Institut für Theoretische Informatik
Universität zu Lübeck

Direktor: Prof. Dr. R. Reischuk

Approximationsalgorithmen für Graphüberdeckungsprobleme

Habilitationsschrift

verfaßt an der Technisch-Naturwissenschaftlichen Fakultät der

Universität zu Lübeck

zur Erlangung der Venia legendi für das Fach Informatik

vorgelegt von

Markus Bläser

Lübeck, 2002.

Danksagungen

Ich danke Jan Arpe, Gerhard Buntrock, Karin Genther, Andreas Jakoby, Maciej Liśkiewicz, Bodo Manthey, Christian Schindelhauer und Thomas Zeugmann, die durch eine Vielzahl von Diskussionen und Kommentaren zum Gelingen dieser Arbeit beigetragen haben.

Jan Arpe, Annette Langer und Bodo Manthey danke ich für die Durchsicht des Manuskripts und die daraus resultierenden Verbesserungsvorschläge.

Mein besonderer Dank gilt Prof. R. Reischuk, der durch viele Diskussionen meine Arbeit gefördert hat und wesentlich zum Gelingen dieser Arbeit beigetragen hat.

Inhaltsverzeichnis

Approximationsalgorithmen für Graphüberdeckungsprobleme

Markus Bläser

1 Einleitung

Das Travelling Salesperson Problem ist eines der wichtigsten Probleme der kombinatorischen Optimierung. Gegeben ist ein Graph mit gewichteten Kanten. Gesucht ist eine Hamiltonsche Tour T, also eine Rundreise, die jeden Knoten genau einmal besucht, so daß das Gesamtgewicht der verwendeten Kanten in T optimal wird. Dabei sind eine Reihe von Variationen möglich: Der Graph kann gerichtet oder ungerichtet sein (man spricht hier auch von dem asymmetrischen oder symmetrischen Fall, kurz ATSP oder TSP) oder die Gewichte können bestimmten Einschränkungen wie z.B. der Dreiecksungleichung unterliegen, um nur die wichtigsten Varianten zu nennen. Hauptgegenstand dieser Arbeit ist das Studium einiger Varianten des Travelling Salesperson Problems unter dem Gesichtspunkt der Approximierbarkeit. Weiterhin werden wir Relaxierungen des Travelling Salesperson Problems untersuchen, sogenannte Zyklenüberdeckungen. Hier soll der Graph nicht mit einem großen Zyklus überdeckt werden, sondern mit einer Reihe von knotendisjunkten Zyklen, die typischerweise eine vorgeschriebene Mindestlänge haben sollen. Eine detaillierte Beschreibung von Anwendungen und Varianten des Travelling Salesperson Problems sowie ausführliche Hintergrundinformationen zur Geschichte dieses Problems geben z.B. Lawler et al. [50].

Wie viele andere kombinatorische Optimierungsprobleme sind die meisten Varianten des Travelling Salesperson Problems NP-harte Optimierungsprobleme. Gleiches gilt für die in dieser Arbeit untersuchten Relaxierungen. Für große Instanzen ist es damit meist nicht möglich, eine optimale Lösung in akzeptabler Zeit zu berechnen. Daher wendet man schon seit einigen Jahrzehnten das Konzept der Approximation auf solche Probleme an. Statt eine optimale Lösung zu berechnen, sucht man nach Verfahren, die in akzeptabler Zeit – meist bedeutet dies in polynomieller Zeit – eine Lösung finden, deren Güte nahe am Optimum liegt.

Interessanterweise wurden Approximationsalgorithmen schon vor der Entwicklung der Theorie der NP-Vollständigkeit definiert. So kann man z.B. Vizings Theorem [67] als einen Approximationsalgorithmus ansehen. (Dies ist im übrigen einer der wenigen bekannten Approximationsalgorithmen, der eine Lösung berechnet, die um maximal eins von dem Optimum entfernt ist.) Die wahre Bedeutung von Approximationsalgorithmen wurde jedoch erst nach der Definition des Konzepts der NP-Vollständigkeit sowie der Formulierung der „$\mathsf{P} \neq \mathsf{NP}$“-Vermutung klar. Gute Einführungen zum Thema Approximationsalgorithmen sind z.B. [4, 65].

Aufbau der Arbeit. In dem ersten Kapitel werden die in der Arbeit betrachteten Probleme eingeführt. Zuvor werden benötigte Definitionen aus der Graphentheorie gegeben sowie kurz in die Theorie der Approximationsalgorithmen eingeführt. Die weiteren Kapitel enthalten die neu gewonnenen Resultate und Algorithmen. Diese Kapitel sind bewußt knapp gehalten, um ein schnelles Lesen zu ermöglichen. Benötigte Definitionen, die Diskussion bisheriger Resultate sowie Anwendungen finden sich alle im ersten Kapitel, das dadurch auch einen guten Überblick über den Rest der Arbeit gibt. Bei der Darstellung der Algorithmen steht zuerst nur der Aspekt im Vordergrund, daß die Algorithmen polynomielle Laufzeit haben. Exakte Laufzeitabschätzungen finden sich dann immer am Ende des jeweiligen Abschnitts.

Eine Reihe der erzielten Resultate sind bereits veröffentlicht worden. Dies sind die Ergebnisse in den Abschnitten 2.1 bis 2.3 (siehe [10]), in den Abschnitten 3.2 und 3.3 (siehe [12]), in dem Abschnitt 4.1 (siehe [8]), in dem Abschnitt 4.3 (siehe [11]) sowie im Kapitel 5 (siehe [9]). Zwei der Arbeiten, nämlich [12, 11], sind in Zusammenarbeit mit Bodo Manthey entstanden. Beide Arbeiten bestehen jeweils aus Approximationsalgorithmen sowie Härteresultaten. Aus beiden Arbeiten habe ich jeweils nur die Algorithmen übernommen, die zum größten Teil von mir entwickelt worden sind, während Bodo Manthey hauptsächlich die Härtebeweise geführt hat.

1.1 Definitionen aus der Graphentheorie

Wir wiederholen kurz wichtige Definitionen und vereinbaren einige Konventionen. Meist liegt in dieser Arbeit ein vollständiger kantengewichteter Graph G vor. Da bei der Betrachtung von Hamiltonschen Touren und Zyklenüberdeckungen Schleifen belanglos sind, genügt es, vollständige schleifenfreie Graphen zu betrachten. Ist V die Knotenmenge von G, so bezeichnet im gerichteten Fall $K(V) = (V \times V) \setminus \{(v,v) \mid v \in V\}$ die Menge der Kanten von G. Im ungerichteten Fall bezeichnen wir die Kanten von G mit $U(V) = \binom{V}{2}$, wobei $\binom{V}{2}$ gerade die Menge aller Teilmengen von V der Kardinalität zwei bezeichnet. Schließlich ist eine Gewichtsfunktion $w : K(V) \to \mathbb{Q}$ bzw. $w : U(V) \to \mathbb{Q}$ gegeben, die den Kanten rationale Gewichte zuweist. Meist werden diese nicht-negativ sein, also aus $\mathbb{Q}_{\geq 0}$.

Hauptgegenstand dieser Arbeit sind *Zyklenüberdeckungen*: Eine Zyklenüberdeckung eines gerichteten bzw. ungerichteten Graphen G ist eine Menge von

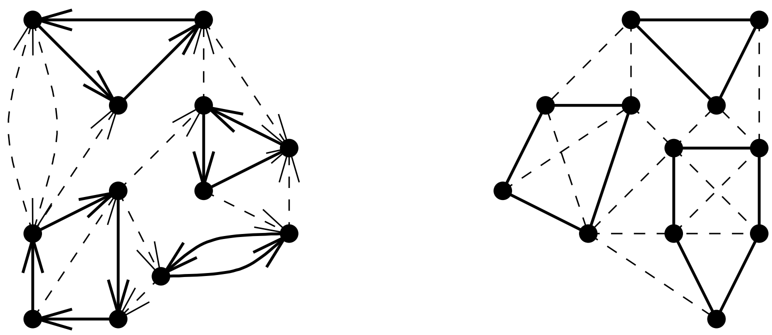

Abbildung 1: Links eine gerichtete Zyklenüberdeckung, rechts eine ungerichtete. Die durchgezogenen Kanten gehören zu der jeweiligen Überdeckung, die gestrichelten Kanten sind weitere Kanten des Graphen.

gerichteten bzw. ungerichteten Zyklen, so daß jeder Knoten zu genau einem Zyklus gehört. (Dabei haben Zyklen im gerichteten Fall immer mindestens zwei, im ungerichteten immer mindestens drei Kanten bzw. Knoten.) Abbildung 1 zeigt Beispiele einer gerichteten und ungerichteten Zyklenüberdeckung. Besteht eine Zyklenüberdeckung aus genau einen Zyklus, so sprechen wir von einer *Hamiltonschen Tour*. Eine Zyklenüberdeckung heißt k-Zyklenüberdeckung, falls jeder Zyklus *mindestens* Länge k hat. Im gerichteten Fall ist jede Zyklenüberdeckung automatisch eine 2-Zyklenüberdeckung, im ungerichteten eine 3-Zyklenüberdeckung. Ein (gerichteter oder ungerichteter) Zyklus heißt ein k-Zyklus, wenn er *genau* Länge k hat, also aus genau k Kanten besteht.

Wir werden in Kapitel 2 auch *partielle Zyklenüberdeckungen* betrachten. Hier verlangen wir, daß jeder Knoten zu höchstens einem Zyklus gehört. Aus Konsistenzgründen zählen wir die isolierten Knoten einer partiellen Zyklenüberdeckung jedoch zu der Anzahl der Zyklen hinzu, wenn wir von der Anzahl der Zyklen einer partiellen Zyklenüberdeckung sprechen.

Wir behandeln Hamiltonsche Touren und Zyklenüberdeckungen als Mengen von Kanten, da die zugrunde liegende Knotenmenge immer aus dem Kontext klar sein wird. Ebenso behandeln wir Pfade oder Zyklen als Mengen von Kanten. Für eine Menge von Kanten E eines Graphen G mit Gewichtsfunktion w sei $w(E)$ die Summe der Gewichte aller Kanten in E, d.h. $w(E) = \sum_{e \in E} w(e)$. So weisen wir insbesondere Zyklenüberdeckungen und Hamiltonschen Touren ein Gewicht zu. (Im Fall von Hamiltonschen Touren spricht man oft auch von deren Länge. In dieser Arbeit sprechen wir allerdings einheitlich von Gewicht.)

Eine Kante e heißt *inzident* mit einem Knoten v, falls $e = (v, u)$ oder $e = (u, v)$ für einen Knoten u gilt (gerichteter Fall) bzw. falls $v \in e$ gilt (ungerichteter Fall). Ein Knoten v *gehört* zu einem Pfad oder Zyklus, falls es eine Kante in diesem Pfad bzw. Zyklus gibt, die inzident mit v ist.

Für eine gerichtete Kante (u, v) nennen wir u den *Startknoten* und v den

Endknoten. Ist P ein gerichteter Pfad und gehören die Knoten $v_1, \dots, v_\ell$ zu P (in dieser Reihenfolge), so heißt v_1 der Startknoten und v_ℓ der Endknoten des Pfades P. Ist P ungerichtet, so sprechen wir von den beiden Endknoten v_1 und v_ℓ.

1.2 Theorie der Approximationsalgorithmen

Bei einem kombinatorischen Optimierungsproblem geht es darum, zu einer vorliegenden Eingabe eine bezüglich eines gegebenen Kostenmaßes „beste“ Lösung aus einer endlichen Menge von Lösungen zu finden. In diesem Abschnitt fassen wir kurz die Theorie der NP-Optimierungsprobleme zusammen. Alle in dieser Arbeit betrachteten Probleme sind NP-Optimierungsprobleme.

1.2.1 NP-Optimierungsprobleme

Ein NP-*Optimierungsproblem* Π besteht aus:

1. Einer Menge von *gültigen Instanzen* INS_Π, die in Polynomialzeit erkannt werden kann. Die *Größe* $|I|$ einer Instanz $I \in \mathrm{INS}_\Pi$ ist die Anzahl der Bits, die benötigt wird, um I hinzuschreiben. (Wir nehmen im folgenden jeweils an, daß alle vorkommenden Zahlen rationale Zahlen sind und daher als Bruch mit endlich vielen Bits geschrieben werden können.)

2. Jede Instanz $I \in \mathrm{INS}_\Pi$ hat eine Menge von *zulässigen Lösungen* $\mathrm{SOL}_\Pi(I)$. Wir verlangen, daß immer $\mathrm{SOL}_\Pi(I) \neq \emptyset$ gilt und daß die Größe jeder Lösung $s \in \mathrm{SOL}_\Pi(I)$ polynomiell in $|I|$ ist. Zu gegebenem (I, s) kann in Polynomialzeit entschieden werden, ob $s \in \mathrm{SOL}_\Pi(I)$ oder nicht. (Einige Autoren verlangen nicht unbedingt $\mathrm{SOL}_\Pi(I) \neq \emptyset$, siehe z.B. [4].)

3. Es gibt eine *Zielfunktion* m_Π, die in polynomieller Zeit berechnet werden kann und die jedem Paar (I, s) einen nicht-negativen Wert zuweist.

4. Π ist entweder ein *Minimierungsproblem* oder ein *Maximierungsproblem*.

Die Klasse aller NP-Optimierungsprobleme bezeichnen wir mit NPO. Obige Definition geht auf Krentel [49] zurück.

Eine *optimale Lösung* für eine Instanz I eines Minimierungsproblems (Maximierungsproblems) ist eine zulässige Lösung, deren Wert unter der Zielfunktion minimal (maximal) ist. Den Wert der optimalen Lösung bezeichnen wir mit $\mathrm{OPT}_\Pi(I)$.

Mit jedem NP-Optimierungsproblem Π kann in natürlicher Weise ein Entscheidungsproblem Π_{dec} assoziiert werden, nämlich das Problem festzustellen, ob eine Instanz eine Lösung mit Wert $\leq B$ (Minimierungsproblem) bzw. $\geq B$ (Maximierungsproblem) hat, wobei B Teil der Eingabe ist.

Im Falle von ATSP wäre zum Beispiel die Menge der gültigen Instanzen die Menge aller vollständigen schleifenfreien gerichteten Graphen $G = (V, K(V))$

mit einer Funktion $w : K(V) \to \mathbb{Q}_{\geq 0}$. Die Menge aller zulässigen Lösungen wäre die Menge aller Hamiltonschen Touren T von G. Dies entspricht gerade der Menge aller Permutationen von V. (Die Menge der zulässigen Lösungen hängt hier nur von der Größe der Instanz ab.) Die Zielfunktion ist gegeben durch $w(T) = \sum_{e \in T} w(e)$.

1.2.2 Approximationsalgorithmen

Ein Optimierungsproblem $\Pi \in \mathsf{NPO}$ heißt NP-*hart*, wenn für jedes Entscheidungsproblem $\Pi' \in \mathsf{NP}$ gilt, daß Π' in Polynomialzeit auf Π Turing-reduziert werden kann. Mit anderen Worten, es gibt einen Polynomialzeit-Algorithmus, der Orakelzugriff auf Π hat und Π' entscheidet. Zu jedem Problem $\Pi \in \mathsf{NPO}$ ist das entsprechende Entscheidungsproblem Π_{dec} in NP. Auf der anderen Seite ist Π NP-hart, falls das entsprechende Entscheidungsproblem Π_{dec} NP-vollständig ist.

Ist ein Optimierungsproblem Π NP-hart, so kann Π nur dann exakt in polynomieller Zeit gelöst werden, wenn $\mathsf{P} = \mathsf{NP}$ gilt. Daher ist es natürlich, nach Approximationsalgorithmen mit polynomieller Laufzeit zu suchen, d.h. nach Algorithmen, die eine Lösung liefern, deren Wert nahe bei dem Optimum liegt.

In Bezug auf polynomielle Laufzeit ist es gleich, welches genaue sequentielle Maschinenmodell man bei der formalen Definition von Approximationsalgorithmen zugrunde legt. Für rein komplexitätstheoretische Belange ist die Turingmaschine das geeignete Maschinenmodell. Da wir hier vor allem an den Algorithmen, also oberen Schranken, interessiert sind, legen wir bei der Angabe von expliziten Laufzeiten die Random Access Machine, kurz RAM, (siehe z.B. [55]) zugrunde. Der Einfachheit halber nehmen wir weiterhin an, daß alle Befehle der RAM Einheitskosten haben. Dies ist gerechtfertigt, da in allen Algorithmen, bei denen wir dieses Maß zugrunde legen, alle verwendeten Zahlen nur polynomiell groß in der Größe der Eingabe sind. Daher würde man unter dem realistischeren logarithmischen Kostenmaß nur einen zusätzlichen logarithmischen Faktor erhalten. Oftmals messen wir die Eingabe in der Anzahl der Knoten des Graphen. In diesem Fall gehen wir davon aus, daß auch die Kantengewichte nur polynomiell groß in der Anzahl der Knoten sind. Sind diese größer, so sind die Laufzeitschranken entsprechend zu modifizieren.

Ein Algorithmus A ist ein α-*Approximationsalgorithmus* für ein Minimierungsproblem Π, falls A für alle Instanzen I eine zulässige Lösung s in polynomieller Zeit berechnet, so daß $m_\Pi(I, s) \leq \alpha(|I|) \cdot \mathrm{OPT}_\Pi(I)$ gilt, wobei $\alpha : \mathbb{N} \to \mathbb{Q}_{\geq 0}$. Die Lösung s hat dann *Approximationsgüte* $\alpha(|I|)$. Wir sagen dann auch, daß A Approximationsgüte α erreicht. Ist Π ein Maximierungsproblem, so ersetzen wir die obige Bedingung durch $m_\Pi(I, s) \geq \alpha(|I|) \cdot \mathrm{OPT}_\Pi(I)$. Für Minimierungsprobleme sind ausschließlich Funktionen α sinnvoll, die nur Werte ≥ 1 annehmen, für Maximierungsprobleme nur solche, die ausschließlich Werte ≤ 1 haben.

Genaugenommen wird die Approximationsgüte eines Algorithmus immer in der Größe der Instanz gemessen. Oft ist es aber bequemer, diese in einer verwandten Größe zu messen. Hier wird dies häufig die Anzahl der Knoten des Eingabegraphen sein.

Wir werden in dieser Arbeit immer eine *asymptotische Approximationsgüte* zugrunde legen: Wir nennen einen Algorithmus auch dann einen α-Approximationsalgorithmus, wenn nur $m_\Pi(I,s) \leq (\alpha(|I|) + o(\alpha(|I|))) \cdot \mathrm{OPT}_\Pi(I)$ (im Fall von Minimierungssproblemen) bzw. $m_\Pi(I,s) \geq (\alpha(|I|) - o(\alpha(|I|))) \cdot \mathrm{OPT}_\Pi(I)$ (im Fall von Maximierungsproblemen) gilt. Die im weiteren Verlauf vorkommenden Approximationsgüten sind meist von der Form $c + O(1/n)$ oder $c - O(1/n)$, wobei c eine Konstante und n die Anzahl der Knoten des Graphen ist. Der Term $o(\alpha(|I|))$ fällt hier also sehr schnell ab. (Die in der O-Notation versteckten Konstanten sind ebenfalls klein.) Wenn der Aufwand nicht zu groß ist, geben wir den Term $o(\alpha(|I|))$ auch explizit an. Z.B. hat der Algorithmus für Max-ATSP in Abschnitt 4.1 Güte $\frac{8}{13}(1 - \frac{1}{n})$.

1.2.3 Relaxierungen

Ein wichtiges Instrument beim Entwurf von Approximationsalgorithmen sind *Relaxierungen*. Unter einer Relaxierung verstehen wir hier eine Struktur, die in Polynomialzeit berechnet werden kann und deren Gewicht eine untere Schranke für das Gewicht der Struktur ist, die wir eigentlich berechnen möchten.

Betrachten wir das symmetrische Travelling Salesperson Problem mit Dreiecksungleichung. Gegeben sei ein vollständiger schleifenfreier ungerichteter Graph G mit einer Gewichtsfunktion w, die jeder Kante ein nicht-negatives Gewicht zuweist. Dabei erfülle w die Dreiecksungleichung, d.h. $w(u,v) \leq w(u,x) + w(x,v)$ für alle Knoten u, v und x. (Ist z.B. $u = v$, so ist formal $w(u,v)$ nicht definiert, da G schleifenfrei ist. Wir setzen dann $w(u,v) = 0$. Weiterhin schreiben wir oftmals $w(u,v)$ abkürzend für $w(\{u,v\})$ bzw. $w((u,v))$.) Jede Hamiltonsche Tour in G induziert einen Spannbaum von G, indem man eine Kante löscht. Also ist das Gewicht eines minimalen Spannbaums eine untere Schranke für das Gewicht einer Hamiltonschen Tour minimalen Gewichts. Minimale Spannbäume können aber effizient berechnet werden, siehe z.B. [22]. Ein mögliches Vorgehen, um eine Hamiltonsche Tour in G zu berechnen, ist nun das folgende: Wir berechnen einen minimalen Spannbaum T. Dupliziert man jede Kante in dem Spannbaum, so erhält man einen Graphen T' (genaugenommen einen Multigraphen), dessen Knoten alle geraden Grad haben. Insbesondere ist T' Eulersch. Eine Eulersche Tour E von T' kann ebenfalls effizient berechnet werden. Aus E berechnet man eine Hamiltonsche Tour H, indem man „Abkürzungen“ nimmt: Man besucht die Knoten in der Reihenfolge, die durch E vorgegeben ist. Sobald man einen Knoten ein zweites Mal besuchen würde, geht man einfach direkt zum nächsten Knoten von E, der noch nicht besucht worden ist. Da w die Dreiecksungleichung erfüllt, können diese Abkürzungen das Gewicht nicht vergrößern. Da T' das doppelte Gewicht von T hat und das Gewicht einer optimalen Hamiltonschen Tour mindestens so groß ist wie das Gewicht eines minimalen Spannbaums, hat die so konstruierte Hamiltonsche Tour H maximal das doppelte Gewicht einer optimalen Hamiltonschen Tour. Wir haben also einen 2-Approximationsalgorithmus erhalten. Christofides [21] erreicht sogar Approximationsgüte $\frac{3}{2}$, indem er einen Spannbaum mit einem Matching minimalen Gewichts kombiniert, statt jede Kante in dem Spannbaum zu verdoppeln.

Im Fall des asymmetrischen TSP mit Dreiecksungleichung wählt man Zyklenüberdeckungen als Relaxierung. Eine Zyklenüberdeckung mit minimalem Gewicht kann in Polynomialzeit berechnet werden. Dies geht durch Reduktion auf die Berechnung eines perfekten bipartiten Matchings minimalen Gewichts. Da jede Hamiltonsche Tour auch eine Zyklenüberdeckung ist, ist das Gewicht einer Zyklenüberdeckung minimalen Gewichts eine untere Schranke für das Gewicht einer Hamiltonschen Tour minimalen Gewichts. Indem man aus jedem Zyklus einen Knoten als Repräsentanten auswählt und rekursiv auf dem von den Repräsentanten induzierten Subgraph fortfährt, erhält man eine Approximation mit Güte $\log n$. (log bezeichnet hier und im folgenden den Logarithmus zur Basis 2.) Die Details werden in Kapitel 2 erläutert.

1.2.4 Die Klasse APX

Die Klasse APX ist eine wichtige Klasse in der Theorie der Approximationsalgorithmen. Diese besteht aus allen NP-Optimierungsproblemen, für die es einen Approximationsalgorithmus mit konstanter Approximationsgüte gibt.

Mit Ausnahme des Problems Δ-ATSP sind alle in dieser Arbeit betrachteten Probleme aus APX. Für Δ-ATSP ist diese Frage offen. Der beste zur Zeit bekannte Algorithmus mit polynomieller Laufzeit hat eine Güte von $\log n$. Jedoch sind keine nicht-konstanten unteren Schranken bekannt.

1.2.5 Die Klasse PTAS

Ein Algorithmus heißt *Polynomialzeit-Approximationsschema* für ein NP-Minimierungsproblem Π, falls er für alle $I \in \mathrm{INS}_\Pi$ und für jedes $\epsilon > 0$ eine Lösung s berechnet mit $m_\Pi(I, s) \leq (1 + \epsilon)\,\mathrm{OPT}_\Pi(I)$, und dies in Zeit polynomiell in $|I|$ geschieht. (Im Falle eines Maximierungsproblems ist die Definition in der offensichtlichen Weise zu modifizieren.) Dabei kann die Laufzeit von $1/\epsilon$ abhängen. Auf diese Weise kann Genauigkeit gegen Laufzeit eingetauscht werden. Die Klasse PTAS ist die Menge aller NP-Optimierungsprobleme, die ein Polynomialzeit-Approximationsschema besitzen.

Ist $\mathsf{P} \neq \mathsf{NP}$, so gilt $\mathsf{PTAS} \neq \mathsf{APX}$. Insbesondere sind APX-vollständige Probleme, wie im nächsten Unterabschnitt definiert, nicht in PTAS, sofern $\mathsf{P} \neq \mathsf{NP}$.

1.2.6 Approximationserhaltende Reduktionen

Eine zentrale Rolle bei der Definition von Vollständigkeit, z.B. für die Klasse NP, spielen *Reduktionen*. Für die Definition von NP-Vollständigkeit haben sich die Polynomialzeit-berechenbaren Many-One-Reduktionen als geeignet erwiesen. (Einige Autoren bevorzugen Logspace-Berechenbarkeit.) Diese sind jedoch ungeeignet, um Vollständigkeit für Klassen von Optimierungsproblemen zu definieren, da sie die Approximationsgüte nicht unbedingt erhalten.

Dies tun hingegen die sogenannten approximationserhaltenden Reduktionen, siehe z.B. [4]. Seien Π_1 und Π_2 zwei Minimierungsprobleme. Π_1 ist *reduzierbar* auf Π_2 mittels approximationserhaltenden Reduktionen, in Zeichen $\Pi_1 \leq_{AP} \Pi_2$, falls Funktionen f und g und eine Konstante $c \geq 1$ existieren, so daß die folgenden Bedingungen erfüllt sind:

1. Für jedes $I \in \mathrm{INS}_{\Pi_1}$ und jede rationale Zahl $\alpha > 1$ ist $f(I, \alpha) \in \mathrm{INS}_{\Pi_2}$.

2. Für jedes $I \in \mathrm{INS}_{\Pi_1}$, jede rationale Zahl $\alpha > 1$ und jedes $s \in \mathrm{SOL}_{\Pi_2}(f(I, \alpha))$ ist $g(I, s, \alpha) \in \mathrm{SOL}_{\Pi_1}(I)$.

3. Für jedes feste $\alpha > 1$ sind f und g Polynomialzeit-berechenbar.

4. Für jedes $I \in \mathrm{INS}_{\Pi_1}$, jede rationale Zahl $\alpha > 1$ und jedes $s \in \mathrm{SOL}_{\Pi_2}(f(I, \alpha))$ gilt: Gilt $m_{\Pi_2}(f(I, \alpha), s) \leq \alpha \cdot \mathrm{OPT}_{\Pi_2}(f(I, \alpha))$, so folgt $m_{\Pi_1}(I, g(I, s, \alpha)) \leq (1 + c(\alpha - 1)) \cdot \mathrm{OPT}_{\Pi_1}(I)$.

Das Tripel (f, g, c) heißt dann *approximationserhaltende Reduktion* von Π_1 auf Π_2. Sind Π_1 oder Π_2 Maximierungsprobleme, so ist jeweils in der entsprechende Ungleichung in 4. „$\leq$“ durch „$\geq$“ und der entsprechende Faktor durch den Kehrwert zu ersetzen.

Der Begriff „approximationserhaltend“ wird durch das folgende Lemma begründet.

Lemma 1.1 *Seien* Π_1 *und* Π_2 NP-*Optimierungsprobleme. Gilt* $\Pi_1 \leq_{AP} \Pi_2$ *und ist* $\Pi_2 \in \mathsf{APX}$ (*bzw.* $\Pi_2 \in \mathsf{PTAS}$), *so ist auch* $\Pi_1 \in \mathsf{APX}$ (*bzw.* $\Pi_1 \in \mathsf{PTAS}$).

In Analogie zum Begriff „NP-hart“ heißt ein Optimierungsproblem Π APX-*hart*, falls $\Pi' \leq_{AP} \Pi$ für alle $\Pi' \in \mathsf{APX}$ gilt. Gilt zusätzlich $\Pi \in \mathsf{APX}$, so nennt man Π APX-*vollständig*.

Eine wie oben definierte approximationserhaltende Reduktion (f, g, c) ist vielmehr ein Reduktionsschema, da die Reduktion auch von der Güte α abhängt. In den meisten, aber nicht allen Beispielen, die sich in der Literatur finden, kann auf diese Abhängigkeit verzichtet werden.

1.3 Travelling Salesperson Probleme und Relaxierungen

Wir stellen nun die in dieser Arbeit betrachteten Travelling Salesperson Probleme sowie entsprechende Relaxierungen vor. Meistens werden wir uns auf die algorithmisch anspruchsvolleren asymmetrischen Fälle konzentrieren. Bei den Varianten mit Gewichten eins und zwei werden wir aber auch die symmetrischen Fälle ausführlich behandeln.

1.3.1 Asymmetrisches TSP mit Dreiecksungleichung

Das allgemeine Travelling Salesperson Problem ist sowohl im ungerichteten als auch im gerichteten Fall nicht innerhalb eines Faktors $\alpha(n)$ in Polynomialzeit approximierbar, wobei $\alpha : \mathbb{N} \to \mathbb{N}$ eine beliebige Polynomialzeit-berechenbare Funktion ist, es sei denn $\mathsf{P} = \mathsf{NP}$ [56]. Dies folgt daraus, daß es NP-vollständig ist zu entscheiden, ob ein gegebener gerichteter oder ungerichteter Graph $G = (V, E)$ eine Hamiltonsche Tour besitzt oder nicht. Wir transformieren G in einen gewichteten Graphen $H = (V, K(V))$ wie folgt:

$$\text{Für alle } e \in K(V) \text{ sei} \quad w(e) = \begin{cases} 1 & \text{falls } e \in E, \\ n \cdot \alpha(n) & \text{falls } e \notin E. \end{cases}$$

Hat G eine Hamiltonschen Tour, so hat H eine Hamiltonsche Tour mit Gewicht n. Hat G keine Hamiltonsche Tour, so hat jede Hamiltonsche Tour von H Gewicht $> n \cdot \alpha(n)$, da mindestens eine Kante mit diesem Gewicht verwendet werden muß. Mit $\alpha(n)$ ist auch $n \cdot \alpha(n)$ in polynomieller Zeit berechenbar. Ein Algorithmus mit Approximationsgüte α kann nun entscheiden, ob H eine Tour vom Gewicht n oder $> n \cdot \alpha(n)$ hat. Folglich kann auch entschieden werden, ob G eine Hamiltonsche Tour besitzt.

Daher hat man nach Einschränkungen des allgemeinen Problems gesucht. Eine sinnvolle Einschränkung ist, daß die Gewichtsfunktion w die Dreiecksungleichung erfüllt, d.h. es gilt

$$w(u, v) \leq w(u, x) + w(x, v) \quad \text{für alle Knoten } u, x, v. \tag{1}$$

Viele Gewichtsfunktionen, die in praktischen Anwendungen vorkommen, erfüllen die Dreiecksungleichung.

Problem 1 (Δ-ATSP)

Instanz: *vollständiger schleifenfreier gerichteter Graph $G = (V, K(V))$,*
Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$,
die die Dreiecksungleichung erfüllt.
Ziel: *Hamiltonsche Tour H minimalen Gewichts $w(H)$.*

Das entsprechende ungerichtete Problem bezeichnen wir mit Δ-TSP. Δ-TSP und damit auch Δ-ATSP sind APX-hart. Dies folgt aus der APX-Härte des (weiter unten betrachteten) Spezialfalls von Δ-TSP, bei dem die Gewichtsfunktion nur Gewichte eins und zwei annimmt. (Dies erfüllt automatisch die Dreiecksungleichung, da $2 \leq 1 + 1$.) Wie weiter oben skizziert gibt es einen polynomiellen 2-Approximationsalgorithmus für Δ-TSP, der auf Berechnung minimaler Spannbäume basiert. Also ist Δ-TSP auch APX-vollständig. Für Δ-ATSP hingegen erreicht der beste zur Zeit bekannte Approximationsalgorithmus mit polynomieller Laufzeit Approximationsgüte $\log n$ [30]. Es ist ein offenes Problem, ob Δ-ATSP in APX liegt.

Der Algorithmus in [30] für Δ-ATSP basiert auf den folgenden Relaxierungen.

Problem 2 (k-DCC, Δ-k-DCC)

Instanz: *vollständiger schleifenfreier gerichteter Graph* $G = (V, K(V))$,
Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$,
die im Falle von Δ-k-*DCC die Dreiecksungleichung erfüllt.*
Ziel: k-*Zyklenüberdeckung* C *minimalen Gewichts* $w(C)$.

2-DCC läßt sich in Polynomialzeit lösen. Dies geht durch folgende Reduktion auf die Berechnung von bipartiten Matchings: Wir definieren einen kantengewichteten bipartiten Graphen $B = (V \cup V', A)$. Auf der einen Seite von B steht jeder Knoten $v \in V$ und auf der anderen Seite befindet sich eine entsprechende Kopie v'. Eine Kante (u, v') ist genau dann in A mit Gewicht $w(u, v)$, wenn $(u, v) \in K(V)$ gilt, d.h $u \neq v$. Aus einem perfekten bipartiten Matching M minimalen Gewichts in B erhält man eine 2-Zyklenüberdeckung C minimalen Gewichts von G, indem man jede Kante (u, v') in M wieder durch (u, v) ersetzt. (Da G schleifenfrei ist, kann es keine Kante der Form (u, u') in A geben.) Ein perfektes bipartites Matching minimalen Gewichts kann in Zeit $O(n^3)$ berechnet werden [52].

Δ-k-DCC ist allerdings schon APX-vollständig für $k \geq 3$. Die untere Schranke folgt aus der APX-Härte des entsprechenden Maximierungsproblems mit Gewichten zwischen eins und zwei [11], die obere Schranke durch zweimaliges Berechnen einer 2-Zyklenüberdeckung (siehe Abschnitt 2.2). Allgemein läßt sich so für Δ-k-DCC Approximationsgüte $\log k$ erreichen.

Im ungerichteten Fall lassen sich sogar 3-Zyklenüberdeckungen in Polynomialzeit berechnen. Dies geschieht durch Reduktion auf das allgemeine Matchingproblem in ungerichteten Graphen [63].

Schließlich studieren wir noch den Fall einer γ-parametrisierten Dreiecksungleichung. Genauer nehmen wir an, daß

$$w(u, v) \leq \gamma \cdot (w(u, x) + w(x, v)) \quad \text{für alle Knoten } u, x, v \tag{2}$$

für ein $1/2 \leq \gamma < 1$ gilt statt (1).

Problem 3 ($\Delta(\gamma)$-ATSP)

Instanz: *vollständiger schleifenfreier gerichteter Graph* $G = (V, K(V))$,
Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$,
die die γ-*parametrisierte Dreiecksungleichung erfüllt* $(1/2 \leq \gamma < 1)$.
Ziel: *Hamiltonsche Tour* H *minimalen Gewichts* $w(H)$.

1.3.2 TSP mit Gewichten eins und zwei

Da zwischen oberer und unterer Schranke für die Approximierbarkeit von Δ-ATSP eine Lücke klafft, hat man begonnen, Spezialfälle von Δ-ATSP zu studieren, unter anderem den Fall, in dem nur die Kantengewichte eins und zwei zugelassen sind. Hierbei ist die Dreiecksungleichung offensichtlich erfüllt.

Problem 4 (ATSP(1,2), TSP(1,2))

Instanz: *vollständiger schleifenfreier gerichteter Graph* $G = (V, K(V))$ *bzw.* *vollständiger schleifenfreier ungerichteter Graph* $G = (V, U(V))$, *Gewichtsfunktion* $w : K(V) \to \{1, 2\}$ *bzw.* $w : U(V) \to \{1, 2\}$.
Ziel: *Hamiltonsche Tour* H *minimalen Gewichts* $w(H)$.

TSP(1,2) und ATSP(1,2) sind, neben Euklidischen Instanzen, die wohl einfachsten Spezialfälle von Δ-TSP bzw. Δ-ATSP. Trotz seiner Einfachheit ist schon TSP(1,2) und damit auch das allgemeinere Problem ATSP(1,2) APX-hart. Dies beweisen Papadimitriou und Yannakakis [53]. Andererseits existiert natürlich ein 2-Approximationsalgorithmus, denn jede Tour hat höchstens Gewicht $2n$ und mindestens n.

Papadimitriou und Yannakakis zeigen, daß TSP(1,2) auch dann noch APX-hart ist, wenn der Graph $H = (V, E)$ beschränkten Grad vier hat, wobei E gerade die Kanten mit Gewicht eins enthält, also $e \in E$ gilt, genau dann wenn $w(e) = 1$. Csaba, Krysta und Karpinski [24] zeigen, daß die APX-Härte sogar gilt, wenn der Grad von H durch drei beschränkt ist. Daher ist es auch sinnvoll, diese Einschränkung von TSP(1,2) zu betrachten.

Problem 5 (3-Deg-TSP(1,2))

Instanz: *vollständiger schleifenfreier gerichteter Graph* $G = (V, U(V))$, *Gewichtsfunktion* $w : U(V) \to \{1, 2\}$, *so daß der Graph* $H = (V, E)$ *beschränkten Grad drei hat, wobei* $E = \{e \in U(V) \mid w(e) = 1\}$.
Ziel: *Hamiltonsche Tour* H *minimalen Gewichts* $w(H)$.

Pfadpackungsprobleme. Eine Anwendung von Travelling Salesperson Problemen mit Gewichten eins und zwei sind sogenannte *Pfadpackungsprobleme*. Gegeben ist ein Graph $G = (V, E)$. Gesucht ist nun eine Kollektion P von knotendisjunkten Pfaden in G. Dabei soll die Anzahl der Kanten in P möglichst groß sein. Ein r-Approximationsalgorithmus für das Pfadpackungsproblem liefert immer einen $(2-r)$-Approximationsalgorithmus für das entsprechende TSP mit Gewichten eins und zwei [66]. Dazu entfernt man einfach alle Kanten mit Gewicht zwei, wendet auf den resultierenden Graph den r-Approximationsalgorithmus für das Pfadpackungsproblem an und fügt dann die erhaltenen Pfade wieder zusammen. Im schlimmsten Fall muß man für das Zusammenfügen ausschließlich Kanten vom Gewicht zwei verwenden. Dies liefert einen $(2 - r)$-Approximationsalgorithmus. (Die umgekehrte Richtung gilt nicht ohne weiteres in dieser Allgemeinheit. Allerdings lassen sich viele TSP Algorithmen in ähnlicher Weise für Pfadpackungsprobleme verwenden, es ist allerdings eine individuelle Analyse erforderlich.)

Pfadpackungsprobleme finden ihre Anwendung in der Code-Optimierung bei der Übersetzung von Hochsprachen in Maschinencode, siehe z.B. [15], sowie dem Abbilden von parallelen Programmen auf parallele Architekturen, siehe z.B. [54].

Schließlich betrachten wir Relaxierungen von ATSP(1,2) und TSP(1,2).

Problem 6 (k-DCC(1,2), k-UCC(1,2))

Instanz: *vollständiger schleifenfreier gerichteter Graph* $G = (V, K(V))$ *bzw. vollständiger schleifenfreier ungerichteter Graph* $G = (V, U(V))$, *Gewichtsfunktion* $w : E \to \{1, 2\}$.
Ziel: *eine* k*-Zyklenüberdeckung* C *minimalen Gewichts* $w(C)$.

Das entsprechende Entscheidungsproblem zu 3-DCC(1,2) ist bereits NP-hart. Dies zeigt Valiant [64] im Zuge des Beweises der #P-Härte der Permanente. Papadimitriou zeigt, daß das Entscheidungsproblem zu 6-UCC(1,2) NP-hart ist. (Dies berichten Cornuéjols und Pulleyblank [23].) Von 4-DCC(1,2) weiß man, daß es nicht in PTAS ist, es sei denn P = NP. Dies gilt auch für 12-UCC(1,2). Durch einfache Padding-Techniken lassen sich die Härteresultate in allen Fällen jeweils auf größere k ausdehnen.

1.3.3 Maximierungsvariante des asymmetrischen TSP

Wir untersuchen auch folgende Maximierungsvariante des Travelling Salesperson Problems, bei der eine möglichst schwere anstatt einer möglichst leichten Hamiltonschen Tour gefunden werden soll. Wiederum steht hier die asymmetrische Version im Vordergrund. Formal ist das Maximierungsproblem Max-ATSP wie folgt definiert.

Problem 7 (Max-ATSP)

Instanz: *vollständiger schleifenfreier gerichteter Graph* $G = (V, K(V))$, *nicht-negative Gewichtsfunktion* $w : K(V) \to \mathbb{Q}_{\geq 0}$.
Ziel: *Hamiltonsche Tour* H *maximalen Gewichts* $w(H)$.

Außer der Tatsache, daß w nur nicht-negative Werte annimmt, werden keine weiteren Voraussetzungen an w gestellt. Insbesondere braucht w nicht die Dreiecksungleichung erfüllen.

Auf den ersten Blick mag es ein wenig befremdlich wirken, eine möglichst lange Tour berechnen zu wollen. Eine Anwendung des Max-ATSP ist aber das Berechnen von *Shortest Common Superstrings.*

Shortest Common Superstrings. Eine Instanz dieses Problems ist eine Menge von Strings $s_1, \ldots, s_n$. Gesucht ist ein String s, der $s_1, \ldots, s_n$ als Substrings hat. Ein mögliches Optimierungskriterium ist die Länge $|s|$ von s, ein anderes die erreichte Kompression $(\sum_{\nu=1}^{n} |s_\nu|) - |s|$. (Bei exakter Optimierung sind beide Kriterien natürlich äquivalent. Bezüglich Approximierbarkeit ergeben sich allerdings Unterschiede.) Im Fall der Kompression kann das Problem unmittelbar als eine

Max-ATSP-Instanz modelliert werden: Die Knoten des Graphen sind die Strings $s_1, \dots, s_n$ und $w(s_i, s_j)$ ist die Überlappung zwischen s_i und s_j, d.h. $w(s_i, s_j)$ ist die Länge des längsten Strings v, so daß $s_i = xv$ und $s_j = vy$ für geeignete Strings x und y. Bei der Optimierung der Länge tritt Max-ATSP als Blackbox in vielen Approximationsalgorithmen auf, siehe z.B. [16] und die dortigen Literaturhinweise. Das Shortest Common Superstring Problem findet z.B. Anwendung in der Datenkompression und dem Shotgun Sequence Assembly [58].

Max-ATSP ist APX-hart, dies folgt aus der APX-Härte von ATSP(1,2) [53], in dem man Gewichte zwei durch Gewichte null ersetzt. Auf der anderen Seite ist leicht zu sehen, daß Max-ATSP in APX liegt: Z.B. kann man ein Matching M maximalen Gewichts berechnen. Da sich jede Hamiltonsche Tour in zwei disjunkte Matchings zerlegen läßt (sofern der Graph eine gerade Anzahl von Knoten hat), beträgt das Gewicht von M mindestens die Hälfte des Gewichtes einer optimalen Hamiltonschen Tour. Fügt man die Kanten von M beliebig zu einer Hamiltonschen Tour zusammen, so kann man kein Gewicht verlieren, da die Kantengewichte alle nicht-negativ sind. Folglich ergibt sich ein $\frac{1}{2}$-Approximationsalgorithmus.

Max-k-DCC ist die folgende Relaxierung von Max-ATSP.

Problem 8 (Max-k-DCC)

Instanz: *vollständiger schleifenfreier gerichteter Graph* $G = (V, K(V))$,
nicht-negative Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$.
Ziel: k*-Zyklenüberdeckung* C *maximalen Gewichts* $w(C)$.

Max-2-DCC ist in Polynomialzeit lösbar. Dies sieht man wie im Fall von 2-DCC. Denn sowohl die Berechnung eines perfekten bipartiten Matchings minimalen als auch maximalen Gewichts kann in Zeit $O(n^3)$ durchgeführt werden. Für $k \geq 3$ ist Max-k-DCC hingegen APX-hart [11]. Dies gilt sogar, wenn w nur Werte aus $[1, 2]$ annimmt, also insbesondere die Dreiecksungleichung erfüllt.

1.3.4 Partielles TSP

Als letzte Variante betrachten wir partielle Zyklenüberdeckungen. Hier wird zusätzlich noch ein Parameter t mitgegeben. Ziel ist es, eine partielle k-Zyklenüberdeckung zu finden, die genau t Knoten abdeckt und dabei minimales Gewicht hat unter allen partiellen k-Zyklenüberdeckungen, die genau t Knoten abdecken. Der gegebene Graph kann dabei gerichtet oder ungerichtet sein.

Problem 9 (t-partielle k-Zyklenüberdeckung)

Instanz: *vollständiger schleifenfreier Graph* $G = (V, E)$,
nicht-negative Gewichtsfunktion $w : E \to \mathbb{Q}_{\geq 0}$,
ganze Zahl t.
Ziel: k*-Zyklenüberdeckung* C *minimalen Gewichts* $w(C)$,
die genau t *Knoten von* G *abdeckt.*

Wählt man $k > t/2$, so erhält man als Spezialfall das t-partielle TSP. Offensichtlich ist das obige Problem mindestens so schwer wie die nicht-partiellen Probleme; man setzt einfach $t = n$. Auf der anderen Seite ist das t-partielle k-Zyklenüberdeckungsproblem in polynomieller Zeit lösbar, wenn t konstant ist.

Anstatt wie bei den vorherigen Problemen nach Approximationsalgorithmen zu suchen, bietet sich im Fall der partiellen Probleme ein anderer Weg an. Unser Ziel ist es zu zeigen, daß das t-partielle k-Zyklenüberdeckungsproblem *fixed parameter tractable* mit Parameter t ist.

Fixed Parameter Tractability. Sei Π ein Optimierungsproblem, und sei I eine Instanz von Π. Nehmen wir an, daß I unter anderem aus einer natürlichen Zahl t besteht. Wir nennen dann das Optimierungsproblem Π *fixed parameter tractable* mit Parameter t, wenn es einen Algorithmus für Π gibt, der jede Instanz I exakt löst und der eine Laufzeit der Form $f(t) \cdot \text{poly}(|I|)$ hat. Die Laufzeit hängt also nur polynomiell von der Größe der Instanz ab. Die „kombinatorische Explosion“ kann einzig in dem Parameter t stattfinden. Downey und Fellows [25] entwickeln dazu eine umfassende Theorie.

Insbesondere wenn t klein ist und f nicht zu stark wächst, kann man ein Problem, das fixed parameter tractable ist, auf effiziente Art und Weise exakt lösen.

Weil wir uns auf den Nachweis konzentrieren, daß die partiellen Probleme fixed parameter tractable sind und nicht die Approximierbarkeit in den Vordergrund stellen, braucht die Gewichtsfunktion w nicht die Dreiecksungleichung erfüllen. Weiterhin muß der Graph G nicht vollständig sein, d.h. wir erlauben sogar Gewicht ∞.

Die verwendeten Methoden, um t-partielle k-Zyklenüberdeckungen zu berechnen, basieren auf dem sogenannten *Colour Coding*, das auf Alon, Yuster und Zwick [3] zurückgeht. Mit dieser Technik lassen sich weitere partielle Probleme effizient lösen, so z.B. das partielle Set Cover Problem.

Problem 10 (t-partielles Set Cover)

Instanz: *Mengen $S_1, \dots, S_m$, die jeweils Teilmenge eines Universums U sind,*
nicht-negative Gewichtsfunktion $w : \{S_1, \dots, S_m\} \to \mathbb{Q}_{\geq 0}$,
ganze Zahl t.

Ziel: *Familie von Mengen $S_{i_1}, \dots, S_{i_\ell}$,*
die mindestens t Elemente von U überdeckt, d.h. $|\bigcup_{\lambda=1}^{\ell} S_{i_\lambda}| \geq t$,
so daß $\sum_{\lambda=1}^{\ell} w(S_{i_\lambda})$ minimal wird.

Das partielle Set Cover Problem enthält als wichtige Spezialfälle das *partielle Vertex Cover Problem* sowie das *partielle Dominating Set Problem*.

Beim partiellen Vertex Cover Problem ist ein ungerichteter Graph $G = (V, E)$ gegeben. Die Menge U ist hier E und zu jedem Knoten $v \in V$ wird die Menge

$S_v = \{e \mid v \in e\}$ gebildet. (Insbesondere kommt also jedes Element aus U in maximal zwei Mengen vor.) Weiterhin kann mit jedem Knoten v und damit mit S_v noch ein nicht-negatives Gewicht assoziiert werden.

Beim partiellen Dominating Set Problem liegt ein gerichteter oder ungerichteter Graph $G = (V, E)$ vor. Das Universum U ist hier die Knotenmenge V. Zu $v \in V$ wird die Menge $S_v = \{v\} \cup \{u \in V \mid \{v, u\} \in E\}$ (ungerichteter Fall) bzw. $S_v = \{v\} \cup \{u \in V \mid (v, u) \in E\}$ (gerichteter Fall) gebildet. Auch hier kann mit jedem Knoten v bzw. mit S_v ein nicht-negatives Gewicht assoziiert werden.

1.4 Einordnung der neuen Resultate

1.4.1 Asymmetrisches TSP mit Dreiecksungleichung

Bisherige Resultate. Der beste zuvor bekannte Approximationsalgorithmus mit polynomieller Laufzeit erreicht Approximationsgüte $\log n$. Dies zeigen Frieze, Galbiati und Maffioli [30]. Wir betonen, daß die Approximationsgüte *exakt* $1 \cdot \log n$ beträgt und nicht etwa nur $O(\log n)$. Trotz intensiver Forschung in den letzten zwei Jahrzehnten konnte dieses Resultat nicht weiter verbessert werden. Auch eine Verringerung des Koeffizienten von $1 \cdot \log n$ konnte nicht erreicht werden. (Letzteres ist auch von Interesse, da eine Reduzierung hier sofort die im Worst Case garantierte Güte der approximativen Lösung verbessert.) Um es mit den Worten von Johnson et al. [45] zu sagen: „Nothing better has been found in two decades." Frieze, Galbiati und Maffioli untersuchen auch eine Reihe weiterer Heuristiken und zeigen, daß diese alle wesentlich schlechtere Approximationsgüte (im Worst Case) erreichen.

Für Δ-TSP konstruiert Christofides [21] einen $\frac{3}{2}$-Approximationsalgorithmus mit polynomieller Laufzeit. Sowohl im symmetrischen als auch im asymmetrischen Fall wird eine Verbesserung der obigen Approximationsgüten als ein wichtiges offenes Problem in der Theorie der Approximationsalgorithmen angesehen (siehe z.B. Vazirani [65, Problem 3.15]).

Aus der Richtigkeit der sogenannten *Held–Karp Vermutung* [42] würde folgen, daß eine bestimmte LP Formulierung von Δ-TSP *integrality gap* $\frac{4}{3}$ hat. Carr und Vempala [18] dehnen die Held–Karp Vermutung auf den asymmetrischen Fall aus. Ein Beweis dieser erweiterten Held–Karp Vermutung würde implizieren, daß eine bestimmte LP Formulierung von Δ-ATSP ebenfalls integrality gap $\frac{4}{3}$ hat. Dies bedeutet jedoch nicht, daß auch eine effiziente Rundungsprozedur existiert. Außerdem ist die Held–Karp Vermutung seit mehr als drei Jahrzehnten unbewiesen.

Kleinberg und Williamson (siehe Williamson [68]) präsentieren einen interessanten Approximationsalgorithmus, der auf wiederholten *minimum mean cycle* Berechnungen beruht. Der Algorithmus erreicht Güte $2 \cdot \ln n$, was schlechter als $\log n$ ist.

Chandran und Ram [19] studieren den Spezialfall der γ-parametrisierten Dreiecksungleichung (2). Ihr Hauptresultat ist ein Approximationsalgorithmus mit polynomieller Laufzeit, dessen Güte asymptotisch gegen $\frac{\gamma}{1-\gamma}$ geht. Für den symmetrischen Fall mit einer γ-parametrisierten Dreiecksungleichung zeigen Böckenhauer et al. [14] Approximationsalgorithmen mit Güten $1 + \frac{2\gamma-1}{3\gamma^2-2\gamma+1}$ sowie $\frac{2}{3} + \frac{\gamma}{3(1-\gamma)}$, die beide polynomielle Laufzeit haben.

Für Δ-k-DCC kann man den Algorithmus von Frieze, Galbiati und Maffioli leicht so modifizieren, daß man eine $\log k$-Approximation erhält. Für das mit Δ-k-UCC verwandte Problem, bei der jeder Zyklus Länge exakt k (anstelle von mindestens k) haben muß, entwerfen Goemans und Williamson [33] einen $4(1-\frac{1}{k})(1-\frac{1}{n})$-Approximationsalgorithmus.

Neue Resultate. Als erstes Hauptresultat (Abschnitt 2.1 bis 2.3) entwerfen wir einen $0.999 \cdot \log n$-Approximationsalgorithmus für Δ-ATSP, der Laufzeit $O(n^3)$ hat. Dies ist die erste nicht-triviale Verbesserung des Resultats von Frieze, Galbiati und Maffioli [30] nach 20 Jahren. („Nicht-trivial“ schließt hier Approximationsgüten der Form $1 \cdot \log n - O(\log\log n)$ aus, die man erreichen kann, indem man kleine Subprobleme in dem Algorithmus von Frieze, Galbiati und Maffioli exakt löst.) Dieses Resultat ist der ermutigende Beweis, daß das Resultat von Frieze, Galbiati und Maffioli nicht das letzte Wort ist. Insbesondere schließt dies ein Resultat aus, wie es für das Set Cover Problem erzielt worden ist: Hier gibt Johnson [44] einen Approximationsalgorithmus an, der Güte $1 \cdot \ln n + 1$ erreicht. Dieses Resultat konnte über zwei Jahrzehnte nicht verbessert werden. Schließlich gibt Feige [28] für dieses Phänomen eine natürliche Erklärung, indem er zeigt, daß Johnsons Güte im wesentlichen optimal ist. Für kein $\epsilon > 0$ gibt es einen polynomiellen $((1-\epsilon) \cdot \ln n)$-Approximationsalgorithmus für Set Cover, es sei denn $\mathsf{NP} \subseteq \mathsf{DTime}(n^{O(\log\log n)})$.

In Abschnitt 2.4 wenden wir uns dem Problem $\Delta(\gamma)$-ATSP zu. Hauptresultat ist ein polynomieller Approximationsalgorithmus mit Güte $\frac{1}{1-\frac{1}{2}(\gamma+\gamma^3)}$. Dies verbessert das Resultat von Chandran und Ram für $\gamma \geq 0.6507$.

1.4.2 TSP mit Gewichten eins und zwei

Bisherige Resultate. 2-DCC(1,2) läßt sich in Polynomialzeit durch die oben beschriebene Reduktion auf bipartites Matching lösen. Für $k \geq 4$ ist k-DCC(1,2) nicht in PTAS, es sei denn $\mathsf{P} = \mathsf{NP}$ [12]. Der Fall $k = 3$ ist unseres Wissens nach offen. Ebenso ist unbekannt, ob k-DCC(1,2) für $k \geq 3$ sogar APX-hart ist.

Das entsprechende Entscheidungsproblem ist für $k = 2$ offensichtlich in P, da dies auch für 2-DCC gilt. Für $k \geq 3$ ist das Problem NP-hart. Dies zeigt Valiant im Zuge des Beweises der #P-Härte der Permanente [64] (siehe auch [32, GT13]).

Die Berechnung einer 3-Zyklenüberdeckung minimalen Gewichts, 3-UCC(1,2), ist in Polynomialzeit möglich. Dies geht via Tuttes Reduktion [63] auf das klassische Matching Problem in allgemeinen Graphen, welches in Polynomialzeit gelöst werden kann (siehe Edmonds [26]). Hartvigsen [36] entwirft einen polynomiellen Algorithmus für das zu 4-UCC(1,2) gehörende Entscheidungsproblem. Ebenso zeigt er einen polynomiellen Algorithmus, um 5-Zyklenüberdeckungen in einem bipartiten Graphen zu finden [37]. Als untere Schranke ist bekannt, daß k-UCC(1,2) für $k \geq 12$ nicht in PTAS ist.

Die Probleme ATSP(1,2) und TSP(1,2) sind beide APX-vollständig. Dies zeigen Papadimitriou und Yannakakis [53]. Explizite untere Schranken für die Nichtapproximierbarkeit beweisen Engebretsen und Karpinski [27]. Die beste obere Schranke für TSP(1,2) zeigen Papadimitriou und Yannakakis [53], sie geben einen polynomiellen $\frac{7}{6}$-Approximationsalgorithmus an. Im Falle von ATSP(1,2) beweist Vishwanathan die Existenz eines $\frac{17}{12}$-Approximationsalgorithmus mit polynomieller Laufzeit [66]. Nutzt man einen Algorithmus von Kosaraju, Park und Stein [47] für Max-ATSP, so erhält man einen polynomiellen Approximationsalgorithmus, der Güte $\frac{88}{63} \approx 1.397$ erreicht, indem man Gewicht zwei durch Gewicht null ersetzt. Ebenso würde man mit dem Algorithmus aus Abschnitt 4.1 Güte $\frac{18}{13} \approx 1.385$ erzielen.

Csaba, Karpinski und Krysta [24] zeigen, daß 3-Deg-TSP(1,2) APX-hart ist und geben auch explizite untere Schranken an. Sie beweisen auch (gegenüber den Algorithmen für die allgemeinen Probleme) verbesserte Approximationsalgorithmen für eine Reihe von Optimierungsproblemen mit sparsen Instanzen, lassen allerdings 3-Deg-TSP(1,2) als offenes Problem aus.

Neue Resultate. In Kapitel 3 verbessern wir eine Reihe der oben genannten Approximationsresulate und dehnen sie auf die Relaxierungen k-DCC(1,2) und k-UCC(1,2) aus. In Abschnitt 3.1 entwerfen wir einen generischen Algorithmus, den wir dann in den folgenden Abschnitten an die Probleme k-DCC(1,2), k-UCC(1,2) sowie 3-Deg-TSP(1,2) anpassen.

Für k-DCC(1,2) erhalten wir so einen $\frac{4}{3}$-Approximationsalgorithmus mit Laufzeit $O(n^{5/2})$. Sowohl Güte als auch Laufzeit sind, wie im folgenden auch für k-UCC(1,2), unabhängig von k. Insbesondere erhalten wir einen $\frac{4}{3}$-Approximationsalgorithmus für ATSP(1,2) mit Laufzeit $O(n^{5/2})$, indem wir $k > n/2$ wählen, und verbessern so die Resultate von Vishwanathan sowie Kosaraju, Park und Stein. Für das gerichtete Pfadpackungsproblem erhalten wir einen $\frac{2}{3}$-Approximationsalgorithmus.

Danach passen wir den generischen Algorithmus so an, daß wir einen $\frac{7}{6}$-Approximationsalgorithmus für k-UCC(1,2) mit polynomieller Laufzeit erhalten. Dieser beinhaltet den $\frac{7}{6}$-Approximationsalgorithmus von Papadimitriou und Yannakakis für TSP(1,2) als Spezialfall.

Schließlich erhalten wir einen 1.15-Approximationsalgorithmus für das Problem 3-Deg-TSP(1,2), der ebenfalls polynomielle Laufzeit hat. Dies verbessert den Faktor $\frac{7}{6}$ von Papadimitriou und Yannakakis, der unseres Wissens nach das beste Resultat auch für den sparsen Fall darstellt.

1.4.3 Maximierungsvariante des asymmetrischen TSP

Bisherige Resultate. Der erste polynomielle Approximationsalgorithmus für Max-ATSP geht auf Fisher, Nemhauser und Wolsey zurück [29]. Sie erreichen Approximationsgüte $\frac{1}{2}$. Dazu berechnen sie zuerst eine Zyklenüberdeckung maximalen Gewichts. Dies geht in Polynomialzeit. Dann entfernen sie aus jedem Zyklus die leichteste Kante und fügen die so erhaltenen Pfade zu einer Hamiltonschen Tour zusammen. Da die Gewichte nicht-negativ sind, kann das Zusammenfügen das Gewicht der Pfade nicht verringern. Im gerichteten Fall haben alle Zyklen mindestens Länge zwei. Folglich hat die Kollektion von Pfaden mindestens die Hälfte des Gewichts der 2-Zyklenüberdeckung, woraus das Resultat folgt. Dieses Resultat wird von Kosaraju, Park und Stein [47] verbessert, die einen polynomiellen Approximationsalgorithmus angeben, der Güte $\frac{38}{63} \approx 0.603$ erreicht.

Wie bereits erwähnt, ist die Relaxierung Max-2-DCC in Polynomialzeit berechenbar. Für $k \geq 3$ ist Max-k-DCC allerdings schon APX-hart, sogar wenn die Gewichte aus dem Intervall $[1, 2]$ kommen [11].

Im ungerichteten Fall kann man eine 3-Zyklenüberdeckung maximalen Gewichts in Polynomialzeit berechnen. Entfernt man die leichteste Kante aus jedem Zyklus, so zeigt eine Analyse wie oben, daß man so eine $\frac{2}{3}$-Approximation für Max-TSP erhält [29]. Der beste zur Zeit bekannte (deterministische) Approximationsalgorithmus mit polynomieller Laufzeit geht auf Serdyukov zurück [57], der einen $\frac{3}{4}$-Approximationsalgorithmus mit Laufzeit $O(n^3)$ entwirft. (Einen guten Überblick über diese und weitere Arbeiten in russischer Sprache geben Barvinok, Gimadi und Serdyukov [6].) Hassin und Rubinstein [39] konstruieren einen randomisierten Algorithmus mit polynomieller Laufzeit, dessen erwartete Approximationsgüte $\frac{25}{33} - \epsilon \approx 0.757$ für jedes $\epsilon > 0$ beträgt. Erfüllt die Gewichtsfunktion zusätzlich die Dreiecksungleichung, so erreicht ein Algorithmus von Kostochka und Serdyukov [48] Approximationsgüte $\frac{5}{6}$. Vor kurzem haben wiederum Hassin und Rubinstein [40] einen randomisierten Approximationsalgorithmus vorgestellt, der erwartete Güte $\frac{7}{8}$ hat. Unter Normen, für die die Einheitskugel ein endliches Polyeder im $\mathbb{R}^k$ ist, ist das Problem sogar in Polynomialzeit lösbar [7].

Eine Vielzahl von Arbeiten beschäftigen sich mit dem Shortest Common Superstring Problem, z.B. [61, 62, 13, 16, 60]. Tarhio und Ukkonen [61] und unabhängig Turner [62] erreichen Kompressionsrate $\frac{1}{2}$. Für die Länge als Optimierungskriterium zeigen Blum et al. [13] den ersten Approximationsalgorithmus, der konstante Approximationsgüte erreicht, nämlich 3. Die zur Zeit beste Längenapproximation erzielt Sweedyk [60], die einen Algorithmus mit Güte $2\frac{1}{2}$ entwirft.

Neue Resultate. Hauptresultat von Abschnitt 4.1 ist ein Approximationsalgorithmus für Max-ATSP mit polynomieller Laufzeit sowie Approximationsgüte $\frac{8}{13}(1 - \frac{1}{n}) \approx 0.615$. Die Laufzeit des Algorithmus ist proportional zu der Laufzeit, die man benötigt, um ein perfektes bipartites Matching maximalen Gewichts zu berechnen.

Kürzlich wurde dieses Resultat von Lewenstein und Sviridenko [51] verbessert. Lewenstein und Sviridenko entwerfen einen Algorithmus, der Approximationsgüte $\frac{5}{8}$ erreicht. Ihr Algorithmus ist LP-basiert, während unser Algorithmus rein kombinatorisch ist. Insbesondere muß der Algorithmus von Lewenstein und Sviridenko ein relaxiertes lineares Programm lösen, was zwar in polynomieller Zeit geht, aber nicht unbedingt effizient ist. Weiterhin muß der Algorithmus im Worst Case n^2 viele Instanzen des sogenannten „path coloring lemmas“ (Lemma 4.3) lösen, was (in der zur Zeit besten Implementierung) insgesamt $O(n^5)$ viele Schritte benötigt.

Falls die Kantengewichte zwischen 1 und B liegen, so kann der Algorithmus auch für ATSP verwendet werden. Wir ersetzen jedes Gewicht ω durch $B - \omega$, wenden auf den Graphen mit den neuen Gewichten unseren Algorithmus für Max-ATSP an und übertragen dann alles wieder zurück. Eine einfache Rechnung zeigt, daß wir so Approximationsgüte $\frac{8}{13} + \frac{5}{13}B + o(1)$ erreichen. Insbesondere erhalten wir einen $\frac{18}{13}$-Approximationsalgorithmus für ATSP(1,2), dessen Approximationsgüte damit nur unwesentlich schlechter als die des Algorithmus in Abschnitt 3.2 ist, der extra auf ATSP(1,2) zugeschneidert worden ist.

Wenden wir unseren Approximationsalgorithmus direkt auf das Shortest Common Superstring Problem an, so erhalten wir Kompressionsrate $\frac{8}{13}$. Benutzen wir unseren Algorithmus als Blackbox in einem Algorithmus von Breslauer, Jiang und Jiang [16], so erreichen wir Längenapproximation $2\frac{15}{26} \approx 2.577$, was nur unwesentlich schlechter ist als das Resultat von Sweedyk [60].

In Abschnitt 4.2 benutzen wir unseren Algorithmus, um damit auch k-Zyklenüberdeckungen maximalen Gewichts zu berechnen. Dazu sehen wir einfach die berechnete Hamiltonsche Tour als eine spezielle k-Zyklenüberdeckung an. Bis auf Terme niedrigerer Ordnung erhalten wir Güten $\frac{9}{16}, \frac{25}{43}, \frac{25}{43}, \frac{49}{83}, \frac{49}{83}, \dots$ für $k = 3, 4, 5, 6, 7, \dots$. Mit wachsendem k geht die Folge der Güten gegen $\frac{8}{13}$.

Eine ähnliche Analyse kann für den neuen Algorithmus von Lewenstein und Sviridenko für Max-ATSP durchgeführt werden. So erhält man z.B. Güte $\frac{7}{12}$ für Max-3-DCC.

Abschnitt 4.3 beschäftigt sich mit dem Spezialfall Max-3-DCC. Hier verbessern wir den Faktor $\frac{9}{16}$ aus dem vorherigen Abschnitt und erzielen Güte $\frac{3}{5}$ (wieder bis auf Terme niedrigerer Ordnung). Dies ist immer noch besser als das mittels des neuen Algorithmus von Lewenstein und Sviridenko erzielte Resultat.

1.4.4 Partielles TSP

Bisherige Resultate. Alon, Yuster, and Zwick [3] präsentieren einen Algorithmus mit Laufzeit $2^{O(t)} \cdot |V|^{b+1} \cdot \log |V|$, der einen Subgraphen minimalen Gewichts von G findet, der isomorph zu einem gegebenen Graphen H mit t Knoten und Baumweite b ist. Daraus folgt insbesondere, daß das t-partielle TSP fixed parameter tractable ist, da ein Zyklus Baumweite zwei hat. Das Problem, t-partielle k-Zyklenüberdeckungen zu berechnen, ist jedoch allgemeiner als das oben geschilderte Problem, da wir hier nicht nach einem bestimmten Graphen minimalen Gewichts suchen, sondern nach einem Graphen aus einer ganzen Klasse von Graphen (k-Zyklenüberdeckungen).

Kommen wir als nächstes zu dem Set Cover Problem. Schon der Fall $t = n$ kann nicht mit einem Faktor von $(1-\epsilon)\ln n$ in Polynomialzeit approximiert werden, es sei denn $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log\log n)})$ [28]. Für die t-partielle Version präsentiert Slavík [59] einen polynomiellen $H(\Delta)$-Approximationsalgorithmus, wobei Δ das Maximum der Kardinalitäten der Mengen $S_1, \ldots, S_m$ ist und $H(\Delta) = \sum_{\delta=1}^{\Delta} 1/\delta = \ln\Delta + O(1)$. Gandhi, Khuller und Srinivasan [31] zeigen einen f-Approximationsalgorithmus für das t-partielle Set Cover Problem mit polynomieller Laufzeit, falls jedes Element von U in höchstens f Mengen vorkommt.

Ein verwandtes Problem ist die Frage, ob es eine ungewichtete Mengenüberdeckung mit höchsten t Mengen gibt, die das komplette Universum U überdeckt. Dieses Problem (und die entsprechenden Spezialfälle) ist intensiv im Kontext der parametrisierten Komplexität untersucht worden. Es zeigt sich, daß dieses Problem $W[2]$-vollständig ist [25]. Unseres Wissens nach sind keine parametrisierten Resultate für das t-partielle Set Cover Problem bekannt.

In Bezug auf Approximierbarkeit sind das Set Cover Problem und das Dominating Set Problem äquivalent mittels approximationserhaltenden Reduktionen (siehe z.B. [4]).

Auch das Problem, ob es eine dominierende Menge der Größe höchstens t gibt, ist $W[2]$-vollständig [25]. Für planare Graphen ist das Dominating Set Problem jedoch fixed parameter tractable in Zeit $O(c^{\sqrt{t}} \cdot n)$ für eine Konstante c [2]. Die dort vorgestellte Methode läßt sich auch auf andere planare Probleme anwenden, wie z.B. das Vertex Cover Problem in planaren Graphen.

Die vollständige Version des Vertex Cover Problems (also $t = n$) kann nicht mit einem Faktor von 1.166 approximiert werden, es sei denn $\mathsf{P} = \mathsf{NP}$ [41]. Dies gilt sogar schon für den ungewichteten Fall. Mehrere Approximationsalgorithmen mit polynomieller Laufzeit sind für das t-partielle Vertex Cover Problem bekannt, die alle Approximationsgüte 2 erreichen, z.B. [43, 17, 5].

Das verwandte Problem, ob es eine ungewichtete Knotenüberdeckung gibt, die $\leq t$ Knoten enthält, ist fixed parameter tractable in Zeit $O(c^t + tn)$. Die zur Zeit beste obere Schranke für c ist 1.285 (siehe [20, 46]). Wiederum sind keine parametrisierten Resultate für das t-partielle Vertex Cover Problem bekannt.

Neue Resultate. Als erstes Resultat entwerfen wir einen randomisierten Algorithmus, um t-partielle k-Zyklenüberdeckungen zu berechnen, dessen Laufzeit $O(e^t(2^t \cdot n \cdot m + 4^t))$ beträgt. Dieser Algorithmus nimmt nicht an, daß w die Dreiecksungleichung erfüllt und läßt auch Gewichte ∞ zu. Der Algorithmus kann effizient derandomisiert werden. Es ergibt sich ein deterministischer Algorithmus mit Laufzeit $2^{O(t)} \cdot n \cdot m \cdot \log n$. Insbesondere ist also die Berechnung t-partieller k-Zyklenüberdeckungen fixed parameter tractable. Man kann dieses Resultat auch noch anders interpretieren: Solange $t = O(\log n)$ gilt, kann man t-partielle k-Zyklenüberdeckungen in Polynomialzeit berechnen.

Ebenso erhalten wir einen randomisierten Algorithmus für das t-partielle Set Cover Problem, der eine Laufzeit von $O(e^t(2^t \cdot t + n)m)$ hat. Auch dieser Algorithmus kann effizient derandomisiert werden. Inbesondere erhalten wir Algorithmen für das t-partielle Vertex Cover Problem sowie das t-partielle Dominating Set Problem, die beide die gleiche Laufzeit haben. Dies zeigt weiterhin, daß alle drei Probleme fixed parameter tractable sind.

1.5 Praktische Relevanz

In der Praxis sind Algorithmen von Bedeutung, die eine Lösung mit einer Fehlerrate von 2% bis 5% erreichen. Was ist also der Sinn von z.B. einem 2-Approximationsalgorithmus? Oder der Verbesserung der Approximationsgüte von 2 auf $\frac{3}{2}$?

Die Approximationsgüte eines Algorithmus ist eine Worst Case Abschätzung. Sie beschreibt das Verhalten des Algorithmus auf allen, also auch den härtesten Instanzen. Auf typischen Instanzen ist das Verhalten meist viel besser. Approximationsalgorithmen sind grundlegende Algorithmen, die auf die tatsächlich vorkommenden Instanzen noch entsprechend angepaßt werden können. Insbesondere haben viele Approximationsalgorithmen, die für eine gute Worst Case Performanz entworfen worden sind, Freiheitsgrade, die in der Praxis mit entsprechenden Heuristiken ausgefüllt werden können. So lassen sich oft praktisch relevante Resultate erreichen, siehe auch [65] für eine weitere Diskussion.

Weiterhin eliminiert die Verbesserung der Approximationsgüte harte Instanzen. Typischerweise wird die Konstruktion harter Instanzen umso schwieriger, je besser die Güte eines Algorithmus ist.

2 Asymmetrisches TSP mit Dreiecksungleichung

Es bezeichne im folgenden $G = (V, K(V))$ einen gerichteten vollständigen schleifenfreien Graphen und $w : K(V) \to \mathbb{Q}_{\geq 0}$ sei eine Gewichtsfunktion, die die Dreiecksungleichung erfüllt, d.h. es gilt

$$w(u, v) \leq w(u, x) + w(x, v) \quad \text{für alle } u, x, v \in V.$$

Es sei A das Gewicht einer 2-Zyklenüberdeckung minimalen Gewichts von G und L das Gewicht einer Hamiltonschen Tour minimalen Gewichts von G. Offensichtlich gilt $A \leq L$.

2.1 Gute partielle Zyklenüberdeckungen

Die Methode für Δ-ATSP, die wir in diesem Kapitel entwickeln, basiert auf wiederholter Berechnung von Zyklenüberdeckungen. Frieze, Galbiati und Maffioli [30] berechnen wiederholt 2-Zyklenüberdeckungen minimalen Gewichts, was in Polynomialzeit geht. So erhalten sie einen $\log n$-Approximationsalgorithmus. Eine bessere Approximationsgüte (nämlich $\log_3 n \approx 0.631 \cdot \log n$) könnte erreicht werden, wenn man z.B. 3-Zyklenüberdeckungen minimalen Gewichts berechnen würde. Das letztere Problem ist jedoch bereits APX-vollständig [11]. Wir führen eine Relaxierung von k-Zyklenüberdeckungen minimalen Gewichts ein, die wir im folgenden *gute partielle Zyklenüberdeckungen* nennen. Basierend auf dieser Relaxierung verallgemeinern wir in Abschnitt 2.2 den Algorithmus von Frieze, Galbiati und Maffioli.

Problem 11 (b-GPCC)

Instanz: *vollständiger schleifenfreier gerichteter Graph* $G = (V, K(V))$, *nicht-negative Gewichtsfunktion* $w : K(V) \to \mathbb{Q}_{\geq 0}$, *die die Dreiecksungleichung erfüllt.*

Ziel: *eine partielle Zyklenüberdeckung mit Gewicht* $\alpha \cdot L$ *und* $\beta \cdot |V|$ *Zyklen* $(\beta < 1)$, *so daß* $\alpha/(-\log \beta) \leq b$.

(Da $0 < \beta < 1$ gilt, ist $-\log \beta$ stets positiv.) Die Tatsache, daß bei b-GPCC auch partielle Zyklenüberdeckungen zugelassen sind, ist entscheidend für unseren Algorithmus. Wir nennen eine partielle Zyklenüberdeckung, die die obige Bedingung $\alpha/(-\log \beta) \leq b$ erfüllt, auch *b-gut*. Die genaue Bedeutung des Wertes $\alpha/(-\log \beta)$ wird in Abschnitt 2.2 klar werden. Intuitiv hat eine gute partielle Zyklenüberdeckung entweder geringes Gewicht oder wenige Zyklen. Unser Ziel ist es für ein $b < 1$ nachzuweisen, daß b-GPCC in P ist. Berechnet man einfach eine optimale 2-Zyklenüberdeckung (die maximal $n/2$ Zyklen hat), so sieht man, daß 1-GPCC in P liegt. Im Abschnitt 2.3 zeigen wir, daß b-GPCC für ein $b \leq 0.999$ in Polynomialzeit gelöst werden kann.

2.2 Wiederholte Berechnung von Zyklenüberdeckungen

Abbildung 2 zeigt einen generischen Approximationsalgorithmus für Δ-ATSP, der auf Ideen von Frieze, Galbiati und Maffioli [30] basiert. Zuerst wird eine partielle Zyklenüberdeckung C des gegebenen Graphen G berechnet. Dann wird aus jedem Zyklus ein Knoten als Repräsentant ausgewählt. In dem Subgraph, der von diesen Repräsentanten induziert wird, wird nun rekursiv ein Hamiltonsche Tour T' berechnet. C und T' (angesehen als ein Zyklus in G) werden schließlich zu einer Hamiltonschen Tour T in G kombiniert.

Lemma 2.1 *Seien D und D' zwei partielle Zyklenüberdeckungen über einer Knotenmenge V. Dann ist jede schwache Zusammenhangskomponente von $(V, D \cup D')$ Eulersch, also insbesondere stark zusammenhängend.*

Beweis. Da D und D' nur aus gerichteten Zyklen und isolierten Knoten bestehen, ist der Eingrad eines Knotens in $(V, D \cup D')$ gleich seinem Ausgrad. Damit

Eingabe: gerichteter Graph $G = (V, K(V))$ mit Gewichtsfunktion w, wobei w die Dreiecksungleichung erfüllt.
Ausgabe: eine Hamiltonsche Tour T.

1. Berechne eine b-gute partielle Zyklenüberdeckung C von G.
2. Wähle einen beliebigen Knoten aus jedem Zyklus in C aus. Es sei V' die Menge dieser Knoten (wobei jeder isolierte Knoten in C auch in V' eingefügt wird).
3. Berechne rekursiv eine Hamiltonsche Tour T' in dem Graphen G', der von V' induziert wird (falls $|V'| > 1$).
4. Kombiniere C und T' zu einer Hamiltonschen Tour T, wie in den Lemmata 2.1 und 2.2 beschrieben.

Abbildung 2: Der generische Algorithmus für Δ-ATSP basierend auf wiederholten Berechnungen von Zyklenüberdeckungen.

ist jede schwache Zusammenhangskomponente Eulersch. (Isolierte Knoten werden dabei auch als Eulersch angesehen.) ∎

Lemma 2.2 *Sei $H = (V, K(V))$ ein gerichteter Graph, $w : K(V) \to \mathbb{Q}_{\geq 0}$ erfülle die Dreiecksungleichung und sei S ein Subgraph von H. Ist jede schwache Zusammenhangskomponente von S Eulersch, dann gibt es eine partielle Zyklenüberdeckung C von H, so daß C genauso viele Zyklen wie S Zusammenhangskomponenten hat und $w(C) \leq w(S)$ gilt. Weiterhin kann solch ein C in Polynomialzeit konstruiert werden.*

Beweis. Für jede Zusammenhangskomponente berechnen wir eine Eulersche Tour. Diese Tour transformieren wir in einen Zyklus, indem wir *Abkürzungen* (siehe z.B. [50]) nehmen: Wir starten in einem beliebigen Knoten und folgen der Eulerschen Tour. Wenn wir auf einen Knoten stoßen, den wir schon einmal besucht haben, so gehen wir stattdessen direkt zum nächsten Knoten entlang der Eulerschen Tour, den wir noch nicht besucht haben. Da w die Dreiecksungleichung erfüllt, kann keine dieser Abkürzungen das Gesamtgewicht erhöhen. ∎

Die beiden Lemmata 2.1 und 2.2 ermöglichen es, $C \cup T'$ in eine Hamiltonsche Tour T von G zu transformieren, da $C \cup T'$ nach Konstruktion nur eine Zusammenhangskomponente besitzt. Das Gewicht von T ist höchstens $w(C) + w(T')$.

Damit die Laufzeit des generischen Algorithmus polynomiell ist, muß b in Schritt 1 so gewählt werden, daß b-GPCC in Polynomialzeit gelöst werden kann.

Bevor wir uns im nächsten Abschnitt dieser Problematik zuwenden, schätzen wir zuerst die Approximationsgüte des Algorithmus (in Abhängigkeit von b) ab. Wir behaupten, daß das Gewicht der berechneten Tour T höchstens $b \cdot \log n$ mal so groß wie das Gewicht einer optimalen Tour ist. Der Beweis geht mittels Induktion in der Anzahl der Knoten von G: Nehmen wir an, die berechnete Tour habe Approximationsgüte $b \cdot \log n'$ für alle Instanzen mit $n' < |V|$ Knoten. Nach Definition von b-gut gibt es α und β mit $\alpha/(-\log \beta) \leq b$, so daß C Gewicht $\alpha \cdot L$ hat und $\beta \cdot |V|$ Zyklen. Insbesondere gilt $|V'| = \beta \cdot |V|$. Nach Induktionsvoraussetzung hat die Tour T' höchstens Gewicht $b \cdot \log |V'| \cdot L'$, wobei L' das Gewicht einer optimalen Hamiltonschen Tour in G' ist. Damit hat die Tour T höchstens Gewicht

$$\begin{aligned} w(C) + w(T') &\leq \alpha \cdot L + b \cdot \log(\beta \cdot |V|) \cdot L' \\ &\leq (\underbrace{\alpha + b \cdot \log \beta}_{\leq 0}) \cdot L + b \cdot \log(|V|) \cdot L \\ &\leq b \cdot \log(|V|) \cdot L, \end{aligned}$$

da $\alpha + b \cdot \log \beta \leq 0$ wegen der Definition von b-gut und $L' \leq L$ nach der Dreiecksungleichung gilt.

Theorem 2.3 *Falls b-GPCC $\in \mathsf{P}$ für ein $0 < b \leq 1$, dann gibt es einen $b \cdot \log n$-Approximationsalgorithmus für Δ-ATSP, der polynomielle Laufzeit hat.* ■

Man erhält aus obigem Theorem den Algorithmus von Frieze, Galbiati und Maffioli [30], indem man ausnutzt, daß 1-GPCC mittels Berechnungen von 2-Zyklenüberdeckungen in Polynomialzeit lösbar ist.

2.3 Ein polynomieller Algorithmus für 0.999-GPCC

Eine 2-Zyklenüberdeckung minimalen Gewichts kann mittels des folgenden bekannten relaxierten linearen Programms berechnet werden:

$$\begin{aligned} &\text{Minimiere} \sum_{(u,v) \in K(V)} w(u,v) x_{(u,v)} \quad \text{wobei} \\ &\sum_{u \in V \setminus \{v\}} x_{(u,v)} = 1 \quad \text{für alle } v \in V, \qquad \text{(Eingrad-Bedingung)} \\ &\sum_{v \in V \setminus \{u\}} x_{(u,v)} = 1 \quad \text{für alle } u \in V, \qquad \text{(Ausgrad-Bedingung)} \\ &x_{(u,v)} \geq 0 \qquad \text{für alle } (u,v) \in K(V). \end{aligned} \tag{3}$$

Die Variablen $x_{(u,v)}$ entsprechen den Kanten (u,v) von G. Dies ist im wesentlichen eine LP-Formulierung des gewichteten bipartiten Matching Problems [52], außer das auf beiden Seiten die gleiche Knotenmenge vorhanden ist. Die zu (3) gehörige Matrix ist total unimodular [52]. Daher ist jede optimale Basislösung von (3) ganzzahlig (sogar $\{0,1\}$-wertig). Der beste zur Zeit bekannte Algorithmus mit stark polynomieller Laufzeit benötigt $O(n^3)$ Zeit zur Lösung von (3).

Im Worst Case besteht eine Zyklenüberdeckung, die man durch Lösen von (3) erhält, nur aus 2-Zyklen. Dies zeigt, daß 1-GPCC in P liegt. Um ein besseres Resultat zu erzielen, fügen wir die folgende Bedingung dem linearen Programm hinzu, die die 2-Zyklen eliminieren soll:

$$x_{(u,v)} + x_{(v,u)} \leq 1 \quad \text{für alle } (u,v) \in K(V) \quad \text{(2-Zyklen-Bedingung)} \tag{4}$$

Diese Bedingungen sind eine Teilmenge der sogenannten *subtour elimination constraints* (siehe z.B. [50]).

Fassen wir (3) und (4) als ganzzahliges lineares Programm auf, so verhindert die Bedingung (4), daß in einer optimalen Lösung 2-Zyklen vorkommen, da höchstens eine der Variablen $x_{(u,v)}$ und $x_{(v,u)}$ gleich eins sein kann. Allerdings ist es unwahrscheinlich, das dieses ganzzahlige lineare Programm in Polynomialzeit gelöst werden kann. Tatsächlich ist das Problem, 3-Zyklenüberdeckungen minimalen Gewichts zu berechnen APX-hart, sogar wenn die Dreiecksungleichung gilt und wenn die Gewichte zusätzlich noch zwischen eins und zwei liegen [11]. Nach dem Hinzufügen der Bedingung (4) ist die entsprechende Matrix auch nicht mehr total unimodular. Folglich kann eine optimale Lösung des relaxierten Programms auch fraktionell sein. Aus solch einer fraktionellen Lösung soll nun im übrigen Teil eine gute partielle Zyklenüberdeckung berechnet werden.

2.3.1 Dekomposition der fraktionellen Lösung

Es sei $x^*_{(u,v)}$ eine optimale Lösung des relaxierten Programms (3) zusammen mit Bedingung (4). Ziel dieses Unterabschnitts ist es, aus der fraktionellen Lösung $x^*_{(u,v)}$ eine Reihe von Zyklenüberdeckungen $C_1, \ldots, C_t$ zu gewinnen. Hierbei folgen wir Ideen von Lewenstein und Sviridenko [51]. Sei $W^* = \sum_{(u,v)\in K(V)} w(u,v)x^*_{(u,v)}$ das Gewicht der fraktionellen Lösung.

Eine erste Strategie wäre die folgende: Wähle B als kleinste ganze Zahl > 0, so daß für alle Kanten (u,v) der Wert $B \cdot x^*_{(u,v)}$ ganzzahlig ist. Es sei $\xi_{(u,v)} = B \cdot x^*_{(u,v)}$. Wir definieren nun einen Multigraphen H mit Knotenmenge V wie folgt: Für jede Kante $(u,v) \in K(V)$ fügen wir $\xi_{(u,v)}$ Kopien von (u,v) in H ein. Wegen der Grad-Bedingungen in (3) ist H ein B-regulärer Multigraph. Das nächste Lemma folgt unmittelbar aus Königs Kantenfärbungstheorem für bipartite Graphen (siehe z.B. [34, Chap. 30]).

Lemma 2.4 *Es sei H ein B-regulärer Multigraph. Dann können die Kanten von H so in B Mengen partitioniert werden, daß jede dieser Mengen eine Zyklenüberdeckung bildet. Solch eine Partition kann in Zeit polynomiell in B und $|V|$ gefunden werden.* ■

Die Zahl B muß allerdings nicht polynomiell in der Eingabegröße sein. Dies kann wie folgt umgangen werden: Eine Matrix heißt *doppelt stochastisch*, falls alle Einträge nicht negativ sind und alle Spalten- sowie Zeilensummen eins ergeben. Sind die Einträge der Matrix zusätzlich noch $\{0,1\}$-wertig, so spricht man von einer *Permutationsmatrix*. Aufgrund der Gradbedingungen in (3) ist die Matrix $X^* = (x^*_{(u,v)})$ doppelt stochastisch. (Um die Matrix X^* aufzubauen, ordnen wir die Knoten in V beliebig. Die Einträge auf der Hauptdiagonalen sind alle null.)

Lemma 2.5 (Birkhoff–Von Neumann) *Jede $n \times n$-Matrix S, die doppelt stochastisch ist, ist eine konvexe Kombination von höchstens n^2 Permutationsmatrizen, d.h. es gibt Permutationsmatrizen $P_1, \ldots, P_t$, $t \leq n^2$, und nicht-negative reelle Zahlen α_i mit $\sum_{i=1}^t \alpha_i = 1$, so daß $S = \sum_{i=1}^t \alpha_i P_i$. Solch eine Zerlegung kann in Polynomialzeit gefunden werden.* ■

Für einen Beweis des Birkhoff–Von Neumann Theorems, siehe [35, Chap. 3].

Wir zerlegen nun X^* gemäß dem Lemma. Jede $n \times n$–Permutationsmatrix induziert eine Zyklenüberdeckung in G. Wir wählen nun B als kleinste ganze Zahl > 0, so daß alle $\gamma_i := B \cdot \alpha_i$ ganzzahlig sind. (Die α_i sind rationale Zahlen mit polynomiell vielen Bits, da dies auch für die $x^*_{(u,v)}$ gilt.) Jede Matrix P_i repräsentiert γ_i viele identische Zyklenüberdeckungen C_i. (Die Matrix P_i repräsentiert tatsächlich eine Zyklenüberdeckung und nicht nur eine partielle Zyklenüberdeckung. Denn die Einträge auf der Hauptdiagonalen von P_i sind alle null, da dies für X^* gilt.) Dies ermöglicht es uns im folgenden, mit den Überdeckungen $C_1, \ldots, C_t$ zu arbeiten, jeweils mit $\gamma_1, \ldots, \gamma_t$ gewichtet, anstatt mit allen B Überdeckungen explizit zu arbeiten. Jede Überdeckung C_i zählt dabei als γ_i Überdeckungen. Es gilt $\sum_{i=1}^t \gamma_i w(C_i) = B \cdot W^*$. (Jede der Überdeckungen $C_1, \ldots, C_t$ ist eine vollständige Überdeckung und keine echt partielle. Später werden jedoch möglicherweise Zyklen entfernt werden.)

Das folgende Lemma, das auf Lewenstein und Sviridenko [51] zurückgeht, besagt, daß jeder 2-Zyklus in höchstens der Hälfte der Zyklenüberdeckungen $C_1, \ldots, C_t$ (gezählt mit Multiplizitäten $\gamma_1, \ldots, \gamma_t$) vorkommt. Dies ist ein wichtiges Werkzeug für die Analyse unseres Algorithmus.

Lemma 2.6 *Sei c ein 2-Zyklus. Es sei I_1 die Menge aller $i \in \{1, \ldots, t\}$, so daß c ein Zyklus in C_i ist. Dann gibt es eine Menge von Indizes I_2, so daß für alle $j \in I_2$ die Überdeckung C_j den 2-Zyklus c nicht enthält und $\sum_{j \in I_2} \gamma_j \geq \sum_{i \in I_1} \gamma_i$ gilt.* ■

2.3.2 Normalisieren der Zyklenüberdeckungen

Im folgenden werden wir die Vereinigung zweier Zyklenüberdeckungen C_i und C_j betrachten. Die Zusammenhangskomponenten der Vereinigung $C_i \cup C_j$ sind alle Eulersch. Jeder Knoten gehört zu einem oder zwei Zyklen. (Der Fall von einem Zyklus tritt ein, wenn C_i und C_j jeweils einen gleichen Zyklus enthalten.)

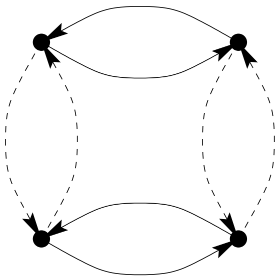

Abbildung 3: Eine Zusammenhangskomponente, die nur aus 2-Zyklen von C_i bzw. C_j besteht. Gestrichelte Kanten sind aus der einen, durchgezogene Kanten aus der anderen Zyklenüberdeckung. Solch eine Komponente hat immer eine gerade Anzahl von 2-Zyklen und kann in zwei große Zyklen mit entgegengesetzten Richtungen zerlegt werden.

Hauptaugenmerk gilt in diesem Unterabschnitt solchen Komponenten von $C_i \cup C_j$, die nur aus 2-Zyklen von C_i und C_j bestehen. Solch eine Komponente K kann aus nur einem 2-Zyklus bestehen; in diesem Fall ist der 2-Zyklus sowohl in C_i als auch C_j vorhanden. Besteht hingegen K aus mehr als einem 2-Zyklus, so muß K aus einer geraden Anzahl von 2-Zyklen bestehen, deren Kanten so in zwei Klassen aufgeteilt werden können, daß jede Klasse einen großen Zyklus bildet. Jeder dieser zwei großen Zyklen spannt die Knoten von K auf. Die beiden Zyklen unterscheiden sich einzig durch ihre Richtung. Abbildung 3 illustriert solch eine Komponente K. Im folgenden bezeichnen d_1 und d_2 diese beiden großen Zyklen.

Es seien w_i und w_j das Gesamtgewicht aller 2-Zyklen in C_i bzw. C_j, die zu K in $C_i \cup C_j$ gehören. Es seien x_1 und x_2 das Gewicht von d_1 bzw. d_2. Da wir nur Kanten umverteilt haben, gilt $w_i + w_j = x_1 + x_2$. O.B.d.A. können wir annehmen, daß $w_i \geq w_j$. Sei $k \in \{1, 2\}$ so gewählt, daß $x_k \leq w_i$. Wir ersetzen nun die 2-Zyklen in C_i, die zu K gehören, durch den Zyklus d_k. Lemma 2.6 gilt dann immer noch, da wir nur 2-Zyklen entfernen. Weiterhin ist das Gesamtgewicht höchstens $B \cdot W^*$: Die Knoten von K in C_i steuerten vor der Änderung Gewicht $\gamma_i \cdot w_i$ bei, nach der Änderung nur noch $\gamma_i \cdot x_k \leq \gamma_i \cdot w_i$. Alle anderen Kanten sind unverändert geblieben.

Die neue Zyklenüberdeckung muß keine zulässige Lösung mehr von (3) und (4) sein. Dies stellt kein Problem dar, da wir im folgenden nur benötigen, daß die Zyklenüberdeckungen Lemma 2.6 erfüllen und daß deren Gesamtgewicht höchstens $B \cdot W^*$ ist.

Indem wir diesen Prozeß gegebenenfalls wiederholen (maximal $n/4$ mal), können wir erreichen, daß keine Komponente von $C_i \cup C_j$, die mehr als zwei Knoten hat, nur aus 2-Zyklen gebildet wird.

Führt man dies für alle $\binom{t}{2}$ möglichen Paare von Zyklenüberdeckungen durch, so erhalten wir das folgende Lemma.

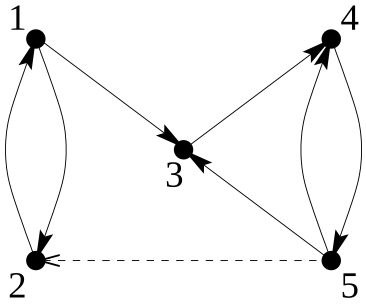

Abbildung 4: Durchgezogene Kanten haben Gewicht 1, gestrichelte Kanten haben Gewicht $\frac{3}{2}$ und nicht gezeichnete Kanten haben Gewicht 2. Eine optimale 3-Zyklenüberdeckung ist $(1,3,4,5,2)$ und hat Gewicht $\frac{11}{2}$. C_1 und C_2, bestehend aus den Zyklen $(1,2)$ und $(3,4,5)$ bzw. $(1,3,2)$ und $(4,5)$, sind eine zulässige Lösung von (3) und (4), wobei C_1 und C_2 jeweils mit $\frac{1}{2}$ gewichtet werden. C_1 hat Gewicht 5 und C_2 hat Gewicht 6.

Lemma 2.7 *Aus den Zyklenüberdeckungen $C_1, \ldots, C_t$ können in Polynomialzeit Zyklenüberdeckungen $C'_1, \ldots, C'_t$ berechnet werden, so daß das Gesamtgewicht von $C'_1, \ldots, C'_t$ höchstens $B \cdot W^*$ beträgt, $C'_1, \ldots, C'_t$ die Aussage von Lemma 2.6 erfüllen und für alle $1 \leq i, j \leq t$ keine Zusammenhangskomponente von $C'_i \cup C'_j$, die mehr als zwei Knoten hat, nur aus 2-Zyklen von C'_i und C'_j gebildet wird.* ■

Im folgenden nennen wir die Zyklenüberdeckung $C'_1, \ldots, C'_t$, die wir mittels des obigen Lemmas aus $C_1, \ldots, C_t$ erhalten, wiederum $C_1, \ldots, C_t$. Weiterhin nehmen wir an, daß deren Gesamtgewicht exakt $B \cdot W^*$ ist, da eine Reduktion des Gesamtgewichts nur die Approximationsgüte verbessert.

2.3.3 Berechnung guter partieller Zyklenüberdeckungen

Im Rest dieses Abschnitts wird nun aus $C_1, \ldots, C_t$ eine 0.999-gute partielle Zyklenüberdeckung gewonnen. Dazu führen wir eine Reihe von Parametern ein:

1. $w_{\min} = \min\{w(C_i) \mid 1 \leq i \leq t\}$ bezeichne das kleinste Gewicht einer Zyklenüberdeckung $C_1, \ldots, C_t$. (Abbildung 4 zeigt ein Beispiel, wo die $w(C_i)$ tatsächlich verschieden sind.)

2. Für alle $1 \leq i \leq t$ seien c_i und q_i so gewählt, daß $c_i \cdot n$ die Anzahl der Zyklen in C_i und $q_i \cdot n$ die Anzahl der 2-Zyklen in C_i ist. Offensichtlich gilt $0 \leq q_i \leq c_i \leq 1/2$. Es sei $\bar{c} = \frac{1}{B} \sum_{i=1}^{t} \gamma_i \cdot c_i$ die durchschnittliche Anzahl der Zyklen (gezählt mit den entsprechenden Multiplizitäten). In gleicher Weise sei $\bar{q} = \frac{1}{B} \sum_{i=1}^{t} \gamma_i \cdot q_i$.

3. $s_{i,j}$ sei so gewählt, daß $s_{i,j} \cdot n$ die Anzahl der 2-Zyklen ist, die sowohl in C_i als auch in C_j auftreten. Es seien

$$\bar{s}_i = \frac{1}{B-1} \sum_{\substack{j=1 \\ i \neq j}}^{t} \gamma_j s_{i,j} \qquad \text{sowie} \qquad \bar{s} = \frac{1}{(B-1)B} \sum_{\substack{i,j=1 \\ i \neq j}}^{t} \gamma_i \gamma_j s_{i,j}.$$

Bei der obigen Definition gehen wir davon aus, daß $B \geq 2$ gilt, da sonst die $x^*_{(u,v)}$ schon ganzzahlig sind und eine optimale 3-Zyklenüberdeckung vorliegt, die insbesondere 0.999-gut ist.

Für die weitere Analyse benötigen wir die folgenden Schranken für $\bar{s}_i$ und $\bar{s}$.

Lemma 2.8 *Für alle $1 \leq i \leq t$ ist $\bar{s}_i \leq q_i/2$. Weiterhin ist $\bar{s} \leq \bar{q}/2$.*

Beweis. Betrachten wir die Summe

$$(B-1)\bar{s}_i \cdot n = \sum_{\substack{j=1 \\ j \neq i}}^{t} \gamma_j s_{i,j} \cdot n.$$

Da jeder 2-Zyklus in C_i in höchstens $B/2 - 1$ anderen Zyklenüberdeckungen (gezählt mit Multiplizitäten) nach Lemma 2.6 vorkommen kann, gilt

$$(B/2-1) \cdot q_i \cdot n \geq \sum_{\substack{j=1 \\ j \neq i}}^{t} \gamma_j s_{i,j} \cdot n.$$

Daraus folgt

$$q_i \geq \frac{B-1}{B/2-1} \bar{s}_i \geq 2\bar{s}_i.$$

Dies zeigt die erste Ungleichung für den Fall, daß $B > 2$ gilt. Ist $B = 2$, so kann kein 2-Zyklus aus C_i in der anderen Zyklenüberdeckung auftreten. Folglich gilt $\bar{s}_i = 0$. Multipliziert man nun mit γ_i und summiert über alle i, so ergibt sich

$$B \cdot \bar{q} = \sum_{i=1}^{t} \gamma_i q_i \geq 2 \sum_{i=1}^{t} \gamma_i \bar{s}_i = 2B\bar{s},$$

was die zweite Ungleichung beweist. ∎

Wir unterscheiden nun eine Reihe von Fällen. In jedem von diesen Fällen wird eine 0.999-gute partielle Zyklenüberdeckung berechnet. Im ersten Fall ist ein Gewicht $w(C_i)$ signifikant kleiner als W^*. Im zweiten Fall hat ein C_i eine signifikante Zahl von Zyklen der Länge drei oder größer. Diese beiden Fälle sind leicht abzuhandeln. Im dritten Fall ist eines der $s_{j,k}$ klein. Der vierte Fall ist das Komplement der ersten drei Fälle und unterteilt sich in zwei Unterfälle.

Fall 1: Sei also $w_{\min} \leq 0.999 \cdot W^*$. Wir wählen ein j mit $w(C_j) = w_{\min}$. C_j ist eine 0.999-gute partielle Zyklenüberdeckung, da W^* höchstens so groß ist wie das Gewicht einer optimalen Hamiltonschen Tour und $0.999/(-\log c_j) \leq 0.999$ gilt.

In allen anderen Fällen können wir nun annehmen, daß mindestens ein Bruchteil von 0.95 der Überdeckungen $C_1, \ldots, C_t$ (gezählt mit Multiplizitäten) höchstens Gewicht $1.019 \cdot W^*$ haben. Mit anderen Worten, es gibt Indizes $j_1, \ldots, j_m$, so daß $w(C_{j_\mu}) \leq 1.019 \cdot W^*$ für alle $1 \leq \mu \leq m$ sowie $\sum_{\mu=1}^{m} \gamma_{j_\mu} \geq 0.95 \cdot B$ gilt. Dies sieht man wie folgt ein: Es gilt $\sum_{i=1}^{t} \gamma_i w(C_i) = B \cdot W^*$ sowie $w(C_i) > 0.999 \cdot W^*$ für alle $1 \leq i \leq t$. Würden mehr als ein Bruchteil von 0.05 der Zyklenüberdeckungen ein Gewicht von mehr als $1.019 \cdot W^*$ haben, so wäre das Gesamtgewicht größer als $B \cdot (0.05 \cdot 1.019 + 0.95 \cdot 0.999) \cdot W^* = B \cdot W^*$, ein Widerspruch.

Im folgenden bezeichnet $X = \{j_1, \ldots, j_m\}$ die Menge der oben definierten Indizes. Es sei $Y = \sum_{\mu=1}^{m} \gamma_{j_\mu} \geq 0.95 \cdot B$. Entsprechend den Größen $\bar{q}$, $\bar{s}_i$ und $\bar{s}$ definieren wir die Größen $\tilde{q}$, $\tilde{s}_i$ und $\tilde{s}$ bezüglich X:

1. $\tilde{q} = \frac{1}{Y} \sum_{i \in X} \gamma_i \cdot q_i$,

2. $\tilde{s}_i = \frac{1}{Y-1} \sum_{j \in X \setminus \{i\}} \gamma_j s_{i,j}$ und $\tilde{s} = \frac{1}{Y} \sum_{i \in X} \gamma_i \tilde{s}_i$.

Fall 2: Angenommen, es gibt ein $j \in X$ mit $c_j \leq 0.4931$. Dann ist C_j 0.999-gut, da $w(C_j) \leq 1.019$ nach Definition von X und $1.019/(-\log 0.4931) \leq 0.999$.

Fall 3: Gibt es $j, k \in X$ mit $j \neq k$, so daß $s_{j,k} \leq 0.2155$, so können wir gemäß dem nächsten Lemma C_j und C_k zu einer guten partiellen Zyklenüberdeckung vereinigen.

Lemma 2.9 *Für alle $j \neq k$ gibt es eine Zyklenüberdeckung von G mit Gewicht $\leq w(C_j) + w(C_k)$, die höchstens $(s_{j,k} + c_j + c_k - q_j - q_k) \cdot n$ Zyklen hat. Solch eine Zyklenüberdeckung kann aus C_j und C_k in Polynomialzeit gewonnen werden.*

Beweis. Betrachte die Kantenmenge $C_j \cup C_k$. Ihr Gesamtgewicht ist höchstens $w(C_j) + w(C_k)$. $C_j \cup C_k$ hat höchstens $s_{j,k} \cdot n$ 2-Zyklen (die dann sowohl in C_j als auch C_k sind). Alle anderen Teile von $C_j \cup C_k$ sind entweder längere Zyklen, die in beiden Überdeckungen vorkommen, oder Eulersche Zusammenhangskomponenten, die aus sich überlappenden Zyklen gebildet werden. Da wir die Zyklenüberdeckungen C_j und C_k gemäß Lemma 2.7 normalisiert haben, enthält jede der zuletzt genannten Eulerschen Zusammenhangskomponenten mindestens einen Zyklus der Länge drei oder größer. Insgesamt gibt es $(c_j - q_j) \cdot n$ solche Zyklen in C_j und $(c_k - q_k) \cdot n$ in C_k. Folglich gibt es höchstens ebenso viele längere Zyklen oder Eulersche Zusammenhangskomponenten in $C_j \cup C_k$. Wie in Lemma 2.2 können wir letztere durch jeweils einen großen Zyklus ohne zusätzliche Kosten ersetzen, da w die Dreiecksungleichung erfüllt. ∎

Es gilt $c_i - q_i \leq \frac{1}{3}(1 - 2q_i)$ für alle $1 \leq i \leq n$. Dies folgt aus der Tatsache, daß es $(1 - 2q_i) \cdot n$ Knoten in C_i gibt, die nicht zu einem 2-Zyklus und daher zu Zyklen der Länge mindestens drei gehören. Also gilt $q_i \geq 3c_i - 1$ für alle $1 \leq i \leq t$. Daraus folgt $c_j - q_j \leq 1 - 2c_j \leq 0.0138$, da wir uns sonst im Fall 2 befinden würden. Gleiches gilt auch für $c_k - q_k$. Damit hat die Zyklenüberdeckung aus Lemma 2.9 höchstens $(0.2155 + 0.0276) \cdot n = 0.2431 \cdot n$ Zyklen und Gewicht $\leq 2 \cdot 1.019 \cdot W^* = 2.038 \cdot W^*$. Daher ist die berechnete Überdeckung 0.999-gut, weil $2.038/(-\log 0.2431) \leq 0.999$.

Fall 4: Für alle i gilt

$$(Y - 1) \cdot \tilde{s}_i = \sum_{j \in X \setminus \{i\}} \gamma_j s_{i,j} \leq \sum_{\substack{j=1 \\ j \neq i}}^{t} \gamma_j s_{i,j} = (B - 1) \cdot \bar{s}_i,$$

folglich auch

$$\tilde{s}_i \leq \frac{B - 1}{Y - 1} \cdot \bar{s}_i \leq \frac{B - 1}{0.95 \cdot B - 1} \cdot \bar{s}_i.$$

Weil $\bar{s}_j \leq 1/4$ nach Lemma 2.8 für alle $1 \leq j \leq t$ gilt und wir angenommen haben, daß $B \geq 2$, folgt

$$\tilde{s}_i \leq \frac{1}{0.95 \cdot 2 - 1} \cdot \bar{s}_i \leq \frac{1}{0.9} \cdot 0.25 \leq 0.2778.$$

Also gibt es für jedes $j \in X$ ein $k \in X$, so daß $s_{j,k} \leq 0.2778$. Es bezeichnen j und k solch ein Paar. Da wir nicht in Fall 3 sind, gilt $s_{j,k} > 0.2155$.

Wir betrachten die folgenden zwei Unterfälle:

Fall 4a: Hier haben die 2-Zyklen, die sowohl in C_j als auch in C_k auftreten, Gesamtgewicht mindestens $0.338 \cdot W^*$. Wir fahren nun ähnlich wie in Fall 3 fort. Allerdings müssen wir das Gewicht der Zyklen, die sowohl in C_j als auch in C_k vorkommen, nur einmal zählen. (Im Fall 3 haben wir dies nicht berücksichtigt, da hier das Gewicht in diesen 2-Zyklen vernachlässigbar klein sein konnte.) Daher ist das Gesamtgewicht der berechneten Zyklenüberdeckung höchstens

$$w(C_i) + w(C_j) - 0.338 \cdot W^* \leq (2.038 - 0.338) \cdot W^* = 1.7 \cdot W^*.$$

Andererseits ist die Zahl der Zyklen in der berechneten Zyklenüberdeckung höchstens $(0.2778 + 2 \cdot 0.0138) \cdot n \leq 0.3054 \cdot n$. Daher ist die berechnete Überdeckung 0.999-gut, weil $1.7/(-\log 0.3054) \leq 0.999$.

Fall 4b: Haben hingegen die 2-Zyklen, die sowohl in C_j als auch in C_k auftreten, Gesamtgewicht $< 0.338 \cdot W^*$, so bilden diese Zyklen eine partielle Zyklenüberdeckung mit maximal $n - 0.2155 \cdot n = 0.7845 \cdot n$ Zyklen, da $s_{i,j} \geq 0.2155$, und Gewicht höchstens $0.338 \cdot W^*$. Folglich ist diese partielle Zyklenüberdeckung 0.999-gut, weil $0.338/(-\log 0.7845) \leq 0.9653 \leq 0.999$.

Da die obige Fallunterscheidung alle möglichen Fälle abdeckt, erhalten wir folgendes Theorem.

Theorem 2.10 *Es gibt einen Algorithmus für* 0.999-*GPCC mit polynomieller Laufzeit.* ■

Zusammen mit Theorem 2.3 beweist dies das folgende Korollar.

Korollar 2.11 *Es gibt einen polynomiellen* $0.999 \cdot \log n$*-Approximationsalgorithmus für* Δ*-ATSP.* ■

Bemerkung 2.12 *Alle Berechnungen wurden mit Hilfe zweier Computeralgebra-Systeme, die Hochpräzisionsnumerik unterstützten, (Mathematica, Maple) verifiziert.*

Fall 4b legt nahe, daß die Analyse weiter verbessert werden kann. Die dadurch zu erreichende Verbesserung ist allerdings nur marginal, der erreichbare Wert ist ≥ 0.9986.

2.4 Parametrisierte Dreiecksungleichung

Abschließend wenden wir uns dem Fall zu, in dem die Gewichtsfunktion w die sogenannte γ-parametrisierte Dreiecksungleichung für ein γ mit $\frac{1}{2} \leq \gamma < 1$ erfüllt, d.h. es gilt

$$w(u,v) \leq \gamma \cdot (w(u,x) + w(x,v)) \quad \text{für alle } u,x,v \in V.$$

Es existiert ein Approximationsalgorithmus für $\Delta(\gamma)$-ATSP, der konstante Approximationsgüte (in Abhängigkeit von γ) erreicht [19], nämlich (asymptotisch) $\frac{\gamma}{1-\gamma}$.

Wir zeigen hier, das auch schon eine geeignete Modifikation des Algorithmus von Frieze, Galbiati und Maffioli (siehe Abbildung 5) konstante Approximationsgüte erreicht. Die erzielte Approximationsgüte ist besser als $\frac{\gamma}{1-\gamma}$, der Güte des Algorithmus von Chandran und Ram [19], für alle Werte $\gamma \geq 0.6507$.

Dazu beweisen wir zuerst die folgenden Abschätzungen. Es seien im folgenden $G = (V, K(V))$ ein gerichteter schleifenfreier Graph mit einer Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$, die die γ-parametrisierte Dreiecksungleichung erfüllt für ein gegebenes γ mit $\frac{1}{2} \leq \gamma < 1$.

Lemma 2.13 *Es seien C eine 2-Zyklenüberdeckung von G und $V_1, V_2 \subseteq V$ so gewählt, daß V_1 und V_2 von jedem Zyklus aus C genau einen Knoten enthält und weiterhin $V_1 \cap V_2 = \emptyset$ gilt. Es seien G_1 und G_2 die von V_1 bzw. V_2 induzierten Subgraphen. Es seien L, L_1 und L_2 die Gewichte einer Hamiltonschen Tour minimalen Gewichts von G, G_1 bzw. G_2. Dann gilt $L_1 + L_2 \leq (1+\gamma^2) \cdot L$.*

Eingabe: gerichteter Graph $G = (V, K(V))$ mit Gewichtsfunktion w, wobei w die γ-parametrisierte Dreiecksungleichung erfüllt für ein $\frac{1}{2} \leq \gamma < 1$.
Ausgabe: eine Hamiltonsche Tour T.

1. Berechne eine 2-Zyklenüberdeckung C von G.
2. Wähle V_1 und V_2 so, daß aus jedem Zyklus von C genau ein Knoten in V_1 und V_2 enthalten ist und weiterhin $V_1 \cap V_2 = \emptyset$ gilt.
3. Berechne rekursiv jeweils Hamiltonsche Touren T_1 und T_2 in den Graphen G_1 und G_2, die jeweils von V_1 bzw. V_2 induziert werden (falls $|V_1| > 1$ und damit $|V_2| > 1$).
4. Kombiniere C und die leichtere der beiden Touren T_1 und T_2 zu einer Tour T wie in Lemma 2.14 beschrieben.

Abbildung 5: Approximationsalgorithmus für $\Delta(\gamma)$-ATSP

Beweis. Wir nehmen zuerst an, daß jeder Zyklus in C ein 2-Zyklus ist. Danach reduzieren wir den allgemeinen Fall auf diesen Spezialfall.

Es sei T eine optimale Hamiltonsche Tour von G und c ein 2-Zyklus aus C. Wir konstruieren nun aus T zwei Hamiltonsche Touren T_1 und T_2 von G_1 bzw. G_2, deren Gewicht zusammen höchstens das $(1 + \gamma^2)$-fache des Gewichts von T ist.

T kann auf zwei verschiedene Arten durch c laufen, wie in Abbildung 6 dargestellt. Es gehöre v_1 zu V_1 und v_2 zu V_2. Wir konstruieren aus T zwei Touren T_1 und T_2 von G_1 bzw. G_2, indem wir entsprechende Abkürzungen nehmen: Auf der linken Seite ersetzen wir im Fall von T_1 die Kanten f und g durch eine Kante, die von v_1 zum Endknoten von g zeigt. Ebenso ersetzen wir bei T_2 die Kanten e und f durch eine Kante, die vom Startknoten von e nach v_2 geht. Da w die γ-parametrisierte Dreiecksungleichung erfüllt, geht f in beiden Touren T_1 und T_2 mit dem Faktor $\gamma \cdot w(f)$ ein. Dies ergibt insgesamt Gewicht von $2\gamma \cdot w(f)$. Da die Kanten e und g noch mit einem Knoten eines anderen 2-Zyklus inzident sind, werden diese noch ein weiteres Mal kontrahiert. Daher geht e entweder in T_1 und T_2 jeweils mit Gewicht $\gamma \cdot w(e)$ ein oder in T_1 mit Gewicht $w(e)$ und in T_2 mit Gewicht $\gamma^2 \cdot w(e)$. Gleiches gilt für g, wobei die Rollen von T_1 und T_2 vertauscht sind. Da $1 + \gamma^2 \geq 2\gamma$ für alle γ gilt, folgt damit, daß $w(T_1) + w(T_2) \leq (1 + \gamma^2) \cdot L$.

Wird der 2-Zyklus wie auf der rechten Seite von Abbildung 6 dargestellt durchlaufen, so ersetzen wir zur Konstruktion von T_1 die Kanten g und h durch die Kante, die vom Startknoten von g zum Endknoten von h geht. Ebenso ersetzen

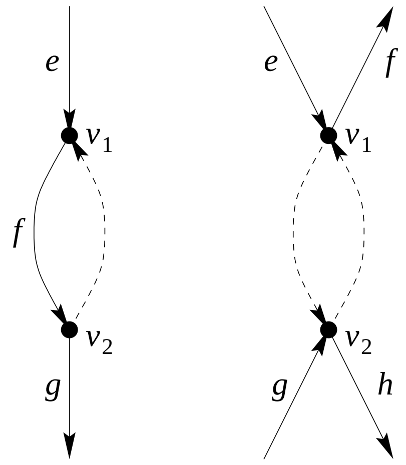

Abbildung 6: Zwei Möglichkeiten, wie eine Hamiltonsche Tour durch einen 2-Zyklus laufen kann.

wir in T_2 die Kante, die vom Startknoten von e zum Endknoten von f geht. Jede der Kanten wird ein zweites Mal kontrahiert, da sie mit einem weiteren 2-Zyklus inzident sind. Die Analyse entspricht der obigen Analyse des Falls auf der linken Seite.

Der allgemeine Fall folgt nun aus dem obigem Spezialfall wie folgt: Hat ein Zyklus c eine größere Länge als zwei, so ersetzen wir alle Kanten, die mit Knoten in c inzident sind, die nicht zu $V_1 \cup V_2$ gehören, durch entsprechende Abkürzungen. Dies verkleinert L nur. Danach haben alle Zyklen nur noch zwei Knoten und wir können obigen Spezialfall anwenden. ∎

Das nächste Lemma zeigt, wie eine rekursiv berechnete Tour sowie eine 2-Zyklenüberdeckung zu einer kompletten Hamiltonschen Tour kombiniert werden können.

Lemma 2.14 *Es sei C eine 2-Zyklenüberdeckung von G und $V' \subseteq V$ so gewählt, daß V' von jedem Zyklus in C genau einen Knoten enthält. Es sei T' eine Hamiltonsche Tour von G', wobei G' der von V' induzierte Graph ist. Dann kann in Polynomialzeit eine Hamiltonsche Tour T von G konstruiert werden, die Gewicht höchstens $w(C) + \gamma \cdot w(T')$ besitzt.*

Beweis. Der Graph $(V, C \cup T')$ ist stark zusammenhängend und sogar Eulersch. Eine Eulersche Tour sieht wie folgt aus: Sie besucht nacheinander die Knoten von T' und für jeden Knoten v von T' durchläuft sie dann jeweils den (eindeutigen) Zyklus c von C, zu dem v gehört, bevor der nächste Knoten von T' besucht wird. Aus dieser Eulerschen Tour konstruieren wir eine Hamiltonsche Tour T wie folgt: Die einzigen Knoten, die von der Eulerschen Tour zweimal besucht werden, sind die Knoten von T'. Wann immer wir einen Knoten von T' ein zweites Mal besuchen, nehmen wir stattdessen eine entsprechende Abkürzung. Damit wird jede Kante e von T' genau einmal abgekürzt, geht also in T nur mit Gewicht $\gamma \cdot w(e)$ ein. Dies beweist die Behauptung. (Es werden auch noch einige Kanten von C abgekürzt; diese können aber vernachlässigbares Gewicht haben, da nicht alle Kanten eines Zyklus abgekürzt werden, sondern nur eine. Daher ist nicht klar, wie man aus dieser Beobachtung weiteren Nutzen ziehen kann.) ∎

Nun können wir die Approximationsgüte unseres Algorithmus abschätzen.

Theorem 2.15 *Der Algorithmus aus Abbildung 5 liefert eine* $\frac{1}{1-\frac{1}{2}(\gamma+\gamma^3)}$*-Approximation einer Hamiltonschen Tour minimalen Gewichts von* G*. Seine Laufzeit ist polynomiell.*

Beweis. Der Beweis erfolgt per Induktion in der Anzahl der Knoten von G. Für einen Graphen G mit einem Knoten ist das Problem trivial.

Habe also G mehr als einen Knoten. Es sei L das Gewicht einer Hamiltonschen Tour minimalen Gewichts in G. Ebenso seien L_1 und L_2 die Gewichte von Hamiltonschen Touren minimalen Gewichts in G_1 bzw. G_2. Nach Lemma 2.13 gilt $L_1 + L_2 \leq (1+\gamma^2)L$, insbesondere hat eine der beiden Touren von G_1 bzw. G_2 höchstens Gewicht $\frac{1}{2}(1+\gamma^2) \cdot L$. O.B.d.A. gelte $L_1 \leq \frac{1}{2}(1+\gamma^2) \cdot L$.

Nach Induktionsvoraussetzung liefert der Algorithmus eine Tour T_1 mit

$$w(T_1) \leq \frac{1}{1-\frac{1}{2}(\gamma+\gamma^3)} \cdot L_1 \leq \frac{1+\gamma^2}{2-(\gamma+\gamma^3)} \cdot L. \tag{5}$$

Weiterhin können wir o.B.d.A. annehmen, daß T_1 auch die tatsächlich leichtere der beiden Touren ist, weil sonst für T_2 ebenfalls Ungleichung (5) gilt, da ja $w(T_2) \leq w(T_1)$.

Die in Schritt 4 aus C und T_1 berechnete Tour T hat nach Lemma 2.14 höchstens Gewicht

$$w(C) + \gamma \cdot w(T_1) \leq \left(1 + \gamma \cdot \frac{1+\gamma^2}{2-(\gamma+\gamma^3)}\right) \cdot L \leq \frac{2}{2-(\gamma+\gamma^3)} \cdot L.$$

Dies beweist die Aussage über die Approximationsgüte.

Es sei $Z(n)$ die Laufzeit des Algorithmus im Worst Case für Instanzen mit n Knoten. Es gilt $Z(1) = 1$ sowie

$$Z(n) \leq 2 \cdot Z(n/2) + O(n^3) \qquad \text{für alle } n > 1,$$

da sich jedes Problem in zwei Teilprobleme der Größe $\leq n/2$ aufteilt und pro Rekursionsschritt Aufwand $O(n^3)$ benötigt wird, um die 2-Zyklenüberdeckung C zu berechnen. Damit ergibt sich $Z(n) = O(n^3)$. ∎

Theorem 2.15 ist eine Verbesserung gegenüber dem von Chandran und Ram erzielten Ergebnis, wenn

$$\frac{\gamma}{1-\gamma} \geq \frac{1}{1-\frac{1}{2}(\gamma+\gamma^3)} \iff -\frac{\gamma^4}{2} - \frac{\gamma^2}{2} + 2\gamma - 1 \geq 0.$$

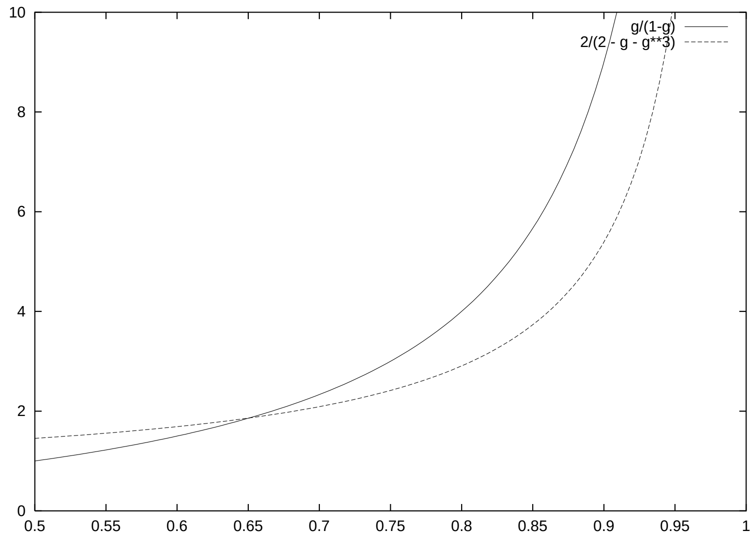

Abbildung 7: Die Approximationsgüte des Algorithmus aus Abbildung 5 (gestrichelt) im Vergleich zu der von Chandran und Ram erzielten Approximationsgüte (durchgezogen).

Die reellen Nullstellen des Polynoms $p(\gamma) := -\frac{\gamma^4}{2} - \frac{\gamma^2}{2} + 2\gamma - 1$ lassen sich exakt bestimmen. Insbesondere gilt $p(\gamma) \geq 0$ für $\gamma \in [0.6507, 1)$. Abbildung 7 vergleicht die erzielten Approximationsgüten anschaulich.

Bei der obigen Analyse des Algorithmus ist es entscheidend, daß jeder Zyklus durch einen Repräsentanten ersetzt wird, und so alle Gewichte „gleichmäßig" in jedem Rekursionsschritt um einen Faktor $\frac{1}{2}(1 + \gamma^2)$ schrumpfen (Lemma 2.13) bzw. nur mit Faktor γ eingehen (Lemma 2.14). Es ist unklar, ob und wie sich die Idee aus Abschnitt 2.2, auch partielle Zyklenüberdeckungen zuzulassen, nutzen läßt, um weitere Verbesserungen zu erzielen.

3 Zyklenüberdeckungen in Graphen mit Gewichten Eins und Zwei

In diesem Abschnitt bezeichnet G einen gerichteten Graphen $(V, K(V))$ oder auch einen ungerichteten Graphen $(V, U(V))$ mit einer Gewichtsfunktion w, die jeder Kante ein Gewicht eins oder zwei zuordnet.

Wir stellen zuerst einen generischen Algorithmus vor, um in G eine Zyklenüberdeckung minimalen Gewichts zu berechnen. Dieser wird dann in den folgenden Abschnitten an die Probleme k-DCC(1,2), k-UCC(1,2) und schließlich 3-Deg-TSP(1,2) angepaßt.

Eingabe: vollständiger schleifenfreier Graph G mit
Gewichtsfunktion $w : K(V) \to \{1, 2\}$ bzw.
Gewichtsfunktion $w : U(V) \to \{1, 2\}$,
ganze Zahl k.
Ausgabe: eine k-Zyklenüberdeckung von G.

1. Berechne eine Zyklenüberdeckung minimalen Gewichts C von G.
2. Bilde den bipartiten Graphen B und die zugehörige Funktion F.
3. Berechne eine Zerlegung von F.
4. Füge die Zyklen in C mit Hilfe der Kanten in F zu einer k-Zyklenüberdeckung zusammen.

Abbildung 8: Generischer Algorithmus

3.1 Generischer Algorithmus

Abbildung 8 zeigt den generischen Algorithmus. Die Eingabe besteht aus einem vollständigen schleifenfreien Graphen G, je nach Problem gerichtet oder ungerichtet. Die Knotenmenge V von G habe Kardinalität n. Zweitens ist eine Gewichtsfunktion w gegeben, die jeder Kante von G Gewicht eins oder zwei zuweist. Drittens besteht die Eingabe aus einer ganzen Zahl $k \leq n$. Das Ziel ist es, eine k-Zyklenüberdeckung minimalen Gewichts zu finden. (Formal ist k nicht Bestandteil der Eingabe sondern eine feste Konstante, die vom untersuchten Problem abhängt. Da aber sowohl Laufzeit als auch Approximationsgüte unabhängig von k sind, können wir k ohne Verlust zur Eingabe hinzunehmen.) Für $k > n/2$ erhält man die entsprechenden Travelling Salesperson Probleme als Spezialfälle.

Für die Analyse nehmen wir an, daß eine k-Zyklenüberdeckung minimalen Gewichts Gewicht $n + \ell$ habe für ein $0 \leq \ell \leq n$. Mit anderen Worten, eine optimale k-Zyklenüberdeckung hat $n-\ell$ Kanten mit Gewicht eins sowie ℓ Kanten mit Gewicht zwei.

Die nachfolgenden Unterabschnitte erläutern die einzelnen Schritte des generischen Algorithmus.

3.1.1 Berechnung einer Zyklenüberdeckung C

Zuerst wird eine Zyklenüberdeckung minimalen Gewichts C in G berechnet. Im Falle von gerichteten Graphen ist C eine 2-Zyklenüberdeckung, da diese in Polynomialzeit berechnet werden kann. Im ungerichteten Fall ist C eine 4-Zyklenüberdeckung, die mit Hilfe von Hartvigsens Algorithmus [36] in Polynomialzeit berechnet werden kann. C bestehe aus den Zyklen $z_1, \ldots, z_r$. Wir bezeichnen die Menge $\{z_1, \ldots, z_r\}$ mit $\mathcal{C}$. Die Länge einiger dieser Zyklen kann k oder mehr betragen, aber andere Zyklen, o.B.d.A. seien dies $z_1, \ldots, z_s$ mit $s \leq r$, haben Länge echt kleiner als k. Es sei $\mathcal{C}^< = \{z_1, \ldots, z_s\}$ die Menge der zu kurzen Zyklen. Im Falle von Travelling Salesperson Problemen besteht $\mathcal{C}^<$ aus allen Zyklen, es sei denn C ist schon eine Hamiltonsche Tour. (In letzterem Fall wären wir fertig.)

Die Grundidee des Algorithmus ist es, das sog. *subtour patching* zu verwenden (siehe z.B. Lawler et al. [50]). Im einfachsten Fall entfernen wir aus jedem Zyklus in $\mathcal{C}^<$ eine Kante, nach Möglichkeit mit Gewicht zwei, und fügen die entstandenen Pfade zu einem langen Zyklus zusammen. Im gerichteten Fall erhalten wir so eine $\frac{3}{2}$-Approximation, im ungerichteten Fall eine $\frac{5}{4}$-Approximation. (Die unterschiedlichen Approximationsfaktoren rühren von der unterschiedlichen minimalen Länge der Zyklen in C, zwei bzw. vier, her.) Die verbesserten Approximationsfaktoren erreichen wir im folgenden, indem beim Zusammenfügen der Pfade versucht wird, Pfade mittels Kanten vom Gewicht eins zu verbinden.

3.1.2 Zusätzliche Kanten

Wir bilden einen gerichteten bipartiten Graphen B wie folgt: die Knoten auf der linken Seite sind $\mathcal{C}^< = \{z_1, \ldots, z_s\}$, die Menge der zu kurzen Zyklen. Auf der anderen Seite befinden sich die Knoten aus V. Es gibt eine Kante (c, v) in B dann und nur dann, wenn v nicht zu c gehört in C und es einen Knoten u in c gibt, so daß (u, v) bzw. $\{u, v\}$ Gewicht eins in G hat. Solch eine Kante ist geeignet, um beim Zusammenfügen der Zyklen zusätzliche Kosten zu vermeiden.

Folgendes Lemma schätzt die Größe eines Matching in B ab. Es gilt unabhängig davon, ob G gerichtet oder ungerichtet ist.

Lemma 3.1 *B hat ein Matching der Kardinalität $\geq s - \ell$.*

Beweis. Es sei C_{opt} eine k-Zyklenüberdeckung minimalen Gewichts von G. Da die Länge der Zyklen in $\mathcal{C}^<$ echt kleiner ist als k, gibt es für jeden Zyklus in c in $\mathcal{C}^<$ eine Kante (u, v) bzw. $\{u, v\}$ in C_{opt}, so daß u zu c gehört aber v nicht. Jedem Zyklus aus $\mathcal{C}^<$ ordnen wir solch eine Kante zu. Dabei wird keine solche Kante zwei Zyklen zugeordnet, da nur Kanten betrachtet werden, die einen Zyklus verlassen. (Im Falle eines ungerichteten Eingabegraphen G geben wir dazu den Zyklen in C_{opt} eine beliebige Orientierung.) Nach Definition von ℓ haben mindestens $s - \ell$ dieser ausgewählten Kanten Gewicht eins, werden also in B eingefügt. Dort bilden sie ein Matching der Kardinalität mindestens $s - \ell$, da die Knoten v dieser Kanten auf der rechten Seite alle paarweise disjunkt sind, weil die Kanten aus einer Zyklenüberdeckung ausgewählt worden sind. ∎

3.1.3 Dekomposition von Funktionen

Wir berechnen als nächstes ein maximales Matching M von B. Mit Hilfe von M konstruieren wir einen gerichteten Graphen $F = (\mathcal{C}, A)$, wobei $(c, d) \in A$ ist, genau dann wenn es eine Kante $(c, u) \in M$ gibt, so daß u ein Knoten des Zyklus d ist. Jeder Knoten in F hat nach Konstruktion Ausgrad höchstens eins, also kann man F als partielle Funktion $\mathcal{C} \to \mathcal{C}$ auffassen, wobei der Definitionsbereich von F eine Teilmenge von $\mathcal{C}^{<}$ ist. Da keine Verwechslungsgefahr besteht, nennen wir die zugehörige Funktion wiederum F. Nach Konstruktion gilt $F(c) \neq c$ für alle $c \in \mathcal{C}$, mit anderen Worten, F ist schleifenfrei.

Mit Hilfe des folgenden Lemmas wird F in „einfache Bestandteile“ zerlegt, nämlich Bäume der Höhe eins, Pfade der Länge zwei sowie isolierte Knoten. Im Fall von totalen Funktionen F wird das Lemma bereits von Papadimitriou und Yannakakis [53, Lem. 2] bewiesen.

Lemma 3.2 *Jede schleifenfreie partielle Funktion F besitzt einen spannenden Subgraphen S, der nur aus knotendisjunkten Bäumen der Höhe eins, Pfaden der Länge zwei sowie isolierten Knoten besteht, so daß jeder Knoten im Definitionsbereich von F kein isolierter Knoten von S ist. Solch ein S kann in Polynomialzeit berechnet werden.*

Beweis. Jede schwach zusammenhängende Komponente von F ist entweder ein Zyklus, wobei möglicherweise zusätzlich einige Bäume in diesen Zyklus münden, oder ein Baum (dessen Wurzel r nicht im Definitionsbereich von F enthalten ist) oder ein isolierter Knoten (der nicht im Definitionsbereich von F enthalten ist). Es reicht, daß Lemma für jede schwach zusammenhängende Komponente von F zu beweisen.

Wir beginnen mit Bäumen. Wir wählen ein Blatt aus, das maximale Entfernung von der Wurzel hat. Es sei s der Nachfolger des Blattes. Falls $s = r$ gilt, so liegt ein Baum der Höhe eins vor und wir sind fertig. Sonst bilden wir einen Baum mit Wurzel s und allen Kindern von s als Blätter, entfernen diesen Baum und fahren rekursiv fort. Wir erhalten eine Kollektion von Bäumen der Höhe eins und möglicherweise einen isolierten Knoten, nämlich die Wurzel r. Diese ist jedoch nicht im Definitionsbereich von F enthalten und somit ist dieser Fall abgehandelt.

Im Falle eines Zyklus gehen wir wie folgt vor: Besteht die Komponente nur aus einem Zyklus, so können wir diesen in eine Kollektion von Pfaden der Länge eins und möglicherweise einem Pfad der Länge zwei zerlegen. Letzterer wird benötigt, falls der Zyklus ungerade Länge hat. Pfade der Länge eins können aber als Bäume der Höhe eins aufgefaßt werden.

Falls zusätzlich noch Bäume in den Zyklus münden, so werden diese Bäume wie weiter oben geschildert zerlegt. Werden dabei einige der Wurzeln der Bäume entfernt, so zerfällt der Zyklus in mehrere Pfade. Aber auch eine Menge von knotendisjunkten Pfaden läßt sich in eine Kollektion von Pfaden der Länge eins und möglicherweise der Länge zwei zerlegen.

Besteht die Komponente aus einem einzelnen Knoten, dann ist dieser Knoten nicht im Definitionsbereich von F enthalten, da F schleifenfrei ist.

Solch ein Graph S kann offensichtlich in Polynomialzeit berechnet werden. ■

3.1.4 Zusammenfügen der Zyklen

Die Zyklen von C werden wie folgt zusammengefügt: In jeder schwachen Zusammenhangskomponente K von S wird aus jedem Knoten von K, welches gerade die Zyklen in $\mathcal{C}$ sind, eine Kante entfernt und die Pfade zu einem größeren Zyklus zusammengefügt. Alle diese neuen Zyklen haben eine Kante mit Gewicht zwei (bzw. wir werden dieses Extragewicht in der Analyse einkalkulieren), daher können diese neuen Zyklen ohne weitere Kosten zu einem großen Zyklus zusammengefügt werden.

3.2 k-DCC(1,2)

Wir passen nun den generischen Algorithmus an die einzelnen Probleme an. Dabei beginnen wir mit k-DCC(1,2).

3.2.1 Berechnung einer Zyklenüberdeckung C

Im gerichteten Fall wird eine 2-Zyklenüberdeckung C minimalen Gewichts berechnet. Dies geht in Zeit $O(n^{5/2})$ durch Reduktion auf das maximale bipartite Matching Problem: Wie schon vorher beschrieben, bilden wir einen bipartiten Graphen mit der Knotenmenge V auf der einen Seite und einer Kopie v' für jeden Knoten $v \in V$ auf der anderen Seite. Für jede Kante (u, v) mit Gewicht eins in G fügen wir nun die Kante (u, v') nach B ein. Dadurch reduziert sich das Problem auf ein ungewichtetes bipartites Matching und kann so in Zeit $O(n^{5/2})$ (statt $O(n^3)$ im gewichteten Fall) berechnet werden.

3.2.2 Zusätzliche Kanten

Der Graph B und die Funktion F werden wie im generischen Algorithmus beschrieben berechnet.

3.2.3 Dekomposition von Funktionen

Ebenso wird ein spannender Subgraph S gemäß Lemma 3.2 berechnet.

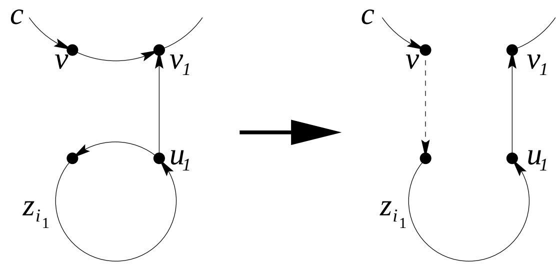

Abbildung 9: Bäume der Höhe eins.

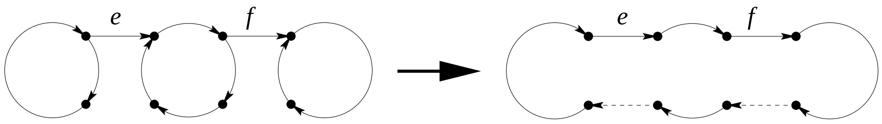

Abbildung 10: Pfade der Länge zwei.

3.2.4 Zusammenfügen der Zyklen

Die isolierten Knoten von S entsprechen Zyklen in $\mathcal{C}$, die nicht im Definitionsbereich von F liegen. Entweder hat solch ein Zyklus Länge mindestens k oder er ist ein nicht gematchter Zyklus aus $\mathcal{C}^<$. Die ersteren Zyklen, die wir im folgenden mit $\mathcal{C}_{\text{iso}}^{\geq}$ bezeichnen, sind schon lang genug, folglich brauchen wir diese im weiteren nicht in Betracht zu ziehen. Die letzteren Zyklen bezeichnen wir mit $\mathcal{C}_{\text{iso}}^<$. Diese haben Länge echt kleiner als k. Wir fügen diese Zyklen zu einem großen Zyklus d zusammen, wobei wir jeweils eine Kante mit Gewicht zwei aufbrechen, wann immer dies möglich ist.

Als nächstes betrachten wir die Bäume der Höhe eins in S. Es sei c die Wurzel eines solchen Baums und es seien $z_{i_1}, \ldots, z_{i_m} \in \mathcal{C}^< \setminus \mathcal{C}_{\text{iso}}^<$ dessen Blätter. Für jeden Zyklus z_{i_μ} gibt es eine Kante von (u_μ, v_μ) mit Gewicht eins in G, so daß u_μ zu z_{i_μ} und v_μ zu c gehört. Nach Konstruktion sind die Knoten $v_1, \ldots, v_m$ paarweise verschieden. Die Zyklen z_{i_1} und c werden wie in Abbildung 9 zusammengefügt. Wir nennen diesen Zyklus wiederum c und fügen nun c mit den übrigen Zyklen $z_{i_2}, \ldots, z_{i_m}$ iterativ in der gleichen Art und Weise zusammen. (Der Knoten v in Abbildung 9 kann dabei gleich $v_{\mu'}$ für ein $\mu' \neq \mu$ sein, dies spielt jedoch keine Rolle.) Danach fügen wir c mit d zusammen. In c entfernen wir dazu eine der Kanten, die in Abbildung 9 gestrichelt gezeichnet sind. In d brechen wir eine der Kanten auf, die nicht zu einem Zyklus in $\mathcal{C}_{\text{iso}}^<$ gehören, also während des Prozeß des Zusammenfügens zu d hinzugefügt wurden. Wir nennen den neuen Zyklus wiederum d.

Schließlich kommen wir zu den Pfaden der Länge zwei. Die drei Zyklen, die zu einem solchen Pfad gehören, werden wie in Abbildung 10 gezeigt zusammengefügt. (Der Endknoten der Kante e kann dabei der Startknoten von f sein.

Weiterhin können die beiden im mittleren Zyklus entfernten Kanten identisch sein. Dann entstehen nur zusätzliche Kosten von eins.) Der resultierende Zyklus wird mit d wie im Falle von Bäumen zusammengefügt.

Am Ende erhalten wir den Zyklus d sowie die Zyklen in $\mathcal{C}_{\text{iso}}^{\geq}$. Falls der Zyklus d immer noch Länge echt kleiner als k hat, so brechen wir einen Zyklus in $\mathcal{C}_{\text{iso}}^{\geq}$ auf und fügen ihn mit d zusammen. Falls möglich wählen wir diesen Zyklus so, daß er eine Kante von Gewicht zwei enthält und entfernen diese. Der resultierende Zyklus hat Länge mindestens k. Damit erhalten wir eine k-Zyklenüberdeckung.

3.2.5 Analyse des Algorithmus für k-DCC(1,2)

Der Algorithmus für k-DCC(1,2) hat offensichtlich polynomielle Laufzeit. Die Laufzeit wird dominiert von der Zeit, um die 2-Zyklenüberdeckung C sowie das maximale Matching M in B zu berechnen. Da beides ungewichtete Probleme sind, kann beides in Zeit $O(n^{5/2})$ implementiert werden. Die Dekomposition von F sowie das Zusammenfügen der Zyklen kann in Zeit $O(n^2)$ ausgeführt werden. Also ist die Gesamtlaufzeit $O(n^{5/2})$.

Das folgende Lemma schätzt die Approximationsgüte ab.

Lemma 3.3 *Für $n \geq 12$ hat die berechnete k-Zyklenüberdeckung maximal $\frac{4}{3}$ des Gewichts einer optimalen k-Zyklenüberdeckung.*

Beweis. Es sei C_{opt} eine optimale k-Zyklenüberdeckung von G. Weil C_{opt} auch eine 2-Zyklenüberdeckung von G ist, gilt natürlich $w(C) \leq w(C_{\text{opt}}) = n + \ell$. Das Gewicht $w(C_{\text{apx}})$ der k-Zyklenüberdeckung C_{apx}, die von dem Algorithmus ausgegeben wird, ist gerade $w(C)$ plus die Extrakosten, die durch das Zusammenfügen entstehen.

Wenn die Zyklen aus $\mathcal{C}_{\text{iso}}^{<}$ zusammengefügt werden, um den Zyklus d zu erhalten, so entstehen pro Zyklus $c \in \mathcal{C}_{\text{iso}}^{<}$ maximal Extrakosten eins.

Als nächstes betrachten wir das Zusammenfügen der Zyklen, die zu einem Baum der Höhe eins in S gehören. Pro Blatt eines Baums enstehen höchstens Extrakosten von eins. Die Kosten für das Zusammenfügen lasten wir dem Knoten v_μ und den Knoten aus z_{i_μ} aus Abbildung 9 an. Dies sind mindestens drei Knoten. Damit sind die Kosten für das Zusammenfügen höchstens 1/3 pro Knoten. Das Zusammenfügen von c und d ist kostenfrei, da wir nur Kanten aufbrechen, für die wir schon Extrakosten einkalkuliert haben.

In dem Fall von Abbildung 10, lasten wir die Kosten des Zusammenfügens allen Knoten der drei Zyklen an. Dies sind mindestens sechs Knoten und die Kosten betragen maximal zwei. Insgesamt sind dies wiederum Kosten höchstens 1/3 pro Knoten. Wie vorhin ist das Zusammenfügen mit d wiederum kostenlos.

Offensichtlich lasten wir jedem Knoten nur einmal Kosten an. Der Einfachheit halber gehen wir erst einmal davon aus, daß der Zyklus d Länge mindestens k hat, also kein weiteres Zusammenfügen notwendig ist. Es sei n_2 die Anzahl der

Knoten aus allen Zyklen in $\mathcal{C}_{\text{iso}}^{<}$. Das Gewicht $w(C_{\text{apx}})$ der k-Zyklenüberdeckung, die der Algorithmus ausgibt, beträgt höchstens

$$w(C_{\text{apx}}) \leq n + \ell + \tfrac{1}{3}(n - n_2) + |\mathcal{C}_{\text{iso}}^{<}|. \tag{6}$$

Es gilt $n_2 \geq 2 \cdot |\mathcal{C}_{\text{iso}}^{<}|$ und nach Lemma 3.1 $|\mathcal{C}_{\text{iso}}^{<}| \leq \ell$. Also folgt $w(C_{\text{apx}}) \leq \frac{4}{3}(n+\ell)$.

Hat d hingegen Länge echt kleiner als k, dann ist ein weiteres Zusammenfügen notwendig. Dies ergibt ad hoc eine Approximationsgüte von $4/3+1/n$, was unter asymptotischen Gesichtspunkten schon ausreichend ist. Eine genauere Analyse zeigt jedoch, daß auch in diesem Fall eine Güte von 4/3 erreicht wird: entweder ist das Zusammenfügen von dem Zyklus aus $\mathcal{C}_{\text{iso}}^{\geq}$ mit d kostenfrei, da dieser Zyklus aus $\mathcal{C}_{\text{iso}}^{\geq}$ bereits eine Kante mit Gewicht zwei enthält. Oder es bestehen alle Zyklen in $\mathcal{C}_{\text{iso}}^{\geq}$ nur aus Kanten mit Gewicht eins. Da dann $\mathcal{C}_{\text{iso}}^{\geq}$ nicht leer ist, sind dies mindestens $n/2$ Kanten. Der Zyklus d enthält mindestens die Hälfte aller Kanten aus den ursprünglichen Zyklen, aus denen d entstanden ist. Also enthalten d und die Zyklen aus $\mathcal{C}_{\text{iso}}^{\geq}$ mindestens 3/4 der Kanten aus der 2-Zyklenüberdeckung C. Daher hat die Zyklenüberdeckung C_{apx} Gewicht höchstens $\frac{5}{4}(n+\ell)+1 \leq \frac{4}{3}(n+\ell)$ für $n \geq 12$. ■

Theorem 3.4 *Es gibt einen $\frac{4}{3}$-Approximationsalgorithmus für k-DCC(1,2) mit einer Laufzeit von $O(n^{5/2})$ für alle $k \geq 3$.* ■

Korollar 3.5 *Es gibt einen $\frac{4}{3}$-Approximationsalgorithmus für ATSP(1,2) mit einer Laufzeit von $O(n^{5/2})$.* ■

Korollar 3.6 *Es gibt einen $\frac{2}{3}$-Approximationsalgorithmus für das Pfadpackungsproblem in gerichteten Graphen mit einer Laufzeit von $O(n^{5/2})$.*

Beweis. Die Grundidee ist, nicht vorhandene Kanten in dem gegebenen Graphen G durch Kanten von Gewicht zwei zu ersetzen und dann den $\frac{4}{3}$-Approximationsalgorithmus für ATSP(1,2) zu verwenden. Wie eingangs erwähnt, überträgt sich die Approximationsgüte in diese Richtung allerdings nicht so ohne weiteres.

Daher modifizieren wir den Algorithmus für ATSP(1,2) direkt, d.h. aus dem Minimierungsalgorithmus für Kantengewichte eins und zwei wird ein Maximierungsalgorithmus für Kantengewichte eins und null. Statt irgendein maximales Matching M bei der Konstruktion von F zu nehmen, berechnen wir nun ein maximales Matching, das mit möglichst wenig Knoten auf der rechten Seite von B inzident ist, die zu einem Zyklus gehören, der eine Kante vom Gewicht null besitzt. (Dies geht in Zeit $O(n^{5/2})$.) Eine komplizierte Analyse zeigt nun, daß diese Wahl von M zu einer $\frac{2}{3}$-Approximation führt (Bläser und Manthey, in Vorbereitung). ■

3.3 k-UCC(1,2)

In diesem Abschnitt passen wir den generischen Algorithmus aus Abbildung 8 für k-UCC(1,2) an. Insbesondere erhalten wir den $\frac{7}{6}$-Approximationsalgorithmus von Papadimitriou und Yannakakis [53] für TSP(1,2) als Spezialfall, indem wir $k > n/2$ wählen. Wir verwenden die gleichen Bezeichnungen wir für k-DCC(1,2).

3.3.1 Berechnung einer Zyklenüberdeckung C

Anstatt mit einer optimalen 2-Zyklenüberdeckung zu starten, berechnen wir hier nun eine 4-Zyklenüberdeckung C. Dies kann in Polynomialzeit erreicht werden mittels Hartvigsens Algorithmus [36]. Insbesondere gilt nun die Ungleichung $n_2 \geq 4 \cdot |\mathcal{C}_{\text{iso}}^{<}|$ (anstelle von $n_2 \geq 2 \cdot |\mathcal{C}_{\text{iso}}^{<}|$ im gerichteten Fall).

Der eigentliche Algorithmus von Hartvigsen berechnet ein maximales einfaches 2-Matching ohne 3-Zyklen maximaler Kardinalität [36]. Ein einfaches 2-Matching eines ungerichteten Graphen $H = (V, E)$ ist ein Subgraph M, so daß jeder Knoten Grad ≤ 2 hat. Mit anderen Worten, M besteht aus knotendisjunkten Zyklen und Pfaden (wobei isolierte Knoten als Pfade der Länge null angesehen werden). M ist ohne 3-Zyklen, wenn alle Zyklen von M Länge vier oder größer haben. M ist zusätzlich von maximaler Kardinalität, wenn kein anderes einfaches 2-Matching ohne 3-Zyklen mehr Kanten als M besitzt.

Es sei jetzt $H = (V, E)$ der Graph, der aus den Kanten mit Gewicht eins des Eingabegraphen $G = (V, U(V))$ besteht. Eine 4-Zyklenüberdeckung mit Gewicht $n + t$ von G ergibt ein einfaches 2-Matching minimalen Gewichts von H mit $n - t$ Kanten, indem man einfach die Kanten mit Gewicht zwei entfernt. Umgekehrt ergibt ein einfaches 2-Matching minimalen Gewichts M von H mit $n - t$ Kanten eine Zyklenüberdeckung C von G, indem man die Pfade von H mit Kanten von Gewicht zwei zu einem Zyklus zusammenfügt. Haben die Pfade von H zusammen mindestens vier Knoten, so hat jeder Zyklus Länge ≥ 4. In diesem Fall ist C eine 4-Zyklenüberdeckung mit Gewicht $n + t$. Haben die Pfade in H zusammen weniger als vier Knoten, so entsteht ein Zyklus der Länge ≤ 3. In diesem Fall brechen wir einen 4-Zyklus von C auf und fügen den so erhaltenen Pfad mit den übrigen Pfaden zusammen. Wir erhalten eine 4-Zyklenüberdeckung mit Gewicht $n+t+1$. Folglich können wir mit Hartvigsens Algorithmus 4-Zyklenüberdeckungen berechnen, deren Gewicht maximal um eins größer ist als eine optimale 4-Zyklenüberdeckung [38]. Da diese Extrakosten von eins bei der Bestimmung der asymptotischen Approximationsgüte keine Rolle spielen, lassen wir diese im folgenden außer Betracht, gehen also davon aus, daß wir eine optimale 4-Zyklenüberdeckung berechnet haben.

Weiterhin normalisieren wir die 4-Zyklenüberdeckung C wie folgt: O.B.d.A. bestehe C aus Zyklen $z_1, \ldots, z_r, z$, wobei z der einzige Zyklus ist, der Kanten vom Gewicht zwei enthält. (z kann natürlich auch fehlen.) Denn gibt es zwei Zyklen mit Kanten vom Gewicht zwei, so können diese beiden Zyklen ohne zusätzliche Kosten zu einem Zyklus zusammengefügt werden. Weiterhin können wir annehmen, daß für keine Kanten $\{u, v\}$ von z mit Gewicht zwei es eine Kante $\{v, x\}$ in G mit Gewicht eins gibt, wobei x ein Knoten ist, der nicht zu z gehört. Auch in diesem Fall kann man nämlich die beiden Zyklen, die $\{v, x\}$ verbindet, ohne Kosten zu einem Zyklus zusammengefügen. Es sei ζ die Anzahl der Kanten mit Gewicht zwei in z. Offensichtlich gilt $\zeta \leq \ell$.

3.3.2 Zusätzliche Kanten

Der Graph B und die Funktion F werden wie im generischen Algorithmus beschrieben berechnet, wobei z jedoch nicht auf der linken Seite von B auftaucht.

3.3.3 Dekomposition von Funktionen

Ebenso wird ein spannender Subgraph S gemäß Lemma 3.2 berechnet.

3.3.4 Zusammenfügen der Zyklen

Zyklen, die isolierten Knoten in S entsprechen, werden wie im gerichteten Fall zu einem großen Zyklus d zusammengefügt.

Für Bäume der Höhe eins verwenden wir ein komplizierteres Verfahren als im gerichteten Fall, das auf Papadimitriou und Yannakakis [53, Fig. 2] zurückgeht und nicht im gerichteten Fall funktioniert. Es seien c die Wurzel des Baumes und $z_{i_1}, \ldots, z_{i_m}$ seine Blätter. Zu jedem z_{i_μ} gibt es eine Kante $\{u_{i_\mu}, v_{i_\mu}\}$ vom Gewicht eins in G, so daß u_{i_μ} auf z_{i_μ} und v_{i_μ} auf c liegt. Nach Konstruktion sind $v_{i_1}, \ldots, v_{i_m}$ paarweise disjunkt. Wir teilen nun die $v_{i_1}, \ldots, v_{i_m}$ in Blöcke auf. Um von links und rechts sprechen zu können, geben wir c eine (beliebige) Orientierung. Die Knoten $v_{j_1}, \ldots, v_{j_p}$, $\{j_1, \ldots, j_p\} \subseteq \{i_1, \ldots, i_m\}$, bilden nun einen *Block*, falls jeweils $v_{j_{\rho+1}}$ der rechte Nachbar von v_{j_ρ} ist für alle $1 \leq \rho < p$ und der linke Nachbar von v_{j_1} sowie der rechte Nachbar von v_{j_r} keiner der Knoten $v_{i_1}, \ldots, v_{i_m}$ ist. (Im Fall, daß $v_{i_1}, \ldots, v_{i_m}$ ganz c sind, bilden $v_{i_1}, \ldots, v_{i_m}$ einen Block.) Wir fügen nun z_{j_1} und z_{j_2} mit c wie in Abbildung 11 zusammen. Ebenso fahren wir mit z_{j_3} und z_{j_4} fort usw. Ist r ungerade, so bleibt z_{j_r} dabei übrig. Diesen fügen wir dann mit c wie im gerichteten Fall (siehe Abbildung 9) zusammen.

Der Fall von Pfaden der Länge zwei wird wie im gerichteten Fall (siehe Abbildung 10) behandelt.

3.3.5 Analyse des Algorithmus für k-UCC(1,2)

Es sei wiederum C_{opt} eine optimale k-Zyklenüberdeckung von G. Da C_{opt} auch eine 4-Zyklenüberdeckung von G ist, gilt $w(C) \leq w(C_{\text{opt}}) = n + \ell$. Das Gewicht $w(C_{\text{apx}})$ der k-Zyklenüberdeckung C_{apx}, die von dem Algorithmus ausgegeben wird, ist gerade $w(C) = n + \zeta$ plus die Extrakosten, die durch das Zusammenfügen entstehen.

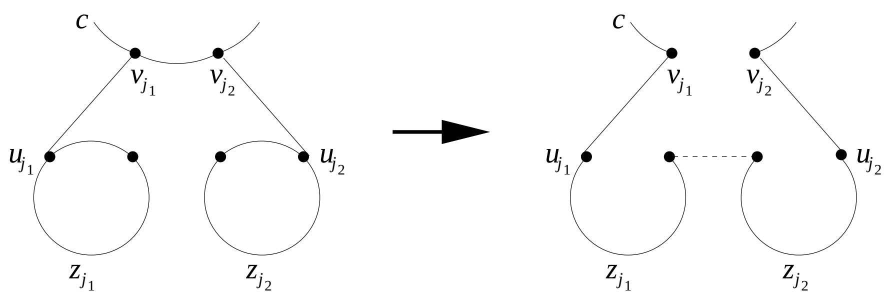

Abbildung 11: Bäume der Höhe eins im ungerichteten Fall.

Wenn die Zyklen aus $\mathcal{C}_{\text{iso}}^{<}$ zusammengefügt werden, um den Zyklus d zu erhalten, so entstehen pro Zyklus $c \in \mathcal{C}_{\text{iso}}^{<}$ Extrakosten eins.

Im Falle von Bäumen der Höhe eins lasten wir die Extrakosten eins, die beim Zusammenfügen von z_{j_1} und z_{j_2} mit c entstehen, den Knoten aus z_{j_1} und z_{j_2} sowie v_{j_1} und v_{j_2} an. Da C eine 4-Zyklenüberdeckung ist, sind dies mindestens zehn Knoten, was Extrakosten von $1/10$ pro Knoten entspricht. Ist r ungerade, so lasten wir die Kosten von eins, die entstehen, wenn man z_{j_r} mit c zusammenfügt, den Knoten von z_{j_r}, v_{j_r} sowie dem rechten Nachbarn von v_{j_r} an. Dies sind insgesamt mindestens sechs Knoten, also erhalten wir mittlere Kosten von $1/6$. (Sind die Knoten von c genau $v_{i_1}, \dots, v_{i_m}$ und ist m ungerade, so ist der rechte Nachbar von v_{i_m} wieder v_{i_1}, also ist die obige Analyse in diesem Fall so nicht möglich. Allerdings ist in diesem Fall $m \geq 5$, da C eine 4-Zyklenüberdeckung ist. Insgesamt gibt es Extrakosten von $\lceil m/2 \rceil$, die sich auf mindestens $4m + m = 5m$ Knoten verteilen. Für $m \geq 5$ ist aber $\frac{\lceil m/2 \rceil}{5m} \leq \frac{3}{25} < \frac{1}{6}$. Dieser Sonderfall wird im übrigen von Papadimitriou und Yannakakis außer acht gelassen.)

Liegt ein Pfad der Länge zwei vor, so entstehen Extrakosten maximal zwei, die auf mindestens zwölf Knoten verteilt werden. Dies ergibt mittlere Kosten von $\leq \frac{1}{6}$.

Da die Zyklenüberdeckung C normalisiert worden ist, brauchen wir nur für $n - n_2 - \zeta$ Knoten (anstelle von $n - n_2$) Extrakosten von $1/6$ einzurechnen. Dies liegt daran, daß mindestens ζ Knoten in z nach Konstruktion nicht gematcht sein können. Damit ergibt sich für das Gewicht von C_{apx}

$$w(C_{\text{apx}}) \leq n + \zeta + \tfrac{1}{6}(n - n_2 - \zeta) + |\mathcal{C}_{\text{iso}}^{<}| \leq \tfrac{7}{6}n + \tfrac{5}{6}\zeta + \tfrac{1}{3}|\mathcal{C}_{\text{iso}}^{<}| \leq \tfrac{7}{6}(n + \ell),$$

da $n_2 \geq 4 \cdot |\mathcal{C}_{\text{iso}}^{<}|$, $|\mathcal{C}_{\text{iso}}^{<}| \leq \ell$ sowie $\zeta \leq \ell$ gilt.

Die Laufzeit des Algorithmus ist polynomiell, da die Laufzeit von Hartvigsens Algorithmus polynomiell ist, nämlich $O(n^3)$. Alle weiteren Schritte können in Zeit $O(n^{5/2})$ oder schneller berechnet werden.

Theorem 3.7 *Es gibt einen $\frac{7}{6}$-Approximationsalgorithmus für k-UCC(1,2) mit Laufzeit $O(n^3)$ für $k \geq 5$.* ■

Korollar 3.8 (Papadimitriou & Yannakakis) *Es gibt einen $\frac{7}{6}$-Approximationsalgorithmus für TSP(1,2) mit Laufzeit $O(n^3)$.* ■

Bemerkung 3.9 *Statt mit einer 4-Zyklenüberdeckung zu starten, kann man auch nur mit einer 3-Zyklenüberdeckung beginnen. Diese läßt sich etwas schneller, in Zeit $O(n^{5/2})$, berechnen. Es ergibt sich damit ein $\frac{11}{9}$-Approximationsalgorithmus für k-UCC(1,2) und TSP(1,2), der Laufzeit $O(n^{5/2})$ hat. Letzteres Resultat für TSP(1,2) erzielen auch Papadimitriou und Yannakakis, wenn sie nicht Hartvigsens Algorithmus verwenden.*

3.4 3-Deg-TSP(1,2)

Als letztes wenden wir den generischen Algorithmus auf 3-Deg-TSP(1,2) an. Herzstück des Algorithmus ist eine Prozedur, die den spannenden Subgraphen S von F „verbessert". Dies ermöglicht eine bessere Approximationsgüte im Vergleich zum allgemeinen Problem TSP(1,2).

In diesem Abschnitt bezeichnet G einen Graphen $(V, U(V))$ mit einer Gewichtsfunktion $w : U(V) \to \{1, 2\}$. Im Graphen $H = (V, E)$, der aus den Kanten von G gebildet wird, die unter w Gewicht eins haben, habe jeder Knoten höchstens Grad drei.

3.4.1 Berechnung einer Zyklenüberdeckung C

Wie im Fall von k-UCC(1,2) berechnen wir zuerst eine 4-Zyklenüberdeckung C mit Hilfe von Hartvigsens Algorithmus [36]. Ebenso normalisieren wir C wie zuvor. Danach bezeichnen $z_1, \dots, z_r, z$ die Zyklen von C, wobei z der einzige Zyklus mit Kanten vom Gewicht zwei ist. (z kann natürlich auch fehlen.) Weiterhin gibt es für jede Kante $\{u, v\}$ von z mit Gewicht zwei keine Kante $\{v, x\}$ in G mit Gewicht eins, wobei x ein Knoten ist, der nicht zu z gehört. Es sei ζ die Anzahl der Kanten von Gewicht zwei, die zu z gehören.

Für die Analyse sei T eine Hamiltonsche Tour minimalen Gewichts. Es sei $n + \ell$ das Gewicht von T. Es sei ℓ' die Anzahl der Kanten $\{u, v\}$ von T, so daß u und v zu unterschiedlichen Zyklen gehören und $w(\{u, v\}) = 2$ gilt. Offensichtlich gilt

$$\ell' \leq \ell \qquad \text{und} \qquad \zeta \leq \ell. \tag{7}$$

3.4.2 Zusätzliche Kanten

Der Graph B und die Funktion F werden wie im generischen Algorithmus beschrieben berechnet.

3.4.3 Dekomposition von Funktionen

Im allgemeinen Fall setzt sich S aus Bäumen der Höhe eins, Pfaden der Länge zwei sowie isolierten Knoten zusammen. Dabei kann man einen Pfad der Länge zwei nicht als Baum der Höhe eins auffassen: Betrachten wir zuerst einen Baum der Höhe eins von S mit Wurzel c und Blättern $c_1, \ldots, c_s$. Jeder Kante (c_σ, c) in S, $1 \leq \sigma \leq s$, entspricht eine Kante $\{u_\sigma, v_\sigma\}$ in G, so daß u_σ ein Knoten von c_σ und v_σ ein Knoten von c ist. Nach Konstruktion von B und F gilt $v_\sigma \neq v_{\sigma'}$ für alle $\sigma \neq \sigma'$. Im Falle eines Pfades c_1, c_2, c_3 der Länge zwei in S mit entsprechenden Kanten $\{u_1, u_2\}$ und $\{v_2, v_3\}$ von Gewicht eins in G, wobei u_1 zu c_1, u_2 und v_2 zu c_2 sowie v_3 zu c_3 gehört, muß nicht $u_2 \neq v_2$ gelten.

Diese Beobachtung gilt jedoch nicht mehr, wenn der Graph H beschränkten Grad drei hat. Nach Konstruktion von B und F hat der mittlere Zyklus c_2 nur Kanten vom Gewicht eins. Daher würde $u_2 = v_2$ der Gradbeschränkung von H widersprechen, da auch die Kanten $\{u_1, u_2\}$ und $\{v_2, v_3\}$ Gewicht eins haben. Daher können wir im Falle von 3-Deg-TSP(1,2) Pfade der Länge zwei als Bäume der Höhe eins ansehen. (Formal ersetzen wir dazu die gerichtete Kante (c_2, c_3) durch (c_3, c_2).)

Zusammen ergibt sich mit Lemma 3.2 aus obigen Überlegungen folgendes Dekompositionsresultat.

Lemma 3.10 *Falls H beschränkten Grad drei hat, so besitzt jede schleifenfreie partielle Funktion F einen spannenden Subgraphen S, der nur aus knotendisjunkten Bäumen der Höhe eins sowie isolierten Knoten besteht, so daß jeder Knoten im Definitionsbereich von F kein isolierter Knoten von S ist. Solch ein S kann in Polynomialzeit berechnet werden.*

Im Rest dieses Unterabschnitts wird nun der spannende Subgraph S Schritt für Schritt „verbessert". Bei der Analyse dieser Verbesserungsprozedur spielt die Tatsache, daß H beschränkten Grad drei hat, eine entscheidende Rolle.

Wir führen zuerst eine Reihe von Definitionen und Notationen ein. Da im folgenden oft zwischen Knoten des Graphen G sowie des Graphen S (bzw. F) unterschieden wird, nennen wir die Knoten von S Z-Knoten bzw. Z-Blätter, da diese Zyklen sind, die wiederum aus Knoten von G bestehen. Wie schon weiter oben erwähnt, entspricht jeder Kante (c, d) in S eine Kante $\{u, v\}$ in G vom Gewicht eins, so daß u ein Knoten von c und v ein Knoten von d ist. Diese Kante in G bezeichnen wir im folgenden mit $ed(c, d) = \{u, v\}$. Der Z-Knoten d ist der Vater von c in S. Es sei $par(c) = v$. Um für die Beschreibung des Algorithmus von links und rechts sprechen zu können, geben wir jedem der Zyklen $z_1, \ldots, z_r, z$ eine beliebig gewählte Orientierung.

Ein Knoten $v \in V$ ist *gematcht*, falls er zu $ed(e)$ für eine Kante e in S gehört. Sei v ein gematchter Knoten und c der Zyklus, zu dem v gehört. Es seien $v_0, v_1, \ldots, v_{t-1}$ die Knoten von c (in dieser Reihenfolge) und es gelte $v = v_0$. Es sei $i > 0$ der kleinste Index, so daß v_i nicht gematcht ist. In gleicher Weise sei

$j > 0$ der kleinste Index, so daß v_{t-j} nicht gematcht ist. Für solch einen Knoten v definieren wir $M(v)$ als $i+j-1$. Mit anderen Worten, wenn wir die Knoten von c in zusammenhängende Blöcke von gematchten Knoten einteilen, dann gibt $M(v)$ gerade die Länge des Blocks an, zu dem v gehört. Existiert solch ein Index i nicht (und damit auch j nicht), dann sind alle Knoten von c gematcht. In diesem Fall ist $M(v)$ einfach die Anzahl der Knoten in c. Für Knoten v, die nicht gematcht sind, setzen wir $M(v) = 0$.

Als nächstes führen wir den Begriff der guten Knoten und guten Z-Blätter ein. Gute Z-Bätter sind eine Verallgemeinerung des Falls, wie er in Abbildung 11 dargestellt ist.

Definition 3.11

1. *Es sei c ein Z-Knoten, dessen Knoten nicht alle gematcht sind. Ein gematchter Knoten v von c heißt* gut, *falls entweder $M(v)$ gerade ist oder $M(v)$ ungerade ist und der rechte Nachbar von v in c gematcht ist. Sonst heißt v* schlecht. *Sind alle Knoten von c gematcht und ist die Anzahl der Knoten in c gerade, so sind alle Knoten in c gut. Ist die Anzahl der Knoten ungerade, so wählen wir willkürlich einen Knoten aus c aus. Dieser Knoten ist schlecht, alle übrigen sind gut.*

2. *Ein Z-Blatt c in S heißt* gut, *falls der Knoten $par(c)$ gut ist. Sonst heißt c* schlecht.

Damit das Zusammenfügen der Zyklen gemäß des Graphen S funktioniert, muß S eine spezielle Form haben:

Definition 3.12 *Ein gerichteter Graph S heißt* gültig, *falls er nur aus Bäumen (mit mindestens zwei Z-Knoten) sowie isolierten Z-Knoten besteht, so daß*

1. *für alle Z-Knoten c und d eines Baumes, die beide nicht die Z-Wurzel sind, $par(c) \neq par(d)$ gilt,*

2. *für alle Z-Knoten c eines Baumes, der weder ein Z-Blatt noch eine Z-Wurzel ist, $par(c)$ gut ist und*

3. *der Vater jedes schlechten Z-Blattes die Z-Wurzel eines Baumes ist.*

Weiterhin hat $ed(e)$ Gewicht eins in G für alle Kanten e in S.

Der ursprüngliche Graph S, der in Lemma 3.10 berechnet wird, ist gültig. Dies liegt daran, daß alle Bäume in S Höhe eins haben. Daher ist die zweite und dritte Bedingung der Definition trivialerweise erfüllt.

Um die weiteren Ergebnisse präzise formulieren zu können, assoziieren wir eine Reihe von Parametern mit einem gültigen Graphen S:

1. b bezeichnet die Gesamtzahl aller Z-Blätter in allen Bäumen mit mindestens drei Z-Knoten.

Eingabe: ein gültiger Graph S.
Ausgabe: ein gültiger Graph, der keine weitere Verbesserung erlaubt.

1. Normalisiere die Bäume mit drei Z-Knoten wie in Lemma 3.14 beschrieben.

2. Wähle ein schlechtes Z-Blatt eines Baums mit mindestens drei Z-Knoten oder einen isolierten Z-Knoten c und eine Kante $\{u, v\}$ mit Gewicht eins in G, so daß u zu c und v nicht zu c gehört und weiterhin v verbessernd ist.

3. Falls solch ein c existiert, dann restrukturiere S wie in Lemma 3.14 beschrieben und beginne erneut mit Schritt 1. Sonst gebe S aus.

Abbildung 12: Verbesserungsprozedur

2. g bezeichnet die Gesamtzahl aller guten Z-Blätter in allen Bäumen mit mindestens drei Z-Knoten.

3. t ist die Anzahl der Bäume mit nur zwei Z-Knoten.

4. s ist die Anzahl der isolierten Z-Knoten.

Beim Zusammenfügen der Zyklen spielt die Größe $b - \frac{1}{2}g + t + s$ eine entscheidende Rolle (siehe Lemma 3.15). Abbildung 12 stellt eine Prozedur dar, die den ursprünglichen Graphen S schrittweise in bezug auf obige Größe verbessert. Die Korrektheit der Verbesserungsprozedur wird in Lemma 3.14 bewiesen. Für die Verbesserungsprozedur benötigen wir noch die folgende Definition.

Definition 3.13 *Ein nicht gematchter Knoten v eines Z-Knotens c heißt* verbessernd*, falls eine der nachfolgenden Bedingungen erfüllt ist:*

1. *c ist ein innerer Z-Knoten eines Baumes und, wenn x und y die Nachbarn von v in c sind, so ist $M(x)$ oder $M(y)$ ungerade.*

2. *c ist ein Z-Blatt eines Baumes mit mindestens drei Z-Knoten und $par(c)$ ist schlecht.*

3. *c ist ein Z-Knoten eines Baumes mit zwei Z-Knoten und v ist einer der beiden Nachbarn des gematchten Knotens von c.*

4. *c ist ein isolierter Z-Knoten.*

Das folgende Lemma zeigt, wie aus einem gegebenen gültigen Subgraphen S ein neuer gültiger Subgraph S' gewonnenen werden kann, der in Bezug auf die Größe $b - \frac{1}{2}g + t + s$ besser ist als S.

Um die Bedingung zu formulieren, unter denen man solch ein S' finden kann, werden noch die folgenden beiden Größen benötigt: Zu einem gegebenen S sei

1. m die Gesamtzahl der gematchten Knoten in S und

2. i die Anzahl aller verbessernden Knoten von S.

Lemma 3.14 *Es sei S gültig. Solange $n - m - i - \zeta < b - g + 2s - \ell'$ gilt, kann in Polynomialzeit ein gültiger Subgraph S' gefunden werden, so daß*

$$b' - \tfrac{1}{2}g' + t' + s' < b - \tfrac{1}{2}g + t + s,$$

wobei die Parameter b', g', t' und s' von S' wie für S definiert sind.

Beweis. Der Beweis des Lemmas gliedert sich in zwei Teile. Zuerst schätzen wir ab, wann eine Verbesserung möglich ist. Danach folgt der eigentliche Verbesserungsschritt.

Abschätzung: Wir diskutieren zuerst den Fall, in dem eine optimale Hamiltonsche Tour Gewicht n hat. Mit anderen Worten, es gilt $\ell = 0$ und damit auch $\ell' = 0$ und $\zeta = 0$ via (7). Mit jedem schlechten Z-Blatt eines Baums mit mindestens drei Knoten sowie mit jedem isolierten Z-Knoten assoziieren wir nun Knoten (aus V).

Mit jedem schlechten Z-Blatt c eines Baums in S mit mindestens vier Knoten assoziieren wir einen Knoten eines Zyklus ungleich c wie folgt: Da eine Hamiltonsche Tour jedes Z-Blatt mindestens einmal betritt und einmal verläßt, c in S aber nur mit einer Kante inzident ist, gibt es eine Kante $\{u, v\}$ vom Gewicht eins in G, so daß u zu c gehört, aber v nicht und weiterhin $\{u, v\} \neq ed(c, c')$ für alle $c' \neq c$. (Wir haben angenommen, daß es eine optimale Hamiltonsche Tour gibt, die nur aus Kanten vom Gewicht eins besteht.) Wir assoziieren v mit c.

In derselben Manier können wir mit jedem isolierten Z-Knoten c sogar zwei Knoten außerhalb von c assoziieren, weil keine Kante in S mit c inzident ist.

Komplizierter ist der Fall eines Baums mit genau drei Z-Knoten: Es seien c_1, c_2 und c_3 die Z-Knoten solch eines Baums und c_1 sei die Z-Wurzel. Da S gültig ist, sind dann c_2 und c_3 Z-Blätter. Ist eines dieser Z-Blätter schlecht, so auch das andere. Wieder möchten wir sowohl mit c_2 als auch mit c_3 einen Knoten assoziieren. Dabei soll allerdings aus Gründen, die im Verlauf der Verbesserungsprozedur klar werden, der mit c_2 assoziierte Knoten nicht zu c_3 gehören und umgekehrt. Eine optimale Hamiltonsche Tour könnte jedoch z.B. den Baum durch c_1 betreten, nach c_2 gehen, dann nach c_3, wieder nach c_1 und den Baum verlassen (siehe auch Abbildung 13). Hier gibt es keine Kante vom Gewicht eins, die c_2 verläßt und nicht in c_3 endet und umgekehrt. Dafür gibt es aber zwei Kanten vom Gewicht eins, die c_1 verlassen und weder in c_2 noch c_3 enden.

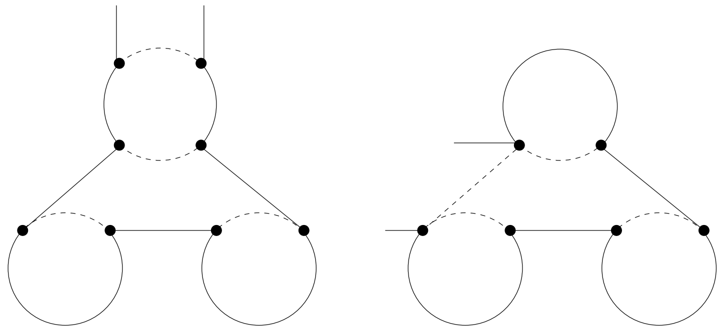

Abbildung 13: Falls eine optimale Hamiltonsche Tour (durchgezogen) einen Baum in S mit drei Z-Knoten wie hier dargestellt durchläuft, so muß der Baum umstrukturiert werden. Links wird eines der beiden Z-Blätter die neue Z-Wurzel, die Z-Wurzel wird ein Z-Blatt. Rechts wir das rechte Z-Blatt die neue Z-Wurzel, die Z-Wurzel ein Z-Blatt. Die Knoten, die man über die beiden Kanten der Hamiltonschen Tour, die den Baum verlassen, erreicht, werden jeweils danach mit den neuen Z-Blättern assoziiert.

Wir strukturieren den Baum wie folgt um: Falls es keine Kante vom Gewicht eins zwischen c_2 und c_3 gibt, so muß es, da eine optimale Hamiltonsche Tour nur Kanten vom Gewicht eins hat, eine Kante vom Gewicht eins geben, die c_2 verläßt, aber nicht c_3 betritt und von $ed(c_2, c_1)$ verschieden ist. Ebenso gibt es eine Kante mit Gewicht eins, die c_3 verläßt aber nicht c_2 betritt und von $ed(c_3, c_1)$ verschieden ist. In diesem Fall ist nichts zu tun und wir assoziieren mit c_2 und c_3 die entsprechenden Endknoten der beiden gefundenen Kanten.

Gibt es jedoch eine Kante vom Gewicht eins zwischen c_2 und c_3, so hat entweder nach obiger Überlegung einer der drei Z-Knoten zwei inzidente Kanten vom Gewicht eins (in G), die den Z-Knoten verlassen, deren anderer Endknoten aber in keinem der drei Z-Knoten liegt (Abbildung 13 links). Oder zwei der Z-Knoten haben je eine Kante vom Gewicht eins (in G), die den jeweiligen Z-Knoten verläßt und keinen anderen Z-Knoten des Baums betritt (Abbildung 13 rechts). Im ersten Fall wird einer der beiden anderen Z-Knoten die neue Z-Wurzel und die beiden übrigen werden Z-Blätter. (Dies ist möglich wegen der Kante vom Gewicht eins zwischen c_2 und c_3.) Im zweiten Fall wird der dritte Z-Knoten die neue Wurzel und die beiden anderen die Z-Blätter. In beiden Fällen assoziieren wir die beiden Knoten von G mit den neuen Z-Blättern wie im Abbildung 13 beschrieben. Im Durchschnitt assoziieren wir einen Knoten pro schlechtem Z-Blatt. (Bei dem Umstrukturieren können auch beide neuen Z-Blätter gut sein. In diesem Fall sind wir fertig.)

Da H beschränkten Grad drei hat und nach Annahme die Zyklen in C nur aus Kanten vom Gewicht eins bestehen, müssen alle assoziierten Knoten paarweise verschieden sein. Nach obigen Überlegungen gibt es daher mindestens $b - g + 2s$

assoziierte Knoten. Keiner von diesen kann ein gematchter Knoten sein, weil H beschränkten Grad drei hat. Falls nun $n - m - i < b - g + 2s$, dann muß einer der assoziierten Knoten ein verbessernder Knoten sein.

Falls eine optimale Hamiltonsche Tour nicht Gewicht n hat, dann gibt es nach der Definition von ℓ' noch mindestens $b-g+2s-\ell'$ assoziierte Knoten. Diese sind alle paarweise verschieden, selbst wenn sie zu z gehören, da wir C normalisiert haben. Solange $n-m-i-\zeta < b-g+2s-\ell'$ gilt, ist einer der assoziierten Knoten verbessernd, da mindestens ζ Knoten von z nicht mit Kanten vom Gewicht eins inzident sind, deren zweiter Knoten nicht zu z gehört.

Verbesserungsschritt: Wir kommen nun zum eigentlichen Verbesserungsschritt: Wir wählen ein schlechtes Z-Blatt eines Baums mit mindestens drei Z-Knoten bzw. einen isolierten Z-Knoten c und eine Kante $\{u, v\}$ vom Gewicht eins aus, so daß u zu c gehört, v aber nicht und v verbessernd ist. Es sei d der Z-Knoten, zu dem v gehört. Wir entfernen die Kante in S, die inzident mit c ist, sofern c nicht isoliert ist, und fügen die Kante (c, d) ein. Es sei S' der resultierende Graph. Wir behaupten, daß danach $b - \frac{1}{2}g + t + s > b' - \frac{1}{2}g' + t' + s'$ gilt.

Dazu unterscheiden wir vier Fälle, jeweils einen für jeden der vier Fälle in Definition 3.13.

Wir nehmen zuerst an, daß c zu einem Baum mit mindestens vier Z-Knoten gehört. In diesem Fall hat der Baum nach Entfernen von c immer noch mindestens drei Z-Knoten.

Im ersten Fall ist $M(x)$ oder $M(y)$ ungerade, wobei x und y die Nachbarn von v sind. In S' ist c nun ein gutes Z-Blatt und ein weiteres Z-Blatt wird dadurch auch gut. Also gilt $g' = g + 2$ und $b = b'$. Die Parameter s' und t' sind gleich s bzw. t. (Im folgenden werden nur die Parameter aufgelistet, die sich ändern.)

Im zweiten Fall gehört d zu einem Baum T mit mindestens drei Z-Knoten und $par(d)$ ist schlecht. Wir entfernen d aus diesem Baum und bilden einen neuen Baum mit Z-Blatt c und Z-Wurzel d. Da $par(d)$ schlecht war, ändert sich die Anzahl der guten Z-Blätter dadurch nicht. Hat T mehr als drei Z-Knoten, so hat der resultierende Baum noch mindestens drei Z-Knoten. Die Anzahl der Z-Blätter von Bäumen mit mindestens drei Knoten sinkt um zwei (nämlich c und d werden entfernt), also gilt $b' = b-2$. (Der Vater von c kann durch das Entfernen von c nicht zu einem Z-Blatt werden, denn da S gültig ist, ist der Vater von c eine Z-Wurzel.) Wir erhalten einen neuen Baum mit zwei Knoten, d.h. $t' = t+1$. Einzige Ausnahme ist der Fall, in dem T genau vier Z-Knoten hat und c auch zu T gehört. In diesem Fall erhalten wir zwei Bäume mit zwei Knoten, also gilt $t' = t + 2$. Dafür gilt $b' = b - 3$, da wir drei Z-Blätter weniger haben, die zu einem Baum mit mindestens drei Knoten gehören. Hat hingegen T genau drei Z-Knoten, so entsteht aus T ein Baum mit zwei Knoten. Es gibt also zwei Bäume mit zwei Z-Knoten mehr, also $t' = t + 2$. Dafür gibt es drei Z-Blätter an Bäumen mit mindestens drei Z-Knoten weniger (nämlich die beiden Z-Blätter von T sowie c), also $b' = b - 3$.

Im dritten Fall erhalten wir einen Baum mit drei Z-Knoten, der zwei gute Z-

Blätter hat. Dabei machen wir d zur neuen Z-Wurzel des Baumes. Es gilt $t' = t-1$, $g' = g + 2$ und $b' = b + 1$.

Im letzten Fall erhalten wir einen Baum mit zwei Z-Knoten. Der Z-Knoten c ist nicht mehr ein Z-Blatt eines Baumes mit mindestens drei Blättern. Also gilt $b' = b - 1$, $t' = t + 1$ und $s' = s - 1$.

Als nächstes betrachten wir die obigen vier Fälle, wenn c zu einem Baum mit genau drei Z-Knoten gehört. In diesem Fall erhalten wir einen Baum mit nur zwei Z-Knoten. Damit verringert sich im Vergleich zu oben b' um eins mehr (denn das zweite Z-Blatt des Baumes gehört nach dem Entfernen von c nicht mehr zu einem Baum mit mindestens drei Knoten), dafür ist t' um eins größer. In der Summe gleicht sich dies aus.

Ist c ein isolierter Knoten, so verringert sich hingegen im Vergleich zu oben s' um eins mehr. Dafür nimmt b' um eins weniger ab, weil mit c nicht ein Blatt eines Baumes mit mindestens drei Knoten entfernt wird. In der Summe gleicht sich dies ebenfalls aus.

Damit gilt also $b - \frac{1}{2}g + t + s > b' - \frac{1}{2}g' + t' + s'$.

Nach Konstruktion ist der so konstruierte Subgraph S' auch gültig. ∎

3.4.4 Zusammenfügen der Zyklen

Das folgende Lemma liefert eine Prozedur zum Zusammenfügen der Zyklen, die eine Verallgemeinerung der Prozedur für k-UCC(1,2) ist. Das Gewicht der berechneten Tour hängt von $b - \frac{1}{2}g + t + s$ ab. Daher ist ein Subgraph S besonders gut, wenn letztere Größe klein ist.

Lemma 3.15 *Aus einem gegebenen gültigen Subgraphen S kann eine Hamiltonsche Tour mit Gewicht $n + \zeta + b - \frac{1}{2}g + t + s$ in Polynomialzeit berechnet werden.*

Beweis. Zuerst fügen wir die Zyklen, die jeweils Z-Knoten einer Zusammenhangskomponente von S sind, zu einem großen Zyklus zusammen. Da jeder dieser großen Zyklen eine Kante mit Gewicht zwei enthält (bzw. wir zumindest dieses Extragewicht einkalkuliert haben), können wir danach diese großen Zyklen ohne neue Kosten zu einer Hamiltonschen Tour zusammenfügen.

Falls die Komponente ein Baum mit zwei Z-Knoten ist, fügen wir diese beiden Zyklen wie in Abbildung 14 dargestellt zusammen. Der neue Zyklus enthält höchstens eine Kante mehr vom Gewicht zwei. Wir erhalten zusätzliches Gewicht von eins pro Baum. Dies ergibt insgesamt einen Beitrag von t.

Falls die Komponente ein isolierter Z-Knoten ist, müssen wir eine (beliebige) Kante des Zyklus aufbrechen. Dies summiert sich zu zusätzlichen Kosten von s.

Es verbleibt der Fall einer Zusammenhangskomponente, die ein Baum mit mehr als zwei Z-Knoten ist. In jedem Z-Knoten c des Baumes, der kein Blatt ist, bilden wir Paare von guten Knoten wie folgt: Wenn $v_1, \ldots, v_m$ ein zusammenhängender Block von gematchten Knoten in c ist, so daß der linke Nachbar

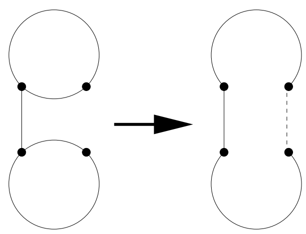

Abbildung 14: Zusammenfügen eines Baums mit zwei Z-Knoten.

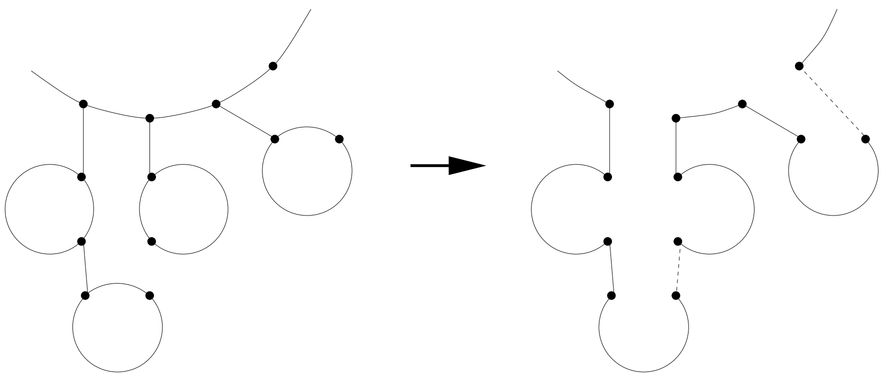

Abbildung 15: Links werden zwei gute Z-Blätter zusammengefügt. Rechts wird ein schlechtes Z-Blatt eingefügt.

von v_1 und der rechte Nachbar von v_m nicht gematcht sind, so paaren wir v_1 mit v_2, v_3 mit v_4, usw. Falls m ungerade ist, so ist v_m schlecht und wird folglich nicht gepaart. Sind alle Knoten in c gematcht und ist die Anzahl der Knoten in c gerade, so starten wir in einem beliebigen Knoten und bilden Paare. Ist die Anzahl der Knoten in c ungerade, so starten wir rechts von dem in Definition 3.11 ausgewählten Knoten und bilden Paare. Der ausgewählte Knoten bleibt damit übrig.

Es sei c ein gutes Z-Blatt mit Vater c', es sei $par(c) = v$ und v sei mit dem Knoten u in c' gepaart (gemäß der gerade beschriebenen Prozedur). Wenn wir nun in c starten, zu c' gehen, dann c' durch die Kante verlassen, die mit u inzident ist, usw., so enden wir in einem anderen guten Z-Blatt. Dies gilt, da $par(d)$ für alle inneren Z-Knoten d des Baumes gut ist, weil S gültig ist.

Für jeden solchen Pfad, der zwei gute Blätter verbindet, fügen wir die Zyklen wie im linken Teil in Abbildung 14 dargestellt zusammen. Pro Pfad, also pro Paar von guten Z-Blättern, entstehen Extrakosten von eins. Jedes schlechte Z-Blatt wird wie im rechten Teil der Abbildung 14 eingefügt. Dies ergibt Extrakosten von eins pro schlechtem Z-Blatt des Baumes. Also erhalten wir insgesamt Extrakosten von $b - \frac{1}{2}g$.

Damit folgt die Behauptung, da C insgesamt Gewicht $n + \zeta$ hat. ∎

3.4.5 Analyse des Algorithmus für 3-Deg-TSP(1,2)

Wir schätzen nun die Approximationsgüte der berechneten Tour ab. Dazu sind die Größen b, g, t und s eines gültigen Subgraphen S durch n und ζ abzuschätzen, um dann mittels Lemma 3.15 das Gewicht der berechneten Tour als Vielfaches des Gewichts einer optimalen Zyklenüberdeckung auszudrücken.

Lemma 3.16 *Für jeden gültigen Subgraphen S gilt $6b - \frac{5}{4}g + 8t + 4s \leq n - \zeta$.*

Beweis. Wir schätzen die Anzahl der Knoten in G wie folgt ab:

Mit jedem schlechten Z-Blatt c assoziieren wir mindestens sechs Knoten, nämlich die Knoten von c selbst – dies sind mindestens vier – den Knoten $par(c)$ sowie den Knoten rechts von $par(c)$, der nicht gematcht ist, da c schlecht ist. Einzige Ausnahme ist der Fall, daß die Knoten der Z-Wurzel r des Baumes, zu dem c gehört, alle gematcht sind und die Anzahl der Knoten in r ungerade ist (siehe Definition 3.11). Der Zyklus r hat dann mindestens fünf Knoten und damit hat der Baum mindestens noch vier gute Z-Blätter. Wir assoziieren einen weiteren (beliebig ausgewählten) Knoten von r mit c. Es gibt $b - g$ schlechte Blätter, mit denen wir insgesamt mindestens $6(b - g)$ viele Knoten assoziiert haben.

Mit jedem guten Z-Blatt c assoziieren wir mindestens fünf Knoten, nämlich die Knoten von c sowie $par(c)$. Dies würde insgesamt $5g$ Knoten liefern. Einzige Ausnahme ist der oben beschriebene Fall, daß die Wurzel des Baumes, zu der c gehört, eine ungerade Anzahl von Knoten besitzt und diese alle gematcht sind. In diesem Fall werden mit m guten Z-Blättern, wobei $m \geq 4$ gilt, nur $5m-1$ Knoten assoziiert, da wir einen Knoten den schlechten Z-Blättern zugeschlagen haben. Daher beträgt der wirkliche Beitrag der guten Z-Blätter nur $\geq \frac{19}{4}g$ Knoten.

Mit jedem Baum mit nur zwei Z-Knoten in S assoziieren wir gerade die Knoten der beiden Z-Knoten. Dies ergibt einen Gesamtbeitrag von mindestens $8t$.

In gleicher Weise liefern die isolierten Z-Knoten einen Beitrag von mindestens $4s$.

Da S gültig ist, wird kein Knoten zweifach gezählt. Schließlich werden nicht die Knoten von z gezählt, die mit einer Kante von Gewicht zwei von z inzident sind, da die Überdeckung C normalisiert worden ist. Es gibt mindestens ζ solcher Knoten in z. ∎

Als nächstes schätzen wir ab, wie oft ein Verbesserungsschritt wie in Lemma 3.14 beschrieben angewendet werden kann. Dazu verwenden wir die folgenden Lemmata.

Lemma 3.17 *Für jeden gültigen Subgraphen S gilt $i \geq 4b - \frac{9}{4}g + 4t + 4s$.*

Beweis. Wir betrachten zuerst die Bäume mit mindestens drei Z-Knoten. Es sei c ein Z-Blatt eines solchen Baumes, d sei dessen Z-Vater und es gelte $\{u, v\} = ed(c, d)$, wobei u zu c gehöre. Ist c gut, so sind die beiden Nachbarn von u verbessernd. Ist c hingegen schlecht, so sind alle Knoten von c außer u verbessernd; dies sind mindestens drei Knoten. Weiterhin ist der rechte Nachbar von v verbessernd. Also gibt es insgesamt mindestens vier verbessernde Knoten pro schlechtem Z-Blatt. Einzige Ausnahme ist der Fall, daß alle Knoten von d gematcht sind und die Anzahl der Knoten von d ungerade ist. (Da S gültig ist, muß d die Z-Wurzel des Baumes sein.) Hier fällt der rechte Nachbar von v weg. In diesem Fall hat der Baum aber m gute Z-Blätter mit $m \geq 4$. Diese haben $2m$ verbessernde Knoten. Von diesen weisen wir einen c zu. (Dieses geschieht nur auf einer gedanklichen Ebene, um die Knoten zu zählen.) Mit den m guten Z-Blättern assoziieren wir danach nur noch $2m-1$ viele verbessernde Knoten. Insgesamt erhalten wir so $\geq 4(b-g)+\frac{7}{4}g = 4b-\frac{9}{4}g$ verbessernde Knoten.

Für jeden Baum mit zwei Z-Knoten gibt es genau vier verbessernde Knoten. Dies ergibt einen Beitrag von $4t$.

Alle Knoten eines isolierten Z-Knoten sind verbessernd, dies sind mindestens vier Knoten. Dies macht insgesamt $4s$ verbessernde Knoten.

Da kein verbessernder Knoten doppelt gezählt wurde, folgt die Behauptung des Lemmas via Aufsummieren. ∎

Lemma 3.18 *Für jeden gültigen Subgraphen S gilt $m \geq 2b+2t$.*

Beweis. Es sei c ein Z-Blatt eines Baums mit mindestens drei c-Knoten und c' sein Z-Vater. Weiterhin sei $\{u,v\} = ed(c,c')$. Mit c assoziieren wir die beiden Knoten gematchten u und v. Da S gültig ist, gilt $par(c) \neq par(c')$ für alle anderen Z-Blätter c', also sind alle auf diese Weise assoziierten Knoten paarweise verschieden.

Mit einem Baum mit zwei Z-Knoten können auf die offensichtliche Weise zwei gematchte Knoten assoziiert werden. ∎

Das nächste Lemma schätzt nun ab, wie oft ein Verbesserungsschritt möglich ist.

Lemma 3.19 *Solange $n+\ell'-\zeta < 7b+6t+6s-\frac{13}{4}g$, gilt $n-m-i-\zeta < b-g+2s-\ell'$.*

Beweis. Nach Lemma 3.17 folgt aus

$$n+\ell'-\zeta < 7b+6t+6s-\tfrac{13}{4}g,$$

daß

$$n+\ell'-\zeta < 3b+2t+2s-g+i.$$

Daraus folgt

$$n+\ell'-\zeta < b+2s-g+i+m$$

mittels Lemma 3.18. ∎

Zusammen mit Lemma 3.14 liefert das obige Lemma das folgende Korollar.

Korollar 3.20 *Wenn ein gültiger Subgraph S keinen weiteren Verbesserungsschritt wie in Abbildung 12 erlaubt, dann gilt* $7b + 6t + 5s - \frac{13}{4}g \leq n + \ell' - \zeta$.

Nun sind wir in der Lage, die Approximationsgüte des Algorithmus für das Problem 3-Deg-TSP(1,2) abzuschätzen.

Theorem 3.21 *Es gibt einen Approximationsalgorithmus mit polynomieller Laufzeit für 3-Deg-TSP(1,2), der Approximationsgüte* 1.15 *erreicht.*

Beweis. Es sei S der gültige Subgraph, der von dem Algorithmus durch die Verbesserungsprozedur in Abbildung 12 berechnet wird. Da S gültig ist, gilt nach Lemma 3.16

$$6b - \tfrac{5}{4}g + 8t + 4s \leq n - \zeta. \tag{8}$$

Da S nach Konstruktion der Verbesserungsprozedur keinen weiteren Verbesserungsschritt erlaubt, gilt nach Korollar 3.20

$$7b - \tfrac{13}{4}g + 6t + 6s \leq n + \ell' - \zeta. \tag{9}$$

Addiert man zweimal (9) zu (8), so erhält man

$$20b - \tfrac{31}{4}g + 20t + 16s \leq 3n + 2\ell' - 3\zeta. \tag{10}$$

Lemma 3.1 und die Tatsache, daß die Verbesserungsprozedur niemals die Anzahl der isolierten Z-Knoten erhöht, liefert $s \leq \ell'$. Damit impliziert (10)

$$b - \tfrac{1}{2}g + t + s \leq \tfrac{1}{20}(3n + 6\ell' - 3\zeta).$$

Zusammen mit Lemma 3.15 zeigt letztere Ungleichung, daß der Algorithmus eine Hamiltonsche Tour mit Gewicht höchstens

$$\tfrac{23}{20}n + \tfrac{6}{20}\ell' + \tfrac{17}{20}\zeta \leq \tfrac{23}{20}(n + \ell)$$

berechnet, da sowohl $\ell' \leq \ell$ als auch $\zeta \leq \ell$ nach (7) gelten. ∎

Die Laufzeit des Algorithmus wird bestimmt durch die Laufzeit des Algorithmus von Hartvigsen, um eine optimale 4-Zyklenüberdeckung zu berechnen, beträgt also $O(n^3)$. Die Verbesserungsprozedur kann in quadratischer Zeit berechnet werden, da pro Verbesserungsschritt nur $O(n)$ Kanten untersucht werden müssen, weil H höchstens $3n$ Kanten hat.

Bemerkung 3.22 *Startet man hier nur mit einer* 3*-Zyklenüberdeckung, so erhält man einen* $\frac{7}{6}$*-Approximationsalgorithmus für 3-Deg-TSP(1,2) mit Laufzeit* $O(n^{5/2})$*. Dies ist besser als die* $\frac{11}{9}$*, die wir für TSP(1,2) in Abschnitt 3.3 erhalten, wenn wir mit einer* 3*-Zyklenüberdeckung starten.*

4 Zyklenüberdeckungen maximalen Gewichts in gerichteten Graphen

In diesem Abschnitt bezeichnet $G = (V, K(V))$ einen vollständigen schleifenfreien gerichteten Graphen. Weiterhin ist $w : K(V) \to \mathbb{Q}_{\geq 0}$ eine Gewichtsfunktion, die jeder Kante ein nicht-negatives Gewicht zuweist.

4.1 Max-ATSP

Hauptergebnis dieses Abschnitts ist ein $\frac{8}{13}(1 - \frac{1}{n})$-Approximationsalgorithmus mit polynomieller Laufzeit für Max-ATSP. Der Algorithmus besteht aus vier separaten Algorithmen. Die schwerste der vier berechneten Hamiltonschen Touren hat mindestens $\frac{8}{13}$ des Gewichts einer Tour maximalen Gewichts. Im folgenden gehen wir davon aus, daß der Graph G eine gerade Anzahl n von Knoten hat. Andernfalls fügen wir einen zusätzlichen Knoten v_0 hinzu und setzen $w(v, v_0) = w(v_0, v) = 0$ für alle anderen Knoten v. Jede Hamiltonsche Tour in dem ursprünglichen Graphen mit Gewicht W induziert eine Tour mit Gewicht $\geq (1 - \frac{1}{n}) \cdot W$ in dem modifizierten Graphen. Folglich kann aus einer $\frac{8}{13}$-Approximation einer Hamiltonschen Tour in dem modifizierten Graphen eine $\frac{8}{13}(1 - \frac{1}{n})$-Approximation in dem ursprünglichen Graphen gewonnen werden.

In jedem der vier Algorithmen wird jeweils zuerst eine 2-Zyklenüberdeckung maximalen Gewichts von G berechnet. Wir verwenden im weiteren die folgenden Notationen: $C_1, \dots, C_\ell$ bezeichnen die Zyklen der berechneten 2-Zyklenüberdeckung $\mathcal{C}$. Es sei $I_2 \subseteq \{1, \dots, \ell\}$ die Menge aller Indizes i, so daß C_i ein 2-Zyklus ist. Genauso bezeichne I_3 die Menge aller Indizes, die 3-Zyklen entsprechen. Schließlich sei $I_{\geq 4} := \{1, \dots, \ell\} \setminus (I_2 \cup I_3)$ und $W = w(\mathcal{C})$.

Zu $i \in I_2$ seien b_i und c_i aus $[0, 1]$ so gewählt, daß $b_i \cdot W$ das Gewicht der schwereren Kante in C_i und $c_i \cdot W$ das Gewicht der leichteren Kante ist. (Im Falle der Gleichheit der Gewichte erfolgt die Unterscheidung in schwerere und leichtere Kante beliebig.) Die zwei Knoten von C_i werden mit s_i und t_i bezeichnet, wobei (s_i, t_i) die schwerere Kante von C_i sei. Es sei $b = \sum_{i \in I_2} b_i$ und $c = \sum_{i \in I_2} c_i$.

Zu $j \in I_3$ seien $x_j \cdot W$, $y_j \cdot W$ bzw. $z_j \cdot W$ die Gewichte der schwereren, mittleren bzw. leichteren Kante. In der gleichen Weise wie b und c seien x, y und z definiert.

Schließlich sei für jedes $k \in I_{\geq 4}$ der Wert $p_k \in [0, 1]$ so gewählt, daß $p_k \cdot W$ das Gewicht der leichtesten Kante von C_k ist und es sei $p = \sum_{k \in I_{\geq 4}} p_k$.

4.1.1 Algorithmus 1

Algorithmus 1 (siehe Abbildung 16) ist ein einfacher $\frac{1}{2}$-Approximationsalgorithmus, wie er schon von Fisher, Nemhauser und Wolsey [29] verwendet worden ist. Da die Gewichtsfunktion nur nicht-negative Werte annimmt, können wir im letzten Schritt des Algorithmus kein Gewicht verlieren. Wenn $b = c = \frac{1}{2}$, so ist

Eingabe: gerichteter Graph $G = (V, K(V))$ mit
Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$.
Ausgabe: eine Hamiltonsche Tour von G.

1. Berechne eine 2-Zyklenüberdeckung $\mathcal{C}$ maximalen Gewichts von G.
2. Entferne die leichteste Kante jedes Zyklus in $\mathcal{C}$. Dies liefert eine Menge von knotendisjunkten Pfaden $\mathcal{P}$.
3. Erhalte eine Hamiltonsche Tour, indem die Pfade in $\mathcal{P}$ beliebig zusammengefügt werden.

Abbildung 16: Max-ATSP: Algorithmus 1

dies der Worst Case für Algorithmus 1. Falls jedoch $\mathcal{C}$ Zyklen der Länge drei oder länger enthält, die einen signifikanten Anteil des Gesamtgewichts von $\mathcal{C}$ haben, oder falls b größer als c ist, dann hat Algorithmus 1 eine bessere Approximationsgüte. Genauer gilt, daß das Gewicht in $\mathcal{P}$ mindestens $(1 - c - z - p) \cdot W$ beträgt. Setzt man $p \leq \frac{1}{4}(1 - b - c - x - y - z)$ ein, so erhält man die folgende untere Schranke für die Approximationsgüte von Algorithmus 1.

Theorem 4.1 *Algorithmus 1 berechnet eine* $(\frac{3}{4} + \frac{1}{4}b - \frac{3}{4}c + \frac{1}{4}x + \frac{1}{4}y - \frac{3}{4}z)$-*Approximation einer Hamiltonschen Tour maximalen Gewichts von* G. ■

4.1.2 Algorithmus 2

Bei Algorithmus 2 wird den 2-Zyklen der Zyklenüberdeckung $\mathcal{C}$ besondere Beachtung geschenkt. In ähnlicher Weise tun dies schon Kosaraju, Park und Stein [47]. Der hier vorgestellte Algorithmus hat eine bessere Approximationsgüte als der von Kosaraju, Park und Stein, falls $b > c$ gilt. Dies wird später für Algorithmus 3 wichtig sein.

Wenn im folgenden davon gesprochen wird, ein *Matching maximalen Gewichts* auf einen gerichteten Graphen zu berechnen, so ist damit gemeint, daß die zwei Kanten zwischen zwei Knoten durch eine ungerichtete Kante ersetzt werden, deren Gewicht dem Maximum der Gewichte der zwei entsprechenden Kanten entspricht, dann ein Matching maximalen Gewichts auf dem ungerichteten Graphen berechnet wird und dann die ungerichteten Kanten des Matchings durch die entsprechenden gerichteten Kanten ersetzt werden. Zwei Kanten $e = (u, v)$ und $f = (x, y)$ in einem gerichteten Graphen heißen *kompatibel*, falls $v = x$ und $u \neq y$ oder aber $y = u$ und $v \neq x$. Mit anderen Worten: e und f können zu einem Pfad der Länge zwei zusammengefügt werden.

Eingabe: gerichteter Graph $G = (V, K(V))$ mit Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$.
Ausgabe: eine Hamiltonsche Tour von G.

1. Berechne eine 2-Zyklenüberdeckung $\mathcal{C}$ maximalen Gewichts von G.
2. Definiere eine neue Gewichtsfunktion w' auf $K(V)$ wie folgt: Für jedes $i \in I_2$ setze $w'((s_i, t_i)) = w'((t_i, s_i)) = 2(b_i - c_i) \cdot W$. Für alle $i, j \in I_2$ mit $i \neq j$ setze
$$w'((t_i, s_j)) = w((t_i, s_j)) + (b_i - c_i) \cdot W + (b_j - c_j) \cdot W.$$
Für alle $i \in I_2$ setze $w'((u, s_i)) = w((u, s_i)) + (b_i - c_i) \cdot W$ und $w'((t_i, v)) = w((t_i, v)) + (b_i - c_i) \cdot W$, falls $u \notin \{t_k \mid k \in I_2\}$ und $v \notin \{s_k \mid k \in I_2\}$. Für alle übrigen Kanten e sei $w'(e) = w(e)$.
3. Berechne ein Matching $\mathcal{M}$ maximalen Gewichts bzgl. der neuen Gewichtsfunktion w'.
4. Entferne die leichteste Kante eines jeden Zyklus der Länge mindestens drei in $\mathcal{C}$. Dies ergibt eine Kollektion $\mathcal{P}$ knotendisjunkter Pfade der Länge mindestens zwei.
5. Es sei $\mathcal{T}$ die Menge aller 2-Zyklen in $\mathcal{C}$, die keine Kante mit $\mathcal{M}$ gemeinsam haben. Es sei $\hat{\mathcal{M}}$ die Menge aller Kanten von $\mathcal{M}$, die nicht in einem 2-Zyklus von $\mathcal{C}$ vorkommen.
6. Es sei H der Multigraph, der aus $(V, \mathcal{T} \cup \mathcal{P} \cup \hat{\mathcal{M}})$ durch Kontrahieren der Zyklen von $\mathcal{T}$ entsteht. Färbe die Kanten von H gemäß Lemma 4.2 und transferiere die Färbung zurück nach $(V, \mathcal{P} \cup \hat{\mathcal{M}})$.
7. Für jeden 2-Zyklus, der nicht in $\mathcal{T}$ ist, füge die schwerere Kante zu beiden Farbklassen hinzu. Für jeden 2-Zyklus C_i in $\mathcal{T}$ seien e_i und f_i die beiden Kanten in $\hat{\mathcal{M}}$, die mit s_i und t_i inzident sind. Falls e_i und f_i die gleiche Farbe haben, dann füge die kompatible Kante von C_i zu beiden Farbklassen hinzu. Sonst füge die zu e_i kompatible Kante von C_i zu der Farbklasse, zu der auch e_i gehört, und die zu f_i kompatible Kante zu der andere Farbklasse.
8. Füge die Pfade der Farbklasse mit dem größeren Gewicht (unter w) zu einer Hamiltonschen Tour zusammen.

Abbildung 17: Max-ATSP: Algorithmus 2

Algorithmus 2 kombiniert eine 2-Zyklenüberdeckung $\mathcal{C}$ mit einem Matching $\mathcal{M}$. Die Kanten (einer Teilmenge) von $\mathcal{C}$ und $\mathcal{M}$ (als Kanten eines Multigraphen gesehen) werden dann mit zwei Farben so gefärbt, daß jede der zwei Farbklassen jeweils eine Kollektion von knotendisjunkten Pfaden bildet, also zu einer Hamiltonschen Tour erweitert werden kann. Damit dies überhaupt möglich ist, darf keine Kante von $\mathcal{M}$ mit einer Kante eines 2-Zyklus von $\mathcal{C}$ übereinstimmen, da sonst auf jeden Fall drei Farben nötig wären. Dies kann zum Beispiel dadurch erreicht werden, daß man ein Matching auf einem modifizierten Graphen berechnet, in dem jede Kante eines 2-Zyklus von $\mathcal{C}$ Gewicht null hat. Die schwerere der beiden Touren ist die Ausgabe von Algorithmus 2.

Im einzelnen arbeitet Algorithmus 2 wie folgt: Zuerst wird eine 2-Zyklenüberdeckung $\mathcal{C}$ von G berechnet. Dann wird eine neue Gewichtsfunktion w' auf G definiert: Jede Kante eines 2-Zyklus C_i in $\mathcal{C}$ erhält das neue Gewicht $2(b_i - c_i) \cdot W$. Zusätzlich wird zu dem Gewicht jeder Kante, die kompatibel mit der schwereren Kante (s_i, t_i) von C_i ist, das Gewicht $(b_i - c_i) \cdot W$ addiert. Anders als oben beschrieben erhalten die Kanten in den 2-Zyklen nicht Gewicht null unter w', sondern ein kleines Korrekturgewicht, das dazu dient, eine verbesserte Approximation im Falle von vielen unbalancierten 2-Zyklen zu liefern. Den gleichen Zweck erfüllen die Modifikationen der Gewichte der mit (s_i, t_i) kompatiblen Kanten. Diese Korrekturgewichte sind gerade so gewählt, das beim späteren Zurückgehen von w' zu w keine Verluste auftreten. Die Details werden im Verlauf des Beweis von Theorem 4.4 klar werden.

Als nächstes berechnet Algorithmus 2 ein Matching maximalen Gewichts $\mathcal{M}$ von G unter der neuen Gewichtsfunktion w'. Da jede Hamiltonsche Tour nur durch eine Kante eines 2-Zyklus laufen kann, läßt sich zeigen, daß $w'(\mathcal{M})$ mindestens $\frac{1}{2}(L + (2b - 3c) \cdot W)$ beträgt, wobei L das Gewicht einer optimalen Hamiltonschen Tour in G ist (unter w). Von jedem Zyklus C der Länge mindestens drei wird nun die leichteste Kante entfernt. Es ergibt sich eine Kollektion von knotendisjunkten Pfaden $\mathcal{P}$. Lemma 4.2 zeigt nun, daß die Kanten in $\mathcal{P}$ und $\mathcal{M}$ 2-pfadfärbbar sind. Dabei heißt ein Multigraph 2-*pfadfärbbar*, falls sich seine Kanten mit zwei Farben so färben lassen, daß jede Farbklasse jeweils eine Menge von knotendisjunkten Pfaden bildet. Darüber hinaus läßt sich zeigen, daß sich zu jeder der beiden Klassen jeweils eine Kante jedes 2-Zyklus von $\mathcal{C}$ hinzufügen läßt und die resultierende Kantenmenge immer noch 2-pfadfärbbar ist. Nun fügt man die Pfade der schwereren Farbklasse zu einer Hamiltonschen Tour zusammen und erhält die Ausgabe von Algorithmus 2.

Färbungslemma. Hauptziel dieses Paragraphen ist es zu zeigen, daß die weiter oben beschriebene Kantenfärbung immer existiert und in Polynomialzeit berechnet werden kann. Ist $G = (V, E)$ ein gerichteter Graph und $\mathcal{Z}$ eine Kollektion knotendisjunkter Zyklen in G (nicht zwangsläufig eine Überdeckung), so produziert der Prozeß des *Kontrahierens* der Zyklen in $\mathcal{Z}$ einen Multigraphen $H = (V', E')$

wie folgt: Die Knotenmenge V' besteht aus allen Knoten in V, die zu keinem Zyklus in $\mathcal{Z}$ gehören sowie einem Extraknoten z_i für jeden Zyklus Z_i in $\mathcal{Z}$. Ist $e = (u, v) \in E$ und liegen u und v in ein und demselben Zyklus in $\mathcal{Z}$, so wird e entfernt. Sonst wird die Kante (u', v') zu E' hinzugefügt, wobei $u' = u$ gilt, falls u nicht zu einem Zyklus in $\mathcal{Z}$ gehört, und $u' = z_i$ falls u in Z_i liegt. In gleicher Weise wird der Knoten v' gewählt. Dabei kann es sein, daß E' eine Multimenge und damit H ein Multigraph ist, d.h. es kann mehr als eine Kante in derselben Richtung zwischen zwei Knoten geben.

Lemma 4.2 *Es sei $\mathcal{T}$ eine Kollektion von knotendisjunkten 2-Zyklen auf einer Knotenmenge V, $\mathcal{P}$ eine Kollektion knotendisjunkter Pfade der Länge mindestens zwei sowie $\mathcal{M}$ ein perfektes Matching auf V. Zusätzlich gelte, daß $\mathcal{T}$ und $\mathcal{P}$ knotendisjunkt sind und $\mathcal{T} \cap \mathcal{M} = \emptyset$ ist. Es sei H der Multigraph, der aus $G = (V, \mathcal{T} \cup \mathcal{P} \cup \mathcal{M})$ durch Kontrahieren der Zyklen in $\mathcal{T}$ hervorgeht. Dann ist H 2-pfadfärbbar. Darüberhinaus kann eine entsprechende Färbung in Polynomialzeit berechnet werden.*

Bevor wir Lemma 4.2 beweisen, führen wir noch eine weitere Graphtransformation ein: Ist $G = (V, E)$ ein gerichteter Graph und $\mathcal{Z}$ eine Kollektion knotendisjunkter Zyklen in G (nicht zwangsläufig eine Überdeckung), so produziert der Prozeß des *Entfernens* der Zyklen in $\mathcal{Z}$ einen Multigraphen $H = (V', E')$ wie folgt: Die Knotenmenge V' besteht aus allen Knoten in V die zu keinem Zyklus in $\mathcal{Z}$ gehören. Ist $e = (u, v) \in E$ und liegen weder u noch v in Zyklen von $\mathcal{Z}$, so wird e zu E' hinzugefügt. Alle anderen Kanten aus E werden entfernt. Weiterhin wird eine Kante (u, v) zu E' hinzugefügt, falls $u, v \in V'$ sind, es keine direkte Kante (u, v) in G gibt und ein Pfad von u nach v in G existiert, der als interne Knoten nur Knoten aus Zyklen in $\mathcal{Z}$ besitzt. (Dieser Pfad kann durchaus durch mehrere Zyklen aus $\mathcal{Z}$ gehen.) Obiger Prozeß kann in gleicher Weise auch ganz allgemein für eine Menge von Subgraphen von G definiert werden.

Zum Beweis von Lemma 4.2 wird das nachfolgende „path coloring lemma“ benötigt, das von Kosaraju, Park und Stein [47, Lem. 2] formuliert, aber nie bewiesen wurde. Ein Beweis dieses Lemmas wird in Anhang A geführt.

Lemma 4.3 *Es sei G ein gerichteter schleifenfreier Graph, so daß*

1. *alle Knoten Ein- und Ausgrad höchstens zwei und*
2. *totalen Grad höchstens drei haben und weiterhin*
3. *G keine 2-Zyklen enthält.*

Dann ist G 2-pfadfärbbar. Solch eine Färbung kann in Polynomialzeit berechnet werden. ∎

Beweis von Lemma 4.2. Wir teilen die Kanten von $\mathcal{M}$ in drei Klassen ein:

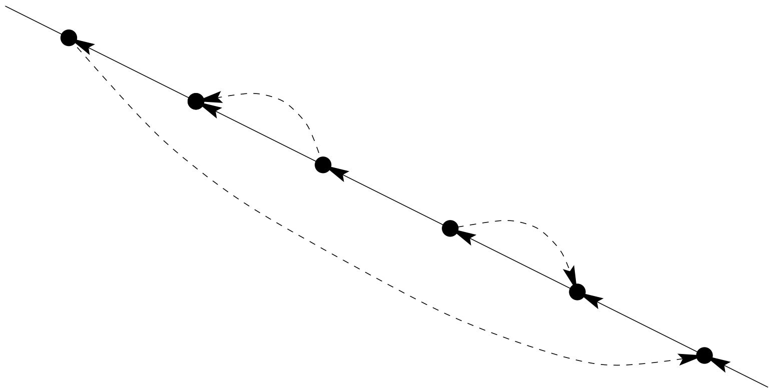

Abbildung 18: Drei Pfadkanten (gestrichelt). (Durchgezogene Kanten sind Kanten aus $\mathcal{P}$, gestrichelte stehen für Kanten aus $\mathcal{M}$.)

1. Eine Kante $(u, v) \in \mathcal{M}$ ist eine *Zyklenkante*, falls beide Knoten (in G) zu einem 2-Zyklus von $\mathcal{T}$ gehören.

2. *Pfadkanten* (u, v) werden induktiv wie folgt definiert: Wenn (u, v) oder (v, u) in $\mathcal{P}$ ist, dann ist (u, v) eine Pfadkante. Falls u und v auf ein und demselben Pfad in $\mathcal{P}$ liegen und alle Knoten zwischen u und v auf diesem Pfad ausschließlich inzident mit Pfadkanten sind, dann ist auch (u, v) eine Pfadkante. (Abbildung 18 illustriert diesen Fall.)

3. Alle anderen Kanten heißen *übrige Kanten*.

Die Knoten in V heißen dann Zyklenknoten, Pfadknoten bzw. übrige Knoten, je nachdem mit welchem Typ von Kante aus $\mathcal{M}$ sie inzident sind. (Dies ist wohldefiniert, weil $\mathcal{M}$ ein Matching ist.) Die Knoten aus $\mathcal{T}$ sind entweder Zyklen- oder übrige Knoten, die Knoten aus $\mathcal{P}$ entweder Pfad- oder übrige Knoten.

Ein Subpfad s eines Pfades aus $\mathcal{P}$ heißt *Pfadknoten-Subpfad*, falls alle Knoten in s Pfadknoten sind. Solch ein Subpfad s heißt *maximal*, wenn für jeden Pfadknoten-Subpfad s', der s als Subpfad enthält, $s = s'$ gilt. Es sei $\mathcal{S}$ die Menge aller maximalen Pfadknoten-Subpfade in $\mathcal{P}$.

Wir kontrahieren nun die Zyklen in $\mathcal{T}$ und erhalten einen Multigraphen H. In H entfernen wir nun alle Pfade in $\mathcal{S}$ und erhalten einen Multigraphen H'. Da später eine Färbung von H' zurück zu H transferiert wird, beschreiben wir nun detailliert, was dabei mit den einzelnen Kanten in H geschieht:

1. Falls ein maximaler Pfadknoten-Subpfad s in $\mathcal{S}$ einen Start- oder Endknoten eines Pfades aus $\mathcal{P}$ enthält, dann wird s komplett entfernt.

2. Andernfalls wird s zu einer einzelnen Kante in H'.

3. Alle Pfadkanten werden entfernt.

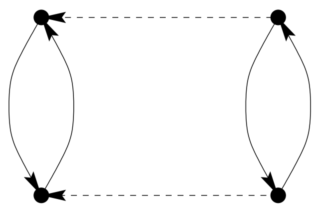

Abbildung 19: Aus obiger Konfiguration entstehen Doppelkanten in H'. (Durchgezogene Kanten entsprechen Kanten aus $\mathcal{T}$, gestrichelte stehen für Zyklenkanten aus $\mathcal{M}$.) Ein 2-Zyklus würde entstehen, wenn die Richtung von einer der beiden Zyklenkanten umgedreht werden würde.

4. Alle Zyklen- und übrigen Kanten bleiben in H'.

Mit Hilfe des Lemmas 4.3 werden nun die Kanten des Multigraphen H' gefärbt. H' kann noch 2-Zyklen oder Doppelkanten enthalten und daher nicht die Voraussetzungen von Lemma 4.3 erfüllen. Aber die 2-Zyklen oder Doppelkanten sind von einer speziellen Form: Keine übrige Kante aus $\mathcal{M}$ kann zu solch einem 2-Zyklus oder solch einer Doppelkante gehören. Also sind die einzigen Kanten, die 2-Zyklen oder Doppelkanten bilden können, Zyklenkanten aus $\mathcal{M}$, die inzident mit den Knoten der gleichen zwei 2-Zyklen aus $\mathcal{T}$ sind (siehe auch Abbildung 19). Solche 2-Zyklen und Doppelkanten sind damit aber isoliert und sind offensichtlich 2-pfadfärbbar. Darüberhinaus ist eine solche Färbung unabhängig von der Färbung des übrigen Multigraphen. Folglich ist H' 2-pfadfärbbar gemäß Lemma 4.3. Weil die Färbung aus Lemma 4.3 in Polynomialzeit berechnet werden kann, kann auch die Färbung von H' in Polynomialzeit berechnet werden.

Als nächstes wird nun die Färbung von H' zurück nach H transferiert. Jede Kante in H, die auch in H' auftritt, erhält in H die gleiche Farbe wie in H'. Alle anderen Kanten werden wie folgt gefärbt:

1. Falls ein maximaler Pfadknoten-Subpfad s in $\mathcal{S}$ mit einem ganzen Pfad aus $\mathcal{P}$ übereinstimmt und daher entfernt wurde, dann werden seine Kanten mit einer beliebigen Farbe gefärbt und die inzidenten Kanten aus $\mathcal{M}$ mit der anderen Farbe gefärbt. Falls s entfernt wird, weil s nur den Start- aber nicht den Endknoten eines Pfades p aus $\mathcal{P}$ enthält, dann wird s mit derselben Farbe wie die Startkante von p (in H) gefärbt, und die inzidenten Kanten aus $\mathcal{M}$ werden mit der anderen Farbe gefärbt. Der Fall, in dem s den End- aber nicht den Startknoten enthält, wird in gleicher Manier behandelt.

2. Falls s durch eine Kante in H' ersetzt wurde, so wird s mit der gleichen Farbe wie diese Kante gefärbt.

3. Pfadkanten aus $\mathcal{M}$ erhalten die Farbe, die der entsprechende maximale Pfadknoten-Subpfad nicht hat.

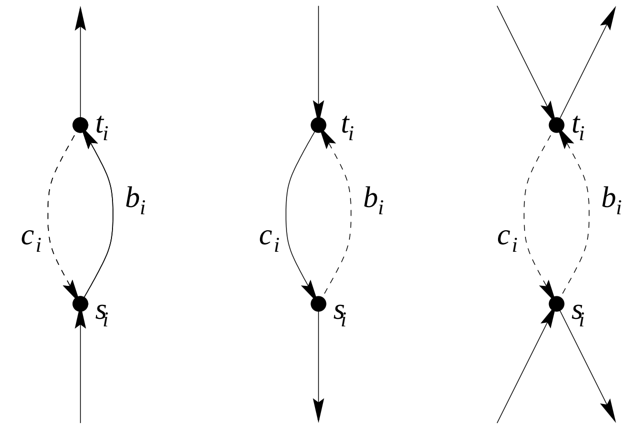

Abbildung 20: Eine Hamiltonsche Tour (durchgezogen gezeichnet) kann einen 2-Zyklus auf drei Arten durchlaufen.

Aufgrund der Konstruktion werden dadurch keine neuen Zyklen in den Farbklassen erzeugt. Offensichtlich kann die Färbung in Polynomialzeit berechnet werden. Damit erfüllt die konstruierte Färbung die Behauptung des Lemmas. ■

Approximationsgüte. Das folgende Theorem schätzt die Approximationsgüte der von Algorithmus 2 berechneten Tour ab.

Theorem 4.4 *Algorithmus 2 berechnet eine* $(\frac{5}{8} + \frac{1}{8}b - \frac{1}{8}c + \frac{1}{8}x + \frac{1}{8}y - \frac{3}{8}z)$*-Approximation einer Hamiltonschen Tour maximalen Gewichts von* G*.*

Beweis. Es sei L das Gewicht einer Hamiltonschen Tour maximalen Gewichts in G. Was ist das Gewicht L' einer Hamiltonschen Tour maximalen Gewichts unter der modifizierten Gewichtsfunktion w'? Abbildung 20 zeigt, wie eine Hamiltonsche Tour in G durch einen 2-Zyklus C_i aus $\mathcal{C}$ laufen kann. Im ersten Fall verlieren wir Gewicht $b_i \cdot W$, bekommen aber ein zusätzliches Gewicht von $4(b_i - c_i) \cdot W$ unter w'. Im zweiten Fall verlieren wir $c_i \cdot W$, aber erhalten zusätzlich $2(b_i - c_i) \cdot W$. Im dritten Fall verlieren wir nichts und bekommen $2(b_i - c_i) \cdot W$ hinzu. Da $b_i \geq c_i$, folgt $L' \geq L + (2b - 3c) \cdot W$. Folglich hat ein Matching $\mathcal{M}$ maximalen Gewichts in Bezug auf w' mindestens Gewicht $\frac{1}{2}(L + (2b - 3c) \cdot W)$. Auf der anderen Seite hat $\mathcal{P}$ mindestens Gewicht $(1 - b - c - z - p) \cdot W$.

Wie ändert sich dies, nachdem im Schritt 7 des Algorithmus die 2-Zyklen zu den Farbklassen hinzugefügt worden sind und dann die Klassen unter w statt w' betrachtet werden? Für jeden 2-Zyklus C_i, der nicht in $\mathcal{T}$ ist, wird die schwerere Kante zu beiden Klassen hinzugefügt. Demnach erhalten wir Gewicht $2b_i \cdot W$, verlieren allerdings $2(b_i - c_i) \cdot W$, da die Matchingkante wegfällt. Der Gesamtgewinn an Gewicht beträgt $2c_i \cdot W$. (Da $\mathcal{M}$ ein Matching ist, kann keine Kante von $\mathcal{M}$

den Zyklus C_i betreten oder verlassen, also müssen die modifizierten Gewichte dieser Kanten nicht berücksichtigt werden.)

Als nächstes betrachten wir die Kanten in $\mathcal{T}$ und die Kanten in $\hat{\mathcal{M}}$, die inzident mit Knoten aus $\mathcal{T}$ sind. Als erstes überzeugen wir uns davon, daß das Einfügen der Kanten von $\mathcal{T}$ keine Zyklen erzeugen kann: Es seien dazu e_i und f_i die zwei Kanten von $\mathcal{M}$, die mit den beiden Knoten s_i bzw. t_i eines 2-Zyklus C_i inzident sind. Falls e_i und f_i unterschiedliche Farben in H haben, so kann offensichtlich kein Zyklus erzeugt werden. Haben hingegen e_i und f_i dieselbe Farbe, so kann das Einfügen auch keinen Zyklus erzeugen, da sonst auch in H ein Zyklus vorhanden sein müßte. Als zweites betrachten wir die Gewichtsveränderung beim Übergang von w' zu w. Dies geschieht nach folgendem Prinzip: Damit eine Kante (u, v) ein anderes Gewicht unter w' als unter w hat, muß entweder $u = t_i$ für ein $i \in I_2$ oder $v = s_j$ für ein $j \in I_2$ oder beides gelten. Im ersten Fall berücksichtigen wir die Veränderung, wenn wir C_i betrachten, im zweiten Fall bei der Betrachtung von C_j. Im dritten Fall teilen wir die Gewichtsveränderung zwischen C_i und C_j auf. Der Verlust von $(b_i - c_i) \cdot W$ wird C_i und der Verlust von $(b_j - c_j) \cdot W$ wird C_j angelastet. Dies erlaubt uns alle drei Fälle einheitlich zu untersuchen. Für jeden Zyklus C_i in $\mathcal{T}$ sei e_i die Kante aus $\mathcal{M}$, die inzident mit s_i ist, und f_i die Kante, die inzident mit t_i ist. Falls $e_i = (u, s_i)$ ist, so verlieren wir Gewicht $(b_i - c_i) \cdot W$, aber gewinnen auf der anderen Seite $b_i \cdot W$, da wir die schwerere Kante von C_i zu der Farbklasse von e_i hinzufügen können. Falls $e_i = (s_i, u)$ ist, verlieren wir kein Gewicht, bekommen aber nur $c_i \cdot W$ hinzu, da die leichtere Kante in die Klasse von e_i eingefügt wird. Die gleiche Überlegung zeigt, daß wir Gewicht $(b_i - c_i) \cdot W$ verlieren und $b_i \cdot W$ gewinnen, falls $f_i = (t_i, v)$, bzw. nichts verlieren und $c_i \cdot W$ hinzuerhalten, falls $f_i = (v, t_i)$. Der Gesamtgewinn an Gewicht beträgt $2c_i \cdot W$.

Alles zusammen genommen beträgt das Gewicht in beiden Farbklassen mindestens $\frac{1}{2}(L + (2b - 3c) \cdot W) + (1 - b - c - z - p) \cdot W + 2c \cdot W$. Folglich hat die schwerere der beiden Klassen mindestens Gewicht $\frac{1}{4}L + (\frac{1}{2} - \frac{1}{4}c - \frac{1}{2}z - \frac{1}{2}p) \cdot W$. Die Behauptung des Theorems folgt durch Einsetzen von $p \leq \frac{1}{4}(1 - b - c - x - y - z)$ sowie $W \geq L$. ■

4.1.3 Algorithmus 3

Jedem Zyklus C_λ in $\mathcal{C}$ entspricht ein zweiter Zyklus, den man durch Umdrehen aller Kanten von C_λ erhält. Ist C_λ ein 2-Zyklus, so liefert diese Prozedur den Ausgangszyklus. Bei längeren Zyklen hingegen erhält man einen anderen Zyklus. Es sei im folgenden r_λ das Gewicht der schwersten Kante in dem zu C_λ gehörenden „umgedrehten“ Zyklus für alle $\lambda \in \{1, \ldots, \ell\} \setminus I_2$.

Im Falle eines 3-Zyklus C_j gilt $r_j \leq x_j + y_j$, da man sonst C_j durch einen 2-Zyklus mit Gewicht mindestens $(r_j + z_j) \cdot W > w(C_j)$ und einen isolierten Knoten ersetzen kann und so eine Zyklenüberdeckung mit größerem Gewicht erhält, was ein Widerspruch ist.

Eingabe: gerichteter Graph $G = (V, K(V))$ mit
Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$.
Ausgabe: eine Hamiltonsche Tour von G.

1. Berechne eine 2-Zyklenüberdeckung $\mathcal{C}$ maximalen Gewichts von G.
2. Ersetze jeden 3-Zyklus in $\mathcal{C}$ durch einen 2-Zyklus mit Gewicht $r_j + z_j$ wie zu Beginn von Abschnitt 4.1.3 beschrieben. Erweitere diese Kollektion von Zyklen zu einer neuen Zyklenüberdeckung $\tilde{\mathcal{C}}$, indem die übrigen Knoten zu einem großen Zyklus zusammengesetzt werden.
3. Wende Algorithmus 2 auf die Zyklenüberdeckung $\tilde{\mathcal{C}}$ an.

Abbildung 21: Max-ATSP: Algorithmus 3

Falls r_j groß ist, wird ein 3-Zyklus durch einen sehr unbalancierten 2-Zyklus ersetzt, was manchmal vorteilhaft ist. Dies ist die wesentliche Idee von Algorithmus 3. Im folgenden sei $r = \sum_{j \in I_3} r_j$. Es zeigt sich, daß Algorithmus 3 Algorithmus 2 unterlegen ist, falls $r < z$. Daher nehmen wir im weiteren an, daß $r \geq z$. Darüberhinaus nehmen wir an, daß die andere (leichtere) Kante in dem neu kreierten 2-Zyklus genau Gewicht $z_j \cdot W$ hat, da eine Verringerung des Gewichtes nur das Gesamtgewicht reduziert.

Theorem 4.5 *Algorithmus 3 berechnet eine* $(\frac{5}{8} + \frac{1}{8}b - \frac{1}{8}c - \frac{3}{8}x - \frac{3}{8}y - \frac{1}{8}z + \frac{1}{2}r)$*-Approximation einer Hamiltonschen Tour maximalen Gewichts von* G.

Beweis. Es sei wiederum L das Gewicht einer Hamiltonschen Tour maximalen Gewichts von G. Die neue Zyklenüberdeckung $\tilde{C}$ hat mindestens Gewicht $\tilde{W} = (1 - x - y + r) \cdot W$. Das Gesamtgewicht aller schwereren Kanten in den 2-Zyklen von $\tilde{\mathcal{C}}$ ist $\tilde{b} \cdot W$, wobei $\tilde{b} = b + r$ ist. Ebenso ist das Gesamtgewicht aller leichteren Kanten $\tilde{c} \cdot W$, wobei $\tilde{c} = c + z$. Da wir angenommen haben, daß $r \geq z$, gilt $\tilde{b} \geq \tilde{c}$. Die Summe aller Gewichte der leichtesten Kanten in allen Zyklen der Länge mindestens vier von $\tilde{\mathcal{C}}$ ist $p \cdot W$.

Nun reicht es, die Analyse von Algorithmus 2 zu imitieren. Das Gewicht des Matchings $\tilde{M}$ beträgt $\frac{1}{2}(L + (2\tilde{b} - 3\tilde{c}) \cdot W)$. Da alle 3-Zyklen entfernt wurden, beträgt das Gewicht der Kollektion von Pfaden $\tilde{\mathcal{P}}$

$$(1 - b - c - x - y - z - p) \cdot W = (1 - \tilde{b} - \tilde{c} - x - y + r - p) \cdot W.$$

Durch das Hinzufügen der 2-Zyklen erhalten wir zusätzlich Gewicht $2\tilde{c} \cdot W$. Wir erhalten zwei Farbklassen von knotendisjunkten Pfaden mit Gesamtgewicht von

$\frac{1}{2}L+(1-\frac{1}{2}\tilde{c}-x-y+r-p)\cdot W$. Die Klasse mit dem größeren Gewicht hat mindestens Gewicht $\frac{1}{4}L + (\frac{1}{2} - \frac{1}{4}c - \frac{1}{2}x - \frac{1}{2}y - \frac{1}{4}z + \frac{1}{2}r - \frac{1}{2}p) \cdot W$. Also liefert Algorithmus 3 eine Tour mit Gewicht mindestens $(\frac{5}{8} + \frac{1}{8}b - \frac{1}{8}c - \frac{3}{8}x - \frac{3}{8}y - \frac{1}{8}z + \frac{1}{2}r) \cdot W$. ■

Im Grenzfall $r = x+y$ berechnet Algorithmus 3 eine $(\frac{5}{8}+\frac{1}{8}b-\frac{1}{8}c+\frac{1}{8}x+\frac{1}{8}y-\frac{1}{8}z)$-Approximation, eine Verbesserung von $\frac{1}{4}z$ im Vergleich zu Algorithmus 2.

4.1.4 Algorithmus 4

Während Algorithmus 3 vorteilhaft ist, wenn r groß ist, wird Algorithmus 4 für den Fall besser sein, daß r klein ist. Wiederum beginnt Algorithmus 4 mit dem Berechnen einer 2-Zyklenüberdeckung $\mathcal{C}$ und einem Matching $\mathcal{M}$. Jeder 3-Zyklus in $\mathcal{C}$ beinhaltet entweder drei Knoten, die mit drei verschiedenen Kanten des Matchings inzident sind (*übrige Kanten*), oder zwei Knoten von $\mathcal{C}$ sind mit derselben Kante des Matchings inzident (*Pfadkante*). Abbildung 23 zeigt diese beiden Möglichkeiten.

Im Verlauf von Algorithmus 2 wird aus jedem Zyklus in $\mathcal{C}$ der Länge mindestens drei die leichteste Kante entfernt, um eine Menge von Pfaden $\mathcal{P}$ zu erhalten. Dies stellt sicher, daß keine Schleifen entstehen, nachdem die Pfadknoten-Subpfade im Beweis von Lemma 4.2 entfernt worden sind. Auf der rechten Seite in Abbildung 23 bildet die mit den beiden Pfadknoten inzidente Kante des 3-Zyklus einen maximalen Pfadknoten-Subpfad. Entfernt man diesen Pfad, so erhält man eine Schleife. Beim Transferieren der Färbung würde dies zu einem Zyklus in einer Farbklasse führen. Andererseits können wir den 3-Zyklus auf der linken Seite in Abbildung 23 gemäß Lemma 4.3 färben. In diesem Fall erhalten wir zusätzlich das Gewicht $z_j \cdot W$, wobei j den Index des 3-Zyklus bezeichnet.

Wie kann ein Gewinn erzielt werden, wenn C_j zwei Pfadknoten enthält? Entfernt man die entsprechende Pfadkante, so kann der 3-Zyklus C_j gemäß Lemma 4.3 gefärbt werden. Andererseits kann die Kante zwischen den beiden Pfadknoten zu beiden Farbklassen hinzugefügt werden, egal wie die Kanten von C_j gefärbt worden sind. Abbildung 24 verdeutlicht diesen Sachverhalt. Falls die Pfadkante mit einer der Kanten von C_j übereinstimmt, erhalten wir zusätzlich Gewicht $z_j \cdot W$. Zeigt die Pfadkante in die andere Richtung, so verlieren wir $r_j \cdot W$, erhalten aber $2z_j \cdot W$ zurück. Da $\mathcal{M}$ ein Matching maximalen Gewichts ist, gilt $r_j \geq z_j$. Also erhalten wir insgesamt mindestens Gewicht $(2z - r) \cdot W$ hinzu.

Färbungslemma. Die folgende Variante von Lemma 4.2 wird in Algorithmus 4 verwendet.

Lemma 4.6 *Es sei $\mathcal{T}$ eine Kollektion von knotendisjunkten 2-Zyklen auf einer Knotenmenge V, $\mathcal{U}$ eine Kollektion von knotendisjunkten 3-Zyklen, $\mathcal{P}$ eine Kollektion knotendisjunkter Pfade der Länge mindestens zwei sowie $\mathcal{M}$ ein Matching auf V, so daß jeder Knoten aus $\mathcal{T} \cup \mathcal{P}$ mit einer Kante von $\mathcal{M}$ inzident*

Eingabe: gerichteter Graph $G = (V, K(V))$ mit
Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$.
Ausgabe: eine Hamiltonsche Tour von G.

1. Berechne eine 2-Zyklenüberdeckung $\mathcal{C}$ maximalen Gewichts von G.

2. Definiere eine neue Gewichtsfunktion w' auf $K(V)$ wie folgt: Für jedes $i \in I_2$ setze $w'((s_i, t_i)) = w'((t_i, s_i)) = 2(b_i - c_i) \cdot W$. Für alle $i, j \in I_2$ mit $i \neq j$ setze
$$w'((t_i, s_j)) = w((t_i, s_j)) + (b_i - c_i) \cdot W + (b_j - c_j) \cdot W.$$
Für alle $i \in I_2$ setze $w'((u, s_i)) = w((u, s_i)) + (b_i - c_i) \cdot W$ und $w'((t_i, v)) = w((t_i, v)) + (b_i - c_i) \cdot W$, falls $u \notin \{t_k \mid k \in I_2\}$ und $v \notin \{s_k \mid k \in I_2\}$. Für alle übrigen Kanten e sei $w'(e) = w(e)$.

3. Berechne ein Matching $\mathcal{M}$ maximalen Gewichts bzgl. der neuen Gewichtsfunktion w'.

4. Entferne die leichteste Kante eines jeden Zyklus der Länge mindestens vier in $\mathcal{C}$. Dies ergibt eine Kollektion $\mathcal{P}$ knotendisjunkter Pfade der Länge mindestens drei.

5. Es sei $\mathcal{T}$ die Menge aller 2-Zyklen in $\mathcal{C}$, die keine Kante mit $\mathcal{M}$ gemeinsam haben.

6. Es sei $\mathcal{U}$ die Menge aller 3-Zyklen in $\mathcal{C}$.

7. Es sei $\hat{\mathcal{M}}$ die Menge aller Kanten (u, v) von $\mathcal{M}$, so daß weder (u, v) noch (v, u) in $\mathcal{T} \cup \mathcal{U}$ vorkommen.

8. Es sei H der Multigraph, der aus $(V, \mathcal{T} \cup \mathcal{U} \cup \mathcal{P} \cup \hat{\mathcal{M}})$ durch Kontrahieren der Zyklen von $\mathcal{T}$ entsteht. Färbe die Kanten von H gemäß Lemma 4.6 und transferiere die Färbung zurück nach $(V, \mathcal{U} \cup \mathcal{P} \cup \hat{\mathcal{M}})$.

9. Für jeden 2-Zyklus, der nicht in $\mathcal{T}$ ist, füge die schwerere Kante zu beiden Farbklassen hinzu. Für jeden 2-Zyklus C_i in $\mathcal{T}$ seien e_i und f_i die beiden Kanten in $\hat{\mathcal{M}}$, die mit s_i und t_i inzident sind. Falls e_i und f_i die gleiche Farbe haben, dann füge die kompatible Kante von C_i zu beiden Farbklassen hinzu. Sonst füge die zu e_i kompatible Kante von C_i zu der Farbklasse hinzu, zu der auch e_i gehört, und die zu f_i kompatible Kante zu der anderen Farbklasse.

10. Für jede Kante $(u, v) \in \mathcal{M}$, so daß (u, v) oder (v, u) in $\mathcal{U}$ ist: Falls $(u, v) \in \mathcal{U}$, so füge (u, v) der anderen Farbklasse hinzu, sonst (v, u).

11. Füge die Pfade der Farbklasse mit den größeren Gewicht (unter w) zu einer Hamiltonschen Tour zusammen.

Abbildung 22: Max-ATSP: Algorithmus 4

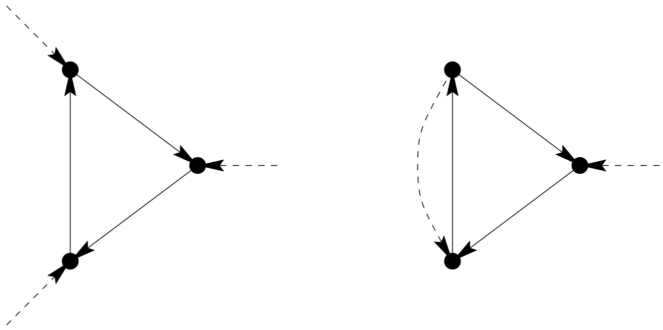

Abbildung 23: Die zwei möglichen Konfigurationen eines 3-Zyklus. (Durchgezogene Kanten sind Kanten des Zyklus, gestrichelte sind Kanten des Matchings. Die Matchingkanten können auch in die entgegengesetzte Richtung zeigen.

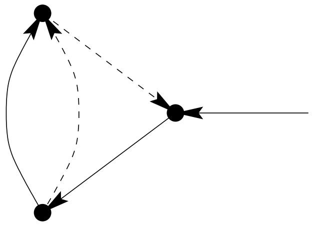

Abbildung 24: Nach Entfernen der Matchingkante kann die Kante zwischen den beiden Pfadknoten zu beiden Farbklassen hinzugefügt werden. (Eine Farbklasse wird durch durchgezogene, die andere durch gestrichelte Kanten dargestellt.)

ist. Zusätzlich gelte, daß $\mathcal{T}$, $\mathcal{U}$ und $\mathcal{P}$ paarweise knotendisjunkt sind und für alle $(u, v) \in \mathcal{T} \cup \mathcal{U}$ weder (u, v) noch (v, u) in $\mathcal{M}$ ist. Es sei H der Multigraph, der aus $G = (V, \mathcal{T} \cup \mathcal{U} \cup \mathcal{P} \cup \mathcal{M})$ durch Kontrahieren der Zyklen in $\mathcal{T}$ hervorgeht. Dann ist H 2-pfadfärbbar. Weiterhin kann eine entsprechende Färbung in Polynomialzeit berechnet werden.

Beweis. Wie in Lemma 4.2 teilen wir die Kanten in $\mathcal{M}$ in drei Klassen ein. (Die Kanten von $\mathcal{M}$, die inzident mit einem Knoten eines 3-Zyklus sind, sind dabei übrige Kanten.)

Die Knoten von V erben ihren Typ von den Kanten von $\mathcal{M}$: Ein Knoten v hat denselben Typ wie die Kante aus $\mathcal{M}$, die inzident mit v ist. Knoten von $\mathcal{T}$ sind entweder Zyklen- oder übrige Knoten, Knoten von $\mathcal{P}$ sind entweder Pfad- oder übrige Knoten. Alle Knoten von $\mathcal{U}$ sind immer übrige Knoten, wobei auch Knoten von $\mathcal{U}$, die nicht inzident mit Kanten von $\mathcal{M}$ sind, als übrige Knoten angesehen werden.

Wie zuvor sei $\mathcal{S} \subseteq \mathcal{P}$ die Menge der maximalen Pfadknoten-Subpfade (wie im Beweis von Lemma 4.2 definiert).

Wir kontrahieren die Zyklen in $\mathcal{T}$ und erhalten einen Multigraphen H. In H entfernen wir die Pfade aus $\mathcal{S}$ und bekommen einen Multigraphen H'. Nach Konstruktion läßt dies die 3-Zyklen von $\mathcal{U}$ unberührt.

Wir verwenden nun Lemma 4.3, um H' zu färben. Die 3-Zyklen in $\mathcal{U}$ erfüllen die Voraussetzung dieses Lemmas. H' kann jedoch noch Doppelkanten oder 2-Zyklen enthalten. Diese können aber wie in Lemma 4.2 gefärbt werden.

Als nächstes wird die Färbung von H' zurück auf H übertragen. Jede Kante in H, die auch in H' vorhanden ist, erhält dieselbe Farbe wie in H'. Dies legt die Farben aller Kanten in $\mathcal{U}$ fest. Die restlichen Kanten werden wie im Beweis von Lemma 4.2 gefärbt.

Nach Konstruktion werden dabei keine Zyklen in den beiden Farbklassen erzeugt. Also erfüllt die Färbung die Behauptungen des Lemmas. Weiterhin kann sie in Polynomialzeit gefunden werden. ∎

Approximationsgüte. Das folgende Theorem schätzt die Approximationsgüte der von Algorithmus 4 berechneten Tour ab.

Theorem 4.7 *Algorithmus 4 berechnet eine* $(\frac{5}{8}+\frac{1}{8}b-\frac{1}{8}c+\frac{1}{8}x+\frac{1}{8}y+\frac{5}{8}z-\frac{1}{2}r)$*-Approximation einer Hamiltonschen Tour maximalen Gewichts von* G*.*

Beweis. Es sei L das Gewicht einer Hamiltonschen Tour maximalen Gewichts von G. Wie im Beweis von Theorem 4.4 kann gezeigt werden, daß das Gewicht L' einer Hamiltonschen Tour maximalen Gewichts von $(V, K(V), w')$ mindestens $L-(2b-3c)\cdot W$ beträgt.

Folglich hat $\mathcal{M}$ mindestens Gewicht $\frac{1}{2}(L+(2b-3c)\cdot W)$. Das Gewicht der Kollektion von Pfaden $\mathcal{P}$ ist mindestens $(1-b-c-x-y-z-p)\cdot W$. Für alle 3-Zyklen aus $\mathcal{U}$ erhalten wir insgesamt Gewicht $(x+y+z)\cdot W$. Aber evtl. muß für jeden dieser Zyklen die Richtung einer Kante aus $\mathcal{M}\setminus\hat{\mathcal{M}}$ geändert werden, dies liefert einen Verlust von maximal $(r-z)\cdot W$. Das Gewicht in den 2-Zyklen kann wie im Beweis von Theorem 4.4 abgeschätzt werden. Insgesamt folgt, daß beide Gewichtsklassen zusammen Gewicht $\frac{1}{2}L+(1-\frac{1}{2}c+z-p-r)\cdot W$ haben.

Folglich beträgt das Gewicht der Farbklasse mit mehr Gewicht mindestens $\frac{1}{4}L+(\frac{1}{2}-\frac{1}{4}c+\frac{1}{2}z-\frac{1}{2}p-\frac{1}{2}r)\cdot W$. Die Behauptung des Theorems folgt durch Einsetzen von $p\le\frac{1}{4}(1-b-c-x-y-z)$ sowie $W\ge L$. ∎

4.1.5 Gesamtalgorithmus

Der Gesamtalgorithmus arbeitet wie folgt: Es werden die Algorithmen 1, 2, 3 und 4 auf dem Eingabegraphen G ausgeführt und die schwerste der vier berechneten Touren ausgegeben.

Die garantierte Approximationsgüte dieses zusammengesetzten Algorithmus kann wie folgt bestimmt werden:

$$\text{Minimiere}\quad \max\{f_1, f_2, f_3, f_4\} \tag{11}$$

wobei

$$
\begin{aligned}
f_1 &= \tfrac{3}{4} + \tfrac{1}{4}b - \tfrac{3}{4}c + \tfrac{1}{4}x + \tfrac{1}{4}y - \tfrac{3}{4}z,\\
f_2 &= \tfrac{5}{8} + \tfrac{1}{8}b - \tfrac{1}{8}c + \tfrac{1}{8}x + \tfrac{1}{8}y - \tfrac{3}{8}z,\\
f_3 &= \tfrac{5}{8} + \tfrac{1}{8}b - \tfrac{1}{8}c - \tfrac{3}{8}x - \tfrac{3}{8}y - \tfrac{1}{8}z + \tfrac{1}{2}r,\\
f_4 &= \tfrac{5}{8} + \tfrac{1}{8}b - \tfrac{1}{8}c + \tfrac{1}{8}x + \tfrac{1}{8}y + \tfrac{5}{8}z - \tfrac{1}{2}r
\end{aligned}
$$

über alle b, c, x, y, z sowie r für die gilt:

$$
\begin{array}{rll}
b, c, x, y, z, r \geq 0, & b \geq c, & r \leq x + y,\\
b + c + x + y + z \leq 1, & x \geq y \geq z. &
\end{array}
$$

Da dieses System nicht allzu groß ist, kann es von Hand gelöst werden und symbolisch mittels eines Computeralgebra-Systems verifiziert werden, siehe auch Anhang B.1. Es stellt sich heraus, daß das obige Minimum $\frac{8}{13}$ beträgt. Es wird angenommen für

$$b = \tfrac{3}{13},\ c = \tfrac{3}{13},\ x = \tfrac{5}{26},\ y = \tfrac{5}{26},\ z = \tfrac{2}{13},\ r = \tfrac{4}{13}$$

sowie

$$b = \tfrac{3}{13},\ c = \tfrac{3}{13},\ x = \tfrac{3}{13},\ y = \tfrac{2}{13},\ z = \tfrac{2}{13},\ r = \tfrac{4}{13}$$

(und allen Punkten auf der Verbindungsstrecke). Dies beweist das folgende Theorem.

Theorem 4.8 *Die schwerste der vier Hamiltonschen Touren, die von den Algorithmen 1, 2, 3 und 4 auf Eingabe G berechnet wird, hat mindestens $\frac{8}{13}$ des Gewichts einer Hamiltonschen Tour maximalen Gewichts von G.* ∎

In der Praxis ist es nicht wirklich notwendig, alle vier Algorithmen auszuführen. Jeder der Algorithmen berechnet zunächst eine Zyklenüberdeckung $\mathcal{C}$. Aus $\mathcal{C}$ ergeben sich die Werte b, c, x, y, z und r. Diese bestimmen, für welchen der vier Algorithmen die Approximationsgüte von $\frac{8}{13}$ erreicht wird.

Die Laufzeit des Algorithmus wird dominiert von der Laufzeit, um eine Zyklenüberdeckung bzw. ein Matching maximalen Gewichts zu berechnen, sowie von der Laufzeit einer Implementierung des „path coloring lemmas" von Kosaraju, Park und Stein (Lemma 4.3). In Anhang A zeigen wir eine Implementierung dieses Lemmas, die kubische Laufzeit hat. Da auch eine Zyklenüberdeckung sowie ein Matching maximalen Gewichts in dieser Zeit berechnet werden kann (siehe z.B. [1]), hat auch der Gesamtalgorithmus eine Laufzeit von $O(n^3)$.

4.2 Max-k-DCC

Die Hamiltonsche Tour, die von dem Algorithmus im vorherigen Abschnitt berechnet wird, kann man auch als eine Approximation einer k-Zyklenüberdeckung maximalen Gewichts von G ansehen. Bei der Abschätzung der Approximationsgüte kann jedoch nicht gefolgert werden, daß ein Matching maximalen Gewichts die Hälfte des Gewichts einer optimalen Zyklenüberdeckung hat, sondern nur einen Bruchteil von $\frac{\tilde{k}-1}{2\tilde{k}}$, wobei $\tilde{k} = k$, falls k ungerade ist, und $\tilde{k} = k + 1$ sonst. Als Beispiel betrachten wir einen Graphen, indem alle Kanten Gewicht null haben, außer einem minimalen spannenden Subgraphen, der nur aus Dreiecken besteht. Die Kanten dieses Subgraphen haben alle Gewicht eins. Dann hat eine optimale 3-Zyklenüberdeckung Gewicht n, wohingegen ein Matching maximalen Gewichts nur Gewicht $\frac{1}{3}n$ hat. Im Falle einer 4-Zyklenüberdeckung besteht der entsprechende Worst Case aus einer Menge von 5-Zyklen und nicht etwa aus 4-Zyklen, da diese von zwei Matchings exakt abgedeckt werden können. Ähnliches gilt für größere Werte von k.

Eine exakte Analyse der Approximationsgüte zeigt, daß der Gesamtalgorithmus (bis auf Terme niedrigerer Ordnung) für $k = 3, 4, 5, 6, 7, \dots$ die Approximationsgüten $\frac{9}{16}, \frac{25}{43}, \frac{25}{43}, \frac{49}{83}, \frac{49}{83}, \dots$ hat. Mit wachsendem k nähert sich die Güte immer weiter $\frac{8}{13}$. Für $k > n/2$ wird schließlich $\frac{8}{13}$ erreicht. Die exakten Approximationsgüten der vier Algorithmen sind

$$
\begin{aligned}
f_1 &= \tfrac{3}{4} + \tfrac{1}{4}b - \tfrac{3}{4}c + \tfrac{1}{4}x + \tfrac{1}{4}y - \tfrac{3}{4}z,\\
f_2 &= \tfrac{3}{8} + \tfrac{\kappa}{2} + (\kappa - \tfrac{3}{8})b - (\tfrac{5}{8} - \tfrac{3}{2}\kappa)c + \tfrac{1}{8}x + \tfrac{1}{8}y - \tfrac{3}{8}z,\\
f_3 &= \tfrac{3}{8} + \tfrac{\kappa}{2} + (\kappa - \tfrac{3}{8})b - (\tfrac{5}{8} - \tfrac{3}{2}\kappa)c - \tfrac{3}{8}x - \tfrac{3}{8}y - (\tfrac{5}{8} - \tfrac{3}{2}\kappa)z + \kappa r,\\
f_4 &= \tfrac{3}{8} + \tfrac{\kappa}{2} + (\kappa - \tfrac{3}{8})b - (\tfrac{5}{8} - \tfrac{3}{2}\kappa)c + \tfrac{1}{8}x + \tfrac{1}{8}y + \tfrac{5}{8}z - \tfrac{1}{2}r,
\end{aligned}
$$

wobei $\kappa = \frac{\tilde{k}-1}{2\tilde{k}}$ gilt und $\tilde{k}$ wie oben definiert ist.

4.3 Max-3-DCC

Für den Spezialfall Max-3-DCC stellen wir in diesem Abschnitt einen verbesserten Approximationsalgorithmus vor, der eine Approximationsgüte von $\frac{3}{5}$ (im Gegensatz zu $\frac{9}{16}$) erreicht. Max-3-DCC ist besonders interessant, da Max-2-DCC in P liegt, Max-3-DCC aber schon APX-vollständig ist [12]. Wir geben im folgenden zwei Algorithmen für Max-3-DCC an. Ähnlich wie in den vorherigen Abschnitten wird die jeweils schwerere der beiden berechneten Zyklenüberdeckungen mindestens $\frac{3}{5}$ des Gewichts einer optimalen Zyklenüberdeckung besitzen. Wir verwenden im folgenden die gleichen Notationen wie zu Beginn von Abschnitt 4.1 eingeführt.

4.3.1 Algorithmus 1

Algorithmus 1 ist ähnlich wie im Falle von Max-ATSP ein einfacher $\frac{1}{2}$-Approximationsalgorithmus. Im Gegensatz zu dem Algorithmus in Abbildung 16 werden nicht alle Zyklen aufgebrochen, sondern nur die 2-Zyklen. Die erhaltenen Kanten werden dann zu einem großen Zyklus zusammengefügt. Gibt es nur einen

Eingabe: gerichteter Graph $G = (V, K(V))$ mit
Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$.
Ausgabe: eine 3-Zyklenüberdeckung $\mathcal{T}$.

1. Berechne eine 2-Zyklenüberdeckung $\mathcal{C}$ maximalen Gewichts von G.
2. Entferne die leichteste Kante jedes 2-Zyklus in $\mathcal{C}$. Dies liefert eine Menge $\mathcal{C}'$ von knotendisjunkten Kanten sowie Zyklen der Länge mindestens drei. Falls $\mathcal{C}$ nur einen 2-Zyklus enthält, so entferne zusätzlich eine Kante, die minimales Gewicht hat unter allen Kanten in Zyklen der Länge mindestens drei.
3. Konstruiere eine 3-Zyklenüberdeckung $\mathcal{T}$, indem die Pfade in $\mathcal{C}'$ zu einem großen Zyklus beliebig zusammengefügt werden.

Abbildung 25: Max-3-DCC: Algorithmus 1

2-Zyklus, so würde dies nur wieder den gleichen 2-Zyklus ergeben. In diesem Fall wird noch einer der längeren Zyklen zusätzlich aufgebrochen. Ähnlich wie in Theorem 4.1 ergibt sich folgende Abschätzung der Approximationsgüte von Algorithmus 1. Der zusätzliche Abzug von $\frac{1}{n}$ rührt daher, daß eventuell eine Kante eines 3-Zyklus aufgebrochen werden muß.

Theorem 4.9 *Algorithmus 1 berechnet eine $(1 - c - \frac{1}{n})$-Approximation einer 3-Zyklenüberdeckung maximalen Gewichts von G.* ■

4.3.2 Algorithmus 2

Algorithmus 2 widmet den 2-Zyklen besondere Aufmerksamkeit. Zuerst wird wieder eine optimale 2-Zyklenüberdeckung $\mathcal{C}$ berechnet. Wie im Falle von Algorithmus 2 für Max-ATSP soll $\mathcal{C}$ wiederum mit einem Matching kombiniert werden. Dazu wird wieder die Gewichtsfunktion w modifiziert: w' weist jeder Kante eines 2-Zyklus von $\mathcal{C}$ Gewicht null zu, für alle anderen Kanten e gilt $w'(e) = w(e)$. Dann wird ein Matching $\mathcal{M}$ maximalen Gewichts in bezug auf w' berechnet. Danach werden die Kanten von $\mathcal{M}$ und $\mathcal{C}$ so mit zwei Farben gefärbt, daß jede der beiden Farbklassen auf knotendisjunkten Pfaden und Zyklen der Länge mindestens drei besteht. Letzterer Punkt stellt die entscheidende Verbesserung im Vergleich zu Algorithmus 2 für Max-ATSP dar: Da in den Farbklassen auch Zyklen zugelassen sind, braucht von den Zyklen in $\mathcal{C}$ keine Kante entfernt zu werden. Leider ermöglicht das weiter unten folgende Färbungslemma keine Kopplung der minimalen Länge der Zyklen in $\mathcal{C}$ und der minimalen Länge der Zyklen in den beiden

Eingabe: gerichteter Graph $G = (V, K(V))$ mit
Gewichtsfunktion $w : K(V) \to \mathbb{Q}_{\geq 0}$.
Ausgabe: eine 3-Zyklenüberdeckung $\mathcal{T}$.

1. Berechne eine 2-Zyklenüberdeckung $\mathcal{C}$ maximalen Gewichts von G.
2. Definiere eine neue Gewichtsfunktion w' wie folgt: Für alle $i \in I_2$ erhalten die beiden Kanten in C_i Gewicht null unter w'. Alle anderen Kanten behalten ihr ursprüngliches Gewicht
3. Berechne ein Matching maximalen Gewichts von G bezüglich w'.
4. Setze $\mathcal{M}' = \mathcal{M} \setminus \mathcal{C}$.
5. Färbe die Kanten von $\mathcal{M}' \cup \mathcal{C}$ gemäß Lemma 4.10.
6. Füge jede Kante $e \in \mathcal{M} \setminus \mathcal{M}'$ zu der Farbklasse hinzu, in der e nicht schon enthalten ist.
7. Füge die Pfade in der Farbklasse mit dem größeren Gesamtgewicht zu einem großen Zyklus zusammen. Dies liefert $\mathcal{T}$. (Falls nötig, breche einen Zyklus auf wie in Algorithmus 1.)

Abbildung 26: Max-3-DCC: Algorithmus 2

Farbklassen, so daß diese Verbesserung nur für Max-3-DCC funktioniert.

Das nächste Lemma zeigt, daß eine Färbung wie in Algorithmus 2 beschrieben immer existiert.

Lemma 4.10 *Sei $G = (V, E)$ ein gerichteter schleifenfreier Graph, so daß*

1. *alle Knoten Ein- und Ausgrad höchstens zwei und*
2. *totalen Grad höchstens drei haben.*

Dann können die Kanten von G mit zwei Farben gefärbt werden, so daß jede Farbklasse aus knotendisjunkten Pfaden sowie Zyklen der Länge mindestens drei besteht.

Beweis. Wir konstruieren einen Konfliktgraphen $A = (E, F)$ wie in Lemma A.1. Wie dort bemerkt, besteht A aus A-Pfaden sowie A-Zyklen gerader Länge. Färbt man diese A-Pfade und A-Zyklen alternierend, so folgt, daß man

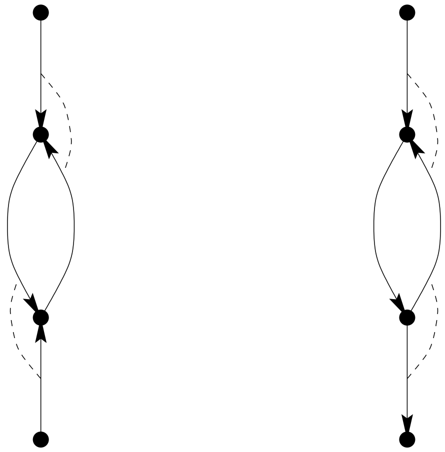

Abbildung 27: Ein 2-Zyklus kann auf zwei Arten mit den anderen Kanten interagieren. (Durchgezogene Kanten sind Kanten von G, gestrichelte Kanten stellen die resultierenden Kanten in A dar.)

die Kanten von G so mit zwei Farben färben kann, daß jede Farbklasse aus knotendisjunkten Pfaden und Zyklen besteht.

Aber jede der beiden Klassen kann noch 2-Zyklen enthalten, die wir nun zu eliminieren haben. Für jeden A-Pfad und A-Zyklus in A gibt es zwei Möglichkeiten, wie man den A-Pfad und A-Zyklus färben kann. Dies ist die entscheidende Beobachtung, um die 2-Zyklen zu eliminieren.

Abbildung 27 zeigt, wie ein 2-Zyklus in G mit den anderen Kanten interagieren kann. Wegen der Gradbeschränkungen in G, kann jeder der zwei Knoten des 2-Zyklus mit maximal einer weiteren Kante inzident sein. Der linke Fall in Abbildung 27 beinhaltet auch den Fall, in dem beide Kanten, die nicht zum 2-Zyklus gehören, in die entgegengesetzte Richtung zeigen. Die rechte Seite behandelt auch den Fall, in dem eine oder beide Kanten, die nicht zum 2-Zyklus gehören, fehlen.

Der Fall auf der rechten Seite kann leicht abgehandelt werden, da hier nur ein langer A-Pfad und ein einzelner A-Knoten in A vorliegt. Dieser A-Knoten entspricht gerade der linken Kante in dem 2-Zyklus in Abbildung 27. Da wir diese Kante beliebig färben können, geben wir ihr einfach die Farbe, die die andere Kante des 2-Zyklus nicht hat.

Die Konfiguration im linken Fall entspricht zwei A-Pfaden in A, deren beide A-Endknoten gerade die Kanten des 2-Zyklus sind. Damit diese beiden A-Endknoten unterschiedliche Farben erhalten, fügen wir die A-Kante (e, f) zu der Kantenmenge von A hinzu, wobei $e, f \in E$ die beiden Kanten des 2-Zyklus sind. Dies kann natürlich neue A-Zyklen in A entstehen lassen. Allerdings kann die neu eingefügte Kante als das Ereignis interpretiert werden, daß die Anfangs- bzw. Endknoten der zwei Kanten e und f identisch sind. Also haben die neu entstandenen A-Zyklen alle gerade Länge.

Folglich können die Kanten von G mit zwei Farben gefärbt werden, so daß jede Farbklasse aus knotendisjunkten Pfaden sowie Zyklen der Länge mindestens drei besteht. ■

Um die Approximationsgüte der von Algorithmus 2 berechneten 3-Zyklenüberdeckung abzuschätzen, benötigen wir eine gute untere Schranke für das Gewicht des Matchings $\mathcal{M}$.

Lemma 4.11 *Es sei $\mathcal{T}_{\text{opt}}$ eine 3-Zyklenüberdeckung maximalen Gewichts in G und $w(\mathcal{T}_{\text{opt}}) = L$. Es sei $I_2' \subseteq I_2$ die Menge aller 2-Zyklen C von $\mathcal{C}$, so daß C und $\mathcal{T}_{\text{opt}}$ eine Kante gemeinsam haben. Weiterhin sei $b' = \sum_{i \in I_2'} b_i$ und $c' = \sum_{i \in I_2'} c_i$. Dann gilt für das von Algorithmus 2 berechnete Matching $\mathcal{M}$*

$$w'(\mathcal{M}) \geq \tfrac{1}{2} \cdot L - \tfrac{1}{6} \cdot W + \tfrac{1}{6}(c' - 2b') \cdot W.$$

Beweis. Wir teilen die Zyklen in $\mathcal{T}_{\text{opt}}$ in zwei Klassen $\mathcal{T}_1$ und $\mathcal{T}_2$. $\mathcal{T}_1$ ist die Menge aller Zyklen, die eine Kante mit einem 2-Zyklus aus $\mathcal{C}$ gemeinsam haben, und $\mathcal{T}_2 = \mathcal{T}_{\text{opt}} \setminus \mathcal{T}_1$. Es gilt $w'(\mathcal{T}_{\text{opt}}) \geq L - b' \cdot W$, da ein Zyklus der Länge mindestens drei nur durch eine Kante eines 2-Zyklus gehen kann. Andererseits ist $w'(\mathcal{T}_2) \leq (1 - c' - b') \cdot W$, da wir sonst die Zyklen C_i mit $i \in I_2'$ zu $\mathcal{T}_2$ hinzufügen könnten und eine Zyklenüberdeckung mit Gewicht größer W erhalten würden, ein Widerspruch. Es sei $D = w'(\mathcal{T}_2)$. (Nach Konstruktion gilt auch $D = w(\mathcal{T}_2)$.) Damit gilt $w'(\mathcal{T}_1) \geq L - b' \cdot W - D$.

Wir konstruieren nun ein Matching $\mathcal{N}$ mit $w'(\mathcal{N}) \geq \frac{1}{2} \cdot L - \frac{1}{6} \cdot W + \frac{1}{6}(c' - 2b') \cdot W$. Dies beweist die Aussage des Lemmas. Die Kanten von $\mathcal{T}_2$ können so mit drei Farben gefärbt werden, daß jede Farbklasse ein (partielles) Matching bildet. (Der Worst Case ist hier eine Menge von 3-Zyklen.) Es sei $\mathcal{N}_2$ die Farbklasse mit dem größten Gewicht. Es gilt $w'(\mathcal{N}_2) \geq \frac{1}{3} D$. Da die Zyklen in $\mathcal{T}_1$ alle eine Kante mit Gewicht null bezüglich w' haben, können deren Kanten mit positivem Gewicht mit zwei Farben so gefärbt werden, daß jede Farbklasse ein (partielles) Matching bildet. Es sei $\mathcal{N}_1$ die Farbklasse mit dem größeren Gewicht. Es gilt $w'(\mathcal{N}_1) \geq \frac{1}{2}(L - b' \cdot W - D)$. Damit hat $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$ mindestens Gewicht

$$\begin{aligned} w'(\mathcal{N}) &\geq \tfrac{1}{2} \cdot L - \tfrac{1}{2} b' \cdot W - \tfrac{1}{6} D \\ &\geq \tfrac{1}{2} \cdot L - \tfrac{1}{6} \cdot W + \tfrac{1}{6}(c' - 2b') \cdot W, \end{aligned}$$

wobei die zweite Ungleichung per $D \leq (1 - c' - b') \cdot W$ folgt. ■

Das folgende Theorem schätzt die Approximationsgüte der von Algorithmus 2 berechneten 3-Zyklenüberdeckung ab.

Theorem 4.12 *Seien b' und c' wie in Lemma 4.11 definiert. Algorithmus 2 berechnet eine $(\frac{2}{3} + \frac{1}{12}(c' - 2b') - \frac{1}{n})$-Approximation einer 3-Zyklenüberdeckung maximalen Gewichts von G.*

Beweis. Nach Schritt 6 von Algorithmus 2 enthalten beide Farbklassen zusammen alle Kanten aus $\mathcal{C}$ und diejenigen Kanten von $\mathcal{M}$, die nicht auch Teil eines 2-Zyklus von $\mathcal{C}$ sind. Daher ist das Gesamtgewicht beider Farbklassen $w(\mathcal{C})+w'(\mathcal{M})$. Es gilt $w(\mathcal{C}) = W$ sowie $w'(\mathcal{M}) \geq \frac{1}{2} \cdot L - \frac{1}{6} \cdot W + \frac{1}{6}(c' - 2b') \cdot W$ nach Lemma 4.11. In Schritt 7 verlieren wir eventuell Gewicht $\frac{1}{n} \cdot W$. Insgesamt ist das Gewicht in der schwereren Farbklasse mindestens

$$\frac{1}{2}(W + \frac{1}{2} \cdot L - \frac{1}{6} \cdot W + \frac{1}{6}(c' - 2b') \cdot W - \frac{1}{n} \cdot W) \geq \frac{2}{3} \cdot L + \frac{1}{12}(c' - 2b') \cdot L - \frac{1}{n} \cdot L,$$

da $W \geq L$. ∎

4.3.3 Gesamtalgorithmus

Der Gesamtalgorithmus arbeitet wie folgt. Es werden Algorithmus 1 und 2 auf G ausgeführt und die schwerere der beiden 3-Zyklenüberdeckungen ausgegeben.

Um die Analyse zu vereinfachen, lassen wir im folgenden die Terme $\frac{1}{n} \cdot W$ weg. Dies hat keinen Einfluß auf die asymptotische Approximationsgüte.

Die Approximationsgüte des Gesamtalgorithmus kann nun wie folgt abgeschätzt werden:

$$\text{Minimiere} \quad \max\{1 - c, \tfrac{2}{3} + \tfrac{1}{12}(c' - 2b')\} \tag{12}$$

über alle c, b' und c' für die gilt:

$$\begin{aligned} 0 \leq c' &\leq b', \\ c' + b' &\leq 1, \\ 0 \leq c &\leq \tfrac{1}{2}(1 - b' + c'). \end{aligned}$$

Es läßt sich leicht zeigen, daß dieses Minimum $\frac{3}{5}$ beträgt. Es wird für $b' = \frac{3}{5}$ und $c = c' = \frac{2}{5}$ angenommen. (Siehe auch Anhang B.2.)

Theorem 4.13 *Die schwerste der zwei 3-Zyklenüberdeckungen, die von Algorithmus 1 und 2 auf Eingabe G berechnet werden, ist eine $\frac{3}{5}$-Approximation einer optimalen 3-Zyklenüberdeckung.* ∎

Die Laufzeit des Algorithmus wird wiederum bestimmt von der Zeit, eine Zyklenüberdeckung und ein Matching maximalen Gewichts zu berechnen, beträgt also z.B. $O(n^3)$ oder $O(n^{5/2}\log(nB))$, wobei B das größte Gewicht einer Kante in G ist [1].

5 Partielle Zyklenüberdeckungen

Hauptaugenmerk in diesem Kapitel gilt der Komplexität der Berechnung partieller Zyklenüberdeckungen. Gegeben ist ein Graph $G = (V, E)$ mit n Knoten und m

Kanten, der sowohl ungerichtet als auch gerichtet sein kann, mit einer Gewichtsfunktion $w : E \to \mathbb{Q}$, die *nicht* notwendigerweise die Dreiecksungleichung erfüllen muß. Es ist hier nicht erforderlich, daß G vollständig ist, mit anderen Worten, der Wert ∞ ist auch als Kantengewicht zugelassen. Weiterhin sind natürliche Zahlen k und t gegeben. Wir suchen nun eine partielle k-Zyklenüberdeckung, die genau t Knoten von G abdeckt und minimales Gewicht hat, wobei das Minimum über alle k-Zyklenüberdeckungen genommen wird, die genau t Knoten abdecken. Da für $k \geq 3$ (gerichteter Fall, siehe [64]) bzw. $k \geq 6$ (ungerichteter Fall, siehe [23]) die entsprechenden Entscheidungsprobleme für $t = n$ schon NP-vollständig sind, gibt es für diese Probleme vermutlich keine exakten Polynomialzeitalgorithmen. Anstatt nach Approximationsalgorithmen zu suchen, gehen wir hier einen anderen Weg: Ziel ist es zu zeigen, daß die Probleme *fixed parameter tractable* mit Parameter t sind, d.h. wir suchen nach exakten Algorithmen, deren Laufzeit polynomiell in n, m und k ist, aber beliebig in t sein kann. Wir betrachten hier nur Minimierungsprobleme. Da alle Algorithmen aber exakt sind, gelten die Resultate auch für die entsprechenden Maximierungsprobleme.

5.1 Fixed Parameter Tractability

Die nächsten zwei Theoreme zeigen, daß die Berechnung t-partieller k-Zyklenüberdeckungen fixed parameter tractable mit Parameter t ist. (Da G nicht unbedingt vollständig ist, muß solch eine Zyklenüberdeckung nicht immer existieren.) Die Laufzeiten sind $O(e^t(2^t \cdot n \cdot m + 4^t))$ (randomisiert) bzw. $2^{O(t)} \cdot n \cdot m \cdot \log n$ (deterministisch). Dieses Resultat kann man auch wie folgt interpretieren: Solange $t = O(\log n)$ gilt, ist die Laufzeit polynomiell, d.h. die Berechnung $O(\log n)$-partieller k-Zyklenüberdeckungen minimalen Gewichts ist in P.

Der Algorithmus basiert auf dem sog. *Colour Coding*, das von Alon, Yuster und Zwick [3] entwickelt worden ist, um kleine Subgraphen beschränkter Baumweite wie z.B. Zyklen in einem gegebenen Graphen zu finden. Wir erweitern diese Technik entsprechend, um nicht nur bestimmte Subgraphen (z.B. Zyklen der Länge t) sondern auch Subgraphen aus einer ganzen Klasse (hier t-partielle k-Zyklenüberdeckungen) zu finden. Wir weisen jedem Knoten von G zufällig eine Farbe aus $1, \ldots, t$ zu. Mit Hilfe von dynamischer Programmierung bestimmen wir dann eine k-Zyklenüberdeckung minimalen Gewichts, die jeweils einen Knoten von jeder Farbe enthält und in der alle Knoten paarweise verschiedene Farben haben. Da die Knoten in solch einer k-Zyklenüberdeckung notwendigerweise paarweise verschieden sind, hat die k-Zyklenüberdeckung auch genau t Knoten. Die Wahrscheinlichkeit, daß die Knoten einer optimalen t-partiellen k-Zyklenüberdeckung alle paarweise verschiedene Farben erhalten, ist $t!/t^t > e^{-t}$. Folglich ist die erwartete Anzahl von Iterationen, bis dieser Fall eintritt, e^t.

Theorem 5.1 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph mit n Knoten und m Kanten. Es gibt einen randomisierten Algorithmus mit Lauf-*

zeit $O(e^t(2^t \cdot n \cdot m + 4^t))$, *der eine* t*-partielle* k*-Zyklenüberdeckung minimalen Gewichts von* G *mit Wahrscheinlichkeit mindestens* $1-(1-e^{-t})^{e^t+1} \geq 1-e^{-1}$ *ausgibt, sofern solch eine Überdeckung existiert.*

Beweis. Wir weisen jedem Knoten $v \in V$ zufällig eine Farbe $c(v)$ aus $\{1, \ldots, t\}$ zu. Wir nehmen an, daß die t Knoten einer t-partiellen k-Zyklenüberdeckung minimalen Gewichts paarweise verschiedene Farben unter c erhalten haben. Dies geschieht mit Wahrscheinlichkeit mindestens e^{-t}. Für $C \subseteq \{1, \ldots, t\}$ mit $|C| \geq k$ bezeichne $W(C)$ das Gewicht einer $|C|$-partiellen k-Zyklenüberdeckung, so daß die Farben der überdeckten Knoten gerade die Farben in C sind. Es sei $T(C)$ das Gewicht eines $|C|$-Zyklus minimalen Gewichts, so daß die Farben der überdeckten Knoten gerade die Farben aus C sind. (Letzteres Problem ist gerade ein partielles TSP.)

Der Algorithmus von Alon, Yuster und Zwick [3], um einfache Zyklen gegebener Länge zu finden, kann so modifiziert werden, daß er auch die Werte $T(C)$ berechnet (siehe auch die Bemerkungen am Ende von [3, Abschnitt 6]). Für jede gerichtete Kante (u, v) bzw. ungerichtete Kante $\{u, v\}$ in E mit $c(u) \neq c(v)$ und jede Menge $C \subseteq \{1, \ldots, t\} \setminus \{c(u), c(v)\}$, sei $P(u, v, C)$ das Gewicht eines Pfads minimalen Gewichts mit $|C|$ inneren Knoten, der in u beginnt und in v endet und bei dem die Farben der inneren Knoten gerade die Farben aus C sind. Es gilt

$$P(u, v, C) = \min\{\, P(u, x, C \setminus \{c(x)\}) + w((x, v)) \mid (x, v) \in E,\ c(x) \in C \,\} \quad (13)$$

mit $P(u, v, \emptyset) = w((u, v))$ für alle u, v. (Für ungerichtete Graphen gilt die Gleichung entsprechend.) Falls in (13) das Minimum über die leere Menge genommen wird, so setzen wir $P(u, v, C) = \infty$.

Die Gesamtlaufzeit, um für ein festes u alle Werte $P(u, v, C)$ zu berechnen, beträgt $O(2^t \cdot m)$. Also können alle Werte $P(u, v, C)$ in Zeit $O(2^t \cdot n \cdot m)$ berechnet werden.

Aus den Werten $P(u, v, C)$ können die $T(C)$ per

$$\begin{aligned} T(C) = \min\{P(u, v, C \setminus \{c(u), c(v)\}) + w((v, u)) \mid \\ (v, u) \in E,\ c(u) \neq c(v),\ c(u), c(v) \in C\,\} \end{aligned}$$

in Zeit $O(2^t \cdot m)$ berechnet werden.

Indem immer ein Knoten als Zeuge abgespeichert wird, für den das Minimum angenommen wird, können wir auch immer einen Pfad bzw. Zyklus berechnen, für den der Wert $P(u, v, C)$ bzw. $T(C)$ angenommen wird, sofern dieser nicht ∞ beträgt.

Nun können die $W(C)$ mittels dynamischer Programmierung wie folgt berechnet werden. Für $k \leq |C| \leq 2k - 1$ gilt $W(C) = T(C)$. Ist $|C| \geq 2k$, so besteht eine $|C|$-partielle k-Zyklenüberdeckung aus einem großen $|C|$-Zyklus oder wir überdecken die Knoten mit einem Zyklus der Länge $\geq k$ und überdecken den

Rest rekursiv. Die entsprechende Formel lautet

$$W(C) = \min\{\, T(A) + W(B) \mid A \cup B = C,\ A \cap B = \emptyset,\ |A|, |B| \geq k \text{ oder } B = \emptyset \,\}, \tag{14}$$

wobei wir aus technischen Gründen $W(\emptyset) = 0$ setzen. Indem wir jeweils die Mengen abspeichern, für die das Minimum angenommen wird, können wir auch die entsprechende Zyklenüberdeckung rekonstruieren.

Ein Wert $W(C)$ kann aus den $T(A)$ in Zeit $O(2^t)$ berechnet werden, alle $W(C)$ in Zeit $O(2^{2t})$. Die Gesamtlaufzeit beträgt damit $O(2^t \cdot n \cdot m + 4^t)$.

Wiederholt man den obigen Prozeß $\lceil e^t \rceil + 1$ Mal, so ergibt sich eine Erfolgswahrscheinlichkeit von mindestens $1 - e^{-1}$. ∎

Der obige Algorithmus kann mit nur einer geringen Erhöhung der Laufzeit derandomisiert werden. Man benötigt dazu nur eine Liste von Färbungen, so daß für jede Menge $W \subseteq V$ mit $|W| = t$ mindestens eine der Färbungen den Elementen von W paarweise verschiedene Farben zuweist. Mit anderen Worten, wir brauchen eine t-perfekte Familie von Hashfunktionen $\{1, \ldots, n\} \to \{1, \ldots, t\}$. Alon, Yuster und Zwick [3, Abschnitt 4] entwerfen solch eine Familie der Größe $2^{O(t)} \log n$, die effizient konstruiert werden kann. Jede dieser Hashfunktionen kann in konstanter Zeit ausgewertet werden. Dies impliziert das folgende Theorem.

Theorem 5.2 *Es sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph mit n Knoten und m Kanten. Es gibt einen deterministischen Algorithmus mit Laufzeit $2^{O(t)} \cdot n \cdot m \cdot \log n$, der eine t-partielle k-Zyklenüberdeckung minimalen Gewichts von G ausgibt, sofern solch eine Überdeckung existiert.* ∎

Das folgende Korollar erhalten wir, indem wir $k \geq t/2$ wählen.

Korollar 5.3 *Eine t-partielle Hamiltonsche Tour minimalen Gewichts kann deterministisch in Zeit $2^{O(t)} \cdot n \cdot m \cdot \log n$ berechnet werden.* ∎

5.2 Weitere partielle Überdeckungsprobleme

Die Techniken des letzten Abschnitts lassen sich auch auf andere Überdeckungsprobleme übertragen. Wir betrachten hier das partielle Set Cover Problem, das als wichtige Spezialfälle das partielle Vertex Cover und das partielle Dominating Set Problem abdeckt.

Gegeben sind m Mengen $S_1, \ldots, S_m$ über einem endlichen Universum U, das aus n Elementen $u_1, \ldots, u_n$ besteht. Jedes dieser Elemente aus U tritt in mindestens einer der Mengen $S_1, \ldots, S_m$ auf. Mit jeder Menge S_μ assoziieren wir ein nicht-negatives Gewicht $w(S_\mu)$. Unser Ziel ist es, Mengen $S_{i_1}, \ldots, S_{i_\ell}$ zu finden, die mindestens t Elemente aus U überdecken, d.h. $|\bigcup_{\lambda=1}^{\ell} S_{i_\lambda}| \geq t$, so daß das Gesamtgewicht $\sum_{\lambda=1}^{\ell} w(S_{i_\lambda})$ minimiert wird unter allen Familien von Mengen, die mindestens t Elemente überdecken. Solch eine Familie nennen wir auch *t-partielle Mengenüberdeckung* minimalen Gewichts.

Theorem 5.4 *Seien $S_1, \ldots, S_m$ Teilmengen eines Universums U der Kardinalität n. Dann gibt es einen randomisierten Algorithmus mit einer Laufzeit von $O(e^k(2^k \cdot k + n)m)$, der mit einer Wahrscheinlichkeit von mindestens $1 - e^{-1}$ eine t-partielle Mengenüberdeckung minimalen Gewichts berechnet.*

Beweis. Wiederum geben wir zufällig jedem Element aus $u \in U$ eine Farbe $c(u)$ aus $\{1, \ldots, t\}$. Wir nehmen an, daß t der Elemente, die von einer optimalen t-partiellen Mengenüberdeckung überdeckt werden, paarweise verschiedene Farben haben. Wie weiter oben bemerkt, geschieht dies mit Wahrscheinlichkeit mindestens e^{-t}.

Für jedes $C \subseteq \{1, \ldots, t\}$ sei $W(C)$ das Gewicht einer Überdeckung minimalen Gewichts, die mindestens ein Element jeder Farbe in C überdeckt. Für jede Menge S_μ, wobei $1 \leq \mu \leq m$, sei $T_\mu \subseteq \{1, \ldots, t\}$ die Menge der Farben der Elemente, die von S_μ abgedeckt werden, d.h. $T_\mu = c(S_\mu) = \{c(u) \mid u \in S_\mu\}$.

Wir berechnen nun die Gewichte $W(C)$ mittels dynamischer Programmierung. Für ein gegebenes C nehmen wir jede Menge S_i mit $1 \leq i \leq m$, entfernen die Farben aus C, die von S_i überdeckt werden, und versuchen nun, die übrig gebliebenen Farben rekursiv zu überdecken. Also gilt

$$W(C) = \min\{\, W(C \setminus T_i) + w(S_i) \mid 1 \leq i \leq m,\ C \cap T_i \neq \emptyset \,\}, \tag{15}$$

wobei $W(\emptyset) = 0$. Weil wir angenommen haben, daß den überdeckten Elementen einer t-partielle Mengenüberdeckung minimalen Gewichts paarweise verschiedene Farben zugeordnet worden sind, gibt es immer mindestens ein T_i mit $C \cap T_i \neq \emptyset$ in (15), da sonst die Mengen $S_1, \ldots, S_m$ nicht ganz U überdecken würden. Schließlich ist $W(\{1, \ldots, t\})$ das Gewicht einer optimalen t-partiellen Mengenüberdeckung, da Elemente, die verschiedene Farben haben, auch notwendigerweise verschieden sind.

Indem wir jeweils einen Index i_0 speichern, für den in (15) das Minimum angenommen wird, können wir auch eine optimale t-partielle Mengenüberdeckung rekonstruieren.

Um alle Werte $W(C)$ zu berechnen, sind 2^t Schritte wie in (15) notwendig. Ein solcher Schritt kann in Zeit $O(t \cdot m)$ ausgeführt werden, da wir m Mengen testen, deren Größe maximal t ist. Zuvor müssen wir die Mengen T_μ aus den S_μ berechnen, was in Zeit $O(n \cdot m)$ geht. Folglich ist die Gesamtlaufzeit $O((2^t \cdot t + n)m)$.

Wiederholen wir die obigen Schritte jeweils $\lceil e^k \rceil + 1$ Mal, so beträgt die Wahrscheinlichkeit, in einer dieser Runden erfolgreich zu sein, mindestens $1 - e^{-1}$. ∎

Der obige Algorithmus kann, wie zuvor beschrieben, wiederum effizient derandomisiert werden.

Theorem 5.5 *Eine t-partielle Mengenüberdeckung minimalen Gewichts kann deterministisch in Zeit $2^{O(t)} \cdot n \cdot m \cdot \log n$ berechnet werden.* ∎

Da das t-partielle Vertex Cover Problem und das t-partielle Dominating Set Problem Spezialfälle des t-partiellen Set Cover Problems sind, folgen unmittelbar die weiteren Korollare.

Korollar 5.6 *Eine t-partielle Knotenüberdeckung minimalen Gewichts eines Graphen $G = (V, E)$, wobei jeder Knoten ein nicht-negatives Gewicht besitzt, kann deterministisch in Zeit $2^{O(t)} \cdot |E| \cdot |V| \cdot \log |E|$ berechnet werden.* ∎

Korollar 5.7 *Eine t-partielle dominierende Menge minimalen Gewichts eines gerichteten oder ungerichteten Graphen $G = (V, E)$, wobei jeder Knoten ein nicht-negatives Gewicht besitzt, kann deterministisch in Zeit $2^{O(t)} \cdot |V|^2 \cdot \log |V|$ berechnet werden.* ∎

A Ein algorithmischer Beweis von Lemma 4.3

In diesem Kapitel wird gezeigt, wie das „path coloring lemma" (Lemma 4.3) von Kosaraju, Park und Stein [47] so implementiert werden kann, daß die Laufzeit polynomiell ist. Dies erscheint notwendig, da Kosaraju, Park und Stein dieses Lemma weder beweisen noch eine Laufzeitabschätzung geben (allerdings implizit benutzen, daß die Laufzeit polynomiell ist).

A.1 Eine erste Färbung

Es bezeichne $G = (V, E)$ den zu färbenden Graphen. Wir definieren zuerst einen *Konfliktgraphen* $A_1 = (E, F_1)$ zu G. Die Knotenmenge von A_1 ist die Kantenmenge E von G. Wenn wir im folgenden von einer Kante sprechen, so meinen wir damit ein Element aus E. Betrachten wir eine Kante aus E als Knoten von A_1, so reden wir auch von einem A-Knoten. In gleicher Weise nennen wir zur besseren Unterscheidung die Kanten in F_1 (und allen weiteren Konfliktgraphen) auch A-Kanten sowie Zyklen und Pfade in A_1 auch A-Zyklen bzw. A-Pfade. Die Menge F_1 ist nun wie folgt definiert: Es gibt eine A-Kante $\{e, f\}$ in F_1 genau dann, wenn es Knoten u, v und w in V gibt, so daß $e = (u, v)$ und $f = (w, v)$ oder $e = (v, u)$ und $f = (v, w)$, mit anderen Worten, e und f müssen andere Farben erhalten, weil Ihre Anfangs- bzw. Endknoten übereinstimmen.

Jeder A-Knoten in A_1 hat Grad höchstens zwei aufgrund der Gradbeschränkungen in G. Also besteht A_1 aus einfachen A-Pfaden und A-Zyklen. Gemäß Konstruktion haben alle A-Zyklen in A_1 gerade Länge. Dies folgt aus der Tatsache, daß eine A-Kante in F_1 dem Zusammentreffen zweier Anfangs- oder Endknoten von zwei Kanten aus E entspricht. Dies zeigt bereits, daß wir die Kanten von G mit zwei Farben färben können, so daß jede Farbklasse aus knotendisjunkten Pfaden und Zyklen besteht. Dazu müssen wir nur die A-Knoten in jedem A-Pfad und A-Zyklus abwechselnd färben.

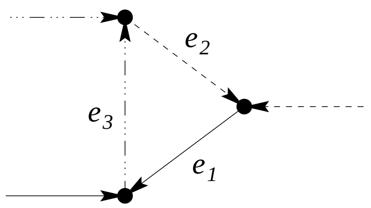

Abbildung 28: Pfade in A_1 (durchgezogen, gestrichelt sowie gepunktet-gestrichelt), die einen Zyklus in G' bilden. Die gezeichneten Knoten und Kanten sind Knoten und Kanten aus G'. Zwei solche Kanten sind durch eine A-Kante verbunden, falls diese im selben Stil (durchgezogen, gestrichelt, gepunktet-gestrichelt) gezeichnet sind.

Lemma A.1 *Die Kanten von G können so mit zwei Farben gefärbt werden, daß jede Farbklasse nur aus knotendisjunkten Pfaden und Zyklen besteht. Solch eine Färbung kann in Polynomialzeit gefunden werden.* ■

Es verbleibt, die Zyklen in den Farbklassen zu eliminieren. Dazu wird die folgende Beobachtung entscheidend sein: Jeder A-Pfad und jeder A-Zyklus in A_1 kann auf zwei verschiedene Art und Weisen gefärbt werden, und davon kann eine beliebig gewählt werden.

A.2 Eine zweite Färbung

Es sei Z die Menge aller A-Knoten, die in einem A-Zyklus von A_1 enthalten sind. P bezeichne die Menge aller A-Knoten, die zu einem A-Pfad der Länge mindestens eins in A_1 gehören. (Die Länge eines A-Pfades ist hier die Anzahl der A-Kanten in dem A-Pfad.) Schließlich sei S die Menge aller A-Knoten in A-Pfaden der Länge null, also die Menge aller isolierten A-Knoten in A_1. Zu einem A-Pfad $e_1, \ldots, e_\ell$ heißen e_1 und e_ℓ die A-Endknoten dieses Pfades.

Als nächstes zeigen wir, daß $G' := (V, Z \cup P)$ 2-pfadfärbbar ist. Die einzige Möglichkeit, wie ein Zyklus in G' in einer Farbklasse entstehen kann, zeigt Abbildung 28. Solch ein Zyklus besteht komplett aus A-Endknoten eines A-Pfades in P. Weil es keine 2-Zyklen in G gibt, hat jeder dieser Zyklen mindestens drei Kanten.

Um nun das Entstehen von Zyklen in einer Farbklasse zu verhindern, haben wir nur dafür zu sorgen, daß zwei der Kanten in jedem Zyklus wie in Abbildung 28 unterschiedliche Farben erhalten. Diese Zyklen in G' sind knotendisjunkt; dies folgt aus der Gradbeschränkung in G und der Tatsache, daß jeder A-Pfad mit Knoten in P die Länge mindestens eins hat, d.h. aus mindestens zwei Kanten in G besteht.

Wenn $e_1, \ldots, e_\ell$ die Kanten eines Zyklus in G' sind, so verbinden wir zwei aufeinanderfolgende dieser Kanten in A_1, d.h. wir fügen z.B. die A-Kante (e_1, e_2)

zu F_1 hinzu. Da die Zyklen (knoten- und kanten-) disjunkt sind, gibt es maximal $|E|/3$ viele Zyklen, und wir können diese in Polynomialzeit finden. Es bezeichne $A_2 = (Z \cup P, F_2)$ den resultierenden Konfliktgraphen. Verglichen mit A_1 kann A_2 neue A-Zyklen enthalten. Aber auch diese haben gerade Länge, da das obige Hinzufügen von (e_1, e_2) als Zusammentreffen von zwei Anfangs- bzw. Endknoten interpretiert werden kann.

Lemma A.2 *Der Graph $G' = (V, Z \cup P)$ ist 2-pfadfärbbar. Eine solche Färbung kann in Polynomialzeit berechnet werden.* ∎

Das obige Hinzufügen neuer A-Kanten hat – wie schon erwähnt – evtl. neue A-Zyklen kreiert. Wir transferieren die entsprechenden A-Knoten von P nach Z. Da keine Verwechslungsgefahr besteht, nennen wir die entsprechenden Mengen wieder Z und P.

A.3 Die entgültige Färbung

Um die entgültige Färbung zu erhalten, müssen wir jetzt noch die Kanten aus S zu G' hinzufügen. Dies geschieht zuerst für den Spezialfall, daß jeder Zyklus in G mindestens zwei Kanten aus S enthält. Danach werden wir den allgemeinen Fall auf diesen Spezialfall reduzieren.

Nehmen wir also an, daß auf jedem Zyklus in G mindestens zwei Kanten aus S liegen. Wir färben die Kanten in $Z \cup P$ gemäß Lemma A.2. (Wir wählen dabei für jeden A-Pfad und A-Zyklus eine der beiden möglichen Färbungen beliebig aus.) Bezüglich dieser Färbung wird nun ein weiterer gerichteter Hilfsgraph B konstruiert: Die A-Knoten von B sind die Kanten in S. Es gibt A-Kante (e, f) in B, falls es einen einfarbigen Pfad in G vom Endknoten von e zum Anfangsknoten von f gibt. Dabei heißt ein Pfad in G *einfarbig*, falls alle Kanten des Pfades die gleiche Farbe haben. Die A-Kante (e, f) erhält die gleiche Farbe wie der entsprechende Pfad in G. Falls es zwei Kanten der Form $e = (u, v)$ und $f = (v, w)$ in S gibt, so fügen wir zwei Kanten von e nach f zu B hinzu, eine in jeder Farbe. Dabei können keine weiteren Kanten in G mit v inzident sein, denn sonst wäre entweder e oder f nicht in S.

Ein A-Zyklus in B heißt einfarbig, falls alle A-Kanten dieses Zyklus die gleiche Farbe haben. Da jeder Zyklus in G nach Annahme mindestens zwei Kanten aus S enthält, kann B keine A-Schleifen enthalten. Aufgrund der Gradbeschränkung in G hat jeder A-Knoten e in B Ein- und Ausgrad höchstens zwei. Falls zwei A-Kanten in einen A-Knoten von B hineinführen, so haben sie unterschiedliche Farben. Gleiches gilt für die A-Kanten, die den A-Knoten verlassen. Insbesondere liegt also jeder A-Knoten in B auf höchstens zwei einfarbigen A-Zyklen.

Nach Konstruktion von B gilt folgendes Lemma.

Lemma A.3 *Falls die A-Knoten von B mit zwei Farben gefärbt werden können, so daß jeder einfarbige A-Zyklus in B mindestens einen A-Knoten der anderen Farbe enthält, dann ist G 2-pfadfärbbar.* ∎

Ausgehend von B definieren wir einen ungerichteten Multigraphen B^* wie folgt: Die Knotenmenge von B^* bilden die einfarbigen A-Zyklen von B, und es gibt eine Kante zwischen zwei solchen A-Zyklen, falls diese einen A-Knoten gemeinsam haben. Jede nicht-triviale starke Zusammenhangskomponente in B induziert eine Zusammenhangskomponente in B^* und umgekehrt. Ein A-Knoten in B heißt *frei*, falls er auf höchstens einem einfarbigen A-Zyklus in B liegt.

Lemma A.4 *Jede nicht-triviale starke Zusammenhangskomponente von B hat entweder einen freien A-Knoten, oder die induzierte Zusammenhangskomponente in B^* hat einen Zyklus oder eine Doppelkante.*

Beweis. Da B keine A-Schleifen hat, muß jede Zusammenhangskomponente in B, die keinen freien Knoten hat, zwangsläufig eine Komponente in B^* induzieren, die einen Zyklus oder eine Doppelkante, was nichts anderes als ein Zyklus der Länge zwei ist, enthält. ∎

Das nächste Lemma zeigt, daß eine Färbung wie in Lemma A.3 beschrieben immer existiert und in Polynomialzeit berechnet werden kann.

Lemma A.5 *Die A-Knoten in B können so mit zwei Farben gefärbt werden, daß jeder einfarbige A-Zyklus in B einen A-Knoten der anderen Farbe enthält. Solch eine Färbung kann in Polynomialzeit gefunden werden.*

Beweis. Wir entfernen zuerst alle A-Kanten aus B, die auf keinem einfarbigen Zyklus liegen. Es genügt, das Lemma für jede starke Zusammenhangskomponente von B zu zeigen.

Falls die Komponente einen freien A-Knoten hat, so färben wir diesen A-Knoten mit der Farbe, die der Kreis, auf dem dieser Knoten liegt, nicht hat. Dann entfernen wir den A-Zyklus und erhalten eine Menge von starken Zusammenhangskomponenten. Jede dieser Komponenten muß aber einen freien A-Knoten haben, nämlich den Knoten, der auf dem entfernten A-Zyklus lag. Folglich können wir rekursiv fortfahren.

Wenn die Komponente hingegen keinen freien A-Knoten hat, so hat die entsprechende Komponente in B^* einen Zyklus oder eine Doppelkante (die als Zyklus der Länge zwei angesehen werden kann). Nach Konstruktion hat jeder Zyklus in B^* gerade Länge, da nur A-Zyklen aus B mit unterschiedlichen Farben durch eine Kante verbunden worden sind. Wir färben die Kanten des Zyklus (welche A-Knoten in B sind) alternierend. Danach besitzt jeder A-Zyklus in B, der einen Knoten aus dem Zyklus in B^* enthält, einen A-Knoten, der mit der anderen Farbe gefärbt ist. Wir entfernen all diese A-Zyklen und erhalten eine Reihe von starken

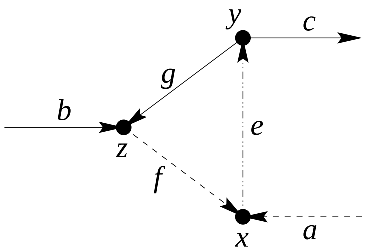

Abbildung 29: Ein Zyklus der Länge drei, der nur eine Kante aus S enthält.

Zusammenhangskomponenten, die aber nun alle einen freien Knoten enthalten. Daher können wir jetzt wie im ersten Fall fortfahren. ■

Es bleibt noch zu zeigen, wie die Annahme umgangen werden kann, daß jeder Zyklus in G mindestens zwei Kanten aus S enthält.

Lemma A.6 *Der Graph G ist 2-pfadfärbbar. Eine solche Färbung kann in Polynomialzeit berechnet werden.*

Beweis. Der Beweis erfolgt per Induktion über die Anzahl der Knoten $|V|$ von G. Falls $|V| \leq 3$, so kann eine Färbung einfach gefunden werden.

Für den Induktionsschritt sei also $|V| \geq 4$. Wir behandeln erst einige triviale Fälle. Falls es eine Kante $e = (u, v)$ in S gibt, so daß mit u oder mit v keine weitere Kante inzident ist, so spielt es keine Rolle, welche Farbe e erhält. Also muß e im folgenden nicht weiter betrachtet werden. Falls es Kanten (u, v) und (v, w) in S gibt, so kann keine weitere Kante mit v inzident sein. (Sonst wären nicht beide Kanten in S.) Daher erhalten die beiden Kanten einfach unterschiedliche Farben und können ebenfalls ignoriert werden. Es bezeichne $T \subseteq S$ die Menge der verbleibenden isolierten A-Knoten.

Für alle $t \in T$ überprüfen wir, ob vom Endknoten von t der Anfangsknoten von t in $(V, Z \cup P)$ erreicht werden kann. Falls dies für kein t der Fall ist, dann enthält jeder Zyklus in $(V, Z \cup P \cup T)$ mindestens zwei Kanten aus T. Aus den Lemmata A.3 und A.5 folgt, daß $(V, Z \cup P \cup T)$ 2-pfadfärbbar ist. Nach obigen Überlegungen ist damit auch $(V, Z \cup P \cup S)$ 2-pfadfärbbar.

Abbildung 29 zeigt einen Zyklus mit nur einer Kante aus S. Solch ein Zyklus besteht aus einer Kante aus S und hat höchstens eine Kante, die kein A-Endknoten eines A-Pfades in A_2 ist. In Abbildung 29 ist dies e. Da G keine 2-Zyklen enthält, ist entweder die Kante unmittelbar vor oder hinter e ein A-Endknoten eines A-Pfades in A_2. In Abbildung 29 ist dies f. Der Fall, in dem diese Kante hinter e liegt, ist symmetrisch. Der Zyklus kann auch mehr als drei Kanten enthalten, diese Situation wird jedoch genau gleich behandelt.

Wir entfernen die Kanten e und f sowie den Knoten x und lassen die Kante a stattdessen auf y zeigen. Es sei $\hat{G}$ der resultierende Graph (siehe Abbildung 30).

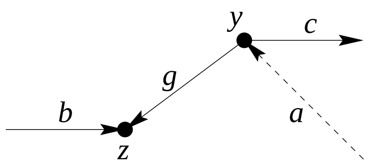

Abbildung 30: Entfernen der Kanten e und f liefert den neuen Graphen $\hat{G}$.

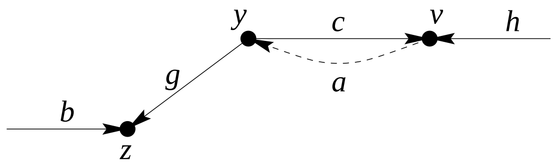

Abbildung 31: Ein 2-Zyklus entsteht. In diesem Fall ist a ein isolierter A-Knoten in $\hat{A}_2$ und wird einfach entfernt.

Der Graph $\hat{G}$ erfüllt die Gradbeschränkungen aus Lemma 4.3. Aus A_2 wird der entsprechende neue Graph $\hat{A}_2$ gewonnen, indem die A-Knoten e und f sowie die A-Kante $\{a, f\}$ entfernt werden. Wir unterscheiden zwei Fälle:

Fall 1: Falls $\hat{G}$ keine 2-Zyklen enthält, d.h. falls der Anfangsknoten von a und der Endknoten von c verschieden sind, können wir die Induktionsvoraussetzung anwenden. Dazu entfernen wir, falls a durch obigen Prozeß ein isolierter A-Knoten in $\hat{A}_2$ geworden ist, a aus P und fügen a zu T hinzu. Weiterhin entfernen wir e aus T sowie f aus P. Ist a nicht in T, so kann es sein, daß durch a und c ein Zyklus in $\hat{G}$ läuft, der keine Kanten aus T enthält. In diesem Fall muß c ein A-Endknoten eines A-Pfades sein. Wie im vorherigen Abschnitt bei der Konstruktion von A_2 fügen wir dann noch $\{c, a\}$ zu $\hat{A}_2$ hinzu.

$\hat{G}$ hat nach Konstruktion einen Knoten weniger als G. Nach der Induktionsvoraussetzung ist $\hat{G}$ 2-pfadfärbbar. Die zugehörige Färbung kann zu einer Färbung von $(V, Z \cup P \cup T)$ ausgedehnt werden, indem man e die gleiche Farbe wie a und f die andere Farbe gibt. Dies kann zu keinem Zyklus führen, denn sonst gäbe es in $\hat{G}$ einen Zyklus durch a und c. Wie weiter oben beschrieben, kann die Färbung von $(V, Z \cup P \cup T)$ zu einer Färbung von $G = (V, Z \cup P \cup S)$ erweitert werden.

Fall 2: Falls der Endknoten von c und der Anfangsknoten von a identisch sind, so entsteht ein 2-Zyklus. Sei v dieser Knoten. Abhängig von der Richtung der dritten Kante h, die inzident mit v ist, fahren wir wie folgt fort:

Fall 2a: Falls v der Endknoten von h ist oder h gar nicht vorhanden ist, dann ist a eine isolierte Kante, siehe Abbildung 31. Wir entfernen a, wenden die Induktionsvoraussetzung an und geben a die Farbe, die c nicht hat. Danach kann kein Zyklus durch a gehen, da a und c verschiedene Farben erhalten haben und kein Weg aus z herausführt.

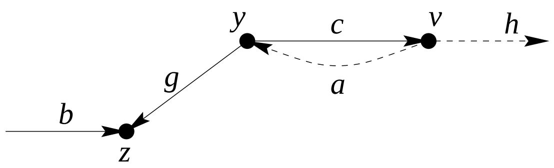

Abbildung 32: Wiederum entsteht ein 2-Zyklus. In diesem Fall werden a und c entfernt, und die Knoten y und v werden zu einem einzelnen Knoten kontrahiert.

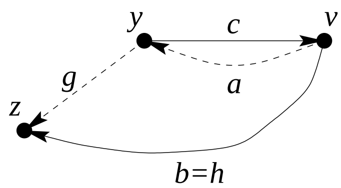

Abbildung 33: Diese Konfiguration liegt vor, wenn $b = h$. Sie kann wie dargestellt gefärbt werden. (Eine Farbklasse wir durch durchgezogene, die andere durch gestrichelte Linien dargestellt.)

Fall 2b: Der Fall, in dem v der Anfangsknoten von h ist, teilt sich in zwei Unterfälle, nämlich $b \neq h$ und $b = h$.

Im ersten Unterfall (siehe Abbildung 32) entfernen wir die Kanten c und a und kontrahieren y und v zu einem einzelnen Knoten. 2-Zyklen oder Doppelkanten können dabei nicht entstehen, da $b \neq h$. Wir fügen die Kante $\{g, h\}$ zu A_2 hinzu und wenden die Induktionsvoraussetzung an. Wir färben c mit der Farbe von h und a mit der von g. Offensichtlich kann kein Zyklus dabei entstehen.

Falls schließlich $b = h$, liegt wegen der Gradbeschränkungen eine isolierte Komponente von G mit vier Knoten vor, lediglich kann z von einer Kante verlassen werden, wenn der Zyklus in Abbildung 29 mehr als drei Kanten hat und f nicht mit z inzident ist. Die Komponente kann wie in Abbildung 33 gezeigt gefärbt werden. Der evtl. vorhandene Rest des Graphen kann nun rekursiv gefärbt werden, da egal ist, welche Farbe die Kante hat, die z verläßt. ■

Um die Laufzeit abzuschätzen, sei angemerkt, daß $|E|$ höchstens $\frac{3}{2}|V|$ sein kann. Die Gesamtlaufzeit wird dominiert von der Laufzeit des Algorithmus aus Lemma A.6. Da die Anzahl der Knoten in jedem Schritt um eins erniedrigt wird, wird die Induktionsvoraussetzung höchstens $|V|$ mal angewandt. In jedem Schritt wird für $|T| \leq |E|$ Kanten getestet, ob Sie auf einem Kreis liegen. Jeder einzelne Test kann in Zeit $O(|E|)$ ausgeführt werden. Folglich ist die Gesamtlaufzeit $O(n^3)$.

B Abschätzung der Approximationsgüten

B.1 Max-ATSP

Wir skizzieren, wie (11) gelöst werden kann. Als erstes ersetzen wir $x + y$ durch eine Variable u. Da die Koeffizienten von b alle positiv und die von c alle negativ sind, erzwingt $b \leq c$, daß $b = c$ im Minimum gilt. Also können $f_1, \ldots, f_4$ geschrieben werden als

$$\begin{aligned} f_1 &= \tfrac{3}{4} - \tfrac{1}{2}c + \tfrac{1}{4}u - \tfrac{3}{4}z, \\ f_2 &= \tfrac{5}{8} + \tfrac{1}{8}u - \tfrac{3}{8}z, \\ f_3 &= \tfrac{5}{8} + \tfrac{3}{8}u - \tfrac{1}{8}z + \tfrac{1}{2}r, \\ f_4 &= \tfrac{5}{8} + \tfrac{1}{8}u + \tfrac{5}{8}z - \tfrac{1}{2}r. \end{aligned}$$

Nur f_3 und f_4 hängen von r ab. Der Koeffizient von r in f_3 ist positiv und negativ in f_4. Schneidet man (die Unterräume gegebenen durch) f_3 mit f_4, so erhält man für festes u und z

$$r = \tfrac{1}{2}u + \tfrac{3}{4}z. \tag{16}$$

Dies ist zulässig, da $\frac{1}{2}u + \frac{3}{4}z \leq u$. Wir ersetzen daher f_3 und f_4 durch ihren Schnitt

$$f_3' = \tfrac{5}{8} - \tfrac{1}{8}u + \tfrac{1}{4}z.$$

Als nächstes untersuchen wir f_2 und f_3', die beide nicht von c abhängen. Die Menge der zulässigen Punkte ist gegeben durch $0 \leq z \leq \frac{1}{2}u$ und $u + z \leq 1$. Überprüft man die Punkte $(1, 0)$, $(0, 1)$, und $(\frac{1}{3}, \frac{2}{3})$ für (z, u), so sieht man, daß $\max\{f_2, f_3'\}$ mindestens $\frac{5}{8}$ in jedem dieser Punkte ist. (Es ist leicht zu sehen, daß das Minimum höchstens $\frac{5}{8}$ ist, in dem man $b = c = \frac{1}{2}$ und alle andere Variablen auf null setzt.) Schneiden von f_2 mit f_3' liefert

$$u = \tfrac{5}{2}z. \tag{17}$$

Dies ist wiederum zulässig, da $2z \leq \frac{5}{2}z$.

Wir substituieren (17) in f_1 und f_2 und erhalten

$$\begin{aligned} f_1' &= \tfrac{5}{8} - \tfrac{1}{16}z, \\ f_2' &= \tfrac{3}{4} - \tfrac{1}{2}c - \tfrac{1}{8}z. \end{aligned}$$

Die Menge der zulässigen Punkte ist nun gegeben durch $0 \leq u + z = \frac{7}{2}z \leq 1 - 2c$ und $0 \leq c \leq \frac{1}{2}$. In den Punkten $(\frac{2}{7}, 0)$, $(0, 0)$ und $(0, \frac{1}{2})$ ist $\max\{f_1', f_2'\}$ immer mindestens $\frac{5}{8}$. Wir schneiden f_1 mit f_2 und erhalten

$$z = 2 - 8c. \tag{18}$$

Das Minimum wird angenommen auf dem Schnitt von (18) mit $z = \frac{2}{7} - \frac{4}{7}c$, was auf dem Rand der Menge der zulässigen Lösungen liegt. Dies zeigt schließlich $c = \frac{3}{13}$. Setzt man dies in (18), (17) und (16) ein, so erhält man $z = \frac{2}{13}$, $u = \frac{5}{13}$ und $r = \frac{4}{13}$.

B.2 Max-3-DCC

Wir skizzieren, wie (12) gelöst werden kann. Das Maximum wird minimal, wenn c groß ist. Also kann $c = \frac{1}{2}(1 - b' + c')$ gesetzt werden. Damit bleibt zu lösen:

$$\text{Minimiere} \quad \max\{\tfrac{1}{2} + \tfrac{1}{2}b' - \tfrac{1}{2}c', \tfrac{2}{3} + \tfrac{1}{12}(c' - 2b')\}$$

über alle b' und c' mit:

$$\begin{aligned} 0 \leq c' &\leq b', \\ c' + b' &\leq 1. \end{aligned}$$

In den Eckpunkten $(1, 0)$, $(\frac{1}{2}, \frac{1}{2})$ und $(0, 0)$ ist der Wert der Zielfunktion jeweils größer als $\frac{3}{5}$. Also wird das Minimum auf der Schnittgeraden

$$\tfrac{1}{2} + \tfrac{1}{2}b' - \tfrac{1}{2}c' = \tfrac{2}{3} + \tfrac{1}{12}(c' - 2b') \iff b' = \tfrac{1}{4} + \tfrac{7}{8}c'$$

angenommen. Auf der Schnittgeraden beträgt der Wert der Zielfunktion $\frac{5}{8} - \frac{1}{16}c'$. Der maximale Wert, den c' auf der Schnittgeraden annehmen kann, ist $\frac{2}{5}$.

Literatur

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[2] Jochen Alber, Henning Fernau, and Rolf Niedermeier. Parametrized complexity: Exponential speed-up for planar graph problems. In *Proc. 28th Int. EATCS Coll. on Automata, Languages and Programming (ICALP)*, number 2076 in Lecture Notes in Comput. Sci., pages 261–272, 2001.

[3] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[4] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer, 1999.

[5] R. Bar-Yehuda. Using homogeneous weights for approximating the weighted vertex cover problem. In *Proc. 10th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, pages 71–75, 1999.

[6] A. I. Barvinok, E. Kh. Gimadi, and A. I. Serdyukov. The maximum traveling salesman problem. In G. Gutin and A. P. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization.* Kluwer, 2002.

[7] A. I. Barvinok, D. S. Johnson, G. J. Woeginger, and R. Woodroofe. Finding maximum length tours under polyhedral norms. In *Proc. 6th Int. Integer Programming and Combinatorial Optimization Conf. (IPCO)*, number 1412 in Lecture Notes in Comput. Sci. Springer, 1998.

[8] Markus Bläser. An $\frac{8}{13}$-approximation algorithm for the asymmetric maximum TSP. In *Proc. 13th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, pages 64–73, 2002.

[9] Markus Bläser. Computing small partial coverings. *Inform. Proc. Letters*, to appear.

[10] Markus Bläser. A new approximation algorithm for asymmetric TSP with triangle inequality. In *Proc. 14th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, to appear in January 2003.

[11] Markus Bläser and Bodo Manthey. Two approximation algorithms for 3-cycle covers. In *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, volume 2462 of *Lecture Notes in Comput. Sci.*, pages 40–50. Springer, 2002.

[12] Markus Bläser and Bodo Siebert. Computing cycle covers without short cycles. In *Proc. 9th Ann. European Symp. on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Comput. Sci.*, pages 368–379. Springer, 2001.

[13] Avrim Blum, Tao Jiang, Ming Li, John Tromp, and Mihalis Yannakakis. Linear approximation of shortest superstrings. *J. ACM*, 41(4):630–647, 1994.

[14] J. Böckenhauer, J. Hromkovič, R. Klasing, S. Seibert, and W. Unger. An improved lower bound on the approximability of metric TSP and approximation algorithms for the TSP with sharpened triangle inequality. In *Proc. 17th Int. Symp. on Theoret. Aspects of Comput. Sci. (STACS)*, volume 1770 of *Lecture Notes in Comput. Sci.*, pages 382–394. Springer, 2000.

[15] F. T. Boesch and J. F. Gimpel. Covering the points of a digraph with point-disjoint paths and its application to code optimization. *J. ACM*, 24:192–198, 1977.

[16] Dany Breslauer, Tao Jiang, and Zhigen Jiang. Rotations of periodic strings and short superstrings. *J. Algorithms*, 24:340–353, 1997.

[17] N. Bshouty and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *Proc. 15th Int. Symp. on Theoret. Aspects of Comput. Sci. (STACS)*, volume 1373 of *Lecture Notes in Comput. Sci.*, pages 298–308, 1998.

[18] Robert Carr and Santosh Vempala. Towards a 4/3 approximation for the asymmetric traveling salesman problem. In *Proc. 11th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, pages 116–125, 2000.

[19] L. Sunil Chandran and L. Shankar Ram. Approximations for ATSP with parametrized triangle inequality. In *Proc. 19th Int. Symp. on Theoret. Aspects of Comput. Sci. (STACS)*, volume 2285 of *Lecture Notes in Comput. Sci.*, pages 227–237, 2002.

[20] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: further observations and further improvements. In *Proc. 25th Int. Workshop on Graph Theoretic Concepts in Computer Science (WG)*, volume 1665 of *Lecture Notes in Comput. Sci.*, pages 313–324, 1999.

[21] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, page 441. Academic Press, 1976.

[22] Thomas H. Cormen, Charles E. Leierson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[23] G. Cornuéjols and W. Pulleyblank. A matching problem with side constraints. *Discrete Math.*, 29:135–159, 1980.

[24] Béla Csaba, Marek Karpinski, and Piotr Krysta. Approximability of dense and sparse instances of minimum 2-connectivity, TSP and path problems. In *Proc. 13th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, pages 74–83, 2002.

[25] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Springer, 1998.

[26] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[27] Lars Engebretsen and Marek Karpinski. Approximation hardness of TSP with bounded metrics. In *Proc. 28th Int. EATCS Coll. on Automata, Languages and Programming (ICALP)*, volume 2076 of *Lecture Notes in Comput. Sci.*, pages 201–212. Springer, 2001.

[28] Uri Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45:634–652, 1998.

[29] M. L. Fisher, L. Nemhauser, and L. A. Wolsey. An analysis of approximations for finding a maximum weight Hamiltonian circuit. *Networks*, 12(1):799–809, 1979.

[30] A. M. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[31] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. In *Proc. 28th Int. EATCS Coll. on Automata, Languages and Programming (ICALP)*, number 2076 in Lecture Notes in Comput. Sci., pages 225–236, 2001.

[32] Michael R. Garey and David S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[33] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *Siam J. Comput*, 24(2):296–317, 1995.

[34] R. L. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of Combinatorics*, volume II. Elvsevier, 1995.

[35] R. L. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of Combinatorics*, volume I. Elvsevier, 1995.

[36] D. B. Hartvigsen. *Extensions of Matching Theory*. PhD thesis, Carnegie-Mellon University, 1984.

[37] D. B. Hartvigsen. The square-free 2-factor problem in bipartite graphs. In *Proc. 7th Int. Integer Programming and Combinatorial Optimization Conf. (IPCO)*, Lecture Notes in Comput. Sci., pages 234–241, 1999.

[38] D. B. Hartvigsen. Personal communication. 2002.

[39] Refael Hassin and Shlomi Rubinstein. Better approximations for maxTSP. *Inform. Proc. Letters*, 75(4):251–258, 2000.

[40] Refael Hassin and Shlomi Rubinstein. A $\frac{7}{8}$-approximation algorithm for metric MaxTSP. In *Proc. 7th Int. Workshop on Algorithms and Data Structures (WADS)*, number 2125 in Lecture Notes in Comput. Sci., pages 205–209, 2001.

[41] Johan Håstad. Some optimal inapproximability results. In *Proc. 29th Ann. ACM. Symp. on Theory of Comput. (STOC)*, pages 1–10, 1997.

[42] Michael Held and Richard M. Karp. The travelling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[43] Dorit S. Hochbaum. The t-vertex cover problem: extending the half integrality framework with budget constraints. In *Proc. 1th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, volume 1444 of *Lecture Notes in Comput. Sci.*, pages 111–122, 1998.

[44] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci*, 9:256–278, 1974.

[45] David S. Johnson, Gregory Gutin, Lyle A. McGeoch, Anders Yeo, Weixiong Zhang, and Alexei Zverovitch. Experimental analysis of heuristics for the ATSP. In G. Gutin and A. P. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*. Kluwer, 2002.

[46] Iyad Kanj. *Vertex Cover: Exact and Approximate Algorithms and Applications.* PhD thesis, Texas A&M University, USA, 2001.

[47] S. Rao Kosaraju, James K. Park, and Clifford Stein. Long tours and short superstrings. In *Proc. 35th Ann. IEEE Symp. on Foundations of Comput. Sci. (FOCS)*, 1994.

[48] A. V. Kostochka and A. I. Serdyukov. Polynomial algorithms with the estimates $\frac{3}{4}$ and $\frac{5}{6}$ for the traveling salesman problem of the maximum. *Upravlyaemye Sistemy*, 26:55–59, 1985. (in Russian).

[49] M. W. Krentel. The complexity of optimization problems. *J. Comput. Syst. Sci*, 36:490–509, 1988.

[50] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem.* Wiley, 1985.

[51] Moshe Lewenstein and Maxim Sviridenko. A 5/8 approximation algorithm for the maximum asymmetric TSP. Manuscript, 2002.

[52] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[53] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Operations Research*, 18:1–11, 1993.

[54] S. Pinter and Y. Wolfstahl. On mapping processes to processors. *Int. J. Parallel Programming*, 16:1–15, 1987.

[55] K. Rüdiger Reischuk. *Komplexitätstheorie, Band I: Grundlagen.* Teubner, 1999.

[56] S. K. Sahni and T. F. Gonzalez. P-complete approximation problems. *J. ACM*, 22:555–565, 1976.

[57] A. I. Serdyukov. An algorithm with an estimate for the traveling salesman problem of the maximum. *Upravlyaemye Sistemy*, 25:80–86, 1984. (in Russian).

[58] João Carlos Setubal and João Meidanis. *Introduction to Computational Molecular Biology.* PWS Publishing Company, 1997.

[59] Petr Slavík. Improved performance of the greedy algorithm for partial cover. *Inform. Proc. Letters*, 64(5):251–254, 1997.

[60] Z. Sweedyk. A $2\frac{1}{2}$–approximation algorithms for shortest superstring. *Siam J. Comput*, 29(3):954–986, 1999.

[61] J. Tarhio and E. Ukkonen. A greedy approximation algorithm for constructing shortest common superstrings. *Theoret. Comput. Sci*, 57(2):131–145, 1988.

[62] J. S. Turner. Approximating algorithms for the shortest common superstring problem. *Information and Computation*, 83(1):1–20, 1989.

[63] W. T. Tutte. A short proof of the factor theorem for finite graphs. *Canad. J. Math.*, 6:347–352, 1954.

[64] Leslie G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci*, 8(2):189–201, 1979.

[65] Vijay V. Vazirani. *Approximation Algorithms.* Springer, 2001.

[66] Sundar Vishwanathan. An approximation algorithm for the asymmetric travelling salesman problem with distances one and two. *Inform. Proc. Letters*, 44:297–302, 1992.

[67] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz.*, 3:25–30, 1964.

[68] David P. Williamson. Lecture notes on approximation algorithms. Technical Report RC 21409, IBM T. J. Watson Research Center, 1999.