bigcup_{i\in I} X_i = V$, da einerseits für jeden Knoten $v \in V$ ein Bag X_v konstruiert wurde und andererseits als Elemente der Bags nur Knoten aus V genutzt werden.

2. Sei $e = \{v, w\}$ eine beliebige Kante des Graphen. Dann kommen v und w gleichzeitig im Bag X_e vor.

3. Seien $i, j, k \in I$ mit j liegt auf dem Pfad zwischen i und k in T und sei $v \in X_i \cap X_k$. Dann wurde v zu X_i und X_k hinzugefügt, weil eine Kante $e = \{v, w\}$ existiert, dass sowohl X_i und X_k auf dem Pfad von v nach w in T liegen. Dann liegt aber auch j auf dem Pfad von v nach w in T und somit ist $v \in X_j$.

Für jeden Knoten $v \in V$ ist die Größe der Menge X_v durch $\mathrm{vr}(G,T)+1$ nach oben beschränkt, für jede Kante $e \in F$ entsprechend X_e durch $\mathrm{er}(G,T)+2$. Damit ist die Baumweite durch $\max\{\mathrm{vr}(G,T_1), \mathrm{er}(G,T_1)+1\}$ nach oben beschränkt. □

Schritt 5 – Minorenoperationen rückgängig machen. Mit dem eben angegebenen konstruktiven Beweis können wir eine Baumzerlegung für H konstruieren, deren Weite wegen (4.30) durch $3r-1$ beschränkt ist. Nun benötigen wir jedoch eine Baumzerlegung für G. Diese erhalten wir, indem wir in jedem Bag X_j alle auftreten der Knoten v^i durch den ursprünglichen Knoten v ersetzen. Dies ist in linearer Zeit in der Größe der Bags möglich, also in $O(r\cdot|V|)$.

Beschränkung der Edge- und Vertex-Remember-Zahl. In diesem Abschnitt wollen wir uns abschließend mit den Vertex-Remember- und Edge-Remember-Zahlen auseinandersetzen, die es uns ermöglichen, dass wir die Baumweite des zu konstruierenden Baumes beschränken können. Wir wollen hier also zeigen, dass der im dritten Schritt konstruierte passende Spannbaum auch tatsächlich den in (4.30) angegebenen Bedingungen genügt.

Satz 4.8. *Sei $G = (V, E)$ ein planarer Graph mit gegebener planarer Einbettung und $G' = (V, E')$ derjenige Graph, der aus G entsteht, wenn man alle*

Kanten entfernt, die an die äußere Fläche grenzen. Sei $T' = (V, F')$ ein aufspannender Wald von G'. Dann existiert ein aufspannender Wald $T = (V, F)$ von G so, dass $\mathrm{er}(G, T) \leq \mathrm{er}(G', T') + 2$ *und* $\mathrm{vr}(G, T) \leq \mathrm{vr}(G', T') + \deg(G)$.

Beweis. Betrachten wir zunächst den Graph $K = (V, (E \setminus E') \cup F')$, also den Graph der entsteht, wenn wir zu F' alle Kanten hinzufügen, die in G an der äußeren Fläche liegen. Sei nun $T = (V, F)$ ein aufspannender Wald von K derart, dass $T' \subseteq T$, dass also T aus T' durch Hinzufügen von Kanten aus $E \setminus E'$ entsteht. Wir sehen leicht, dass T ein aufspannender Wald für G ist. Weiterhin begrenzt jeder Basiskreis in K bezüglich T eine innere Fläche in K. Da nun jede Kante zu höchstens zwei und jeder Knoten zu höchstens $\deg(G)$ inneren Flächen benachbart ist, muss gelten $\mathrm{er}(K, T) \leq 2$ und $\mathrm{vr}(K, T) \leq \deg(G)$. Da T ein aufspannender Wald für G ist und jeder Basiskreis bezüglich T in G ein Basiskreis in K oder ein Basiskreis in G' bezüglich T ist, folgt

$$\mathrm{er}(G, T) \leq \mathrm{er}(G', T') + \mathrm{er}(K, T) \leq \mathrm{er}(G', T') + 2 \tag{4.31}$$

und

$$\mathrm{vr}(G, T) \leq \mathrm{vr}(G', T') + \mathrm{vr}(K, T) \leq \mathrm{vr}(G', T') + \deg(G). \tag{4.32}$$

□

Satz 4.9. *Sei $G = (V, E)$ ein r-außerplanarer Graph mit $deg(G) \leq 3$. Dann gibt es für G einen aufspannenden Wald $T = (V, F)$ mit* $\mathrm{er}(G, T) \leq 2r$ *und* $\mathrm{vr}(G, T) \leq 3r - 1$.

Beweis. Die Grundstruktur dieses Beweises ist eine Induktion über r. Zuerst betrachen wir als Induktionsanfang den Fall $r = 1$. Sei also G ein außerplanarer Graph. Entfernen wir aus G alle Kanten, die an der äußeren Fläche liegen, so erhalten wir einen Wald $T' = (V, F')$. Offensichtlich gilt $\mathrm{er}(T', T') = \mathrm{vr}(T', T') = 0$. Weiterhin ist klar, dass jeder Knoten zu höchstens zwei inneren Flächen benachbart ist. In Analogie zum Beweis des vorherigen Satzes folgt damit die Aussage für außerplanare Graphen.

Die Aussage dieses Satzes für r-außerplanare Graphen folgt dann mittels Induktion über r. Den Fall $r = 1$ haben wir bereits gezeigt. Es gelte also die Aussage für $(r-1)$-außerplanare Graphen mit $r \geq 2$. Wenn wir nun von einem r-außerplanaren Graphen alle Kanten entfernen, die an der äußeren Fläche liegen, so erhalten wir einen $(r-1)$-außerplanaren Graphen. Mit einer analogen Argumentation wie im Induktionsanfang folgt die Aussage des Satzes. □

Zusammenfassung. Die Schritte 1 bis 5 garantieren uns die Korrektheit von Satz 4.5. Damit ist es uns nun möglich eine Baumzerlegung für einen r-außerplanaren Graphen zu konstruieren.

4.3 Zusammenfassung

Nun genügt uns folgendes leicht zu sehende Lemma, um eine Baumzerlegung für eine beliebige planare Multipoint-Relay-Instanz zu bestimmen und dann mittels des in diesem Kapitel vorgestellten Ansatzes über dynamisches Programmieren eine optimale Multipoint-Relay-Menge zu berechnen.

Lemma 4.10. *Jede planare MPR-Instanz ist höchstens 3-außerplanar.*

Beweis. Betrachten wir nun eine planare Multipoint-Relay-Instanz G. Es ist leicht zu sehen, dass ein derarig strukturierter Graph 3-außerplanar ist, indem man eine Einbettung betrachtet, bei der x an die äußere Fläche grenzt. Nachdem x gelöscht ist, liegt nun jeder y-Knoten an der äußeren Fläche und nach dem entfernen aller y-Knoten muss dann auch jeder z-Knoten an der äußeren Fläche liegen. Dies garantiert aber genau die 3-Außerplanarität. □

Für den angegebenen Algorithmus über dynamisches Programmieren wird der Knoten x nicht verwendet. Wenn wir also eine Graphinstanz als Eingabe für diesen Algorithmus nutzen, können wir den Knoten x bereits vorher löschen und haben so einen 2-außerplanaren Graphen, den wir als Eingabe nutzen können.

Also können wir abschließend festhalten:

Satz 4.11. *Eine optimale MPR-Menge lässt sich für beliebige planare Probleminstanzen über dynamisches Programmieren in $O(2^5 \cdot n) = O(n)$ Schritten finden.*

Kapitel 5

MPR mit Distanz-d-Informationen

In den vorherigen Kapiteln haben wir das Problem Multipoint Relay auf planaren Graphen ausführlich betrachtet. Wir haben gezeigt, wie sich die Anzahl der sendenden Knoten reduzieren lässt, sobald ein Knoten seine Nachbarn und deren Nachbarn kennt. Dies legt nun aber die Frage nahe, was passiert, sobald ein Knoten auch seine Distanz-3-Nachbarn kennt. Von dieser Frage geleitet können wir das Problem derart modifizieren, dass nun ein Knoten nicht mehr nur seine Distanz-2-Nachbarschaft kennt, sondern alle Nachbarn bis zu einer bestimmten Entfernung d. Die Knoten, die zu x eine Distanz genau d besitzen, wollen wir im Folgenden auch als Knoten der *Schicht* d bezeichnen. Auch hier wollen wir wieder planare Graphen zugrunde legen.

5.1 Problembeschreibung und optimale Lösungen

Wir ändern für dieses Kapitel das Problem derart ab, dass der Knoten x nun alle Knoten und Kanten mit dem Abstand $\leq d$ kennt. Dies lässt sich in einem mobilen Netzwerk mittels d-maligem Austausch von Kontrollmitteilungen bewirken. Nun wirft diese Modifikation sofort einige Fragen auf.

- Wie verändert sich die Komplexität des Problems?
- Betrachten wir die Anzahl aller sendenden Knoten im gesamten Netzwerk. Welche Unterschiede weist diese Anzahl auf, wenn wir im Netzwerk einmal den ursprünglichen MPR-Ansatz und einmal die hier beschriebene Problemmodifikation als Berechnungsgrundlage nutzen?

- Welche Vor- und Nachteile hat dieses Modell?
- Was ist eigentlich eine optimale Lösung für dieses Problem?

Bevor wir beginnen dieses allgemeinere Problem zu analysieren, müssen wir uns damit auseinandersetzen, welche Forderungen wir an die Lösung stellen.

Was ist eine optimale Lösung? Diese Frage ist nicht so direkt zu beantworten, wie es zunächst scheint. Wir wollen hier zwei Möglichkeiten einer optimalen Lösung vorschlagen und später auf eine davon näher eingehen.

Eine erste Forderung wäre die, dass der Knoten x sicherstellen muss, dass jeder für x „sichtbare“ Knoten die Information erhält. Diese Forderung scheint zunächst keinerlei Unterschiede zum ursprünglichen Problem aufzuweisen. Doch durch die Definition des Multipoint-Relay-Problems war implizit gegeben, dass jeder z-Knoten nach zwei Sendeschritten die Nachricht erhielt. Betrachten wir nun noch einmal die oben angeführte Forderung. Da keine Einschränkungen über die Länge des Sendeweges gemacht wurden, kann der Fall auftreten, dass ein Knoten der Schicht i die Information von einem Knoten der Schicht $i + 1$ erhält. Wenn wir annehmen, dass Sendungen sehr viel schneller stattfinden als sich die Topologie des Netzwerkes verändert, ist eine solche Lösungsbetrachtung durchaus sinnvoll, kann im Extremfall jedoch zu unnötig langen Kommunikationswegen führen. Außerdem fällt auf, dass eine Suche nach einer solchen Lösung sehr eng verbunden ist mit der Suche nach einem sogenannten „Connected Dominating Set“. Die Suche nach einem Connected Dominating Set wurde bereits viel diskutiert [GK98, BCOP03, RDJ+04], weshalb wir uns mit dieser Forderung an eine Lösung nicht weiter auseinandersetzen wollen.

Eine andere Forderung an eine optimale Lösung wäre, dass ein Knoten der Schicht i nach spätestens i Sendeschritten die Nachricht tatsächlich erhält. Das bedeutet dann, dass eine gewisse Anzahl von Knoten der Schicht $i - 1$ sendet und dadurch alle Knoten der Schicht i erreicht werden. Diese Art der optimalen Lösungen wollen wir im Folgenden zu Grunde legen.

Dafür wird es nötig eine weitere Sorte Knoten einzuführen. Wir berechnen ausgehend vom Sender x Knoten, die weitersenden sollen. Wenn nun jeder der weitersendenden Knoten wieder seine Multipoint Relays berechnet, haben wir nichts gewonnen. Wir müssen neue d-Hop-MPR-Berechnungen nur für Knoten der vorletzten Schicht – o.B.d.A. $(d-1)$ – durchführen. Alle Knoten der Schicht d werden von dieser vorletzten Schicht benachrichtigt, demzufolge ist es auch nur von diesen Knoten erforderlich, weitere Berech-

nungen anzustellen. All diejenigen Knoten, die von x die Mitteilung bekommen weiterzusenden, jedoch keine Berechnungen selbst anstellen sollen, nennen wir im folgenden *Gateways* und die Knoten, die MPR-Berechnungen durchführen sollen *echte Multipoint-Relay-Knoten*, die Gesamtheit beider Knotenarten bezeichnen wir wie bisher als *Multipoint-Relay-Knoten*.

Somit lässt sich das Problem genau beschreiben.

d-Hop Multipoint Relay
Eingabe: Ein Graph, ein Knoten x des Graphen, die Distanz d.
Problem: Angabe einer kleinstmöglichen Menge von Gateway- und echten Multipoint-Relay-Knoten.

Das ursprüngliche Problem stellt einen Spezialfall dieses Problems für $d = 2$ dar.

5.2 Komplexität

Die erste Frage, die wir untersuchen wollen, ist, wie sich die Komplexität des Problems dadurch verändert.

Satz 5.1. *Das Problem d-Hop Multipoint Relay ist auf planaren Graphen in Polynomialzeit lösbar.*

Beweis. Betrachten wir zunächst den Baumzerlegungsalgorithmus aus Abschnitt 4.1. Da es leicht ist eine Probleminstanz mit Kenntnis der Distanz-d-Nachbarn zu finden, die d-außerplanar ist, liegt die Vermutung nahe, dass sich damit auch die Laufzeit des Baumzerlegungsalgorithmus' verschlechtert. Nun benötigen wir aber für die Berechnung der Gateways der Schicht 1 nur die Informationen der Schichten 1 und 2, das entspricht aber gerade dem ursprünglichen Problem. Damit ist sichergestellt, dass alle Knoten der Schicht 2 benachrichtigt werden. Diese Überlegung lässt sich iterativ fortführen. Um sicherzustellen, dass die Knoten der Schicht $i + 1$ die Nachricht erhalten, benötigen wir nur die Informationen der Schichten i und $i + 1$ und können somit den Baumzerlegungsalgorithmus unverändert anwenden. Dementsprechend ändert sich die Komplexität des Problems im Planaren nicht und wir können eine optimale Lösung in Polynomialzeit berechnen. □

Mit der Idee des Beweises ist es klar, dass sich die beschriebenen Reduktionsregeln zur Berechnung der Gateways und echten Multipoint-Relay-Knoten eignen, wenn wir stets zwei aufeinanderfolgende Schichten betrachten.

Zusammenfassend lässt sich also sagen, dass sich die Komplexität des Problems für planare Graphen nicht ändert.

Greifen wir an dieser Stelle einmal die Frage auf, weshalb wir diese Betrachtungen auf planare Graphen beschränken. Wenn wir nochmals die gerade angestellten Überlegungen aufgreifen, so stellen wir fest, dass wir für die Berechnung der weitersendenden Knoten der Schicht i lediglich die Knoten der Schicht i und der Schicht $i+1$ betrachtet haben. Dabei erhalten die Schichten mit kleinerem Index implizit die Rolle des Knotens x und wir haben die Berechnung auf das ursprüngliche Problem reduziert. Auf diese Art und Weise reduzieren wir unseren Graphen um eine Probleminstanz des ursprünglichen Problems zu erhalten. Wenn wir nun von planaren Graphen ausgehen, können wir die innere Schichten zu einem Knoten zusammenziehen und erhalten wieder einen planaren Graphen. Die Reduktion ist also invariant auf der Klasse der planaren Graphen. Doch bereits bei dem Modell der Unit-Disk-Graphen erhält man durch Zusammenziehen der inneren Schichten nicht mehr zwingend einen Unit-Disk-Graphen. Deshalb können die hier geführten Überlegungen nicht auf beliebige Graphklassen verallgemeinert werden.

5.3 Lösungsgüte

In diesem Abschnitt wollen wir vergleichen, wieviele Knoten insgesamt in einem Netzwerk bei Berechnung der Multipoint-Relay-Menge durch verschiedene Algorithmen als Multipoint Relays ausgewählt werden. Wir haben bereits gezeigt, dass wir für planare Graphen aus der Sicht des sendenden Knotens x in der Lage sind, optimale Multipoint-Relay-Mengen zu berechnen. Doch die aus lokaler Sicht optimalen Lösungen stellen im Allgemeinen global keine optimale Lösung dar, wie wir später demonstrieren werden. Wir wollen nun den Vorgang des Broadcastens im gesamten Netz betrachten. Ein Knoten initiiert den Broadcast und berechnet mit einem gegebenen Algorithmus seine Multipoint Relays. Anschließend sendet er, und alle von ihm als Multipoint Relays bestimmten Knoten ermitteln wieder eine für sie optimale Multipoint-Relay-Menge usw. Wenn wir diesen Vorgang betrachten, so können wir alle Knoten zählen, die während der Verbreitung der Broadcast-Nachricht im Netz zum Senden aufgefordert wurden. Benutzen wir auf dem selben Graphen und ausgehend vom selben Knoten einen Algorithmus für ein anderes Multipoint-Relay-Modell, etwa einen d-Hop-Multipoint-Relay-Algorithmus, so können wir auch hier die sendenden Knoten zählen und mit der Gesamtanzahl der Multipoint Relays aus dem ersten Durchgang

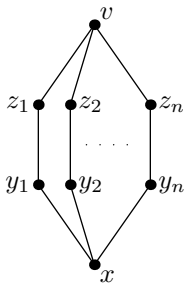


Abbildung 5.1: Im ursprünglichen Algorithmus wählt x alle y-Knoten als Multipoint-Relays aus. Diese wiederum berechnen ihren entsprechenden z-Nachbarn als Multipoint-Relay. Kennt nun aber x auch seine Distanz-3-Nachbarn stellen zwar immer noch alle y-Knoten Gateways dar, es wird aber nur einer der z-Knoten in die Menge der echten Multipoint-Relay-Knoten aufgenommen.

vergleichen.

Zunächst einmal halten wir fest, dass eine Lösung bei der Kenntnis der Distanz-d-Nachbarn nicht schlechter sein kann, als eine Lösung auf dem selben Graphen über einen Algorithmus für Multipoint Relay, da dieser einen normalen Multipoint-Relay-Algorithmus simulieren kann. Andererseits lässt sich jedoch bereits bei Kenntnis der Distanz-3-Nachbarschaft ein Graph konstruieren, bei dem der normale Algorithmus eine doppelte Anzahl Knoten als Multipoint Relays auswählen würde. Ein Beispiel dazu ist in Abb. 5.1 zu sehen. Nutzen wir für die Multipoint-Relay-Berechnung hier einen Algorithmus für Multipoint Relay, so befinden sich alle y-Knoten des Graphen in der kleinstmöglichen Multipoint-Relay-Menge für x. Dann bestimmen alle y-Knoten unabhängig voneinander wieder ihre Multipoint Relays, im Bildbeispiel wären dies jeweils die zu ihnen benachbarten z-Knoten[1]. Benutzen wir für die Berechnung der Multipoint Relays den ursprünglichen Algorithmus, also einen 2-Hop-MPR-Algorithmus, werden in diesem Netzwerk alle y- und alle z-Knoten zum Senden aufgefordert. Dies sind insgesamt $2n$ Knoten. Nutzen wir im Gegensatz dazu einen 3-Hop-MPR-Algorithmus, so stellen alle y-Knoten Gateways und einer der z-Knoten einen echten Multipoint-Relay-Knoten dar. Die Anzahl der sendenden Knoten ist durch Verwendung eines 3-Hop-MPR-Algorithmus um $n - 1$ Knoten kleiner.

[1]Wir können o.B.d.A. davon ausgehen, dass sich x in keiner Multipoint-Relay-Menge eines y-Knotens befindet, da die y-Knoten die Nachricht von x erhalten haben.

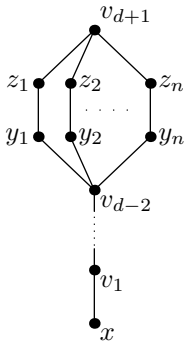


Abbildung 5.2: Bei so strukturierten Graphen begeht ein Algorithmus für d-Hop-MPR in der Schicht $d-1$, hier durch die Knoten $y_1, \ldots, y_n$ dargestellt, den Fehler all diese Knoten in die Lösungsmenge aufzunehmen, obwohl global betrachtet nur einer dieser Knoten nötig wäre.

Wenn wir wie im Beispiel die Lösungen von 2-Hop-MPR und 3-Hop-MPR vergleichen, werden höchstens in der zweiten Schicht Unterschiede in den Lösungen auftauchen. Diese Beobachtung begründet gleichzeitig, weshalb bei der Verwendung dieser beiden Algorithmen keine größeren Differenzen zwischen den sendenden Knoten auftreten können, als in Abb. 5.1 dargestellt. Führt man diesen Gedanken fort so stellt man leicht fest, dass bei der Berechnung der Knoten lediglich in der vorletzen Schicht, bei d-Hop Multipoint Relay also in der Schicht $d-1$, zu viele Knoten ausgewählt werden können. Derartige Extrembeispiele wie in Abb. 5.1 lassen sich für beliebiges d finden.

Wie wir in Satz 5.2 zeigen werde bedeutet das, dass sich immer ein Graph konstruieren lässt, bei der ein $(d-1)$-Hop-MPR-Algorithmus global nahezu doppelt so viele Knoten zum senden auffordert, wie dies ein d-Hop-MPR-Algorithmus tun würde.

Satz 5.2. *Bezeichne für einen beliebigen Graphen $OPT_{x,d}$ eine optimale Menge an Multipoint-Relay-Knoten, die ein Algorithmus bei Kenntnis der Distanz-d-Nachbarn und Startknoten x im gesamten Graphen auswählt. Für ein beliebiges $\epsilon > 0$ lässt sich immer ein Graph so finden, dass $(2-\epsilon)OPT_{x,d} = OPT_{x,d-1}$*

Beweis. Betrachten wir dazu einen Graph wie in Abb. 5.2 dargestellt. Es ist leicht einsichtig, dass ein d-Hop-MPR-Algorithmus nur einen der Knoten $\{z_1, \ldots, z_n\}$ zum Senden auffordert. Im Gegensatz dazu stellen bei Verwendung eines $(d-1)$-Hop-MPR-Algorithmus alle mit $y_i, 1 \leq i \leq n$, bezeichne-

ten Knoten echte Multipoint Relays dar und für diese muss wiederum eine $(d-1)$-Hop-MPR-Berechnung stattfinden. Daraufhin werden alle Knoten $z_j, 1 \leq j \leq n$, zum Senden aufgefordert. Während des Sendevorganges unterscheiden sich die Aufforderungen zum Broadcasten nur in der Schicht, die die Knoten z_j enthält. Der Algorithmus mit Kenntnis der $(d-1)$-Nachbarschaft fordert alle der Knoten z_j zum Senden auf, wohingegen bei der d-Hop-MPR-Berechnung nur einer der Knoten z_j einen sendenden Knoten darstellt.

Bezeichne A_d die Anzahl der Knoten, die während der d-Hop-MPR-Berechnung zum Senden aufgefordert werden, A_{d-1} analog. Dann gilt:

$$A_{d-1} = (d-2) + n + n, \tag{5.1}$$

$$A_d = (d-2) + n + 1. \tag{5.2}$$

Da

$$\frac{A_{d-1}}{A_d} = \frac{(d-2)+n+n}{(d-2)+n+1}, \tag{5.3}$$

berechnet sich der Approximationsfaktor bei festem d für wachsendes n so:

$$\lim_{n\to\infty} \frac{A_{d-1}}{A_d} = \frac{2n}{n} = 2. \tag{5.4}$$

Das bedeutet aber, dass für beliebiges $\epsilon > 0$ ein n so existiert, dass der Approximationsfaktor mindestens $2 - \epsilon$ beträgt. □

Interessant ist auch, dass die erste Idee einer optimalen Lösung über ein Connected Dominating Set, bei der lediglich die Erreichbarkeit aller Knoten gefordert wurde, bei den in den Abbildungen gezeigten Beispielen deutlich weniger Knoten auswählen würde, zum Beispiel in Abb. 5.1 lediglich drei Knoten, o.B.d.A. y_1, z_1 und v.

Erstaunlich ist jedoch, dass eine solche Konstruktion bei der wir uns für beliebiges d einem konkreten Faktor annähern in den angesprochenen speziellen planaren Graphklassen, also Gabriel-Graphen bzw. Relative-Neighbourhood-Graphen, nicht machbar ist. Wenn wir den Beweis von Satz 3.6 leicht modifizieren sehen wir, dass für $i \geq 1$ kein Knoten einer Schicht i mehr als zwei Nachbarn in einer Schicht $i+1$ und umgekehrt haben kann. Mit dieser Überlegung sehen wir leicht, dass eine Konstruktion wie in Abb. 5.1 für diese Graphen nicht möglich ist.

Lemma 5.3. *Ein $(d-1)$-Hop-MPR-Algorithmus wählt in einem Gabriel-Graph bzw. einem Relative-Neighbourhood-Graph in der Schicht $d-2$ höchstens dreimal mehr Knoten aus, als ein d-Hop-MPR-Algorithmus dies tun würde.*

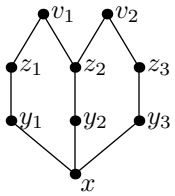


Abbildung 5.3: Ein 2-Hop-MPR-Algorithmus würde hier jeden der z-Knoten zum Weitersenden auffordern, wobei ein 3-Hop-MPR-Algorithmus nur z_2 wählen würde.

Beweis. Einerseits können wir einen Graph angeben, bei dem ein $(d-1)$-Hop-MPR-Algorithmus im Vergleich zum d-Hop-MPR-Algorithmus genau den genannten Fehler begeht. Für $d = 3$ ist in Abb. 5.3 ein Beispiel dargestellt.

Andererseits kann keine größere Differenz entstehen. Betrachten wir dazu einen Knoten u_1 der Schicht $d-1$ und seine zwei Nachbarn v_1 und v_2 der Schicht d. Die Knoten v_1 und v_2 haben jeder höchstens einen weiteren Nachbarn, o.B.d.A. u_2 bzw. u_3, in der Schicht $d-1$. Da wir mindestens den Knoten u_1 in eine optimale Lösung aufnehmen müssen, im schlechtesten Fall jedoch u_1, u_2 und u_3 aufgenommen werden, kann der entstehende Fehler in der Schicht $d-1$ den Faktor drei nicht übersteigen. □

Vergleichen wir dieses Lemma einmal mit Satz 5.2 so wird klar, dass wir bei den beiden Routing-Graphklassen im Unterschied zu allgemeinen planaren Graphen mit wachsendem d immer näher an ein globales Optimum kommen.

5.4 Vor- und Nachteile

Wie wir gezeigt haben, können sich mit Kenntnis einer größeren Umgebung als der bisher betrachteten Distanz-2-Informationen eines Knotens die MPR-Mengen verkleinern. Dabei ist jedoch zu beachten, dass bereits für den Austausch der Nachbarschaften der Knoten Kommunikationsaufwand nötig ist. Um möglichst wenig Kommunikationskapazität des Netzes durch den Austausch von solchen Kontrollnachrichten zu verbrauchen, werden ebendiese Nachrichten kürzestmöglich gehalten. Dies kollidiert nun aber mit der Tatsache, dass ständig Informationen über weit entfernte Nachbarschaften übermittelt werden müssen. Weiterhin benötigt der Austausch der Nachbarschaftsinformationen auch Zeit. Da wir aber eine MPR-Berechnung nicht auf

veralteten Informationen durchführen wollen, muss ebenfalls gewährleistet sein, dass das Netzwerk zum Zeitpunkt einer Multipoint-Relay-Berechnung auch noch der Topologie unterliegt, die der Knoten x als aktuelle Informationen gespeichert hat.

Demzufolge müssen die Vorteile, die uns die Kenntnis über weiter entfernte Knoten für die Berechnung bringt, gegen die Nachteile, die in wirklichen Netzwerken durch die Forderung der Informationen entstehen, abgewogen werden. Da hier von Netzwerk zu Netzwerk und von Anwendung zu Anwendung die unterschiedlichsten Gegebenheiten aufeinandertreffen, kann eine solche Abwägung nicht allgemein erfolgen, sondern muss in jedem Einzelfall gesondert durchgeführt werden.

Weiterhin haben wir in Lemma 5.3 gesehen, dass die in dieser Arbeit vorgestellten Routing-Graphen keine bloße Einschränkung der planaren Graphen darstellen, sondern bei dieser Form der Berechnung, insbesondere für große Werte von d, deutlich bessere Lösungen besitzen können als allgemeine planare Graphen.

Kapitel 6

Ein Approximationsalgorithmus für Unit-Disk-Graphen

Neben den bisher beleuchteten planaren Graphen und den bereits erwähnten Routing-Graphen [BB06] existiert eine weitere Klasse von Graphen, die zur Modellierung von mobilen Netzwerken verwendet wird, die Unit-Disk-Graphen. Es ist naheliegend diese Graphklasse zu verwenden, da in derartigen Netzwerken jeder Knoten einen Sender mit einer bestimmten Sendeleistung darstellt. Bisher ist uns jedoch für Unit-Disk-Graphen kein exakter Algorithmus bekannt, der eine polynomiale Laufzeit vorweisen kann. Calinescu, Mandoiu, Wan und Zelikovsky [CMWZ04] zeigen die bisher bekannten Ergebnisse auf und behaupten, dass die Komplexität dieses Problems auf Unit-Disk-Graphen zur Zeit unbekannt ist. Demzufolge ist es bisher weder gelungen einen Algorithmus anzugeben, der eine optimale Lösung in weniger als exponenziellem Aufwand findet, noch ließ sich bisher beweisen, dass es keinen besseren Algorithmus geben kann. Weiterhin präsentieren Calinescu et al. einen exakten Algorithmus mit einer Laufzeit von $O(n \log^2 n)$ unter der Einschränkung, dass sie durch den x-Knoten zwei orthogonale Geraden legen und lediglich Knoten eines der so entstehenden Quadranten betrachten. Setzt man die so entstehenden Lösungen aller vier Quadranten zusammen, kann man zeigen, dass die Gesamtlösung höchstens dreimal größer ist, als eine optimale Lösung. Weiterhin stellen sie einen 3-Approximationsalgorithmus für Multipoint Relay vor, der eine Laufzeit von $O(n \log^2 n)$ besitzt.

Wir präsentieren in diesem Kapitel einen additiven Approximationsalgo-

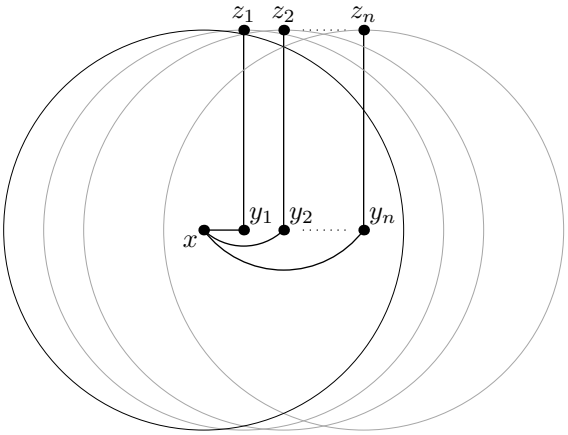


Abbildung 6.1: In diesem Beispiel stellt MPR_x eine n-elementige Menge dar. In diesem Netzwerk würden z.B. y_1 und z_1 als sendende Knoten genügen. Im dargestellten Graph sind nur die für eine normale MPR-Berechnung relevanten Kanten dargestellt.

rithmus mit einer Laufzeit von $O(n)$. Wir legen dafür ein leicht modifiziertes Problem zugrunde, können also die Ergebnisse von Calinescu et al. nicht direkt mit den unseren vergleichen.

Wir können beobachten, dass selbst optimale Lösungen zu einer großen Anzahl sendender Knoten führen können, wenn die Knoten dicht liegen. Es können demzufolge Probleminstanzen existieren, bei denen große Multipoint-Relay-Mengen berechnet werden, obwohl nur wenige Knoten des Netzwerkes senden müssten. Ein Beispiel dafür ist in Abb 6.1 zu sehen. Obwohl alle zu erreichenden z-Knoten untereinander benachbart sind und somit ein Knoten des Netzwerks genügen würde, um alle z-Knoten zu dominieren, stellt MPR_x eine große und zugleich optimale Multipoint-Relay-Menge dar.

Wir wollen in diesem Kapitel einen Ansatz vorstellen, der sowohl dieses Problem umgeht, als auch Lösungen liefert, die höchstens doppelt so groß sind, wie eine optimale Lösung. Dafür benötigen wir lediglich noch die Zusatzinformation der Verbindung der z-Knoten untereinander. Diese Information bekommen wir indem wir entweder einmal mehr als beim ursprünglichen Modell Kontrollinformationen austauschen oder indem die geographischen Positionen der Knoten bekannt sind. Da der Algorithmus im wesentlichen darauf basiert, eine maximale unabhängige Menge von z-Knoten zu benachrichtigen, wollen wir ihn im Folgenden auch als Independent-Set-MPR-Algorithmus oder kurz IS-MPR-Algorithmus bezeichnen.

6.1 Der IS-MPR-Algorithmus

Die Idee hinter dem Algorithmus ist die, dass wir nun nicht mehr versuchen, alle z-Knoten direkt von den y-Knoten benachrichtigen zu lassen, sondern einen weiteren Schritt zulassen. Damit ist gemeint, dass nun auch indirekt z-Knoten zum Weitersenden aufgefordert werden können. Diese Modifikation des Algorithmus' ist für die Berechnung erst hilfreich, wenn wir mehr Informationen über die Nachbarschaften der z-Knoten besitzen als im ürsprünglichen Algorithmus, hier sind dies die Kanten der z-Knoten untereinander. Dazu bestimmen wir im ersten Schritt eine nicht vergrößerbare unabhängige Menge von z-Knoten, bestimmen im nächsten Schritt eine optimale Menge von y-Knoten, die diese z-Knoten benachrichtigen und fordern dann von den y-Knoten unserer Multipoint-Relay-Menge, dass sie die z-Knoten dieser unabhängigen Menge in ihre eigene Multipoint-Relay-Menge aufnehmen.

Schritt 1 - Bestimmen einer maximalen unabhängigen Knotenmenge. Wir setzen in diesem Modell voraus, dass wir die Verbindungen der z-Knoten untereinander kennen, es ist uns also möglich, eine maximale unabhängige Menge von z-Knoten zu bestimmen. Eine solche Menge von z-Knoten stellt eine dominierende Menge für alle z-Knoten dar, d.h. alle anderen z-Knoten sind zu wenigstens einem Knoten aus dieser unabhängigen Menge benachbart. Demzufolge genügt es nun also, jeden z-Knoten dieser unabhängigen Menge zu benachrichtigen und von ihnen zu fordern ihrerseits nochmals zu senden.

Bevor wir uns jedoch damit beschäftigen, wollen wir uns diese unabhängige Menge etwas genauer anschauen und fragen, wie groß eine solche unabhängige Menge werden kann.

Lemma 6.1. *Eine maximale unabhängige Menge von z-Knoten in einer Unit-Disk-MPR-Instanz kann höchstens 18 Knoten enthalten.*

Beweis. Dieses Problem ist im Kern eine Variante des sogenannten Disc Packing, bei dem in eine vorgegebene Fläche möglichst viele Kreise gleicher Größe eingepasst werden sollen, ohne dass diese sich überschneiden. Wir könnten dieses Lemma auf ein Disc-Packing-Problem übertragen, indem wir die z-Knoten als Kreise mit dem Radius $r = 1/2$ darstellen und den Kreisring, auf dem wir die z-Knoten platzieren können (siehe Abb. 6.2), jeweils nach außen und nach innen um diesen Radius vergößern. Da aber für solche speziellen Disc-Packing-Instanzen bisher keine Lösungen dokumentiert wurden, müssen wir einen anderen Ansatz versuchen.

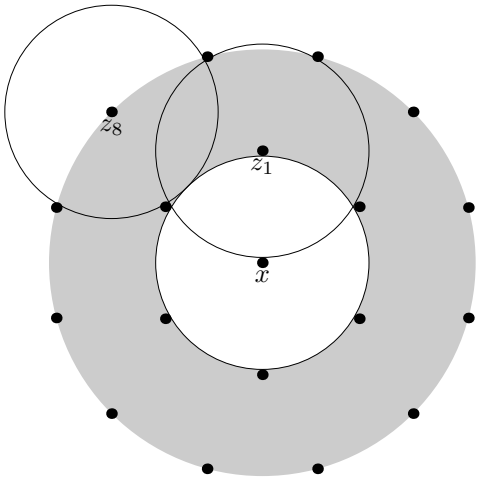


Abbildung 6.2: Die grau unterlegte Fläche beschreibt das Gebiet, indem sich z-Knoten befinden können. Die darin dargestellten z-Knoten zeigen eine dichtestmögliche Anordnung derart, dass kein z-Knoten zu einem weiteren benachbart ist.

Betrachten wir dazu den Extremfall, dass wir möglichst viele z-Knoten im Abstand 1 und möglichst viele z-Knoten im Abstand 2 zum Knoten x platzieren. In Abb. 6.2 ist der Fall dargestellt. Um die Übersicht zu wahren, sind nur von einem inneren und einem äußeren z-Knoten die Unit-Disks eingezeichnet. Man sieht auch leicht, dass so keine weiteren z-Knoten plaziert werden können, die nicht zu wenigstens einem der unabhängigen Knoten benachbart sind. □

Eine solche maximale unabhängige Knotenmenge lässt sich effizient mittels eines einfachen Greedy-Ansatzes finden. Wir nehmen einen z-Knoten in die Menge auf und löschen alle seine z-Nachbarn. Solange noch Knoten vorhanden sind, wird dieser Schritt wiederholt.

Schritt 2 - Bestimmen einer optimalen MPR-Menge für die unabhängigen Knoten. In einer MPR-Instanz für das ursprüngliche Problem kann die Anzahl der zu betrachtenden y- und z-Knoten beliebig groß sein, da die Knoten beliebig dicht beieinander liegen dürfen. Wie bereits gezeigt, ist nach der Berechnung der maximalen unabhängigen Menge die Anzahl der z-Knoten im so modifizierten Problemgraph durch eine Konstante beschränkt. Deshalb ist auch die Anzahl aller y-Knoten mit unterschiedlichen Nachbarschaften beschränkt und somit auch die Größe des gesamten

Graphen. Deshalb ist es ohne weiteres möglich, in konstanter Zeit eine optimale Lösung über einen beschränkten Suchbaum oder eine vergleichbare Strategie zu finden.

Schritt 3 - Nachricht und Kontrollinformation übermitteln. Nachdem die maximale unabhängige Menge bestimmt wurde und eine optimale Multipoint-Relay-Menge für diese z-Knoten berechnet wurde, sendet der Knoten x und teilt gleichzeitig allen Multipoint Relays mit, welche der unabhängigen z-Knoten sie zu benachrichtigen haben. Da diese z-Knoten direkte Nachbarn der y-Knoten darstellen, können diese direkt in deren Multipoint-Relay-Menge aufgenommen werden.

Es ist leicht zu sehen, dass sich der angeführte Algorithmus effizient ausführen lässt.

Satz 6.2. *Auf einer gegebenen Unit-Disk-MPR-Instanz können wir eine Lösung durch den IS-MPR-Algorithmus in $O(n)$ Schritten berechnen.*

Beweis. Für die Bestimmung einer nicht vergrößerbaren MPR-Menge benötigen wir mit einem Greedy-Ansatz offensichtlich $O(n)$ Schritte. Anschließend können wir in konstanter Zeit eine minimale Anzahl Multipoint Relays für diese unabhängige Menge berechnen. Dies zeigt, dass die gesamte Berechnung in $O(n)$ Schritten möglich ist. □

6.2 Gütebetrachtungen

In diesem Abschnitt wollen wir untersuchen, wie sich eine so gefundene Lösung von einer optimalen Multipoint-Relay-Lösung unterscheidet.

Satz 6.3. *Eine durch den IS-MPR-Algorithmus gefundene Lösung besitzt höchstens 18 Multipoint Relays mehr als eine optimale Lösung.*

Beweis. Offensichtlich muss eine optimale Lösung auch alle Knoten der unabhängigen Menge benachrichtigen, für die der IS-MPR-Algorithmus eine Multipoint-Relay-Menge berechnet. Da die vom IS-MPR-Algorithmus berechnete Menge optimal ist, muss eine optimale MPR-Menge mindestens ebensoviele Knoten beinhalten.

Jetzt wollen wir untersuchen, wieviele Knoten nach der Berechnung einer optimalen Menge für die unabhängigen z-Knoten durch den IS-MPR-Algorithmus zusätzlich zum Broadcasten der Nachricht aufgefordert werden. Dies sind aber offensichtlich gerade die Knoten, die x als unabhängige Menge von z-Knoten berechnet hat, also nach Lemma 6.1 höchstens 18 Knoten. □

Untersuchen wir mit Kenntnis dieses Ansatzes nun noch einmal das Anfangsbeispiel in Abbildung 6.1. Wir hatten zu Beginn dieses Kapitels bereits festgestellt, dass ein gewöhnlicher MPR-Algorithmus hier eine große Menge Multipoint Relays berechnet. Betrachten wir nun einmal eine IS-MPR-Berechnung auf dem gezeigten Graphen. Im ersten Schritt ermittelt der Algorithmus eine maximale unabhängige Menge von z-Knoten mit genau einem Element und berechnet demzufolge eine IS-MPR-Menge der Größe eins. Dieser y-Knoten fordert dann den z-Knoten der unabhängigen Menge wiederum zum Senden auf und nach zwei Schritten sind alle Knoten des Graphen benachrichtigt. Dieses Beispiel verdeutlicht, dass eine Berechnung mit dem IS-MPR-Algorithmus zu wesentlich besseren Lösungen führen kann, als ein MPR-Algorithmus liefern könnte.

6.3 Zusammenfassung

In diesem Kapitel ist es uns gelungen, einen additiven Approximationsalgorithmus anzugeben. Allerdings weichen wir dafür leicht von dem ursprünglichen MPR-Modell ab. Zum Einen fordern wir Wissen über die Nachbarschaften oder die geographischen Positionen der z-Knoten, denn sonst hätten wir keine Möglichkeit, eine maximale unabhängige Menge dieser Knoten zu bestimmen. Zum Anderen fordern wir nicht mehr, wie es das ursprüngliche MPR-Modell implizit tut, dass alle z-Knoten direkt von den y-Knoten benachrichtigt werden. Deshalb lässt sich die Lösungsqualität unseres Algorithmus auch nicht direkt mit dem eingangs erwähnten 3-Approximationsalgorithmus von Calinescu et al. [CMWZ04] vergleichen. Allerdings legt der von uns gefundene Algorithmus für das so modifizierte Problem die Vermutung nahe, dass zumindest diese Variante kein NP-hartes Problem darstellt, da für solche Probleme normalerweise keine additiven Approximationen gefunden werden können.

Zu beobachten ist auch, dass sich dieser Ansatz nicht mit einer vergleichbaren Lösungsgüte auf beliebige Graphklassen ausdehnen lässt. Dafür können wir als Beispiel die bereits untersuchten planaren Graphen heranziehen. Hier ist es auch mit Kenntnis der Nachbarschaften der z-Knoten nicht möglich, eine Beschränkung für die maximale Unabhängige Menge anzugeben, die aber für die Güte des Algorithmus entscheidend ist. Will man diesen Algorithmus für Disk-Graphen nutzen, so stellet man fest, dass die unabhängige Menge der z-Knoten auch hier nicht beschränkt werden kann. Beschränkt man aber die Radien der Kreise im Disk-Graph sowohl nach oben als auch nach unten, so lässt sich der Algorithmus für diese Graphen

anwenden. Eine derartige Einschränkung der Disk-Graphen bei der Modellierung von mobilen Netzwerken ist gut denkbar, da die Senderadien der Teilnehmer weder beliebig große noch beliebig kleine Werte aufweisen.

Kapitel 7

Schlußbemerkungen

In diesem Kapitel wollen wir eine kurze Zusammenfassung der Ergebnisse präsentieren und mögliche Ansatzpunkte für eine zukünftige Forschung auf diesem Gebiet festhalten.

7.1 Zusammenfassung

In Kapitel 2 haben wir die für diese Arbeit wesentlichen Grundlagen der Graphentheorie vorgestellt und gezeigt, dass eine optimale Multipoint-Relay-Menge auf allgemeinen Graphen nicht effizient berechenbar ist. Daraufhin haben wir uns in Kapitel 3 zunächst auf planare Graphen als Probleminstanz beschränkt und gezeigt, dass auf diesen Graphen eine optimale Lösung durch Datenreduktion in $O(n^3)$ Schritten möglich ist. In Kapitel 4 haben wir einen weiteren Ansatz vorgestellt, eine Lösung durch dynamisches Programmieren auf einer Baumzerlegung. Zum Einen haben wir dadurch einen Polynomialzeitalgorithmus mit einer Laufzeit von $O(n)$ für planare Graphen erhalten, da eine planare Multipoint-Relay-Instanz eine Baumweite $\omega \leq 5$ besitzt. Zum Anderen ist so auch ein Algorithmus für alle Graphklassen gegeben, bei denen wir für das Problem Multipoint Relay eine beschränkte Baumweite nachweisen können. Anschließend haben wir uns in Kapitel 5 von dem ursprünglichen Modell gelöst und untersucht, wie sich die Berechnung und die Komplexität des Problems ändert, wenn ein Knoten alle umliegenden Knoten mit Abstand $\leq d$ kennt. Dabei haben wir festgestellt, dass in der Berechnung keine wesentlichen Änderungen auftreten, es jedoch durch die größere Menge von Informationen möglich ist, die Menge aller sendenden Knoten im gesamten Netzwerk zu verringern. Da jedoch die Kenntnis der Distanz-d-Nachbarschaft in der Praxis mit Kommunika-

tionsaufwand verbunden ist, ist der Nutzen dieser Verbesserung von Fall zu Fall unterschiedlich zu bewerten. In Kapitel 6 haben wir schließlich die Klasse der Unit-Disk-Graphen untersucht und für ein leicht abgeändertes Multipoint-Relay-Modell einen Approximationsalgorithmus mit einer additiven Konstante vorgestellt, der in Polynomialzeit durchführbar ist.

7.2 Ausblick

Wir haben für die Klasse der planaren Graphen optimale Polynomialzeitalgorithmen vorgestellt. Für welche Graphklassen existieren ebenfalls effizient berechenbare Algorithmen und für welche Klassen stellt die Multipoint-Relay-Berechnung ein NP-hartes Problem dar? In diesem Zusammenhang ist noch immer offen, ob sich Multipoint Relay auf Unit-Disk-Graphen effizient berechnen lässt [CMWZ04]. Sobald die Schwierigkeit des Problems für Unit-Disk-Graphen bekannt ist, lässt sich die Frage nach der Komplexität von Multipoint Relay auf Disk-Graphen ausdehnen.

Eine weitere Fragestellung ist, welche Graphen, für die sich das Problem effizient lösen lässt, sich als Routing-Graphen für mobile Netzwerke eignen. Da wir bereits einen Algorithmus für Graphen mit beschränkter Baumweite angegeben haben, wären hier Graphen denkbar, von denen bereits bekannt ist, dass sie eine Beschränkung der Baumweite vorweisen.

In den vorherigen beiden Kapiteln haben wir uns von dem ursprünglichen Modell gelöst und gesehen, dass mit leichten Modifikationen des Problems ebenfalls ein Broadcasten mit geringer Netzbelastung möglich ist. Demzufolge sollten auch Modifikationen und Erweiterungen des Modells gesucht werden, die effiziente Berechnungen zulassen.

7.3 Danksagungen

Ich möchte an dieser Stelle meinen Betreuern Rolf Niedermeier, Michael Dom und Falk Hüffner für die engagierte Unterstützung danken. Sie haben mir hilfreiche Tipps gegeben, in Gesprächen konstruktive Unterstützung geleistet und die nötige Geduld für mich aufgebracht. Außerdem danke ich meiner lieben Mutter für die tatkräftige Unterstützung während meiner gesamten Studienzeit und darüber hinaus.

Literaturverzeichnis

[ABF+02] Jochen Alber, Hans L. Bodlaender, Henning Fernau, Ton Kloks, and Rolf Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002. 16, 50

[BB06] Kevin Buchin and Maike Buchin. Topology control. In Dorothea Wagner and Roger Wattenhofer, editors, *Algorithms for Sensor and Ad Hoc Networks*, LNCS. Springer, 2006. To appear. 12, 68

[BCOP03] Sergiy Butenko, Xiuzhen Cheng, Carlos A. S Oliveira, and P. M. Pardalos. A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks, 2003. 59

[Bod93] Hans L. Bodlaender. A tourist guide through treewidth. *Acta Cybern.*, 11(1-2):1–22, 1993. 16

[Bod98] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. 50

[CMWZ04] Gruia Calinescu, Ion I. Mandoiu, Peng-Jun Wan, and Alexander Zelikovsky. Selecting forwarding neighbors in wireless ad hoc networks. *Mobile Networks and Applications*, 9(2):101–111, 2004. 7, 68, 73, 77

[Com96] ETSI STC-RES10 Comittee. Radio equipment and systems: High performance local area network (hiperlan) type 1, functional specifications. In *ETS*, pages 300–652, 1996. 7

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999. 19

[Die06] Reinhard Diestel. *Graphentheorie*. Springer, 2006. 10, 13

[GK98] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998. 59

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. 19, 40

[OKD06] Sadao Obana, Naoto Kadowaki, and Peter Davis. Breakthroughs in large-scale ad hoc wireless networking and application for vehicle safety. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)*, page 88, Washington, DC, USA, 2006. IEEE Computer Society. 4

[QVL02] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *HICSS*, page 298, 2002. 7

[RDJ+04] Lu Ruan, Hongwei Du, Xiaohua Jia, Weili Wu, Yingshu Li, and Ker-I Ko. A greedy approximation for minimum connected dominating sets. *Theor. Comput. Sci.*, 329(1-3):325–330, 2004. 59

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. 16

[SCS02] Yoav Sasson, David Cavin, and André Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. Technical report, 2002. 5

[TK84] Hideaki Takagi and Leonard Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984. 14

[WLD06] Jie Wu, Wei Lou, and Fei Dai. Extended multipoint relays to determine connected dominating sets in MANETs. *IEEE Trans. Computers*, 55(3):334–347, 2006. 7

[XMKS04] Qing Xu, Tony Mak, Jeff Ko, and Raja Sengupta. Vehicle-to-vehicle safety messaging in DSRC. In *VANET '04: Proceedings*

of the 1st ACM international workshop on Vehicular ad hoc networks, pages 19–28, New York, NY, USA, 2004. ACM Press. 4

[ZA05] Qi Zhang and Dharma P. Agrawal. Dynamic probabilistic broadcasting in MANETs. *J. Parallel Distrib. Comput.*, 65(2):220–233, 2005. 5

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Jena, 26. Juli 2006

Alexander Fanghänel