Albert-Ludwigs-Universität Freiburg i. Brsg.
Fakultät für Angewandte Wissenschaften
Institut für Informatik
Lehrstuhl für Grundlagen der Künstlichen Intelligenz
Prof. Dr. Bernhard Nebel

Approximationseigenschaften von Transportproblemen in der Handlungsplanung

Diplomarbeit
im Fach Informatik
von

Michael Drescher

Januar 2006

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem habe ich alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, als solche kenntlichgemacht. Ich erkläre hiermit auch, dass diese Diplomarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Michael Drescher, im Januar 2006.

Danksagung

Ich möchte mich hiermit bei Malte Helmert für die Betreuung dieser Arbeit und das Korrekturlesen einer früheren Version dieser Arbeit bedanken.
Auch meinen Eltern ein herzliches Danke!

Inhaltsverzeichnis

1 Einleitung

1.1 Einführung

In der International Planning Competition (IPC), einem seit 1998 alle zwei Jahre stattfindenden Planungswettbewerb, werden Planungsprogramme anhand von wechselnden Testaufgaben (Benchmark-Aufgaben) verglichen. Diese Testaufgaben stammen dabei aus verschiedenen Anwendungsbereichen für Planer, die jeweils durch eine sogenannte Planungsdomäne formalisiert sind, auf deren Grundlage dann verschiedene Aufgaben gestellt werden.

Um nun Erfolge und Misserfolge von Planungsprogrammen bei bestimmten Planungsdomänen dahingehend einschätzen zu können, ob z. B. Erfolge eher auf originelle theoretische Konzepte oder auf eine effiziente Implementierung zurückzuführen sind, ist es hilfreich, ein Maß für die Schwierigkeit der Aufgaben einer Domäne zur Verfügung zu haben. Dafür ist es sinnvoll, diese Planungsdomänen als algorithmische Probleme und die Testaufgaben als Instanzen dieser Probleme aufzufassen: Dann kann man die Schwierigkeit der Domänen mit in der Informatik etablierten Konzepten wie NP-Vollständigkeit oder (in dieser Arbeit) Approximierbarkeit untersuchen.

Malte Helmert hat in seiner Diplomarbeit [Hel01] derartige komplexitätstheoretische Untersuchungen für die in den Wettbewerben von 1998 und 2000 verwendeten Planungsdomänen durchgeführt und abstrakte Domänen eingeführt, die jeweils mehrere der in den Wettbewerben verwendeten Domänen verallgemeinern. Eine der von ihm untersuchten abstrakten Domänen ist die sogenannte *Transportdomäne*.

Die in ihr vertretenen Instanzen sind gekennzeichnet durch

- durch Wege verbundene Orte (in Form eines ungerichteten Graphen),
- Fahrzeuge, die sich entlang dieser Wege fortbewegen und dabei je nach Fahrzeug eine bestimmte Anzahl von Gegenständen gleichzeitig transportieren können,
- zuzustellende Gegenstände mit jeweils eigenem Start- und Zielort,
- Fahrzeugarten (wie etwa: Schiff, Flugzeug, Zug, Lastwagen, PKW), die bestimmen, welchen Teil des Wegenetzes ein Fahrzeug benutzen darf, und
- Treibstoff, der an den Knoten des Wegenetzes gelagert wird.

Je nach erlaubter Anzahl von Fahrzeugarten, nach erlaubter Anzahl und Kapazität der Fahrzeuge sowie verfügbarem Treibstoffvorrat unterscheidet M. Helmert verschiedene Subklassen.

Für die folgenden Entscheidungsprobleme gibt er in [Hel01] eine vollständige komplexitätstheoretische Klassifikation an, in der diese für jede Subklasse entweder als in polynomieller Zeit lösbar oder als NP-vollständig eingeordnet werden:

- Existiert für eine gegebene Instanz (Testaufgabe) ein Plan, der alle Gegenstände zu ihren Zielen bringt?
- Existiert für eine gegebene Instanz und eine gegebene natürliche Zahl k ein solcher Plan, der aus höchstens k Aktionen besteht?

Lässt sich ersteres in polynomieller Zeit lösen, während das zweite NP-hart ist, so kann man sich fragen, Lösungen welcher Qualität man in polynomieller Zeit erreichen kann. Das ist der Fall für alle diejenigen Probleme, in denen der Treibstoffvorrat unbeschränkt ist, und führt auf die Frage der *Approximierbarkeit* dieser Probleme. Die Güte eines Plans wird dabei durch die Anzahl der in ihm ausgeführten Aktionen gemessen.[1] Erzeugt ein Algorithmus stets in polynomieller Zeit Pläne, deren Anzahl von Aktionen für Instanzen der Größe n durch $f(n) \times opt$ beschränkt ist, wenn opt die Länge eines kürzestmöglichen Plans für die jeweilige Instanz ist, so heißt er $f(n)$-Approximationsalgorithmus.

Ich werde mich in dieser Arbeit der Frage widmen, für welche Funktionen f es solche $f(n)$-Approximationsalgorithmen für die betrachteten Transportprobleme geben kann.

Auch in der Studienarbeit von Robert Mattmüller [Mat05] wurde die Approximierbarkeit einer Planungsdomäne untersucht, nämlich die der sogenannten Grid-Domäne, die auch in [Hel01] vorgestellt und in Form von Entscheidungsproblemen komplexitätstheoretisch klassifiziert wurde. In der Studienarbeit von Gabi Röger [Rög05] wurde für die in neueren Planungswettbewerben verwandten Domänen Satellite, Depots und Driverlog eine entsprechende komplexitätstheoretische Klassifizierung zusammen mit Approximationsalgorithmen angegeben.

Für die Frage der Approximierbarkeit ist es neben dem Entwickeln von Approximationsalgorithmen auch wichtig, untere Schranken angeben zu können. Dies wird seit Anfang der 1990er Jahre durch Folgerungen aus dem damals bewiesenen PCP-Theorem erleichtert, mit dessen Hilfe man z. B. zeigen konnte, dass das Problem MAX-3SAT nicht für alle ε einen $(1 + \varepsilon)$-Approximationsalgorithmus haben kann. Mithilfe sogenannter approximationserhaltender Reduktionen werden wir derartige Resultate nutzen, um untere Schranken auch für die hier betrachteten Probleme herzuleiten. Hintergründe zur Approximierbarkeit und Literaturangaben finden sich in Abschnitt 2.1.1.

Das *Planen* ist ein Teilgebiet der Künstlichen Intelligenz und beschäftigt sich mit Aufgaben, bei denen für ein gegebenes Ziel eine dieses Ziel erreichende Folge von Aktionen gesucht ist (vgl. Russell und Norvig [RN03]). Die hier untersuchten Probleme fallen in den Bereich des sogenannten klassischen Planens: Der für das Planen relevante Teil der Umgebung ist statisch (d.h. nur durch eigene Aktionen veränderbar), deterministisch (ausführbare Aktionen haben vorher bekannte eindeutige Effekte), diskret (im Gegensatz zu kontinuierlich), endlich und stets vollständig bekannt.

[1] Je kürzer der Plan, desto besser.

Transportprobleme ähnlicher Art wurden auch schon in der Informatik und den Wirtschaftswissenschaften (genauer gesagt im Bereich Operations Research) betrachtet (vgl. Abschnitt 2.2.2).

1.2 Aufbau der Arbeit

Zu Beginn dieser Arbeit definiere ich in Abschnitt 2.1 die als Hintergrund nötigen Begriffe aus den Gebieten der kombinatorischen Optimierung, der Approximierbarkeit und des Planens präzise, ebenso wie die betrachteten Klassen von Transportproblemen. Danach schauen wir[2] uns schon vorliegende Ergebnisse über die Approximierbarkeit von verwandten Problemen (und deren Definitionen) in Abschnitt 2.2 an.

Im dritten Kapitel, das sich dann mit den eigentlichen Ergebnissen dieser Arbeit befasst, beginnen wir damit, mit approximationserhaltenden Reduktionen untere Schranken für die Approximierbarkeit der Transportprobleme herzuleiten. Nachdem wir danach in Abschnitt 3.2 die Unterschiede zweier Varianten dieser Probleme untersucht haben, wenden wir uns in 3.3 den oberen Schranken für die Approximierbarkeit in Form von Approximationsalgorithmen zu. Leider werden wir dabei feststellen, dass zwischen den hier gefundenen oberen und unteren Schranken zum Teil erhebliche Lücken klaffen. Kurz darauf werden wir in 3.4 weitere Varianten der Transportprobleme ansprechen, mit unterschiedlich guten Ergebnissen. In 3.5 werden dann noch untere Schranken für die Approximierbarkeit von Spezialfällen eines alternativen Planungsformalismus gezeigt, die sich als Verallgemeinerung bestimmter Transportprobleme erweisen.

[2] In dieser Arbeit verwende ich häufig „wir“ mit dem Ziel besserer Lesbarkeit. Sollte das stattdessen zu sprachlicher Verwirrung beitragen (ich bin der einzige Autor), so ist das nicht beabsichtigt.

2 Grundlagen und Verwandte Probleme

2.1 Definitionen

Kurz einige Bemerkungen zur verwendeten Notation:

Für eine Menge A bezeichnen wir mit A^* die Menge der Folgen aus Elementen von A inklusive der leeren Folge ε, für eine solche Folge w die Anzahl ihrer Folgenglieder (ihre Länge) mit $|w|$. Der Definitionsbereich einer Funktion f wird auch mit $dom(f)$ bezeichnet. Eine partielle Funktion $f : A \to B$ (d.h. $dom(f) \subset A$) kennzeichnen wir auch durch $f : A \nrightarrow B$. Für eine Funktion f mit $a \in dom(f)$ sei der Update-Operator $\oplus$ definiert durch $f \oplus \{a \mapsto b\} := f'$ mit $f'(a) = b$ und $f'(x) = f(x)$ für alle $x \neq a$. Für ein Tupel x mit benannten Komponenten bezeichnen wir mit $Name(x)$ den Wert der durch $Name$ benannten Komponente, also z. B. für einen ungerichteten Graphen G sei $V(G)$ die Menge seiner Knoten und $E(G)$ die Menge seiner Kanten. Für eine Menge A sei $Pot(A)$ ihre Potenzmenge. Für einen Knoten v eines gerichteten Graphen $G = (V, A)$ sei $\delta^+(v) := |\{\, u \,|\, (v, u) \in A \,\}|$ der Ausgangsgrad von v und $\delta^-(v) := |\{\, u \,|\, (u, v) \in A \,\}|$ der Eingangsgrad.

Außerdem werden folgende graphentheoretische Begriffe verwendet:

Als *metrischen Abschluss* eines (kantengewichteten) ungerichteten Graphen G (bzw. (G, c)) bezeichnet man den kantengewichteten Graphen $(\bar{G}, \bar{c})$ mit $V(\bar{G}) = V(G)$ und $E(\bar{G}) = \{\, \{u, v\} \,|$ es existiert ein Weg zwischen u und v in $G\}$ mit der Kantengewichtung $\bar{c}$, die jeder Kante $\{u, v\}$ die Länge des kürzesten Wegs zwischen u und v zuordnet.

Eine *Rundtour* in einem Graphen sei ein beliebiger geschlossener Weg in diesem Graphen, Wiederholungen von Knoten und Kanten sind dabei erlaubt.

Ein Graph G' heißt *Teilgraph* oder *Subgraph* eines Graphen G, wenn $V(G') \subseteq V(G)$ und $E(G') \subseteq E(G)$.

Für einen gerichteten Graphen $G = (V, A)$ bezeichnen wir mit $U(G)$ den ungerichteten Graphen (V, E) mit $E = \{\, \{u, v\} \,|\, (u, v) \in A \,\}$. Ein ungerichteter Graph heißt zusammenhängend, wenn es zwischen je zwei Punkten einen Weg gibt. Ein gerichteter Graph G heißt *schwach zusammenhängend*, wenn $U(G)$ zusammenhängend ist, und *stark zusammenhängend*, wenn es von jedem $u \in V(G)$ zu jedem $v \in V(G)$ einen gerichteten Pfad gibt. Eine *starke (bzw. schwache) Zusammenhangskomponente* eines gerichteten Graphen G ist ein maximaler stark (bzw. schwach) zusammenhängender Teilgraph von G.

2.1.1 Optimierung und Approximierbarkeit

Die *kombinatorische Optimierung* beschäftigt sich mit der Lösung diskreter Optimierungsprobleme und ist ein klassisches Teilgebiet der Informatik. Eine ebenfalls schon klassische Einführung in dies Gebiet bieten Papadimitriou und Steiglitz [PS82], einen sehr aktuellen, mathematisch orientierten und recht umfassenden Überblick über das Thema bieten Korte und Vygen [KV06].

Speziell mit der Lösung schwieriger Optimierungsprobleme beschäftigen sich Hromkovič [Hro03] (mithilfe verschiedener Techniken, z. B. Randomisierung, aber auch Approximationsalgorithmen) und Vazirani [Vaz01] (ausschließlich Approximation).

Die Approximierbarkeit von NP-Optimierungsproblemen und Approximationsalgorithmen sind Thema von Ausiello et al. [ACG$^+$99]. Bei Wegener [Weg03] findet sich ebenfalls eine Einführung in die Approximierbarkeit aus komplexitätstheoretischer Sicht. Auch in [Vaz01], Kap. 29, findet sich eine Einführung in Approximierbarkeitsresultate, im Gegensatz zu den anderen Übersichten hier auch mit ausgearbeiteten (Nicht-)Approximierbarkeitsresultaten für Set-Cover und Clique. Einen recht aktuellen Überblick über Approximierbarkeitsresultate kann man bei Trevisan [Tre04] finden.

Im Folgenden orientieren wir uns grob an [ACG$^+$99].

Definition 2.1.1 (Optimierungsproblem) Ein Optimierungsproblem ist ein 4-Tupel $P = (I, S, m, \mu)$ bestehend aus:

- einer Menge I von Instanzen, dargestellt als endliche Wörter über einem endlichen Alphabet,
- einer Funktion S, die jeder Instanz aus I eine Menge sogenannter zulässiger Lösungen (ebenfalls jeweils endliche Wörter über einem endlichen Alphabet) zuordnet,
- einer Funktion m, die jedem Paar aus einer Instanz und einer zulässigen Lösung dieser Instanz einen ganzzahligen positiven Wert (bzw. derartige Kosten) zuordnet,[1] und
- einem $\mu \in \{\min, \max\}$, das anzeigt, ob es sich um ein Minimierungs- oder ein Maximierungsproblem handelt. $\square$

Sei im Folgenden $opt(x) := \mu\{\, m(x,s) \mid s \in S(x)\,\}$ der optimale Wert einer zulässigen Lösung für die Instanz x. Zu einem solchen Optimierungsproblem korrespondieren folgende algorithmische Probleme:

1. Das *Konstruktionsproblem*: Gegeben eine Instanz $x \in I$, finde eine zulässige Lösung $s \in S(x)$ des optimalen Werts $opt(x)$ und ihren Wert.
2. Das *Auswertungsproblem*: Gegeben eine Instanz $x \in I$, berechne $opt(x)$.
3. Das *Entscheidungsproblem*: Gegeben eine Instanz $x \in I$ und eine natürliche Zahl k, entscheide ob $opt(x) \geq k$ (falls $\mu = \max$) bzw. ob $opt(x) \leq k$ (falls $\mu = \min$).

Definition 2.1.2 (NP-Optimierungsproblem, NPO) Ein Optimierungsproblem $P = (I, S, m, \mu)$ heißt *NP-Optimierungsproblem*, wenn außerdem gilt:

[1] Manchmal lässt man auch rationale oder sogar reelle Zahlen als Werte zu, z. B. wenn man Distanzen von Punkten in der Ebene betrachtet.

- Die zu I korrespondierende Sprache ist in polynomieller Zeit entscheidbar.[2]
- Es gibt ein Polynom q mit den Eigenschaften:
  - Für alle Instanzen $x \in I$ und alle zulässigen Lösungen dieser Instanzen $s \in S(x)$ gilt $|s| \leq q(|x|)$.
  - Das Problem, ob ein gegebenes s mit $|s| \leq q(|x|)$ eine zulässige Lösung für x ist (also ob $s \in S(x)$), ist in polynomieller Zeit (in $|s|$ und $|x|$) entscheidbar.
- Die Funktion m ist in polynomieller Zeit berechenbar.

Mit *NPO* bezeichnen wir die Klasse aller NP-Optimierungsprobleme. □

Kommen wir zur Definition eines Approximationsalgorithmus:

Definition 2.1.3 (Approximationsalgorithmus) Sei ein Optimierungsproblem $P = (I, S, m, \mu)$ gegeben. Ein Algorithmus A heißt *Approximationsalgorithmus* für P, wenn er für jede Instanz $x \in I$ eine zulässige Lösung $\mathrm{A}(x) \in S(x)$ zurückgibt.[3] In diesem Zusammenhang nennt man zulässige Lösungen auch *approximative Lösungen.* □

In den wenigsten Fällen kann man etwas mit einem einfachen Approximationsalgorithmus anfangen. Interessant wird es meist erst dann, wenn die zurückgegebenen Lösungen „gut“ sind im Sinne der folgenden Definition:

Definition 2.1.4 (Approximationsgüte) Für ein Optimierungsproblem $P = (I, S, m, \mu)$, eine Instanz $x \in I$ dieses Problems und eine zulässige Lösung dieser Instanz $y \in S(x)$ ist die *Approximationsgüte* dieser Lösung (auch: performance ratio) definiert durch:

$$G(x, y) := \max \left\{ \frac{m(x, y)}{opt(x)}, \frac{opt(x)}{m(x, y)} \right\}$$

Dabei stehe $opt(x)$ für den optimalen Wert einer zulässigen Lösung für die Instanz x. □

Für die im Folgenden betrachteten Minimierungsprobleme gilt also $G(x, y) = \frac{m(x,y)}{opt(x)}$.

Definition 2.1.5 ($f(n)$-Approximationsalgorithmus) Sei ein Optimierungsproblem P gegeben, ein Approximationsalgorithmus A für P und eine Funktion $f : \mathbb{N} \to (1, \infty)$. A heißt $f(n)$-*Approximationsalgorithmus* (oder $f(n)$-Approximation) *für* P, wenn für jede Instanz x von P mit $S(x) \neq \emptyset$ die Approximationsgüte der zurückgegebenen approximativen Lösung $A(x)$ durch $f(|x|)$ beschränkt ist, d.h. $G(x, A(x)) \leq f(|x|)$. □

[2] D.h. es ist in polynomieller Zeit entscheidbar, ob eine gegebene Zeichenkette die Kodierung einer Instanz darstellt. Wir geben [ACG+99] folgend diese Sprachen nicht explizit als Teil der Probleme an, sondern identifizieren Instanzen mit ihrer Kodierung über einem festen endlichen Alphabet. Der an formaleren Definitionen interessierte Leser sei an [Hro03] oder [KV06] verwiesen.

[3] Natürlich nur, wenn so eine Lösung existiert, d.h. wenn $S(x) \neq \emptyset$. Diese Definition ist Def. 3.1 aus [ACG+99]. Es versteht sich von selbst, dass „vernünftige“ Approximationsalgorithmen auch erkennen sollten, ob eine Instanz zulässige Lösungen besitzt, und in diesem Fall eine entsprechende Meldung ausgeben sollten.

Analog zu den Komplexitätsklassen für Entscheidungsprobleme (vgl. Garey und Johnson [GJ79]) teilt man auch NP-Optimierungsprobleme je nach erreichbarer Approximationsgüte in Klassen ein:

Definition 2.1.6 (PTAS, FPTAS) Ein *polynomielles Approximationsschema* (polynomial time approximation scheme, kurz: *PTAS*) für ein Optimierungsproblem P ist ein Algorithmus A, der für Eingaben $(x, \varepsilon) \in I(P) \times \mathbb{Q}_{\geq 0}$ für festes ε in polynomieller Zeit bezogen auf $|x|$ eine Lösung mit maximaler Approximationsgüte $1 + \varepsilon$ berechnet.

Ein *vollpolynomielles Approximationsschema* (fully polynomial time approximation scheme, kurz: *FPTAS*) für ein Optimierungsproblem P ist ein Algorithmus A, der für Eingaben $(x, \varepsilon) \in I(P) \times \mathbb{Q}_{\geq 0}$ in polynomieller Zeit bezogen auf $|x|$ und $\frac{1}{\varepsilon}$ eine Lösung mit maximaler Approximationsgüte $1 + \varepsilon$ berechnet.

Mit *PTAS* (bzw. *FPTAS*) bezeichnen wir auch die Klasse aller NP-Optimierungsprobleme, die ein PTAS (bzw. FPTAS) besitzen. □

Einen c-Approximationsalgorithmus ($c \in \mathbb{Q}$) nennen wir im Folgenden auch Konstant-Faktor-Approximation.

Definition 2.1.7 (*F*-APX, APX, log-APX, poly-APX, exp-APX) Gegeben sei eine Menge F von Funktionen. Die Klasse F-APX sei nun die Klasse all derjenigen NP-Optimierungsprobleme, für die für ein $f \in F$ ein in polynomieller Zeit laufender $f(n)$-Approximationsalgorithmus existiert.
Es seien insbesondere:

- APX := F_1-APX für $F_1 := \{ f : \mathbb{N} \to (1, \infty) \mid f(n) \in O(1) \}$,
- log-APX := F_2-APX für $F_2 := \{ f : \mathbb{N} \to (1, \infty) \mid f(n) \in O(\log n) \}$,
- poly-APX := F_3-APX für $F_3 := \{ f : \mathbb{N} \to (1, \infty) \mid f(n) \in \bigcup_{k>0} O(n^k) \}$ und
- exp-APX := F_4-APX für $F_4 := \{ f : \mathbb{N} \to (1, \infty) \mid f(n) \in \bigcup_{k>0} O(2^{n^k}) \}$. □

Sei nun noch PO die Klasse der NP-Optimierungsprobleme, für die eine zulässige Lösung mit optimalem Wert in polynomieller Zeit gefunden werden kann. Dann ist klar, dass PO $\subseteq$ FPTAS $\subseteq$ PTAS $\subseteq$ APX $\subseteq$ log-APX $\subseteq$ poly-APX $\subseteq$ exp-APX $\subseteq$ NPO. Wenn P $\neq$ NP, so ist sogar jede dieser Inklusionen echt (vgl. [ACG+99], S. 255).

Um für einzelne NP-Optimierungsprobleme entscheiden zu können, in welchen dieser Klassen sie liegen, ist es hilfreich, die Approximierbarkeit zweier Probleme miteinander vergleichen zu können. Dafür sehr nützlich ist folgendes Konzept:

Definition 2.1.8 (AP-Reduktion) Seien $P_1 = (I_1, S_1, m_1, \mu_1)$ und $P_2 = (I_2, S_2, m_2, \mu_2)$ NP-Optimierungsprobleme. Dann heißt P_1 *AP-reduzierbar auf* P_2 (geschrieben $P_1 \leq_{AP} P_2$) genau dann, wenn es ein Tripel (f, g, α) gibt mit:

- f und g sind in polynomieller Zeit berechenbare Funktionen und $\alpha \in \mathbb{R}_{\geq 1}$

- $f : I_1 \times \mathbb{R}_{>1} \to I_2$ bildet Paare aus Instanzen von P_1 und Zahlen > 1 auf Instanzen von P_2 ab mit:
$$S_1(x) \neq \emptyset \implies S_2(f(x,r)) \neq \emptyset \quad \forall r > 1$$
- $g : I_1 \times (\bigcup_{z\in I_2} S_2(z)) \times \mathbb{R}_{>1} \nrightarrow (\bigcup_{z\in I_1} S_1(z))$ bildet Tripel aus einer Zahl $r > 1$, einer Instanz x von P_1 und einer zulässigen Lösung für $f(x,r)$ bezüglich P_2 auf eine zulässige Lösung von P_1 ab mit:
$$\forall x \in I_1\, \forall r > 1\, \forall y \in S_2(f(x,r))\colon g(x,y,r) \in S_1(x)$$
- α begrenzt die Verschlechterung der Approximationsgüte bei der Rückabbildung g wie folgt:[4]
$$\forall x \in I_1\, \forall r > 1\, \forall y \in S_2(f(x,r))\colon$$
$$G_2(f(x,r),y) \leq r\, (= 1 + (r-1)) \implies G_1(x, g(x,y,r)) \leq 1 + \alpha(r-1)$$

Das Tripel (f,g,α) wird dann auch als *AP-Reduktion von P_1 auf P_2* bezeichnet. □

Die AP-Reduzierbarkeit ist als Relation aufgefaßt reflexiv und transitiv, erfüllt also die für Reduktionen üblichen Eigenschaften.

Lemma 2.1.9 *Wenn $P_1 \leq_{AP} P_2$ und $P_2 \in APX$ (bzw. PTAS), so ist auch $P_1 \in APX$ (bzw. PTAS).*

Beweis Siehe z. B. [ACG+99], Lemma 8.1. ■

Ein Vorteil von AP-Reduktionen ist ihre universelle Anwendbarkeit. So gibt es zwar noch viele andere Reduktionsarten für Optimierungsprobleme (für eine Übersicht vgl. [ACG+99], S. 284, daneben gibt es noch sogenannte lückenerhaltende Reduktionen, vgl. [Vaz01], Kap. 29), für diese Arbeit genügt aber die AP-Reduzierbarkeit, die sich laut [ACG+99] auch als Standardreduzierbarkeit durchzusetzen scheint.

Man kann auch für Klassen von Optimierungsproblemen analog zu Klassen von Entscheidungsproblemen einen Vollständigkeitsbegriff definieren:

Definition 2.1.10 (A-hart, A-vollständig) Sei P ein NP-Optimierungsproblem und A eine Komplexitätsklasse für Optimierungsprobleme. Dann heißt P *A-hart*, wenn für alle $P' \in A$ gilt : $P' \leq_{AP} P$. P heißt *A-vollständig*, wenn P A-hart ist und $P \in A$. □

Im folgenden Absatz orientieren wir uns an Trevisan [Tre04]:

Anfang der 1990er Jahre wurde von Arora et al. das PCP-Theorem bewiesen [ALM+98]. Dieser Satz gibt nicht nur eine Charakterisierung von NP über probabilistische Beweissysteme an (kurz und unerläutert: $NP = PCP(\log(n), 1)$), sondern erlaubte es etwa auch zu zeigen, dass das Problem MAX-3SAT kein PTAS besitzt.[5] Dieses Resultat ließ sich dann mithilfe der schon bekannten Resultate von Papadimitriou

[4]Dabei stehe G_i für die Approximationsgüte bezogen auf Problem P_i.
[5]Unter der Voraussetzung P $\neq$ NP.

und Yannakakis [PY91] auf eine Reihe weiterer Probleme ausdehnen. In [PY91] wurde nämlich eine logikbasierte Definition einer Klasse von Approximationsproblemen namens MAXSNP angegeben und gezeigt, dass MAX-3SAT vollständig in dieser Klasse bezüglich bestimmter approximationserhaltender Reduktionen ist (der L-Reduktionen). Seitdem wurden verschiedene Varianten des PCP-Theorems genutzt, um stärkere Nichtapproximierbarkeitsresultate herzuleiten.[6] In dem Übersichtskapitel von Arora und Lund [AL96] werden solche Resultate in 4 Klassen eingeteilt: $\Theta(c)$-Approximierbarkeit für ein $c \in \mathbb{Q}$ (z. B. MAX-3SAT), $\Theta(\log(n))$-Approximierbarkeit (z. B. das Set-Cover-Problem), $2^{(\log n)^c}$-Nichtapproximierbarkeit (z. B. das $Labelcover_{min}$-Problem) und n^c-Approximierbarkeit (z. B. das Max-Clique-Problem). Die Hoffnung, alle natürlichen Optimierungsprobleme würden sich bezüglich ihrer Approximierbarkeit in eine dieser Klassen einordnen lassen, erfüllte sich nicht (vgl. [Tre04], Abschnitt 7.4): So ist etwa das Group-Steiner-Tree-Problem polylogarithmisch approximierbar. Näheres findet der interessierte Leser etwa in [Tre04], aber auch in den zu Beginn dieses Abschnitts angegebenen Referenzen.

Hier nur eine Definition eines Problems und dessen Nichtapproximierbarkeitsresultat. Die anderen verwendeten Approximierbarkeitsresultate für Probleme werden an den entsprechenden Stellen im Text eingeführt.

Definition 2.1.11 (Set-Cover-Problem) Die Instanzen des *Set-Cover-Problems* (das wir auch SetCover nennen) bestehen aus jeweils einer Grundmenge U und einer Menge $\{S_1, \ldots, S_k\}$ von Teilmengen von U. Zulässige Lösungen einer Instanz bestehen aus einer Teilmenge $\{S_{i_1}, \ldots, S_{i_j}\} \subseteq \{S_1, \ldots, S_k\}$, sodass jedes Element aus U auch Element einer gewählten Teilmenge S_{i_r} ist. Man sagt auch, das Element sei durch die Menge S_{i_r} überdeckt. Die Kosten einer Lösung sind gegeben durch die Anzahl der gewählten Teilmengen (also hier j). Es handelt sich um ein Minimierungsproblem. □

Satz 2.1.12 ([Fei98]) *Das Set-Cover-Problem ist unter der Voraussetzung* $NP \not\subseteq DTIME(n^{\log\log(n)})$ *nicht* $((1-\varepsilon)\ln|U|)$*-approximierbar* $(\forall \varepsilon > 0)$. □

2.1.2 Planen

Die Definitionen in diesem und dem nächsten Abschnitt orientieren sich sehr stark an denen aus [Hel01].

Definition 2.1.13 (Zustandsmodell) Ein *Zustandsmodell* ist ein Fünf-Tupel $M = (S, s_0, S_G, A, \delta)$ mit

- einer endlichen Menge von Zuständen S,
- einem Anfangszustand $s_0 \in S$,

[6] Unter verschiedenen komplexitätstheoretischen Voraussetzungen. Im Folgenden wird in den Beweisen und den Formulierungen der Sätze zwar immer auf diese Voraussetzungen eingegangen, im normalen Text jedoch nicht immer. In solchen Fällen gelten die Aussagen natürlich trotzdem nur unter den in den Sätzen beschriebenen Voraussetzungen.

- einer Menge von Zielzuständen $S_G \subseteq S$,
- einer endlichen Menge von Aktionen A und
- einer partiellen Übergangsfunktion $\delta : S \times A \nrightarrow S$.

Die Menge der in einem Zustand s *anwendbaren Aktionen* ist definiert als $\{ a \in A \mid (s, a) \in dom(\delta) \}$. Die Menge aller Zustandsmodelle sei $\mathcal{M}$. □

Im Folgenden sei stets ein Zustandsmodell $M = (S, s_0, S_G, A, \delta)$ gegeben.

Definition 2.1.14 (erweiterte Übergangsfunktion) Die *erweiterte Übergangsfunktion* $\hat{\delta} : S \times A^* \nrightarrow S$ ist diejenige partielle Funktion mit minimalem Definitionsbereich, die Folgendes erfüllt:

- $\hat{\delta}(s, \varepsilon) = s$
- $\hat{\delta}(s, aw) = \hat{\delta}(\delta(s, a), w) \quad \forall a \in A, w \in A^*$,
  wenn $(s, a) \in dom(\delta)$ und $(\delta(s, a), w) \in dom(\hat{\delta})$ □

Definition 2.1.15 (Plan) Eine Folge von Aktionen $p \in A^*$ heißt *Plan für* M $:\Longleftrightarrow$ $(s_0, p) \in dom(\hat{\delta})$ und $\hat{\delta}(s_0, p) \in S_G$. Ein Plan p heißt *optimal*, wenn seine Länge minimal ist unter den Längen aller Pläne für M. M heißt *lösbar*, wenn ein Plan für M existiert. Die Menge aller Pläne für M bezeichnen wir hier[7] mit $P(M)$. □

Definition 2.1.16 (Planungsdomäne) Eine *Planungsdomäne* ist eine Funktion $\mathcal{D} : L_{\mathcal{D}} \to \mathcal{M}$ für eine feste Sprache $L_{\mathcal{D}}$ über einem festen endlichen Alphabet Σ. $L_{\mathcal{D}}$ heißt dann die *Kodierungssprache* dieser Domäne. Jedes $w \in L_{\mathcal{D}}$ heißt eine *Planungsinstanz* und $|w|$ deren *Kodierungslänge*. □

Wir werden im Folgenden $L_{\mathcal{D}}$ nie explizit angeben und identifizieren hier also Instanzen mit ihren Kodierungen, wie z. B. auch in der nächsten Definition. Man kann sich vorstellen, dass eine Planungsdomäne einer Sprache ihre Semantik in Form von Zustandsmodellen zuweist.

Helmert führt in [Hel01] diese für Planungsprobleme relevanten Entscheidungs- und Suchprobleme ein:

Definition 2.1.17 (Entscheidungs- und Suchprobleme für Planungsdomänen) Sei $\mathcal{D}$ eine Planungsdomäne.

Dann sei *PlanEx-*$\mathcal{D}$ die Sprache aller lösbaren Instanzen für $\mathcal{D}$.

PlanLen-$\mathcal{D}$ sei die Sprache aller Paare (i, l), die aus einer Instanz i von $\mathcal{D}$ und einer Zahl l bestehen, sodass es für i einen Plan der Länge höchstens l gibt.

Das Suchproblem *PlanGen-*$\mathcal{D}$ besteht daraus, für eine Instanz i von $\mathcal{D}$ einen Plan p beliebiger Länge zu finden, falls diese Instanz lösbar ist, und ansonsten „No“ auszugeben.

Das Suchproblem *PlanOpt-*$\mathcal{D}$ besteht daraus, für eine Instanz i von $\mathcal{D}$ einen Plan p optimaler Länge zu finden, falls diese Instanz lösbar ist, und ansonsten „No“ auszugeben. □

[7] Dieser Teil der Definition ist nicht aus [Hel01] übernommen.

M. Helmert zeigt dann noch, dass die Suchprobleme wenigstens so schwer sind wie die korrespondierenden Entscheidungsprobleme und erreicht so eine komplexitätstheoretische Klassifikation auch der Suchprobleme.

Auf naheliegende Weise können wir Planungsprobleme als Optimierungsprobleme im Sinne von Def. 2.1.1 auffassen (einmal nicht aus [Hel01]):[8]

Definition 2.1.18 (Planungsdomänen als Optimierungsprobleme) Sei $\mathcal{D} : L \to \mathcal{M}$ eine Planungsdomäne. Dann sei *OptPlan-*$\mathcal{D} = (\mathcal{D}(L), S, m, \min)$ mit:

- $S(M) = P(M)$ für alle $M \in \mathcal{D}(L)$ und
- $m(M, p) = |p|$ für alle $M \in \mathcal{D}(L)$ und $p \in S(M)$. □

2.1.3 Die Transportprobleme

Ab jetzt beschränken wir uns für diesen Abschnitt auf die hier hauptsächlich betrachteten Transportdomänen und beginnen dabei mit der Definition einzelner Instanzen.[9]

Definition 2.1.19 (Transport-Instanz) ([Hel01], Def. 3.1) Eine *Transport-Instanz* ist ein 10-Tupel $I = (V, E, M, P, P_G, loc_0, loc_G, cap, fuel, road)$. Dabei sind

- (V, E) ein endlicher ungerichteter Graph (auch *Karte* genannt); seine Knoten heißen auch *Orte* und seine Kanten *Straßen*;
- M eine endliche Menge von Fahrzeugen,
- P eine endliche Menge transportierbarer Gegenstände,
- $P_G \subseteq P$ die Menge der Zielgegenstände,
- $loc_0 : (M \cup P) \to V$ die Menge der Anfangsorte,
- $loc_G : P_G \to V$ die Menge der Zielorte,
- $cap : M \to N_0$ die Kapazitätsfunktion,
- $fuel : V \to N_0$ die Treibstoffsfunktion und
- $road : M \to Pot(E)$ die Bewegungseinschränkungsfunktion.

Hierbei seien V, M und P disjunkte Mengen. □

---

[8] Strenggenommen sind die hier definierten Optimierungsprobleme keine NP-Optimierungsprobleme im Sinne von Definition 2.1.2, da auch beliebig lange Pläne noch zulässige Lösungen darstellen können (z.B. bei Herumfahren von Fahrzeugen im Kreis). Wir können uns aber auf Pläne polynomieller Länge beschränken (vgl. [Hel03], Thm. 11 und Thm. 13).

[9] Zunächst sozusagen syntaktisch.

Wir werden mit $c(v,w)$ die Kosten (d.h. die Länge) des kürzesten Wegs von v nach w bezeichnen und $road(m)$ für ein Fahrzeug m auch als dessen Karte.

Die Kodierungslänge einer Instanz ist mindestens und ungefähr[10] $|V| + |E| + |M| + 2|P| + \Sigma_{m\in T}|road(m)|$, wenn wir mit T eine Teilmenge der Fahrzeuge bezeichnen, in der für jede der gegebenen Fahrzeugkarten ein Fahrzeug enthalten ist.

Da es im Fall beschränkten Treibstoffs schon NP-hart ist, überhaupt einen (beliebig schlechten) zulässigen Plan zu finden (vgl. [Hel01]), sodass die entsprechenden Probleme nicht einmal in exp-APX sein können und die Untersuchung ihrer Approximierbarkeitseigenschaften keine interessanten Ergebnisse liefern würde, werden wir uns im Folgenden auf Instanzen beschränken, in denen „genügend“ Treibstoff vorhanden ist und $fuel$ nicht mehr explizit angeben. Insbesondere lassen wir $fuel$ auch schon bei der folgenden Definition der Transportdomäne weg.

Definition 2.1.20 (Transport-F_∞-Domäne) (eine Einschränkung der Def. 3.2 aus [Hel01]) Sei L die Menge aller Transportinstanzen[11]. Die *Transport-F_∞-Domäne* sei die wie folgt Instanzen auf Zustandsmodelle abbildende Funktion $D : L \to \mathcal{M}$:

$(V, E, M, P, P_G, loc_0, loc_G, cap, road) \mapsto (S, s_0, S_G, A, \delta)$ mit:

- $S := (M \to V) \times (P \to V \cup M)$
- $s_0 := (loc_{0|M}, loc_{0|P})$
- $S_G := \{(l_M, l_P) \in S \mid l_{P|P_G} = loc_G\}$
- $A := \{\, move_{m,v} \mid m \in M, v \in V\,\} \cup \{\, pickup_{m,p} \mid m \in M, p \in P\,\} \cup \{\, drop_{m,p} \mid m \in M, p \in P\,\}$
- $\forall m \in M, v, v' \in V, s = (l_M, l_P) \in S :$
  $\quad l_M(m) = v \wedge \{v, v'\} \in road(m)$
  $\qquad \Longrightarrow \delta(s, move_{m,v'}) := (l_M \oplus \{m \mapsto v'\}, l_P)$
  $\forall m \in M, p \in P, s = (l_M, l_P) \in S$:
  $\quad l_M(m) = l_P(p) \wedge |\{\, p \in P \mid l_P(p) = m\,\}| < cap(m)$
  $\qquad \Longrightarrow \delta(s, pickup_{m,p}) := (l_M, l_P \oplus \{p \mapsto m\})$
  $\forall m \in M, p \in P, s = (l_M, l_P) \in S$:
  $\quad l_P(p) = m$
  $\qquad \Longrightarrow \delta(s, drop_{m,p}) := (l_M, l_P \oplus \{p \mapsto l_M(m)\})$
  sowie: δ sei sonst undefiniert. □

Ein Zustand ist also gekennzeichnet durch die Aufenthaltsorte der Fahrzeuge (M) und der Gegenstände (P). Die Startorte sind dabei durch loc_0 festgelegt. Ziel ist es, die

[10] Strenggenommen müßte man z. B. statt $|M|$ etwa $|M| \log |V| + |M| \log(\max\{cap(m)|m \in M\})$ ($\leq |M|(\log |V| + \log |P|)$) oder etwas Ähnliches schreiben. Da diese Angabe aber nur eine Orientierung bieten soll und wir uns nicht auf eine spezielle Kodierung (etwa in den Planungssprachen STRIPS oder PDDL) festlegen wollen, verzichten wir auf größere Genauigkeit.

[11] Wir identifizieren eine Instanz mit ihrer Kodierung.

(Ziel-)Gegenstände $P_G \subseteq P$ an ihre durch loc_G festgelegten Zielorte zu befördern. Dafür stehen die Fahrzeuge zur Verfügung. Ein Fahrzeug m kann bis zu $cap(m)$ Gegenstände gleichzeitig transportieren und nur die Wege aus $road(m)$ befahren. Aktionen sind die Bewegungen von Fahrzeugen entlang einer Kante sowie das Aufnehmen (bei genügend freier Kapazität) und Ablegen einzelner Gegenstände durch ein Fahrzeug, das sich an demselben Ort befindet.

Wir werden im Folgenden stets o.B.d.A. annehmen, dass $P_G = P$ in allen Transportinstanzen gilt.

Definition 2.1.21 (Spezialfälle der Transport-F_∞-Domäne) (nach [Hel01], Def. 3.4) Für $i \in \{1, \infty, *\}$ und $j \in \{1, +, *\}$ ist die Transport-F_∞-C_iM_j-Domäne die Einschränkung der Transport-F_∞-Domäne auf solche Instanzen, die die folgenden Bedingungen erfüllen:

- Wenn $i = 1$ ist, so $cap \equiv 1$. Jedes Fahrzeug kann zu jedem Zeitpunkt nur einen Gegenstand tragen.
- Wenn $i = \infty$ ist, so $cap \equiv |P|$. Jedes Fahrzeug kann zu jedem Zeitpunkt beliebig viele der Gegenstände tragen.
- Wenn $i = *$ ist, so gibt es keine Einschränkungen bezüglich der Kapazitäten der Fahrzeuge (d.h. beliebige Kapazitäten sind zugelassen).
- Wenn $j = 1$ ist, so $|M| = 1$ und $road \equiv E$. Es gibt also nur ein Fahrzeug. Dieses kann die ganze Karte befahren.
- Wenn $j = +$ ist, so $road \equiv E$. Es kann also jedes Fahrzeug die ganze Karte befahren (es gibt nur einen Fahrzeugtyp).
- Wenn $j = *$ ist, so gibt es keinerlei Einschränkungen bezüglich Anzahl oder Typ der Fahrzeuge. □

Wir unterscheiden Fahrzeuge nach der von ihnen befahrbaren Karte. Können zwei dieselbe Karte befahren, sagen wir, sie haben den gleichen Typ. Abkürzend sprechen wir häufig nur von C_iM_j statt von der Transport-F_∞-C_iM_j-Domäne und identifizieren C_iM_j mit dem zugehörigen in Def. 2.1.18 definierten Optimierungsproblem. Bei Angabe konkreter Instanzen zu diesen Klassen werden häufig die durch die Klasse schon festgelegten Komponenten weggelassen (z. B. *road* bei M_1- oder M_+-Instanzen).

Wir sagen, eine Domäne C_iM_j ist eine *Spezialisierung* von C_kM_l, wenn jede Instanz von C_iM_j auch als Instanz von C_kM_l aufgefasst werden kann und sich dabei die Menge der zulässigen Lösungen nicht verändert. Diese Spezialisierungsbeziehungen für die eben definierten Klassen sind auch in Abb. 2.1 auf S. 18 angegeben.

Für diese Klassen gilt außerdem:

Satz 2.1.22 *([Hel01], Thm. 3.5, Thm. 3.11, Thm. 3.16)*
PlanGen-$C_iM_k \in P$.
PlanLen-C_iM_k ist NP-vollständig. □

$$\begin{tikzcd}
 & C_*M_* & \\
C_1M_* \arrow[ur] & & C_\infty M_* \arrow[ul] \\
 & C_*M_+ \arrow[uu] & \\
C_1M_+ \arrow[uu] \arrow[ur] & & C_\infty M_+ \arrow[uu] \arrow[ul] \\
 & C_*M_1 \arrow[uu] & \\
C_1M_1 \arrow[uu] \arrow[ur] & & C_\infty M_1 \arrow[uu] \arrow[ul]
\end{tikzcd}$$

Abbildung 2.1: Die Spezialisierungsbeziehungen zwischen den Transport-F_∞-C_iM_j-Domänen. Ein Pfeil von C_iM_j zu C_kM_l bedeute dabei, dass C_iM_j eine Spezialisierung von C_kM_l ist. Eine mit den in dieser Arbeit erzielten Approximationsresultaten ergänzte Version findet sich in Abb. 4.1 auf S. 68.

Genauer ist PlanLen-$C_\infty M_1$ schon NP-vollständig, wenn die Karte ein planarer Graph sein muss und alle Gegenstände am selben Startort wie das Fahrzeug beginnen müssen oder wenn die Karte ein vollständiger Graph sein muss ([Hel01], Thm. 3.11, Thm. 3.13). PlanLen-C_1M_1 ist schon NP-vollständig, wenn die Karte ein Grid-Graph[12] sein muss und die Gegenstände nur zu benachbarten Positionen gebracht werden dürfen ([Hel01], Thm. 3.16).

Damit ist bezüglich der Approximierbarkeit dieser Probleme erst klar, dass sie sich in exp-APX\PO befinden. Eine genauere Klassifikation wird in dieser Arbeit gegeben.

2.2 Verwandte Probleme

2.2.1 Handlungsreisende

Eines der bekanntesten Optimierungsprobleme überhaupt ist sicher das Problem des Handlungsreisenden.

[12] Informell ein rechteckiger endlicher Ausschnitt aus $\mathbb{N} \times \mathbb{N}$.

Definition 2.2.1 (Traveling-Salesman-Problem) Gegeben sei ein vollständiger, kantengewichteter Graph. Gesucht ist beim *Traveling-Salesman-Problem* (kurz TSP) eine Rundtour minimalen Gewichts, die jeden Knoten genau einmal besucht. □

Für beliebige Instanzen ist das TSP sogar NPO-vollständig. Glücklicherweise kann man sich üblicherweise auf metrische Instanzen beschränken, d.h. solche, deren Distanzen die Dreiecksungleichung erfüllen. Dieses sogenannte Metrische TSP ist nun nur noch APX-vollständig [PY93] und 1.5-approximierbar durch den Algorithmus von Christofides (vgl. z. B. [JP85]). Die beste bisher bekannte untere Schranke für die Approximierbarkeit des Metrischen TSP ist $\frac{220}{219}$ [PV00].

Fordert man nur, dass jeder Knoten mindestens einmal besucht wird und lässt dafür beliebige (zusammenhängende) Graphen als Eingabe zu (auch nicht-vollständige), so ist diese Variante zum metrischen TSP äquivalent, da man den metrischen Abschluss des Eingabegraphen betrachten kann.

In der Praxis werden für das Metrische TSP sehr verschiedene Lösungs- bzw. Approximationsverfahren angewandt, so z. B. Verfahren, die auf lokaler Suche beruhen, wie k-OPT, der Lin-Kernighan-Algorithmus oder Simulated-Annealing-Verfahren, aber auch LP-Relaxierungen, bei denen man also das TSP als (ganzzahliges) Lineares Programm formuliert. Man untersucht auch optimale Lösungsverfahren, z. B. Branch-and-Cut-Verfahren. Zu diesen Themen vergleiche man auch Korte und Vygen [KV06], Kap. 21. Wir beschränken uns hier jedoch auf Approximationsalgorithmen mit polynomieller Laufzeit und beweisbar guter Approximationsgüte.

Eine wichtige Klasse von Approximationen für das TSP ist die Klasse der sogenannten Einfügeheuristiken (Rosencrantz, Stearns und Lewis [RSL77], vgl. auch [JP85]): Hierbei wird eine Rundtour durch einen Teil des Graphen iterativ durch Einfügen einzelner Knoten an die bestmögliche Stelle zu einer Tour durch den ganzen Graphen erweitert. Die einzelnen Heuristiken unterscheiden sich nur in der Wahl des jeweils nächsten einzufügenden Knoten. Zwei der dort vorgestellten Heuristiken, die Nearest-Insertion- und die Cheapest-Insertion-Heuristik, erreichen Approximationsfaktoren von 2 und korrespondieren grob zum MST-Algorithmus von Prim. Ähnliche Heuristiken wie die Nearest-Merger-Heuristik, die eher zum MST-Algorithmus von Kruskal korrespondiert, erreichen ebenfalls einen Approximationsfaktor von 2.

Fordert man beim TSP statt einer Rundtour den kürzestmöglichen Pfad, der alle Knoten genau einmal besucht, so spricht man vom Traveling-Salesman-Pfad-Problem (kurz TSPP). Wenn keiner der Endpunkte dieses Pfades fest vorgegeben ist, so ist das TSPP 1.5-approximierbar, ebenso wenn nur Anfangs- oder Endpunkt fest vorgegeben ist (Hoogeveen [Hoo90], jeweils ähnlich dem Christofides-Algorithmus). Sind beide Endpunkte vorgegeben, so ist die beste bekannte Approximation eine 5/3-Approximation ([Hoo90] und [GHK+00]).

2.2.2 Vehicle Routing

Eine kurze Übersicht über einige sogenannte Vehicle-Routing-Probleme (k-Delivery-TSP, Dial-a-Ride-Problem, k-Tour-Cover) findet sich bei Asano et al. [Asa00]. Gemeinsames

Kennzeichen dieser Problemklasse ist, dass die Probleme das Zustellen von Gegenständen durch Fahrzeuge bestimmter Kapazität betreffen. Eine etwas ältere Übersicht über Vehicle-Routing-Probleme findet man auch bei Federgruen und Simchi-Levi [FS95].

Das am engsten mit den hier betrachteten Transportproblemen für ein einzelnes Fahrzeug verwandte Problem ist vermutlich das sogenannte präemptive Capacitated-Dial-a-Ride-Problem.

Definition 2.2.2 (CDARP, pCDARP) Eine Instanz des *Capacitated-Dial-a-Ride-Problem* (kurz: *CDARP*) besteht aus einem ungerichteten, kantengewichteten Graphen (V, E, w), einer Menge $P \subseteq V \times V$ von Paaren $p = (s(p), t(p))$, einer ganzen Zahl $k \geq 1$ und einem Startknoten $v_0 \in V$. Gesucht ist eine Rundtour minimalen Gewichts von v_0 zu v_0, während der ein Fahrzeug jeden Gegenstand p von seinem Startpunkt $s(p)$ zu seinem Zielpunkt $t(p)$ transportiert, ohne jemals mehr als k Gegenstände gleichzeitig zu transportieren. Dabei dürfen Gegenstände nicht vor ihrem Zielpunkt abgelegt werden.
Ist es doch erlaubt, Gegenstände an beliebigen Punkten abzulegen und später wieder aufzunehmen, so spricht man vom *präemptiven CDARP* (kurz: *pCDARP*). □

Die wesentlichen Unterschiede zu C_*M_1 sind zum einen die fehlenden Kosten für das Aufnehmen und Ablegen von Gegenständen, zum anderen die nichtuniformen Kosten für das Durchlaufen von Kanten des Graphen. Außerdem wird beim CDARP statt eines Pfades stets eine Rundtour gesucht.

Da es nicht verboten ist, in der Rundtour Knoten mehrfach zu besuchen, kann man immer annehmen, dass die Kantenkosten die Dreiecksungleichung erfüllen (man betrachte sonst den metrischen Abschluss des Graphen). Manchmal wird als Eingabe statt eines Graphen auch nur ein allgemeiner metrischer Raum gefordert.

Während die Einschränkung des CDARP auf Pfade schon NP-hart ist [Gua98], aber immerhin 3-approximierbar [KRW00], ist das pCDARP auf Pfaden noch in polynomieller Zeit lösbar [Gua98]. Für die Einschränkung auf eine spezielle Klasse höhenbalancierter Bäume wird bei Charikar und Raghavachari [CR98] für das DARP eine $O(\sqrt{k})$-Approximation angegeben, für das pCDARP eine 2-Approximation. Diese lassen sich mithilfe eines Resultats über die Einbettbarkeit von Metriken aus [FRT03] in eine $O(\sqrt{k} log(n))$-Approximation für das allgemeine DARP und eine $O(log(n))$-Approximation für das allgemeine pCDARP umwandeln.[13]

In [CR98] wird außerdem gezeigt, daß die optimale nichtpräemptive Lösung einer Instanz um einen Faktor von $\Omega(k^{2/3})$ schlechter sein kann als die optimale präemptive Lösung derselben Instanz. Darauf werden wir in Abschnitt 3.2.3 noch einmal zurückkommen.

Legt man die Kapazität des Fahrzeugs auf 1 fest, entsteht eine neue Problemvariante mit Ähnlichkeiten zu C_1M_1:

Definition 2.2.3 (Stacker-Crane-Problem/DARP, pDARP) Das *Stacker-Crane-Problem* (auch *DARP* genannt) ist das CDARP eingeschränkt auf Fahrzeuge der Kapazität 1. Die analoge Einschränkung des pCDARP heiße *pDARP*. □

[13] In [CR98] wurde noch ein älteres Resultat benutzt, das statt eines Verlustfaktors von $O(log(n))$ einen Verlustfaktor von $O(log(n)log(log(n))$ ergab.

Für Pfade sind sowohl DARP als auch pDARP in polynomieller Zeit lösbar (für letzteres s. [AK88]). Während für Bäume das pDARP immer noch in polynomieller Zeit lösbar ist [FG92], ist das DARP eingeschränkt auf solche Instanzen schon NP-hart, aber noch $\frac{5}{4}$-approximierbar (beides in [FG93]). Allgemeine Instanzen des DARP lassen sich 1.8-approximieren (Frederickson, Hecht und Kim [FHK78]), solche des pDARP 2.5-approximieren (das Swapping-Problem aus [AH92] ist eine Verallgemeinerung des pDARP). Die 1.8-Approximation für das DARP besteht aus der Anwendung zweier Algorithmen namens SmallArcs und LargeArcs, auf deren letzteren wir in Abschnitt 3.3 noch zurückkommen werden. Die bessere von diesen Algorithmen ausgegebene Lösung ergibt dabei die 1.8-Approximation.

Ein ähnliches Problem wie das CDARP, allerdings mit gleichartigen zuzustellenden Gegenständen, ist das sogenannte k-Delivery-TSP.

Definition 2.2.4 (k-Delivery-TSP) Gegeben sind ein ungerichteter, kantengewichteter Graph $G = (V, E, w)$, zwei Multimengen S und T über V mit $|S| = |T|$ (Start- und Zielorte der gleichartigen Gegenstände), eine natürliche Zahl k und ein Knoten $v_0 \in V$ (Kapazität und Startort des einzigen Fahrzeugs). Gesucht ist eine Rundtour minimaler Kosten von v_0 zu v_0, während der das Fahrzeug alle Zielorte aus T mit je einem Gegenstand von den Startorten aus S versorgt ohne dabei zu einem Zeitpunkt mehr als k Gegenstände geladen zu haben. $\square$

Dieses Problem ist deshalb interessant, weil es auch für beliebige Kapazitäten des Fahrzeugs und auf allgemeinen Graphen noch konstant-Faktor-approximierbar ist. Eine 9.5-Approximation von Chalasani und Motwani [CM99] wurde über eine $(7 - \frac{3}{k})$-Approximation aus [AB99] von Charikar, Khuller und Raghavachari [CKR01] zu einer $6\frac{1}{6}$-Approximation für die nichtpräemptive und eine $(6 + \frac{k}{|S|})$-Approximation für die präemptive Variante dieses Problems erweitert.[14]

Setzt man hier die Kapazität auf 1 fest, so entsteht das 2-approximierbare 1-Delivery-TSP, das auch Bipartites TSP genannt wird [CM99]. Ist die Kapazität immer wenigstens $|S|$, so spricht man vom ∞-Delivery-TSP, das auch 2-approximierbar ist (vgl. z. B. [CM99]).

Wiederum hiermit eng verwandt ist das sogenannte k-Tour-Cover-Problem, das auch unter dem Namen Capacitated-Vehicle-Routing-Problem (CVRP) bekannt ist:

Definition 2.2.5 (CVRP) Das CVRP ist das k-Delivery-TSP, eingeschränkt auf Instanzen, in denen alle Gegenstände und das Fahrzeug denselben Startort haben (das sogenannte Depot). $\square$

Das CVRP ist 2.5-approximierbar (Altinkemer und Gavish [AG90]). Man beachte, dass es für das CVRP weder relevant ist, ob die Gegenstände gleichartig sind oder jeweils einen festen eigenen Zielort haben, noch, ob sich im Depot eines oder mehrere Fahrzeuge befinden.

[14] Die in der Einleitung und Zusammenfassung von [CKR01] behauptete 5-Approximation bezieht sich auf eine Version, in der der Startpunkt des Fahrzeugs beliebig gewählt werden kann und die Kapazität k eine gerade Zahl ist.

Man kann das CVRP wiederum verallgemeinern zum sogenannten Multiple-Depot--Capacitated-Vehicle-Routing-Problem (MCVRP), das mehrere Depots besitzt. In jedem dieser Depots befinden sich hinreichend viele der gleichartigen Gegenstände, um alle Ziele zu beliefern, und ebensoviele Fahrzeuge gleicher einheitlicher Kapazität. Minimiert wird auch dort die Gesamtlänge der Touren. Je nachdem, ob Fahrzeuge zu ihrem Startdepot zurückkehren müssen, oder zu einem beliebigen zurückkehren können, ist das Problem 4-approximierbar oder 2.5-approximierbar (vgl. [FS95]).

Die in der Literatur betrachteten Vehicle-Routing-Fälle beschränken sich also meist auf ein einzelnes Fahrzeug oder auf gleichartige Gegenstände oder nehmen an, dass alle Gegenstände und das Fahrzeug am selben Punkt starten.

Eine weitere Variante des Themas Routing entsteht, wenn man in (gerichteten, ungerichteten oder gemischten) Graphen Mengen von Kanten durchlaufen muss statt Mengen von Knoten besuchen zu müssen. Auch unter dem Überbegriff „Arc Routing“ bekannt, umfasst diese Klasse z. B. die sogenannten Postbotenprobleme wie das Rural-Postman-Problem oder das Chinese-Postman-Problem. Ebenso kann man das oben erwähnte Stacker-Crane-Problem als einen Vertreter dieser Klasse ansehen. Eine Übersicht über Arc-Routing-Probleme findet sich z. B. bei Assad und Golden [AG95]. Für uns sind diese im Folgenden nicht mehr relevant.

2.2.3 Steinerbäume und -wälder

Ein ebenfalls verwandtes und auch in Form einer Verallgemeinerung in den Approximationsalgorithmen aus Abschnitt 3.3 verwendetes klassisches Optimierungsproblem ist das sogenannte Steinerbaum-Problem (vgl. auch Korte und Vygen [KV06], Kap. 20, oder Vazirani [Vaz01], Kap. 3).

Definition 2.2.6 (Steinerbaum, Steinerbaum-Problem) Eine Instanz des Steinerbaum-Problems besteht aus einem ungerichteten Graphen $G = (V, E)$ mit Kantengewichten $c : E \to \mathbb{R}_+$ und einer Teilmenge $R \subseteq V$. Ein Teilbaum S von G, dessen Knotenmenge R umfasst, heißt *Steinerbaum* für R in G. Die Knoten aus $V(S) \setminus R$ heißen dabei *Steinerknoten*. Ziel des *Steinerbaum-Problems* ist es, einen Steinerbaum minimaler Kosten für R in G zu finden. □

Das Steinerbaum-Problem ist APX-hart, selbst für uniforme Kantengewichte (vgl. [KV06], Satz 20.3, oder den Originalartikel [BP89]). Andererseits ist das euklidische Steinerbaum-Problem ebenso wie das sogenannte Manhattan-Steinerbaum-Problem (bzw. das Steinerbaum-Problem eingeschränkt auf Grid-Graphen) in PTAS (vgl. [KV06], Abschnitt 20.1).

Ein aus einem minimalen Spannbaum des metrischen Abschlusses des Graphen hervorgegangener Steinerbaum[15] stellt eine 2-Approximation für das Steinerbaum-Problem dar und ist in Zeit $O(n^3)$ berechenbar. Die beste bekannte in polynomieller Zeit berechenbare Approximation ist eine 1.55-Approximation (vgl. [KV06], Abschnitt 20.2).

[15] Dadurch, dass man ggf. Kanten streicht im in G zurückübertragenen Graphen.

Eine Verallgemeinerung des Steinerbaum-Problems, die auch insbesondere in den Approximationsalgorithmen für C_∞ in Abschnitt 3.3 verwendet wird, ist das folgende Problem:

Definition 2.2.7 (Steinerwald-Problem) Eine Instanz des Steinerwald-Problems besteht aus einem ungerichteten Graphen $G = (V, E)$ mit Kantengewichten $c : E \to \mathbb{R}_+$ und disjunkten Teilmengen $S_1, \ldots, S_k \subseteq V$. Das *Steinerwald-Problem* (auch unter Generalized-Steiner-Tree-Problem bekannt) besteht darin, einen Teilgraphen von G minimaler Kosten zu finden, in dem für jedes S_i alle Knoten aus S_i miteinander verbunden sind. □

Das Steinerwald-Problem ist 2-approximierbar (s. z. B. Goemans und Williamson [GW96], Abschnitt 4.6.1) über einen Primal-Dual-Algorithmus für eine Verallgemeinerung des Problems. Eine speziell auf das Steinerwald-Problem zugeschnittene Variante dieses Algorithmus findet sich etwa in [Vaz01], Kap. 22.

Die Spezialisierung des Steinerwald-Problems auf Instanzen mit $|S_i| = 2$ (für alle i) heißt auch *Point-to-Point-Connection-Problem*.

3 Die Ergebnisse für die Transportdomäne

3.1 Untere Schranken

Als erstes werden wir untere Schranken für die Approximierbarkeit von Transportproblemen untersuchen.

3.1.1 Allgemeine untere Schranken

Ein FPTAS kann es dabei für keines der Probleme geben: Man kann sich nämlich bei der Menge der zulässigen Lösungen auf solche beschränken, deren Wert durch ein festes Polynom in der Eingabelänge beschränkt ist (z. B. durch (2|P||V|+1)(|V||M|+1), vgl. [Hel03], Thm. 11 und Thm. 13). Damit handelt es sich bei jedem der Teilprobleme um ein sogenanntes „Problem mit kleinen Lösungswerten“ und es existiert für keines der Probleme ein FPTAS (vgl. Wegener [Weg03], Theorem 8.3.3.).

Lässt man für die Karten der Fahrzeuge beliebige Graphen zu, so sind sogar alle Probleme APX-hart, besitzen also insbesondere kein PTAS:

Definition 3.1.1 (1,2)-TSP ist das Optimierungsproblem, das durch Einschränkung des TSP auf vollständige Graphen mit Kantengewichten aus $\{1,2\}$ entsteht. □

Satz 3.1.2 *(1,2)-TSP ist APX-hart.*

Beweis In [PY93] wird gezeigt, dass es MAXSNP-hart ist. Daraus folgt die APX-Härte (vgl. [ACG+99]). ■

Satz 3.1.3 *C_1M_1 und $C_\infty M_1$ sind APX-hart.*

Beweis Wir geben eine AP-Reduktion von (1,2)-TSP an.

Sei eine Instanz des (1,2)-TSP gegeben durch $G = (V, E)$ und $w : E \to \{1, 2\}$. Diese werde auf folgende Transport-Instanz $(V', E', M, P, P_G, loc_0, loc_G, cap, road)$ abgebildet, wobei wir $v_0 \in V$ beliebig, aber fest wählen:

$$\begin{aligned}
&V' = \\
&\quad V \times \{start, ziel\} \cup V_{edges} \cup V_{0,edges} \cup \{v_0', v_{0,1}, v_{0,2}\} \\
&\quad \text{mit } V_{edges} = \{\, v_e \,|\, e \in E \text{ und } w(e) = 2 \,\} \\
&\quad \text{und } V_{0,edges} = \{\, v_{0,e} \,|\, e \in E \text{ und } w(e) = 2 \text{ und } v_0 \in e \,\} \\
&E' = \\
&\quad \{\, \{(v, start), (v, ziel)\} \,|\, v \in V \backslash \{v_0\} \,\}
\end{aligned}$$

$$
\begin{aligned}
&\quad \cup \{\, \{(v_0, start), v_{0,1}\}, \{v_{0,1}, v_{0,2}\}, \{v_{0,2}, (v_0, ziel)\} \,\} \\
&\quad \cup \{\, \{(v, start), (w, start)\} \mid \{v, w\} \in E \text{ und } w(\{v, w\}) = 1 \,\} \\
&\quad \cup \{\, \{(v, start), v_e\} \mid v \in e \text{ und } v_e \in V_{edges} \,\} \\
&\quad \cup \{\, \{(v, start), v_{0,e}\} \mid v \in e \text{ und } v_{0,e} \in V_{0,edges} \,\} \\
&\quad \cup \{\, \{v_0', (w, start)\} \mid \{v_0, w\} \in E \text{ und } w(\{v_0, w\}) = 1 \,\} \\
&\quad \cup \{\, \{v_0', v_{0,e}\} \mid v_{0,e} \in V_{0,edges} \,\} \\
&M = \{m\} \\
&P = P_G = V \\
&\forall p \in P : loc_0(p) = (p, start) \\
&loc_0(m) = v_0' \\
&\forall p \in P : loc_G(p) = (p, ziel) \\
&cap(m) = 1 \text{ oder } cap(m) = |V|; \\
&\quad \text{je nachdem, ob } C_1 M_1 \text{ oder } C_\infty M_1 \text{ Ziel der Reduktion ist.}
\end{aligned}
$$

Der Graph des Planungsproblems entspricht so im Wesentlichen dem des TSPs, wobei die Kanten der Länge 2 durch Pfade der Länge 2 ersetzt werden und pro Originalknoten ein Gegenstand eingeführt wird, dessen Zielknoten nur mit dessen Startknoten verbunden ist.

Würde man beim TSP einen Pfad kürzester Länge suchen statt eines Kreises, so wäre man schon fertig. Leider muss aber wegen der gewünschten AP-Reduktion insbesondere ein optimaler Plan (der einem Pfad im Graphen entspricht) auch wieder auf eine optimale (Rund-)Tour zurückabgebildet werden.

Daher wird ein Knoten (v_0) gewählt und verdoppelt, eine Version wird wie üblich Startknoten eines Gegenstands und ist im Gegensatz zu den anderen Gegenstandsstartknoten durch einen Pfad der Länge 3 mit dessen Zielknoten verbunden statt durch eine einzelne Kante. Die Version ohne Gegenstand (v_0') wird als Startknoten des Fahrzeugs gewählt.

Eine Beispieltransformation ist in Abb. 3.1 angegeben.

Lösungen für das Planungsproblem werden nun wie folgt auf Lösungen für das (1,2)-TSP zurückabgebildet:

Wir können annehmen, dass im Plan jeder Gegenstand sofort an sein Ziel befördert wird, nachdem er aufgenommen wurde (daher ist die Kapazität des Fahrzeugs egal), dass jeder Knoten der Form $(v, start)$ außer dem letzten genau zweimal besucht wird (wird er besucht, wenn das Fahrzeug weder gerade den Gegenstand aufnimmt noch vom Zielort des Gegenstands zurückkehrt, so kann man ihn durch eine (Original-)Kante der Länge höchstens 2 umgehen) und dass der Plan mit einer Drop-Aktion an einem Ort der Form $(w, ziel)$ endet (sonst streiche man die Aktionen nach der letzten solchen Aktion, der gekürzte Plan bleibt offensichtlich zulässig).

Genauer können wir sogar annehmen, dass er am Ort $(v_0, ziel)$ endet: Sonst bauen wir den Plan wie folgt um, ohne seine Länge zu vergrößern (wir beschreiben den Plan hier durch die Folge der besuchten Knoten, die Lage der Drop- und Pickup-Aktionen ist nach der Vorbemerkung klar): Ersetze die Teilfolge $(v, start) \dots (v_0, start) \dots (v_0, ziel) \dots (v_0, start) \dots (w, start)$ (wobei $(v, start)$ und $(w, start)$ die im Plan direkt zu $(v_0, start)$ benachbarten Startknoten sein sollen) durch

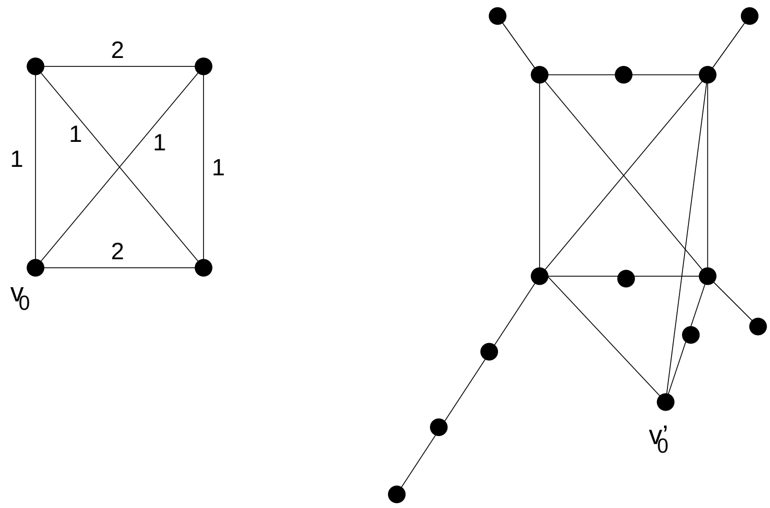

Abbildung 3.1: Eine Beispieltransformation von einer (1,2)-TSP-Instanz (links) auf eine Planinstanz (rechts).

$(v, start) \dots (w, start)$[1] (das spart mindestens 6 Schritte ein) und hänge an das Ende des Plans (es sei $(u, ziel)$) noch an: $(u, start) \dots (v_0, start) \dots (v_0, ziel)$ (das sind höchstens 6 Schritte).

Jeder solche Plan enthält also insbesondere genau eine Teilfolge, die zu einem Pfad von v_0 zu v_0 in der TSP-Instanz korrespondiert, der alle Knoten genau einmal besucht. Dies sei die Lösung für das TSP.

Ein optimaler Plan hat offensichtlich die Länge $(4n+1)+k^*$, wobei k^* der Wert einer optimalen Lösung für das TSP ist und $n = |V|$ (pro Gegenstand je einmal Drop und Pickup und zwei Bewegungsschritte, für den v_0-Gegenstand drei Bewegungen). Habe der Plan eine Approximationsgüte von $\frac{4n+1+k}{4n+1+k^*} = 1 + \frac{k-k^*}{4n+1+k^*}$, so hat die TSP-Tour eine Approximationsgüte von

$$\begin{aligned}
\frac{k}{k^*} = 1 + \frac{k-k^*}{k^*} &= 1 + \frac{\frac{k-k^*}{k^*}(4n+1+k^*)}{4n+1+k^*} \\
&= 1 + \frac{(k-k^*) + (\frac{4n+1}{k^*}(k-k^*))}{4n+1+k^*} \\
&\leq 1 + 6\frac{k-k^*}{4n+1+k^*} \quad \text{(da } 1 \leq k^* \text{ und } n \leq k^*\text{).}
\end{aligned}$$

■

Dieser Satz liefert leider noch keine gute untere Schranke für einen konstanten Faktor für die Approximierbarkeit, da bezüglich des (1,2)-TSP nur bekannt ist, dass es nicht besser als 741/740 -approximierbar ist (Engebretsen und Karpinski [EK04]), sodass wir

[1] Die Pünktchen stehen hier stets für die kürzestmögliche Verbindung.

hiermit strenggenommen nur wissen, dass C_1M_1 und $C_\infty M_1$ nicht besser als 4441/4440-approximierbar sind. Da wir aber für konstantes $c \in \mathbb{N}$ in der obigen Reduktion eine Kante der Länge 1 (bzw. 2) auch auf einen Pfad der Länge c (bzw. $2c$) abbilden könnten und sich so das α in der obigen Reduktion von 6 auf $1 + (\frac{5}{c})$ verringern würde, folgt tatsächlich doch eine untere Schranke von immerhin 741/740.

Da alle anderen hier betrachteten Transportklassen entweder C_1M_1 oder $C_\infty M_1$ als Spezialfälle enthalten, gilt:

Korollar 3.1.4 *Alle hier betrachteten Transportprobleme sind APX-hart.* □

3.1.2 Untere Schranken für M_*-Domänen

Um untere Schranken für die Approximierbarkeit von C_1M_*, $C_\infty M_*$ und C_*M_* zu erhalten, erweist sich das Set-Cover-Problem (vgl. Def. 2.1.11) als hilfreich. Wir werden hier drei sich ähnelnde, aber komplizierter werdende Reduktionen des Set-Cover-Problems auf Transportprobleme angeben.

Satz 3.1.5 *SetCover* $\leq_{AP} C_1M_*$ *und SetCover* $\leq_{AP} C_\infty M_*$.

Beweis Sei $(U, \{S_1, \dots, S_k\})$ eine SetCover-Instanz. Die Idee ist nun, pro Element der Grundmenge U einen Gegenstand einzuführen und pro gegebener Teilmenge S_i von U ein Fahrzeug eines eigenen Typs mit eigener Karte, mithilfe derer das Fahrzeug genau die Gegenstände erreichen kann, die in der entsprechenden Teilmenge sind. Start- und Zielorte der Gegenstände werden wiederum (ähnlich wie in Satz 3.1.2) so nah beieinander sein, dass sie mit einem einzelnen Bewegungsschritt zugestellt werden können, sodass die Kapazität der Fahrzeuge irrelevant sein wird. Damit das Wählen eines Fahrzeugs zum Zustellen von Gegenständen der kostenbestimmende Faktor ist, hat jedes Fahrzeug eine gewisse Strecke zurückzulegen, bis es zu den Gegenständen kommt.

Genauer seien wieder die Komponenten der Transportinstanz gegeben durch:

$V = U \times \{start, ziel\} \cup \bigcup_{1 \leq m \leq k} V_m$
mit $V_m = \{m\} \times \{0, \dots, |U|\}$

$E = \bigcup_{1 \leq m \leq k} E_m \cup \{\{(p, start), (p, ziel)\} \mid p \in U\}$
mit $E_m = \{\{(m, i), (m, i+1)\} \mid i = 0, \dots, |U| - 1\} \cup \{\{(m, |U|), (p, start)\} \mid p \in S_m\}$

$M = \{1, \dots, k\}$

$P = P_G = U$

$\forall m \in M : loc_0(m) = (m, 0)$

$\forall p \in P : loc_0(p) = (p, start)$

$\forall p \in P : loc_G(p) = (p, ziel)$

$\forall m \in M : cap(m) = 1$ oder $cap(m) = |U|$;
je nachdem, ob C_1M_* oder $C_\infty M_*$ Ziel der Reduktion ist.

$\forall m \in M : road(m) = E_m \cup \{\{(p, start), (p, ziel)\} \mid p \in S_m\}$

Als Beispiel ist in Abb. 3.2 eine Karte eines Fahrzeugs angegeben, die zu einer Teilmenge der Mächtigkeit 4 einer Menge der Mächtigkeit 7 korrespondiert.

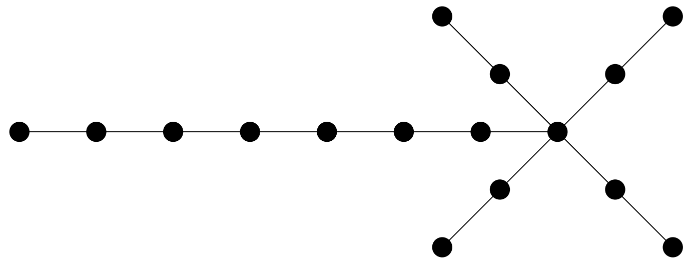

Abbildung 3.2: Beispielkarte für ein Fahrzeug

Die Gesamtgröße der übersetzten Instanz ist also $O(k|U|)$.

Die Rückübersetzung einer Lösung geht nun so:

Wir können davon ausgehen, dass in einem Plan jeder Gegenstand von seinem Startort aus direkt zu seinem Ziel gebracht wird und nur Fahrzeuge benutzt werden, die auch irgendwann einmal einen Gegenstand transportieren.

Von einem Punkt der Form $(m, |U|)$ aus betrachtet, kostet das Zustellen eines Gegenstands genau 6 Schritte (2 zum Aufnehmen und Ablegen des Gegenstands, 4 Bewegungsschritte). Also werden insgesamt $6|U|$ Schritte zum Transportieren der Gegenstände benötigt. Außerdem entstehen für jedes benutzte Fahrzeug (Anfahr-)Kosten von $|U|$. Somit sind die Kosten eines Plans o.B.d.A. genau $(6+j)|U|$, wenn j die Anzahl der benutzten Fahrzeuge bezeichnet.[2]

Man wähle nun in der Set-Cover-Instanz genau die Mengen, die zu den benutzten Fahrzeugen korrespondieren.

Offensichtlich ist die Korrespondenz zwischen Mengen und Fahrzeugen in beiden Richtungen gegeben und es gilt: Es gibt genau dann eine Mengenüberdeckung der Mächtigkeit $\leq j$, wenn es einen Plan der Länge $\leq (6+j)|U|$ gibt.
Sei die Approximationsgüte eines Plans $\frac{(6+j)|U|}{(6+j^*)|U|} = 1 + \frac{j-j^*}{6+j^*}$. Dann ist die Güte der zurückübersetzten Set-Cover-Lösung

$$\frac{j}{j^*} = 1 + \frac{j-j^*}{j^*} = 1 + \frac{6\frac{j-j^*}{j^*} + j - j^*}{6+j^*} \leq 1 + 7\frac{j-j^*}{6+j^*}.$$

■

Korollar 3.1.6 *Unter der Annahme, dass $NP \not\subseteq DTIME(n^{\log\log n})$, gibt es ein $c > 0$, sodass C_1M_*, $C_\infty M_*$ und C_*M_* nicht $(c-\varepsilon)\ln|P|$-approximierbar sind ($\forall\varepsilon > 0$), auch wenn Start- und Endpunkt jedes Gegenstands benachbart sind in jeder Karte, die einen dieser Punkte enthält.*

Beweis Bei Feige [Fei98] wird gezeigt, dass unter dieser Annahme SetCover nicht $(1-\varepsilon)\ln|U|$-approximierbar ist ($\forall\varepsilon > 0$). Also sind dann mit der gesehenen Reduktion aus

[2] Genauer gesagt müssten wir eigentlich den Anfahrtsweg jedes Fahrzeugs um 2 verlängern, um dies Resultat zu erhalten, da ein Fahrzeug m nach dem Zustellen seines letzten Gegenstands nicht mehr zu $(m, |U|)$ zurückkehren muss.

Satz 3.1.5 C_1M_* und $C_\infty M_*$ nicht $(\frac{1}{7}-\varepsilon)\ln|P|$-approximierbar ($\forall\varepsilon>0$). Wenn wir den Anlauf pro Fahrzeug auf $c|U|$ statt $|U|$ verlängern für ein $c\in\mathbb{N}$, so sinkt der Faktor α von 7 auf $1+\frac{6}{c}$, sodass wir auch hier sogar davon sprechen könnten, dass die untersuchten Probleme nicht $(1-\varepsilon)\ln|P|$-approximierbar sind. ∎

Um eine ähnliche untere Schranke für $C_\infty M_*$ und C_*M_* auch bezüglich der Anzahl der Fahrzeuge zu bekommen, folgt gleich die nächste Reduktion (die Aussage des Satzes ist dabei weniger interessant als sein weiterverwendbarer Beweis).

Satz 3.1.7 *SetCover* $\leq_{AP} C_\infty M_*$.

Beweis Sei $(U,\{S_1,\ldots,S_k\})$ eine SetCover-Instanz.

Die in Abb. 3.3 auf S. 30 illustrierte Idee ist jetzt, wieder pro Element des Universums U einen Gegenstand einzuführen, aber nun pro Element ebenfalls ein Fahrzeug. Außerdem hat man ein Extrafahrzeug, das die Gegenstände von ihrem gemeinsamen Startort aus zu jeweils einem einer Teilmenge S_i und einem Gegenstand dieser Teilmenge zugeordneten Übergabepunkt bringen kann. Von jedem seiner Übergabepunkte aus kann jeder Gegenstand in konstanter Zeit durch das ihm zugeordnete Fahrzeug zu seinem Ziel gebracht werden. Das Extrafahrzeug kann dabei für jede Teilmenge in insgesamt $O(|U|)$ Schritten beliebig viele Gegenstände, die zu Elementen dieser Teilmenge korrespondieren, zu ihren Übergabepunkten bringen. Die Übergabepunkte eines Gegenstands sind für das Extrafahrzeug nur über den Startort verbunden, die Teilmengen so für dies Fahrzeug voneinander getrennt. Außerdem gibt es pro forma eine sternförmige Verbindung der Zielorte der Gegenstände untereinander, die einerseits dafür sorgt, dass es insgesamt nur 2 verschiedene zusammenhängende Karten gibt, die andererseits aber so lang ist, dass es sich nie lohnt, sie zu benutzen.

Nun die formale Definition (l wird unten noch genauer festgelegt):

$$
\begin{aligned}
&V := U\times\{\mathit{vertex}\}\cup\textstyle\bigcup_{1\leq i\leq k}V_i\cup\{\mathit{start}\}\cup U\times\{1,\ldots,l\}\cup\{\mathit{dummy}\}\\
&\text{mit } V_i := \{i\}\times\{1,\ldots,|U|\}\cup\{i\}\times S_i\\
&E := \{\,\{\mathit{start},(i,1)\}\mid 1\leq i\leq k\,\}\\
&\quad\cup\{\,\{(i,j),(i,j+1)\}\mid 1\leq i\leq k, 1\leq j\leq |U|-1\,\}\\
&\quad\cup\{\,\{(i,|U|),(i,p)\}\mid 1\leq i\leq k, p\in S_i\,\}\\
&\quad\cup E_{port}\cup E_{rest}\\
&\quad\text{mit } E_{port} := \{\,\{(i,p),(p,\mathit{vertex})\}\mid p\in U,(i,p)\in V\,\}\\
&\quad\text{und } E_{rest} := \{\,\{(p,\mathit{vertex}),(p,1)\}\mid p\in U\,\}\\
&\qquad\cup\{\,\{(p,j),(p,j+1)\}\mid p\in U, 1\leq j\leq l-1\,\}\\
&\qquad\cup\{\,\{(p,l),\mathit{dummy}\}\mid u\in U\,\}\\
&M := \{m_0\}\cup U\times\{\mathit{vehicle}\}\\
&P := P_G = U\times\{\mathit{portable}\}\\
&\mathit{loc}_0(m_0) := \mathit{start}\\
&\mathit{loc}_0((p,\mathit{vehicle})) := (p,\mathit{vertex})\ \ (\forall p\in U)\\
&\mathit{loc}_0((p,\mathit{portable})) := \mathit{start}\ (\forall p\in U)\\
&\mathit{loc}_G((p,\mathit{portable})) := (p,\mathit{vertex})\ \ (\forall p\in U)
\end{aligned}
$$

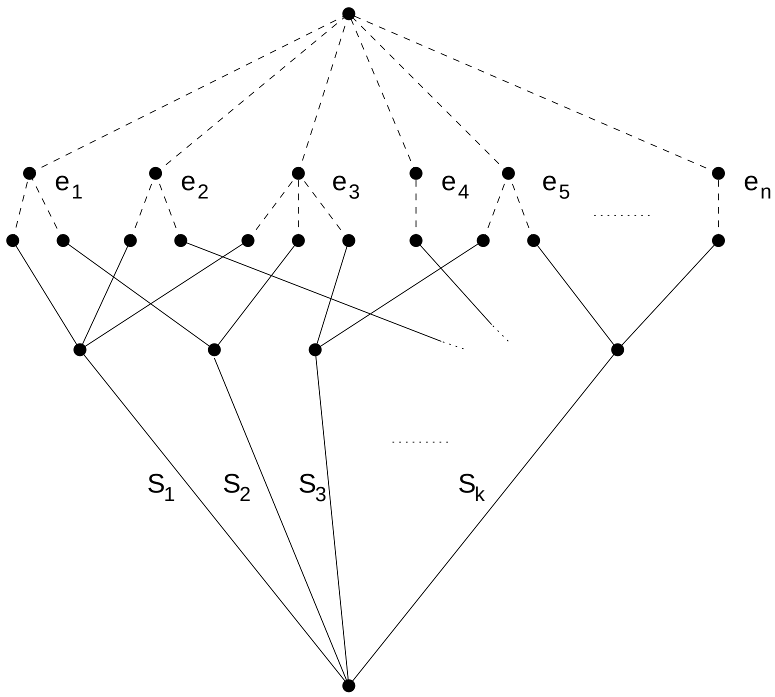

Abbildung 3.3: Eine Beispielkarte einer mittels der Reduktion aus Satz 3.1.7 entstandenen Planungsinstanz. Die Beschriftungen beziehen sich auf die verwendeten Teile der Set-Cover-Instanz.

$$\begin{aligned}
&cap(m) := |U| \ (\forall m \in M)\\
&road(m_0) := E \setminus (E_{port} \cup E_{rest})\\
&road((p, vehicle)) := E_{port} \cup E_{rest}
\end{aligned}$$

Wie sieht nun eine Lösung für das entstehende Transportproblem aus?
Wählt man für die entsprechende Set-Cover-Instanz j Teilmengen aus, die die Grundmenge überdecken, und lässt das Extrafahrzeug die Gegenstände über die zu diesen Mengen korrespondierenden Wege zu ihren Übergabepunkten bringen, um sie danach in jeweils konstanter Zeit zu ihrem Ziel bringen zu lassen, so hat das Extrafahrzeug Kosten von $(2j-1)|U|+4|U|-1$ und jedes der anderen Fahrzeuge Kosten[3] von 4. Die Gesamtkosten für eine solche Lösung betragen also $(2j+3)|U|-1+4|U| = (2j+7)|U|-1$. Wir können nun annehmen, dass $j \leq |U|$. So eine „gute“ Lösung lässt sich nun so zurückübersetzen, dass gerade die Teilmengen für die Set-Cover-Instanz gewählt werden, für die m_0 den Weg von *start* zu $(i, |U|)$ zurückgelegt hat. Offensichtlich ergibt dies eine zulässige Lösung für die Set-Cover-Instanz.

Ist die Güte des Plans $\frac{(2j+7)|U|-1}{(2j^*+7)|U|-1} = 1 + \frac{2(j-j^*)|U|}{(2j^*+7)|U|-1}$, so ist die Güte der Set-Cover-

[3] Hinfahren, Aufnehmen, Zurückfahren, Ablegen.

Lösung

$$\begin{aligned}
\frac{j}{j^*} &= 1+\frac{j-j^*}{j^*} = 1+\frac{\frac{j-j^*}{j^*}((2j^*+7)|U|-1)}{(2j^*+7)|U|-1} \\
&= 1+\frac{2(j-j^*)|U|+\frac{7}{j^*}(j-j^*)|U|-\frac{j-j^*}{j}}{(2j^*+7)|U|-1} \\
&\leq 1+\frac{2(j-j^*)|U|+7(j-j^*)|U|}{(2j^*+7)|U|-1} \leq 1+4.5\frac{2(j-j^*)|U|}{(2j^*+7)|U|-1}.
\end{aligned}$$

Wir definieren nun $l := (|U|+4)|U|$. Wird stattdessen in einem „schlechten“ Plan ein Gegenstand über die Verbindung der Elementfahrzeuge untereinander transportiert, so erzeugt dies mindestens Kosten von $2l = (2|U|+8)|U|$. Eine solche Lösung werde auf eine triviale Lösung der Set-Cover-Instanz zurückabgebildet, für die man pro Element eine beliebige Teilmenge wählt, die dieses Element enthält, und die Menge dieser Mengen als Lösung der Set-Cover-Instanz wählt. Die Güteabschätzung gilt so weiterhin. Wir können immer annehmen, dass für die Pläne einer dieser Fälle eintritt. ∎

Korollar 3.1.8 *Unter der Annahme, dass $NP \not\subseteq DTIME(n^{\log\log n})$, gibt es ein $c > 0$ und ein $c' > 0$, sodass $C_\infty M_*$ und $C_* M_*$ nicht $(c-\varepsilon)\ln|P|$-approximierbar sind ($\forall\varepsilon > 0$) und nicht $(c'-\varepsilon)\ln|M|$-approximierbar ($\forall\varepsilon > 0$), auch wenn die Startorte aller Gegenstände gleich sind und es nur 2 verschiedene Fahrzeugtypen (d.h. Fahrzeugkarten) gibt und die Karte jedes Fahrzeugs zusammenhängend ist.*

Beweis Die im Beweis von Satz 3.1.7 angegebene Reduktion erzeugt eine Transportinstanz, in der alle Gegenstände denselben Startort haben und in der es nur 2 verschiedene Karten gibt. Außerdem ist die Anzahl der Fahrzeuge nur um 1 größer als die Anzahl der Elemente der Grundmenge in der Set-Cover-Instanz, die auf die Transportinstanz abgebildet wird, und die Anzahl der Gegenstände ist gerade gleich der Anzahl der Elemente der Grundmenge. Die Behauptung folgt nun analog zu Korollar 3.1.6. ∎

Die eben angegebene Reduktion können wir noch etwas verfeinern, um fast analoge Ergebnisse auch für $C_1 M_*$ zu erhalten:

Satz 3.1.9 *$SetCover \leq_{AP} C_1 M_*$.*

Beweis Sei $(U, \{S_1, \dots, S_k\})$ eine SetCover-Instanz.

Wir führen nun pro Gegenstand 2 Fahrzeuge ein, die im Wesentlichen getrennt voneinander sind und von denen eines für den Start und eines für das Ziel des Gegenstands zuständig ist. Das immer noch vorhandene Extrafahrzeug hat nun nur noch die Aufgabe, die Gegenstände von ihren Startfahrzeugen zu ihren Zielfahrzeugen zu bringen. Die Kosten und nötigen Fahrten des Extrafahrzeugs bleiben dabei im Prinzip so wie in der Reduktion aus Satz 3.1.7. Start- und Zielfahrzeuge haben analog zur vorigen Reduktion eine gemeinsame Straßenkarte, die aber diese Fahrzeuge nur pro forma verbindet. Die eigentlichen Kosten dieser Fahrzeuge sind immer konstant. Eine Illustration findet sich in Abb. 3.4 auf S. 32. Nun noch formal (l wird wieder unten noch genauer festgelegt):

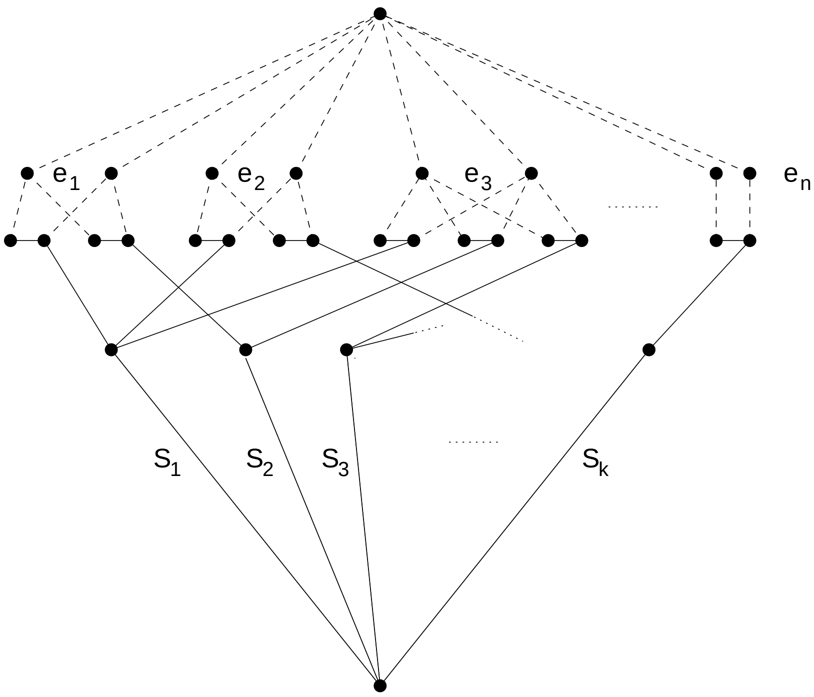

Abbildung 3.4: Eine Beispielkarte einer mittels der Reduktion aus Satz 3.1.9 entstandenen Planungsinstanz. Die Beschriftungen beziehen sich auf die verwendeten Teile der Set-Cover-Instanz.

$$
\begin{aligned}
&V := U \times \{startvertex, goalvertex\} \cup \bigcup\nolimits_{1\le i\le k} V_i \cup \{start\}\\
&\quad \cup U \times \{startvertex, goalvertex\} \times \{1,\dots,l\} \cup \{dummy\}\\
&\text{mit } V_i := \{i\} \times \{1,\dots,|U|\} \cup \{i\} \times S_i \times \{1,2\}\\
&E := \{\, \{start, (i,1)\} \mid 1 \le i \le k \,\}\\
&\quad \cup \{\, \{(i,j),(i,j+1)\} \mid 1 \le i \le k, 1 \le j \le |U|-1 \,\}\\
&\quad \cup \{\, \{(i,|U|),(i,p,1)\} \mid 1 \le i \le k, p \in S_i \,\}\\
&\quad \cup \{\, \{(i,p,1),(i,p,2)\} \mid p \in S_i \,\}\\
&\quad \cup E_{port} \cup E_{rest}\\
&\text{mit } E_{port} := \{\, \{(i,p,1),(p,startvertex)\} \mid p \in U, (i,p,1) \in V \,\}\\
&\quad \cup \{\, \{(i,p,2),(p,goalvertex)\} \mid p \in U, (i,p,2) \in V \,\}\\
&\text{und } E_{rest} := \{\, \{(p,startvertex),(p,startvertex,1)\} \mid p \in U \,\}\\
&\quad \cup \{\, \{(p,startvertex,j),(p,startvertex,j+1)\} \mid p \in U, 1 \le j \le l-1 \,\}\\
&\quad \cup \{\, \{(p,startvertex,l), dummy\} \mid u \in U \,\}\\
&\quad \cup \{\, \{(p,goalvertex),(p,goalvertex,1)\} \mid p \in U \,\}\\
&\quad \cup \{\, \{(p,goalvertex,j),(p,goalvertex,j+1)\} \mid p \in U, 1 \le j \le l-1 \,\}\\
&\quad \cup \{\, \{(p,goalvertex,l), dummy\} \mid u \in U \,\}\\
&M := \{m_0\} \cup U \times \{startvehicle, goalvehicle\}\\
&P := P_G = U \times \{portable\}
\end{aligned}
$$

$loc_0(m_0) := start$
$loc_0((p, startvehicle)) := (p, startvertex) \ \ (\forall p \in U)$
$loc_0((p, goalvehicle)) := (p, goalvertex) \ \ (\forall p \in U)$
$loc_0((p, portable)) := (p, startvertex) \ \ (\forall p \in U)$
$loc_G((p, portable)) := (p, goalvertex) \ \ (\forall p \in U)$
$cap(m) := 1 \ \ (\forall m \in M)$
$road(m_0) := E \setminus (E_{port} \cup E_{rest})$
$road((p, startvehicle)) := E_{port} \cup E_{rest}$
$road((p, goalvehicle)) := E_{port} \cup E_{rest}$

Die Kosten einer analog zur vorigen Reduktion „guten“ Lösung sind jetzt $2j|U|+13|U|-|U|-2 = 2((j+6)|U|-1)$ (pro Gegenstand: für das Startfahrzeug Kosten von 3, für das Extrafahrzeug Kosten von 6, für das Zielfahrzeug Kosten von 4). Die Rückabbildung funktioniert ebenfalls analog.

Ist die Approximationsgüte eines solchen Plans $\frac{2((j+6)|U|-1)}{2((j^*+6)|U|-1)} = 1 + \frac{(j-j^*)|U|}{(j^*+6)|U|-1}$, so ist die Güte der Set-Cover-Lösung

$$\begin{aligned} \frac{j}{j^*} = 1 + \frac{j-j^*}{j^*} &= 1 + \frac{\frac{j-j^*}{j^*}((j^*+6)|U|-1)}{(j^*+6)|U|-1} \\ &= 1 + \frac{(j-j^*)|U| + \frac{6}{j^*}(j-j^*)|U| - \frac{j-j^*}{j}}{(j^*+6)|U|-1} \\ &\leq 1 + \frac{(j-j^*)|U| + 6(j-j^*)|U|}{(j^*+6)|U|-1} = 1 + 7\frac{(j-j^*)|U|}{(j^*+6)|U|-1}. \end{aligned}$$

Wir setzen nun $l := (|U|+6)|U|$. „Schlechte“ Lösungen können wir dann analog zur vorigen Reduktion auf triviale Lösungen für Set-Cover abbilden. ∎

Korollar 3.1.10 *Unter der Annahme, dass $NP \not\subseteq DTIME(n^{\log\log n})$, gibt es ein $c > 0$ und ein $c' > 0$, sodass C_1M_* nicht $(c-\varepsilon)\ln|P|$-approximierbar ist ($\forall \varepsilon > 0$) und nicht $(c'-\varepsilon)\ln|M|$-approximierbar ($\forall \varepsilon > 0$), auch wenn es nur 2 verschiedene Fahrzeugtypen (d.h. Fahrzeugkarten) gibt und die Karte jedes Fahrzeugs zusammenhängend ist.*

Beweis Die im Beweis von Satz 3.1.9 angegebene Reduktion erzeugt eine Transportinstanz, in der es nur 2 verschiedene Karten gibt. Außerdem ist die Anzahl der Fahrzeuge nur um 1 größer als die zweifache Anzahl der Elemente der Grundmenge in der Set-Cover-Instanz, die auf die Transportinstanz abgebildet wird, (also $|M| = 2|U|+1$) und die Anzahl der Gegenstände ist gerade gleich der Anzahl der Elemente der Grundmenge. Die Behauptung folgt nun analog zu Korollar 3.1.8. ∎

Damit haben wir insbesondere gesehen, dass C_1M_*, $C_\infty M_*$ und C_*M_* nicht $(c-\varepsilon)\ln|P|$- und nicht $(c'-\varepsilon)\ln|M|$-approximierbar sind für feste $c, c' \in \mathbb{Q}$ und alle $\varepsilon > 0$.[4]

[4] Unter der Voraussetzung $NP \not\subseteq DTIME(n^{\log\log n})$.

3.2 Präemptiv vs. nichtpräemptiv

Bevor wir uns mit Approximationsalgorithmen beschäftigen, wollen wir uns vorbereitend Gedanken machen über das Verhältnis zwischen den optimalen Kosten eines nichtpräemptiven Plans für eine Instanz und den optimalen Kosten eines präemptiven Plans. In diesem Abschnitt bezeichnen wir diese Kosten mit opt_{np} (für „nichtpräemptiv“) und opt_p (für „präemptiv“).

Sinn macht ein solcher allgemeiner Vergleich nur für die Probleme, in denen alle Fahrzeuge dieselben Straßen befahren können. Sonst kann man natürlich einfach Instanzen angeben, für die es zwar präemptive Lösungen gibt, aber keine nichtpräemtiven. Auch wenn sowohl präemptive als auch nichtpräemptive Lösungen existieren, kann das hier betrachtete Verhältnis dann beliebig groß werden.[5]

Für die Transportprobleme C_1M_+ und $C_\infty M_+$ werden die Approximationsalgorithmen in Abschnitt 3.3 stets nichtpräemptive Lösungen erzeugen, die hier im Folgenden behandelten unteren und oberen Schranken sind dafür nicht wichtig. Interessanter ist da die untere Schranke für C_*M_1, die es wenigstens aussschließt, mithilfe nichtpräemptiver Pläne eine Konstant-Faktor-Approximation zu erreichen.

Zunächst die Definitionen von präemptiv und nichtpräemptiv, nachdem diese Begriffe in informeller Weise schon in Abschnitt 2.2.2 aufgetaucht sind:

Definition 3.2.1 Ein zulässiger Plan für ein Transportproblem heißt *nichtpräemptiv*, wenn darin jeder von einem Fahrzeug aufgenommene Gegenstand erst an dessen Zielort wieder abgelegt wird. Wird ein Gegenstand hingegen vor seinem Zielort abgelegt (und somit später wieder aufgenommen, von welchem Fahrzeug auch immer), so heißt der Plan *präemptiv*.

Analog bezeichnet man eine *Problemklasse* als *nichtpräemptiv*, wenn pro Gegenstand nur je ein Aufnehmen und Ablegen erlaubt ist, und als *präemptiv*, wenn pro Gegenstand beliebig häufiges Aufnehmen und Ablegen erlaubt sind.[6] □

3.2.1 C_1

Wir zeigen: Im Fall C_1M_+ ist das Verhältnis vom Wert einer optimalen nichtpräemptiven Lösung zu einer optimalen präemptiven Lösung durch 2 beschränkt. Dieses Verhältnis von 2 kann aber schon im Fall C_1M_1 auch asymptotisch angenommen werden. Sehr einfach kann man zunächst eine obere Schranke von 3 einsehen:

Lemma 3.2.2 *In C_1M_+ kann jeder präemptive Plan in einen nichtpräemptiven höchstens dreifacher Länge umgewandelt werden.*

Beweis Idee: Man lasse das jeweils erste Fahrzeug, das einen Gegenstand transportiert, diesen gleich zu seinem Zielort transportieren und danach wieder zurücklaufen. Die anderen Teilstücke des Plans, in denen der Gegenstand transportiert wird, ersetze man durch Leerfahrten. ■

[5] Zumindest bis $\Omega(|V|)$.

[6] Damit sind auch nichtpräemptive Pläne zulässige Lösungen für ein präemptives Problem. Diese Bezeichnungsweise ist üblich.

Zum Beweis der genaueren oberen Schranke zuerst ein Lemma:

Lemma 3.2.3 *Sei G ein beliebiger gerichteter Multigraph und sei P eine Menge von Pfaden als Folgen gerichteter Kanten über der Knotenmenge von G. Dann gibt es in dem Multigraphen $M = (V(G), E(G) \dot{\cup} E(P))$ von jedem Knoten aus eine Rundtour durch seine Zusammenhangskomponente, in der jede Kante einmal in der gegebenen Richtung und einmal in Rückrichtung durchlaufen wird und jeder Pfad in dieser Zusammenhangskomponente einmal am Stück in der gegebenen Richtung.*

Beweis Vorbemerkungen:

- Wir bezeichnen mit M^* den aus M dadurch entstehenden Graphen, dass wir für jede Kante (u, v) eine entsprechende Kante (v, u) hinzufügen. Da wir in M^* jede ursprüngliche Kante durch eine in Gegenrichtung ergänzt haben, sind alle schwachen Zusammenhangskomponenten auch stark zusammenhängend.

- Wenn es einen solchen Durchlauf durch einen (Teil-)Graphen von M gibt, wird außerdem jeder innere Knoten eines Pfades mindestens zweimal besucht, also auch einmal nicht während des „Durchlaufens des Pfades am Stück in der gegebenen Richtung“.

Der Beweis geschieht nun durch Induktion über die Mächtigkeit von P:

$|P| = 0$: Im Multigraphen M^* gilt für jeden Knoten v offensichtlich $\delta^+(v) = \delta^-(v)$ und so existiert durch jede Zusammenhangskomponente ein Eulerkreis. Dieser hat die gewünschten Eigenschaften (den Start dieses Kreises kann man dabei beliebig wählen).

Induktionsschritt:

Sei p ein beliebiger Pfad mit Startpunkt p_{start} und Endpunkt p_{ende}. Sei M' der aus M durch Entfernen der Kanten aus p entstehende Graph. Wir konstruieren zunächst eine bei p_{start} beginnende und endende Rundtour durch M: Man bewege sich entlang p zu p_{ende} und gehe dann diesen Weg wieder zurück. Während des Rückwegs durchläuft man die an p hängenden Zusammenhangskomponenten von M' mit den nach Induktionsvoraussetzung existierenden Rundtouren ausgehend vom jeweiligen Knoten des Pfads p (dabei beachtet man, dass man jede an p hängende Komponente nur einmal durchläuft).

Sei nun v ein beliebiger Knoten aus der Zusammenhangskomponente von p in M (befindet sich v in einer anderen Zusammenhangskomponente, existiert nach Induktionsvoraussetzung eine Tour durch seine Zusammenhangskomponente). Die eben konstruierte Tour durch die Komponente sei $p_{start} \dots p_{ende} \dots uvw \dots p_{start}$, wobei das hier angegebene Vorkommen von v gerade eines sei, das nicht Teil eines am Stück in gegebener Richtung durchlaufenen Pfades ist. Die gewünschte Tour ist nun $vw \dots p_{start} \dots p_{ende} \dots uv$. ■

Damit ist der Beweis der eigentlichen Behauptung nicht mehr schwierig:

Satz 3.2.4 *In C_1M_+ gilt $\frac{opt_{np}}{opt_p} \leq 2$.*

Beweis Sei ein (präemptiver) Plan gegeben. Wir wandeln ihn in einen nichtpräemptiven Plan höchstens doppelter Länge um:[7]

[7] Die Anzahl der Bewegungsaktionen verdoppelt sich. Die Anzahl der Pickup- und Drop-Aktionen nimmt natürlich nicht zu, sondern eher ab.

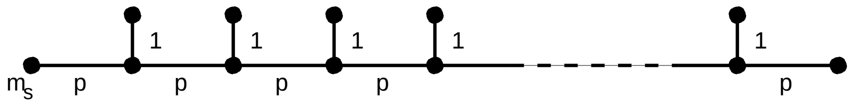

Abbildung 3.5: Das Beispiel zur unteren Schranke von 2 für C_1M_+.

Wir betrachten den gerichteten Multigraphen G, der aus den Kanten besteht, die von den Fahrzeugen durchlaufen werden. Pro Durchlauf einer Kante existiert eine Kopie der in Durchlaufrichtung gerichteten Kante. Wir beschriften jede Kante entweder mit $\emptyset$ (bei Durchlauf mit einem leeren Fahrzeug) oder mit dem gerade transportierten Gegenstand. Sei $l : E(G) \to \{\emptyset, p_1, \ldots, p_k\}$ die entsprechende Beschriftungsfunktion.

Sei $G_\emptyset = (V(G), E_\emptyset)$ der Teilgraph von G mit $E_\emptyset := \{\, e \in E(G) \,|\, l(e) = \emptyset \,\}$. Für jeden Gegenstand p ist $E_p := \{\, e \in E(G) \,|\, l(e) = p \,\}$ offensichtlich als Pfad darstellbar. Sei S die Menge dieser Transportpfade der Gegenstände aus G. Wir können nun Lemma 3.2.3 auf $G_\emptyset$ und S anwenden. Für einen beliebigen Fahrzeugstartpunkt v korrespondiert die so entstehende bei v startende und endende Rundtour durch G auf offensichtliche Weise zu einem nichtpräemptiven Plan (man nehme einen Gegenstand vor dem Durchlauf seines Pfades auf und lege ihn danach sofort ab). Da jede Kante genau zweimal durchlaufen wird, ist dieser höchstens doppelt so lang wie der ursprüngliche Plan. ■

Andererseits ist 2 aber asymptotisch auch die bestmögliche konstante obere Schranke.

Satz 3.2.5 *Es gibt eine Familie von (Grid-)Instanzen von C_1M_1 mit p Gegenständen ($p \in \mathbb{N}$) der Größe $O(p^2)$, sodass $\frac{opt_{np}}{opt_p} \to 2$ (für $p \to \infty$).*

Beweis Betrachten wir die folgende Verallgemeinerung der Instanz aus Abb. 3.5: Wir haben eine Strecke der Länge p^2, entlang der alle p Schritte ein Gegenstand liegt, der ein Feld zur Seite transportiert werden muss. Außerdem liegt am Beginn der Strecke ein Gegenstand, der an das Ende der Strecke gebracht werden muss. Der Startort des einzigen Fahrzeugs liegt am Beginn der Strecke.

Jeder nichtpräemptive Plan muss einmal den Gegenstand vom Startort des Fahrzeugs zum Ende der Strecke transportieren und kann sich währenddessen nicht um andere Gegenstände kümmern. Die Kosten insgesamt sind also wenigstens $(p^2 + 2) + (p - 1) * 4 - 1 + (p^2 - p) = 2p^2 + 3p - 3$ (für das Transportieren der Gegenstände und nötige Bewegungen, um die Startorte der anderen Gegenstände zu erreichen).

Hingegen sind die Kosten eines optimalen präemptiven Plans durch $p^2 + 6p$ beschränkt (man lege den von links nach rechts zu transportierenden Gegenstand zwischendurch immer wieder ab und nehme ihn beim Weitergehen wieder auf).
Nun gilt $\frac{2p^2+3p-3}{p^2+6p} = 2 - \frac{9p}{p^2+6p} - \frac{3}{p^2+6p} = 2 - \frac{9}{p+6} - \frac{3}{p^2+6p} \to 2$ (für $p \to \infty$). ■

3.2.2 C_∞

Ebenso ist im Fall $C_\infty M_+$ das Verhältnis vom Wert einer optimalen nichtpräemptiven Lösung zu einer optimalen präemptiven Lösung durch 2 beschränkt. Dieses ist zwar auch die beste konstante obere Schranke, in diesem Fall ist es aber interessant, auch die Anzahl der zur Verfügung stehenden Fahrzeuge noch miteinzubeziehen. Dann erhält man eine genaue obere Schranke von $2 - \frac{1}{|M|}$.

Im Fall $C_\infty M_1$ macht es für das Fahrzeug wegen der ausreichenden Kapazität natürlich keinen Sinn, einen Gegenstand vor dessen Zielort abzulegen.

Um die obere Schranke von $2 - \frac{1}{|M|}$ zu beweisen, erst einmal wieder einige Lemmata:

Lemma 3.2.6 *Sei G ein gerichteter azyklischer Graph. Dann gibt es ein $v \in V(G)$ mit $\delta^+(v) = 0$.*

Beweis Angenommen, das gilt nicht. Wir wählen einen beliebigen Knoten aus und laufen entlang der gerichteten Kanten in deren Richtung durch den Graphen, solange das möglich ist, ohne eine Kante ein zweites Mal zu durchlaufen. Wenn dies weniger als n-mal möglich ist, kann dies nur deshalb sein, weil sich ein Knoten wiederholt hat (wegen der Annahme). Ist es hingegen mindestens n-mal möglich, so hat man dabei wenigstens $n+1$ Knoten besucht, also wiederum einen doppelt. In beiden Fällen existiert ein gerichteter Kreis, der Graph ist also nicht azyklisch. Widerspruch. ■

Lemma 3.2.7 *Sei G ein schwach zusammenhängender gerichteter azyklischer Graph. Für alle $v \in V(G)$ gibt es einen Durchlauf von $U(G)$ (der ungerichteten Variante von G), der bei v beginnt und endet und der jede Kante in jeder Richtung höchstens einmal durchläuft sowie Kanten erst dann in der aus G gegebenen Richtung durchläuft, wenn schon alle ihre Vorgängerkanten bzgl. der von G induzierten partiellen Ordnung so durchlaufen wurden.*

Beweis Induktion über die Anzahl n der Knoten:

Fall n=1 oder n= 2: klar.

Induktionsschritt: Habe G $n+1$ Knoten, sei v ein beliebiger davon und v_0 ein Knoten mit $\delta^+(v_0) = 0$ (existiert nach Lemma 3.2.6). Wir betrachten die Zusammenhangskomponenten $G_1, \ldots, G_k$ von $G \setminus \{v_0\}$ und die direkten Vorgänger $v_{11}, \ldots, v_{1i_1}, \ldots, v_{k1}, \ldots, v_{ki_k}$ von v_0 in G mit $v_{ij} \in G_i$. O.B.d.A. sei dabei v in G_1.[8] Nach Induktionsvoraussetzung existieren Touren T_i durch G_i, die bei v_{i1} starten und enden. Sei nun S_1 eine Tour innerhalb G_1 von v nach v und $S_1 = S'_1 v_{11} S''_1$ mit $v_{11} \notin S''_1$. Sei auch o.B.d.A. $S'_1 = S'_{11} v_{12} S'_{12} v_{13} \ldots v_{1r} S'_{1r}$ und $S''_1 = S''_{1r} v_{1(r+1)} \ldots v_{1i_1} S''_{1i_1}$, wobei die hier gekennzeichneten Vorkommen von v_{1j} jeweils das letzte Vorkommen von v_{1j} innerhalb von S_1 bezeichnen sollen. Dann sind nach $S'_1 v_{11}$ alle Vorgänger von v_{11} durchlaufen und bei den gekennzeichneten Vorkommen von v_{1j} auch alle Vorgänger von v_{1j}. Folgende Tour erfüllt die gewünschten Eigenschaften: $S'_{11} v_{12} v_0 v_{12} S'_{12} v_{13} v_0 v_{13} \ldots v_{1r} v_0 v_{1r} S'_{1r} v_{11} v_0 T_2 v_0 v_{21} v_0 v_{22}$ $v_0 v_{23} \ldots v_{ki_k} v_0 v_{11} S''_{1r} v_{1(r+1)} v_0 v_{1(r+1)} \ldots v_{1i_1} v_0 v_{1i_1} S''_{1i_1}$. ■

Satz 3.2.8 *In $C_\infty M_+$ gilt $\frac{opt_{np}}{opt_p} \leq 2 - \frac{1}{|M|}$.*

[8] Für $v = v_0$ ist es noch einfacher.

Beweis Sei ein beliebiger zulässiger (präemptiver) Plan gegeben. Wir wandeln diesen in einen zulässigen nichtpräemptiven Plan höchstens doppelter Länge um.

Dazu repräsentieren wir zunächst den Plan als Folge $(v_i)_{i=0\ldots r}$ von Tupeln $v_i = (m_i, \bar{e}_i, P_i) \in M \times E^+ \times Pot(P)$ (wobei E^+ die Menge der nichtleeren Folgen von Kanten sei). Hierbei stehe ein Tupel v_i dafür, dass das Fahrzeug m_i entlang der Kanten $\bar{e}_i$ die Gegenstände P_i transportiert. Diese Folge sei minimal im folgenden Sinn: Wenn $\exists m, A\, \exists i < j : v_i = (m, \bar{e}, A), v_j = (m, \bar{e}', A)$, so $\exists k, A'$: $A' \neq A$ und $i < k < j$ und $v_k = (m, \bar{e}'', A')$ (wir können die Aktionen jedes Plans offensichtlich derart ordnen). Die nötigen Pickup-Aktionen werden implizit kurz vor dem Durchlaufen der Kanten ausgeführt, die Drop-Aktionen kurz nachher (welche Aktionen da auszuführen sind, ergibt sich daraus, wie das entsprechende Fahrzeug weiterfährt).

Nun betrachtet man zunächst einen gerichteten Graphen G, der diese Tupel als Knoten hat. Hierbei definieren wir die gerichteten Kanten als (v_i, v_j), wenn $i < j$ und entweder ($m_i = m_j$ und $\nexists k$: $i < k < j$ und $m_i = m_k$) oder ($P_i \cap P_j \neq \emptyset$ und $\nexists k$: $i < k < j$ und $P_i \cap P_j \cap P_k \neq \emptyset$).

Nun wandeln wir G in einen kantengewichteten gerichteten Graphen $D(G)$ mit Kantengewichtsfunktion w um. Jede seiner bisherigen Kanten erhalte Gewicht 0. Wir ersetzen außerdem jeden bisherigen Knoten durch zwei Knoten, von denen der eine gerade die eingehenden Kanten des bisherigen erhält, der andere die ausgehenden, und vom Eingangs- zum Ausgangsknoten eine Kante des Gewichts $|\bar{e}_i|$ existiert, wenn der Knoten zu einem Tupel mit dem Kantenzug $\bar{e}_i$ korrespondiert.

Auch wenn der zugrundeliegende ungerichtete Graph noch Zyklen enthalten kann, ist der eigentliche Graph $D(G)$ offensichtlich azyklisch (es gibt keinen gerichteten Kreis). Wir nehmen an, dass der Graph zusammenhängend ist (bzw. betrachten jede (schwache) Zusammenhangskomponente einzeln).

Die Idee ist nun, dass die Wegstrecke des Fahrzeugs mit dem längsten Weg nur einmal durchlaufen werden muss:

Sei E_m der Pfad eines beliebigen Fahrzeugs m in $D(G)$ mit $E_m = p_1 a_1 p_2 \ldots a_{2l-1} p_{2l}$ mit $p_i \in V(D(G))$ sowie $w(a_{2j}) = 0$ und $w(a_{2j-1}) > 0$ für alle $j \in \{1, \ldots, l\}$. Sei $D(G) \setminus E_m$ der Subgraph $(V(D(G)),\ E(D(G)) \setminus \{a_1, \ldots, a_{2l-1}\})$ von $D(G)$, wir streichen also gerade diesen Pfad. Dieser Subgraph zerfalle derart in die schwachen Zusammenhangskomponenten $C_1, \ldots, C_k$, dass es Punkte $p_{i_1}, \ldots, p_{i_k}$ auf E_m gibt mit $i_j < i_{j+1}$, sodass C_j den Pfad E_m in p_{i_j} berührt, aber in keinem Punkt p_r des Pfades mit $r < i_j$.

Nach Lemma 3.2.7 gibt es durch die ungerichtete Version $U(C_j)$ jeder Komponente C_j einen Durchlauf T_j, der jede Kante der Komponente genau zweimal durchläuft, bei p_{i_j} startet und endet und die Ordnung der Originalkanten respektiert.

Dann ist $p_1 \ldots p_{i_1} T_1 p_{i_1} \ldots p_{i_k} T_k p_{i_k} \ldots p_{2l}$ ein Durchlauf durch $U(D(G))$, der die Ordnung der Originalkanten respektiert, jede Kante auf E_m nur einmal und jede andere Kante zweimal durchläuft. Dieser Durchlauf korrespondiert damit zu einem nichtpräemptiven Plan mit Bewegungskosten von höchstens $c_m := 2cost(D(G)) - cost(E_m)$, wenn $cost$ jeweils für die Summe der Kantenkosten steht.

Nun sei m_0 das Fahrzeug mit den größten Bewegungskosten. Diese sind sicher wenigstens $\frac{1}{|M|}$-mal so groß wie die Gesamtkosten von $D(G)$, also $cost(E_{m_0}) \geq \frac{1}{|M|} cost(D(G))$,

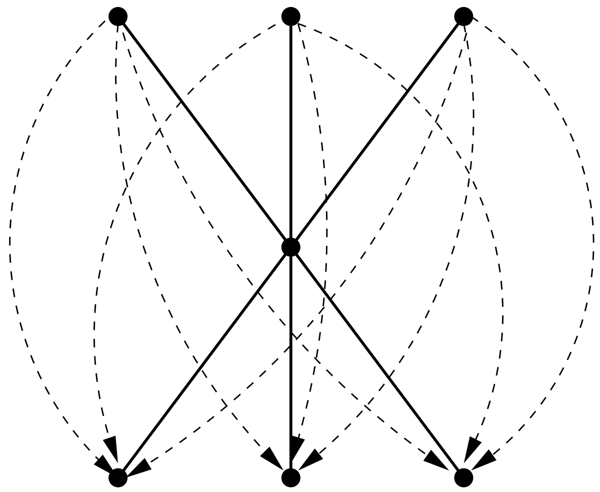

Abbildung 3.6: Das Beispiel zur unteren Schranke von $2-\frac{1}{|M|}$ für $C_\infty M_+$ für 3 Fahrzeuge. Der mittlere Punkt sei das Zentrum z. Die gestrichelten Pfeile gehen jeweils vom Startort eines Gegenstands zu seinem Zielort.

also ist $c_{m_0} \leq (2-\frac{1}{|M|})cost(D(G))$. Es gibt damit einen nichtpräemtiven Plan, der höchstens $(2-\frac{1}{|M|})$-mal so große Bewegungskosten wie der gegebene präemptive Plan hat (und dabei über jede Kante wenigstens dieselben Gegenstände transportiert wie dieser). Die Kosten für das Aufnehmen und Ablegen beim nichtpräemptiven Plan sind natürlich nicht größer als beim präemptiven. ■

Nun geben wir wieder eine Familie von Instanzen an, die diese Schranke asymptotisch auch erreicht:

Satz 3.2.9 *Es gibt eine Familie von (planaren) Instanzen $I(l,n)$ von $C_\infty M_+$ der Größe $O(ln)$ mit n Fahrzeugen und n^2 Gegenständen ($n \in N$), sodass $\frac{opt_{np}(I(l,n))}{opt_p(I(l,n))} \to 2 - \frac{1}{n}$ (für $l \to \infty$).*

Beweis Betrachten wir analog zu Abb. 3.6 folgende Instanzen:

Die Karte ist sternförmig mit $2n$ Strahlen der Länge l. An n der Endpunkte der Strahlen liegt jeweils ein Startort eines Fahrzeugs. An jedem dieser n Fahrzeugstartorte liegen n Gegenstände, die zu den Endpunkten der noch nicht betrachteten Strahlen transportiert werden müssen, und zwar jeder an einen anderen Endpunkt.

In einem *präemptiven Plan* kann jedes Fahrzeug m_i die an seinem Startort liegenden Gegenstände aufnehmen, sie zum Zentrum z des Sterns bringen und sie dort ablegen. Nachdem dies alle Fahrzeuge getan haben, nimmt jedes Fahrzeug m_i die zu einem der restlichen Endpunkte zu bringenden Gegenstände auf und bringt sie dorthin. Die Gesamtkosten sind höchstens $2ln + 4n^2$.

Die Kosten für das Aufnehmen und Ablegen von Gegenständen eines *nichtpräemptiven Plans* sind genau $2n^2$. Für die Bewegungskosten sei P ein optimaler präemptiver Plan.

Wir werden zeigen, dass wir annehmen können, dass in P nur ein einziges Fahrzeug benutzt wird. Benutzen wir nur ein Fahrzeug, so hat dies wenigstens Bewegungskosten von $(4n-2)l$ (es muss alle Endpunkte von Strahlen besuchen), kann aber auch mit solchen Bewegungskosten auskommen. Benutze P nun mehr als ein Fahrzeug und o.B.d.A. transportiere dabei jedes Fahrzeug wenigstens einmal einen Gegenstand. Wir können P dann wie folgt in einen Plan umwandeln, der ein Fahrzeug weniger benutzt, aber nicht mehr Kosten hat:

Wenn jedes benutzte Fahrzeug alle an seinem Startort liegenden Gegenstände aufnehmen würde, müssten diese auch jeweils zu ihren n verschiedenen Zielen gebracht werden und die Fahrtkosten pro Fahrzeug wären wenigstens $2nl$, bei wenigstens 2 Fahrzeugen also schon insgesamt $4nl$, mehr als beim Benutzen eines einzelnen Fahrzeugs. Widerspruch.

Es gibt also ein Fahrzeug m_0, das nicht alle an seinem Startort liegenden Gegenstände aufnimmt. Sei m_1 ein Fahrzeug, das auch Gegenstände aufnimmt, die sich anfangs am Startort von m_0 befinden. Wir lassen nun m_1 die Transporte von m_0 miterledigen: Wenn m_1 zum ersten Mal am Startort von m_0 ist, nehme es alle dort von m_0 oder sich selbst im ursprünglichen Plan aufgenommenen Gegenstände auf und gehe zum Mittelpunkt z. Da es nicht wieder zum Startort von m_0 zurückkehren wird, werden bisher Kosten von $c(loc_0(m_0), z) = l$ eingespart. Nun folgt m_1 dem bisherigen Plan von m_0. Da es sich um nichtpräemptive Pläne handelt, gibt es keine Interaktion zwischen den verschiedenen Fahrzeugen und dies ist problemlos möglich. Dieser Plan ende o.B.d.A. am Endpunkt eines Gegenstands, von dort aus kehre m_1 zum Mittelpunkt z zurück (mit Zusatzkosten l) und folge dem Rest seines eigenen bisherigen Plans. Die Kosten insgesamt verändern sich also nicht.

Damit können wir annehmen, dass nur ein Fahrzeug im optimalen nichtpräemptiven Plan benutzt wird. Dieses hat wenigstens Bewegungskosten von $(4n-2)l$. Es gilt also: $\frac{opt_{np}}{opt_p} \geq \frac{(4n-2)l+2n^2}{2ln+4n^2} = \frac{(2n-1)l+n^2}{ln+2n^2} = \frac{(2n-1)l+n^2+(2n-1)2n-(2n-1)2n}{n(l+2n)} = \frac{2n-1}{n} + \frac{n^2-(2n-1)2n}{n(l+2n)} = 2 - \frac{1}{n} - \frac{4n-2-n}{l+2n} = 2 - \frac{1}{n} - \frac{3n-2}{l+2n} \to 2 - \frac{1}{n}$ (für $l \to \infty$). ■

3.2.3 C_*

Das Verhältnis vom Wert einer optimalen nichtpräemptiven Lösung zum Wert einer optimalen präemptiven Lösung ist bei C_1M_+ und $C_\infty M_+$ wie gesehen jeweils durch eine Konstante beschränkt (die angegebenen Approximationsalgorithmen erzeugen auch in beiden Fällen nichtpräemptive Lösungen).

Bei C_*M_1 ist dies anders (und damit natürlich auch bei C_*M_+): Wenn wir in C_*M_1 nämlich die Kapazität des einzigen Fahrzeugs mit cap bezeichnen, so kann eine optimale nichtpräemptive Lösung um einen Faktor $\Omega(\sqrt{cap})$ schlechter sein als die optimale präemptive Lösung.

Um das einzusehen, passen wir ein entsprechendes Resultat für das eng verwandte Capacitated-Dial-a-Ride-Problem (CDARP) an, das bei Charikar und Raghavachari [CR98] angegeben wird[9] (vgl. auch Abschnitt 2.2.2). Hierzu zunächst einige Vorarbeiten:

[9]Dort wird sogar auch ein Faktor von $\Omega(cap^{2/3})$ bewiesen, allerdings wäre das Resultat etwas komplizierter anzupassen.

Für diese untere Schranke genügt es, spezielle Instanzen zu betrachten.

Definition 3.2.10 (höhenbalancierter Baum) ([CR98], Def. 4) Ein *höhenbalancierter Baum* ist ein gewichteter Baum mit ausgezeichneter Wurzel, für den außerdem gilt:

- Die Distanz von der Wurzel zu einem Blatt ist für jedes Blatt dieselbe.
- Für jedes Blatt ist die Anzahl der Kanten auf dem Weg von der Wurzel zum Blatt gleich.
- Alle Kanten auf derselben Ebene haben dieselbe Länge (hierbei sagt man, eine Kante habe Ebene i, wenn sie die i-te Kante auf dem einfachen Pfad von der Wurzel zu einem Blatt ist; wir bezeichnen die Menge der Kanten der Ebene i auch mit $lev(i)$). □

Lemma 3.2.11 (Wait-Schranke) *(analog [CR98], Lemma 5) Sei I eine Instanz, deren Karte die Form eines höhenbalancierten Baums hat und bei der die Start- und Zielpunkte der Gegenstände an den Blättern dieses Baums liegen. Für 2 Kanten e und e' derselben Ebene l sei $y_{e,e'}$ die Anzahl der Gegenstände, die aus dem Teilbaum unter e in den Teilbaum unter e' transportiert werden müssen. Dann wird in jedem zulässigen Plan mindestens $\frac{1}{\sqrt{cap}}\Sigma_{e,e'\in lev(l)}\lceil\frac{y_{e,e'}}{cap}\rceil$ -mal eine Kante der Ebene l durchlaufen (für jede Ebene l).*

Beweis Die Anzahl der Durchläufe ist offensichtlich unabhängig von möglichen Kosten für Aufnehmen und Ablegen von Gegenständen. Damit kann man den Beweis des Lemmas 5 aus [CR98] wörtlich übernehmen. ■

Wir kommen nun zu einer Familie von Instanzen, die das angesprochene Verhältnis von nichtpräemptivem zu präemptivem Optimum haben:

Definition 3.2.12 (endliche projektive Ebene) ([CR98], Def. 5) Eine *endliche projektive Ebene der Ordnung m* ist ein System von m^2+m+1 Teilmengen $S_1,\ldots,S_{m^2+m+1}$ der Menge $U_m := \{1,\ldots,m^2+m+1\}$ mit:

- $\forall 1 \leq i < j \leq m^2+m+1 : |\{\, S_i \,|\, \{i,j\} \subseteq S_i \,\}| = 1$
- $\forall i \neq j : |S_i \cap S_j| = 1$
- $\forall u \in U_m : |\{\, S_i \,|\, u \in S_i, i \in U_m \,\}| = m+1$
- $\forall i \in U_m : |S_i| = m+1$ □

Man weiß, dass solche Mengensysteme existieren, wenn m eine Primzahlpotenz ist. Sei nun $(U_m,(S_i))$ eine solche projektive Ebene der Ordnung m. Dann definieren wir wie folgt eine Instanz I_m des Planungsproblems:

- $cap = m+1$

- $V = \{r\} \cup (U_m \times \{1, \ldots, m\})$
- $E = \{\, \{r, (i,1)\} \mid i \in U_m \,\} \cup \{\, \{(i,j), (i,j+1)\} \mid i \in U_m, j \in \{1, \ldots, m-1\} \,\}$
- $P = \{\, (i,j) \mid i \in U_m, j \in S_i \,\}$
- $loc_0((i,j)) = (i,m)$
- $loc_G((i,j)) = (j,m)$
- $loc_0(m) = r$

Die Karte ist also im Wesentlichen sternförmig, der einzige Unterschied zu [CR98] ist die Länge der „Strahlen". An jedem Endpunkt eines solchen Strahls liegen $m+1$ Gegenstände, die allesamt zu verschiedenen Zielpunkten gebracht werden müssen. Die Kapazität des Fahrzeugs ist ebenfalls $m+1$, es startet in der Mitte des Sterns.

Satz 3.2.13 *(ähnlich [CR98], Thm. 14) Seien $opt_{np}(m)$ die minimalen Kosten einer nichtpräemptiven Lösung, $opt_p(m)$ die minimalen Kosten einer präemptiven Lösung für die Instanz I_m. Dann gilt $\frac{opt_{np}(m)}{opt_p(m)} = \Omega(cap^{1/2})$.*

Beweis Betrachten wir zunächst die Kosten einer nichtpräemptiven Lösung:

Da die Karte offensichtlich einen höhenbalancierten Baum nach Def. 3.2.10 darstellt, lässt sich Lemma 3.2.11 anwenden: Für jedes $l \in \{1, \ldots, m\}$ gilt: Es gibt $m^2 + m + 1$ Kanten der Ebene l. Sei e eine Kante der Ebene l, etwa $e = (i,l)$. Dann müssen $m+1$ Gegenstände von S_i bzw. e wegtransportiert werden und jeder von diesen zu einem anderen Zweig. Somit ist (für alle Kanten e und e') $y_{e,e'} \leq 1$ und $\Sigma_{e'} \lceil \frac{y_{e,e'}}{cap} \rceil = m+1$. Also $s := \Sigma_{e,e' \in lev(l)} \lceil \frac{y_{e,e'}}{cap} \rceil = (\#Level - l - Kanten)(m+1) = (m^2+m+1)(m+1)$. Da es nun m verschiedene Ebenen gibt, sind die Gesamtbewegungskosten jedes Plans mit Lemma 3.2.11 wenigstens $m\frac{1}{\sqrt{m+1}}s = \Omega(m^{3.5})$.

Andererseits besteht eine mögliche präemptive Lösung darin, zuerst nacheinander zu den Spitzen der Strahlen (also den (i,m)) zu fahren, dort alles einzusammeln und es bei r wieder abzuladen; in einer zweiten Phase pickt man sich nun aus den bei r lagernden Gegenständen für jedes (i,m) gerade die heraus, die dorthin geliefert werden müssen, und bringt sie zu (i,m). Die Kosten dieses Plans lassen sich wie folgt nach oben abschätzen: Die Kosten für Aufnehmen und Ablegen der $(m^2+m+1)(m+1)$ Gegenstände liegen jeweils bei 4, die Transportkosten sind dadurch beschränkt, dass man jeden Strahl nur je einmal fürs Einsammeln und einmal für das Austeilen von Gegenständen durchlaufen muss, also höchstens $4(m^2+m+1)m$, insgesamt $O(m^3)$. Da nun die Kapazität gerade $m+1 = \Theta(m)$ ist, folgt die Behauptung. ∎

Umgekehrt gilt, dass das Verhältnis von optimaler nichtpräemptiver Lösung zu optimaler präemptiver Lösung durch $O(cap)$ beschränkt ist. Dies werden wir z. B. in Satz 3.3.7 sehen.

3.3 Approximationsalgorithmen

Nachdem nun die Vorarbeiten erledigt sind und wir z. B. wissen, dass wir für die M_*-Varianten keine Konstant-Faktor-Approximationen erwarten dürfen und solche für die C_*-Varianten höchstens durch präemptive Pläne erreichbar sind, kommen wir endlich zu den Approximationsalgorithmen.

In diesem Abschnitt bezeichne n stets die Gesamtgröße der Eingabe.

3.3.1 C_1

$C_1 M_1$

Das nichtpräemptive $C_1 M_1$-Problem ist fast genau das 1.8-approximierbare Stacker-Crane-Problem aus dem Artikel von Frederickson, Hecht und Kim [FHK78] (man wähle als Anfangsknoten den Startort des Fahrzeugs und führe für jedes Paar aus Anfangs- und Zielknoten eines Gegenstands eine gerichtete Kante ein), nur dass man jetzt einen Pfad statt einer Tour sucht (und außerdem Kosten für das Aufnehmen und Ablegen von Gegenständen hat, die allerdings nicht relevant für den Approximationsfaktor sind). Die optimale Lösung für das Stacker-Crane-Problem kann als Rundtour also höchstens doppelt so groß sein wie der optimale Plan für das nichtpräemptive $C_1 M_1$-Problem, der wiederum höchstens doppelt so lang sein kann wie der optimale Plan für $C_1 M_1$. Wir bekommen so einfach eine 7.2-Approximation. Diese ließe sich durch Anpassen des LargeArcs-Algorithmus aus [FHK78] auch leicht zu einer 4-Approximation verändern (vgl. $C_1 M_+$ im folgenden Abschnitt).

Man kann aber noch besser direkt die Anpassung aus Robert Mattmüllers Studienarbeit [Mat05] des Algorithmus von Anily und Hassin [AH92] verwenden (die im übrigen auch etwas dem eben erwähnten LargeArcs-Algorithmus ähnelt) und erhält eine 3.5-Approximation. Streng genommen werden wir auch diese Anpassung noch etwas anpassen.[10]

Satz 3.3.1 *$C_1 M_1$ ist 3.5-approximierbar.*

Beweis Sei $move_{opt}$ die Anzahl der Bewegungsaktionen eines beliebigen, aber festen optimalen Plans. Wir betrachten zunächst den vollständigen bipartiten Graphen über den Knotenmengen $loc_0(P_G)$ und $loc_G(P_G)$, in dem die Kanten gewichtet sind mit der Länge der kürzesten Pfade zwischen den Knoten.[11] Die Summe der Distanzen der Gegenstände von ihren Start- zu ihren Zielorten (also der Kanten $\{loc_0(p_i), loc_G(p_i)\}$) ist $\leq move_{opt}$, da ja die Kapazität jedes Fahrzeugs gleich eins ist. Da die zu diesen Distanzen korrespondierenden Kanten ein bipartites Matching zwischen der Menge der Startorte der Gegenstände und der Menge der Zielorte der Gegenstände bilden, sind die Kosten eines minimalen gewichteten bipartiten Matchings zwischen diesen beiden Mengen auch $\leq move_{opt}$ (diese jeweiligen Abschätzungen sind genau, da präemptive Pläne und Pfade statt Touren betrachtet werden). Fassen wir diese beiden Matchings zusammen und

[10] Bei der Wahl der Repräsentanten der Kreise, vgl. den Beweis des folgenden Satzes 3.3.1.

[11] Ist ein Knoten Start- und/oder Zielort mehrerer Gegenstände, so tauche er hier in entsprechend vielen Kopien auf.

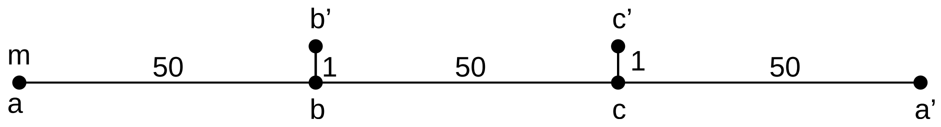

Abbildung 3.7: Beispiel zum Fall C_1M_1

orientieren dabei die Kanten des ersten Matchings in Start-Ziel-Richtung und die des zweiten andersherum, entsteht eine Menge gerichteter Zyklen $\{C_1, \ldots, C_k\}$.

Während Leerfahrten im Fall einer gesuchten Rundtour auch als Transporte „leerer Objekte" aufgefasst werden können (vgl. [AH92]) und so die Gesamtkosten der Kreise durch $move_{opt}$ nach oben abgeschätzt werden können, können wir, da wir einen Pfad statt einer Rundtour suchen, diese Kosten leider nur durch $2move_{opt}$ abschätzen, wie das in Abb. 3.7 gegebene Beispiel zeigt: Im Bild stehe m für den Startort des einzigen Fahrzeugs, die Startpunkte der Gegenstände seien a, b, c, die entsprechenden Zielpunkte a', b', c'. Die Bewegungskosten eines optimalen Plans sind 154, die Kosten des minimalen Matchings und der Start-Ziel-Kanten jeweils 152.

Für jeden Kreis C_j wählen wir nun jeweils einen in ihm enthaltenen Startort $loc_0(p_{i_j})$ eines Gegenstands als Repräsentanten. Falls der Startort $loc_0(m)$ des einzigen Fahrzeugs auch Startort eines Gegenstands p ist, können wir diesen Startort als Repräsentanten eines Kreises wählen, der $loc_0(m)$ enthält.[12]

Nun erstellen wir einen vollständigen Graphen über der Knotenmenge $\{loc_0(m)\} \cup \bigcup_{j\in\{1,\ldots,k\}}\{loc_0(p_{i_j})\}$, dessen Kanten gewichtet sind mit der Kürzeste-Wege-Distanz in (V, E) ihrer Endknoten. Für diesen Graphen erhalten wir mithilfe der Anpassung von Hoogeveen [Hoo90] des Christofides-Algorithmus eine 1.5-Approximation P für den kürzesten bei $loc_0(m)$ startenden Pfad, der jeden Repräsentantenknoten besucht. Da auch der optimale Pfad des Fahrzeugs alle Startknoten von Gegenständen besucht, gilt, dass die Länge von P höchstens $1.5move_{opt}$ ist.

Das Fahrzeug durchlaufe nun den zu diesem Pfad P korrespondierenden Pfad in (V, E). Wenn es dabei auf einen Repräsentantenknoten trifft, durchläuft es den zugehörigen gerichteten Kreis und stellt dabei die entsprechenden Gegenstände zu. Dabei durchläuft es natürlich für jede gerichtete Kante einen Weg gleicher Kosten in (V, E).

Insgesamt durchläuft das Fahrzeug also jeden der Kreise einmal und einmal den verbindenden Pfad. Seine Bewegungskosten sind also höchstens $2move_{opt} + 1.5move_{opt} = 3.5move_{opt}$. Die Kosten für das Aufnehmen und Ablegen von Gegenständen sind, weil es sich um einen nichtpräemptiven Plan handelt, ohnehin nicht größer als die eines optimalen Plans. Es handelt sich also um eine 3.5-Approximation.

Noch kurz zur Laufzeit: Der Algorithmus von Hoogeveen kann analog zum Christofides-Algorithmus in Zeit $O(n^3)$ ausgeführt werden, das Berechnen des bipartiten Matchings minimalen Gewichts und der kürzesten Wege zwischen allen Start- und Zielknoten benötigt asymptotisch nicht mehr Zeit (vgl. Papadimitriou und Steiglitz [PS82] oder Korte und Vygen [KV06]). Die Gesamtlaufzeit ist also $O(n^3)$ und damit polynomiell. ■

[12] Dies ist aber nicht nötig für die Abschätzung.

C_1M_+

Für das C_1M_+-Problem erhalten wir nun eine fast genauso gute Approximation:

Satz 3.3.2 *C_1M_+ ist 4-approximierbar.*

Beweis Ähnlich dem LargeArcs-Algorithmus aus [FHK78]:

Sei $move_{opt}$ wiederum die Anzahl der Bewegungsaktionen eines beliebigen, aber festen optimalen Plans. Analog zum Beweis von Satz 3.3.1 erstellen wir zuerst eine Menge gerichteter Zyklen $\{C_1, \ldots, C_k\}$ über $loc_0(P_G)$ und $loc_G(P_G)$, sodass sich jedes Paar $\{loc_0(p_i), loc_G(p_i)\}$ in Form einer gerichteten Kante $(loc_0(p_i), loc_G(p_i))$ in einem dieser Kreise wiederfindet und die Gesamtkosten der Kreise höchstens $2move_{opt}$ sind, wenn man die Kosten einer Kante als Länge des kürzesten Wegs in (V, E) zwischen den Endknoten der Kante festsetzt.

Nun betrachten wir einen neuen Graphen mit der Knotenmenge $\{s\} \cup loc_0(M) \cup \{C_1, \ldots, C_k\}$. Wir fassen also jeden dieser Kreise als einen neuen Knoten auf und führen Knoten für die Startorte der Fahrzeuge ein sowie einen neuen Knoten s. Dieser sei mit allen Startorten der Fahrzeuge durch Kanten der Länge Null verbunden. Ansonsten führen wir in diesem Graphen Kanten zwischen allen Kreisen ein, die jeweils die Länge der minimalen Distanz zweier Knoten (aus $loc_0(P_G) \cup loc_G(P_G)$) aus diesen Kreisen haben, und Kanten zwischen den Startorten der Fahrzeuge und den Kreisen, deren Länge wiederum der jeweils minimalen Distanz zwischen den Fahrzeugstartorten und Punkten aus den Kreisen entspricht. Für diesen neuen Graphen erstellen wir einen minimalen Spannbaum (kurz MST). Die Länge dieses MST ist offensichtlich $\leq move_{opt}$: Denn wenn man die von einem optimalen Plan durchlaufenen Straßen auf diesen Graphen projiziert und die Startknoten der Fahrzeuge mit dem fiktiven Knoten s verbindet, entsteht ein zusammenhängender Graph, der einen Spannbaum des erstellten Graphen enthält (da der optimale Plan z. B. alle Startorte von Gegenständen besuchen muss) und nicht teurer ist als die optimale Route.

Wir streichen nun aus dem MST wieder die Kanten zu s. Es entsteht ein Spannwald, den wir M' nennen. Wir betrachten wieder den Ursprungsgraphen und ersetzen jede Kante von M' durch einen entsprechenden Pfad gleicher Länge im Ursprungsgraphen. Jedes Fahrzeug durchläuft nun seinen Teil von M', dabei wird jede Kante aus M' doppelt durchlaufen (einmal hin, einmal zurück) und jeder Kreis nur einmal: Das Fahrzeug betritt den Kreis in einem Punkt und durchläuft ihn dann in der gegebenen Richtung. Dabei nimmt es jede Abzweigung zu Nachbarkreisen, besucht den dortigen Teilbaum rekursiv und kehrt auf derselben Kante zurück. Es nimmt Gegenstände immer genau dann auf, wenn es schon alle von seinem Standort auf dem Kreis abgehenden Kanten aus M' besucht hat (also der nächste zu besuchende Start- oder Zielort eines Gegenstands auf demselben Kreis liegt und damit auch schon der Zielort des Gegenstands ist).[13]

Insgesamt werden also die Kreise je einmal und die MST-Kanten je zweimal durchlaufen. Damit ist die Summe der Bewegungskosten $\leq \textit{Kreislängen} + 2 * (\textit{MST-Länge}) \leq$

[13] Oder der letzte Absatz kurz gefasst: Wir führen für jede Kante jedes Teil-MST zwei entgegengesetzt gerichtete Kanten ein. Im resultierenden Graphen gibt es pro Teil-MST jeweils einen Eulerkreis. Diese Eulerkreise werden, übertragen auf den Originalgraphen, zur Lösung.

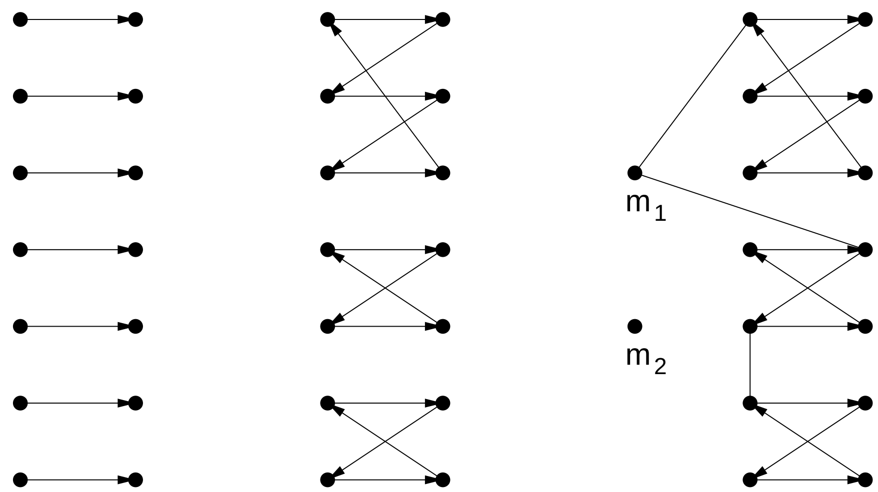

Abbildung 3.8: Ein Beispiel für einen Verlauf des Algorithmus aus Satz 3.3.2. Links die Knoten und Kanten der Start-Ziel-Verbindungen, in der Mitte die entstandenen Kreise, rechts der Spannwald M'.

$2move_{opt} + 2move_{opt} = 4move_{opt}$.

Die Gesamtkosten fürs Aufnehmen und Ablegen sind (da es sich beim hier konstruierten Plan um einen nichtpräemptiven handelt) offensichtlich höchstens so groß wie die für das Aufnehmen und Ablegen in einem optimalen Plans.

Ein Beispiel für den Verlauf des Algorithmus ist in Abb. 3.8 angegeben.

Zur Laufzeit: Das Berechnen der kürzesten Pfade und des bipartiten Matchings kann in Zeit $O(|V||E|+|V|^2\log(|V|))$ geschehen (vgl. [KV06]). Das führt auch zur polynomiellen Gesamtlaufzeit von $O(n^2\log(n))$. ∎

C_1M_*

Wir wissen ja schon, dass wir für C_1M_* keinen konstanten Approximationsfaktor erwarten dürfen, und geben hier nur eine einfache $O(|V|)$-Approximation an.

Satz 3.3.3 *C_1M_* ist $O(|V|)$-approximierbar.*

Beweis Der Algorithmus aus [Hel01], Thm. 3.5, ist eine $O(|V|)$-Approximation. Dieser Algorithmus sei hier kurz wiederholt:

Wir schränken zunächst $road(m)$ für jedes Fahrzeug m auf die von seinem Startort $loc_0(m)$ erreichbaren Kanten ein, etwa per Breitensuche.

Aus den übriggebliebenen Kanten formen wir einen sogenannten *Erreichbarkeitsgraphen RG* (reachability graph), einen beschrifteten Subgraphen von (V, E), in dem wir jede Kante mit der Menge derjenigen Fahrzeuge beschriften, die die Kante benutzen können.

Nun betrachten wir die zu bewegenden Gegenstände nacheinander. Sei p ein solcher. Wir berechnen einen kürzesten Pfad $v_0v_1 \dots v_{k_p}$ von $loc_0(p)$ zu $loc_G(p)$ in RG. Für $i = 0, \dots, k_p - 1$ machen wir nun Folgendes: Wir wählen ein Fahrzeug m, das die Kante $\{v_i, v_{i+1}\}$ durchlaufen kann, entsprechend der Beschriftung in RG, lassen es auf dem kürzesten Pfad innerhalb von $road(m)$ von seinem momentanen Standort aus zu v_i laufen, den Gegenstand dort aufnehmen, ihn in einem Schritt zu v_{i+1} bringen und ihn dort wieder ablegen.

Da die Kapazität jedes Fahrzeugs gleich 1 ist, ist die Summe der kürzesten Wege von den Start- zu den Zielorten der Gegenstände in RG eine untere Schranke für die Kosten eines optimalen Plans. Der angegebene Algorithmus durchläuft diese kürzesten Wege und schafft für jede Kante eines solchen Wegs ein Fahrzeug heran, das diese Kante durchlaufen kann. Die Anfahrtkosten pro solcher Kante sind offensichtlich durch die Anzahl der Orte $|V|$ beschränkt, die Gesamtkosten pro Kante also durch $|V| + 3$. Damit handelt es sich um eine $(|V| + 3)$-Approximation.

Zur Laufzeit: Das Erstellen von RG ist in Zeit $O(|M||V|)$ möglich, die Berechnung aller kürzesten Pfade benötigt höchstens Zeit $O(|V||E| + |V|^2 \log(|V|))$ (vgl. [KV06]), die Gesamtlaufzeit ist also polynomiell (in $O(n^3)$, wenn man den Plan explizit repräsentiert und so manche kürzeste Pfade eventuell mehrfach einfügen muss). ■

3.3.2 C_∞

$C_\infty M_1$

$C_\infty M_1$ ist auf einfache Weise konstant-Faktor-approximierbar:

So korrespondiert der optimale Plan ja zu einem Pfad, der beim Startort des Fahrzeugs beginnt und sowohl die Start- als auch die Endorte der Gegenstände umfasst. Nun erstelle man mit [Hoo90] einen solchen Pfad, dessen Länge höchstens 1.5-mal so groß ist wie die eines solchen Pfades kürzester Länge. Man durchläuft diesen Pfad einmal in Hin- und einmal in Rückrichtung. Dabei sammelt man jeden sich noch nicht an seinem Endort befindenden Gegenstand ein (ist damit spätestens nach dem Durchlaufen in Hinrichtung fertig) und legt ihn beim ersten Besuch seines Endorts dort ab.

Das führt insgesamt zu einer (in Zeit $O(n^3)$ erreichbaren) Approximationsgüte von 3.

Korollar 3.3.4 *$C_\infty M_1$ ist 3-approximierbar.* □

$C_\infty M_+$

Satz 3.3.5 *$C_\infty M_+$ ist 10-approximierbar.*

Beweis Zuerst erstellt man einen Steinerwald für die Mengen $T_p = \{loc_0(p), loc_G(p)\}$ $(p \in P)$. Dieses Problem ist 2-approximierbar (vgl. auch Abschnitt 2.2.3). Da in einer optimalen Lösung (als Menge der durchlaufenen Kanten aufgefasst) auch jeder Startort eines Gegenstands mit dem Zielort desselben Gegenstands verbunden ist, hat der so entstandene Steinerwald höchstens doppelt so große Kosten wie die optimale Lösung.

Seien nun $C_1, \dots, C_k$ die Zusammenhangskomponenten dieses Walds. Wir fassen diese als neue Knoten auf und erstellen ganz analog zum Beweis von Satz 3.3.2 einen optimalen an der Menge der Startorte der Fahrzeuge gewurzelten Spannwald M' für diese Komponenten, wobei wir wiederum die Kürzeste-Wege-Distanzen zugrunde legen. Da die durchlaufenen Kanten der Pfade in der optimalen Lösung insgesamt jeden Startknoten eines Gegenstands mit einem Startknoten eines Fahrzeugs verbinden, sind die Kosten dieses Spannwalds höchstens so groß wie die einer optimalen Lösung.

Jetzt ersetzt man die Kanten des Spannwalds durch entsprechende Wege im Ursprungsgraphen.[14] Dann kann ein Fahrzeug den mit ihm verbundenen Graphen nun so durchlaufen, dass jeder zu einer Kante aus M' korrespondierende Weg nur zweimal durchlaufen wird: Erreicht das Fahrzeug eine Steinerwaldkomponente, so durchläuft es diese und sammelt alle ihm begegnenden Gegenstände ein. Trifft es dabei auf den Beginn eines zu einer Kante aus M' korrespondierenden Pfades, der zu einer noch nicht durchlaufenen Steinerwaldkomponente führt, so nimmt es diese Abzweigung und behandelt rekursiv die dadurch zu erreichenden Komponenten. Sind schon alle abzweigenden Kanten einer Steinerwaldkomponente durchlaufen und alle Gegenstände dieser Komponente eingesammelt, so durchläuft das Fahrzeug diese Komponente erneut und stellt dabei die Gegenstände an ihren Zielen zu.

Die Fahrtkosten aller Fahrzeuge sind also $\leq 2 * \textit{Spannwaldgröße} + 4 * \textit{Steinerwaldgröße} \leq 2 * move_{opt} + 4 * 2move_{opt} \leq 10move_{opt}$. Da es sich beim konstruierten Plan um einen nichtpräemptiven handelt, ist dieser eine 10-Approximation.

Zur Laufzeit: Die Berechnung der Steinerwald-Approximation ist in Zeit $O(|V|^2 \log(|V|))$ möglich (vgl. Goemans und Williamson [GW96], S. 164), die Berechnung der kürzesten Wege in $O(|V||E| + |V|^2 \log(|V|))$ (vgl. [KV06]). Die Gesamtlaufzeit ist also polynomiell ($O(n^2 \log(n))$). ∎

$C_\infty M_*$

Satz 3.3.6 *$C_\infty M_*$ ist $O(n)$-approximierbar.*

Beweis Man kann o. B. d. A. davon ausgehen, dass $road(m)$ für jedes Fahrzeug m zusammenhängend ist (sonst schränke man es auf die Zusammenhangskomponente des Startorts des Fahrzeugs ein). Man betrachtet nun folgenden Graphen (V', E'):

$V' = \{\, (v, m) \,|\, v \in V \text{ und } m \in M \text{ und } \exists e \in road(m) \text{ mit } v \in e \,\} \cup V$

mit der Kantenmenge

$E' = \{\, \{(v, m), (w, m)\} \,|\, m \in M, (v, w) \in road(m) \,\} \cup \{\, \{(v, m), v\} \,|\, (v, m) \in V' \,\}$

In diesem Graphen erstellt man einen Steinerwald bezüglich der Mengen $\{loc_0(p), loc_G(p)\}$ $(p \in P)$ (jeder Start- ist mit dem entsprechenden Endpunkt verbunden) mittels der in Abschnitt 2.2.3 erwähnten 2-Approximation.

[14]Dabei nehmen wir an, dass diese Pfade jeweils nur in ihren Endpunkten Steinerwaldkomponenten berühren. Ansonsten ignoriere ein Fahrzeug, das diesen Pfad durchläuft, einfach die anderen Berührungspunkte.

In diesem Wald gibt es nun für jeden Gegenstand einen eindeutigen kürzesten Weg vom Start zum Ziel. Man ersetzt im Wald jede Kante durch 2 entgegengesetzt gerichtete Kanten und bezeichnet jede gerichtete Kante mit der Menge der Gegenstände, deren kürzester Pfad über diese Kante führt. Jede zu einer Originalkante korrespondierende gerichtete Kante muss dabei nur einmal durchlaufen werden, jede zu einer Pickup- oder Drop-Aktion korrespondierende Kante höchstens $|P|$-mal (das sind Kanten der Form $\{(v, m), v\}$). Man schaffe für das Durchlaufen einer Originalkante das entsprechende Fahrzeug heran mit einem Anlauf von höchstens $O(|V|)$. Eine zu einer Originalkante korrespondierende gerichtete Kante werde immer erst dann durchlaufen, wenn alle durch sie zu transportierenden Gegenstände schon an dem Beginn der Kante sind. Da es sich bei dem zugrundeliegenden Graphen um eine Menge von Bäumen handelt und die Wege eines Gegenstands den ungerichteten kürzesten Wegen in diesen Bäumen entsprechen, gibt es immer eine solche Kante. Insgesamt ergibt sich also eine $O(\max\{|P|, |V|\})$-Approximation.

Noch kurz zur Laufzeit: Die Größe von V' ist in $O(|M||V|) = O(n^2)$, zusammen mit der in Satz 3.3.5 erwähnten Steinerwald-Implementierung ergibt sich eine polynomielle Laufzeit von $O(n^4 \log(n))$. ■

3.3.3 C_*

C_*M_1

Sei im Folgenden stets *cap* die Kapazität des einzigen Fahrzeugs.

Sehr eng verwandt mit dem C_*M_1-Problem ist das präemptive Capacitated-Dial-a-Ride-Problem (pCDARP) (vgl. Abschnitt 2.2.2 oder Charikar und Raghavachari [CR98]). Der Unterschied zu C_*M_1 besteht nur darin, dass einerseits hier das Aufnehmen und Ablegen von Gegenständen Kosten verursacht, im pCDARP nicht, und andererseits im pCDARP eine Rundtour gefordert wird.[15] In [CR98] wird (halb implizit, s. u.) eine $O(\log |V|)$-Approximation angegeben, meines Wissens die bisher beste Approximation für dies Problem. Leider lässt sich diese nicht gut auf den vorliegenden Fall übertragen. Sie besteht nämlich aus einer $O(1)$-Approximation für das pCDARP auf speziellen höhenbalancierten Bäumen (vgl. dazu Abschnitt 3.2.3), die dann mithilfe eines Resultats über die Einbettbarkeit von Metriken auf andere Instanzen erweitert wird. Nun kann man sich einerseits Gedanken machen, inwieweit man dieses Übertragungsresultat bei zusätzlichen Kosten für das Aufnehmen und Ablegen noch sinnvoll verwenden kann (ein zusätzlicher Faktor von $\Omega(|P|)$ würde nicht helfen) oder ob man diese Zusatzkosten irgendwie in die Metrik integrieren kann, aber der Versuch einer sinnvollen Übertragung scheitert schon früher: Der angegebene Algorithmus auf (gewurzelten) Bäumen arbeitet nämlich mit sehr vielen Ablege- und Aufnehmeaktionen und kann beim C_*M_1-Problem im schlechtesten Fall einen um den Faktor $\Omega(cap)$ gegenüber einem optimalen Plan schlechteren Plan liefern:

[15] Die optimale Rundtour ist aber höchstens doppelt so lang wie der optimale Pfad. Da wir ohnehin keine Konstant-Faktor-Approximation angeben werden, ist dieser Unterschied also für die folgenden Ausführungen nicht sehr wichtig.

Der Algorithmus besteht aus 2 Phasen, einer Up-Phase und einer Down-Phase. In der Up-Phase wird jeder Gegenstand zum höchsten Punkt des kürzesten Pfades zu seinem Zielpunkt gebracht. Dies geschieht rekursiv über die Teilbäume. Dabei wird jede Kante nur einmal betrachtet, alle über sie hinweg zu transportierenden Gegenstände liegen dabei schon am unteren Punkt dieser Kante und werden durch mehrfaches Auf- und Ablaufen dieser Kante an deren oberen Punkt gebracht. In der Down-Phase werden die Gegenstände auf analoge Weise zu ihren Zielorten gebracht. Als Beispiel betrachte man einen Baum, von dessen Wurzel zwei gleichlange Ketten der Länge cap abgehen und von einem der Blätter zum anderen $x * cap$ Gegenstände transportiert werden müssen. Der Algorithmus hat $4x * cap^2$ Aufnahme- und Ablegeaktionen. Es besteht aber auch die Möglichkeit, die Gegenstände in Portionen von je cap Stück nichtpräemptiv direkt zu ihrem Ziel zu transportieren, was nur $4x * cap + 2x * cap = 6x * cap$ Aktionen erfordert.

Eine eventuelle direkte Übertragung auf allgemeine Karten würde so einen schlechteren Approximationsfaktor liefern als der erste Algorithmus im folgenden Absatz.

Wir können recht einfach eine $O(\sqrt{|P|})$-Approximation erreichen.

Lemma 3.3.7 *C_*M_1 ist $O(cap)$-approximierbar.*

Beweis Wir betrachten nun eine feste C_*M_1-Instanz und wenden den vorgestellten Approximationsalgorithmus für C_1M_1 darauf an. Da die Kapazität des jetzt verwendeten Fahrzeugs höchstens größer als 1 ist, stellt dieser Plan eine zulässige Lösung für unsere C_1M_1-Instanz dar. Eine untere Schranke für die Kosten jedes Plans ist $\frac{1}{cap}\Sigma_{p\in P}c(loc_0(p), loc_G(p))$, da pro Schritt höchstens cap Gegenstände gleichzeitig transportiert werden können. Die vom C_1M_1-Algorithmus ausgegebene Lösung besteht aus einer Menge von Kreisen und einer Verbindung dieser Kreise durch einen Pfad. Die Kosten dieser Lösung seien $c_{cyc} + c_{rest}$, wobei c_{cyc} die Gesamtkosten der Kreise bezeichne. Dann gilt $c_{cyc} \leq 2\Sigma_{p\in P}c(loc_0(p), loc_G(p))$, also sind die Kosten einer optimalen Lösung wenigstens $\frac{1}{2cap}c_{cyc}$. Die minimalen Kosten eines Pfades durch die Startpunkte der Gegenstände und den des Fahrzeugs sind auch im C_*M_1-Fall eine untere Schranke für die Gesamtkosten, also ist auch $\frac{2}{3}c_{rest}$ eine untere Schranke. Somit gilt für die Kosten der vom C_1M_1-Algorithmus ausgegebenen Lösung $c_{cyc} + c_{rest} \leq (2cap + 3/2)c_{opt}$, wenn c_{opt} die optimalen Kosten einer Lösung bezeichnet. Damit ist der C_1M_1-Algorithmus eine $O(cap)$-Approximation. ■

Lemma 3.3.8 *C_*M_1 ist $O(\frac{|P|}{cap})$-approximierbar.*

Beweis Wir betrachten wieder eine feste C_*M_1-Instanz und wenden nun den vorgestellten Approximationsalgorithmus für $C_\infty M_1$ darauf an. Dieser besteht einfach daraus, einen Pfad durch alle Start- und Endpunkte zu finden, diesen einmal in Hin- und einmal in Rückrichtung zu durchlaufen und dabei alles sobald wie möglich einzusammeln und auszuteilen. Wir durchlaufen diesen Pfad nun mehrmals mit dem Fahrzeug beschränkter Kapazität, sammeln wieder so viel wie jeweils möglich ein und legen es ebenfalls so früh wie möglich an den Zielpunkten wieder ab. Dabei genügen sicher $\lceil\frac{|P|}{cap}\rceil$ Durchläufe. Man erhält also eine $3\lceil\frac{|P|}{cap}\rceil$-Approximation. ■

Korollar 3.3.9 *C_*M_1 ist $O(\sqrt{|P|})$-approximierbar.*

Beweis Es gilt $cap\frac{|P|}{cap} = |P|$, also $min\{cap, \frac{|P|}{cap}\} \leq \sqrt{|P|}$. Mit den Lemmata 3.3.7 und 3.3.8 folgt die Behauptung (man berechne beide Approximationen und verwende die bessere Lösung). Die Laufzeit ist dabei wieder $O(n^3)$. ■

C_*M_+

In dem *Spezialfall*, in dem alle Gegenstände entweder von demselben Startort aus starten oder zu demselben Zielort gebracht werden müssen, ist C_*M_+ und damit auch C_*M_1 konstant-Faktor-approximierbar. O. B. d. A. sei der Startort gemeinsam. Sei nämlich nun m das Fahrzeug mit der größten Kapazität unter denen, die in einer optimalen Lösung benutzt werden. Dann kann man sich darauf beschränken, nur dies zu benutzen, da man es nach jedem Ausliefern wieder zurück zum gemeinsamen Startort fahren kann, um es für die nächste Auslieferungsrunde zu benutzen, und dadurch höchstens doppelt so viele Bewegungsschritte hat wie in der optimalen Lösung. Außerdem kann man nun genausogut die Gegenstände als gleichartig annehmen (man nehme bei jeder Auslieferungsrunde genau die entsprechenden Gegenstände mit). Das verbleibende Problem ist im Wesentlichen bekannt unter dem Namen Capacitated-Vehicle-Routing-Problem (CVRP) und 2.5-approximierbar durch eine nichtpräemptive Tour (vgl. Abschnitt 2.2.2 und Altinkemer und Gavish [AG90]). Das ergibt dann eine 5-Approximation für den Spezialfall unseres Problems.

Außerdem lassen sich für den *allgemeinen Fall* leicht analog zum Fall C_*M_1 $O(maxcap)$- oder $O(|P|/mincap)$-Approximationen angeben, das Problem ist also insbesondere $O(|P|)$-approximierbar. Hierbei sei $maxcap = \max_{m \in M} cap(m)$ und $mincap = \min_{m \in M} cap(m)$. Für die $O(maxcap)$-Approximation verwende man einfach den Algorithmus für C_1M_+ und für die $O(|P|/mincap)$-Approximation den für $C_\infty M_+$, bei dem man dann die Pfade der einzelnen Fahrzeuge mehrmals von denselben Fahrzeugen durchlaufen lässt. Die Gütebeweise verlaufen dabei völlig analog zum Fall C_*M_1.

Korollar 3.3.10 *C_*M_+ ist $O(|P|)$-approximierbar.* □

C_*M_*

C_*M_* ist $O(min\{|V||P|, |V|L\})$-approximierbar (wobei L für die Länge des längsten kürzesten Pfades von einem Start- zu einem Endpunkt steht)[16], also insbesondere $O(n^2)$-approximierbar. Ein Algorithmus, der diese Güte hat, ist nämlich der aus [Hel01], Thm. 3.5, der hier auch in Satz 3.3.3 vorgestellt wurde. Denn die Kosten des von ihm erzeugten Plans sind durch $3 * |P| * L * |V|$ beschränkt (pro Kante Drop, Pickup, Transport und $|V|$ Anlauf, Länge jedes Pfades durch L beschränkt, $|P|$ Pfade), die optimalen Kosten mindestens $2|P| + L$ (Pickup- und Drop-Aktionen, Transport).

Korollar 3.3.11 *C_*M_* ist $O(n^2)$-approximierbar.* □

[16] Strenggenommen wieder bezogen auf den Erreichbarkeitsgraphen RG aus Satz 3.3.3.

Die in diesem Abschnitt 3.3 erzielten Approximationsresultate sind in Abb. 4.1 auf S. 68 noch einmal bildlich zusammengefasst.

3.4 Abgewandelte Problemstellungen

Wir betrachten nun noch verschiedene abgewandelte Problemstellungen, stellen fest, welche Resultate übertragbar sind, und diskutieren hierfür weitergehende Approximierbarkeitsfragen, meist ohne sie definitiv zu beantworten.

3.4.1 Parallele Pläne

Man kann statt der Gesamtlänge des Plans auch andere Kostenmaße verwenden: Ein naheliegendes ist es, zu erlauben, dass Aktionen verschiedener Fahrzeuge gleichzeitig ablaufen, sofern sie nicht miteinander kollidieren (z. B. dadurch, dass zwei Fahrzeuge denselben Gegenstand aufheben), und dann die Anzahl der Zeitschritte zu messen, bis die letzte Aktion des Plans ausgeführt ist (also etwa der letzte Gegenstand zugestellt ist).

Sei par-C_iM_j die Variante von C_iM_j mit derart veränderter Kostenfunktion.

Glücklicherweise müssen wir uns keine neuen Härteresultate überlegen, die Korollare 3.1.8 und 3.1.10 bleiben nämlich auch für diese Varianten gültig, da nur das dort Extrafahrzeug genannte Fahrzeug nichtkonstante Kosten hat und so im Wesentlichen das Ende jedes „guten“ Plans bestimmt.[17]

Wenn die Fahrzeuge nicht miteinander interagieren, wie das etwa bei nichtpräemptiven Plänen der Fall ist, besteht eine enge Verbindung zu den sogenannten Minmax-Überdeckungsproblemen in Graphen. Eine spezielles sei kurz vorgestellt:

Definition 3.4.1 (Minmax-R-Rooted-Tree-Cover-Problem, aus [EGK⁺03]) Gegeben sei ein ungerichteter Graph G mit ganzzahligen, positiven Kantengewichten. Eine Baumüberdeckung des Graphen ist eine Menge $T = \{T_i\}_i$ von Teilbäumen T_i des Graphen mit $\bigcup_i V(T_i) = V(G)$. Das Gewicht $w(T)$ eines Baumes sei die Summe seiner Kantenkosten, die Kosten einer Baumüberdeckung $T = \{T_i\}_i$ seien $\max_i w(T_i)$. Ist außer G eine Teilmenge $R \subseteq V$ gegeben und sucht man eine kostenminimale Baumüberdeckung $T = \{T_i\}_i$ mit der Eigenschaft, dass jeder Baum T_i einen anderen Knoten aus R als ausgezeichnete Wurzel besitzt, so spricht man vom *Minmax-R-Rooted-Tree-Cover-Problem.* Dabei können sich die Bäume T_i Knoten und Kanten teilen, nur die jeweils ausgezeichneten Wurzeln müssen verschieden sein. □

Das Minmax-R-Rooted-Tree-Cover-Problem ist 4-approximierbar (Even et al. [EGK⁺03]). Identifiziert man die Menge der Startpunkte von Fahrzeugen mit der Menge R, so gibt es eine gewisse Ähnlichkeit zu den M_+-Varianten der Transportprobleme.

Es gibt auch viele andere Überdeckungsprobleme: Ist außer G in einer Instanz eine natürliche Zahl k gegeben und sucht man eine ungewurzelte kostenminimale Baumüberdeckung der Mächtigkeit höchstens k, dann spricht man vom *Minmax-k-Tree-Cover-Problem.* Auch dieses ist 4-approximierbar [EGK⁺03]. Hat man in jeder Instanz als Ein-

[17] Die Kosten dieses Fahrzeugs plus 3 sind die Kosten eines entsprechenden parallelen Plans.

gabe einen Knoten als feste Wurzel gegeben, die jeder Baum einer Überdeckung enthalten muss, und außerdem ebenso Gewichte für Knoten wie für Kanten, spricht man vom *Minmax-Rooted-Subtree-Cover-Problem*, das $(3-\frac{2}{k+1})$-approximierbar ist (Nagamochi [Nag05]). Verwandt damit sind die von Frederickson, Hecht und Kim in [FHK78] untersuchten Minmax-Varianten des Stacker-Crane-, des Traveling-Salesman- und des Chinese-Postman-Problems, in denen auch ein Knoten gegeben ist, von dem aus eine Überdeckung gefunden werden muss, allerdings durch Kreise. Ungewurzelte Varianten von Vehicle-Routing-Problemen werden etwa von Arkin, Hassin und Levin in [AHL04] untersucht, u.a. mit Überdeckungen durch Pfade oder Überdeckungen durch solche Kreise, die gegebene Kantenmengen statt Knotenmengen überdecken.

Für par-C_1M_+ könnte man den vollständigen mit der Länge der kürzesten Wege gewichteten Graphen über den Startknoten der Gegenstände und Fahrzeuge betrachten, der ergänzt wird durch je einen zusätzlichen Knoten für jeden Gegenstand, der mit dem entsprechenden Startknoten des Gegenstands durch eine mit der Start-Ziel-Distanz dieses Gegenstands gewichteten Kante verbunden ist. Als Wurzelmenge R könnte man dann die Startorte der Fahrzeuge wählen. Mit der 4-Approximation für das R-Rooted-Tree-Cover-Problem würde man so eine Konstant-Faktor-Approximation für die nichtpräemptive Variante von par-C_1M_1 erhalten. Falls nun für die Varianten paralleler Pläne ähnliche Abschätzungen für das Verhältnis nichtpräemptiver zu präemptiver Lösungen gelten wie für die Originalprobleme, wäre damit auch par-C_1M_+ in APX.

Für par-$C_\infty M_+$ wird die Fragestellung nicht einfacher: Startet man mit einer Steinerwaldapproximation so wie bei $C_\infty M_+$, so stellt sich die Frage, wie man verschiedene Komponenten dieses Waldes möglichst simultan so zerlegen kann, dass die maximale Größe einer Komponente abnimmt, ohne dass die Zahl der Komponenten zu sehr zunimmt. Es müssten ja immer noch Start- und Zielknoten jedes Gegenstands in einer Komponente verbunden sein.

Das Entwickeln von Algorithmen für die M_*-Fälle wäre auch noch interessant.

3.4.2 Vollständige Graphen

Lassen wir als Karten der Fahrzeuge nur vollständige Graphen zu, so werden die Transportprobleme meist einfacher. Das liegt aber nur daran, dass die Kantenkosten uniform sind. Denn da es nicht verboten ist, Knoten mehrfach zu besuchen, könnten wir in den eigentlichen Problemen auch jede Karte als vollständig annehmen, indem wir neue Kanten mit den Kürzeste-Wege-Distanzen zwischen Knoten einführen und uns für jedes Fahrzeug auf die von ihm erreichbaren Orte seiner Karte beschränken.

Nicht viel einfacher wird $C_\infty M_1$. Zumindest bleibt es in diesem Fall APX-hart, wie Helmert gezeigt hat [Hel05].

Heiße die Variante mit vollständigem Graphen von C_iM_j nun complete-C_iM_j.

Ich gebe nun für complete-C_1M_1 und complete-C_1M_+ einfache polynomielle Algorithmen an.

Satz 3.4.2 *complete-C_1M_1 ist in P.*

Beweis Da die Kapazität jedes Fahrzeugs ebenso gleich 1 ist wie die Distanz zwischen Start- und Zielort jedes Gegenstands, können wir annehmen, dass jeder Gegenstand in einem optimalen Plan sofort nach dem Aufnehmen zu seinem Ziel gebracht und dort abgelegt wird. Jeder zulässige Plan für eine complete-C_1M_1-Instanz, als Folge gerichteter Kanten aufgefasst, kann dann durch Hinzufügen einer einzelnen zusätzlichen Kante der Kosten 1 zu einer Rundtour ergänzt werden, die jede gerichtete Start-Ziel-Kante einmal durchläuft.

Betrachten wir also den gerichteten Multigraphen $G = (V', A)$ über $V' = loc_0(P) \cup loc_G(P)$, der für jeden Gegenstand p eine Kante $(loc_0(p), loc_G(p))$ enthält. Diese Kanten seien als „alte" Kanten bezeichnet und korrespondieren nun zu den auf jeden Fall von einem (wie eben beschriebenen) Plan zu durchlaufenden Kanten. Wir ergänzen jetzt diesen Graphen zunächst auf minimale Weise zu einer Rundtour.

Dazu ergänzen wir ihn zuerst so durch gerichtete Kanten, dass für den neuen Eingangs- bzw. Ausgangsgrad $\delta_n^{+/-}$ jedes Knotens v gilt $\delta_n^{+/-}(v) = \max\{\delta^+(v), \delta^-(v)\}$. Dies ist in linearer Zeit möglich. Es entsteht eine Menge starker Zusammenhangskomponenten, die jeweils eine minimale Ergänzung der in ihnen vertretenen alten Kanten zu einer derartigen Rundtour durch die in ihnen vertretenen Punkte darstellen, die jede Kante einmal durchläuft (also zu einem Eulerkreis).

Nun verschmelzen wir Komponenten paarweise derart, dass die Kanten in den jeweils neu entstehenden Komponenten weiterhin eine solche minimale Ergänzung bilden. Seien für eine Komponente K die Kosten einer optimalen Rundtour, die alle alten Kanten innerhalb dieser Komponente einmal durchläuft, $opt(K)$.

Seien N_1 und N_2 zwei Komponenten mit neuen Kanten (u_1, v_1) und (u_2, v_2). Wir ersetzen diese Kanten durch die Kanten (u_1, v_2) und (u_2, v_1) und erhalten eine neue Komponente ohne zusätzliche Kosten.
Sei A eine Komponente, die nur aus alten Kanten besteht, und N eine mit wenigstens einer neuen Kante. In einer Rundtour durch beide müssen diese beiden Komponenten verbunden sein durch $2k$ Kanten für ein $k \geq 1$. Dann zerfällt der Teil der Tour innerhalb von N in höchstens k getrennte Teilpfade, die also insgesamt mindestens Kosten von $opt(N) - k$ haben. Die Gesamtkosten sind dann $opt(A) + opt(N) - k + 2k = opt(A) + opt(N) + k$, die für $k = 1$ minimiert werden. Sei also v ein Knoten aus A und (u, w) eine neue Kante aus N. Wir streichen (u, w) und ersetzen sie durch neue Kanten (u, v) und (v, w). So erhalten wir eine optimale Rundtour durch A und N.
Seien nun A_1 und A_2 zwei nur aus alten Kanten bestehende Komponenten. Diese können durch zwei zusätzliche Kanten zu einem Eulerkreis durch beide ergänzt werden (für zwei feste Knoten $v_1 \in A_1$ und $v_2 \in A_2$ führe man neue Kanten (v_1, v_2) und (v_2, v_1) ein), was offensichtlich auch optimal ist.

Wir erhalten also so eine optimale Rundtour, die alle alten Kanten durchläuft.

Sei nun noch v_0 der Startknoten des Fahrzeugs.
Ist $v_0 \in V'$, so liegt v_0 auf der Tour. Diese kann dann auch als Plan benutzt werden, falls v_0 keine eingehende neue Kante besitzt. Sonst können wir noch eine eingehende neue Kante (v, v_0) zu v_0 streichen, falls so eine existiert, denn eine Rundtour der Form $v_0 \dots v v_0 \dots v_1 v_0$ kann natürlich durch eine Rundtour der Form $v_0 \dots v_1 v_0 \dots v v_0$ ersetzt

werden, deren letzte Kante dann gestrichen werden kann.

Sonst sei (v, w) eine neue Kante in der Tour, falls so eine existiert. Wir ersetzen sie durch (v_0, w) und erhalten analog zu eben einen optimalen Pfad für das Fahrzeug, entlang dem die Gegenstände zugestellt werden können. Ist in der sonstigen Rundtour keine neue Kante enthalten, so ergänzen wir einfach (v_0, w) für einen beliebigen Knoten $w \in V'$ und erhalten einen optimalen Pfad. ∎

Satz 3.4.3 *complete-C_1M_+ ist in P.*

Beweis Seien $M = \{m_1, \ldots, m_{|M|}\}$ und I_k das Teilproblem, das nur die Benutzung der Fahrzeuge aus $\{m_1, \ldots, m_k\}$ erlaubt. Sei k mit $1 \leq k < |M|$ gegeben und eine optimale Lösung als Menge gerichteter Kanten für I_k in Analogie zu Satz 3.4.2 (für I_1 können wir eine solche optimale Lösung mit vorigem Satz erstellen). Wir werden diese in konstanter Zeit in eine optimale Lösung für I_{k+1} umwandeln: Zunächst ist klar, dass wir durch das Benutzen eines zusätzlichen Fahrzeugs höchstens eine Kante einsparen können. Sei nun v_1 der Startort von m_{k+1}. Hat v_1 eine eingehende neue Kante (bezogen auf den bisherigen Plan), so können wir diese streichen und im bisherigen Plan den von einem Fahrzeug gelaufenen Weg $vv_1w\ldots$ durch den von m_{k+1} gelaufenen Weg $v_1w\ldots$ ersetzen (da jeder Gegenstand sofort zugestellt wird, ist das problemlos möglich). Ist das nicht der Fall, so können wir keine Kosten einsparen, wie man sich leicht überlegen kann. ∎

Wenigstens sind nun auch alle derartigen Varianten der Originalprobleme konstant-Faktor-approximierbar:

Satz 3.4.4 *complete-C_*M_* ist 2-approximierbar.*

Beweis Sei eine C_*M_*-Instanz gegeben, in der die Karte jedes Fahrzeugs vollständig ist, eingeschränkt auf die von ihm erreichbaren Knoten, und in der die Vereinigung dieser eingeschränkten Karten wiederum ein vollständiger Graph ist. Die optimale Lösung hat durch Aufnehmen und Ablegen wenigstens Kosten von 2 pro Gegenstand. Fährt man pro Gegenstand ein Fahrzeug heran, das die Kante zwischen dessen Start- und Zielpunkt durchfahren kann, und lässt es den Gegenstand zustellen, hat es dabei höchstens Kosten von 4. ∎

3.4.3 Planare Graphen und euklidische Distanzen

Auch eingeschränkt auf Instanzen, deren Karte planar ist, werden die Transportprobleme sicherlich nicht schwieriger. Aber werden sie so leicht, dass man für C_1M_1 oder $C_\infty M_1$ in diesem Spezialfall ein *PTAS* angeben kann? Ich weiß es nicht, werde das allerdings jetzt noch näher ausführen.

Diese Spezialfälle heißen im Folgenden planar-C_1M_1 und planar-$C_\infty M_1$.

Ein Graph heißt *planar*, wenn er so in der Ebene gezeichnet werden kann, dass sich keine zwei seiner Kanten überschneiden. Man spricht auch von einer entsprechenden *Einbettung* in die Ebene. Eine formalere Definition von Planarität findet sich etwa bei Korte und Vygen [KV06], S. 33, ist für die Diskussion hier aber nicht notwendig.

Es sei noch darauf hingewiesen, dass es für die Definition von Planarität egal ist, ob man den Graphen in die Ebene ($\mathbb{R}^2$) oder in die Oberfläche einer Kugel einbettet (vgl. McHugh [McH90]).

Planare Graphen haben einige Vorteile gegenüber allgemeinen Graphen, die sie algorithmisch leichter behandelbar machen: So gilt z. B., dass die Anzahl der Kanten eines planaren Graphen mit $n \geq 3$ Knoten höchstens $3n - 6$ sein kann. Auch findet man für planare Graphen stets aufspannende Subgraphen, die gleichzeitig „klein" im Vergleich zur Größe eines MST sind und Kürzeste-Wege-Distanzen „relativ gut" erhalten (vgl. z. B. [AGK$^+$98]). Nicht zu vergessen die Möglichkeit, eine Einbettung in die Ebene in linearer Zeit zu finden (vgl. [McH90]) und den eingebetteten Graphen z. B. so einfach in Teilgraphen zerlegen zu können. Für planare Graphen gelten auch sogenannte Separator-Sätze, die ihre Zerlegbarkeit erleichtern (vgl. z. B. [AGK$^+$98]).

Viele NP-Optimierungsprobleme, die für allgemeine Graphen als Eingabe APX-hart sind, besitzen eingeschränkt auf planare Graphen ein PTAS: So z. B. Maximum-Independent-Set, Minimum-Dominating-Set, Minimum-Vertex-Cover (Baker [Bak94], vgl. auch Khanna und Motwani [KM96]) und insbesondere das TSP. Die TSP-Variante, die als Eingabe einen (gewichteten) planaren Graphen hat und in der eine Rundtour gesucht wird, die jeden Knoten mindestens einmal besucht, heißt auch *(Weighted-)Planar-Graph-TSP*. Für Weighted-Planar-Graph-TSP wurde bei Arora et al. [AGK$^+$98] ein PTAS angegeben, aufbauend auf dem ungewichteten Fall, für den von Grigni, Koutsoupias und Papadimitriou [GKP95] ein PTAS angegeben wurde.

Während Baker in [Bak94] selbst sagt, ihre Techniken seien z. B. nicht auf Hamiltonsche Pfade anwendbar (S. 157), und die Verallgemeinerung dieser Techniken in [KM96] laut den Autoren zumindest nicht in offensichtlicher Weise für das planare TSP anwendbar sind (S. 330-332), kann man sich fragen, ob die Techniken aus [AGK$^+$98] auch dafür genutzt werden können, für C_1M_1 oder $C_\infty M_1$ ein PTAS zu erhalten. Schließlich sind die Instanzen, die in der Reduktion von (1,2)-TSP in Satz 3.1.2 entstehen, nicht planar, ein solches PTAS ist also nicht ausgeschlossen.

Wenn wir, statt einen ungewichteten Graphen als Karte zu nehmen, die Orte als Punkte in einem zweidimensionalen euklidischen Raum auffassen, und die Kostenfunktion entsprechend anpassen, sodass nun für Fahrten Kosten entsprechend der euklidischen Distanz von Punkten anfallen, entstehen weitere Problemvarianten. Für das Euklidische TSP werden von Arora [Aro98] und Mitchell [Mit98] Approximationsschemata angegeben. Arora gibt in [Aro04] einen Überblick über die für euklidische Probleme verwendeten Techniken.

Die Funktionsweise dieser Algorithmen für das (Weighted-)Planar-Graph-TSP oder das Euklidische TSP ist nun wie folgt: Man zerlegt einen speziellen Spanngraphen des Graphen bzw. die Ebene hierarchisch, bei Graphen mithilfe sogenannter Separatorsätze. Es gibt in diesen Zerlegungen jeweils Grenzknoten oder „Portale" (Punkte in der Ebene) zwischen verschiedenen Teilgraphen bzw. Regionen in der Ebene. Man kann nun dafür sorgen, dass eine Tour die Teilgraphen nur in diesen Grenzknoten bzw. Portalen wechselt, und dabei die Gesamthäufigkeit dieses Wechselns mithilfe sogenannter Patching-Lemmata beschränken, ohne dass die Länge einer optimalen unter den so eingeschränkten Touren (wesentlich) größer wird als die einer optimalen uneingeschränkten Tour. Eine optimale eingeschränkte Tour ist durch dynamische Programmierung in polynomieller Zeit berechenbar. Während diese im euklidischen Fall schon als Lösung genutzt werden kann, muss sie im Fall planarer Graphen noch auf eine zulässige Lösung im Graphen zurückabgebildet werden.

Diese Beschränkung der Anzahl des Wechselns von Teilgraphen ist wesentlich dadurch möglich, dass verschiedene Teiltouren einer Tour, die sich in einem Knoten treffen, in beliebiger Reihenfolge zusammengesetzt werden können, und dass eine Tour, die ein Liniensegment in der Ebene mehr als zweimal schneidet, durch Umordnung von Teiltouren der Tour so verändert werden kann, dass sie dies Segment nur noch zweimal schneidet und dafür Zusatzkosten in der Größenordnung der Länge dieses Segments hat. Dies sind die jeweiligen Aussagen der erwähnten Patching-Lemmata.

Bei den hier betrachteten Transportproblemen hingegen ist die Reihenfolge des Besuchens von Knoten keineswegs egal, schliesslich muss ein Gegenstand erst aufgenommen werden, bevor er an seinem Zielort abgelegt werden kann. Damit ist ein Plan nicht so leicht lokal veränderbar, wie das für die Traveling-Salesman-Touren gilt.

Ein weiteres Problem für die Übertragung auf die betrachteten Transportprobleme ist, dass man für eine Traveling-Salesman-Tour annehmen kann, dass sie sich nicht selbst überschneidet, was man bei Transportproblemen so nicht annehmen kann. Ganz abgesehen von der Schwierigkeit, bei den Transportproblemen einen optimalen Pfad statt einer optimalen Rundtour finden zu müssen.

Arora gibt in [Aro04], S. 15f., auch Kriterien an für die Anwendbarkeit der Techniken für das euklidische TSP auf andere Probleme. Darunter fallen etwa die Möglichkeit, mithilfe kleiner Schnittstellen zwischen angrenzenden Gebieten dortige Teilprobleme unabhängig voneinander zu lösen, und die Anwendbarkeit von Analoga für Patching-Lemmata.

Es scheinen also zumindest die hier erwähnten Techniken nicht direkt für die Lösung von Transportproblemen mit planaren Karten oder euklidischen Distanzen anwendbar zu sein.

3.4.4 Weitere Varianten

Malte Helmert spricht in [Hel01] davon, in parallelen Plänen auch zu erlauben, dass verschiedene Gegenstände von einem Fahrzeug gleichzeitig aufgenommen werden können.

Geht man bei der Ursprungsvariante der Probleme noch weiter, schafft Kosten für das Aufnehmen und Ablegen von Gegenständen ganz ab und lässt dafür *beliebige Kosten für das Durchlaufen von Kanten* zu, so gibt es eine größere Ähnlichkeit zu den in der Informatik für einzelne Fahrzeuge schon betrachteten Problemen (vgl. Abschnitt 2.2.2). Die Härteresultate aus Abschnitt 3.1 gelten dann weiterhin, da die Kosten für Aufnehmen und Ablegen von Gegenständen dabei keine größere Rolle spielen, ob im Fall der Ursprungsvariante oder einer entsprechend abgewandelten Zielfunktion für parallele Pläne. Die Approximationsalgorithmen aus Abschnitt 3.3 für C_1M_1, C_1M_+, $C_\infty M_+$ und $C_\infty M_+$ behalten ihre Laufzeit und ihre jeweilige konstante Approximationsgüte, ebenso bleibt die Approximationsgüte der Algorithmen für C_*M_1 und C_*M_+. Die Variante von C_*M_1 ist dann sogar ein Spezialfall des pCDARP (vgl. Abschnitt 2.2.2) und somit $O(\log(n))$-approximierbar. Nur für die M_*-Varianten der Probleme müsste man neue Algorithmen

entwerfen.
Für diese Variante von C_*M_1 ist sogar angekündigt, es befände sich nicht in APX und sei für kein $\varepsilon > 0$ $\min\{O((\log(n))^{1/4-\varepsilon}), cap^{1-\varepsilon}\}$-approximierbar unter der Voraussetzung, dass nicht alle Probleme in NP durch randomisierte Algorithmen mit erwarteter Laufzeit in $O(n^{polylog(n)})$ entschieden werden können (Gørtz [Gør05]). Es liegt mir aber bisher kein (auch nur vorläufiger) Beweis dafür vor.

Eine weitere Variante entsteht, wenn man analog zum SAS^+-Planen (vgl. Abschnitt 3.5) *gerichtete Karten* zulässt, die aber stark zusammenhängend sein müssen. Dann lassen sich vermutlich stärkere Nichtapproximierbarkeitsresultate für die entsprechenden Varianten von C_*M_+ und C_*M_* beweisen: Wie wir auch in Abschnitt 3.5 sehen werden, lässt sich diese Bedingung (dass die Karten stark zusammenhängend sein müssen) nämlich meist sehr leicht umgehen.
Die Variante von C_*M_+ ist wohl nicht in APX, was man durch eine Reduktion von Set-Cover zeigen könnte. Dazu führt man pro Element einen Gegenstand, ein Fahrzeug der Kapazität 1 und einen Knoten ein, an dem Gegenstand und Fahrzeug starten. Pro Teilmenge gibt es einen Knoten und ein Fahrzeug derjenigen Kapazität, die der Mächtigkeit der Teilmenge entspricht (oder beliebig groß ist). Von jedem Elementknoten gehen (ungewichtete) gerichtete Kanten zu den Teilmengenknoten, von jedem Teilmengenknoten aus lange Wege (z. B. der Länge der Mächtigkeit der Grundmenge) zu einem gemeinsamen Knoten, der Zielort aller Gegenstände ist. Außerdem führen gerichtete Pfade sehr großer Länge von diesem gemeinsamen Zielort zu den Startorten der Gegenstände. Diese sind allerdings so lang, dass es sich nie lohnt, sie zu benutzen. Ist die Set-Cover-Instanz unlösbar (leicht prüfbar in Linearzeit), bildet man sie auf eine Instanz ab, in der ein Gegenstand an einem isolierten Knoten startet und an einen anderen Knoten transportiert werden muss.
Die Variante von C_*M_* ist wohl nicht $2^{\log(n)^{c-\varepsilon}}$-approximierbar für alle $\varepsilon > 0$ und ein $c > 0$ durch Reduktion z. B. vom gerichteten Minimum-Edge-Cost-Flow-Problem (MECF) mit einheitlichen Kantenkosten (vgl. [EKS05], Satz 9.1). Dabei führt man für jede gerichtete Kante der MECF-Instanz einer festen Kapazität eine Kante und ein Fahrzeug dieser Kapazität ein, das im Prinzip nur genau diese Kante durchlaufen kann (pro forma ergänzt durch einen langen Rückweg vom Endpunkt jeder solchen Kante zu ihrem Anfangspunkt). Außerdem führt man f Gegenstände mit der Quelle der MECF-Instanz s als gemeinsamem Startort und der Senke t der MECF-Instanz als gemeinsamem Ziel ein, wenn f der Wert des maximalen s-t-Flusses in der MECF-Instanz ist (f ist in polynomieller Zeit berechenbar).[18]

[18] Um die Kosten für Aufnehmen und Ablegen von Gegenständen pro Kante vernachlässigen zu können, müsste man genauer gesagt pro MECF-Kante statt einer einzelnen Kante einen langen Pfad fester Länge (z. B. f^2) für ein Fahrzeug einführen.

3.5 SAS^+-Planen

3.5.1 Einführung

Man kann sich nun Gedanken darüber machen, wie sich die Schwierigkeit der Transportprobleme ändert, wenn man nur einen einzelnen Gegenstand transportieren muss. Zunächst ist dabei offensichtlich die Kapazität der Fahrzeuge egal. Bei einem einzelnen Gegenstand und einer Menge von Fahrzeugen mit gleicher Karte ist das Transportproblem noch in P, da die optimale Lösung nur ein Fahrzeug benötigt und mit diesem den Gegenstand direkt zu seinem Ziel bringt.[19]

Eine Verallgemeinerung des Problems mit mehreren Fahrzeugen verschiedener Karten stellt das sogenannte SAS^+-1-Planen dar.

SAS^+-Planen stellt eine alternative Formalisierungsmöglichkeit für Planungsaufgaben dar (vgl. etwa Bäckström und Nebel [BN95]). Ich werde mich im Folgenden an der Darstellungsweise von Helmert [Hel04] orientieren.

Definition 3.5.1 (SAS^+-Planungsinstanz) Eine *SAS^+-Planungsinstanz* (SAS^+ task) ist ein 4-Tupel $\Pi = (V, O, s_0, s_*)$ und besteht aus:

- $V = \{v_1, \ldots, v_n\}$, einer Menge von *Zustandsvariablen*, jeweils assoziiert mit einem endlichen Wertebereich D_{v_i}. Eine partielle Funktion $s : V \nrightarrow \bigcup_{v \in V} D_v$ heißt *partielle Variablenbelegung über V*, wenn $s(v) \in D_v$ für alle $v \in dom(s)$. Eine partielle Variablenbelegung über V heißt *Zustand*, wenn sie eine totale Funktion ist.
- O, einer Menge von *Operatoren* $o = \langle pre_o, eff_o \rangle$ mit partiellen Variablenbelegungen pre_o und eff_o über V.
- einem Zustand s_0, genannt *Startzustand*, und
- einer partiellen Variablenbelegung s_*, genannt *Ziel*. □

Hierbei entsprechen die Operatoren den Aktionen einer Planungsinstanz (vgl. Abschnitt 2.1.2). Ein Operator $o = \langle pre_o, eff_o \rangle$ ist immer dann in einem Zustand s anwendbar, wenn $s_{|dom(pre_o)} = pre_o$. Seine Anwendung verändert dann den Wert jeder Variablen v aus $dom(eff_o)$ auf den Wert $eff_o(v)$.

Die Planexistenz für SAS^+ ist $PSPACE$-vollständig (vgl. [BN95]).

Definition 3.5.2 (Kausalgraph) ([Hel04], Def.3) Sei $\Pi = (V, O, s_0, s_*)$ eine SAS^+-Planungsinstanz. Ihr *Kausalgraph* (causal graph) $CG(\Pi)$ ist der gerichtete Graph (V, A) mit $(u, v) \in A :\Longleftrightarrow u \neq v \wedge (\exists \langle pre, eff \rangle \in O : v \in dom(eff)$ und $(u \in dom(pre)$ oder $u \in dom(eff)))$. □

Definition 3.5.3 (SAS^+-1-Planen) ([Hel04], Def.4) Eine *SAS^+-1-Planungsinstanz* ist eine SAS^+-Planungsinstanz Π mit einer ausgezeichneten Variable v, genannt High-Level-Variable, sodass $CG(\Pi) = (V, \{ (u, v) \mid u \neq v, u \in V \})$. Die Variablen aus $V \setminus \{v\}$ heißen Low-Level-Variablen. Das Ziel muss dabei genau für die High-Level-Variable v definiert sein. □

[19] Man probiere also etwa die Fahrzeuge nacheinander aus.

Damit können in einer SAS^+-1-Planungsinstanz insbesondere alle Low-Level-Variablen unabhängig voneinander geändert werden.

Definition 3.5.4 (Domain-Transition-Graph) ([Hel04], Def.5) Sei eine SAS^+-1-Planungsinstanz mit Variablenmenge V gegeben und sei v in V. Der *Domain-Transition-Graph*[20] (kurz DTG) G_v ist der beschriftete gerichtete Multigraph über der Knotenmenge D_v, der genau eine Kante (d, d') enthält für jeden Operator $\langle pre, eff \rangle$ mit ($pre(v) = d$ oder $pre(v) \notin dom(pre)$) und $eff(v) = d'$. Diese Kante ist beschriftet mit $pre_{|V \setminus \{v\}}$. Für jede Kante (d, d'), die mit L beschriftet ist, sprechen wir von einem *Übergang* von v von d zu d' unter der Vorbedingung L. □

PlanEx und PlanLen werden hier analog zu Def. 2.1.17 verwendet.

Satz 3.5.5 *([Hel04], Satz 6) PlanEx-SAS^+-1 ist NP-vollständig, sogar wenn jeder Operator höchstens eine Vorbedingung an Low-Level-Variablen hat.* □

3.5.2 SAS^+-1sc

Sei nun SAS^+-1sc die Einschränkung von SAS^+-1 auf Instanzen, deren DTGs stark zusammenhängend sind. Diesen Spezialfall können wir als Verallgemeinerung des Falles mehrerer Fahrzeuge verschiedener Karten mit insgesamt einem einzelnen Gegenstand verstehen. Dabei kann man sich eine grobe Korrespondenz zwischen der High-Level-Variablen und dem einzelnen Gegenstand vorstellen sowie zwischen den Low-Level-Variablen und den Fahrzeugen. Die Domain-Transition-Graphen der Low-Level-Variablen würden dann den Karten der Fahrzeuge entsprechen.

Satz 3.5.6 *([Hel04]) PlanEx-SAS^+-1sc ist in P.*

BEWEIS Jedes Tupel von Werten von Low-Level-Variablen kann angenommen werden, da man die Werte der Low-Level-Variablen jeweils ohne Rücksicht auf und ohne Effekte auf andere Variablen ändern kann und die DTGs stark zusammenhängend sind. Damit kann die High-Level-Variable jeden Wert annehmen, der in ihrem DTG durch Kanten erreichbar ist, was wegen des starken Zusammenhangs wiederum jeder ist. Die Antwort lautet also, dass immer ein Plan existiert, wenn es sich um eine gültige Kodierung einer SAS^+-1sc-Instanz handelt. ■

Optimale Pläne haben dabei offensichtlich polynomielle Länge (vgl. auch [Hel04]), und es ist auch in polynomieller Zeit ein Plan polynomieller Länge konstruierbar. Außerdem gilt:

Satz 3.5.7 *([Hel04], Satz 7) PlanLen-SAS^+-1sc ist NP-vollständig.* □

Damit macht es in diesem Fall wieder einmal Sinn, die Approximierbarkeit zu untersuchen.

[20] Da sich „Wertebereichsübergangsgraph" für mich zu sperrig anhörte, bleibe ich bei der englischen Bezeichnungsweise.

Satz 3.5.8 *SetCover* $\leq_{AP}$ *SAS^+-1sc.*

Beweis Die folgende Reduktion ist sehr ähnlich der in [Hel04], Thm. 7, von $X3C$ auf PlanLen-SAS^+-1 gegebenen.
Sei eine SetCover-Instanz $(U, \{S_1, \ldots, S_k\})$ gegeben mit $U = \{e_1, \ldots, e_n\}$. Formal bilden wir sie zunächst auf folgende SAS^+-1-Instanz $\Pi = \langle V, O, s_0, s_* \rangle$ mit ausgezeichneter Variable m_U ab (die DTGs werden unten noch stark zusammenhängend gemacht):

$$
\begin{aligned}
&V = \{m_{S_1}, \ldots, m_{S_k}, m_U\} \text{ mit den Wertebereichen } D_{m_U} = D_{m_{S_i}} = \{0, \ldots, n\} \\
&O = \\
&\quad \{\, \langle \{m_U \mapsto i, m_{S_j} \mapsto n\}, \{m_U \mapsto i+1\} \rangle \,|\, i \in \{0, \ldots, n-1\}, j \in \{1, \ldots, k\}, e_i \in S_j \,\} \\
&\quad \cup \{\, \langle \{m_{S_j} \mapsto i\}, \{m_{S_j} \mapsto i+1\} \rangle \,|\, i \in \{0, \ldots, n-1\}, j \in \{1, \ldots, k\} \,\} \\
&s_0 = \{m_U \mapsto 0\} \cup \{\, m_{S_i} \mapsto 0 \,|\, i \in \{1, \ldots, r\} \,\} \\
&s_* = \{m_U \mapsto n\}
\end{aligned}
$$

Etwas anschaulicher: Wir haben eine Variable für die Gesamtmenge, die High-Level-Variable m_U, und für jede der Teilmengen eine weitere Variable. Die High-Level-Variable soll ihren Wert von 0 auf n ändern und kann dies nur durch Hochzählen in Schritten von 1. Dabei kann ein solcher Übergang von $i-1$ nach i erst dann genommen werden, wenn „das i-te Element überdeckt ist“, soll heißen: Wenn eine Variable den Wert n hat, die zu einer das i-te Element enthaltenden Teilmenge korrespondiert. Bei den Variablen der Teilmengen ist ebenfalls nur schrittweises Erhöhen von 0 aus möglich. Das Erhöhen pro Teilmenge von 0 auf n korrespondiert zum Wählen der Teilmenge.

Offensichtlich würde es keinen Sinn für einen Plan machen, den Wert einer Variablen wieder zu erniedrigen, daher können wir entsprechende Aktionen (Erniedrigen jeder Variablen um 1 mit denselben Vorbedingungen wie beim Erhöhen) zur obigen Instanz hinzunehmen und erhalten stark zusammenhängende Domain-Transition-Graphen (die sogar zu jeder Kante eine Rückkante mit denselben sonstigen Vorbedingungen haben).

Die Plan-Instanz hat $2((\Sigma_{i=1}^r |S_i|) + nk)$ Operatoren, die Wertebereiche der $k+1$ Variablen haben jeweils die Größe $n+1$. Damit ist die Größe der transformierten Instanz polynomiell in der Größe der Set-Cover-Instanz.

Nun gilt offensichtlich: Es gibt ein Set Cover der Mächtigkeit höchstens r genau dann, wenn es einen zulässigen Plan der Länge höchstens $rn + n = (r+1)n$ gibt („Anlauf“ der Mengenvariablen plus Übergänge der High-Level-Variablen). Die Hin- und Zurückübersetzungen sollten nach den bisherigen Bemerkungen ebenso offensichtlich sein.

Wenn also die Approximationsgüte des Plans $\frac{(r+1)n}{(r^*+1)n} = \frac{r+1}{r^*+1} = 1 + \frac{r-r^*}{r^*+1}$ ist, so ist die Approximationsgüte der korrespondierenden SetCover-Lösung $\frac{r}{r^*} = 1 + \frac{r-r^*}{r^*} = 1 + \frac{r^*+1}{r^*}\frac{r-r^*}{r^*+1} \leq 1 + 2\frac{r-r^*}{r^*+1}$ (da $1 \leq r^*$). ■

Korollar 3.5.9 *SAS^+-1sc* $\notin$ *APX.* □

3.5.3 SAS^+-bipsc

Man kann das auf stark zusammenhängende DTGs eingeschränkte SAS^+-1-Planen noch verallgemeinern, indem man mehrere High-Level-Variablen zulässt und somit einen bipartiten Kausalgraphen.

Definition 3.5.10 (SAS^+-bipsc) Eine *SAS^+-bipsc-Instanz* ist eine SAS^+-Instanz Π mit stark zusammenhängenden DTGs und bipartitem Kausalgraph $CG(\Pi) = (V, A)$. Damit ist gemeint, dass es eine Zerlegung von V in V_1 und V_2 gibt, sodass für jede Kante $(u, v) \in A$ gilt, dass $u \in V_1$ und $v \in V_2$ ist. Die Variablen aus V_1 heißen *Low-Level-Variablen*, die aus V_2 *High-Level-Variablen*.[21] Das Ziel sei genau für die High-Level-Variablen definiert. □

Die Planexistenz hierfür ist immer noch in polynomieller Zeit entscheidbar, da man etwa die High-Level-Variablen nacheinander betrachten kann. Klar ist auch, dass weiterhin minimale Pläne polynomielle Länge haben. Damit folgt:

Lemma 3.5.11 *PlanEx-SAS^+-bipsc ist in P.* □

Lemma 3.5.12 *PlanLen-SAS^+-bipsc ist NP-vollständig.*

Beweis Es handelt sich um eine Verallgemeinerung von PlanLen-SAS^+-1sc und ist daher NP-hart. ■

Es macht also Sinn, sich Gedanken über die Approximierbarkeit von SAS^+-bipsc zu machen.

Natürlich ist auch dieses Problem als Verallgemeinerung des eben betrachteten nicht in APX, es würde sogar dieselbe Reduktion funktionieren, mit der das für $C_\infty M_+$ gezeigt wird (s. Satz 3.1.5). Das zeigt die Verwandtschaft der Probleme.

Man kann nämlich *$C_\infty M_+$ als Spezialfall von SAS^+-bipsc* auffassen. Hierbei führt man pro Fahrzeug m eine Low-Level-Variable ein, deren Werte gerade die möglichen Positionen des Fahrzeugs sind, und pro Gegenstand eine High-Level-Variable, deren Werte die möglichen Aufenthaltsorte des Gegenstands sind ($V \dot\cup M$). Ein Gegenstand kann von einem Fahrzeug aufgenommen werden, wenn sich beide am selben Ort befinden, und an einem Ort v von einem Fahrzeug m abgelegt werden, wenn er sich in m und m sich am Ort v befindet. Die DTGs der Fahrzeuge entsprechen so im Wesentlichen den Karten der Fahrzeuge und haben sogar „ungerichtete" Kanten, d.h. für jede Kante gibt es auch eine in Rückrichtung.

Für diese Verallgemeinerung kann man aber auch eine stärkere untere Approximierbarkeitsschranke zeigen. Dazu reduzieren wir zunächst das *$LabelCover_{min}$*-Problem darauf:

Definition 3.5.13 (*$LabelCover_{min}$*) (aus [AL96]) Das *$LabelCover_{min}$*-Problem (auch kurz *LC_{min}*) besteht aus folgenden Komponenten:

Die *Probleminstanzen* $L = (G, N, \Pi)$ bestehen jeweils aus einem regulären, bipartiten Graphen $G = (V_1 \dot\cup V_2, E)$[22], einer natürlichen Zahl N, die die Menge der sogenannten *Labels* $\{1, \ldots, N\}$ definiert, und der Menge $\Pi = \{\, \Pi_e \mid e \in E \,\}$, die für jede Kante $e \in E$

[21] O.B.d.A. gebe es im Kausalgraphen keine isolierten Knoten. Die zu solchen Knoten korrespondierenden Variablen könnte man nämlich unabhängig von allen anderen Variablen verändern und die für das Erreichen eines bestimmten Wertes nötige Anzahl an Aktionen per einfacher Kürzeste-Wege-Suche optimal bestimmen.

[22] Regulär bedeutet in diesem Fall: Alle Knoten aus V_1 haben den gleichen Grad d_1 und alle Knoten aus V_2 den gleichen Grad d_2.

eine partielle Funktion $\Pi_e : \{1, \ldots, N\} \nrightarrow \{1, \ldots, N\}$ enthält mit: $\forall e \, \exists b \in \{1, \ldots, N\} : \Pi_e(b) = 1$. Eine *Beschriftung* (labelling) $l : V_1 \cup V_2 \rightarrow Pot(\{1, \ldots, N\}) \setminus \{\emptyset\}$ assoziiert eine nichtleere Menge von Labels mit jedem Knoten $v \in V_1 \cup V_2$ und *überdeckt* eine Kante (v_1, v_2) mit $v_1 \in V_1$ und $v_2 \in V_2$ genau dann, wenn $\forall a_2 \in l(v_2) \, \exists a_1 : a_1 \in l(v_1) \wedge \Pi_e(a_1) = a_2$.
Die *Menge der zulässigen Lösungen* zu einer Instanz L besteht aus den Beschriftungen, die *alle* Kanten überdecken.
Die *Kosten* einer Lösung sind $\frac{\Sigma_{v \in V_1} |l(v)|}{|V_1|}$.
Es handelt sich um ein Minimierungsproblem. □

Es sei darauf hingewiesen, dass es für jede Instanz des $LabelCover_{min}$-Problems eine Lösung mit den Kosten N gibt: Man beschrifte jeden V_1-Knoten mit $\{1, \ldots, N\}$ und jeden V_2-Knoten mit $\{1\}$.

Lemma 3.5.14 *$LabelCover_{min} \leq_{AP} SAS^+$-bipsc.*

Beweis Sei eine $LabelCover_{min}$-Instanz gegeben durch $G = (V_1 \dot{\cup} V_2, E)$ bipartit ($E \subseteq V_1 \times V_2$)[23] und regulär mit d_1 als Grad der V_1-Knoten und d_2 als Grad der V_2-Knoten, $N \in \mathbb{N}$ sowie $\forall e \in E : \Pi_e : \{1, \ldots, N\} \nrightarrow \{1, \ldots, N\}$ partiell mit $\forall e \, \exists b : \Pi_e(b) = 1$. O.B.d.A. sei $E \neq \emptyset$.
Wir werden im Folgenden die Knoten von V_1 auch als „linke" Knoten und die von V_2 als „rechte" Knoten bezeichnen.

Diese Instanz bilden wir nun wie folgt auf eine SAS-Instanz ab:

Wir führen kurz gesagt für jeden Knoten der LC_{min}-Instanz mehrere Low-Level-Variablen ein und für jede Kante eine High-Level-Variable. Ziel ist es ja bei LC_{min}, alle Kanten mit möglichst wenig für V_1-Knoten gewählten Labels zu überdecken. Dies Überdecken einer Kante wird nun zum Erreichen des Zielwerts der entsprechenden High-Level-Variablen korrespondieren.

Wir werden der Übersichtlichkeit halber zunächst die Domain-Transition-Graphen nicht als stark zusammenhängend angeben und später nur pro forma „Rückwege" in diese einfügen, deren Benutzung zu Plänen mit sehr hohen Kosten führen wird.

Zuerst zu den *Low-Level-Variablen*:

Es gibt für jeden linken Knoten N Variablen, insgesamt die Variablen $V_1 \times \{1, \ldots, N\}$ mit den Wertebereichen $\{0, \ldots, l\}$ (der Wert von l wird unten festgelegt) und den Operatoren $\{ \langle \{(v, i) \mapsto j\}, \{(v, i) \mapsto j + 1\} \rangle \mid j \in \{0, \ldots, l - 1\} \}$.

Für jeden rechten Knoten gibt es wiederum N Variablen, insgesamt die Variablen $V_2 \times \{1, \ldots, N\}$ mit den Wertebereichen $\{unent, gew, ngew\}$ (für „unentschieden", „gewählt" und „nicht gewählt") und den Operatoren $\{ \langle \{(v, i) \mapsto unent\}, \{(v, i) \mapsto gew\} \rangle , \langle \{(v, i) \mapsto unent\}, \{(v, i) \mapsto ngew\} \rangle \}$.

Nun zu den *High-Level-Variablen*:

Für jede Kante eine Variable, insgesamt die Variablen E mit den Wertebereichen $\{nl, le\} \times \{1, \ldots, N\}$ (nl für „nichtleer" und le für „leer", bezogen auf die bisher gewählte Menge von Labels pro Knoten) und den Operatoren $\{ \langle \{e \mapsto (le, i), (v, i) \mapsto ngew\}, \{e \mapsto$

[23] Wir nehmen die Kanten hier als derart gerichtet an.

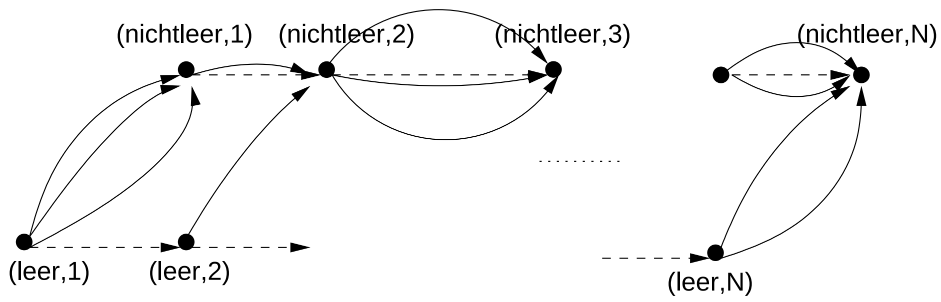

Abbildung 3.9: Ein Domain-Transition-Graph einer High-Level-Variablen in der Reduktion von Lemma 3.5.14. Hierbei sind Operatoren des Typ 1 durch gestrichelte Pfeile dargestellt, Operatoren des Typ 2 durch normale.

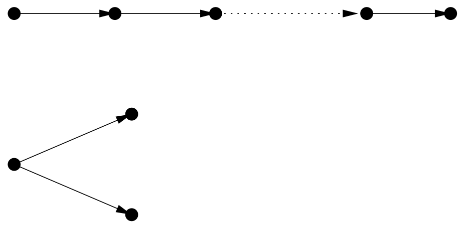

Abbildung 3.10: Die Domain-Transition-Graphen der Low-Level-Variablen.

$(le, i+1)\}\rangle \,|\, v \in V_2,\ i \in \{1, \ldots, N-1\}\, \exists v_1 \in V_1\colon e = (v_1, v_2)\}$ und $\{\, \langle \{e \mapsto (nl, i), (v, i+1) \mapsto ngew\}, \{e \mapsto (nl, i+1)\}\rangle \,|\, v \in V_2,\, i \in \{1, \ldots, N-1\}\, \exists v_1 \in V_1\colon e = (v_1, v_2)\}$ (Typ-1-Operatoren) sowie $\{\, \langle \{e \mapsto (le, i), (v_1, j) \mapsto l, (v_2, i) \mapsto gew\}, \{e \mapsto (nl, i)\}\rangle \,|\, \Pi_e(j) = i,\ e = (v_1, v_2),\ i, j \in \{1, \ldots, N\}\,\}$ und $\{\, \langle \{e \mapsto (nl, i), (v_1, j) \mapsto l, (v_2, i) \mapsto gew\}, \{e \mapsto (nl, i+1)\}\rangle \,|\, \Pi_e(j) = i,\ e = (v_1, v_2),\ i, j \in \{1, \ldots, N\}\,\}$ (Typ-2-Operatoren).
Hier unterscheiden wir (übertragen auf die LC_{min}-Instanz) danach, ob schon ein Label für den rechten Knoten der korrespondierenden Kante gewählt wurde oder nicht. Außerdem gehen wir davon aus, dass jede Entscheidung für oder gegen ein Label eines rechten Knotens endgültig ist und lassen so ein Erreichen von (nl, N) nur dann zu, wenn diese Kante überdeckt wurde.

Eine grobe Illustration eines möglichen Domain-Transition-Graphs einer High-Level-Variablen ist in Abb. 3.9 angegeben, die Domain-Transition-Graphen von Low-Level-Variablen in Abb. 3.10, ein Kausalgraph in Abb. 3.11.

Der *Anfangszustand* ist gegeben durch
$s_0 = \{\, v \mapsto 0 \,|\, v \in V_1 \,\} \cup \{\, v \mapsto unent \,|\, v \in V_2 \,\} \cup \{\, e \mapsto (le, 1) \,|\, e \in E \,\}$.

Der partielle *Zielzustand* ist $s_* = \{\, e \mapsto (nl, N) \,|\, e \in E \,\}$.

Die *Kosten einer zulässigen Lösung* (noch ohne Rückwege) lassen sich nun wie folgt abschätzen:

Jede Kantenvariable wird N-mal ihren Wert ändern, bevor sie ihren Zielwert erreicht hat. Das bedeutet Gesamtkosten von $N|E| = N|V_1|d_1 = N|V_2|d_2$. Dazu müssen die V_2-

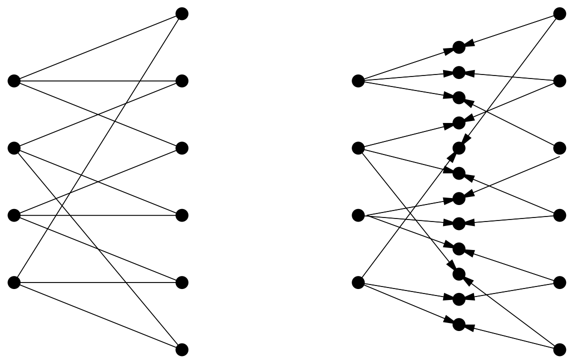

Abbildung 3.11: Links der Graph einer Labelcover-Instanz, rechts der Kausalgraph der entsprechend konstruierten SAS-Instanz.

Variablen auf „gewählt“ oder „nicht gewählt“ festgelegt werden, was dem Bezeichnen der entsprechenden Knoten in der LC-Instanz entspricht. Dies bedeutet für diese Variablen Kosten $|V_2|N$.

Also hat man bisher feste Kosten von $N|V_2|(d_2 + 1)$, die unabhängig vom jeweiligen Plan anfallen.

Außerdem müssen die V_1-Variablen gewählt oder nicht gewählt werden, was dem Bezeichnen der V_1-Knoten entspricht und insgesamt Kosten von xl verursacht, wenn x die Anzahl der gewählten Variablen (also der verteilten Labels) bezeichnet,[24] l sind ja die Kosten für das „Wählen“ einer Variable. Dabei gilt $|V_1|l \leq xl \leq |V_1|Nl$.

Wählen wir nun $l := N|V_2|(d_2 + 1)$, so sind die Gesamtkosten einer Lösung gerade $N|V_2|(d_2+1)+xN|V_2|(d_2+1) = (x+1)l$. Nun gilt ja $|V_2|d_2 = |E|$, also $|V_2|d_2+|V_2| \leq n$ (wobei wir mit n die Gesamtgröße der LC_{min}-Instanz bezeichnen), also gilt $l = N|V_2|(d_2+1) \leq Nn$. Wir können außerdem abschätzen: $x \leq |V_1|N$, also $x + 1 \leq |V_1|N + 1 \leq Nn$.

Die Kosten jeder solchen Lösung sind also durch N^2n^2 beschränkt.

Jetzt kommen wir endlich zu dem Punkt, an dem wir die Domain-Transition-Graphen zusammenhängend machen: Wir hängen Rückwege der Länge N^3n^2 an die jeweiligen „Endpunkte“ an (also an Knoten der Form l, gew, $ngew$ oder (nl, N) jeweils zum Knoten, der zum Wert der Variablen im Anfangszustand korrespondiert; dabei hängen wir für eine V_2-Variable denselben Rückweg an gew und $ngew$). Dann hat jede Lösung, die diese Rückwege benutzt, Kosten, die wenigstens um den Faktor N von denen einer optimalen Lösung entfernt sind.

Die Gesamtgröße der konstruierten SAS-Instanz ist nun

$$\begin{aligned} &\leq |V|N + |E| \text{ (die Variablen)} \\ &\quad +(4N + N^3n^2)|E| \text{ (Operatoren für die High-Level-Variablen)} \\ &\quad +|V_1|N(N^3n^2 + Nn) \end{aligned}$$

[24] Wir können annehmen, dass keine V_1-Variable nur „teilweise“ gewählt wird, ansonsten streichen wir die entsprechenden Aktionen und erhalten einen Plan geringerer Kosten, der zulässig bleibt.

$+|V_2|N(2 + N^3 n^2)$ (Operatoren für die Low-Level-Variablen)
$\leq 7N^4 n^3 \leq 7n^7$

Wir können nämlich davon ausgehen, dass $n \geq N$, da ja z. B. die Π_e auch Teil der Eingabe sind; außerdem gilt $N < |V|$ in den im Härtebeweis für *Labelcover* in [AL96] konstruierten Instanzen.

Nun zur Rückübersetzung eines Plans:
Wenn Rückwege benutzt werden, so ist die Lösung wenigstens um den Faktor N schlechter als die optimale Lösung, und wir bilden sie auf die triviale LC_{min}-Lösung ab, die an den V_2-Knoten überall die 1 und an den V_1-Knoten jeweils alle möglichen Labels auswählt. Der Wert dieser Lösung ist ebenfalls nur um den Faktor N schlechter als der der optimalen LC_{min}-Lösung.

Werden hingegen keine Rückwege benutzt, so wählen wir die Labels entsprechend (nichtleere Labelmengen pro Knoten haben wir durch die DTGs der High-Level-Variablen erzwungen).

Wenn dann die Güte der SAS-Lösung $\frac{(x+1)l}{(x^*+1)l} = \frac{x+1}{x^*+1} = 1 + \frac{x-x^*}{x^*+1}$ beträgt, so beträgt die der rückübersetzten LC_{min}-Lösung gerade $\frac{x}{x^*} = 1 + \frac{x-x^*}{x^*} = 1 + 2\frac{x-x^*}{2x^*} \leq 1 + 2\frac{x-x^*}{x^*+1}$.■

Daraus erhalten wir:

Satz 3.5.15 *Unter der Voraussetzung $NP \not\subseteq DTIME(n^{polylog(n)})$ ist SAS^+-bipsc nicht $\frac{1}{2}2^{(-\frac{1}{7}log7+\frac{1}{7}logn)^{1-\gamma}}$-approximierbar* $(\forall\gamma > 0)$.

Beweis Angenommen, SAS^+-bipsc ist (für ein monoton steigendes f) $f(n)$-approximierbar. Dann ist LC_{min} $2f(7n^7)$-approximierbar. Denn wir können dann ja eine LC_{min}-Instanz der Größe n mit Lemma 3.5.14 auf eine SAS^+-bipsc-Instanz der Größe $m \leq 7n^7$ abbilden und diese $f(m)$-approximieren mit $f(m) \leq f(7n^7)$. Diese Lösung können wir mit einem erneuten Verlust eines Faktors 2 bezogen auf den Approximationsfaktor zurückabbilden (ebenfalls mit dem Beweis von Lemma 3.5.14).
Da wir aber wissen, dass LC_{min} nicht $2^{(logn)^{1-\gamma}}$-approximierbar ist (unter der Voraussetzung $NP \not\subseteq DTIME(n^{polylog(n)})$, vgl. [AL96]), also auch nicht $2^{(log(7^{-\frac{1}{7}}(7n^7)^{\frac{1}{7}}))^{1-\gamma}}$-approximierbar, ist also SAS^+-bipsc nicht $\frac{1}{2}2^{(log(7^{-\frac{1}{7}}n^{\frac{1}{7}}))^{1-\gamma}}$-approximierbar. Dabei gilt $\frac{1}{2}2^{(log(7^{-\frac{1}{7}}n^{\frac{1}{7}}))^{1-\gamma}} = \frac{1}{2}2^{(log(7^{-\frac{1}{7}})+log(n^{\frac{1}{7}}))^{1-\gamma}} = \frac{1}{2}2^{(-\frac{1}{7}log7+\frac{1}{7}logn)^{1-\gamma}}$. ■

Als Beispiel ergibt dieser Term für $\gamma = 0$ den Wert $\frac{1}{2}\frac{1}{7^{\frac{1}{7}}}n^{\frac{1}{7}}$. Das ist ca. $0.38n^{\frac{1}{7}}$.
Ich vermute, dass dies Resultat durch eine Reduktion des verwandten Min-Rep-Problems (aus [Kor01]) statt des $Labelcover_{min}$-Problems noch leicht verbessert werden könnte.

4 Zusammenfassung und Ausblick

4.1 Zusammenfassung

Zuerst habe ich untere Schranken für die Approximierbarkeit der Transportprobleme von M. Helmert (vgl. Def. 2.1.21) gezeigt: Alle Probleme sind APX-hart, d.h. insbesondere besitzt keines ein PTAS.[1] Lässt man verschiedene Fahrzeugkarten zu, so ist keines der Probleme noch konstant-Faktor-approximierbar; gezeigt wurden untere Schranken von ungefähr $\ln|P|$ und $\ln|M|$ (wobei $|P|$ für die Anzahl der Gegenstände und $|M|$ für die Anzahl der Fahrzeuge steht).[2]

Danach habe ich untersucht, wie sehr sich die Länge optimaler Pläne ändert, wenn Gegenstände nicht zwischengelagert werden dürfen, also nur noch nichtpräemptive Pläne erlaubt sind. Dabei wurde gezeigt, dass optimale Lösungen dieser eingeschränkten (d.h. nichtpräemptiven) Probleme in den Fällen C_1M_+ und $C_\infty M_+$ nur um einen konstanten Faktor (im wesentlichen um 2) schlechter sein können als optimale Lösungen der (präemptiven) Originalprobleme. Im Fall C_*M_1 kann das Verhältnis allerdings nicht mehr durch eine Konstante beschränkt werden (was etwa das Entwickeln von Approximationsalgorithmen erschwert).

Dann habe ich Approximationsalgorithmen angegeben. Für C_1M_1, $C_1M_+, C_\infty M_1$ und $C_\infty M_+$ gibt es Konstant-Faktor-Approximationen, woraus folgt, dass diese Probleme APX-vollständig sind. Für die anderen Probleme wurden keine mit den unteren Schranken übereinstimmenden Algorithmen angegeben, insbesondere bei C_*M_1 und C_*M_+ bleiben große Lücken. Bezüglich der in Abschnitt 2.1.1 definierten Klassen ist mit diesen Ergebnissen keine eindeutige Zuordnung möglich: C_*M_1 und C_*M_+ sind in poly-APX\PTAS, C_1M_*, $C_\infty M_*$ und C_*M_* in poly-APX\APX. Die hier gezeigten unteren und oberen Schranken sind auch in Abb. 4.1 zusammengefasst.

Danach wurden noch abgewandelte Problemstellungen kurz vorgestellt und diskutiert wie etwa die Verwendung anderer Kostenmaße (parallele Pläne, beliebige Kantenkosten) oder Einschränkungen auf bestimmte Graphentypen (planar, vollständig).

Schließlich habe ich untere Schranken für die Approximierbarkeit von Spezialfällen des sogenannten SAS^+-Planens angegeben, die Ähnlichkeiten zu $C_\infty M_*$ aufweisen.

4.2 Ausblick

Es bleiben offene Fragen, etwa die nach der Approximierbarkeit von C_*M_1 und C_*M_+. Die Lücken zwischen den von mir gezeigten unteren Schranken (APX-hart) und oberen

[1] Unter der Voraussetzung P $\neq$ NP.

[2] Unter der Voraussetzung NP $\not\subseteq DTIME(n^{\log\log n})$.

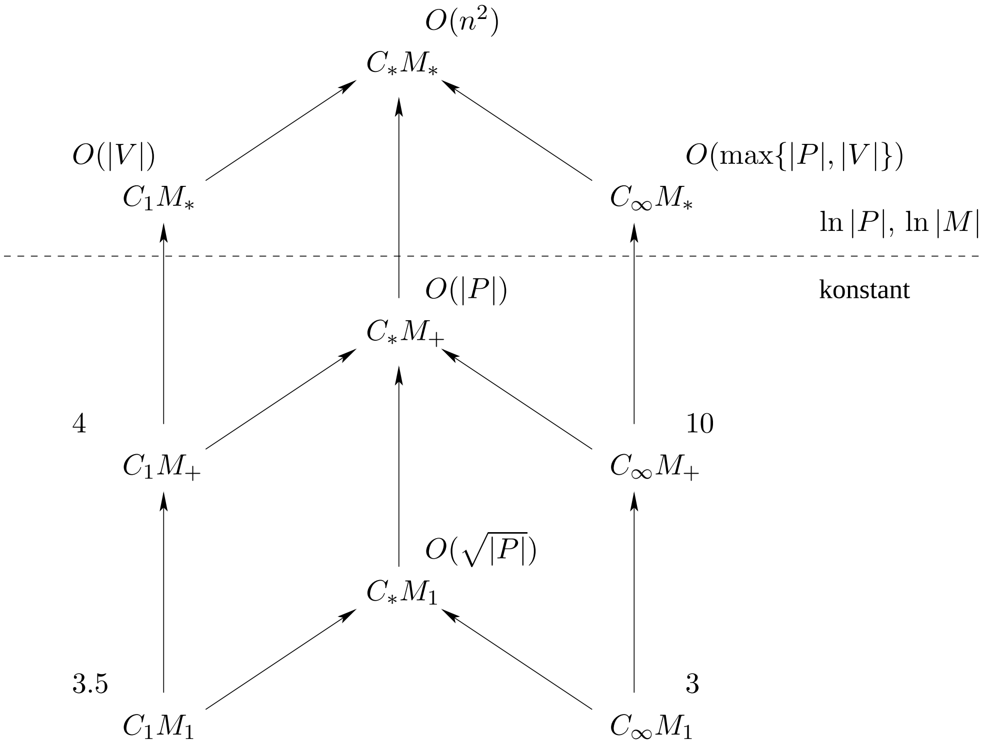

Abbildung 4.1: Die Spezialisierungsbeziehungen zwischen den Transport-F_∞-C_iM_j-Domänen, übernommen aus Abschnitt 2.1.3. Neben den Namen der Domänen die Approximationsgüte der entwickelten Approximationsalgorithmen aus Abschnitt 3.3, rechts die unteren Schranken für die Approximationsgüte aus Abschnitt 3.1.

Schranken ($O(\sqrt{|P|})$- bzw. $O(|P|)$-approximierbar) sind nämlich sehr groß. Vielleicht können sowohl bessere untere Schranken gezeigt (vgl. dazu auch Abschnitt 3.4.4) als auch bessere Approximationsalgorithmen entwickelt werden.

Außerdem denke ich, dass allgemein für die C_*- und M_*-Fälle vermutlich bessere Approximationsalgorithmen erstellt werden können. Die hier analysierten sind recht simpel.

Denkbar wäre auch eine stärkere untere Schranke für C_*M_*.

Ebenfalls interessant scheint mir die Entwicklung fortgeschrittener Approximationsalgorithmen für abgewandelte Fälle (vgl. Abschnitt 3.4) wie die beliebiger Kantenkosten (insbesondere in den M_*-Fällen) und die paralleler Pläne.

Literaturverzeichnis

[AG90] Altinkemer, K.; Gavish, B.: Heuristics for Delivery Problems with Constant Error Guarantees. In: *Transportation Science* 24 (1990), S. 294-297

[AB99] Anily, Shoshana; Bramel, Julien: Approximation Algorithms for the Capacitated Traveling Salesman Problem with Pickups and Deliveries. In: *Naval Research Logistics* 46 (1999), S. 654-670

[AH92] Anily, Shoshana; Hassin, R.: The Swapping Problem. In: *Networks* 22 (1992), S. 419-433

[AHL04] Arkin, E.M.; Hassin, R.; Levin, A.: Approximations for Minimum and Minmax Vehicle Routing Problems. Manuskript 2004, erhältlich unter der URL: http://www.ams.sunysb.edu/~estie/papers/vrp.ps.gz

[Aro98] Arora, Sanjeev: Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems. *Journal of the ACM* 45 (1998), 5, S. 753-782

[Aro04] Arora, Sanjeev: Approximation Schemes for NP-hard Geometric Optimization Problems: A Survey. In: *Mathematical Programming* 97 (2003), 1-2, S. 43-69

[AGK+98] Arora, Sanjeev; Grigni, Michelangelo; Karger, David; Klein, Philipp; Woloszyn, Andrzej: A Polynomial-Time Approximation Scheme for Weighted Planar Graph TSP. In: *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms* (1998), S. 33-41

[ALM+98] Arora, S.; Lund, C.; Motwani, R.; Sudan, M.; Szegedy, M.: Proof Verification and Hardness of Approximation Algorithms. In: *Journal of the ACM* 45 (1998), 3, S. 501-555

[AL96] Arora, Sanjeev; Lund, Carsten: Hardness of Approximations. In: [Hoc96]

[Asa00] Asano, T.; Iwama, K.; Takada, H.; Yanashita, Y.: Designing High-Quality Approximation Algorithms for Combinatorial Optimization Problems. In: *IEICE Transactions on Information and Systems* E83-D (2000), 3, S. 462-478

[AG95] Assad, A.A.; Golden, B.L.: *Arc Routing Methods and Applications.* In: [BMM+95]

[AK88] Atallah, M.J.; Kosaraju, S.R.: Efficient Solutions to Some Transportation Problems with Applications to Minimizing Robot Arm Travel. In: *SIAM Journal on Computing* 17 (1988), 5, S. 849-869

[ACG+99] Ausiello, G.; Crescenzi, P.; Gambosi, G.; Kann, V.; Marchetti-Spaccamela, A.; Protasi, M.: *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Berlin: Springer, 1999 [2. korr. Druck 2003]

[BN95] Bäckström, C.; Nebel, B.: Complexity Results for SAS+-Planning. In: *Computational Intelligence* 11 (1995), 4, S. 640-645

[Bak94] Baker, Brenda S.: Approximation Algorithms for NP-complete Problems on Planar Graphs. In: *Journal of the ACM* 41 (1994), 1, S. 153-180

[BMM+95] Ball, M.O.; Magnanti, T.L.; Monma, C.L.; Nemhauser, G.L.: *Handbooks in Operations Research and Management Science: Volume 8: Network Routing.* Amsterdam: Elsevier, 1995

[BP89] Bern, M.; Plassmann, P.: The Steiner Problem with Edge Lengths 1 and 2. In: *Information Processing Letters* 32 (1989), S. 171-176

[CM99] Chalasani, Prasad; Motwani, Rajeev: Approximating Capacitated Routing and Delivery Problems. In: *SIAM Journal of Computing* 28 (1999), 6, S. 2133-2149

[CKR01] Charikar, Moses; Khuller, Samir; Raghavachari, Balaji: Algorithms for Capacitated Vehicle Routing. In: *SIAM Journal of Computing* 31 (2001), 3, S. 665-682

[CR98] Charikar, Moses; Raghavachari, Balaji: The Finite Capacity Dial-a-Ride Problem. In: *Proc. of the 38th IEEE Symposium on Foundations of Computer Science* (1998), S. 458-467

[EK04] Engebretsen, Lars; Karpinski, Marek: TSP with Bounded Metrics. Approximation Hardness for the (1,2) Case. 2004. Manuskript erhältlich unter der URL: http://www.nada.kth.se/~enge/papers/12-TSP.pdf

[EGK+03] Even, G.; Garg, N.; Könemann, J.; Ravi, R.; Sinha, A.: Min-Max Tree Covers of Graphs. In: *Operations Research Letters* 32 (2004), 4, S. 309-315

[EKS05] Even, G.; Kortsarz, G.; Slany, W.: On Network Design Problems: Fixed Cost Flows and the Covering Steiner Problem. In: *ACM Transactions on Algorithms* 1 (2005), 1, S. 74-101

[FRT03] Fakcharoenphol, Jittat ; Rao, Satish; Talwar, Kunal: A Tight Bound on Approximating Arbitrary Metrics by Tree Metrics. In: *Proc. of the 35th ACM Symposium on Theory of Computing* (2003), S. 448-455

[FS95] Federgruen, A.; Simchi-Levi, D.: Analysis of Vehicle Routing and Inventory Routing Problems. In: [BMM+95]

[Fei98] Feige, Uriel: A threshold of ln n for approximating set cover. In: *Journal of the ACM* 45 (1998), 4, S. 634-652

[FG92] Frederickson, Greg N.; Guan, D.J.: Preemptive Ensemble Motion Planning on a Tree. In: *SIAM Journal of Computing* 21 (1992), 6, S. 1130-1152

[FG93] Frederickson, Greg N.; Guan, D.J.: Nonpreemptive Ensemble Motion Planning on a Tree. In: *ACM Journal of Algorithms* 15 (1993), 1, S. 29-60

[FHK78] Frederickson, Greg N.; Hecht, Matthew S.; Kim, Chul E.: Approximation algorithms for some routing problems. In: *SIAM Journal of Computing* 7 (1978), 2, S. 178-193

[GJ79] Garey, M.R.; Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: W.H. Freeman, 1979

[GW96] Goemans, Michel X.; Williamson, David P.: The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems. In: [Hoc96]

[Gør05] Gørtz, Inge Li: Kommunikation per E-Mail im Dezember 2005 über den auf ihrer Homepage (unter http://www.imada.sdu.dk/~inge/publ.htm) angekündigten Artikel „Hardness of Preemptive Finite Capacity Dial-a-Ride“.

[GKP95] Grigni, M.; Koutsoupias, E.; Papadimitriou, C.H.: An Approximation Scheme for Planar Graph TSP. In: *Proc. of the 36th IEEE Symposium on Foundations of Computer Science* (1995), S. 640-645

[Gua98] Guan, D.J.: Routing a Vehicle of Capacity Greater than One. In: *Discrete Applied Mathematics* 81 (1998), 1, S. 41-57

[GHK+00] Guttmann-Beck, N.; Hassin, R.; Khuller, S.; Raghavachari, B.: Approximation Algorithms with Bounded Performance Guarantees for the Clustered Traveling Salesman Problem. In: *Algorithmica* 28 (2000), S. 422-437

[Hel01] Helmert, Malte: On the Complexity of Planning in Transportation and Manipulation Domains. Freiburg: Albert-Ludwigs-Universität, Diplomarbeit, 2001.

[Hel03] Helmert, Malte: Complexity Results for Standard Benchmark Domains in Planning. In: *Artificial Intelligence* 143 (2003), 2, S. 219-262

[Hel04] Helmert, Malte: A Planning Heuristic Based on Causal Graph Analysis. In: *Proc. of the 16th International Conference on Automated Planning and Scheduling (ICAPS)* (2004), S. 161-170

[Hel05] Helmert, Malte: Mündliche Kommunikation im Dezember 2005.

[Hoc96] Hochbaum, Dorit (Hrsg.): *Approximation Algorithms for NP-hard Problems.* Boston: PWS, 1996

[Hoo90] Hoogeveen, J.A.: Analysis of Christofides' Heuristic: Some Paths are more Difficult than Cycles. Amsterdam: Centre for Mathematics and Computer Science, 1990 (Report BS-R9005)

[Hro03] Hromkovič, Juraj: *Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics.* Berlin: Springer, 2003 [korr. Druck 2004]

[JP85] Johnson, D.S.; Papadimitriou, C.H.: Performance Guarantees for Heuristics. In: Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G.; Shmoys, D.B. (Hrsg.): *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Chichester: John Wiley and Sons 1985 [Nachdruck mit Index 1990]

[KM96] Khanna, Sanjeev; Motwani, Rajeev: Towards a Syntactic Characterization of PTAS. In: *Proc. of the 28th Annual ACM Symposium on Theory of Computing (STOC)* (1996), S. 329-337

[KV06] Korte, Bernhard; Vygen, Jens: *Combinatorial Optimization: Theory and Algorithms.* 3. Aufl. Berlin: Springer 2006

[Kor01] Kortsarz, G.: On the Hardness of Approximating Spanners. In: *Algorithmica* 30 (2001), S. 432-450

[KRW00] Krumke, Sven O.; Rambau, Jörg; Weider, Steffen: An Approximation Algorithm for the Non-Preemptive Capacitated Dial-a-Ride Problem. ZIB-Report 00-53. Berlin: Konrad-Zuse-Zentrum für Informationstechnik Berlin. 2000

[Mat05] Mattmüller, Robert: Approximatives Planen in der Grid-Domäne. Freiburg: Albert-Ludwigs-Universität, Studienarbeit, 2005.

[McH90] McHugh, James A.: *Algorithmic Graph Theory.* Englewood Cliffs, New Jersey: Prentice Hall, 1990

[Mit98] Mitchell, J.B.S.: Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric k-MST, TSP, and Related Problems. SIAM Journal of Computing 28 (1998), 4, S. 1298-1309

[Nag05] Nagamochi, H.: Approximating the Minmax Rooted-Subtree Cover Problem. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E88-A (2005), 5, S. 1335-1338

[PS82] Papadimitriou, Christos; Steiglitz, Kenneth: *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, New Jersey: Prentice Hall, 1982 [auch erhältlich als korr. Nachdruck, erschienen 1998 bei Dover in Mineola, New York]

[PV00] Papadimitriou, C.; Vempala, Santosh: On the Approximability of the Travelling Salesman Problem. In: *Proc. of the 32nd ACM Symposium on Theory of Computing*, 2000. S. 126-133 [Korr. Version erhältlich unter http://www.cs.berkeley.edu/~christos/tsp.ps]

[PY91] Papadimitriou, C.; Yannakakis, M.: Optimization, Approximation, and Complexity Classes. In: *Journal of Computer and System Sciences* 43 (1991), S. 425-440

[PY93] Papadimitriou, C.; Yannakakis, M.: The Traveling Salesman Problem with Distances One and Two. In: *Mathematics of Operations Research* 18 (1993), 1, S. 1-11

[Rög05] Röger, Gabi: Approximationsverfahren für Planungsprobleme in den Domänen Satellite, Depots und Driverlog. Freiburg: Albert-Ludwigs-Universität, Studienarbeit, 2005.

[RSL77] Rosencrantz, Daniel J.; Stearns, Richard E.; Lewis, Philip M. II: An Analysis of Several Heuristics for the Traveling Salesman Problem. In: *SIAM Journal of Computing* 6 (1977), 3, S. 563-581

[RN03] Russell, Stuart; Norvig, Peter: *Artificial Intelligence: A Modern Approach*. 2. Aufl. Englewood Cliffs, New Jersey: Prentice Hall, 2003

[Tre04] Trevisan, Luca: Inapproximability of Combinatorial Optimization Problems. Electronic Colloquium on Computational Complexity, 2004 (Report TR04-065) [erhältlich unter: http://www.cs.berkeley.edu/~luca/pubs/inapprox.pdf]

[Vaz01] Vazirani, Vijay V.: *Approximation Algorithms*. Berlin: Springer, 2001 [2. korr. Druck 2003]

[Weg03] Wegener, Ingo: *Komplexitätstheorie: Grenzen der Effizienz von Algorithmen*. Berlin: Springer, 2003 [2. korr. Druck 2003]