Standortplanung mit Blick auf Online-Strategien

Riko Jacob

22. September 1997

Diplomarbeit am
Lehrstuhl für Informatik I
der Universität Würzburg

Inhaltsverzeichnis

1 Einleitung

Mobile Telephone werden immer selbstverständlicher, die entsprechenden Funknetze dichter und die Anzahl verschiedener Anbieter wächst. Dementsprechend wichtig ist es, begrenzte Ressourcen so gut wie möglich auszuschöpfen. Die beschränkteste und teuerste Ressource ist dabei sicher das Funkspektrum. Seit geraumer Zeit streiten sich etwa einige Funkdienste (Amateurfunk, CB, kommerzieller Funk, Radio und Betriebsfunk) um die Frequenzen im 2m-Band (144–149 MHz) und im 70cm-Band (430–450MHz).

Eine der in diesem Zusammenhang interessanten Fragestellungen ist es, die Senderanlagen für ein Mobilfunknetz so zu plazieren, daß ein möglichst effektives Netz entsteht. Um hier ein vernünftiges Qualitätsmaß zu definieren, muß man zunächst das Verhalten des Systems während des Betriebes kennen und verstehen.

Diese Thematik ist offenbar schon einige Zeit bekannt und bearbeitet, es gibt schließlich schon einige kommerzielle Netze. Diese Tatsache ist allerdings weit davon entfernt, ein Beweis zu sein, daß so das verfügbare Funkspektrum tatsächlich optimal oder zumindest gut genutzt wird. Die mir bekannte Fachliteratur, die sich mit dem Betrieb von Mobilfunknetzen beschäftigt, konzentriert sich auf verschiedene spezielle Modelle. Zum einen werden hexagonale Zellen betrachtet, also eine sehr regelmäßige geographische Verteilung der Feststationen. Desweiteren wird die Dynamik des Systems oft probabilistisch modelliert. Eine andere Richtung läßt dann zwar beliebige Senderstandorte zu, konzentriert sich aber auf statische Betrachtungen.

So werden etwa in [Sim89] und [Sim88] Probleme betrachtet, die aus der Kanalzuweisung für hexagonale Zellen entstehen. Dabei werden auch sogenannte hybride Situationen untersucht, also der Fall, daß bestimmte Zellen aus historischen Gründen mit festen Frequenzen arbeiten und die übrigen eine dynamische Frequenzverteilung erlauben.

Etwa in [DV93, ZY89] wird die Dynamik des Systems als einfaches probabilistisches Modell angenommen, was eine weitverbreitete Betrachtungsweise zu sein scheint. Die untersuchten Strategien werden meist experimentell mit anderen bekannten Strategien verglichen. Dabei steht nicht immer die Qualität, sondern auch die Implementierbarkeit auf verteilten Rechnern, also eine spezielle Parallelisierbarkeit, im Vordergrund. Es ergibt sich die Frage, ob diese unscharfe Betrachtungsweise angemessen ist, oder ob es Strategien geben kann, die eine Qualitätsgarantie haben, also auch im ungünstigsten Fall noch bestimmte Minimalforderungen erfüllen.

Eine allgemeinere Betrachtung, die uneingeschränkte Senderstandorte zuläßt findet man z.B. in [Mal95, MP96]. Dort führt eine hybride Situation zu sogenannten List-Coloring-Problemen. Ein noch allgemeinerer Ansatz wird in [DJLS95] untersucht. Dort werden auch die Auswahl eines Senders für eine konkrete Anfrage und Variationen in der Sendeleistung zugelassen. Es wird gezeigt, daß auch dieses Problem **NP**-hart ist. Außerdem wird ein Algorithmus angegeben, für den allerdings keine Qualitätsgarantie gezeigt wird. Dieser Algorithmus beendet bestehende Verbindungen unter Umständen vorzeitig.

Die Mobilfunk-Problematik ist die Motivation für die hier vorgestellten Ergebnisse. Im Gegensatz zu der erwähnten Literatur soll in dieser Arbeit deren Online-Charakter im Mittelpunkt stehen. Dazu werden graphentheoretische Probleme formuliert, die mögliche zentrale Fragestellungen der Anwendung aufgreifen. Diese werden daraufhin untersucht, ob und unter welchen Voraussetzungen effiziente Online-Strategien existieren. Dabei trägt die Online-Formulierung der Tatsache Rechnung, daß z.B. bei der Kanalzuteilung an Mobilfunkgespräche bereits ein Kanal zugewiesen werden muß, bevor alle Anfragen bekannt sind, und damit genügend Informationen vorhanden sind, um das Problem optimal zu lösen. Auch wenn die global optimale Offline-Lösung nicht direkt zur Lösung des Problems in der Praxis genutzt werden kann, so ist sie doch ein wertvoller Maßstab, mit dem sich verschiedene Online-Strategien beurteilen lassen. Dieses Vorgehen, die Qualität zu messen, wird kompetitive Analyse genannt [OSH94a, OSH94b]. Diese Art von Analyse hat den Vorteil, daß die Qualität der Aussagen nicht von Annahmen über Wahrscheinlichkeitsverteilungen abhängt, sondern alle möglichen Situationen betrachtet, und damit auch Voll- und Überlastsituationen einschließt.

In dieser Arbeit werden dementsprechend Probleme der Online-Frequenzzuteilung im Sinne einer kompetitiven Analyse betrachtet. Desweiteren wird die Berechnungskomplexität von Fragestellungen, die die Standortauswahl von Feststationen modellieren, untersucht. Diese Probleme erweisen sich als im allgemeinen schwer lösbar. Die Graphen, an denen sich diese Berechnungsschwierigkeit zeigt, geben Einblick in die Struktur des Problems, und können so einen Hinweis geben, auf welchen speziellen Graphen das Problem noch exakt lösbar ist. Die hier vorgestellten Resultate sind in diesem Sinne sehr stark, da sie mit nicht allzu komplizierten Graphen starke Hardness-Resultate erzielen, die zum Teil Nicht-Approximierbarkeiten einschließen. Genauer handelt es sich um Graphen von beschränkter Baumweite, für die viele im allgemeinen schwer berechenbare und approximierbare Probleme, wie die Bestimmung einer optimalen Färbung, in polynomieller Zeit exakt gelöst werden können. Dies gilt insbesondere für das hier betrachtete Umfärbeproblem, bei dem die Schwierigkeit, zulässige Lösungen durch lokale Veränderungen ineinander umzuwandeln, nicht aus der Schwierigkeit Lösungen zu finden entsteht.

Für die Frequenzzuteilungen zeigt sich, daß neben bestimmten Schranken für reine Online-Fragestellungen auch Berechnungsaspekte zu Schwierigkeiten führen. Diese Betrachtungen werden abgerundet, indem Graphklassen vorgestellt werden, für die die Probleme exakt lösbar sind.

Für die Problematik der Senderauswahl wird gezeigt, daß verschiedene natürliche Optimierungsprobleme schwer zu approximieren sind.

Aufbau der Arbeit

Zunächst werden in Kapitel 2 Grundlagen, Definitionen und Resultate aus Graphen- und Komplexitätstheorie vorgestellt.

Im Kapitel 3 werden die Anforderungen beim Betrieb des Netzes modelliert. Das dabei entstehende Frequenzzuteilungsproblem wird auf Komplexität und Online-Charakter untersucht, sowohl im Bezug auf die Anzahl der verwendeten Frequenzen, als auch auf die Anzahl der angenommenen Gespräche. Die vorgestellten neuen Ergebnisse zeigen zum einen, daß allgemeine Fragestellungen sehr schwer zu beantworten (**PSPACE**-hart) sind, und geben zum anderen spezielle Graphklassen an, für die die Probleme in vernünftiger Zeit gelöst werden können.

Schließlich befaßt sich Kapitel 4 mit der Fragestellung, wie die Standorte von neuen Feststationen ausgewählt werden sollen. Es wird das Problem betrachtet, eine Senderauswahl so zu treffen, daß der Interferenzgraph zu einer Klasse gehört, die im vorhergehenden Kapitel als günstig erkannt wurde. Diese und einige verwandte Fragestellungen erweisen sich als schwer berechenbar, es werden **NP**-Hardness- und Nicht-Approximierbarkeits-Resultate vorgestellt.

Danksagung

An dieser Stelle möchte ich allen danken, die zum Entstehen dieser Arbeit beigetragen haben.

Zuallererst gilt dies meinen Betreuern Herrn Prof. Dr. Noltemeier und Herrn Dr. S. O. Krumke, die mich auch über große Entfernungen hinweg durch ungezählte Diskussionen und Anregungen unterstützt haben. Desweiteren bedanke ich mich bei Hans-Christoph Wirth, Christa Mauer und meiner Mutter für das sorgfältige Korrekturlesen. Außerdem gilt mein Dank den Mitarbeitern des TRANSIMS-Projektes am Los Alamos National Laboratory, insbesondere Herrn Madhav Marathe, deren Geduld und Unterstützung es mir erlaubt haben, die Arbeit hier in Los Alamos fertigzustellen.

2

Grundlagen

In diesem Kapitel werden einige nicht ganz selbstverständliche Definitionen und Notationen eingeführt. Außerdem werden bekannte Ergebnisse zusammengetragen, die in der ganzen Arbeit immer wieder benutzt werden.

2.1 Graphentheorie

Definition 1
Ein (endlicher ungerichteter) ***Graph*** $G = (V, E)$ *besteht aus der endlichen Knotenmenge* V *und einer Menge* E *von Kanten. Die Elemente von* E *sind zweielementige Teilmengen von* V*, die durch Paare* (u, v) *repräsentiert werden, es wird also nicht zwischen* (u, v) *und* (v, u) *unterschieden.*

Ein Knoten und eine Kante heißen ***inzident****, wenn einer der Endpunkte der Kante der Knoten ist. Zwei Knoten heißen* ***adjazent****, wenn sie durch eine Kante verbunden sind. Unter dem* ***Grad*** *eines Knotens versteht man die Anzahl der inzidenten Kanten. Falls alle Knoten vom gleichen Grad* k *sind, nennt man den Graphen* k*-regulär.*

Definition 2
Sei $G = (V, E)$ *ein Graph. Eine (gültige)* ***Färbung*** *von* G *ist eine Abbildung* $f \colon V \to \mathbb{N}$ *mit der Eigenschaft, daß adjazente Knoten* $u \in V$ *und* $v \in V$*,* $(u, v) \in E$*, verschieden gefärbt sind, d.h.* $f(u) \neq f(v)$ *gilt.*

Unter $|f| = |\{ f(x) \mid x \in V \}|$ *versteht man die Anzahl der in einer Färbung* f *verwendeten Farben.*

Für einen Graphen G *nennt man die minimale Anzahl an Farben einer Färbung die* ***chromatische Zahl****, bezeichnet mit* $\chi(G)$*.*

Definition 3
Eine Knotenteilmenge eines Graphen, in der alle Knoten paarweise adjazent sind, heißt ***Clique****. Die größtmögliche Mächtigkeit einer Clique in*

einem Graphen gibt dessen ***Cliquenzahl*** *an, die auch mit* $\omega(G)$ *bezeichnet wird.*

Offenbar müssen in einer gültigen Färbung alle Knoten einer Clique verschieden gefärbt sein, für alle Graphen G gilt also $\chi(G) \geq \omega(G)$.

Eine wichtige Klasse sind die Graphen von beschränkter Baumweite. Für eine Diskussion dieser Graphen, die auch die Aussage enthält, daß die chromatische Zahl für diese Graphen leicht bestimmt werden kann, sei etwa auf [Bod87] verwiesen.

Eine der als äquivalent erkannten Definitionen von Baumweite verwendet eine **Baumzerlegung**. Dabei handelt es sich um einen Baum, dessen Knoten aus Knotenteilmengen des Graphen bestehen, und der folgende Eigenschaften erfüllt:

- Jeder Knoten ist in mindestens einer Teilmenge enthalten.
- Für jede Kante gibt es mindestens eine Teilmenge, in der beide Endpunkte der Kante enthalten sind.
- Seien U und V zwei Teilmengen. Dann gibt es einen eindeutigen simplen Pfad zwischen U und V in dem Baum. Für jede Teilmenge W auf diesem Pfad gilt $U \cap V \subseteq W$.

Die größte Kardinalität einer Teilmenge gibt die Breite einer Baumzerlegung an. Die **Baumweite** eines Graphen ist die minimale Breite einer Baumzerlegung des Graphen. Wenn man anstatt eines Baumes einen Pfad, also einen Baum mit Maximalgrad 2 fordert, ergibt sich analog die **Pfadweite**.

2.2 Komplexitätstheorie

Ein wesentlicher Aspekt der Arbeit ist zu analysieren, wieviel algorithmischer Aufwand für die Lösung bestimmter Probleme notwendig ist. Da sich die meisten Berechnungsmodelle gegenseitig mit höchstens polynomiell verschlechterten Schranken für Laufzeit bzw. Speicherplatzbedarf simulieren können, werden hier polynomiell unscharfe Klassen verwendet, die auf einer Turingmaschine definiert sind. Dieses Vorgehen ist allgemein üblich, Details findet man z.B. in [HU79].

Die Laufzeit und der Speicherbedarf werden dabei immer in Abhängigkeit von der Eingabegröße betrachtet. Dazu muß man für Eingaben, die kein String sind, adäquate Kodierungen betrachten. In dieser Arbeit wird als Eingabegröße immer die Anzahl der Knoten im Graphen verwendet. Im Sinne der folgenden Klassen ist dies sicher statthaft. Eine konkrete Eingabe bzw. der Graph, den sie repräsentiert, wird dabei als Instanz bezeichnet.

Dementsprechend werden die folgenden bekannten Komplexitätsklassen betrachtet:

FP Die Klasse der in polynomieller Zeit deterministisch berechenbaren Funktionen.

P Die Klasse der in polynomieller Zeit deterministisch berechenbaren Entscheidungsprobleme.

NP Die Klasse der in polynomieller Zeit nichtdeterministisch berechenbaren Entscheidungsprobleme.

PSPACE Die Klasse der in polynomiellem Platz (deterministisch) berechenbaren Entscheidungsprobleme.

Tatsächlich macht eine polynomiell unscharfe Platzbeschränkung die Unterscheidung zwischen deterministischer und nichtdeterministischer Berechnung unnötig.

Dieses Resultat formuliert der Satz von Savitch.

Satz 4 (Savitch)
Sei $S\colon \mathbb{N} \to \mathbb{N}$ *eine voll raumkonstruierbare Funktion mit* $S(n) \geq \log_2(n)$. *Dann gilt* $\mathbf{NSPACE}(S) \subseteq \mathbf{DSPACE}(S^2)$.

Voll raumkonstruierbar bedeutet für eine Funktion S, daß es eine Turingmaschine gibt, die für alle Eingaben x genau $S(|x|)$ Bandzellen benutzt.

Beweis: Siehe etwa [HU79], Seite 301 ff. ■

Für viele Komplexitätsklassen sind die Fragen, ob die sich direkt aus der Definition der Maschinen ergebenden Inklusionen echt sind, berühmte offene Probleme. Trotzdem kann man die schwierigsten Probleme einer Klasse identifizieren.

Dazu betrachtet man die Möglichkeit, die Lösung für eine Instanz eines Problems A aus der Lösung für eine daraus konstruierte Instanz eines anderen Problems B zurückzugewinnen. Wenn der Rechenaufwand für diese Umwandlung vergleichsweise klein ist, kann man dann schließen, daß B nicht mit deutlich weniger Aufwand lösbar ist als A. Dieser Vorgang heißt **Reduktion**, A wird auf B reduziert.

Probleme, auf die alle Probleme aus einer Komplexitätsklasse reduziert werden können, heißen **hart** für die Klasse unter der verwendeten Reduktion. Dabei gibt es verschiedene Reduktionsbegriffe. Probleme, die zusätzlich in dieser Klasse enthalten sind, heißen für die Klasse **vollständig**.

Eine der wichtigen Eigenschaften solcher vollständiger Probleme ist, daß sie zeigen können, daß eine Klasse in der anderen enthalten ist, indem man für sie einen entsprechenden Algorithmus angibt. Wenn man etwa für ein beliebiges **NP**-vollständiges Problem einen deterministischen Polynomialzeitalgorithmus findet, kann dieser zur schnellen Lösung von allen Problemen in **NP** genutzt werden. Damit würde man $\mathbf{P} = \mathbf{NP}$ zeigen.

Daraus ergibt sich umgekehrt, daß unter der Voraussetzung $\mathbf{P} \neq \mathbf{NP}$ für kein **NP**-vollständiges Problem ein schneller deterministischer Algorithmus existiert.

Dies Betrachtungsweise ist seit einiger Zeit bekannt. Eine detailierte Darstellung findet man z.B. in [HU79].

2.3 Approximation

Definition 5
*Ein **Optimierungsproblem** Π ist durch folgende drei Komponenten charakterisiert:*

Instanzen D_Π*: die Menge der gültigen Eingabeinstanzen.*

Lösungen $S_\Pi(I)$*: die Menge aller zulässigen Lösungen für eine Eingabeinstanz* I.

Wert $m_\Pi \colon S_\Pi(I) \to \mathbb{Q}$*: eine Zielfunktion, die jeder zulässigen Lösung einen rationalen Wert zuweist.*

Ein Maximierungsproblem Π besteht darin, eine Lösung $\sigma^{I}_{opt} \in S(I)$ zu finden, für die gilt:

$$\forall \sigma \in S(I): \quad m_{\Pi}(\sigma^{I}_{opt}) \geq m_{\Pi}(\sigma) \ .$$

*Das **zugehörige Entscheidungsproblem** hat als zusätzlichen Eingabeparameter einen Wert s und stellt die Frage, ob es eine Lösung gibt, deren Wert größer oder gleich s ist:*

$$\exists \sigma \in S(I): \quad m_{\Pi}(\sigma) \geq s$$

Ein Minimierungsproblem wird analog definiert.

Unter der Voraussetzung $\mathbf{P} \neq \mathbf{NP}$ gibt es für Optimierungsprobleme, deren Entscheidungsversion **NP**-vollständig ist, keinen effizienten Algorithmus, genauer sind die Optimierungsfunktionen nicht in **FP**.

In dieser Situation versucht man, die Lösung mit effizienten Algorithmen, die höchstens polynomiellen Zeitbedarf haben, anzunähern. Dieses Vorgehen wird im folgenden exakt definiert.

Definition 6
Für ein Optimierungsproblem Π, einen passenden Algorithmus A und eine Instanz I heißt

$$R_A(I) := \max \left\{ \frac{A(I)}{\mathrm{OPT}(I)}, \frac{\mathrm{OPT}(I)}{A(I)} \right\}$$

*das **Approximationsverhältnis** von A bei Eingabe I. $\mathrm{OPT}(I)$ steht dabei für den Wert der Optimallösung.*

Dieses Maß muß nicht unbedingt eine Konstante sein. Oft wird auch eine Funktion in Abhängigkeit von der Eingabe(größe) verwendet.

Somit stellt sich für ein Problem nicht nur die Frage, wie schwierig die Optimallösung zu berechnen ist, sondern auch wie gut sie zu approximieren ist.

Ein Problem, das bereits unter diesem Aspekt sehr gut untersucht wurde, ist SETCOVER, das wie folgt definiert ist: Eine Instanz besteht aus einer endlichen Menge U, dem Universum, und einem Teilmengensystem $F = \{f_1, \ldots, f_m\} \subseteq 2^U$ über U. Die gültigen Lösungen bestehen aus einer Auswahl $A \subseteq F$ von Teilmengen, die U überdecken, also $\cup_{X \in A} X = U$. Die gesuchte Lösung ist eine gültige Teilmengenauswahl, die aus möglichst wenigen Mengen besteht.

Es stellt sich heraus, daß unter der Voraussetzung $\mathbf{P} \neq \mathbf{NP}$ SETCOVER für ein c nicht besser als auf einen Faktor $c \ln(|U|)$ approximierbar ist. [Fei96, RS97, AS97]

Der wesentliche Bestandteil eines **NP**-Hardness-Resultats ist fast immer eine Reduktion. Eine solche ist im allgemeinen nicht approximationserhaltend, d.h. daß sich Approximationsalgorithmen für das eine Problem nicht unbedingt auf das andere Problem übertragen lassen. Man kann Definitionen angeben, die diese Werkzeuge auch für Optimierungsprobleme und Approximationsgüten zur Verfügung stellen. Da hier aber nur Reduktionen verwendet werden, in denen sich die Qualitätsmaße eins zu eins übertragen, ist dieser Aufwand hier nicht nötig.

2.4 Kompetitive Analyse von Online-Strategien

Die Formulierung eines Problems als Online-Fragestellung trägt der Tatsache Rechnung, daß mitunter bereits Entscheidungen getroffen werden müssen, bevor die komplette Information über die Eingabe vorhanden ist.

Ein typisches Beispiel ist die Steuerung eines Aufzugs. Die Eingabe besteht aus einer Sequenz von Stockwerken und Zeiten, wobei die Zeit immer weiter fortschreitet. Die Ausgabe ist ein Bewegungsplan für den Aufzug, also etwa eine Reihenfolge, in der die Anfragen bearbeitet werden.

Wenn hier gewartet wird, bis alle Anfragen bekannt sind, bevor der Aufzug bewegt wird, ist die Antwort mit Sicherheit inzwischen irrelevant.

Einen guten Überblick über den theoretischen Hintergrund und viele Beispiele findet man in [OSH94a, OSH94b].

Eine kompetitive Analyse besteht darin, die Leistung eines Online-Algorithmus (auch als Online-Strategie bezeichnet) an der optimalen Offline-Lösung zu messen. Obwohl diese im allgemeinen unrealistisch gut ist, ist sie ein unabhängiger Maßstab.

Definition 7
*Ein deterministischer Online-Algorithmus $\mathcal{A}$ für ein Minimierungsproblem bzw. Maximierungsproblem heißt c-**kompetitiv**, wenn für alle Anfragefolgen σ die Ungleichung $C_{\mathcal{A}}(\sigma) \leq c \cdot OPT(\sigma)$ bzw. $C_{\mathcal{A}}(\sigma) \geq c \cdot OPT(\sigma)$ gilt.*

Dabei steht C für das zu optimierende Maß, also etwa maximale Wartezeit, und OPT für das Optimum dieses Maßes über alle möglichen Antworten für die gegebene Sequenz.

Wenn die Algorithmen, wie in dieser Arbeit, deterministisch sind, kann man die Antworten reproduzieren. Dementsprechend kann eine einzelne Anfragefolge zeigen, daß ein Algorithmus schlechter als c-kompetitiv ist. In dieser Situation kann man auch mit einem sogenannten Online-Gegenspieler argumentieren. Dieser stellt Online-Anfragen an den Algorithmus, und kann dabei auf die bisherigen Ausgaben reagieren. Da der Algorithmus deterministisch ist, ergibt sich so eine konkrete Eingabe, an der sich zeigt, daß der Algorithmus eine bestimmte Qualität nicht überschreiten kann.

3 Online-Frequenzzuteilung

Die Aufgaben, die sich beim Betrieb eines Mobilfunknetzes stellen, sind sehr vielseitig. Es ist nicht zu erwarten, daß man ein einziges universelles Modell finden kann, das alle möglicherweise interessanten Fragestellungen umfaßt. Daher versucht man, sich auf Fragen zu konzentrieren, die mit jedem vernünftigen Modell erfaßt werden müssen.

Dies führt zu dem sogenannten Online-Frequenzzuteilungs-Problem. Dabei konzentriert man sich auf das Phänomen der Interferenz, der physikalischen Tatsache, daß sich zwei Sender, die auf der gleichen Frequenz arbeiten, im gemeinsamen Empfangsgebiet gegenseitig stören.

Es wird ein sich verändernder Graph betrachtet, in dem für jede benötigte Verbindung ein Knoten vorhanden ist, und eine Kante, falls die Verwendung derselben Frequenz für diese Verbindungen zu Interferenz führen würde.

Ein einfaches Szenario ist also etwa, daß für jedes Handygespräch der näheste Sender ausgewählt wird. Für diese Verbindung muß dann eine Frequenz zur Verfügung stehen, die sich nicht mit anderen Verbindungen stört. Also wird zu allen anderen schon bestehenden Verbindungen, die nicht auf derselben Frequenz arbeiten dürfen, eine Kante eingefügt.

Auch kompliziertere Situationen, etwa die, daß zwischen dem Funkkanal vom Handy zur Feststation und umgekehrt unterschieden wird, sind in diesem Rahmen ohne weiteres modellierbar.

Selbst in dem Fall, daß eine Frequenz mehrere Kanäle aufnehmen kann, bleibt als Teilproblem die Aufgabe, den Kanalbündeln Frequenzen zuzuweisen.

In der theoretischen Formulierung wird eine Zuweisung von Frequenzen an die Knoten gesucht. Dies entspricht einer Färbung, und scheint somit zu dem bekannten Problem des Online-Node-Coloring in einem Graphen zu führen. Diese Modell ist etwa in [KVV90, Vis92, Ira94] unter verschiedenen Gesichtspunkten untersucht worden, und wie folgt formuliert:

jedem der neu auftretenden Knoten wird sofort eine Farbe zugewiesen, die nachträglich nicht mehr geändert werden darf. Dies ist insofern unrealistisch, als die im Mobilfunk verwendete Hardware immer die Möglichkeit eines sich bewegenden Handies vorsieht. Daher kann davon ausgegangen werden, daß ein Frequenzwechsel ohne Unterbrechung des Gesprächs möglich ist.

Wenn man auf der anderen Seite davon ausgeht, daß die Möglichkeit besteht, allen Verbindungen gleichzeitig eine neue Frequenz zuzuteilen, verliert die Aufgabenstellung den Online-Charakter. Dann kann nämlich immer wieder eine Färbung errechnet werden, auf die dann das ganze Netz gleichzeitig umgestellt wird. Dabei entsteht allerdings die Notwendigkeit, das Netz zu synchronisieren.

Dies führt zu einem Modell, in dem nur sequentielle Frequenzwechsel erlaubt werden. D.h., daß Frequenzen grundsätzlich geändert werden dürfen, aber nicht gleichzeitig und nur so, daß zu keinem Zeitpunkt Interferenz entsteht. Dabei wird die Anzahl dieser Umfärbeschritte nicht begrenzt, man ist im allgemeinen aber natürlich an möglichst kurzen Sequenzen interessiert, damit das neue Gespräch möglichst schnell angenommen werden kann. Es ist vorstellbar, daß kurzfristig auch weitere Frequenzen benutzt werden.

In der Sprache der Graphentheorie entsteht daraus das folgende abstrakte Problem:

Es wird ein Graph Knoten für Knoten eingegeben, jeweils zusammen mit den inzidenten Kanten. Für jeden neuen Knoten sind zunächst beliebig viele Umfärbeschritte in dem bereits eingegebenen Graphen erlaubt, dann wird dem neuen Knoten eine Farbe zugewiesen. Die Färbung des Graphen muß dabei zu jedem Zeitpunkt gültig sein. Die Anzahl der Farben gibt die Anzahl der zur Verfügung stehenden Frequenzen an und ist i.a. fix beschränkt. Genauso kann ein eingegebener Knoten wieder gelöscht werden. Auch in diesem Fall dürfen beliebig Umfärbeschritte vorgenommen werden. Die Eingabe eines Knoten kann als Anfrage interpretiert werden. Diese wird entweder durch eine passende Umfärbesequenz zusammen mit einer freien Farbe für den neuen Knoten erfüllt oder muß verworfen werden. Somit entsteht die Aufgabe, möglichst viele dieser Anfragen zu erfüllen. Dieses Problem wird als relaxiertes ONLINENODECOLORING bezeichnet.

Für eine einzelne Anfrage entstehen so die Fragen nach Existenz und Länge einer Umfärbesequenz sowie nach der Anzahl der dabei temporär zusätzlich benötigten Farben.

In diesem Zusammenhang erweist es sich als sinnvoll, das etwas allgemeinere Problem einer Graphumfärbung zu betrachten, das wie folgt formuliert ist:

Gegeben sei ein Graph und zwei gültige (Knoten-) Färbungen, sowie die maximale Anzahl von zusätzlichen Farben und die maximale Länge einer Umfärbesequenz. Die Frage, ob es eine entsprechend kurze Folge von gültigen Färbungen des Graphen gibt, die sich jeweils nur an einem Knoten unterscheiden, ist das als GRAPHRECOLORING bezeichnete Problem.

Erweiterungen, in denen mehrere Knoten gleichzeitig die Farbe ändern dürfen, sind vorstellbar.

Diesem zentralen Problem, das sich auch von einem theoretischen Standpunkt aus als interessant erweist, sind daher die folgenden Abschnitte gewidmet.

3.1 Graphumfärbung

Die im folgenden vorgestellten neuen Ergebnisse habe ich im Laufe dieser Arbeit gesammelt. Sie sind möglicherweise auch auf andere Gebiete anwendbar. Der Ansatz, den Raum der Lösungen eines Problems als Graphen zu betrachten, und Pfade darin zu untersuchen, scheint im allgemeinen zu interessanten Fragestellungen zu führen. Typisch für dieses Vorgehen, in einem Graphen von Lösungen, durch verschiedenen Mechanismen geleitet Pfade zu verfolgen, sind z.B. lokale Suchverfahren.

Das einzige wesentliche Resultat, auf das aufgebaut wird, ist die Aussage über reversible Berechnungen von Bennett in [Ben89].

Definition 8
*Unter einer **Umfärbesequenz** versteht man eine Folge von Färbungen, die sich jeweils nur an einem Knoten unterscheiden. Die Anzahl der in ihr benötigten Farben ist das Maximum der verwendeten Farben bezüglich aller Färbungen der Sequenz. Zwei Färbungen, die in einer Umfärbesequenz ohne zusätzliche Farben vorkommen, heißen **äquivalent für das Umfärbeproblem**.*

3.2 Anzahl zusätzlicher Farben

Satz 9
Sei $G = (V, E)$ *ein Graph, und* $h, g \colon V \to \mathbb{N}$ *Färbungen von* G. *Durch* $k = |g(V) \cap h(V)|$ *sei die Anzahl der gemeinsamen Farben bezeichnet. Dann genügen* $k - 1$ *zusätzliche Farben für eine Umfärbesequenz von* h *nach* g *oder umgekehrt. Deren Länge ist linear in der Anzahl der Knoten des Graphen beschränkt.*

Beweis: O.E. wird angenommen, daß alle gemeinsamen Farben durch die Menge $g(V) \cap h(V) = \{1, \ldots, k\}$ gegeben sind, und es ein K gibt, so daß $\{K+1, \ldots, K+k\} \cap (g(V) \cup h(V)) = \emptyset$ gilt.

Beschrieben wird eine Umfärbung von h nach g. Als Zwischenfärbung wird folgendes f verwendet:

$$f(v) = \begin{cases} g(v) + K & \text{falls } g(v) < k \\ g(v) & \text{falls } g(v) > k \\ h(v) & \text{falls } g(v) = k \end{cases} .$$

Diese Färbung benutzt $k - 1$ zusätzliche Farben.

Knotenweises Umfärben zu f ist möglich, da die neuen Farben in der alten Färbung nicht vorkommen, und die umgefärbten Knoten wieder Teil einer gültigen Färbung sind.

Von f aus werden zunächst die Knoten, für die $g(v) = k$ gilt, mit Farbe k gefärbt, was möglich ist, da es sich um eine Farbklasse handelt, und keiner der übrigen Knoten mit k gefärbt ist.

Dann werden die Knoten $g(v) < k$ mit $g(v)$ gefärbt. Dies ist möglich, da diese Farben am Anfang nicht, und dann nur gemäß g verwendet werden.

Dieser Vorgang benötigt höchstens $2n$ Umfärbungsschritte, wenn n die Anzahl der Knoten im Graph bezeichnet. ∎

Wie der folgende Satz zeigt, ist diese Grenze im allgemeinen nicht zu verbessern.

Satz 10
Für jedes $k \in \mathbb{N}$ *gibt es einen Graphen, der zwei verschiedene* k*-Färbungen besitzt, so daß jede Umfärbungssequenz mindestens* $k - 1$ *zusätzliche Farben benutzen muß.*

Beweis: Der Graph besteht aus k^2 Knoten, die als Gitter angeordnet sind, also $V = \{1, \ldots, k\}^2$. Zwischen allen Knoten, die nicht in einer Zeile und nicht in einer Spalte liegen, gibt es eine Kante, also

$$E = \{ ((i, j), (p, q)) \in V^2 \mid i \neq p \text{ und } j \neq q \} .$$

Die zwei Färbungen haben als Farbklassen die Zeilen bzw. Spalten. Formal also die Spaltenfärbung $s(x, y) = x$ bzw. die Zeilenfärbung $z(x, y) = y$. Offenbar sind beides gültige Färbungen. Da z.B. die Diagonale eine Clique der Größe k bildet, sind beide Färbungen optimal, was nicht unbedingt gefordert ist.

Falls in einer Zeile zwei Knoten dieselbe Farbe haben, kann diese in keiner anderen Zeile vorkommen. Man sagt, die Farbe *gehört* zu dieser Zeile.

Betrachtet wird eine beliebige Umfärbesequenz von der anfänglichen Zeilenfärbung hin zur Spaltenfärbung. Dann gehört am Anfang zu jeder Zeile eine Farbe. Da am Ende in der Spaltenfärbung keine Farbe mehr zu einer Zeile gehört, gibt es in der Umfärbesequenz den Zeitpunkt, an dem es zum ersten Mal eine Zeile gibt, zu der keine Farbe gehört. Sei dies o.E. die oberste Zeile. In dieser werden also k verschiedene Farben benutzt. Zu allen anderen Zeilen gehört noch mindestens eine Farbe. Diese sind untereinander und zu denen in der obersten Zeile benutzten verschieden. Also sind zu diesem Zeitpunkt $k + k - 1$ verschiedene Farben nötig. ■

3.3 Kompetitive Analyse

Um zur eigentlichen Aufgabenstellung zurückzukommen, muß zunächst das betrachtete Problem definiert werden.

Definition 11
Bei dem ***relaxierten*** **ONLINE-NODE-COLORING** *Problem wird ein Graph knotenweise eingegeben. Einzelne Knoten können gelöscht werden. Der Online-Algorithmus muß gültig färben. Dabei hat er die Möglichkeit, einzelne Knoten umzufärben, wenn dabei zwischendurch keine ungültigen Färbungen entstehen.*

Die Anzahl der von einer Online-Strategie benötigten Farben ergibt sich relativ direkt aus den Resultaten über die Anzahl der zum Umfärben benötigten Farben.

Satz 12
Es gibt eine deterministische Strategie für das relaxierte ONLINE-NODE-COLORING *Problem, die bezüglich der Anzahl der verwendeten Farben 2-kompetitiv ist.*

Beweis: Es wird für jede Anfrage ein neues, optimales Coloring ausgerechnet. Dieses verwende k Farben. Mit dem Algorithmus aus Satz 9 kann der Graph mit $k-1$ zusätzlichen Farben zu diesem hin umgefärbt werden. Da jede gültige Lösung die entstehenden Graphen färben muß, benötigt sie mindestens k Farben. So ergibt sich der angegebene kompetitive Faktor. ∎

Dieser Online-Algorithmus hat, wie der folgende Satz zeigt, bereits die optimale Kompetitivität.

Satz 13
Für jedes $\epsilon > 0$ gilt: Es kann keine deterministische Strategie für das relaxierte ONLINE-NODE-COLORING *Problem geben, die bezüglich der Anzahl der verwendeten Farben besser als $(2-\epsilon)$-kompetitiv ist.*

Beweis: Es wird sukzessive der gitterartige Graph aus Satz 10 der Größe k eingegeben, der entweder zeilenweise oder spaltenweise optimal gefärbt werden kann. Dabei sei k so gewählt, daß $\epsilon > 1/k$ gilt. Da es sich um einen deterministischen Algorithmus handelt, kann der Online-Gegenspieler darauf eingehen, wie der Graph bisher gefärbt wurde. Man ist also frei die Namen für Zeilen und Spalten zu vertauschen. Dementsprechend kann man o.E. davon ausgehen, daß sich die deterministische Strategie für eine Zeilenfärbung entscheidet. Dann werden weitere k Zeilen angefügt. Solange der Graph zeilenorientiert gefärbt ist, wird für jede neue Zeile eine neue Farbe benötigt. Da aber auch für die Umfärbung zur Spaltenorientierung insgesamt $2k-1$ Farben benötigt werden, muß die deterministische Strategie irgendwann $2k-1$ Farben gleichzeitig verwenden. Somit ergibt sich

$$\frac{2k-1}{k} = 2 - \frac{1}{k} > 2 - \epsilon\ ,$$

und damit die Aussage des Satzes ∎

Ein anderes wichtiges Maß für die Qualität der Frequenzzuteilung ist die Frage, wieviele Gespräche bei einer fest vorgegebenen Schranke für die Anzahl der zur Verfügung stehenden Frequenzen korrekt bearbeitet werden können.

Auch wenn die Möglichkeit besteht einzelne Frequenzen im nachhinein zu ändern, kann kein Online-Algorithmus eine konstante Kompetitivität garantieren.

Satz 14
Keine Online-Strategie für das relaxierte ONLINE-NODECOLORING *Problem, die mit* k *Farben auskommt, hat einen kompetitiven Faktor* $\epsilon > 0$ *bezüglich der Anzahl an korrekt gefärbten Knoten.*

Beweis: Der Online-Gegenspieler gibt zunächst zwei voneinander unabhängige k-Cliquen ein. Diese werden von dem deterministischen Algorithmus mit im ganzen k Farben gefärbt. Dann sucht sich der Gegenspieler zwei Knoten, die in den verschiedenen Cliquen sind, und nicht dieselbe Farbe haben. Er gibt einen neuen Knoten ein, der mit allen außer den zwei ausgewählten Knoten adjazent ist. Dieser Knoten kann von dem Algorithmus nicht bearbeitet werden, da es für jede Farbe einen Nachbarn gibt, der mit dieser Farbe gefärbt ist. Da es sich außerdem um Cliquen der Größe k handelt, kann auch an keinem Knoten die Farbe geändert werden. Der optimale Offline-Algorithmus hingegen kann die beiden Cliquen so färben, daß alle Knoten, die mit dem neuen verbunden sind, mit Farben aus $\{1, \ldots, k-1\}$ gefärbt sind, so daß der neue Knoten mit k gefärbt werden kann. Diese Anfrage wird wiederholt, es wird immer wieder ein Knoten eingegeben, der mit den oben beschriebenen Knoten verbunden ist. Somit ergibt sich für n eingegebene Knoten der kompetitive Faktor zu $\frac{2k}{n}$, was für ausreichend großes n kleiner als jedes $\epsilon > 0$ ist. ■

In der Anwendung ergibt sich sofort die folgende Aussage:

Korollar 15
Sei $\epsilon > 0$. *Dann kann keine Online-Frequenzzuteilungs-Strategie, die mit einem festen Frequenzspektrum arbeitet, bezüglich der Anzahl der angenommenen Gespräche* ϵ*-kompetitiv sein.*

Dies steht im Kontrast zu den Ergebnissen in Kapitel 3.5, wo für eingeschränkte Graphklassen optimale Strategien angegeben werden.

Es zeigt sich, daß die Aufgabe, ein fest vorgegebenes Frequenzspektrum optimal auszuschöpfen, zusätzlich zu den Schwierigkeiten, die sich aus dem nicht vollständigen Wissen in einer Online-Situation ergeben, auch bezüglich der benötigten Berechnungen sehr aufwendig ist.

So stellt sich unter anderem die Frage, ob von einer beliebigen aktuellen Färbung ausgehend weitere Knoten gefärbt werden können, bzw. ob für eine bestimmte aktuelle Situation der Frequenzverteilung eine weitere Anfrage angenommen werden kann. Dies ist insofern eine relevante Frage, als das System durch entsprechende Anfragen in jeden beliebigen zulässigen Zustand gebracht werden kann.

Satz 16
Sei G *ein Graph und* f *eine Färbung von* G*. Dann gibt es eine Sequenz von Anfragen für das relaxierte* ONLINE-NODECOLORING *Problem ohne zusätzliche Farben, die dafür sorgt, daß* G *mit einer Färbung entsteht, die für das Umfärbeproblem äquivalent zu* f *ist.*

Beweis: Zunächst wird eine Clique der Größe $|f|$ erzeugt, deren Farben nicht mehr geändert werden können. Dann wird der Graph knotenweise eingefügt, wobei zusätzliche Verbindungen zur Clique die gewünschte Farbe erzwingen. In diesem Graph kann keine einzige Farbe geändert werden. Zum Schluß werden die Knoten der Clique gelöscht. Alle Umfärbungen, die in dieser Phase möglich sind, sind sicher auch am Ende möglich. ■

Die zentrale Aussage, die hier als nächstes entwickelt wird, ist, daß es **PSPACE**-vollständig ist zu entscheiden, ob ein nächster Knoten durch entsprechendes Umfärben bearbeitbar ist oder nicht. Es stellt sich heraus, daß Umfärbeschritte in einem gewissen Sinn Berechnungsschritten einer Turingmaschine entsprechen können. Im Gegensatz zu einer Turingmaschine, für die sich aus einer Konfiguration eine nächste ergibt, kann beim Umfärben keine Richtung erzwungen werden. Daher muß man sich zunächst mit platzbeschränkten symmetrischen Turingberechnungen beschäftigen.

3.4 Simulation von Berechnungen

Im folgenden wird gezeigt, daß die Berechnung einer platzbeschränkten Turingmaschine durch ein Graphumfärbungsproblem, in dem keine zusätzlichen Farben erlaubt sind, simuliert werden kann. Die Größe des Graphen hängt dabei linear von der Platzschranke ab.

Dies hat folgende Konsequenzen:

- Die Länge einer Umfärbesequenz kann im allgemeinen exponentiell von der Größe des Graphen abhängen.
- Die Frage nach der Existenz einer Umfärbesequenz ist **PSPACE**-hart.

- Die Länge einer Umfärbesequenz ist genauso schwer zu approximieren wie die Dauer einer platzbeschränkten Berechnung.

Es zeigt sich, daß sich aus dem letzten Punkt ein sehr starkes Nicht-Approximierbarkeits-Resultat ergibt.

Dieses neue Resultat ergibt sich über mehrere Zwischenschritte und mündet schließlich in die Frage, ob durch Umfärben einzelner Knoten erreicht werden kann, daß ein neuer Knoten mit einer bereits verwendeten Farbe eingefärbt werden kann.

Die verschiedenen Schritte dieser Umwandlung, die auch für sich eigenständige, neue Probleme und interessante Ergebnisse darstellen, werden in den nächsten Abschnitten entwickelt.

Eine Grundschwierigkeit bei der Simulation einer Berechnung ist die unvermeidbare Symmetrie einer Umfärbung. Wenn es eine Sequenz von A nach B gibt, so gibt es sicher auch eine von B nach A. Diese Symmetrie ist für eine Turingmaschine nicht selbstverständlich, es ist vielmehr nötig, diesen Begriff exakt zu fassen.

Diesen technischen Schwierigkeiten ist der nächste Abschnitt gewidmet.

3.4.1 Symmetrische Berechnungen

Aus der Komplexitätstheorie ist schon länger bekannt, daß deterministische Berechnungen durch reversible simuliert werden können, und dabei die Laufzeit nur um ein Polynom in der Eingabelänge verlängert wird. Dieses Ergebnis ist in [Ben89] dargestellt und wird der Vollständigkeit halber hier kurz dargestellt.

Zunächst muß der Begriff der Reversibilität genau erklärt werden.

Definition 17
*Eine Turingmaschine M heißt **umkehrbar**, falls sie deterministisch ist, und es für jede Konfiguration höchstens eine Vorgängerkonfiguration gibt.*

$$(A \overset{M}{\underset{1}{\Longrightarrow}} C \ \wedge\ B \overset{M}{\underset{1}{\Longrightarrow}} C) \Rightarrow (A = B)$$

Dies erzwingt, daß aus jeder Konfiguration der Maschine die Eingabe rekonstruierbar ist. Daher kann eine solche Maschine niemals mit leerem Band anhalten. Um diesem Problem aus dem Weg zu gehen, hält die Maschine immer mit der Eingabe auf dem Arbeitsband.

Satz 18
Zu jeder deterministischen Turingmaschine, die in Zeit T *und Platz* S *arbeitet, gibt es eine äquivalente Zweibandturingmaschine, die umkehrbar ist und in Zeit* $O(T^2)$ *und Platz* $O(S \log T)$ *arbeitet.*

Insbesondere benötigt diese Simulation nur polynomiell mehr Platz als die Originalberechnung.

Beweis: Dies ist die zentrale Aussage von [Ben89], deren Beweis hier nur skizziert wird.

Zunächst wird festgestellt, daß durch ein zusätzliches Band, auf dem die Arbeit protokolliert wird, Reversibilität erzwungen werden kann. Um nun genau die Eingabe und das erwünschte Ergebnis der Berechnung zu erzeugen, wird eine sogenannte Brücke verwendet. Dabei wird erst die durch das neue Band reversibel gemachte Berechnung ausgeführt, dann wird das Ergebnis kopiert, was reversibel möglich ist, und dann wird durch Umkehren der ursprünglichen Berechnung die Eingabe wieder erzeugt.

Da hier das zusätzliche Band im allgemeinen zuviel Platz benötigt, werden hierarchisch Checkpoints in Form von gespeicherten Konfigurationen verwaltet, zwischen denen dann nach obiger Idee gearbeitet werden kann. Dadurch werden nur in der Rechenzeit logarithmisch viele dieser Checkpoints benötigt, was zu dem angestrebten Faktor beim Platzverbrauch führt.

Von der verwendeten Konstruktion wird im folgenden benötigt, daß sie, um den Begriff der lokalen Reversibilität fassen zu können, immer abwechselnd Bandveränderungs- und Kopfbewegungsschritte ausführt. In einem **Bandveränderungsschritt** werden die Kopfpositionen nicht verändert, es wird also bei einem Zustandsübergang der Bandinhalt an den Kopfpositionen verändert. In einem **Bewegungsschritt** werden unabhängig vom Bandinhalt bei einem Zustandsübergang die Köpfe bewegt. Dadurch wird die Eigenschaft der Umkehrbarkeit in den Regeln erkennbar. ■

Diese Umkehrbarkeit muß noch adäquat in lokale Symmetrie verwandelt werden. Dies ist insbesondere wegen der zwei Bänder nicht ohne weiteres möglich.

3.4.2 Wortersetzungssysteme

Als Zwischenstufe soll hier eine verallgemeinerte Regelgrammatik vorgestellt werden. Der Unterschied zu einer Grammatik besteht darin, daß in diesem formalen Mechanismus nicht zwischen Terminalen und Nichtterminalen unterschieden wird und auf das Startsymbol verzichtet werden kann. Stattdessen wird ein Start- und ein Zielwort angegeben und die Frage gestellt, ob man durch Regelanwendungen das eine in das andere umwandeln kann.

Definition 19
*Ein **normalisiertes Wortersetzungssystem** $S = (\Sigma, R)$ besteht aus einem endlichen Alphabet Σ und einer Relation $R \subseteq (\Sigma^2)^2$.*

Zwei Wörter $u, v \in \Sigma^*$ heißen durch S ineinander überführbar, $u \overset{S}{\Rightarrow} v$, falls sie mit $\alpha, \omega \in \Sigma^*$ als $u = \alpha a b \omega$ und $v = \alpha c d \omega$ darstellbar sind, und $(ab, cd) \in R$ gilt.

Es wird die reflexive transitive Hülle von R gebildet. Dementsprechend heißen zwei Wörter in dem System ineinander umwandelbar, falls es eine Sequenz von Wörtern mit entsprechendem Anfang und Ende gibt, die jeweils in einem Schritt innerhalb des Systems ineinander überführt werden können.

*S heißt **symmetrisch**, falls R symmetrisch ist, also für alle $a, b, c, d \in \Sigma$ die Beziehung $(ab, cd) \in R \Rightarrow (cd, ab) \in R$ gilt.*

*S heißt **verändernd**, falls $(ab, cd) \in R \Rightarrow (a \neq c) \wedge (b \neq d)$ gilt.*

Offenbar kann sehr leicht erzwungen werden, daß das System verändernd ist. Für eine Regel der Form $AB \to AC$ werden zwei neue Alphabetszeichen α und β eingeführt. Zusammen mit den neuen Regeln $AB \to \alpha\beta$ und $\alpha\beta \to AC$ ergibt sich ein veränderndes Wortersetzungssystem, das genau dieselben Wörter über dem Originalalphabet ineinander überführen kann. Auch die Länge einer kürzesten Sequenz verdoppelt sich dabei höchstens.

Ein solches symmetrisches Wortersetzungssystem soll nun zur Simulation einer umkehrbaren Zweibandberechnung verwendet werden. Es wird also ein solches System zusammen mit Anfangs- und Zielwort angegeben, die genau dann ineinander umgewandelt werden können, wenn die Berechnung akzeptiert. Darüberhinaus läßt sich aus der Sequenz im Wortersetzungssystem die Abfolge der Berechnung zurückgewinnen. Ein Schritt der Maschine wird durch mehrere (linear im Platzbedarf der Maschine) Wortersetzungen simuliert.

Lemma 20
Ein symmetrisches veränderndes Wortersetzungssystem kann die Berechnung einer umkehrbaren k-Band-Maschine simulieren. Die Länge der Sequenz ist polynomiell in der Dauer der Berechnung, die Länge der zu betrachtenden Strings ist durch den Platzbedarf der Maschine beschränkt.

Beweis: Es genügt Regeln anzugeben, die eindeutig von einer Konfiguration zu einer anderen führen, falls dies einem Schritt der Maschine entspricht. Da eine Sequenz von Konfigurationen einer umkehrbaren Berechnung, falls sie existiert, eindeutig ist, ist auch der Pfad im Wortersetzungssystem eindeutig. Daher kann die Relation R schadlos symmetrisch ergänzt werden. Dadurch wird lediglich ein nutzloses Rückwärtsgehen auf dem eindeutigen Pfad im Konfigurationengraphen erlaubt.

Satz 18 erlaubt es, sich auf spezielle Turingmaschinen zu konzentrieren. Diese führen immer abwechselnd einen Bandveränderungs- und einen Bewegungsschritt durch. Dabei hängen die Bewegungen nicht vom Bandinhalt ab. So zeigt sich die Umkehrbarkeit der Berechnung direkt an den Regeln, es reicht aus festzustellen, daß nie zwei rechte Seiten eines Übergangs gleich sind.

Die Konstruktion hier orientiert sich an den einfachen Standardkonstruktionen zur Simulation einer k-Bandmaschine auf einer Einbandmaschine. Es werden k Spuren für den Bandinhalt, sowie zusätzliche Spuren für Kopfpositionsmarker und neue Zustände vorgesehen. Die neuen Zustände sind $(k+1)$-Tupel bestehend aus altem Zustand und k Zeichen. Dabei ist eines der Zeichen ein neues Leerzeichen. Außerdem werden für jeden Bewegungsschritt zwei eindeutige zusätzliche Zustände erzeugt, die für zwei Phasen stehen, in die der Bewegungsschritt aufgeteilt wird.

Dies alles ist offenbar durch eine endliche Zustandsmenge realisierbar. Zur Simulation eines Schrittes wird immer über das ganze Band gelaufen, der dabei verwendete 'Arbeitskopf' ist durch die (einzige) Stelle gegeben, an der auf der Aktionsspur ein neuer Zustand gespeichert ist.

Die Bandveränderungsschritte werden in eine Vor- und eine Nachbereitungsphase, Phase 1 und 2 aufgeteilt. Zum Vorbereiten wird der aktuelle Bandinhalt in den Zustand aufgenommen. Dazu wird von links nach rechts über das Band gelaufen, die Zeichen an den Kopfpositionen werden durch Leerzeichen ersetzt. Dann wird am rechten Rand mit Hilfe der gespeicherten Zeichen der eigentliche Zustandsübergang ausgeführt, statt sofort auf das Band zu schreiben werden die gespeicherten Zeichen entsprechend verändert. Auf dem Rückweg, der 2. Phase, werden die neuen

Inhalte in das Band übertragen, der Zustand ist dann wieder mit k Leerzeichen versehen.

Auch die Bewegungsschritte werden in zwei Phasen, Phase 3 und 4, aufgeteilt. Vor der 3. Phase wird vom reinen Zustand in einen Zustand, der für den Übergang steht und die auszuführenden Kopfbewegungen zeigt, übergegangen. Dies ist möglich, da Bewegungsschritte nur vom aktuellen Zustand abhängen. Dieser Zustand läuft in der dritten Phase von links nach rechts über das Band und führt Kopfbewegungen nach links aus. Am rechten Rand wird der Zustand markiert, so daß er in der 4. Phase von rechts nach links laufend die Kopfbewegungen nach rechts ausführen kann. Abschließend wird am linken Rand der Endzustand des ursprünglichen Bewegungsschrittes erzeugt. Dieser Zustand ist mit Leerzeichen markiert und befindet sich am linken Rand des Bandes.

Damit sind die Voraussetzungen für die 1. Phase wieder erfüllt, das nächste Paar von Originalschritten wird in vier Phasen simuliert.

Es muß noch gezeigt werden, daß dieses Vorgehen reversibel ist. Dies ist für die Zustandsübergänge zwischen den Phasen, die an den Bandrändern stattfinden, ein Ergebnis von Satz 18. Dabei ist auch das Aufteilen der Bewegungsschritte unproblematisch. Ein Zustand ist entweder Start oder Ziel eines solchen Schrittes. Je nachdem kann er in den markierten oder unmarkierten zu diesem Schritt gehörenden Zustand verändert werden. Diese können dann nur ineinander am anderen Rand verwandelt werden.

Der Vollständigkeit halber und um die Reversibilität des Regelsystems zeigen zu können, werden die notwendigen Regeln im einzelnen skizziert. Da sich 1. und 2. bzw. 3. und 4. Phase bis auf die Richtung gleichen, genügt es, jeweils für eine davon Regeln aufzuzeigen und diese auf Reversibilität zu untersuchen. Das oben beschriebene erweiterte Alphabet wird wie folgt dargestellt:

$$\begin{bmatrix} a_+ \\ b \\ Q(\sqcup, c) \end{bmatrix}$$

Dies bedeutet, daß auf der ersten Spur ein a gespeichert ist, und auf dieser Position der Kopf des ersten Bandes steht (a_+); auf der zweiten Spur steht ein b, der Zustand ist ein Q, der Inhalt an der Kopfposition des ersten Bandes ist noch nicht ausgelesen ($\sqcup$), der Inhalt an der Kopfposition des zweiten Bandes ist ausgelesen (und durch ein Leerzeichen ersetzt), und war c.

Im folgenden werden die Regeln lediglich für eine Spur angegeben. Sie müssen mit allen Kombinationen von der beschriebenen Veränderung und reinen Bewegungsschritten ausgeführt werden. Dabei bleibt die Reversibilität erhalten.

Derartige reine Bewegungsschritte entsprechen Regeln der folgenden Gestalt:

$$\begin{bmatrix} a \\ Q(x) \end{bmatrix} \begin{bmatrix} b \\ . \end{bmatrix} \rightarrow \begin{bmatrix} a \\ . \end{bmatrix} \begin{bmatrix} b \\ Q(x) \end{bmatrix}$$

Diese müssen mit allen unten angegebenen Regeln entsprechend kombiniert werden.

Für die 1. Phase gibt es Regeln der Form

$$\begin{bmatrix} a_+ \\ Q(\sqcup) \end{bmatrix} \begin{bmatrix} b \\ . \end{bmatrix} \rightarrow \begin{bmatrix} \sqcup \\ . \end{bmatrix} \begin{bmatrix} b \\ Q(a) \end{bmatrix}$$

Es werden also bei einer Bewegung des neuen Zustands nach rechts Zeichen aus den Kopfpositionen der alten Bänder ausgelesen.

Zwischen 1. und 2. Phase wird der zugrundeliegende Bandveränderungsschritt ausgeführt. Dieser gehe von Q nach Q' über und ändere dabei die Bandinhalte auf dem ersten Band von a nach x und auf dem zweiten von b nach y. Dies wird durch eine Regel der folgenden Gestalt verwirklicht:

$$\begin{bmatrix} \$ \\ Q(a,b) \end{bmatrix} \# \rightarrow \begin{bmatrix} \$ \\ Q'(x,y) \end{bmatrix} \#$$

Dabei steht \$ für eine Bandende-Markierung auf allen Spuren für simulierte Bänder, und # für das Ende des neuen Arbeitsbandes.

Die zweite Phase ergibt sich aus den Regeln der ersten Phase von rechts nach links gelesen, da hier die Bandinhalte an die Kopfpositionen zurückübertragen werden. Daher ist es nicht nötig in den Zuständen mitzuverfolgen ob in 1. oder 2. Phase gearbeitet wird.

Dabei ist gewährleistet, daß alle Regeln sowohl in den rechten als auch in den linken Seiten eindeutig sind. Dies liegt zum einen an dem erweiterten Zustand, und zum anderen an den mit dem neuen Leerzeichen belegten Positionen.

Zwischen Phase 2 und 3 wird am linken Rand in den ersten der neuen Zustände übergegangen, der für die durch den Zustand eindeutig bestimmten Bandbewegungen steht.

Für die 3. Phase werden, neben den reinen Bewegungen, Regeln der folgenden Form verwendet:

$$\begin{bmatrix} a_+ \\ Q \end{bmatrix} \begin{bmatrix} b \\ . \end{bmatrix} \rightarrow \begin{bmatrix} a \\ . \end{bmatrix} \begin{bmatrix} b_+ \\ Q \end{bmatrix}$$

Diese sorgen dafür, daß der Arbeitskopf auf dem Band nach rechts bewegt wird. Am rechten Rand wird zwischen Phase 3 und 4 in den zweiten neuen Zustand für den Bewegungsschritt übergegangen. In der 4. Phase werden dann analog zu Phase 3 die Kopfbewegungen nach links ausgeführt:

$$\begin{bmatrix} a \\ . \end{bmatrix}\begin{bmatrix} b_+ \\ Q' \end{bmatrix} \to \begin{bmatrix} a_+ \\ Q' \end{bmatrix}\begin{bmatrix} b \\ . \end{bmatrix}$$

Nach der 4. Phase wird dann am linken Rand in den Endzustand des Bewegungsschrittes übergegangen, so daß wieder mit Phase 1 fortgefahren werden kann.

Diese Regeln sind sowohl durch ihre rechten als auch durch ihre linken Seiten eindeutig bestimmt.

Zwischen den verschiedenen Arten von Regeln kann es keine gemeinsamen rechten oder linken Seiten geben, weil jeweils verschiedene Arten von Zuständen beteiligt sind.

Aus dem Ergebnis von Bennett [Ben89] ergibt sich, daß als Endkonfiguration ein akzeptierender Zustand und die Eingabe angenommen werden kann. Somit sind auch Start- und Zielwort mit einer sehr einfachen Berechnung bestimmbar. ■

Satz 21
Die Frage, ob zwei Wörter in einem Wortersetzungssystem ineinander umgewandelt werden können, ist im Sinne einer Log-Space-Reduktion **PSPACE**-*vollständig.*

Beweis: Die Frage ist sicher mit einer nichtdeterministischen linear platzbeschränkten Maschine beantwortbar. Diese muß lediglich immer wieder eine Regel und eine Position raten, dort die Ersetzung auszuführen versuchen und anschließend testen, ob das Zielwort bereits erreicht ist. Mit Satz 4 (Savitch) ergibt sich, daß es eine äquivalente deterministische Maschine mit Platzbedarf n^2 gibt.

Andererseits hängen die Regeln, die in Lemma 20 beschrieben werden, nur von der Turingmaschine ab, können also sicherlich ohne Platzbedarf ausgegeben werden. Bei der Umwandlung des Eingabewortes in Start- und Zielkonfiguration wird dieses lediglich um Anfangs-, Ende- und Füllzeichen ergänzt. All dies ist sicherlich mit logarithmischem Platz berechenbar. ■

Satz 22
Es gelte $\mathbf{P} \neq \mathbf{PSPACE}$. *Dann ist das Optimierungsproblem, die Länge einer linear platzbeschränkten Berechnung zu minimieren, in deterministischer polynomieller Zeit für kein* $\delta > 0$ *auf einen Faktor* $2^{\delta n}$ *approximierbar.*

Beweis: Im Widerspruch zur Behauptung sei das Problem auf einen Faktor $2^{\delta n}$ approximierbar. Für ein beliebiges Problem aus **PSPACE** gibt es dann eine Funktion in **FP**, die einer linear platzbeschränkten Berechnung vorangestellt werden muß, um dasselbe Resultat zu erhalten. (Dies besteht im wesentlichen aus einer Verlängerung der Eingabe.) Diese linear platzbeschränkte Berechnung ist für ein c durch 2^{cn} in der Laufzeit beschränkt. Im Falle des Verwerfens halte die Maschine nicht, sondern zähle (evtl. auf zusätzlichen Spuren) bis 2^{dn}, wobei $d > c + \delta$ gewählt sei.

Die Länge dieser Berechnung kann nach Widerspruchsannahme in polynomieller Zeit auf einen Faktor $2^{\delta n}$ approximiert werden. Das approximierte Ergebnis sei A. Es gilt also $m \leq A \leq 2^{\delta n} m$, wobei m die tatsächliche Länge der Berechnung ist.

Folgender deterministische Polynomialzeitalgorithmus arbeitet also korrekt: Zunächst wird die Umwandlung hin zu einer linear platzbeschränkten Berechnung vorgenommen. Auf diese wird der Approximationsalgorithmus angewendet, der nach Widerspruchsannahme existiert. Dessen Ergebnis A wird genutzt, indem genau dann akzeptiert wird, wenn $A < 2^{dn}$ gilt.

Dies liefert immer die korrekte Antwort. Für den Fall, daß die Originalberechnung akzeptiert, gilt $m \leq 2^{cn}$ und damit

$$A \leq 2^{\delta n} 2^{cn} = 2^{(c+\delta)n} < 2^{dn}.$$

Somit ist in diesem Fall die Antwort des **FP**-Algorithmus korrekt.

Akzeptiert die Originalberechnung nicht, so gilt $m \leq A$, also $A \geq 2^{dn}$, und somit ist auch in diesem Fall die Antwort des **FP**-Algorithmus korrekt.

Dies zeigt im Widerspruch zur Voraussetzung $\mathbf{PSPACE} \subseteq \mathbf{P}$. ∎

Diese Aussagen sollen in die Situation der Graphumfärbungen übertragen werden. Die folgenden Ergebnisse weisen den Weg.

3.4.3 Verbotene Substrings

Der nächste Schritt in Richtung Graphumfärbung findet beinahe noch in der Welt der formalen Sprachen statt. Um der Tatsache, daß ein Graph lediglich bestimmte Färbungen verbieten kann, Rechnung tragen zu können, wird hier eine neuartige Konstruktion zwischen Grammatik, Graph und Schaltkreis eingeführt. Es zeigt sich, daß die Funktionalität eines normalisierten Wortersetzungssystem in ein sogenanntes System verbotener Strings eingebettet werden kann.

Definition 23
*Ein **System verbotener Strings** besteht aus einem endlichen Alphabet Σ und einem Arbeitsband. Auf diesem Arbeitsband darf immer höchstens ein Buchstabe geändert werden. Bestimmte Arbeitsbandinhalte sind verboten.*

Es gibt endlich viele Verbote spezifiziert durch zwei Zahlen i und k sowie einen String w aus Σ^*. An allen Positionen beginnend, die modulo k gleich i sind, also $(i, i+k, i+2k, \ldots)$, wird der String mit dem Arbeitsband verglichen. Falls Gleichheit herrscht, ist dieser Arbeitsbandinhalt verboten.

Offenbar können in den Verbotsregeln auch Stellen mitten im String unberücksichtigt bleiben. In diesem Fall soll ein Arbeitsbandinhalt verboten sein, unabhängig davon mit welchen Symbolen diese unberücksichtigten Stellen belegt sind. Dementsprechend werden alle Möglichkeiten diese Stellen zu belegen zu neuen Verboten ohne unberücksichtigte Stellen. Da die Anzahl der Stellen und das Alphabet konstant ist, handelt es sich dabei nur um endlich viele neue Verbote. Wenn man an möglichst kurzen Verboten interessiert ist, kann es daher sinnvoll sein, unberücksichtigte Stellen zu erlauben und nicht zu zählen.

Eine alternative Betrachtungsweise eines Systems verbotener Strings ist die folgende: Es sei eine reguläre Sprache L fixiert. Es wird die Frage gestellt, ob zwei Wörter der Sprache, die die gleiche Länge haben, durch Veränderungen einzelner Buchstaben ineinander umgewandelt werden können, so daß alle Zwischenwörter auch in der Sprache L liegen.

Dies umfaßt offenbar auch ein System verbotener Strings, da die erlaubten Arbeitsbandinhalte eine reguläre Sprache darstellen. Da die regulären Sprachen unter Komplementbildung abgeschlossen sind, genügt es nachzuweisen, daß alle verbotenen Arbeitsbandinhalte durch eine reguläre Sprache beschrieben werden können. Für ein Verbot (i, k, w) leistet der folgende reguläre Ausdruck das Gewünschte:

$$\Sigma^i(\Sigma^k)^* w \Sigma^*$$

Die Vereinigung über alle diese Ausdrücke ergibt die gewünschte Sprache, die alle verbotenen Arbeitsbandinhalte beschreibt.

Dieser formale Mechanismus ist hier interessant, da sich die Arbeit eines Wortersetzungssystems durch verbotene Strings simulieren läßt.

Satz 24
Für jedes symmetrische, normalisierte Wortersetzungssystem gibt es ein System verbotener Strings (zu jeder Regelmenge existiert eine Menge von Verboten), für das gilt: Für jedes Paar von Start- und Zielworten für das Wortersetzungssystem gibt es ein Arbeitsband mit Start- und Zielinhalt, die äquivalent sind, also folgende Eigenschaften haben:

- *Es gibt eine Sequenz von gültigen Arbeitsbandinhalten, die sich jeweils nur an einer Position unterscheiden, genau dann, wenn das Zielwort aus dem Startwort im Wortersetzungssystem ableitbar ist.*
- *Die Länge dieser Sequenzen unterscheidet sich nur um einen konstanten Faktor, die Sequenzen kann man leicht auseinander gewinnen.*

Beweis: Das normalisierte Wortersetzungssystem bestehe aus den Regeln R über dem Alphabet Σ. Der Länge n des Startwortes entsprechend wird ein Arbeitsband der Länge $4n-2$ eingerichtet. Auf diesem sind n Paare von aufeinanderfolgenden Positionen verteilt. An diesen $2n$ sogenannten Buchstabenpositionen stehen Zeichen des Wortersetzungssystems. Auf jedem dieser Paare wird ein Buchstabe des Startwortes zum Startinhalt.

Zwischen je zwei solchen Paaren gibt es zwei weitere Positionen, die die (Original-) Buchstabengrenze markieren. Auf diesen ist das Zeichen $\circ$, sowie für jede Regel des Wortersetzungssystem ein anderes Zeichen, daß im folgenden durch x dargestellt wird, erlaubt.

Um diese Buchstabengrenzen herum werden alle Strings der Länge 6, außer den im folgenden beschriebenen, verboten. Dabei entstehen weniger als $(\Sigma + |R| + 1)^6$ Verbote, dies ist unabhängig von der Länge des Startwortes.

Für jede Regel der Form `AB`$\rightarrow$`CD` im Wortersetzungssystem werden folgende lokale Situationen erlaubt:

```
AAooBB
AAxoBB
AAxxBB
ACxxBB
ACxxBD
CCxxBD
CCxxDD
CCoxDD
CCooDD
```

Dies erzwingt, daß nur den Regeln entsprechende Umformungen stattfinden. In dem Fall, daß das Regelsystem reversibel ist, gibt es immer höchstens zwei Stellen, an denen eine Veränderung vorgenommen werden kann.

Solange ein `o` vorkommt, müssen die Buchstabenpaare gleich sein. Damit sind zwischen den ersten und letzten drei Zeilen nur vernünftige Umformungen erlaubt.

Dabei ist das `o` ein einziges Symbol, während `x` für ein Symbol steht, das für jede Regel verschieden ist.

Sobald eines der Paare von Buchstaben aufgelöst ist, darf sich das `xx` nicht ändern. Dadurch, daß in `x` die Regel gespeichert ist, kann nicht zwischen verschiedenen Regeln gesprungen werden. Insbesondere kann weder `AAxxDD` noch `CCxxBB` entstehen.

Es muß noch ausgeschlossen werden, daß statt eine Regelanwendung abzuschließen, eine ähnliche mit einem anderen Nachbarn scheinbar weitergeführt wird (die verbleibende Art zwischen Regeln zu springen). Dazu wird verboten, daß an zwei benachbarten Buchstabengrenzen gleichzeitig etwas von `oo` verschiedenes auftritt. ■

Definition 25
Ein k***-simultanes*** *System verbotener Strings ist ein System verbotener Strings bei dem die Buchstaben an bis zu* k *Positionen gleichzeitig verändert werden dürfen.*

Satz 26
Für alle $k \geq 1$ *kann ein System verbotener Strings durch ein* k*-simultanes System simuliert werden.*

Beweis: Für ein beliebiges $k \geq 1$ wird aus einem beliebigen System verbotener Strings und ein äquivalentes k-simultanes System konstruiert.

Dazu entsteht aus jeder Position des Original-Systems ein Block von k Positionen im k-simultanen System, das Alphabet wird übernommen. Innerhalb eines solchen Blockes wird verboten, daß Positionen verschieden belegt sind. Da die Anzahl der für einen Block benötigten Verbote nur von der Alphabetsgröße und k abhängt, ist sie unabhängig von der Länge des Systems.

Diese Verbote erzwingen, daß Veränderungen im k-simultanen System auf genau einem Block stattfinden.

Zusätzlich werden die Verbote des Original-Systems auf die jeweils ersten Positionen eines Blockes angewandt. Dadurch übernehmen die neuen Blöcke die Funktion der alten Positionen. Dies ist für den Fall, daß Verbote, die sich auf nicht aufeinanderfolgende Positionen beziehen, erlaubt sind, sofort realisierbar, dann bleibt die Länge der Verbote erhalten. Falls dies nicht der Fall ist, werden auch in den Verboten Buchstaben durch Blöcke von Buchstaben der Länge k ersetzt.

Diese Regeln erzwingen, daß ein Block im k-simultanen System genau dann verändert werden darf, wenn die entsprechende Änderung für den Buchstaben im Original-Systems erlaubt ist.

Daher entsprechen sich Belegungen des Arbeitsbandes und Sequenzen der Systeme direkt. ∎

Bei den bisherigen Resultaten ist die Anzahl der benötigten Farben zwar durch eine Konstante beschränkt, diese kann aber mitunter sehr groß sein. Das heißt aber nicht unbedingt, daß die vorgestellten Ergebnisse von dieser großen Zahl von Farben abhängen. Zum einen kann durch die Verwendung einer sogenannten universellen Turingmaschine die Anzahl der Zustände der Maschine und die Anzahl der Regeln begrenzt werden. So ergibt sich zwar durch die verschiedenen Schritte der Reduktion wieder eine bestimmte größere Anzahl von Farben, die aber im Vergleich zum direkten Vorgehen bereits deutlich verbessert ist. Eine andere Möglichkeit wäre es, bei der Umwandlung eines Wortersetzungssystems in ein System verbotener Strings durch andere, kompliziertere Konstruktionen eine Abhängigkeit von der Alphabetsgröße und der Anzahl der Regeln zu vermeiden. Da dies im folgenden nicht unbedingt benötigt wird, soll auf diese technischen Ergebnisse nicht weiter eingegangen werden.

3.4.4 Umfärben ohne zusätzliche Farben

Derartige Ausschlußregeln können in ein Graphumfärbungsproblem umgewandelt werden.

Satz 27
Die Frage, ob ein Graph ohne zusätzliche Farben umgefärbt werden kann, ist **PSPACE***-vollständig. Die Länge einer kürzesten Umfärbesequenz ist genauso schwer zu approximieren wie die Dauer einer linear platzbeschränkten Berechnung.*

Beweis: Zunächst stellt man fest, daß eine nichtdeterministische, linear platzbeschränkte Maschine das Problem durch Ausprobieren lösen kann. Mit Satz 4 (Savitch) folgt, daß das Problem in **PSPACE** liegt.

Dann reduziert man das in Satz 24 als **PSPACE**-vollständig gezeigte Problem der verbotenen Strings auf das Umfärbeproblem ohne zusätzliche Farben wie folgt:

Sei Σ das Alphabet des Systems verbotener Strings, $m = \max_{w \in R} |w|$ die maximale Länge eines verbotenen Strings. Dann kann eine Clique mit $|\Sigma| + m$ Elementen nicht ohne zusätzliche Farben umgefärbt werden. Diese Knoten erlauben es, Farben festzuhalten, und mit Alphabetszeichen aus Σ bzw. mit m verschiedenen Positionszeichen $\alpha, \beta, \ldots$ zu identifizieren.

Ein weiterer Knoten, der mit einem Knoten c der Clique verbunden ist, darf daher niemals die Farbe $f(c)$ bzw. das mit c assoziierte Alphabetszeichen annehmen. Mehrere Kanten dieser Form können also die an einem Knoten verwendbaren Farben bzw. Zeichen beliebig einschränken. Daher wird im folgenden nicht mehr explizit von Verbindungen zur Clique gesprochen, sondern es wird die Menge von erlaubten Symbolen angegeben.

So wird zunächst für jede Position des Systems verbotener Strings ein Knoten angelegt, der nur mit Farben aus Σ gefärbt werden darf. Dies Knoten ersetzen das Arbeitsband. Die verbotenen Strings werden durch die in Abbildung 3.1 dargestellte Konstruktion, sogenannten Gadgets, realisiert. Offenbar sind, genauso wie beim System der verbotenen Strings, ausgelassene Positionen unproblematisch. Diese Konstruktion leistet das Gewünschte: Falls der verbotene String nicht anliegt, gibt es eine Position, an der sich dies zeigt. An dieser kann in der zweiten Zeile die Alphabetsfarbe gewählt werden, für die dritte Zeile ist das Symbol der entsprechenden Position frei. Wenn dagegen der verbotene String anliegt, müssen in der zweiten Zeile jeweils die Positionsfarben benutzt werden. Dann kann allerdings der Knoten in der dritten Zeile nicht mehr gefärbt werden.

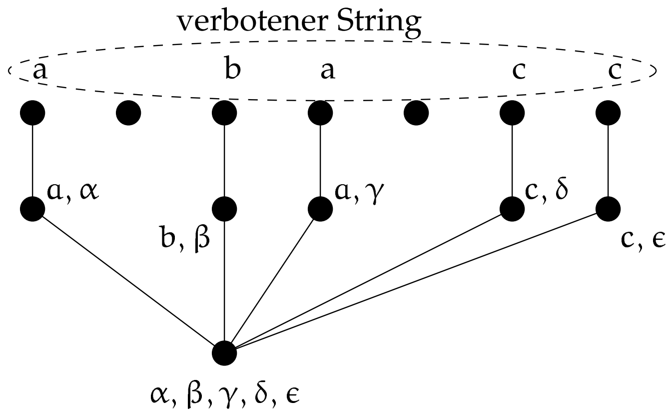

Abbildung 3.1: Graph-Gadget zur Realisierung von verbotenen Strings / Farbkombinationen; die angegebenen Zeichen stehen für die an den einzelnen Knoten erlaubten Symbole.

Außerdem können beliebige erlaubte Änderungen am Arbeitsband tatsächlich vorgenommen werden: Falls die Änderung dem System verbotener Strings nicht widerspricht, können in allen Gadgets, die an dieser Stelle einen der Buchstaben verbieten, die Positionssymbole verwendet werden, da es mindestens eine weitere Position gibt, an der Gadget und aktueller Bandinhalt nicht übereinstimmen. Wäre dies nicht der Fall, ist die Veränderung nicht erlaubt.

Es kann außerdem sowohl Anfangs- als auch Endstring ohne weiteres in eine Färbung umgewandelt werden. Diese lassen sich genau dann ohne zusätzliche Farbe ineinander umfärben, wenn man die Strings innerhalb des Systems verbotener Strings ineinander umwandeln kann.

Das Nicht-Approximierbarkeits-Resultat überträgt sich, da sich bei kürzesten Folgen die Anzahl der Umfärbungen linear zur Anzahl der Buchstabenänderung verhält. ∎

Diese Aussage über Graphumfärbungen muß noch auf die Situation des relaxierten ONLINE-NODE-COLORING übertragen werden.

Satz 28
Die Frage, ob bei relaxiertem Online-Coloring im aktuellen Schritt eine neue Farbe benötigt wird, ist **PSPACE***-hart.*

Beweis: Die Reduktion von einer symmetrischen Berechnung über die verschiedenen Zwischenschritte in ein Graphumfärbungsproblem (Lemma 20, Satz 24 und Satz 27) wird leicht modifiziert. In der Konstruktion aus Lemma 20, in der aus einer symmetrischen Maschine ein Wortersetzungssystem konstruiert wird, wird das Band links außerhalb der Bandanfangsmarkierung `%` um ein `a` ergänzt. Dies beeinflußt die Simulation überhaupt nicht. Dann werden für akzeptierende Zustände `X` die Regeln `%X`→`X%` und `aX`→`bX` hinzugenommen: Damit kann an dieser Stelle genau dann von `a` zu `b` gewechselt werden, wenn die Maschine akzeptiert.

Bei der Umwandlung in ein GRAPHRECOLORING-Problem nach Satz 24 und Satz 27 entsteht so die Eigenschaft, daß der neue Knoten, der diese neue Stelle des Arbeitsbandes repräsentiert, genau dann von der Farbe, die ein `a` repräsentiert, zu der Farbe, die ein `b` repräsentiert, umgefärbt werden kann, wenn die Maschine akzeptiert.

Mit den Methoden aus Satz 16 wird der oben beschriebene Graph zusammen mit der Anfangsfärbung erzeugt.

Jetzt wird ein neuer Knoten eingegeben, der einerseits so mit der Clique verbunden ist, daß er nur die Farbe `a` annehmen kann, und andererseits mit dem oben beschriebenen Knoten verbunden ist. Dieser kann genau dann von `a` zu `b` umgefärbt werden, wenn die Berechnung akzeptiert. Somit kann genau in diesem Fall der neu eingefügte Knoten ohne zusätzliche Farbe gefärbt werden.

Die Länge einer Umfärbesequenz ist genauso schwer zu approximieren wie im allgemeinen Umfärbeproblem.

Dieses Ergebnis kann auch auf den Fall, daß mehrere Knoten gleichzeitig umgefärbt werden dürfen, erweitert werden. Dies folgt analog zu dem Beweis von Satz 26. ■

Es zeigt sich, daß der konstruierte Graph in polynomieller Zeit optimal gefärbt werden kann. Dies liegt daran, daß seine Baumweite nur von dem der Reduktion zugrundeliegenden Systems verbotener Strings, und damit von der am Anfang der Reduktionskette stehenden Turingmaschine, abhängt. Da man von einer festen Maschine ausgehen kann, die ein beliebiges **PSPACE**-vollständiges Problem löst, kann man die Baumweite durch eine Konstante beschränken.

Die Tatsache, daß Umfärben schwierig ist, ergibt sich also unabhängig davon, wie schwer es ist, eine einzelne Färbung des Graphen zu berechnen.

Satz 29
Der Graph aus Satz 27 hat eine Pfadzerlegung (path-decomposition), deren Größe nur von dem in der Reduktion verwendeten System verbotener Strings abhängt.

Beweis: Sei k die maximale Länge eines Verbotes, gemessen an den am weitesten entfernten Positionen und nicht an der Anzahl der beteiligten Zellen. Dann hängt k nur von dem System verbotener Strings ab, von dem aus reduziert wird.

Die Mengen der Zerlegung bestehen jeweils aus der Clique und den Knoten, die zu einem Verbotsgadget gehören, sowie aus dem Intervall von k Bandknoten, die die beteiligten Bandknoten abdecken. Diese Zerlegung wird gemäß der Anfangspositionen der Intervalle als Pfad angeordnet.

Die beiden einfachen Bedingungen, daß dadurch alle Knoten und Kanten überdeckt sind, ist somit sofort klar. Die dritte Bedingung, daß wann immer zwei solche Mengen a und b einen nichtleeren Schnitt haben, dieser Schnitt in allen Mengen c, die auf dem eindeutigen Pfad zwischen a und b liegen, enthalten sein muß, ist auch erfüllt. Da die Clique in allen, und die echten Gadgetknoten immer nur in einer Menge enthalten sind, sind sie für diese Eigenschaft irrelevant. Bei den Bandknoten ergibt sich diese Eigenschaft direkt aus der Definition der Mengen. ∎

Aus diesen Betrachtungen lassen sich auch Ergebnisse über die Anzahl der Knoten, die in dem beschriebenen Online-Szenario korrekt gefärbt werden können, ableiten. Dies wird im folgenden direkt für das Frequenzzuteilungsproblem formuliert.

3.4.5 Anwendung zur kompetitiven Analyse

Die in den letzten Abschnitten entwickelten Resultate über die Beziehung zwischen dem Graphumfärbungsproblem und einer polynomiell platzbeschränkten Berechnung sollen hier wieder in den Zusammenhang zum ONLINE-FREQUENZZUWEISUNGSPROBLEM gebracht werden. So kann die Frage betrachtet werden, wieviele Gespräche ein effizienter Online-Algorithmus korrekt bearbeiten kann.

Dazu wird das Interferenzproblem bei der Kanalzuweisung an Mobilfunkgespräche als ein relaxiertes ONLINE-NODE-COLORING interpretiert, und man schließt aus den letzten Abschnitten, daß auch für dieses Problem kein effizienter Algorithmus existieren kann.

Satz 30
Es gelte $\mathbf{P} \neq \mathbf{PSPACE}$. *Dann kann es keinen deterministischen Polynomialzeitalgorithmus für das Frequenzzuweisungsproblem geben, der eine optimale deterministische Online-Strategie verwirklicht.*

Genauer kann kein solcher Algorithmus einen konstanten Anteil der annehmbaren Gespräche annehmen, ohne unnötige Frequenzen zu verwenden.

Beweis: Angenommen, es gäbe einen solchen Algorithmus $\mathcal{A}$, der einen Anteil ϵ der Anfragen ohne unnötige Frequenzen zu verwenden verwirklichen kann. Sei S eine Sequenz von Anfragen für das relaxierte ONLINE-NODE-COLORING Problem, t die letzte Anfrage dieser Sequenz, für die es nach Satz 28 **PSPACE**-hart ist, zu entscheiden, ob sie ohne Verwendung einer zusätzlichen Farbe verwirklicht werden kann. Sei $c \geq \frac{1}{\epsilon}$. Die Anfrage t wird $(c+1) \cdot |S|$ mal wiederholt, d.h. es wird jedesmal ein neuer Knoten eingegeben, der mit denselben Knoten verbunden ist, mit denen der Knoten von t verbunden ist, der in der nächsten Anfrage wieder gelöscht wird. Da nach dem Beweis von Satz 28 die Antwort, daß der Knoten mit einer bereits verwendeten Farbe gefärbt werden kann, durch eine Umfärbesequenz belegt werden muß und diese eindeutig bestimmt ist, kann in dieser Situation die Antwort des Algorithmus $\mathcal{A}$ nicht von der korrekten Antwort abweichen. Wenn also $\mathcal{A}$ auch nur eine einzige der wiederholten Anfragen ohne zusätzliche Farbe färbt, konnte t ohne zusätzliche Farbe gefärbt werden. Wenn andererseits t ohne zusätzliche Farbe gefärbt werden konnte, muß $\mathcal{A}$ auch mindestens eine der neuen Anfragen ohne zusätzliche Farbe färben. Wäre dies nicht der Fall, ergäbe sich das Verhältnis zwischen angenommenen und annehmbaren Anfragen zu $\frac{|S|}{(c+1)\cdot|S|} = \frac{1}{c+1} < \epsilon$, was im Widerspruch zur Voraussetzung steht. ■

Dieser Satz kann so verstanden werden, daß ein starker Algorithmus, also etwa eine Turingmaschine, die mit einem **PSPACE**-Orakel versehen ist, und dementsprechend in sehr kurzer Zeit die Antwort, ob der Knoten angenommen werden kann, gibt, verglichen wird mit einem Algorithmus, der genau dies nicht schnell entscheiden kann. Da beide Algorithmen in dem Fall, daß eine neue Anfrage bearbeitet werden kann, eine möglicherweise exponentiell lange Frequenzumverteilung durchführen müssen, scheint der starke Algorithmus auf den ersten Blick nicht allzusehr überlegen zu sein. Eine genau Formulierung dieser Idee besteht darin, ausgabeabhängige Komplexitäten zu betrachten, also die Laufzeit eines Algorithmus nicht nur an der Eingabelänge, sondern auch an der Länge der korrekten Ausgabe zu messen. In dem Fall, daß die Sequenz der Frequenzumverteilungen eindeutig ist, erscheinen in diesem Maß die beiden Algorithmen gleich schnell. Falls die neue Anfrage aber nicht ohne zusätzliche Frequenz angenommen werden kann, kann der starke Algorithmus die Antwort sofort geben, während unter den angegebenen Voraussetzungen kein Algorithmus eine schnelle Antwort geben kann. Da in diesem Fall die Ausgabelänge konstant ist, zeigt sich der Unterschied zwischen den Algorithmen auch in einem ausgabeabhängigen Komplexitätsmaß.

Dies zeigt, daß die Anzahl der angenommenen Gespräche bei fest gegebenem Frequenzspektrum nicht gut zu lösen ist.

3.5 Online-Strategien

Die Ergebnisse der letzten Kapitel zeigen, daß effiziente Online-Strategien auf uneingeschränkten Graphen nicht zu erwarten sind.

Dementsprechend ergibt sich die Frage, ob es Einschränkungen der Fragestellung gibt, die effiziente Lösungen erlauben. Da mit einer Feststation mehr als eine Mobilstation versorgt werden soll, wird die Einschränkung nicht direkt für die entstehenden Graphen angegeben, sondern ein Graph betrachtet, in den die Anfragen eingebettet werden. Falls dieser Graph in der Klasse der bipartiten Graphen liegt, zu denen Gitter und Bäume gehören, gibt es eine optimale Online-Strategie. Für den Fall, daß dieser Graph eine Gradschranke erfüllt, wird eine effiziente Online-Strategie angegeben.

Definition 31
Seien G und H Graphen. H heißt in G eingebettet, wenn G ein homomorphes Bild von H ist, und die Urbilder von adjazenten Knoten Cliquen bilden.

Ein Knoten ist also mit allen Knoten, die an derselben oder einer benachbarten Stelle eingebettet sind, verbunden.

Anstelle einer Färbung eines eingebetteten Graphen kann auch ein sogenanntes **Multi-Coloring** des Graphen, in den eingebettet wird, betrachtet werden. Dazu wird jeder Knoten um eine Zahl erweitert, die angibt, wieviele verschiedene Farben ihm zugewiesen werden müssen. Für einen solchen Graphen besteht eine gültige Färbung darin, daß allen Knoten genügend Farben zugewiesen werden und benachbarte Knoten keine gemeinsame Farbe haben. Dies ist offenbar äquivalent zum Färbungsproblem auf eingebetteten Graphen. Dementsprechend werden die Begriffe austauschbar verwendet.

Definition 32
Sei G ein Graph. Das ***Online-Coloring Problem auf in** G **einbettbare Graphen*** *ist ein Online-Coloring Problem mit der zusätzlichen Eigenschaft, daß alle entstehenden Graphen in G einbettbar sind und jeder Knoten zusammen mit seiner Einbettung eingegeben wird.*

Satz 33
Sei G ein bipartiter Graph. Für das Online-Coloring auf G gibt es eine bezüglich der Anzahl der verwendeten Farben 1-kompetitive Online-Strategie. Die benötigten Berechnungen sind sehr einfach.

Beweis: Sei k die Anzahl der zur Verfügung stehenden Farben $\{1, \ldots, k\}$. Den Knoten von G seien im Sinne einer Zweifärbung die Farben rot bzw. blau zugewiesen. Falls ein neuer Knoten in einen rot gefärbten Knoten von G eingebettet wird, so wird ihm die kleinste Farbe aus $\{1, \ldots, k\}$ zugewiesen, die im Moment möglich ist. Andernfalls ist der neue Knoten in einen blau gefärbten Knoten von G eingebettet, und es wird ihm die größtmögliche Farbe aus $\{1, \ldots, k\}$ zugewiesen. So bilden die Farben der Knoten, die auf einen roten bzw. blauen Knoten G eingebettet sind, eine Menge der Form $\{1, \ldots, i\}$ bzw. $\{j, \ldots, k\}$. Diese Eigenschaft der Färbung kann auch in dem Fall, daß ein Knoten gelöscht wird, durch entsprechendes Umfärben aufrechterhalten werden.

Falls mit dieser Strategie ein Knoten nicht mehr eingefärbt werden kann, so kann dies keine Strategie leisten, weil an zwei benachbarten Positionen im ganzen mehr als k Knoten vorhanden sind. ■

Falls der Graph, in den das Problem eingebettet ist, nicht bipartit ist, aber eine kleine Gradschranke erfüllt, dann gibt es eine im folgenden beschriebene Strategie von garantierter Qualität.

Satz 34
Sei G ein Graph mit maximalem Grad d, und eine Anfrage für MULTI-COLORING *mit Maximalgröße der Liste k. Dann gibt es eine Färbung, die höchstens $k \cdot (d + 1)$ Farben benutzt.*

Beweis: Die Knoten werden in einer beliebigen Reihenfolge betrachtet. Den einzelnen Positionen der Liste wird jeweils die kleinstmögliche freie Farbe zugewiesen. Da an einem Knoten höchstens k und an seinen Nachbarn höchstens $k \cdot d$ Farben verwendet werden, wird niemals eine Farbe größer als $k \cdot (d + 1)$ benutzt. ■

Dieses Ergebnis überträgt sich auf die Situation der Frequenzzuweisung.

Definition 35
Sei G ein Graph. Bei dem **ONLINE-FREQUENZZUTEILUNGS** *Problem auf G werden auf den Knoten Bedarfsanforderungen gestellt und gelöscht. Ein korrekte Lösung weist jeder dieser Anfragen eine Frequenz zu, die verschieden ist zu allen Frequenzen, die bereits an diesem Knoten oder an einem Nachbarn verwendet wird. Dabei dürfen den Anforderungen nacheinander neue Frequenzen zugewiesen werden.*

Damit läßt sich formulieren, inwiefern Graphen von kleinem Grad für dieses Problem vorteilhaft sind.

Satz 36
Sei G ein Graph mit maximalem Grad d. Dann gibt es einen effizienten Algorithmus für das ONLINE-FREQUENZZUTEILUNGS *Problem auf G, der bezüglich der Anzahl der verwendeten Farben $(d+1)$-kompetitiv ist. Die Laufzeit des Algorithmus nach jeder Eingabe ist polynomiell in der Anzahl der aktuell vorhandenen Anforderungen beschränkt.*

Beweis: Die Frequenzen und die Knoten von G werden beliebig linear angeordnet. Der Algorithmus sorgt dafür, daß immer eine Färbung vorhanden ist, die der im Beweis von Satz 34 angegebenen gleicht.

Falls eine neue Anforderung eingeht, wird ihr die kleinstmögliche, noch nicht an diesem Knoten oder einem Nachbarn verwendete Frequenz zugewiesen.

Falls eine Anforderung gelöscht werden soll, wird zunächst die zugehörige Frequenz an dem Knoten freigegeben. Dann untersucht der Algorithmus alle Knoten des Netzes der Reihe nach daraufhin, ob nach dem Löschen der Anforderung Frequenzen mit hoher Nummer durch solche mit kleiner Nummer ersetzt werden können. Sei der aktuell betrachtete Knoten v. Sei A die Anforderung am Knoten v, die die größte Frequenz m benutzt. Falls es eine freie Frequenz gibt, die kleiner als m ist, wird A auf diese Frequenz umgestellt. In diesem Fall wird v erneut betrachtet.

Sei k die maximale Anzahl der in einer Situation an einem Knoten vorhandenen Anforderungen. Es werden höchstens $(d+1) \cdot k$ Frequenzen verwendet. Angenommen, dies wäre nicht der Fall. Dann gibt es ein erstes Mal einen Knoten v, an dem eine Frequenz verwendet wird, deren Index m größer als $(d+1) \cdot k$ ist. An v und allen seinen Nachbarn können zusammen höchstens $(d+1) \cdot k - 1$ Anforderungen vorhanden sein. Also gibt es eine freie Frequenz, deren Index $f \leq (d+1) \cdot k$ ist. Falls die letzte Anfrage ein Löschen war, hat der Algorithmus dafür gesorgt, daß dies nicht der Fall ist. Also war die letzte Anfrage eine neue Anforderung. Dann kann durch diese Anfrage die Frequenz f nicht frei geworden sein, war also schon im vorigen Schritt frei. Da in dieser Situation der Algorithmus die Frequenz m am Knoten v nicht neu verwendet, war im Widerspruch zur Voraussetzung schon vor dem letzten Schritt Frequenz m verwendet und Frequenz f nicht benutzt.

Da jeder Algorithmus mindestens k Frequenzen benutzt, ergibt sich, daß der vorgestellte Algorithmus $(d + 1)$ kompetitiv ist. ∎

Daher stellt sich die Frage, ob man die Sender so wählen kann, daß der Verbrauch an Funkfrequenzen in Grenzen gehalten werden kann. Dies wäre der Fall, wenn ein kleiner Grad des Interferenzgraphen erreicht werden könnte. Diese Fragestellung wird im nächsten Kapitel behandelt.

4

Standortplanung

Die für das Online-Problem gefundenen Ergebnisse führen zu bestimmten Anforderungen an das Sendernetz. Im folgenden wird untersucht, wie schwierig es ist, in diesem Sinn gute Standorte auszuwählen.

Die Fragestellungen werden als Optimierungsprobleme formuliert. Die in diesem Kapitel vorgestellten neuen Ergebnisse umfassen **NP**-Hardness und Nicht-Approximierbarkeits-Resultate. Das einzige wesentliche Resultat auf das aufgebaut wird, sind die entsprechenden Aussagen über das SETCOVER-Problem.

4.1 Modellierung

Zunächst müssen die physikalischen und geographischen Gegebenheiten modelliert werden. Dabei wird davon ausgegangen, daß schon eine endliche Menge von denkbaren Senderstandorten identifiziert ist. Daraus ergeben sich dann die Versorgungs- oder Bedarfsregionen als Gebiete, die von den gleichen Sendern versorgt werden können. Es wird vereinfachend davon ausgegangen, daß ein Sender, sobald er stört, auch versorgen kann. Versorgen bedeutet hier, daß mit der geplanten Sendeleistung eine Verbindung zwischen einer Mobilstation in der Bedarfsregion und diesem Sender möglich ist. Stören bedeutet, daß die Benutzung einer Frequenz an dem Sender die gleichzeitige Benutzung dieser Frequenz für eine andere Verbindung in der Bedarfsregion unmöglich macht. Dies ist nicht in der physikalischen Realität begründet, sondern in der Tatsache, daß ein ausreichend allgemeines Modell auch dies erlauben muß.

Dementsprechend wird als Eingabe der bipartite **Standortgraph** $G = (V, E)$ verwendet. Die Eckenmenge teilt sich in Sender und Regionen $V = S \dot{\cup} R$. Eine Kante besteht genau dann, wenn der Sender die Region versorgen und damit auch stören kann.

Eine **Senderauswahl** $C \subseteq S$ ist gültig, wenn alle Regionen versorgt werden, also für alle $x \in R$ ein $s \in S$ existiert, so daß $(x, s) \in E$ eine Kante des Graphen ist. Dabei entsteht ein **Interferenzgraph**. Seine Eckenmenge C besteht aus den ausgewählten Sendern. Die Kantenmenge F ist so definiert, daß zwei ausgewählte Sender genau dann verbunden sind, wenn es eine Bedarfsregion gibt, die von beiden Sendern versorgt wird, und damit die gleichzeitige Verwendung derselben Frequenz an diesen Sendern verboten ist, $(p, q) \in F \iff \exists x \in R\colon (p, x) \in E \land (x, q) \in E$. Graphentheoretisch ist dies genau die Einschränkung des Quadratgraphens auf die Senderauswahl. Dabei besteht der **Quadratgraph** eines beliebigen Graphen aus denselben Ecken wie der Originalgraph. Kanten gibt es zwischen Ecken, die im Originalgraphen durch Wege der Länge höchstens zwei verbunden waren. Kanten, die einen Knoten mit sich selbst verbinden, werden dabei ausgeschlossen.

4.2 Optimierungsprobleme

Mit diesen Definitionen ergeben sich verschiedene Möglichkeiten, eine gute Standortauswahl zu definieren. Zunächst soll für einen Standortgraphen eine Senderauswahl getroffen werden, so daß der entstehende Interferenzgraph einen möglichst kleinen Grad hat. Dieses Problem wird MINGRADSENDERAUSWAHL genannt. Diese Frage ist interessant, da Graphen mit kleinem Grad nach Satz 36 zu einer vergleichsweise guten Ausnutzung des Frequenzspektrums führen.

Satz 37
Es gelte $\mathbf{P} \neq \mathbf{NP}$*. Dann ist* MINGRADSENDERAUSWAHL *nicht auf einen Faktor* $(1 - \epsilon)\ln(|R|)$ *approximierbar.*

Beweis: Das Resultat ergibt sich aus einer appproximations-erhaltenden Reduktion von SETCOVER. Sei (U, F) eine SETCOVER-Instanz, bestehend aus einer endlichen Menge $U = \{e_1, \ldots, e_n\}$ und einem Teilmengensystem $F = \{f_1, \ldots, f_m\} \subseteq 2^U$. Daraus wird eine Instanz von MINGRADSENDERAUSWAHL konstruiert. Wie in Abbildung 4.1 dargestellt besteht der bipartite Graph aus den Sendern $f_1, \ldots, f_m$ und dem zusätzlichen Sender p. Als Bedarfsregionen gibt es die $e_1, \ldots, e_n$ und zusätzlich die Ecken u und v. Diese sind mit p verbunden. Die f_i sind mit u und den Knoten e_j, die für Elemente der Teilmenge f_i stehen, verbunden.

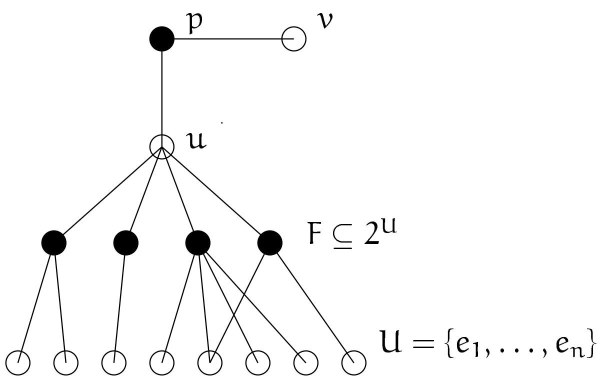

Abbildung 4.1: Bei der Reduktion konstruierter Graph

Für eine korrekte Lösung muß der Sender p ausgewählt werden, da sonst ν nicht versorgt werden würde. Die übrigen ausgewählten Sender versorgen alle Bedarfsregionen, die zugehörigen Mengen bilden also eine Überdeckung von U. Der entstehende Interferenzgraph hat als Maximalgrad den Grad in p, der genauso groß ist wie die Anzahl der ausgewählten f_i, da die f_i höchstens ebensoviele Nachbarn haben können. Die Zielfunktion überträgt sich also mit Faktor 1. Da die hierfür benötigten Berechnungen, die Konstruktion des Graphen und die Interpretation der Senderauswahl als Mengenauswahl sicher in polynomieller Zeit, tatsächlich sogar in logarithmischem Platz, ausgeführt werden können, führt die Existenz eines Approximationsalgorithmus für MINGRADSENDERAUSWAHL zu einem für SETCOVER. Unter der Voraussetzung $\mathbf{P} \neq \mathbf{NP}$ kann es einen solchen mit Qualität $(1-\epsilon)\ln(|U|)$ nach [RS97, Fei96] nicht geben. Da die Bedarfsregionen den Elementen im Universum entsprechen, gilt $|U| + 2 = |R|$. Somit gilt die Behauptung. ■

Korollar 38
Das MINGRADSENDERAUSWAHL *entsprechende Entscheidungsproblem ist* **NP***-vollständig.*

Beweis: Eine nichtdeterministische Maschine rät die Senderauswahl und überprüft dann, ob Gradschranke und Versorgungsbedingung erfüllt sind. Zusammen mit Satz 37 ergibt sich die Behauptung. ■

Ob umgekehrt SETCOVER zur Lösung der Gradbeschränkung verwendet werden kann, ist unklar. Auch ob sich die Qualität eines Greedy-Algorithmus abschätzen läßt, ist unabhängig von diesen Ergebnissen.

Eine Interpretation dieses Resultats wäre, den Maximalgrad als adäquates Qualitätsmaß in Frage zu stellen. Dies führt zu einer genaueren Analyse von anderen naheliegenden Maßen für die Qualität des entstehenden Interferenzgraphen. Es zeigt sich, daß die hier betrachteten Maße auch sehr schwer zu optimieren sind. Auch wenn die folgenden Ergebnisse nicht erschöpfend sind, stärken sie die Vermutung, daß es im allgemeinen schwierig ist, ein „vernünftiges“ Netzwerk durch Senderauswahl zu gewinnen.

Satz 39
Das Problem, in einem gegebenen Standortgraphen eine gültige Senderauswahl zu treffen, die mit möglichst wenigen Sendern auskommt, ist genauso schwer zu approximieren wie SETCOVER. *Dies gilt auch, wenn die Bedarfsregionen bzw. Teilmengen mit Gewichten versehen werden dürfen.*

Beweis: Man kann die Instanzen der Probleme durch Uminterpretation auseinander gewinnen.

Sei $G = (S \cup R, E)$ ein Standortgraph. Dann wähle man als Universum die Bedarfsregionen R und als Teilmengen für jeden Sender die Menge, die alle adjazenten Bedarfsregionen enthält. Eine gültige Senderauswahl stellt dann sicher ein Set-Cover dar und umgekehrt. Die Qualitätsmaße, d.h. die Anzahl der ausgewählten Sender bzw. Teilmengen, übertragen sich direkt. Daher ist dieses Problem der Senderauswahl mindestens so schwierig zu approximieren wie SETCOVER.

Falls die Bedarfsregionen gewichtet sind, erhalten die Teilmengen die Summe der Gewichte der ihren Elementen entsprechenden Regionen.

Sei umgekehrt $I = (U, F)$ eine SETCOVER-Instanz. Dann wähle man als Bedarfsregionen die Elemente des Universums U. Für jede Teilmenge stehe ein Sender, der mit allen Elementen der Teilmenge verbunden ist. Die Qualitätsmaße übertragen sich genauso wie oben. Dies zeigt, daß SETCOVER nicht schwieriger ist, als diese Senderauswahl.

Seien die Teilmengen der SETCOVER-Instanz mit Gewichten $w_1, \dots, w_m$ versehen. Es wird der in Abbildung 4.2 dargestellte Graph konstruiert. Die Gewichte der e_i werden auf 1 gesetzt, das Gewicht der u_i ist jeweils durch $n \cdot m \cdot w_i$ gegeben.

Eine beliebige Senderauswahl mit Gewicht K korrespondiert zu einem Set-Cover mit Gewicht $K/(n \cdot m)$. Dazu stellt man fest, daß der Beitrag der

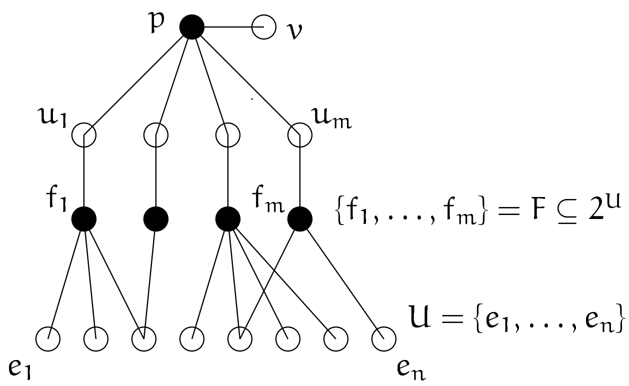

Abbildung 4.2: Graph der die Verbindung zu gewichtetem SETCOVER zeigt

Universumsknoten zum Gesamtgewicht den Wert $n \cdot m$ nicht überschreiten kann, und die Topologie einen Set-Cover erzwingt. Daher übertragen sich die Nicht-Approximierbarkeits-Resultate. ∎

Satz 40
Die Zahl der Sender, die gleichzeitig eine Region erreichen, durch eine möglichst kleine Zahl zu beschränken, ist genauso schwer zu approximieren wie SETCOVER.

Beweis: Dies zeigt der bei der Reduktion in Satz 37 verwendete Graph, der in Abbildung 4.1 dargestellt ist.

Genau die ausgewählten Sender (Teilmengen) sind zu u benachbart, damit übertragen sich die Anzahlen direkt. ∎

Das folgende Ergebnis zeigt, daß es noch schwieriger zu optimierende Maße gibt.

Satz 41
Es sei $\mathbf{P} \neq \mathbf{NP}$. *Dann kann das Problem, die Anzahl der im Interferenzgraphen vorhandenen Kanten zu minimieren, genannt* MINNREDGES, *durch keinen Polynomialzeitalgorithmus auf einen Faktor* $(1-\epsilon)(\log|R|)^2$ *approximiert werden.*

Beweis: Es wird das Nicht-Approximierbarkeits-Resultat für SETCOVER und die Tatsache verwendet, daß bei dieser Konstruktion $\Theta(k^2)$ Kanten für k Mengen stehen. Der konstruierte Graph ist der aus Abbildung 4.1 lediglich ohne die Knoten p und v.

Sei $\mathcal{A}$ ein Approximationsalgorithmus für MINNREDGES mit Approximationsgarantie $\alpha(|R|)$, wobei R für die Anzahl der Bedarfsregionen im Eingabegraphen steht.

Es wird daraus ein Approximationsalgorithmus $\mathcal{B}$ für SETCOVER mit Qualitätsgarantie $\beta(|U|)$ konstruiert, wobei U für das Universum in der SETCOVER-Instanz steht.

Aus der Konstruktion des Graphen ergibt sich, daß eine Auswahl von k Sendern bzw. Teilmengen zu genau $\binom{n}{2} = \frac{n(n-1)}{2}$ Kanten im Interferenzgraphen führt, der hier auch immer eine Clique ist. Sei x die Größe der optimalen Auswahl bei fester Eingabeinstanz und $y = \binom{x}{2}$ die optimale Anzahl von Kanten. Dann gilt $\beta = \frac{\mathcal{B}}{x}$ und $\alpha = \frac{\mathcal{A}}{y}$. Somit gilt

$$\alpha = \frac{\mathcal{A}}{y} = \binom{\mathcal{B}}{2} \cdot \binom{x}{2}^{-1} = \frac{\mathcal{B}(\mathcal{B}-1)}{x(x-1)} \geq \left(\frac{\mathcal{B}-1}{x}\right)^2 \geq_{\text{ae}} (1-\epsilon)\beta^2 .$$

Hierbei steht $\geq_{\text{ae}}$ für die Tatsache, daß für ausreichend großes $\mathcal{B}$ und ein entsprechendes $\epsilon > 0$ an dieser Stelle $\geq$ gilt.

Da für $\beta = (1-\epsilon)\ln(|U|)$ die Approximation **NP**-hart ist, folgt die Behauptung. ■

Auch die Frage, ob man einen bipartiten Sendergraphen auswählen kann, erweist sich als schwierig.

Satz 42
Das Problem, in einem Standortgraphen eine Senderauswahl zu treffen, so daß der entstehende Interferenzgraph bipartit ist, genannt BIPARTITE-AUSWAHL, *ist* **NP**-*vollständig.*

Beweis: Eine nichtdeterministische Maschine rät eine entsprechende Senderauswahl, und überprüft in polynomieller Zeit, ob der entstehende Interferenzgraph bipartit ist. Dies zeigt, daß das Problem in **NP** liegt.

Daß das Problem **NP**-hart ist, zeigt eine Reduktion von EXACT COVER BY 3-SETS ([GJ79], SP2). Die Eingabe ist eine SETCOVER-Instanz, in der alle Teilmengen Mächtigkeit 3 haben. Die Frage ist, ob es eine Auswahl von Teilmengen gibt, so daß jedes Element in genau einer dieser ausgewählten Mengen liegt. Dieses Problem ist **NP**-vollständig.

Die konstruierte Instanz ergibt sich aus Abbildung 4.2. Der entstehende Graph ist genau dann bipartit, wenn keine von den Universumsregionen mehrfach versorgt wird. Genau dann ergeben die entsprechenden Teilmengen auch ein Exaktes 3-Cover. ■

5 Ausblick

Die in dieser Arbeit untersuchten Fragestellungen ergeben sich aus der allgemeinen Problematik, wie ein Mobilfunknetz effizient betrieben werden kann, und wie Feststationen positioniert werden sollen, um dafür möglichst gute Voraussetzungen zu schaffen.

Die Komplexitätsaussagen über die untersuchten Umfärbeprobleme haben direkte Konsequenzen auf die durch effiziente Online-Algorithmen erreichbare Qualität. Es zeigte sich, daß eine einfache Struktur des Netzes nötig ist, um effiziente Online-Algorithmen guter Kompetitivität zu ermöglichen. Die dabei betrachteten theoretischen Probleme scheinen auch für sich interessant, und können möglicherweise auch zur Lösung von anderen Fragestellungen beitragen. Insbesondere das hier als Umfärbeproblem betrachtete Phänomen, durch schrittweise Veränderungen Sequenzen im Lösungsraum eines Problems zu betrachten, erscheint vielversprechend. Dieses Vorgehen kann auch als Konnektivität eines impliziten Graphen verstanden werden. Derartige Fragen treten z.B. im Zusammenhang mit Faltungen von RNA-Sequenzen oder auch von lokalen Suchverfahren auf.

Eine offene, interessante Frage ist, wie schwer es ist, die Anzahl der beim Umfärben zusätzlich benötigten Farben zu bestimmen. Eine ähnlich Frage ist, wieviele Knoten mindestens gleichzeitig umgefärbt werden müssen, um mit einer bestimmten Anzahl von zusätzlichen Farben eine Umfärbung zu erreichen.

Eine andere Richtung, das Umfärbeproblem zu erweitern, ist es sogenannte swaps zu betrachten. In diesem Fall ist es (zusätzlich) erlaubt, in einem Schritt die Farben zweier Knoten zu vertauschen. Für einige der vorgestellten Ergebnisse zeigt sich, daß die verwendete Konstruktion so modifiziert werden kann, daß diese zusätzliche Möglichkeit keinen Einfluß hat, oder nur die angegebenen Schranken leicht verschlechtert. Die Details dieser Modifikationen und die Frage, ob ein allgemeiner Zusammenhang zwischen dem Modell mit und ohne swaps aufgezeigt werden kann, ist eine eigene Richtung weiterer Betrachtungen.

Literaturverzeichnis

[AA95] B. Awerbuch and Y. Azar. Competitive multicast routing. *Wireless networks*, 1:107–114, 1995.

[AS97] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. In *STOC*, pages 485–495. ACM, 1997.

[Ben89] Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM J. Comput.*, 18(4):766–776, August 1989.

[Bod87] Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. Technical Report RUU-CS-87-22, Departement of Computer Science, University of Utrecht, P.O. Box 80.012, 3508 TA Utrecht the Netherlands, November 1987.

[BOL94] Bayern Online. Dokumentation zur Ausstellung der Bayerischen Staatskanzlei, 1994.

[DJLS95] G. Dahl, K. Jörnsten, G. Løvnes, and S. Svaet. Graph optimization problems in connection with the management of mobile communications systems. *Telecommunication Systems*, 3:319–339, 1995.

[DV93] Dragomir D. Dimitijević and Jelena Vučetić. Design and performance analysis of the algorithms for channel allocation in cellular networks. *IEEE Transactions on Vehicular Technology*, 42(4):526–534, November 1993.

[Fei96] U. Feige. A threshold of $\ln n$ for approximating set cover. In *28th Ann. ACM Symp. on Theory of Comp.*, pages 314–318. ACM, 1996.

[FMS+95] A. Feldmann, B. Maggs, J. Sgall, D. D. Sleator, and A. Tomkins. Competitive analysis of call admission algorithms that allow delay. Technical Report CMU-CS-95-102, School of Computer Science, Carnegie Mellon University, January 1995.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 79.

[GSW94] A. Gräf, M. Stumpf, and G. Weißenfels. On coloring unit disk graphs. Technical Report 17, Musikwissenschaftliches Institut der Universität Mainz, Juli 1994.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.

[Ira94] S. Irani. Coloring inductive graphs on-line. *Algorithmica*, 11, 1994.

[KVV90] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 352–358, 1990.

[Mal95] Ewa Malesińska. List coloring and optimization criteria for a channel assignment proplem. Technical Report 458/1995, TU Berlin, Fachbereich 3, Mathematik, 1995.

[MP96] Ewa Malesińska and Alessandro Panconesi. On the hardness of allocating frequencies for hybrid networks. Technical Report 498/1996, TU Berlin, Fachbereich 3, Mathematik, 1996.

[OSH94a] T. Ottmann, S. Schuierer, and C. Hipke. Kompetetive Analyse für Online–Algorithmen — Eine kommentierte Bibliographie (Teil I). Technical report, Institut für Informatik, Albert–Ludwigs–Universität Freiburg, August 1994.

[OSH94b] T. Ottmann, S. Schuierer, and C. Hipke. Kompetetive Analyse für Online–Algorithmen — Eine kommentierte Bibliographie (Teil II). Technical report, Institut für Informatik, Albert–Ludwigs–Universität Freiburg, August 1994.

[PRR95] C. Papadimitriou, S. Ramanathan, and P. Rangan. Optimal information delivery. In John Staples, Peter Eades, Naoki Katoh, and Alistair Moffat, editors, *Algorithms and Computation*, volume 1004 of *Lecture Notes in Computer Science*, pages 181–187, 1995.

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC*, pages 475–484. ACM, 1997.

[Sim88] Hans Ullrich Simon. The analysis of dynamic and hybrid channel assignment. Technical Report 10/1988 SFB 124–B1, Fachbereich Informatik, Universität des Saarlandes, D-6600 Saarbrücken, 1988.

[Sim89] Hans Ullrich Simon. Approximation algorithms for channel assignment in cellular radio networks. In J. Csirik, J. Demetrovics, and F. Gécse, editors, *Fundamentals of Computation Theory*, volume 380 of *LNCS*, pages 405–416, August 1989.

[Vis92] Sundar Vishwanathan. Randomized online graph coloring. *Journal of Algorithms*, 13:657–669, 1992.

[ZY89] Ming Zhang and Tak-Shing P. Yum. Comparisons of channel-assignment strategies in cellular mobile telephone systems. *IEEE Transactions on Vehicular Technology*, 38(4):211–215, November 1989.

Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Würzburg, September 1997

Riko Jacob