Technische Universität Berlin
Fachbereich Mathematik
Straße des 17. Juni 136
10623 Berlin

Winkelminimierung bei Überdeckungsproblemen in Graphen

Diplomarbeit bei PD Dr. Marco E. Lübbecke

vorgelegt von Olaf Maurer

30. November 2009

Die selbstständige und eigenhändige Anfertigung dieser Arbeit versichere ich an Eides statt.

30. November 2009

Olaf Maurer

Inhaltsverzeichnis

Einleitung

Diese Arbeit befasst sich mit einem diskreten Optimierungsproblem, nämlich mit der Suche nach einer kostenminimalen Überdeckung eines ebenen Graphen. Entstehen die Kosten der Überdeckung als Summe von Gewichten von Kanten, so sind diese Fragestellungen schon gut untersucht. Das neue hier besteht darin, das die Kosten nicht als Summe von Kantengewichten entstehen, sondern mit Winkeln zwischen Kanten zusammenhängen. Es wird also angenommen, dass es gerade Verbindungen zwischen den Knoten in der Ebene gibt und man fragt sich nun, wie man die Winkel zwischen ihnen beziehungsweise die Anzahl der Knicke unter einer vorgegebenen Fragestellung minimieren kann.

Eine klassische Fragestellung ist das Problem des Handlungsreisenden, dies ist eines der wenigen Probleme, die unter dem Gesichtspunkt der Winkelkosten schon als schwer bekannt sind. Ein anderes als schwer bekanntes Problem ist die Suche nach nicht nur einer Tour wie beim Problem des Handlungsreisenden, sondern nach einer beliebigen Anzahl von Touren, deren Gesamtgewicht minimiert werden soll. In einigen ebenen Problemen ist also die Schwere des Problems bekannt.

Dieses Problem findet in der Richtung Anwendung, wenn man sich Roboter dabei vorstellt, wie sie sehr schnell polygonale Strecken abfahren. Bei einem Richtungswechsel müssen sie jedoch als Preis dafür einen langwierigen Brems- und Beschleunigungsvorgang in Kauf nehmen. In dieser Problemumgebung werden die Kosten durch die Bremsvorgänge dominiert. Bei Problemen mit Winkelkosten könnte man sich auch vorstellen, dass zum Beispiel Drehvorgänge im Vergleich zu anderen Zeitverzögerungen den Gesamtzeitverbrauch bestimmen. Eine weitere, nicht so offensichtliche Anwendung tritt auf, wenn man Lichtwellenleiter betrachtet. So ist die Lichtgeschwindigkeit sehr hoch - Verzögerungen bei der Datenübertragung entstehen vor allem dann, wenn ein Signal von einer Frequenz auf eine andere umgewandelt werden muss.

Eine interessante Fragestellung ergibt sich, wenn die zugrundeliegenden Graphen Gitter sind und man sich nur innerhalb dieser Gitter bewegen darf. Für solche Probleme ist für Kreisüberdeckungen wenig bekannt, es gibt lediglich einen Approximationsalgorithmus konstanter Güte. Insbesondere kennt man viele offene Fragen im Zusammenhang mit der Komplexität dieser Probleme. Auf Probleme im Gitter wird in der Arbeit ein besonderes Augenmerk gelegt.

Ein Mittel, das bisher bei der Analyse dieser Probleme wenig zum Einsatz kam, ist das Hilfsmittel der Programmierungstechniken. So wird man sehen, dass sich die klassifizierten Probleme allesamt zumindest approximativ als ganzzahlige Programme darstellen lassen und auf diese Weise durchaus Optimallösungen gefunden werden können oder bei größeren Instanzen wenigstens Lösungen, deren Zielfunktionswert im Vergleich mit unteren Schranken eine Gütegarantie liefert.

Die Arbeit ist in drei Teile gegliedert. Im ersten Teil werden die Probleme mit Abbiegekosten einer Klassifikation unterworfen, in die sich die meisten der damit zusammenhängenden Resultate einordnen lassen. Im zweiten Teil werden eigene theoretische Untersuchungen angestellt, die

teilweise zu Ergebnissen führen. Schließlich werden im letzten Teil einige Heuristiken vorgestellt und Bilder von Lösungen gezeigt.

Danksagung

An dieser Stelle möchte ich mich bei allen jenen lieben Menschen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben. Ich möchte mich bei Dr. Marco Lübbecke bedanken, der mir das Thema vorschlug und mir die Möglichkeit gab, darin eine Diplomarbeit zu verfassen. Es war nicht immer leicht, an einem offenen Problem zu arbeiten und nicht genau zu wissen, ob der gerade eingeschlagene Weg zu einem Ziel führt. Auch mussten auf dem Weg viele Rückschläge verkraftet werden, wenn schon gesichert geglaubte Erkenntnisse sich doch als unwahr herausstellten.

Weiterhin möchte ich mich bei meinen Eltern für die Unterstützung während dieser Zeit bedanken und noch einmal besonders vielen Dank für das intensive Korrekturlesen kurz vor der Abgabe.

Weiterer Dank gebührt Torsten Gellert für den vielen Austausch über Probleme mathematischer sowie nichtmathematischer Art. Auch das Korrekturlesen am Ende war sehr hilfreich.

Schließlich möchte ich mich noch bei Sebastian Stiller für ein stets offenes Ohr und hilfreiche Diskussionen bedanken.

Ein letzter Dank geht an Matthias Walter für die bereitwillige Hilfe bei der Untersuchung einer Matrix auf vollständige Unimodularität.

1 Grundlagen

1.1 Definitionen

In diesem Abschnitt werden die benutzten Definitionen aus der Graphentheorie fixiert. Diese sind grundlegend für die ganze Arbeit.

Ein Paar $G = (V, E)$ von endlichen Mengen V, E mit $E \subseteq V^2$ und $E \cap \{(v_i, v_i) \mid v_i \in V\} = \emptyset$ heißt ein einfacher **Digraph** (auf V). Wenn für G die Eigenschaft gilt, dass aus $(u, v) \in E$ stets $(v, u) \in E$ folgt, so heißt G ein **ungerichteter, einfacher Digraph** oder einfach ein **Graph**. Alle vorkommenden Graphen werden stets als ungerichtet und einfach angenommen, wenn nicht ausdrücklich etwas anderes spezifiziert wird.

Die Elemente von V werden **Knoten**, die von E **Kanten** genannt. Für die Kante $e = (u, v)$ heißt u der **Anfangsknoten** und v der **Endknoten**. Man setzt $n := |V|$ und $m := |E|$. Kanten und Knoten werden mit Kleinbuchstaben bezeichnet, während Großbuchstaben Mengen bedeuten sollen.

$G' = (V', E')$ heißt ein **Subgraph** von $G = (V, E)$, falls G' ein Graph ist, $V' \subseteq V$, $E' \subseteq E$. Man schreibt auch kurz: $G' \subseteq G$ und meint, dass G' ein Subgraph von G ist. Der Subgraph G' heißt **induziert**, wenn $E' = E \cap (V')^2$, in Worten G' also alle Kanten von G enthält, von welchen beide Endpunkte in V' existieren.

Ein Knoten v heißt mit einer Kante $e = (w_1, w_2)$ **inzident**, falls $v = w_1$ oder $v = w_2$. Die beiden Knoten, aus denen eine Kante besteht, heißen deren **Endpunkte**. Zwei Knoten $u, v \in V$ heißen **adjazent**, falls $(u, v) \in E$ oder $(v, u) \in E$. Die Menge der zu $v \in V$ adjazenten Knoten heißt die **Nachbarschaft** von v. Die Anzahl der zu einem Knoten $v \in V$ adjazenten Kanten heißt der **Grad** von v. Die Menge der zu einem Knoten adjazenten Kanten wird mit $\delta(v)$ bezeichnet.

Ein Weg in G ist ein Tupel $(v_0, v_1, \ldots, v_k)$ mit $(v_j, v_{j+1}) \in E \forall j \in \{0, \ldots, k-1\}$. Der Weg **beginnt** bei v_0 und **endet** bei v_k. Ein Weg lässt sich auch als Abfolge seiner Kanten darstellen: $(e_0, \ldots, e_{k-1})$, wobei der Endknoten einer Kante stets mit dem Anfangsknoten der darauffolgenden Kante übereinstimmen muss. Weiterhin kann ein Weg durch eine Knoten-Kanten-Abfolge $(v_0, e_0, v_1, e_1, \ldots, e_{k-1}, v_k)$ repräsentiert werden. Alle diese Darstellungsmöglichkeiten werden je nach Praktikabilität benutzt, ohne darauf im Einzelnen hinzuweisen. Ein Weg heißt **elementar**, wenn je zwei seiner Knoten verschieden sind.

Ein **Kreis** in G ist ein Weg, dessen Anfangspunkt gleich seinem Endpunkt ist. Sei der Kreis C durch $(v_0, \ldots, v_k)$ beschrieben. Dann heißt C **elementar**, wenn $v_0, \ldots, v_{k-1}$ ein elementarer Weg ist. Die **Länge** eines Weges ist die Anzahl seiner Kanten. Ein Weg heißt **gerade**, wenn seine Länge gerade ist, sonst **ungerade**.

Ein Graph $G = (V, E)$ heißt **zusammenhängend**, falls es für jedes Paar von Knoten $(u, v) \in E^2$ einen Weg gibt, der bei u beginnt und bei v endet. Er heißt **bipartit**, falls $V = V_1 \uplus V_2$, $E \cap (V_1)^2 =$

$\emptyset = E \cap (V_2)^2$. V_1 und V_2 heißen **Partitionen**. Der Graph ist also bipartit, falls alle Kanten ihre Endpunkte in verschiedenen Partitionen haben.

Ein Graph, der keine Kreise der Länge 3 oder größer enthält, heißt **kreisfrei**. Ein Graph $G = (V, E)$ heißt ein **Baum**, wenn er zusammenhängend und kreisfrei ist. Ein Subgraph G' heißt **Baum**, wenn er, als eigenständiger Graph betrachtet, ein Baum ist. Ist seine Knotenmenge gleich der Knotenmenge von G, heißt er **(auf-)spannend**.

Der **Komplementärgraph** $\overline{G}$ zum Graphen G hat die gleiche Knotenmenge wie G. In $\overline{G}$ sind zwei Knoten henau dann durch eine Kante verbunden, wenn sie in G nicht durch eine Kante verbunden sind.

Der **Linegraph** oder **Liniengraph** zum Graphen G hat als Knotenmenge die Kantenmenge von G. Im Liniengraphen sind zwei Knoten durch eine Kante verbunden, wenn die entsprechenden Kanten in G inzident sind. Eine Menge $M \subset V$ eines Graphen $G = (V, E)$ heißt **stabil** oder **unabhängig**, falls der von M induzierte Subgraph keine Kanten enthält.

Als K_n wird der Graph auf n Knoten bezeichnet, in dem je zwei Knoten durch eine Kante verbunden sind.

Ein **Vertex Cover** in einem Graphen G ist eine Teilmenge V' der Knotenmenge mit der Eigenschaft, dass jede Kante zu mindestens einem Knoten aus V' adjazent ist. Ein **Matching** ist eine Teilmenge E' der Kantenmenge mit der Eigenschaft, dass jeder Knoten in V zu höchstens einer der Kanten aus E' adjazent ist. Ein 2-Matching ist ein Inzidenzvektor x, so dass für jeden Knoten v $\sum_{p \in \delta v} x_p = 2$ gilt.

Diese Arbeit beschäftigt sich mit Überdeckungsproblemen geometrischer Natur. Daher werden viele Graphen mit geometrischen Objekten in Verbindung stehen. Zu diesem Zweck werden folgende Definitionen eingeführt:

Ein Graph $G = (V, E)$ heißt **straight-line** oder **SLG**, wenn $v \in \mathbb{R}^2$ für alle $v \in V$ und wenn die Kanten lineare Verbindungsstücke der Knoten sind.

Da es nur dann Sinn ergibt, von Winkeln zwischen Kanten zu sprechen, wenn der geometrische Ort der Kanten und Knoten feststeht, werden die zu überdeckenden Knoten stets als Elemente des $\mathbb{R}^2$ angenommen (bzw. als Elemente des $\mathbb{Q}^2$ aus Kodierbarkeitsgründen).

Die euklidische Norm im $\mathbb{R}^2$ wird mit $|\cdot|$ bezeichnet, ohne erneut darauf hinzuweisen.

Durch zwei zum selben Knoten inzidente Kanten wird ein **Abbiegewinkel** definiert. Dieser ist der Winkel zwischen der Verlängerung einer der Kanten und der anderen Kante. In Abbildung 1.1 ist der Abbiegewinkel zwischen der Kante L_1 und L_2 also der Winkel α. Wie man direkt sieht, ist der Abbiegewinkel symmetrisch, hängt also nicht davon ab, ob man von L_1 auf L_2 oder umgekehrt knickt.

1.2 Gitter

Gitter spielen eine wichtige Rolle in vielen Teilen der Mathematik, zum Beispiel in der Codierungstheorie, Kryptographie und Algebra. Für unsere Betrachtungen interessiert speziell das ganzzahlige Gitter im $\mathbb{R}^2$.

Ein **Unit-Distance-Graph** in der euklidischen Ebene ist eine endliche Menge von Punkten $V \subset \mathbb{R}^2$ mit der Kantenmenge

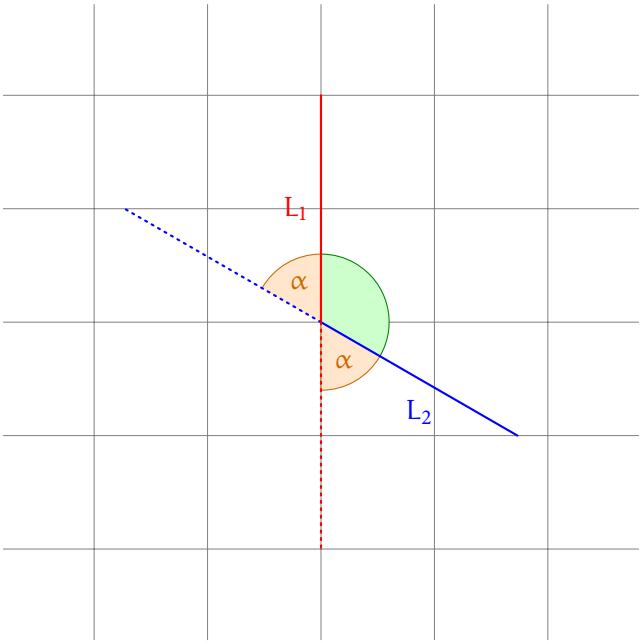

Abbildung 1.1: Abbiegewinkel

$$E = \left\{(v_1, v_2) \in V^2 \mid |v_1 - v_2| = 1\right\}. \tag{1.1}$$

Haben dabei alle Punkte aus V ganzzahlige Koordinaten, so heißt G ein **Gitter**.

Lemma 1.1. *Gitter sind bipartite Graphen.*

Beweis. Für ein Gitter $G = (V, E)$ definiere $V_1 := \{(x, y) \in V \mid (x + y) \text{ gerade}\}$ und $V_2 := V \setminus V_1$. Dann verlaufen alle Kanten zwischen V_1 und V_2. □

Mit

Lemma 1.2. *[Die05] Ein Graph ist bipartit genau dann, wenn er keine ungeraden Kreise enthält.*

erhält man als Korollar:

Korollar 1.3. *In Gittern sind alle Kreise gerade.*

1.3 Komplexität

In diesem Abschnitt wird eine knappe Einführung in die benutzten Definitionen aus der Komplexitätstheorie gegeben. Dabei soll der Begriff des Algorithmus genügen, eine weitere Präzisierung durch den Begriff der Turing-Maschine ist für diese Arbeit nicht erforderlich.

Im Rahmen der Komplexitätstheorie wird versucht, auf Computern lösbare Probleme anhand ihrer Komplexität zu klassifizieren, d.h., welche Anzahl von Berechnungsschritten (bzw. welche

„Größenordnung“) ein Algorithmus mindestens durchführt, der alle Instanzen eines Problems zu lösen vermag. Dabei wird eine Vielzahl von Komplexitätsklassen definiert und es werden ihre Beziehungen zueinander untersucht (vgl. z. B. [Pap95]).

Die Anzahl der durchgeführten Schritte eines Algorithmus für die Lösung eines Problems wird als seine **Laufzeit** bezeichnet. Die **Eingabegröße** oder **Kodierungslänge** einer Instanz ist die Länge eines Strings, der die Instanz in einer vorgegebenen Kodierung beschreibt (das dabei benutzte Kodierungsverfahren gehört zur Spezifikation des Problems). Hier wird stets binäre Codierung angenommen und für Knoten und Kanten eine Kodierungslänge von 1.

Eine der dabei wichtigsten Unterscheidungen ist die Unterscheidung zwischen den Klassen $\mathcal{P}$ (polynomial) und $\mathcal{NP}$ (für: nichtdeterministisch polynomial). Während die Probleme aus der ersten Klasse in Laufzeiten polynomial in der Eingabegröße der Instanz gelöst werden können, ist das für die Probleme aus letzterer Klasse im Allgemeinen unklar. Diese Frage gehört zu den wichtigsten offenen Fragen der Mathematik (vgl. [Coo03]).

Zu den Problemen der ersten Klasse gehören beispielsweise die Sortierung von Zahlen oder die Suche eines kürzesten Weges in einem Graphen mit nichtnegativen Kantengewichten. Die Suche nach einer kürzesten Rundreise gehört beispielsweise zur zweiten Klasse.

Die folgenden Definitionen werden mit Hilfe des Begriffs des **Algorithmus** gegeben, also einer endlichen Folge von Operationen, die durch ein **Turing-vollständiges Berechnungssystem** ausgeführt werden können. Die genaue Laufzeit eines Algorithmus hängt von dem konkret betrachteten Rechnermodell ab. Wir werden für alle arithmetischen Operationen ein **Unit-Cost-Model** annehmen. Dieses kann in Polynomialzeit durch Turing-Maschinen simuliert werden, sofern die Kodierungslänge der berechneten Zahlen polynomial in der Eingabelänge bleibt. Dies wird hier aber stets der Fall sein.

In der Formalisierung der erwähnten Fragen werden sogenannte **Entscheidungsprobleme** betrachtet, die als Antwort ein „Ja“ oder ein „Nein“ erfordern. Eine **Sprache** L ist eine (nicht notwendig echte) Teilmenge der Menge aller endlichen Zeichenketten, die nur aus 0 und 1 bestehen. Die Menge aller dieser Zeichenketten wird mit $\{0,1\}^*$ bezeichnet. Eine Sprache heißt **polynomial entscheidbar**, wenn es einen Algorithmus gibt, der bei jeder Eingabe nach endlich vielen Schritten stoppt und für $x \in L$ eine 1, für $x \notin L$ eine 0 zurückgibt.

Ein **Entscheidungsproblem** ist ein Paar $D = (X, Y)$, wobei $X \subseteq \{0,1\}^*$ eine polynomial entscheidbare Sprache und $Y \subseteq X$ ist. Die Elemente von Y heißen **Ja-Instanzen**, die von $X \setminus Y$ heißen **Nein-Instanzen**. Ein Algorithmus für ein Entscheidungsproblem berechnet die Funktion $f : X \to \{0,1\}$ mit $f(x) = 1$ für $x \in Y$ und $f(x) = 0$ für $x \in X \setminus Y$.

Die Menge aller Entscheidungsprobleme, für die ein in der Laufzeit in der Kodierungslänge der Eingabe polynomial beschränkter Algorithmus existiert, wird mit $\mathcal{P}$ bezeichnet. Die Menge aller Entscheidungsprobleme L, für die eine polynomial entscheidbare Relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ existiert, so dass $L = \{x \mid (x,y) \in R \text{ für ein } y\}$, so dass für $(x,y) \in R$ stets $|y| \leqslant |x|^k$ für ein $k \geqslant 1$ gilt, wird mit $\mathcal{NP}$ bezeichnet (dies sind also die „leichten“ Probleme). Man kann also sagen, dass die Probleme in $\mathcal{NP}$ dadurch charakterisiert sind, dass es für sie ein polynomial beschränktes „Zertifikat“ gibt, das die Zugehörigkeit beweist. Über die Laufzeit zum Finden dieses Zertifikats wird keine Annahme gemacht.

Die Probleme in $\mathcal{P}$ sind in Polynomialzeit lösbar und bei den Problemen in $\mathcal{NP}$ gibt es ein in Polynomialzeit überprüfbares Zertifikat der Richtigkeit der Antwort. Es folgt sofort $\mathcal{P} \subseteq \mathrm{NP}$.

Um die Polynomialität der Laufzeit eines Algorithmus leichter entscheiden zu können, wird

der Begriff der **Reduzierbarkeit** eingeführt. Die Idee dabei ist, dass man eine Instanz eines Problems auf eine Instanz eines anderen Problems überführt, das man schon in Polynomialzeit lösen kann und durch die Rücktransformation eine polynomialzeitliche Lösung des ursprünglichen Problems erhält.

Ein Entscheidungsproblem P_1 heißt auf das Entscheidungsproblem P_2 **polynomial reduzierbar**, wenn es eine Abbildung f gibt, die jede Instanz von P_1 auf eine Instanz von P_2 abbildet, so dass die Instanz von P_2 in Zeit polynomial in der Kodierungslänge der Instanz von P_1 konstruiert werden kann und die Instanz von P_1 genau dann eine Ja-Instanz ist, wenn $f(P_1)$ eine Ja-Instanz ist.

Ein Entscheidungsproblem $D \in \mathcal{NP}$ heißt $\mathcal{NP}$-**vollständig**, wenn jedes Problem in $\mathcal{NP}$ polynomial auf D reduziert werden kann.

A priori ist die Existenz von solchen Problemen nicht klar. Es gilt aber:

Theorem 1.4. *[Coo71] SATISIFIABILITY ist $\mathcal{NP}$-vollständig.*

Problem 1 (Satisfiability Problem)**.**

Instanz:	Menge $\{x_1, \ldots, x_n\}$ von Variablen endliche Menge von Klauseln, diese bestehend aus endlich vielen Literalen (x_i oder $\overline{x_i}$ („nicht x_i“))
Gesucht:	Belegung der Variablen mit booleschen Wahrheitswerten, so dass in jeder Klausel mindestens ein Literal erfüllt wird

Ein Zertifikat für SATISFIABILITY wäre beispielsweise eine Zuordnung von Variablen zu Wahrheitswerten. Die Erfüllung der Instanz durch die durch das Zertifikat gegebene Belegung kann dann leicht in Polynomialzeit überprüft werden.

1.4 Überdeckungsprobleme

In der Arbeit werden vor allem Überdeckungsprobleme betrachtet. Diese werden nun definiert. Anwendung in der Praxis finden diese vor allem als Teilprobleme bei Verfahren zur Lösung anderer Probleme wie z.B. bei der Suche nach kürzesten Rundwegen.

Eine **Kreisüberdeckung** oder **Cycle-Cover** eines Graphen $G = (V, E)$ ist eine Familie von Kreisen $(C_1, \ldots, C_k)_{k \in \mathbb{N}}$ mit der Eigenschaft, dass jeder Knoten $v \in V$ in einem der Kreise enthalten ist. Eine **Kreispartitionierung** oder **Cycle-Partition** ist eine Familie von (notwendigerweise elementaren) Kreisen $(C_1, \ldots, C_k)_{k \in \mathbb{N}}$, so dass jeder Knoten $v \in V$ in genau einem der Kreise genau einmal enthalten ist. Um diese Probleme soll es in der Arbeit vor allem gehen. Weitere bekannte Überdeckungs- bzw. Partitionierungsprobleme umfassen:

- das **Hamiltonkreisproblem**: Suche einen elementaren Kreis mit n Kanten in einem Graphen G mit n Knoten. Existiert ein solcher Kreis in G, heißt G **hamiltonsch**.
- das **Hamiltonpfadproblem**: Suche einen elementaren Pfad der Länge $n - 1$.

- das **Eulertourproblem**: Suche einen Kreis, der jede Kante genau einmal benutzt.
- das **Chinese-Postman-Problem**: Suche einen kürzesten Kreis, der jede Kante mindestens einmal benutzt.

Während die ersten beiden Probleme zur Kategorie der $\mathcal{NP}$-vollständigen Probleme gehören, sind die letzten beiden Probleme in Polynomialzeit lösbar, wenn die Kosten der gesuchten Objekte wie üblich als Summe der Kosten der benutzten Kanten entstehen.

Ein viel beachtetes Überdeckungsproblem ist die Suche nach einem kürzesten Hamiltonkreis, auch als das Traveling-Salesman-Problem bekannt:

Problem 2 (Traveling-Salesman-Problem)**.**

Instanz: Graph $G = (\{v_0, \ldots, v_{n-1}\}, E)$, Kosten $c : E \to \mathbb{Q}$

Gesucht: Permutation $\pi \in \Sigma_n$, so dass
$e_i := (v_{\pi(i)}, v_{\pi((i+1) \bmod n)}) \in E$ für alle $i \in \{0, \ldots, n-1\}$

Zielfunktion: $\min f(x) = \sum_{i=0}^{n-1} c(e_i)$

Anwendungen für das TSP finden sich beispielsweise in der Reihenfolgeplanung beim Bohren von Leiterplatten.

1.5 Klassifizierung

Diese Arbeit beschäftigt sich mit Überdeckungsproblemen, die die Besonderheit aufweisen, dass die Kosten der Lösungen mit Winkeln zwischen den benutzten Kanten zusammenhängen. Einige dieser Probleme wurden bereits (unabhängig voneinander) untersucht. Es wird hier eine Klassifizierung und Einordnung der bereits bestehenden Resultate vorgestellt.

Es wird hierzu von einer endlichen Menge von Punkten $V \subset \mathbb{R}^2$ ausgegangen. Bei einigen der betrachteten Probleme dürfen Knoten nicht direkt nacheinander überdeckt werden. Diese Eigenschaft lässt sich sich durch einen Graphen $\overline{G} = (V, \overline{E})$ beschreiben: zwei Knoten sind mit einer Kante verbunden, wenn sie nicht direkt nacheinander überdeckt werden dürfen. Dieser Graph wird als **Konfliktgraph** bezeichnet. Er ist das Komplement des **Kompatibilitätsgraphen** G: zwei Knoten sind durch eine Kante verbunden, wenn sie konsekutiv überdeckt werden dürfen.

Die untersuchten Probleme unterscheiden sich dabei hinsichtlich der gemachten Einschränkungen, der Art der gesuchten Objekte sowie der Zielfunktion. Gemeinsam ist ihnen, dass das Ziel stets heißt, die Menge V auf die eine oder andere Art und Weise zu überdecken.

Zunächst wird bei den Problemen nach der Struktur des gesuchten Objekts unterschieden. Üblich sind Familien von Kreisen (**K**) oder auch ein Spannpfad (**S**), der ohne weitere Einschränkungen zunächst Kanten und Knoten doppelt besuchen darf. Weiterhin besteht die Möglichkeit, einen Spannkreis (**H**) zu suchen, also einen Kreis, der alle Knoten mindestens einmal besucht .

Im vorhandenen Kontext hat die Zielfunktion meist mit Knicken zu tun, sei es der Winkel eines Knicks oder nur die Anzahl. Auch hier ergeben sich verschiedene Fälle in Abhängigkeit

davon, wo diese Knicke geschehen dürfen. Es wurden die Problemvarianten untersucht, bei denen Knicke nur in V sein dürfen (**N**), Knicke an beliebigen Knickorten auch außerhalb der Knoten (**B**) oder auch an beliebigen Knickorten mit ausschließlich achsenparallelen Kanten (**A**). Letztere Problemvariante führt dazu, dass man in den meisten Fällen ohne Einschränkung annehmen kann, dass die Knickorte aus $X \times Y$ stammen, wenn X die Menge der x-Koordinaten der Knoten und Y die Menge der $y-$Koordinaten der Knoten bezeichnet.

Eine weitere Unterscheidung betrifft die Art der Überdeckung. Es wird danach unterschieden, ob jeder Knoten genau einmal überdeckt werden muss (**P**) oder jeder Knoten mindestens einmal überdeckt werden muss (**C**). Bei P kann es in einigen Problemen trotzdem auftreten, dass ein Knoten mehr als einmal überdeckt wird, nämlich wenn zwei Knoten durch eine Kante verbunden werden und aufgrund der geometrischen Orte der Punkte ein weiterer Knoten auf dieser Kante liegt. Dies wird dann nicht als weitere Überdeckung des Knotens angesehen. Eine solche Überdeckung wird auch als Überdeckung durch eine „lange Kante“ bezeichnet.

Die letzte Unterscheidung betrifft die Art der Zielfunktion. Es werden stets knickbasierte Probleme betrachtet, einmal interessiert nur die Anzahl der Knicke (**L**). Das wird auch als *link distance* bezeichnet. Im anderen Fall verursachen die Knicke Kosten, die mit dem Knickwinkel zusammenhängen (**W**). Dieser Fall wird als *angle cost* bezeichnet.

Bei Problemen mit der Einschränkung P kann nur dann von einem Knoten v auf einen Knoten w und von dort aus wieder zurück zu v geknickt werden, falls dieser Kreis Länge 2 hat. Dieser Fall soll ausgeschlossen sein. Daher werden wir bei P zusätzlich fordern, dass Kreise stets mindestens Länge 3 haben.

Es wird nun eine Tabelle angegeben, die angibt, welche Arbeiten sich bereits mit Problemen befasst haben, die sich durch obige Klassifizierung beschreiben lassen, oder sich zumindest mit Problemen befasst haben, die den obigen sehr ähnlich sind.

	PL	PW	CL	CW
KN		[ABD+05]		[AKMS97]
KB				
KA				
SN				
SB	[KKM94]			
SA	[KKM94][BBD+08]			
HN			[AMP03]	[AKMS97]
HB			[SW00]	
HA			[SW00]	

Man könnte auf die Idee kommen, dass bei achsenparallelen Partitionsproblemen, bei denen U-Turns verboten sind, stets nur $0°$ oder $90°$-Knicke auftreten und die Winkelkosten daher bis auf einen konstanten Faktor mit den Knickkosten übereinstimmen. Das ist so aber nicht korrekt, wenn man Überdeckungen durch lange Kanten zulässt. Liegen dann nämlich drei Knoten auf einer Geraden, kann von den mittleren auf einen der äußeren und danach zurück auf den anderen äußeren geknickt werden, aber nicht zum letzten Knoten, sondern zum vorletzten. Selbst beim Verbot von langen Kanten besteht diese Möglichkeit, denn in einigen Fällen muss der erste der beschriebenen Knicke nicht zwangsläufig in einem Knoten erfolgt sein.

Nun werden die in den referenzierten Arbeiten bereits erzielten Resultate näher beleuchtet. Zunächst wird folgendes Resultate festgehalten:

Theorem 1.5. *[IPS82] Das Hamilton-Tour- und das* (s-t)*-Hamilton-Pfad-Problem in Gittergraphen sind $\mathcal{NP}$—vollständig.*

Daraus folgt bereits, dass obige Probleme mit der Kombination PS und PH (also Partitionierungsprobleme mit Suche nach Spannpfad oder Spannkreis) für die Klassen von Kompatibilitätsgraphen $\mathcal{NP}$-schwer sind, für die das Hamilton-Pfad-Problem schwer ist (z.b. Gitter), da bereits die Entscheidung über die Existenz einer zulässigen Lösung schwer ist.

Nach Theorem 1.5 ist auch das TSP auf Gittern $\mathcal{NP}$—vollständig (man kann zur Reduktion alle Kantengewichte auf −1 setzen). Offen ist die Frage nach der Komplexität des Traveling-Salesman-Problems auf **soliden** Gittern, wie sie in [LU97] betrachtet wurden. Diese sind definiert als Gitter, deren Rand eine zusammenhängende Menge im $\mathbb{R}^2$ bildet (im Sinne der Analysis). Auf diesen Graphen ist das Hamilton-Tour-Problem in Polynomialzeit lösbar. Die eben getroffene Komplexitätsaussage gilt für diese also nicht. Insbesondere sind diese Graphen als Kompatibilitätsgraphen also Kandidaten für möglicherweise polynomiale Spezialfälle der obigen Probleme.

1.5.1 Probleme mit Winkelkosten

Minimum Angle NodeTurn Cycle-Cover [KNCW]

Das Minimum-Angle Covering-Tour Problem wird in [AKMS97] betrachtet. Dort wird es als "Angular-Metric Cycle-Cover Problem" bezeichnet. Es wird dort zusammen mit mit dem Problem HNCW analysiert und es wird durch eine Reduktion von ONE-IN-THREE-3SAT gezeigt, dass KNCW $\mathcal{NP}$-vollständig ist. Es folgt eine Skizze dieser Reduktion.

Problem 3 (ONE-IN-THREE 3SAT)**.**

Instanz:	eine endliche Menge von Klauseln mit jeweils genau drei Literalen
Gesucht:	eine Belegung der Variablen mit booleschen Wahrheitswerten, so dass in jeder Klausel genau ein Literal wahr ist

Für die Reduktion wird ein gitterförmiger Graph konstruiert. Jeder Variablen werden drei Zeilen, jeder Klausel drei Spalten zugeordnet. Die ersten beiden Variablenzeilen werden als TRUE und FALSE interpretiert: Diese bestimmen also die Belegung. An den Enden werden horizontal und vertikal sogenannte V-Turn-Gadgets hinzugefügt, die sehr viele Punkte enthalten und sehr weit weg von den Gitterknoten sind (mindestens N^5, wenn N die Summe der Variablenanzahl und der Klauselanzahl bezeichnet). Auf diese Art und Weise wird die Form der Überdeckung erzwungen. Dass die V-Turn-Gadgets sehr weit entfernt sind, spielt eine wichtige Rolle, da verschiedene zulässige Variablenbelegungen nicht unbedingt die gleichen Überdeckungskosten verursachen. Durch die Entfernung wird aber ein Winkel sehr klein gemacht im Vergleich zu den Kosten, die durch eine unzulässige Überdeckung verursacht werden. Dadurch wird die Korrektheit der Reduktion sichergestellt. Diese Besonderheit führt auf Schwierigkeiten, wenn man versucht, die

Reduktion auf Gitter zu übertragen, da dort nicht durch große Entfernung der Winkel einfach klein gemacht werden kann, sondern immer Kosten von 90° verursacht werden.

An jedem Schnittpunkt einer Klauselspalte mit einer Variablenzeile, die zu einer Variablen gehört, die in der Klausel vorkommt, existieren drei Knoten. Die Variable sei x_i und die Klausel C_j. Dann hat man folgende Knoten:

- ein Knoten in der zu x_i gehörenden Spalte von C_j geschnitten mit der ersten Zeile von x_i (falls $x_i \in C_j$) bzw. mit der zweiten Zeile von x_i (falls $\overline{x_i} \in C_j$)
- zwei Knoten in den Spalten von C_j, die nicht zu x_i gehören und zwar in der zweiten Zeile, falls $x_i \in C_j$ und in der ersten Zeile, falls $\overline{x_i} \in C_j$

Ein Beispiel dafür kann man in Abbildung 1.2 sehen. Man muss beachten, dass die V-Turn-Gadgets sehr viel weiter weg sein sollten, als in dem Bild dargestellt, so dass die auftretenden Abbiegewinkel von einer Zeile auf eine benachbarte Zeile bzw. einer Spalte auf eine benachbarte Zeile so klein werden, dass sie für die Auswahl der zu überdeckenden Zeilen bzw. Spalten nur bei Mehrdeutigkeit einer optimalen Lösung eine Rolle spielen.

Eine optimale Überdeckung dieser Konstruktion sieht so aus, dass einerseits zeilenweise überdeckt wird: Für jede Variable die freie Zeile und eine der TRUE oder FALSE-Zeilen. Die überdeckten Spalten sind die beiden, die nicht den Schnittpunkt mit dem Literal enthalten, das die aktuelle Klausel erfüllt.

Für ein Beispiel siehe Abbildung 1.3.

Man kann dann zeigen, dass nur dann eine gültige Variablenbelegung existiert, wenn ein Cycle-Cover der Kosten höchstens $2\pi N + 4N^{-4}$ existiert, wobei N die Summe der Variablenanzahl und der Klauselanzahl bezeichnet.

Weiterhin wird ein $\mathcal{O}(\log n)$-Approximation über dynamische Programmierung vorgestellt. Dieser ist identisch mit dem $\mathcal{O}(\log n)$-Approximationsalgorithmus für HNCW und wird weiter unten vorgestellt.

Minimum-Turn Cycle-Cover [MTCC])

Problem 4 (Minimum-Turn Cycle Cover Problem)**.**

Instanz:	endlich viele Punkte $P \in \mathbb{R}^2$
Gesucht:	Kreise $p_{k1}, \ldots, p_{k\ell}, p_{k1}, k = 1, \ldots, n$, so dass jeder Punkt in mindestens einem Kreis enthalten ist
Minimiere:	Winkelkosten

Dieser Abschnitt beschäftigt sich mit den bekannten Resultaten zum Minimum-Turn Cycle Cover Problem. Im Rahmen der obigen Klassifikation handelt es sich wie im vorigen Abschnitt um KNCW mit der zusätzlichen Einschränkung, dass alle Knoten in der Knotenmenge V ganzzahlige Koordinaten haben und der Kompatibilitätsgraph der Unit-Distance-Graph auf diesen Knoten ist. Dabei wird vorausgesetzt, dass keine isolierten Knoten existieren.

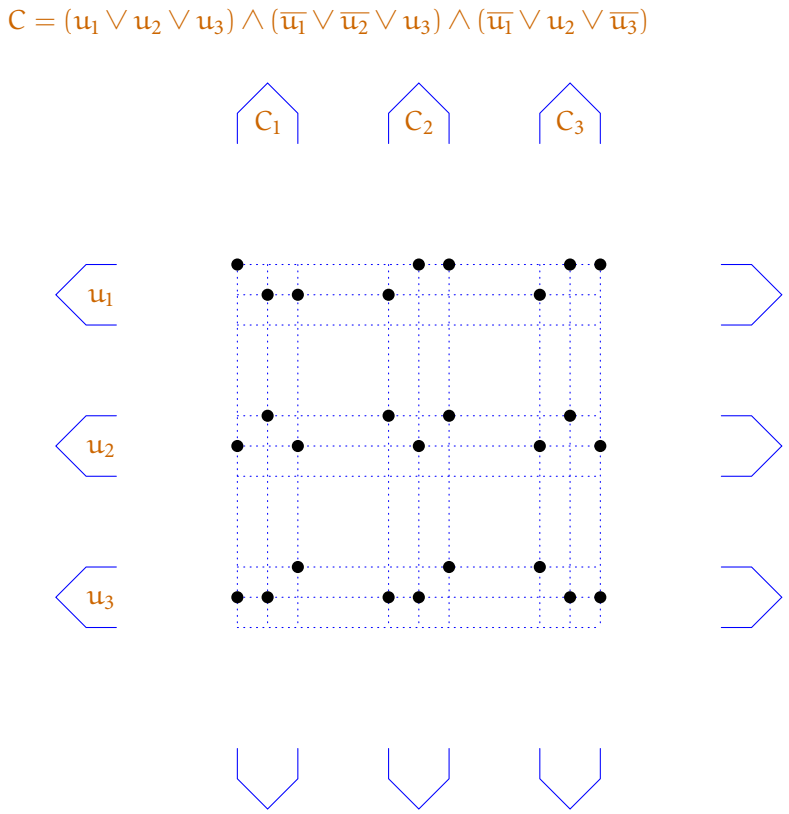

Abbildung 1.2: Reduktion einer Instanz des Angular-Metric Cycle-Cover-Problems

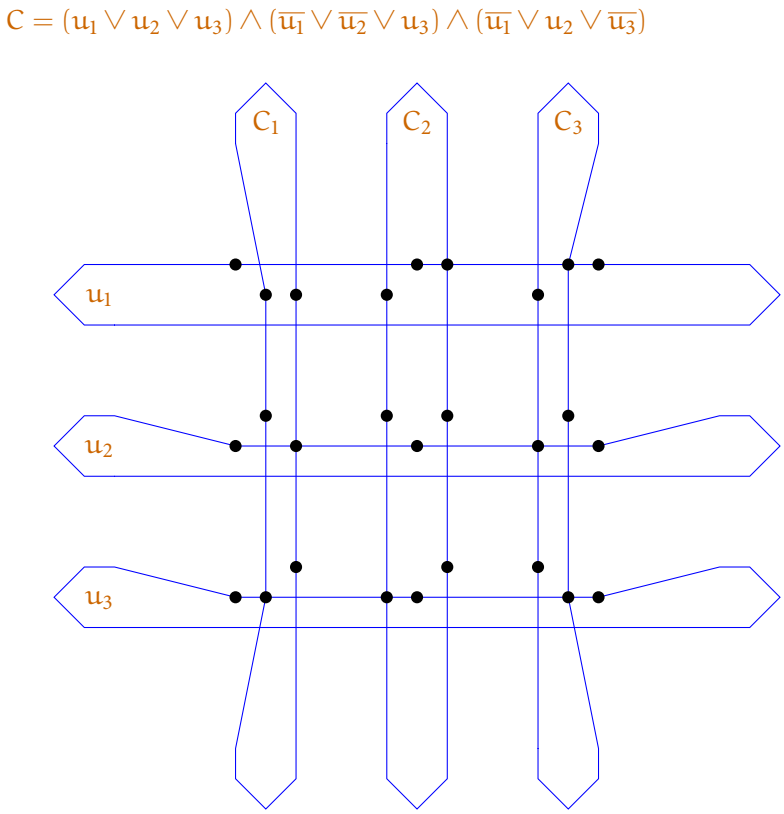

Abbildung 1.3: Optimallösung einer Reduktion des Angular-Metric Cycle-Cover-Problems

Durch die Bedingungen kann man sich eine Probleminstanz als Gitter vorstellen. Für das KNCW wurde oben die $\mathcal{NP}$-Vollständigkeit genannt. Bei der Reduktion wird entscheidend die Vollständigkeit des Kompatibilitätsgraphen ausgenutzt, um unerwünschte Effekte zu verhindern. Durch die zusätzliche Starrheit, die aufgrund der Gitterstruktur hinzukommt, funktioniert die Reduktion des KNCW für dieses Problem so nicht. Die Komplexität ist offen.[ABD+05].

Leichte Spezialfälle Sei T ein Subgraph von $G = (V, E)$ mit Maximalgrad 3. Dann kann das MTCC auf die Suche nach einem minimalgewichteten Matching reduziert werden.

Das liegt im Wesentlichen daran, dass sich Knickkosten in Gittern vom Maximalgrad 3 auf Kantenkosten zurückführen lassen: Hat ein Knoten Grad 1, bekommt seine Kante Gewicht 2. Hat ein Knoten Grad 2, bekommen seine beiden Kanten ein Gewicht von 0.5, falls der Knick echt ist, sonst 0. Hat ein Knoten Grad 3, gibt es zwei Kanten, die auf der selben Geraden liegen und eine, die dazu orthogonal ist. Diese bekommt Gewicht 1. Damit hat man zumindest lokal die Abbiegekosten auf Kantenkosten zurückgeführt. Dass man diese Idee zu einem Lösungsverfahren für Graphen mit Maximalgrad 3 ausbauen kann, zeigt der nächste Abschnitt.

Approximation Als Hilfsmittel bei der Approximation des MTCC-Problems wird zunächst ein exakter, polynomialer Algorithmus für **Boundary Cycle Cover** vorgestellt. Diese sind Kreisüberdeckungen von Knotenmengen, die Teilmenge des Gitterrands sind. Bei einem optimalen Boundary-Cycle-Cover können auch Knoten überdeckt werden, die nicht am Rand liegen - von diesen werden jedoch nicht notwendig alle überdeckt.

Ein minimalgewichtetes Boundary-Cycle-Cover kann durch die Suche eines minimalgewichteten, perfekten Matchings in einem Hilfsgraphen optimal gelöst werden. Dazu muss nur der Hilfsgraph geeignet definiert werden. Die benutzte Konstruktion findet sich in ähnlicher Weise auch in [ABD+05].

Die wesentliche Idee bei der Konstruktion besteht darin, eine optimale Kreisüberdeckung einer Teilmenge am Rand als ein perfektes 2-Matching zwischen den Knoten zu verstehen. Zwei Knoten sind dabei durch eine Kante verbunden, wenn sie auf ein gemeinsamen Kreis liegen und sie in einem der beiden möglichen Durchlaufsinne des Kreises als Randknoten direkt aufeinanderfolgen, zwischen ihnen also keine weiteren Randknoten liegen.

Das Gewicht der Kanten bei der Suche nach dem 2-Matching entspricht dabei den Abbiegekosten einer Verbindung zwischen diesen beiden Knoten. Ein Knoten kann dann von mehreren Kreisen überdeckt werden, da nichts über die Art und Weise gesagt ist, wie die verbindenden Wege zwischen den Randknoten verlaufen.

Durch eine Knotenverdopplung und das Verbinden der beiden zusammengehörenden Knoten durch eine Kante kann das Finden des 2-Matchings auf das Finden eines Matchings reduziert werden.

Die Menge der ausgewählten Randknoten sei mit P bezeichnet. Es wird nun beschrieben, wie der Hilfsgraph G_P für das minimalgewichtete perfekte Matching konstruiert werden kann. Für jeden Knoten $p \in P$ gibt es in G_P ein Paar von Knoten (p_0, p_1). p_0 und p_1 seien durch eine Kante verbunden. Da der Knoten p am Rand liegt, kann man vier Fälle unterscheiden:

Setze

$$c(p) := \begin{cases} 0 & \deg(p) = 3 \\ 0 & \deg(p) = 2 \text{ und liegt mit seinen adjazenten Knoten auf einer Geraden} \\ 1 & \deg(p) = 2 \text{ und liegt mit seinen adjazenten Knoten nicht auf einer Geraden} \\ 2 & \deg(p) = 1 \end{cases}$$

Mit δ_i^+ sei eine der beiden Richtungen bezeichnet, mit δ_i^- ihre Gegenrichtung. Entsprechend den minimalen Abbiegekosten eines verbindenden Weges werden nun (ungerichtete) Kanten von allen Knoten v_i^k zu allen Knoten v_j^ℓ, $j \neq k$ eingefügt. Die Kosten einer solchen Kante (p_i^k, p_j^ℓ) entsprechen den minimalen Abbiegekosten eines Weges, der am Knoten i in Richtung k beginnt, zum Knoten j führt und dort in Richtung ℓ endet. Hat der Graph $2n$ Knoten, so hat jeder Knoten in ihm also Grad $2n - 2$.

Man sucht nun ein perfektes, minimalgewichtetes Matching in G_P. Dieses benutzt abwechselnd die Verbindungskanten zwischen einem Knotenpaar p_0 und p_1 und die Kanten, die den minimalen Abbiegekosten eines verbindenden Weges entsprechen.

Außerdem entspricht jeder Lösung des MTCC-Problems ein solches 2-Matching. Sei nämlich L eine solche Lösung. Betrachte nun die Menge der Randknoten und verbinde zwei Randknoten mit einer Kante, wenn sie auf einem der Kreise der Lösung L unter Vernachlässigung der Nichtrandknoten direkt aufeinander folgen. Dadurch entsteht zunächst kein 2-Matching, sondern ein (nicht notwendig einfacher) Graph $\overline{M}$, in dem jeder Knoten geraden Grad besitzt. Die Kantenkosten im $\overline{M}$ übernehme man aus G_P.

Es soll nun gezeigt werden, dass man aus diesem Graphen Kanten entfernen kann, bis ein 2-Matching entsteht, dessen Kosten den Kosten der Lösung L entsprechen und das die Lösung L von der Auswahl seiner Kanten her auch ermöglicht. Dieses lässt sich auf natürliche Weise als Familie von Kreisen $C_i, i = 1 \,\mathtt{dots}, k$ verstehen, die den Kreisen im Ausgangsgraphen entsprechen. Es soll nun spezifiziert werden, was mit den Knoten geschehen soll, deren Grad größer als 2 ist, weil der entsprechende Knoten im Ausgangsgraphen in der Lösung L auf mehr als einem Kreis liegt.

Dazu geht man folgendermaßen vor. Sei v ein Knoten mit $\deg(v) > 2$. Sei C_i ein Kreis, der v enthält. Seien w_1 und w_2 die Nachbarn von v auf dem Kreis C_i. Man entferne im Graphen $\overline{M}$ die Kanten (v, w_1) und (v, w_2) (existieren sie mehrfach, dann eine davon). Füge nun eine Kante (w_1, w_2) mit dem entsprechenden Gewicht aus G_P ein. Die Kosten von $\overline{M}$ bleiben dadurch gleich, da durch die Entfernung der Kanten die Kosten von $\overline{M}$ höchstens kleiner werden können, die Lösung aber nach Voraussetzung optimal war. Der abbiegeminimale Weg, durch die Kantenabfolge (w_1, v, w_2) induziert wird, hat also die gleichen Kosten wie der Weg, der durch (w_1, w_2) induziert wird. Daher ist er weiterhin bezüglich der Kosten zulässig, was zeigt, dass auch nach der Reduktion des Grades in $\overline{M}$ die Lösung L eine der Lösungen ist, die im ursprünglichen Graphen induziert wird.

Es gilt:

Theorem 1.6. *Jedes Boundary-Cycle-Cover in* G *mit Kosten* t *entspricht einem perfekten Matching in* G_P *der Kosten*

$$t - \sum_{p \in P} c(p).$$

Die Aussage folgt direkt aus der Konstruktion der Kantengewichte, vergleiche Lemma 4.1 in [ABD+05]. Das durch das perfekte Matching induzierte Boundary-Cycle-Cover ist also optimal. Dieses kann man zur Konstruktion von Approximationsalgorithmen für das volle Problem benutzen. So wird später dieser Algorithmus zusammen mit einer Streifenüberdeckung zu einer 2.5-Approximation für das MTCC führen.

Streifenüberdeckungen Der **Träger** eines Geradenstücks ist die Menge der Punkte des Geradenstücks in der Ebene. Ein **Streifen** in einem Gitter ist ein im Träger der Kanten enthaltenes, inklusionsweise maximales Geradenstück. Eine **Streifenüberdeckung** ist eine Menge von Streifen, die alle Knoten eines Gitters überdecken. Eine **minimale Streifenüberdeckung** ist eine Streifenüberdeckung mit der geringsten Anzahl an Streifen.

Lemma 1.7. *[ABD+05] Die Größe einer minimalen Streifenüberdeckung ist eine untere Schranke an die Minimalzahl von Knicken eines Cycle Covers des Graphen.*

Beweis. Cycle Cover induzieren Streifenüberdeckungen: Verlängere jede Kante des Cycle Covers auf Maximallänge. Man erhält eine Streifenüberdeckung mit höchstens so vielen Streifen wie Knicken. □

Das im Sinn der linearen Programmierung duale Problem ist eine **Turmstellung**: Stelle möglichst viele Türme auf die Knoten des Gitters, so dass sich keine zwei Türme schlagen können. Es gilt:

Lemma 1.8. *Die Größe einer maximalen Turmstellung ist eine untere Schranke an die Minimalzahl von Knicken eines Cycle Covers des Graphen.*

Beweis. Sei C ein Kreis eines Cycle Covers. Angenommen, der Kreis besucht die Türme $q_1, \ldots, q_k$ in dieser Reihenfolge. Weil sich keine zwei Türme gegenseitig schlagen können, muss zwischen zwei Türmen stets ein Knick sein. Da C ein Kreis ist, muss die Anzahl der Knicke in C also mindestens k sein. □

Es wird nun die Dualität der beiden Probleme bewiesen.

Lemma 1.9. *Sei* G *ein Gitter. Dann haben eine minimale Streifenüberdeckung und eine maximale Turmstellung die gleiche Größe.*

Beweis. Das Turmstellungsproblem kann als Matching in einem bipartiten Graphen B verstanden werden. Sei dazu V_1 die Menge der vertikalen und V_2 die Menge der horizontalen Streifen in G. Diese sollen die Knoten des Graphen B sein. Es gebe in B eine Kante von $v_1 \in V_1$ nach $v_2 \in V_2$, falls v_1 und v_2 in G einen Knoten gemeinsam haben. Ein kardinalitätsmaximales Matching in B entspricht dann einer maximalen Turmstellung in G.

Bei einer minimalen Streifenüberdeckung muss jeder Knoten des Graphen G überdeckt werden. Dazu äquivalent ist, dass jeder Streifen von G überdeckt wird. Auf diese Weise sieht man: Eine minimale Streifenüberdeckung in G entspricht einem minimalen Vertex Cover in B. □

Diese Aussage folgt dann mit

Theorem 1.10 (König)**.** *Sei* G *ein bipartiter Graph. Dann ist die Anzahl der Kanten in einem kardinalitätsmaximalen Matching in* G *gleich der Anzahl der Knoten in einem minimalen Vertex Cover von* G.

Theorem 1.11. *[ABD+05] Es gibt eine 4-Approximation für das MTCC mit Laufzeit* $\mathcal{O}(n^{2.5} \log n)$.

Beweis. Suche eine optimale Streifenüberdeckung und mache aus jedem Streifen einen Kreis mit 4 Knicken. Weil optimale Streifenüberdeckungen nach Lemma 1.7 untere Schranken sind, folgt die Behauptung. □

Theorem 1.12. *Es gibt eine 2.5-Approximation für MTCC mit Laufzeit* $\mathcal{O}(n^3)$.

Beweis. Suche eine optimale Streifenüberdeckung. Betrachte die Endknoten der Streifen. Um einen besseren Approximationsfaktor zu erreichen, sollen nun die Endpunkte der Streifen auf kostenminimale Weise verbunden werden.

Konstruiere einen Hilfsgraphen G_P. Die Konstruktion ist nun ähnlich wie die Konstruktion oben für das Boundary-Cycle-Cover. Für jeden Streifenendknoten habe G_P **einen** Knoten. G_P sei hier ein vollständiger Graph. Die Kosten einer Kante $e = (u, v)$ seien die Kosten eines abbiegeminimalen Weges, der von dem u überdeckenden Streifen u senkrecht zum Streifen verlässt und bei v in Richtung senkrecht zu dem v überdeckenden Streifen ankommt. Berechne nun in G_P ein perfektes, kostenminimales Matching M. Da ein optimales Cycle-Cover zwei solche Matchings induziert und sich die Kosten des Cycle-Covers aus der Summe der Kosten der beiden matchings ergibt, folgt $d(M) \leqslant \frac{OPT}{2}$, wenn OPT den Zielfunktionswert eines optimalen Cycle-Covers bezeichnet.

Es wird nun angenommen, dass von den Endpunkten der Streifen ein Knick auf einen solchen ausgewählten Weg erfolgt. Nun werden Kreise gebildet, indem abwechselnd Streifen und die minimalen Verbindungswege aneinandergehängt werden.

Die Knicke von Streifen zu Wegen und umgekehrt sind genau so viele wie Endpunkte der Streifen. Aus dem vorherigen Argument für die 4-Approximation erhält man sofort, dass die Anzahl der Streifenendpunkte höchstens 2·OPT beträgt. Das kostenminimale Matching hatte $d(M) \leqslant \frac{OPT}{2}$, beides zusammen ergibt also eine 2.5-Approximation. □

Theorem 1.13. *Für jede jedes Cycle Cover in* G *gibt es ein Cycle Cover mit gleich vielen Knicken und maximaler Überdeckungszahl 4.*

Beweis. Sei ein Knoten 5 mal überdeckt. Dann gibt es eine Kante e, die 3 mal benutzt ist. Dann gibt es drei Paare $(1, 1')$, $(2, 2')$, $(3, 3')$ von Knoten, deren Verbindungswege die Kante e benutzen. Diese sollen jeweils die Knoten sein, an denen ausgehend von dem horizontalen Geradenstück, das die Kante e enthält, das nächste Mal geknickt wird. Die Verbindungen definieren dann ein perfektes Matching M auf dem K_6 mit den Knoten $\{1, 2, 3, 1', 2', 3'\}$. Die Knoten $\{1, 2, 3\}$ heißen die **linken Endpunkte**, die Knoten $\{1', 2', 3'\}$ die **rechten Endpunkte**.

Sei M' ein weiteres solches Matching, das mit M zusammen einen Kreis bildet.

Es werden nun zwei Fälle unterschieden. Zunächst wird Fall 1 betrachtet: es sind im M' zwei rechte Endpunkte durch eine Kante verbunden, ohne Beschränkung der Allgemeinheit $1'$ und

$2'$. Dann müssen auch zwei linke Endpunkte durch eine Kante verbunden sein. Diese können jedoch nicht 1 und 2 sein, da die beiden Matchings zusammen einen Kreis bilden. Wir nehmen ohne Beschränkung der Allgemeinheit an, dass es die Knoten 1 und 3 sind. Dann müssen 2 und $3'$ verbunden sein. Wir können $(1,1')$, $(2,2')$, $(3,3')$ dann durch $(1,2')$, $(2,3)$, $(1',3')$ ersetzen. Die neue Tour hat dann dieselbe Anzahl von Knicken. Im anderen Fall sind alle rechten Endpunkte mit linken Endpunkten verbunden. Sei ohne Beschränkung der Allgemeinheit $1'$ mit 2 verbunden. Dann ist das zweite Matching $(1,3')$, $(2,1')$, $(3,2')$. Man kann nun $(1,1')$, $(2,2')$, $(3,3')$ durch $(1,3)$, $(2,3')$, $(1',2')$ ersetzen.

In beiden Fällen erhöht sich die Anzahl der Knicke nicht und kein Knoten ist öfter überdeckt als vorher, außerdem sind auch einige Knoten nun weniger überdeckt als vorher.

Diese Prozedur kann wiederholt werden, bis kein Knoten mehr als 4 mal überdeckt ist. Das passiert nach endlich vielen Schritten, da durch die Verringerung der Knotenüberdeckung diese Prozedur nicht zu einem vorherigen Zustand zurückkehren kann. □

MTCCNU

Als MTCCNU wird die Variante des MTCC-Problems bezeichnet, bei dem U-Turns verboten sind, jeder Knick also Kosten von 1 verursacht, aber trotzdem mehrfach überdeckt werden darf.

Man könnte vermuten, dass in diesem Fall auf Graphen, auf denen das MTCP-Problem (siehe Abschnitt 1.5.1) zulässig ist, optimale Lösungen des MTCP-Problems auch für MTCCNU optimal sind.

Das ist aber nicht so. Es gilt:

Lemma 1.14. *Seien* $c_1(G)$ *die Kosten einer optimalen Lösung des MTCP-Problems auf dem Graphen* G *und sei* $c_2(G)$ *die Kosten einer optimalen Lösungen des MTCCNU-Problems auf dem Graphen* G*, die Werte seien* ∞*, falls keine Lösung existiert. Dann gilt:*

$$\limsup_{G \text{ MTCP-zulässig}} \frac{c_2(G)}{c_1(G)} \geqslant 2.$$

Beweis. Sei C_k der Graph aus Abbildung 1.4, hier gezeichnet für $k = 3$. k bezeichne die Anzahl Blöcke aus der Abbildung. Die einzige zulässige Lösung des MTCP-Problems ist dann in Abbildung 1.5 zu sehen. Aufgrund der Notwendigkeit der Überdeckung der oberen und der unteren Reihe sowie der linken und der rechten Spalte gibt es offenbar keine andere Überdeckungsmöglichkeit.

Das MTCCNU-Problem hat dagegen eine deutlich kostengünstigere Überdeckungsmöglichkeit, die in Abbildung 1.6 zu sehen ist.

Die Kosten für die MTCP-Überdeckung aus den 4 Knicken zur Überdeckung der horizontalen Doppelreihe und den zu erkennenden 8 Knicken pro Doppelspalte. Die Kosten für MTCCNU setzen sich aus einem Rechteck mit Kosten 4 pro Doppelspalte und aus den 4 Knicken zur Überdeckung der horizontalen Doppelreihe zusammen. Es ist

$$\limsup_{k\to\infty} \frac{8k+4}{4k+4} = 2.$$

□

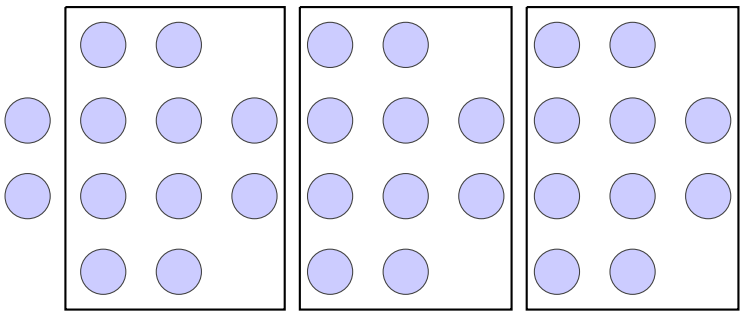

Abbildung 1.4: MTCP-Instanz aus Lemma 1.14

Kosten: $8k + 4$

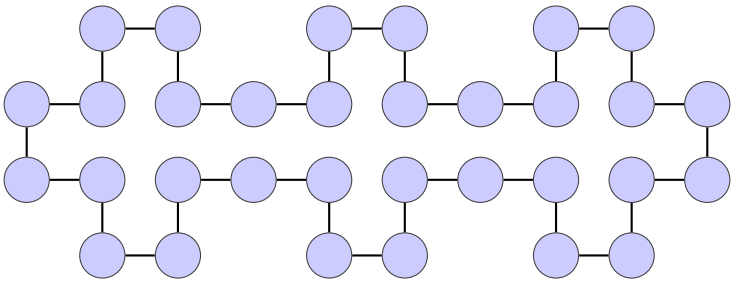

Abbildung 1.5: MTCP-Optimallösung aus Lemma 1.14

Kosten: $4k + 4$

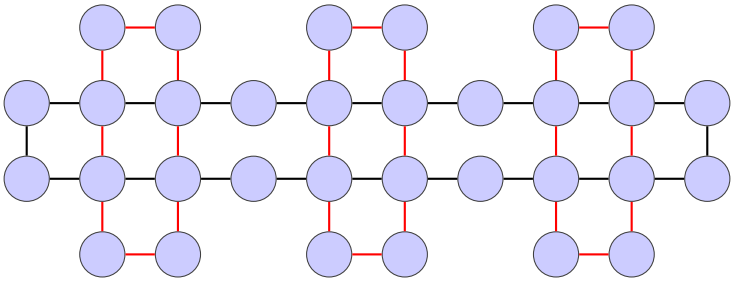

Abbildung 1.6: MTCCNU-Optimallösung aus Lemma 1.14

Minimum-Angle NodeTurn Covering-Tour Problem [HNCW]

Das Minimum-Angle Covering-Tour Problem wird in einer Variante in [AKMS97] betrachtet. Dort heißt es "Angular-Metric Traveling Salesman Problem", obwohl Knoten beliebig oft überdeckt werden dürfen, während beim klassischen Traveling Salesman Problem die zugrundeliegende Struktur eine Hamilton-Tour ist, eine Tour also, die jeden Knoten genau einmal besucht.

Theorem 1.15. *Das Minimum-Angle NodeTurn Covering-Tour Problem ist $\mathcal{NP}$-vollständig.*

Der Beweis des Theorems wird in [AKMS97] grob skizziert.

Theorem 1.16. *[AKMS97] Das Minimum Angle NodeTurn Covering-Tour Problem erlaubt eine $\mathcal{O}(\log n)$-Approximation.*

Zum Beweis wird ein Lemma aus [AKMS97] benutzt:

Lemma 1.17. *Sei $P \subset \mathbb{R}^2$, $|P| = n$. Dann gibt es einen polynomialen Algorithmus, der einen kardinalitätsmaximalen Pfad mit Gesamtwinkel höchstens π findet.*

Beweis. Für $1 \leqslant k \leqslant n$ sei $A_s[e,k]$ für eine gerichtete Kante e der minimale Winkel eines Pfad der Kardinalität k, der mit s beginnt und mit e endet.

Die $A_s[e,k]$ können über Bottom-Up Dynamic-Programming berechnet werden: Setze in jedem Schritt

$$A_s[e,k] := \begin{cases} \min_f\{A_s[f,k-1] + \theta_{f,e}\} & \text{min ist kleiner oder gleich } \pi \\ \infty & \text{sonst} \end{cases}$$

In einem Pfad mit Winkel π können sich weder Kanten noch Knoten wiederholen.

□

Beweis. [von Theorem 1.16]

Sei θ der Zielfunktionswert einer Optimallösung des Problems. Setze $K = \frac{\theta}{\pi}$. Dann muss es einen Subpfad der Kardinalität $\frac{n}{K}$ geben, der einen Winkel von höchstens π hat.

Durch Induktion zeigt man, dass man für jede Instanz des Problems mit Optimalwert $\theta \leqslant K\pi$ in Polynomialzeit ein Cycle-Cover berechnen kann, das $K \cdot \log n$ Kreise hat und jeder von den Kreisen Winkel höchstens 3π.

Durch Zusammenfügen von zwei Kreisen entstehen Winkelkosten von höchstens 2π. Dadurch erhält man die Behauptung.

□

Demnach ist HNCW ohne Einschränkungen an die Kompatibilität $\mathcal{NP}$-vollständig. Es kann $\mathcal{O}(\log n)$-approximiert werden.

Minimum Angle NodeTurn Cycle-Partition [KNPW]

Minimum-Turn Cycle-Partition [MTCP]

Problem 5 (Minimum-Turn Cycle Partition Problem)**.**

Instanz:	endlich viele Punkte $P \in \mathbb{R}^2$
Gesucht:	elementare Touren $p_{k1}, \ldots, p_{k\ell}, p_{k1}, k = 1, \ldots, n$, so dass jeder Punkt in genau einer Tour enthalten ist.
Minimiere:	Winkelkosten

Es handelt sich hierbar nach der obigen Klassifikation um das Problem mit der zusätzlichen Einschränkung, dass alle Knoten in der Knotenmenge V ganzzahlige Koordinaten haben und der Kompatibilitätsgraph der Unit-Distance-Graph auf diesen Knoten ist. Es wird vorausgesetzt, dass keine isolierten Knoten existieren.

Durch die Bedingungen kann man sich eine Probleminstanz als Gitter vorstellen. Das Interesse an diesem Problem stammt vor allem daher, dass dieses Problem nach unserem Wissen nie zuvor untersucht wurde und es Indizien gibt, die darauf hinweisen, dass das Problem möglicherweise in $\mathcal{P}$ liegt. Die längenbasierten Versionen des MTCC- und MTCP-Problems sind $\mathcal{NP}$-schwer respektive in $\mathcal{P}$ und die Vermutung liegt nahe, dass sich dieser Sachverhalt auf die abbiegekostenbasierten Versionen überträgt.

Offenbar ist nicht jedes Gitter für das Problem zulässig. Es dürfen zum Beispiel keine Knoten auftreten, deren Grad kleiner als 2 ist. Eine notwendige und hinreichende Bedingung für die Zulässigkeit eines Gitters ist die folgende:

Theorem 1.18. *[Bel50][Sch03] Für zwei Teilmengen K und S der Knotenmenge eines Graphen bezeichne E[K, S] die Menge der Kanten, die zwischen diesen Teilmengen verlaufen. Dann gilt:*
Ein Graph G = (V, E) hat einen einfachen 2-Faktor genau dann, wenn

$$|S| <= |U| + \sum_{K} \left\lfloor \frac{1}{2} |E[K, S]| \right\rfloor \tag{1.2}$$

für jedes Paar von disjunkten Teilmengen U, S von V, wobei S eine stabile Menge ist und die Summe über alle Komponenten von G − U − S läuft.

Zum Beweis wird die Tutte-Berge-Formel benutzt:

Theorem 1.19 (Tutte-Berge-Formel)**.** *Sei für einen Graphen G mit o(G) die Anzahl der ungeraden Zusammenhangskomponenten von G bezeichnet. Sei ν(G) die maximale Größe eines Matchings in G.*
Dann gilt

$$\nu(G) = \frac{1}{2} \cdot \min_{U \subseteq V} (|V| + |U| - o(G - U)) \tag{1.3}$$

Zunächst wird eine Konstruktion von Tutte betrachtet, mit der man die maximale Größe eines einfachen 2-Matchings mit der maximalen Größe eines Matchings in einem abgeleiteten Graphen in Verbindung setzen kann.

Lemma 1.20. *Sei* $G = (V, E)$ *ein Graph. Konstruiere einen Graphen* G' *wie folgt. Für jeden Knoten* v *von* G *habe* G' *Knoten* v' *und* v''*. Für jede Kante* $e = (u, v)$ *hat* G' *Knoten* $p_{e,u}$ *und* $p_{e,v}$*. Das definiert die Knoten von* G'*. Für jede Kante* $e = (u, v)$ *von* G *hat* G' *die Kanten* $(u', p_{e,u})$*,* $(u'', p_{e,u})$*,* $(p_{e,u}, p_{e,v})$*,* $(v', p_{e,v})$ *und* $(v'', p_{e,v})$*.*

G' *wird im weiteren als* ***Tutte-Graph*** G' *zu einem Graph* G *bezeichnet. Sei* $\nu(G)$ *die maximale Größe eines Matchings in* G*,* $\nu_2^s(G)$ *die maximale Größe eines 2-Matchings in* G*. Dann gilt*

$$\nu_2^s(G) = \nu(G') - |E| \tag{1.4}$$

Beweis. Zunächst gibt es ein Matching M in G' mit der Eigenschaft, dass für jede Kante $e = (u, v)$ von G beide Knoten $p_{e,u}$ und $p_{e,v}$ von M überdeckt werden. Ist nämlich keiner der beiden Knoten in einem Matching M überdeckt, kann die Kante zwischen den Knoten hinzugefügt werden, was einen Widerspruch zur Maximalität ergibt. Ist nur einer der beiden Knoten überdeckt, kann an dem anderen der beiden Knoten die Kante entfernt werden und die Kante $(p_{e,u}, p_{e,v})$ hinzugefügt werden, was an der Maximalität des Matchings nichts ändert.

Sei M ein solches Matching in G'. Dann bilden die Kanten e von G, für die die Kante $(p_{e,u}, p_{e,v})$ nicht zu M gehört, ein einfaches 2-Matching N in G. Den Kanten in N entsprechen im Matching M zwei Kanten. Den Kanten $e = (u, v)$ aus E, die nicht in N sind, entspricht in M eine Kante, nämlich die Kante $(p_{e,u}, p_{e,v})$. Daher sind in M genau $|E|$ mehr Kanten enthalten als in N. Es gilt also

$$|N| = |M| - |E|.$$

Daher folgt " $\geqslant$" in (1.4).

Sei nun N ein maximales 2-Matching in G. Für jede Kante $e = (u, v)$ in N füge in M die Kanten $(u', p_{e,u})$ und $(p_{e,v}, v'')$ ein. Für jede Kante $e = (u, v)$ in E, aber nicht in N, füge zu M die Kante $(p_{e,u}, p_{e,v})$ hinzu. Das liefert ein Matching M in G' mit

$$|N| = |M| - |E|.$$

Es folgt " $\leqslant$" in (1.4). □

Nun wird das Theorem 1.18 bewiesen. Der Beweis folgt der Argumentation in [Sch03].

Beweis. [von Theorem 1.18]

Zum Beweis wird zunächst angenommen, dass G einen einfachen 2-Faktor M besitzt. Dann ist also zu zeigen, dass (1.2) gilt.

Seien also U und S disjunkt und sei S eine stabile Menge. Dann hat M höchstens $2|U|$ Kanten, die mit Knoten aus U inzidieren. Sei K eine Zusammenhangskomponente von $G \setminus \{U \cup S\}$.

Dann ist mit der Stabilität von S

$$\begin{aligned} 2|M \cap E[K \cup S]| &= 2|M \cap E[K]| + 2|M \cap E[K, S]| \\ &\leqslant 2|M \cap E[K]| + |M \cap E[K, S]| + |E[K, S]| \\ &\leqslant 2|K| + |E[K, S]|, \end{aligned}$$

da man in der vorletzten Zeile die ersten beiden Summanden durch $|K|$ abschätzen kann. Es folgt, dass die Anzahl der Kanten in M, die von $K \cup S$ aufgespannt werden, höchstens $|K| + \lfloor \frac{1}{2}|E[K,S]| \rfloor$ ist. Summiert über alle Komponenten, erhält man

$$\begin{aligned} |V| &= |M| \\ &\leqslant 2|U| + \sum_K \left(|K| + \left\lfloor \left| \frac{1}{2} E[K,S] \right| \right\rfloor \right) \\ &= 2|U| + |V| - |U| - |S| + \sum_K \left(\left\lfloor \left| \frac{1}{2} E[K,S] \right| \right\rfloor \right) \\ &= |U| + |V| - |S| + \sum_K \left(\left\lfloor \left| \frac{1}{2} E[K,S] \right| \right\rfloor \right) \end{aligned}$$

Daher folgt " $\Rightarrow$".

Es gelte nun also (1.2). Zu zeigen ist, dass G einen einfachen 2-Faktor besitzt. Konstruiere den Graphen G' wie in Lemma 1.20. Dann hat G nach (1.4) genau dann einen einfachen 2-Faktor, wenn G' ein Matching der Größe $|V| + |E|$ besitzt.

Durch Anwendung der Tutte-Berg-Formel auf den Graphen G' erhält man, dass es eine Teilmenge $X \subseteq V'$ gibt, die mindestens

$$|V'| + |X| - 2\nu(G')$$

viele ungerade Zusammenhangskomponenten besitzt. Sei X unter diesen inklusionsweise minimal.

Für jedes $v \in V$ gehören dann von v' und v'' entweder beide oder keiner zu X. Gehöre nämlich v' zu X , x'' aber nicht. Entfernt man v' aus X, so wird die Anzahl der ungeraden Komponenten von $G \setminus X$ um höchstens eins geringer, wenn nämlich die Komponente, in der x'' lag, dadurch von ungerade auf gerade wechselt. Anderes kann nicht passieren, da x' und x'' dieselben Nachbarn haben und x'' bereits in $G \setminus X$ enthalten war. Es können durch das Entfernen von x' aus X also in $G \setminus X$ keine Komponenten verbunden werden. Durch Entfernen von v' aus X wird auch $|X|$ um 1 geringer. Die Ungleichung gilt also weiterhin und man hat einen Widerspruch zur inklusionsweisen Minimalität von X.

Betrachte nun eine Kante $e = (u, v)$ von G mit $p_{e,v} \in X$. Es soll gezeigt werden, dass die drei Nachbarn von $p_{e,v}$ in drei verschiedenen ungeraden Komponenten von $G' \setminus X$ liegen. Durch das Entfernen von $p_{e,v}$ aus X verringert sich $|X|$ um 1. Es reicht also zu zeigen, dass bei weniger als drei Nachbarn in verschiedenen ungeraden Komponenten sich die Anzahl der ungeraden Komponenten um höchstens 1 verringert.

Liegen die Nachbarn in zwei verschiedenen ungeraden Komponenten, werden diese durch Entfernen in X in $G \setminus X$ verbunden. Da die Summe von zwei ungeraden Zahlen gerade ist, verringert sich die Anzahl nur um 1. Liegen die Nachbarn in nur einer Komponente, so verringert sich die Anzahl offenbar um 1. Liegen sie in keiner ungeraden Komponente, erhöht sich die Anzahl der ungeraden Komponenten sogar. In allen Fällen ergibt sich demnach ein Widerspruch zur inklusionsweisen Minimalität.

Ist also $p_{e,v} \in X$, so folgt, dass $p_{e,u}, v', v'' \notin X$. Weiterhin folgt $p_{f,v} \in X$ für alle Kanten f, die zu v inzident sind. Wäre nämlich $p_{f,v} \notin X$, so wären v' und v'' in $G' \setminus X$ über $p_{f,v}$ verbunden (und damit in derselben Zusammenhangskomponente).

Sei nun U die Menge aller $v \in V$ mit $v', v'' \in X$. Sei S die Menge aller $v \in V$, für die alle $p_{e,v}$ in X sind. Dann sind U und S disjunkt. Außerdem ist S eine stabile Menge nach Konstruktion des Graphen. Für jeden Knoten in U sind die beiden Knoten v' und v'' in X. Für jeden Knoten in S sind alle $p_{e,v}$ in X, also genausoviele wie die von diesem Knoten ausgehenden Kanten. Also ist

$$|X| = 2|U| + |\delta(S)|.$$

Jedem Knoten in S entsprechen zwei (nach den vorangehenden Überlegungen) isolierte Knoten in $G' \setminus X$. Diese bilden dort zwei ungerade Komponenten.

Sei $v \in U$. Dann sind v' und v'' in X und damit alle $p_{e,v}$ in $G' \setminus X$. Sei $e = (u, v)$ eine entsprechende Kante.

Falls $u \in U$, so bilden $p_{e,u}$ und $p_{e,v}$ eine gerade Komponente in $G' \setminus X$. Die Kante e verläuft dann von U nach U. Sei andernfalls $u \in S$. Dann ist $p_{e,v}$ ein isolierter Knoten in $G' \setminus X$, bildet also eine ungerade Komponente. Die Kante e verläuft dann also von U nach S. Sei nun $u \notin \{U \cup S\}$. Dann ist $u \notin X$ und damit ist $p_{e,u}$ in $G' \setminus X$, liegt also mit $p_{e,v}$ in einer Zusammenhangskomponente, die aus mehr als einem Knoten besteht und nicht von einer Kante herkommt, die innerhalb von U verläuft.

Diese werden nun betrachtet. Sei dazu u nicht in X und entspreche keinem der bisherigen Fälle. u ist einer eindeutigen maximalen Zusammenhangskomponente von $G' \setminus X$ enthalten. Diese enthält, wenn sie Teilgraphen erreicht, die nach U gehen, immer beide Endpunkte der Kante. Nach Konstruktion des Graphen sind jedem Knoten und jeder Kante in G stets zwei Knoten zugeordnet. Erreicht sie Kanten, die nach S gehen, enthält sie nur einen der beiden Endpunkte. Auf diese Weise kann sie also ungerade werden.

Sie ist genau dann ungerade, wenn ungerade viele Kanten von ihr nach S zeigen. Sei κ die Anzahl der Komponenten von $G \setminus \{U \cup S\}$ mit $|E[K, S]|$ ungerade. Dann hat man:

$$o(G' \setminus X) = 2|S| + |E[U, S]| + \kappa$$

Daher ist

$$\begin{aligned}
\nu_2^s(G) &\geqslant \nu(G') - |E| \\
&\geqslant \frac{1}{2}(|V'| + |X| - o(G' \setminus X)) - |E| \\
&= |V| + |U| + \frac{1}{2}|\delta(S)| - |S| - \frac{1}{2}|E[U, S]| - \frac{1}{2}\kappa \\
&= |V| + |U| - |S| - \sum_{K} \lfloor |E[K, S]| \rfloor \\
&\geqslant |V|
\end{aligned}$$

Die Größe eines maximalen 2-Matchings ist als mindestens $|V|$. Damit folgt die Behauptung.

□

1.5.2 Probleme mit Knickkosten

In diesem Abschnitt werden Probleme mit Knickminimierung betrachtet. Es wird zunächst überblicksartig die Literatur zu dem Thema referenziert.

Während die axiomatische Untersuchung von geometrischen Problemen bereits bis auf Euklid von Alexandrien zurückgeht, wurde die systematische Suche nach effizienten Algorithmen für geometrische Probleme erst durch [Sha75] begründet. Diese wird seitdem als "algorithmische Geometrie" (computational geometry) bezeichnet.

Die Verbindung von algorithmischer Geometrie mit der Bewegungssteuerung von Robotern erfolgte dann in [Rei79].

Das Problem, eine Tour durch eine Region im $\mathbb{R}^2$ zu finden, so dass die Minkowski-Summe der Tour mit einer geometrischen Figur die Region genau ausfüllt, tritt beim NC Pocket-Machining auf. (vgl. [Hel91]). Weil auf diese Weise Werkstücke ausgefräst werden, wird es in [ABD$^+$05] Milling-Problem genannt.

Durch die Arc-Routing-Community wurden einige Heuristiken für Probleme mit *Turn Penalties* entwickelt. (vgl. z.B. [CMMS02], [Mcb82]).

Probleme mit Link-Distance wurden zuerst in [Sur86] untersucht.

Die Komplexität von Watchman-Touren mit minimaler euklidischer Länge wird in [CN86] analysiert.

Die Verbindung von Watchman-Touren mit der Link-Distance wurde schließlich in [AL93] vorgenommen. Dort wird gezeigt, dass das Problem in einfachen Polygonen schon $\mathcal{NP}-$schwer ist und es dort eine polynomiale 3-Approximation gibt.

Eine weitere Analyse liefert das Paper [AMP03], dort wird das Problem als **Point Covering by a Tour**-Problem bezeichnet.

Minimum Link NodeTurn Covering-Tour [HNCL]

Theorem 1.21. *[AMP03] Das Minimum-Link NodeTurn Covering-Tour Problem ist $\mathcal{NP}$-vollständig.*

Der Beweis erfolgt dort durch eine Reduktion von POINT COVERING BY LINES. Dazu wird die Punktmenge mit einer affinen Transformation geeignet transformiert und es werden große V's zu den Punkten hinzugefügt. Die Kosten einer Überdeckung der Punkte durch Geraden entsprechen dann den Kosten bei der Überdeckung durch eine Tour bis auf einen Skalierungsfaktor, siehe [AMP03].

Problem 6 (POINT COVERING BY LINES)**.**

Instanz:	Punkte $p_1, \ldots, p_n \in \mathbb{R}^2$
Gesucht:	endlich viele Geraden, so dass jeder Punkt auf mindestens einer der Geraden liegt
Zielfunktion:	minimiere Anzahl der Geraden

Ein weiteres Resultate bezüglich dieses Problems ist das folgende:

Theorem 1.22. *[SW00] Es gibt einen $(2\rho+2)$-Approximationsalgorithmus für das Minimum-Bends TSP, wobei ρ die Approximationsschranke für SET COVER ist.*

Minimum Link axis-aligned Covering-Tour [HACL]

Das Problem wird von [SW00] analysiert. Dort heißt es Rectilinear Minimum-Bends TSP.

In [SW00] wird eine 2-Approximation angegeben. Der Algorithmus basiert auf der Idee, die Punkte zunächst mit Geraden zu überdecken und diese Geraden dann zu einer Tour zu verbinden. Diese Idee wird auch später zu einem 2.5-Approximationsalgorithmus für das Minimum-Turn Cycle-Cover Problem führen. Im allgemeinen Fall ist POINT COVERING BY LINES $\mathcal{NP}$-vollständig (siehe 1.5.2). Im achsenparallelen Fall genügt es aber, RECTLINEAR POINT COVERING BY LINES zu betrachten, das die zusätzliche Einschränkung aufweist, dass die Geraden entweder horizontal oder vertikal verlaufen müssen. Das reduziert die Komplexität des Problems erheblich:

Theorem 1.23. *RECTLINEAR POINT COVERING BY LINES kann in Zeit $\mathcal{O}(n^{1.5})$ gelöst werden.*

Beweis. Man kann ohne Einschränkung annehmen, dass die x- und y-Koordinaten paarweise verschieden sind (sonst kann man sie durch eine leichte Verschiebung dazu machen). Man konstruiert einen Graphen $G = (V, E)$: x-Koordinaten und y-Koordinaten sind Knoten. In G existiert die Kante (x, y), falls es einen Punkt mit den Koordinaten (x, y) in der Instanz von RECTLINEAR POINT COVERING BY LINES gibt.

Da die Kanten stets zwischen x- und y-Koordinaten verlaufen, ist G bipartit. Einer minimalen Geradenüberdeckung entspricht nun ein Vertex Cover in G. Bipartites Vertex Cover kann in Zeit $\mathcal{O}(n^{1.5})$ mit dem Algorithmus von Hopcroft und Karp gelöst werden. □

Aufbauend auf dieser Beobachtung kann man eine Bounding-Box um alle Punkte nehmen und diese erst horizontal S-förmig und dann vertikal S-förmig ablaufen. Ist m die Anzahl der Geraden, so hat diese Lösung höchstens $2m + \mathcal{O}(1)$ Knicke.

Minimum Link Covering-Tour [HBCL]

Das Problem wird in [SW00] bearbeitet, dort heißt es Minimum-Bends TSP.

Es wird für den allgemeinen Fall eine $\mathcal{O}(\log z)$-Approximation angegeben. Dabei bezeichnet z die maximale Anzahl kollinearer Punkte.

Sind stets nur zwei Punkte kollinear, so benötigt eine optimale Tour mindestens $\lfloor \frac{n}{2} \rfloor$ Knicke, da auf jedem Geradenstück höchstens 2 Punkte liegen können. Eine 2-Approximation ist in diesem Fall also trivial, denn die Triviallösung hat Kosten n.

Der $\mathcal{O}(\log z)$-Approximationsalgorithmus basiert auf der gleichen Idee wie der Algorithmus für HACL. Die Geraden werden durch maximal kollineare Teilmengen gebildet. Dort wird gesagt, dass möglicherweise Geraden auch durch nur einen Punkt gehen können und deshalb werden die Punkte als degenerierte Geraden zur Kandidatenmenge hinzugefügt. Das Hauptproblem bei dem Algorithmus ist, dass aus BIPARTITE VERTEX COVER nun ein ein SET COVER-Problem geworden ist, dass sich nur logarithmisch approximieren lässt. SET COVER kann $\log k$-approximiert werden, wenn k die maximale Kardinalität einer Menge bezeichnet. Damit erhält man dann diesen Approximationsfaktor.

Minimum Link Hamilton Path [SBPL]

[KKM94] sieht sich das Problem an, allerdings unter der zusätzlichen Voraussetzung, dass jeder Knoten auch tatsächlich nur genau einmal überdeckt wird (auch keine Überdeckungen durch "lange Kanten" erlaubt).

2 Theoretische Untersuchungen

2.1 Ganzzahlige lineare Programmierung

In diesem Abschnitt wird eine kurze Einführung in die ganzzahlige lineare Programmierung gegeben.

Problem 7. *Integer Linear Programming (ILP)*

Instanz: $A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m, c \in \mathbb{Q}^n$
Gesucht: $x \in \mathbb{Z}^n, x = \arg\min\{c^\top x \mid Ax \leqslant b, x \in \mathbb{Z}^n\}$

Die Aufgabe bei der linearen Programmierung besteht darin, eine linear-affine Zielfunktion in einer polyedrischen Menge zu maximieren oder zu minimieren. Auf diese Art und Weise lassen sich manche Probleme der kombinatorischen Optimierung mit einer gemeinsamen Methode in Polynomialzeit lösen. Doch ist auch für viele Probleme keine Formulierung als lineares Programm bekannt.

Ein allgemeinerer Ansatz, für den aber im Allgemeinen keine polynomialzeitliche Lösungsmethode verfügbar ist, besteht in der der ganzzahligen linearen Programmierung (Integer Linear Programming, ILP). Die Ausgangskonfiguration ist ähnlich wie bei der linearen Programmierung, doch wird die Ganzzahligkeit der Lösung zusätzlich gefordert (die Lösungsmenge ist also der Schnitt eines Polyeders im n-dimensionalen Raum mit $\mathbb{Z}^n$).

Bei der linearen Programmierung ist die zulässige Menge als eine Menge im $\mathbb{R}^n$ der Form $\{x \mid Ax \leqslant b\}$ mit einer $\mathbb{Q}^{m \times n}$-Matrix A gegeben. Diese Mengenbeschreibung lässt sich als ein endlicher Schnitt von abgeschlossenen Halbräumen verstehen, was sofort zur Konvexität der betrachteten Menge führt.

Die zulässigen Lösungen der ganzzahligen Programmierung sind nun die in dem beschriebenen Polyeder liegenden ganzzahligen Punkte. Die Lösungsmenge ist also nur implizit gegeben. Gesucht wird ein Punkt, bei dem die lineare Zielfunktion je nach Formulierung ihr Maximum oder ihr Minimum annimmt.

Zunächst kann man feststellen, dass das Problem der ganzzahligen linearen Programmierung in $\mathcal{NP}$ liegt. Wird also eine Lösung angegeben, kann man ihre Korrektheit in Polynomialzeit überprüfen, indem man für jede Variable die Ganzzahligkeit, für jede Nebenbedingung die Erfüllung prüft und die Kosten in Linearzeit berechnet.

Liegen in dem Polyeder nur Punkte mit Koordinaten in $\{0, 1\}$, kann man 0 als FALSE und 1 als TRUE verstehen und auf diese Weise von SATISFIABLITY reduzieren. Man erhält:

Theorem 2.1. *[GJ79] INTEGER LINEAR PROGRAMMING [ILP]ist $\mathcal{NP}$-vollständig.*

Schwierig wird das Problem durch die implizite Beschreibung der Punkte Lösungsmenge, die im Inneren des Polyeders liegen. Wäre das Polyeder das kleinste, welches diese ganzzahligen Punkte alle enthält, so wäre das Problem einfach: Der optimale Zielfunktionswert würde stets am Rand angenommen und man könnte das Problem in Polynomialzeit als lineares Programm mit der Ellipsoid-Methode lösen.

2.2 Problemmodellierung mit ILP

Im vorherigen Kapitel wurde eine Klassifikation von Überdeckungsproblemen mit Abbiegekosten angegeben, die in der Literatur vor allem unabhängig voneinander betrachtet wurden. In diesem Abschnitt wird aufgezeigt, dass die klassifizierten Probleme als ganzzahlige Programme, die im letzten Abschnitt eingeführt wurden, formuliert werden können und auf diese Weise für kleinere Instanzen optimal, für größere immerhin mit mit einer Güteabschätzung gelöst werden können. Zunächst muss aber auf Kodierungsprobleme bei den Winkeln eingangen werden.

Kodierbarkeit der Winkel Selbst bei der Annahme, dass alle Punkte ganzzahlige Koordinaten haben, treten bei den im vorherigen Kapitel klassifizierten Probleme teilweise dennoch Abbiegewinkel auf, die auch durch π geteilt keine rationale Zahl ergeben. Mit diesem Problem muss man sich bei der Aufstellung des ganzzahligen linearen Programms auseinandersetzen.

Ein ähnliches Problem tritt bei der Formulierung des klassischen Traveling Salesman Problems auf. So wird bei Komplexitätsuntersuchungen in der algorithmischen Geometrie in den entsprechenden Fällen von einem Berechnungsmodell ausgegangen, bei dem **reelle** Zahlen in einer einzigen Speicherzelle, also mit konstantem Platz, gespeichert werden können und sich die Unit-Cost-Operationen auf ebendiese reellen Zahlen beziehen. Wie man das Problem in einem Rechnermodell mit endlicher Präzision angeht, ist offen. Man kann durch Skalierung annehmen, dass die Knoten ganzzahlige Koordinaten haben, dann sind die Längen als Wurzeln von Quadraten schreibbar. Der Kern der Schwierigkeiten ist folgendes Problem:

Problem 8 (Radikalsummenproblem)**.**

Instanz: $c_i \in \mathbb{Q}, q_i \in \mathbb{Q}, d_i \in \mathbb{N}^+, i = 1, \ldots, k$

Frage: Ist $\sum_{i=1}^{k} c_i \sqrt[d_i]{q_i} > 0$?

Die Zugehörigkeit des Radikalsummenproblems zu $\mathcal{NP}$ ist offen [Blo91]. Bei der Kodierung der Winkel treten ebenso irrationale Zahlen auf, so dass dieselbe Problematik auch hier gilt. Zur Vereinfachung wird daher im Rest der Arbeit angenommen, dass bei Problemen, in denen auch nichtachsenparallele Knicke erlaubt sind, zur Eingabe der Instanz auch sämtliche Winkel gehören. Weiterhin wird angenommen, dass diese alle rational sind (die entsprechenden Längen können durch eine hinreichend gute rationale Approximation in einem Preprocessing-Schritt er-

mittelt werden). Nur in diesem vereinfachten Setting konnte die Lösbarkeit durch ganzzahlige Programmierung erzielt werden.

Für Probleme, bei denen Knicke in beliebigen Punkten erlaubt sind, werden weitere Daten in der Eingabe benötigt. Dazu wird definiert:

Definition 2.1. *Sei* $P \subset \mathbb{R}^2$ *eine endliche Punktmenge. Sei* $G(P)$ *die Menge aller Geraden, die von zwei Punkten in* P *aufgespannt werden.* $G(P)$ *heiße der* ***Geradenspann*** *von* P.

Die Geraden aus dem Geradenspann definieren wiederum Punkte durch ihre Schnittpunkte. Die Menge aller Schnittpunkte von Geraden aus $G(P)$ *heiße die* ***Schnittergänzung*** *von* P.

Bei Problemen, bei denen Knicke auch außerhalb der Knoten möglich sind, seien alle Punkte der Schnittergänzung in der Eingabe gegeben (aus der Rationalität der ursprünglichen Punkte folgt die Rationalität der Punkte in der Schnittergänzung). Ebenso seien die Winkel wie oben beschrieben approximiert Teil der Eingabe.

Die Formulierung Sei $V \subset \mathbb{R}^2$ endlich und $G = (V, E)$ der Kompatibilitätsgraph auf V. Die Knoten seien mit einer beliebigen Nummerierung als $v_1, \ldots, v_n$ bezeichnet. Es sollen nun die Variablen des aufzustellenden ganzzahligen Programms erklärt werden. Jeder Variable entspricht hierbei ein Knick, das heißt, einer Auswahl von zwei Geradenstücken an einem Knoten. Dabei wird unterschieden, ob es sich um ein Partitionierungs- oder ein Überdeckungsproblem handelt. Bei der Definition der Partitionierungsprobleme im letzten Abschnitt wurden nämlich Knicke zu einem Knoten und direkt wieder zurück verboten, so dass im Partitionierungsfall einige Knicke verboten sind und die zugehörigen Variablen nicht im ganzzahligen Programm auftauchen.

Ein Tripel (i, j, k) heiße bezüglich E **überdeckungszulässig**, wenn $(j, i), (i, k) \in E$ und $j \leqslant k$. Das Tripel (i, j, k) heiße **partitionszulässig**, falls (i, j, k) überdeckungszulässig ist und $j < k$ gilt. Wie der Name schon sagt, werden über die überdeckungszulässigen Tripel die Variablen für die Überdeckungsprobleme und über die partitionszulässigen Tripel die Variablen für die Partitionierungsprobleme definiert. Die Winkelkoeffizienten w_{ijk} sind die Abbiegekosten des Tripels (j, i, k). Der erste Index der Koeffizienten gibt stets an, an welchem Knoten der Knick stattfindet.

Die Probleme aus dem vorherigen Kapitel mit der Klassifikation NK, bei denen Kreisfamilien gesucht werden und die Knicke stets in den Knoten stattfinden, können damit schon als ganzzahliges Programm formuliert werden. Dazu bezeichne $\mathcal{U}$ je nach Problem die Menge der partitionszulässigen bzw. die Menge der überdeckungszulässigen Tripel und c_{ijk} je nach Problem die Winkel- beziehungsweise die Knickkoeffizienten.

Setze rel :="$\geqslant$", falls das Problem ein Überdeckungsproblem ist, sonst rel :="$=$".

Die Graderhaltungsbedingungen werden nur für $j > i$ gebraucht. Für $i < j$ wären die Bedingungen redundant, es wären die gleichen mit umgekehrtem Vorzeichen. Das liegt an der Struktur: Die Anzahl der Knicke vom Knoten j zum Knoten i wird von der Anzahl der Knicke vom Knoten i zum Knoten j abgezogen, die Differenz soll 0 ergeben. Die Bedingung ist auf natürliche Weise symmetrisch und wird daher nur für eine der beiden Richtungen benötigt.

minimize	$\sum_{(i,j,k)\in\mathcal{U}} c_{ijk}x_{ijk}$	
subject to		
(rel i)	$\sum_{(i,j,k)\in\mathcal{U}} x_{ijk}$ rel 1	$\forall i \in V$
(degree compat ij)	$\sum_{(i,j,k)\in\mathcal{U}} x_{ijk} + \sum_{(i,k,j)\in\mathcal{U}} x_{ikj} -$	
	$\sum_{(j,i,k)\in\mathcal{U}} x_{jik} - \sum_{(j,k,i)\in\mathcal{U}} x_{jki} = 0$	$\forall (i,j) \in E, j > i$
	$x_{ijk} \in \mathbb{N}$	

Tabelle 2.1: IP für die Suche von Kreisfamilien mit Knicken in Knoten(NK)

Genauso erklärt sich bei den Variablen die Einschränkung, dass $k > j$ sein soll: Jedem Knick entsprächen sonst zwei Variablen und man müsste noch Gleichheitsbedingungen für die entsprechenden Variablenpaare einführen.

Es wird nun erklärt, wie man Probleme mit der Bedingung H statt K formulieren kann, bei der also ein Spannkreis gesucht wird statt einer Familie von Kreisen.

Aus dem Versuch, das Traveling Salesman Problem durch ein ganzzahliges Programm zu formulieren, kennt man die Subtour-Eliminations-Ungleichungen:

Theorem 2.2. *[KV00] Die Inzidenzvektoren von Touren im K_n sind genau die ganzzahligen Vektoren $x \in \mathbb{Z}^{\binom{n}{2}}$ mit*

$$0 \leqslant x_e \leqslant 1, \qquad e \in E(K_n) \tag{2.1}$$

$$\sum_{e\in\delta(v)} x_e = 2, \qquad v \in V(K_n) \tag{2.2}$$

$$\sum_{e\in\delta(X)} x_e \geqslant 2, \qquad \emptyset \neq X \subset V(K_n) \tag{2.3}$$

Dabei bezeichnet man die Ungleichungen 2.3 als Subtour-Eliminations-Ungleichungen. Ein ähnliches Resultat gilt für überdeckende Kreise. Der Inzidenzvektor gebe in diesem Fall nicht nur an, ob eine Kante benutzt wird (1) oder nicht (0), sondern auch die Kardinalität der Benutzung.

Theorem 2.3. *Die Inzidenzvektoren von Kreisen im K_n, die alle Knoten überdecken, sind genau die ganzzahligen Vektoren $x \in \mathbb{Z}^{\binom{n}{2}}$ mit*

$$\sum_{e\in\delta(X)} x_e \geqslant 2, \qquad \emptyset \neq X \subset V(K_n) \tag{2.4}$$

$$\sum_{e\in\delta(v)} x_e = 2 \cdot k, \qquad v \in V(K_n), k \in \mathbb{N}^+ \tag{2.5}$$

$$x_e \in \mathbb{N} \qquad \forall e \in E \tag{2.6}$$

Beweis. Sei x der Inzidenzvektor eines überdeckenden Kreises. G' entstehe aus G, indem jede Kante e von G durch x_e viele Kopien ihrer selbst ersetzt wird; G' ist also im Allgemeinen kein einfacher Graph. Aufgrund von Bedingung 2.5 hat jeder Knoten in G' geraden Grad. Daher ist G' eulersch. Wegen Bedingung 2.4 ist G' zusammenhängend. Sei T eine Euler-Tour von G'. Diese induziert eine Tour in G, indem die Benutzung einer Kantenkopie wieder durch die Benutzung der ursprünglichen Kante ersetzt wird. Damit erhält man einen Kreis, der jede Kante so oft benutzt, wie es im Inzidenzvektor vorgegeben ist.

Sei nun C ein Kreis. Da C ein Kreis ist, der alle Knoten überdeckt, muss jede echte, nichtleere Teilmenge von V betreten und wieder verlassen werden, daraus folgt Bedingung 2.4. Durch Anwendung dieses Arguments auf einelementige Teilmengen erhält man Bedingung 2.5. Damit folgt die Behauptung. □

Da in der IP-Formulierung oben keine Inzidenzvektoren benutzt werden, müssen die Nebenbedingungen angepasst werden. Man kann folgende Nebenbedingungen zu dem ganzzahligen Programm hinzugefügen:

$$\sum_{\substack{(i,j,k)\in\mathcal{U}\\ i\in X, j\notin X}} x_{ijk} + \sum_{\substack{(i,j,k)\in\mathcal{U}\\ i\in X, k\notin X}} x_{ijk} \geqslant 2, \qquad \emptyset \neq X \subset V(K_n) \tag{2.7}$$

Die Nebenbedingungen bedeuten also, dass aus jeder echten Knotenteilmenge mindestens zwei Knicke hinausführen müssen. Diese Bedingung impliziert Bedingung 2.4. Bedingung 2.5, dass jeder Knoten geraden Grad hat, wird durch die Graderhaltungsbedingung in obiger IP-Formulierung sichergestellt. Die Ganzzahligkeit der Variablen ist sowieso gefordert. Damit können also Probleme gelöst werden, die als Lösung überdeckende oder partitionierende Kreise verlangen.

Für die Spannpfadprobleme wird der Kompatibilitätsgraph modifiziert. Man fügt einen Superknoten hinzu, der Verbindungen zu allen Knoten hat und dessen Kostenkoeffizienten stets gleich 0 sind. Dadurch kann man diesen Fall auf den vorherigen Fall reduzieren.

Es soll nun gezeigt werden, dass auf ähnliche Weise auch die Probleme mit beliebigen Knicken gehandhabt werden können. Um nicht immer wieder auf triviale Fälle eingehen zu müssen, wird im Folgenden angenommen, dass die Punkte in der Punktmenge P nicht alle auf einer Geraden liegen. Dies impliziert einerseits, dass es mindestens 3 Punkte gibt, andererseits, dass jeder der Punkte in P in der Schnittergänzung von P liegt.

Theorem 2.4. *Bei Problemen mit beliebigen Knicken (B) und Knickkosten (L) kann auf Knicke verzichtet werden, deren Basispunkt nicht in der Schnittergänzung liegt oder deren Knickrichtungen nicht beide im Geradenspann liegen. Dadurch wird der optimale Zielfunktionswert nicht schlechter.*

Beweis. Zunächst soll gezeigt werden, dass auf Knicke außerhalb der Schnittergänzung verzichtet werden kann. Betrachte dazu einen Kreis C mit einer beliebigen, fest gewählten Orientierung. Sei k ein Knick auf diesem Kreis, der nicht in der Schnittergänzung liegt. Der Fall, dass ein Pfad gesucht wird und hinter k nur ein Knoten liegt, ist trivial. Es wird also angenommen, dass hinter k noch zwei Knoten folgen. Ebenso wird angenommen, dass vor k zwei Knoten liegen.

Seien w, x die Knoten vor k und x, y die Knoten hinter k. Da k kein Punkt aus der Schnittergänzung ist, liegt k insbesondere nicht sowohl auf der von w und x aufgespannten Geraden als

auch der von y und z aufgespannten Geraden, es sei denn, w, x, k, y und z liegen alle auf einer Geraden, dann gibt es aber in k keinen Knick.

Liegt k auf keiner der beiden Geraden, kann offenbar auf k verzichtet werden. Es wird nun angenommen, dass k zumindestens auf einer der beiden Geraden liegt, ohne Einschränkung liege k auf der von y und z aufgespannten Geraden(sonst wechsle die Orientierung des Kreises).

Nun kann man zwei Fälle unterscheiden. Der erste Fall besteht darin, dass im Knoten x ein Knick erfolgt in Richtung von k. Da k auf yz liegt, hat die Überdeckung dieser vier Knoten also 2 Knicke gekostet. Dann kann man aber auch durch Knicke in x und y überdecken und benötigt auch nicht mehr Knicke.

Es kann also kein Knick am Knoten x in Richtung k erfolgen. Da k nicht auf wx liegt, muss woanders ein Knick erfolgen, dieser Punkt sei mit t bezeichnet. Die Knicke erfolgen also nun in t und k. Auch hier kann man alternativ durch Knicke in x und y ersetzen und erhält die Behauptung.

Nun zum zweiten Fall.

Es liege also eine der beiden Richtungen nicht im Geradenspann. Der Knick erfolge beim Punkt k. Insbesondere kann k dann nicht in P liegen.

Die beiden Knoten vor k seien mit w, x, die danach mit y, z bezeichnet. Liegt k nicht auf der von w und x aufgespannten Geraden, so kann ohne Nachteil direkt in x geknickt werden, da nicht durch den Knick in k eine Richtung eingeschlagen wird, auf der sowohl y als auch z liegen (nach Annahme, denn sonst lägen beide Richtungen im Geradenspann).

k muss also auf der von w und x aufgespannten Geraden liegen. Da aber wie eben argumentiert z nicht auf der von k und y aufgespannten Geraden liegt, kann ohne Nachteil der Knick in k durch einen Knick in x ersetzt werden. Damit folgt die Behauptung. □

Für Winkelkosten statt Knickkosten gilt dieselbe Aussage.

2.2.1 LP-Analyse für die Suche nach Kreisfamilien mit Knicken in Knoten

Bei der Analyse von ganzzahligen Programmen sowie zum Finden von unteren Schranken wird die LP-Relaxation betrachtet. Diese entsteht, indem die ganzzahligen Variablen durch stetige ersetzt werden, die LP-Relaxation ist also ein lineares Programm. Der optimale Zielfunktionswert der LP-Relaxation ist (bei Minimierungsproblemen) stets eine untere Schranke an den optimalen Zielfunktionswert des IPs. Dies folgt auch direkt daraus, dass die ganzzahligen Lösungen für die LP-Relaxation auch zulässig sind.

Das duale Problem Gemäß dem Dualitätssatz der linearen Programmierung hat das lineare Programm den gleichen Zielfunktionswert wie sein duales Programm, so es denn mit einem endlichen Zielfunktionswert lösbar ist. Daher hängt das Programm eng mit seinem dualen zusammen:

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j,k)\in\mathcal{U}} c_{ijk}x_{ijk} & \\
\text{subject to} & & \\
\text{(rel i)} & \displaystyle\sum_{(i,j,k)\in\mathcal{U}} x_{ijk} \text{ rel } 1 & \forall i \in V \\
\text{(degree compat ij)} & \displaystyle\sum_{(i,j,k)\in\mathcal{U}} x_{ijk} + \sum_{(i,k,j)\in\mathcal{U}} x_{ikj} - & \\
 & \displaystyle\sum_{(j,i,k)\in\mathcal{U}} x_{jik} - \sum_{(j,k,i)\in\mathcal{U}} x_{jki} = 0 & \forall (i,j) \in E, j > i \\
 & x_{ijk} \geqslant 0 &
\end{array}
$$

Abbildung 2.1: LP-Relaxation des IPs für die Suche von Kreisfamilien[NK]

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{i\in V} \pi_i & \\
\text{subject to} & & \\
 & \pi_i + \pi_{ij} + \pi_{ik} \leqslant c_{ijk} & \forall (i,j,k) \in \mathcal{U}, i < j, i < k \\
 & \pi_i + \pi_{ij} - \pi_{ki} \leqslant c_{ijk} & \forall (i,j,k) \in \mathcal{U}, i < j, i > k \\
 & \pi_i + \pi_{ik} - \pi_{ji} \leqslant c_{ijk} & \forall (i,j,k) \in \mathcal{U}, i > j, i < k \\
 & \pi_i - \pi_{ji} - \pi_{ki} \leqslant c_{ijk} & \forall (i,j,k) \in \mathcal{U}, i > j, i > k \\
 & \pi_i \geqslant 0 & \text{wenn rel } =\text{“}\geqslant\text{”}
\end{array}
$$

Abbildung 2.2: Duales Problem der LP-Relaxation des IPs für die Suche von Kreisfamilien

Die Variablen des dualen Programms lassen sich mit dem ursprünglichen Graphen identifizieren, da es einerseits die Überdeckungs- bzw. Partitionierungsbedingungen für jeden Knoten gibt und andererseits die Graderhaltungsbedingungen für jede Kante. Für jede Kante $e = (u, v)$ des ursprünglichen Graphen erhält man hier eine Variable π_{ij} mit $i < j$.

Jeder der Bedingungen im dualen entspricht ein Knick der Form (i, j, k), wobei wie üblich der erste Index den Knoten bezeichne, über den der Knick verläuft. Sieht man sich die Variablenvorzeichen des dualen Programms näher an, so kann man beobachten, dass π_{ij} und π_{ik} stets positiv, π_{ji} und π_{ki} stets negativ signiert sind.

Sei C ein Kreis. Betrachtet man die Abfolge von Knicken in C, so tritt durch die Vorzeichenwahl oben jede Kante genauso oft positiv wie negativ auf.

Lemma 2.5. *Sei* (i, j, k) *ein Knick. Die Vorzeichen sind so gewählt, dass für* $i < \ell, \ell \in \{j, k\}$ *stets die Kante positiv, für* $i > \ell$ *die Kante stets negativ gezählt wird. Da jede Kante auf einem Kreis zu zwei Knicken gehört und stets einmal eine, einmal die andere Bedingung erfüllt, fallen bei Summation*

über einen Kreis die Kantenvariablen heraus.

Ist G zum Beispiel ein Gitter und sind die Knoten zeilenweise nummeriert (aufsteigend nach rechts innerhalb einer Zeile), so führt die Vorzeichenbelegung dazu, dass Kanten, die an einem Knoten einen N oder *W*-Anteil bilden, negativ gezählt werden, während Kanten, die einen S oder E-Anteil bilden, positiv gezählt werden. Summiert man also die Bedingungen entlang eines Kreises, bleiben auf der linken Seite der Ungleichung nur die Summe der Knotenvariablen, auf der rechten Seite genau die Anzahl der Knicke übrig.

Insgesamt kann man also festhalten, dass für jede Lösung des dualen Programms die Bedingung gilt, dass entlang eines Kreises die Summe der Knotenvariablen höchstens so groß ist wie die Knickkosten des Kreises.

Ein LP zusammen mit seinem dualen wird auch als **primal-duales Paar** bezeichnet. Sei x eine primale, y eine duale Lösung. Sind beide Lösungen optimale Lösungen der entsprechenden Programme, sagt man, (x, y) sei ein **optimales primal-duales Paar**. Das ist genau dann der Fall, wenn die Bedingungen vom komplementären Schlupf (*Complementary Slackness*)-Bedingungen erfüllt sind.

Lemma 2.6. *[KV00] Seien* $\min\{cx : Ax = b, x \geqslant 0\}$ *und* $\max\{yb : Ay \leqslant c\}$ *ein primal-duales Paar. Dann sind folgende Aussagen äquivalent:*

1. x *und* y *sind Optimallösungen.*
2. $cx = yb$
3. $x(c - Ay) = 0$

Für Partitionierungsprobleme gilt demnach: Für die Optimalität einer primalen Lösung werden aufgrund der Gleichheitsstruktur des primalen Programms nur die primalen Komplementärschlupfbedingungen benötigt. Diese besagen, dass für jede Nichtnullvariable im primalen die zugehörige duale Ungleichung mit Gleichheit erfüllt sein muss (Bedingung 3), man sagt, es dürfe kein „komplementärer Schlupf“ existieren.

Ist insbesondere ein Kreis im primalen vollständig gewählt, müssen bei Optimalität im dualen alle zugehörigen Bedingungen mit Gleichheit erfüllt sein. Ist also eine dual optimale Lösung gegeben, so dürfen bei der Konstruktion einer optimalen primalen Lösung nur Kreise ausgewählt werden, deren Knickkosten gleich der Summe der dualen Knotenvariablen sind. Da das duale lineare Programm in Polynomialzeit konstruiert und gelöst werden kann, können auf diese Weise in einem Preprocessing-Step bei der Lösung des primalen Problems bestimmte Kreise von vornherein ausgeschlossen werden.

2.3 Generische Lösungsversuche für MTCP

In diesem Abschnitt werden Ansätze gemacht, das MTCP-Problem mit allgemeinen Lösungsmethoden wie ganzzahliger linearer oder ganzzahliger quadratischer Programmierung zu lösen.

Definition 2.7 (MTCP-Problem)**.** *Gegeben sei ein Gitter* $G = (V, E)$*. Gesucht ist eine Menge von Kreisen* $\mathcal{C} := \{C_1, C_2, \ldots, C_k\}$ *mit* $\#\{C \in \mathcal{C} \mid v \in C\} = 1 \; \forall v \in V$*. Die Knickkosten von* $\mathcal{C}$ *sind dabei zu minimieren.*

Repräsentationsmöglichkeiten der Lösungen

Für die Lösungen des MTCP-Problem werden drei Arten der Darstellung benutzt. 2-Faktoren lassen sich einerseits als eine Menge Kanten verstehen. Ebenso kann man sie aber als eine Menge von knotendisjunkten Kreisen verstehen. Eine dritte Möglichkeit ergibt sich durch die Einführung des Begriffs des Knicks. In einem 2-Faktor hat jeder Knoten genau zwei Nachbarn. Diese beiden Kanten lassen sich zusammengefasst als ein Knick verstehen. Es muss dabei eine Graderhaltung gelten, geht ein Knick vom Knoten i zum Knoten j, so muss der Knick am Knoten j die Kante ji als eine seiner beiden Kanten benutzen. Die letzte Art der Darstellung lässt sich benutzen, um das Problem als ganzzahliges Programm zu notieren (s.o.). Bei der Formulierung über Kreise wären exponentiell viele Variablen erforderlich, diese Formulierung könnte in einem Ansatz mittels Spaltengenerierung münden.

2.3.1 LP-Relaxation

Wie man in Kapitel 3 nachlesen kann, wurde bei der Untersuchung der LP-Relaxation des MTCP-Problems festgestellt, dass das Simplexverfahren für dieses Problem bei allen überprüften Instanzen eine ganzzahlige Lösung liefert.

Es ergeben sich daraus verschiedene Theorien, die es zu verfolgen lohnt. Eine davon ist die Möglichkeit, dass das durch die Formulierung beschriebene Polytop ganzzahlig ist, also nur ganzzahlige Ecken besitzt.

Diese Vermutung konnte jedoch widerlegt werden: In Abbildung 2.3 ist eine zulässige Basislösung angegeben, die halbzahlig ist. Dabei werden alle eingezeichneten Knicke mit Wert $\frac{1}{2}$ benutzt.

Lemma 2.8. *Die zulässige Lösung in Abbildung 2.3 ist eine Basislösung, also keine Linearkombination von ganzzahligen Lösungen.*

Beweis. In diesem Beweis wird die Schachbrettnummerierung benutzt, die Spalten werden also mit a-d, die Zeilen mit 1-4 bezeichnet. Links unten ist der Knoten a1. Desweiteren werden zur Richtungsangabe die englischen Himmelsrichtungen ESWN für East, West, South, North benutzt.

Der Beweis basiert hauptsächlich auf folgendem Argument: Nimmt man an, dass die abgebildete Lösung eine Linearkombination von ganzzahligen Lösungen ist, so können in jeder der ganzzahligen Lösungen nur die Knicke benutzt werden, die auch in der linearkombinierten Lösung benutzt werden. Trotzdem müssen alle Knoten partitioniert werden.

Zunächst wird der Knoten c2 betrachtet. Da der SE-Knick benutzt wird, muss er auch in einer der ganzzahligen Lösung benutzt werden. Diese soll mit L bezeichnet sein. Es wird nun die Existenz der Lösung L zum Widerspruch geführt.

Um den Graderhaltungsbedingungen zu genügen, muss in L im Knoten d2 der SW-Knick und im Knoten c1 der NE-Knick benutzt werden. Dann muss auch in d1 der NW-Knick benutzt werden, der orangene Kreis muss in L also vollständig enthalten sein.

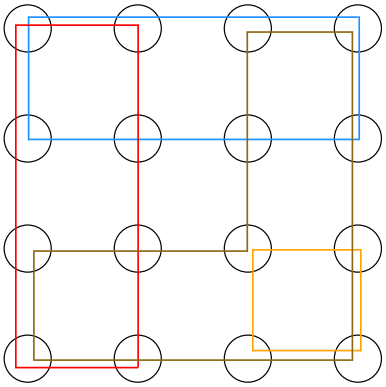

Abbildung 2.3: Eine zulässige Basislösung des MTCP-Polytops

Betrachte nun den Knoten b2. Da der Knoten c2 nicht mehr zur Verfügung steht, muss der NS-Knick benutzt werden. Damit ist aber auch schon die Knickauswahl in den Knoten b3, b4 und b1 durch die Graderhaltung vorgegeben. Es muss dann in a4 NE, in a3 NS, in a2 NS und in a1 NE benutzt werden.

Im Knoten c3 kann dann aber keiner der beiden Knicke benutzt werden, da es bei beiden nicht mehr möglich ist, die Graderhaltungsbedingungen zu erfüllen. ↯ □

Daher ist das Polytop im Allgemeinen nicht ganzzahlig. Eine weitere Möglichkeit wäre, dass die optimale Facette bezüglich der Kostenfunktion stets ganzzahlig ist oder auch, dass es sich schlicht um Zufall handelt. Diese Frage konnte jedoch nicht geklärt werden.

2.3.2 Ansatz über ganzzahlige quadratische Programmierung

Im letzten Abschnitt wurde eine Formulierung entlang der durch das allgemeine ganzzahlige Programm vorgezeichneten Linie vorgestellt. Ein anderer Ansatz, das Problem mit generischen Methoden zu behandeln, führt über die quadratische Programmierung.

Das lineare Programm beruhte auf der Formulierung der Kosten durch Knickvariablen. Ein Knick besteht jedoch aus zwei adjazenten Kanten. Die jetzt vorgestellte Formulierung kommt hingegen mit Kantenvariablen aus, wodurch sich die Graderhaltung, die vorher explizit gefordert werden musste, von allein ergibt.

Zunächst wird die Notation für die ganzzahlige quadratische Programmierung festgelegt.

Problem 9 (Integer Quadratic Programming(IQP))**.**

Instanz:	$n, m \in \mathbb{Z}^+$,$Q \in \mathbb{Q}^{n\times n}$, $d \in \mathbb{Q}^n$ $b \in \mathbb{Z}^m$, $A \in \mathbb{Z}^{m\times n}$
Gesucht:	$x = \arg\min\{f(x) \mid Ax \leqslant b, x \in \mathbb{Z}^n\}$
Zielfunktion:	$f(x) = x^\top Qx - d^\top x$

Der wesentliche Unterschied besteht also darin, dass sich die Zielfunktion aus einem quadratischen und einem linearen Teil zusammensetzt. Nun wird erklärt, wie das MTCP-Problem damit formuliert werden kann. Dazu sei

$$c_{ef} := \begin{cases} 1 & e \text{ und } f \text{ stehen aufeinander senkrecht} \\ 0 & \text{sonst} \end{cases}$$

Der Koeffizient c_{ef} gibt also an, ob durch die Auswahl von zwei adjazenten Kanten Knickkosten entstehen oder nicht. Damit kann das Problem nun formuliert werden:

minimize	$\sum_{i \in V} \sum_{e \in \delta(v)} \sum_{\substack{f \in \delta(v) \\ f \neq e}} x_e \cdot x_f$
subject to	
	$\sum_{e \in \delta(v)} x_e = 2 \qquad \forall e \in E$
	$x_e \in \{0,1\}$

Tabelle 2.2: Formulierung des MTCP-Problems als IQP

Bei dieser Art der Formulierung verliert man die Übertragbarkeit auf Überdeckungsprobleme, da sich deren Lösungen nicht ohne weiteres als Mengen von Kanten beschreiben lassen. So würden durch die verwendete Zielfunktion alle Paare von ausgewählten adjazenten Kanten, die nicht kollinear verlaufen, zur Zielfunktion beitragen. Bei der Auswahl von allen vier Kanten an einem Knoten würden beispielsweise Kosten von 4 entstehen. Im Partitionierungsfall besteht jedoch kein Problem, da jede Kante höchstens einmal benutzt wird.

2.3.3 Komplexität

Im letzten Abschnitt wurde eine Formulierung des Problems als quadratisches Programm mit binären Entscheidungsvariablen vorgestellt. In diesem Abschnitt werden die bekannten Resultate zur Komplexität solcher Programme vorgestellt.

Zunächst wird das Convex-Cost-Flow-Problem vorgestellt, das in [AMO93] analysiert wird. Die genaue Definition dieses Problems kann man in Abbildung 2.4 sehen.

Im Convex-Cost-Flow-Problem ist die Zielfunktion **separabel**, es treten also keine Produkte verschiedener Variablen auf.

Sei $S(n, m, C)$ die Laufzeit für die Suche nach einem kürzesten Weg auf einem Graphen mit n Knoten, m Kanten und nichtnegativer Kostenfunktion C. Dann gilt:

minimize	$\sum_{(i,j)\in E} c_{ij}(x_{ij})$	c_{ij} konvexe Funktionen
subject to		$c_{ij} : \mathbb{Q}_0^+ \to \mathbb{Q}$
	$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = b(i)$	$\forall i \in V$
	$0 \leqslant x_{ij} \leqslant u_{ij}$	$\forall (i,j) \in E$
	$x_{ij} \in \{0,1\}$	

Abbildung 2.4: Convex Cost Flow Problem

Theorem 2.9. *[AMO93] Der Kapazitätsskalierungsalgorithmus bestimmt einen optimalen ganzzahligen Fluss für ein Convex-Cost-Flow-Problem in Zeit* $\mathcal{O}((m \log U)S(n, m, C))$ *Zeit, wobei* U *die Zahl mit der größten Kodierungslänge unter den* $b(i)$ *und unter den oberen Schranken* u_{ij} *bezeichnet.*

Das Resultat lässt sich für das MTCP nicht verwenden, weil die dabei entstehende Zielfunktion aus der Auswahl zweier adjazenter Kanten entsteht, was durch die Multiplikation dieser Variablen ausgedrückt wird. Wir betrachten also von nun an Resultate, die für nichtseparable Zielfunktionen gefunden wurden. Auch in diesem Fall gibt es effizient lösbare Fälle.

So wird in [HSS92] ein Scheduling-Problem als nichtseparables IQP formuliert und als solches in in polynomialer Zeit gelöst. Dazu wird statt dem Ausgangsproblem eine besondere fraktionale Relaxation gelöst, aus der sich eine Optimallösung des ursprünglichen Problems rekonstruieren lässt.

Ein anderer lösbarer Fall liegt beispielsweise vor, wenn es sich um ein binäres, unbeschränktes Minimierungsproblem handelt, die Q−Matrix symmetrisch und auf der Diagonale gleich 0 ist und als Adjazenzmatrix aufgefasst einen serienparallelen Graphen beschreibt[Bar86]. Außerdem kann das unbeschränkte binäre Minimierungsproblem

$$\min_{x\in\{0,1\}^n} cx - x^\top Qx$$

mit $Q > 0$ als ein Max-Fluss-Problem gelöst werden [WN88].

Letztere Ergebnisse werden in [Bal95b] verallgemeinert. Um die Polynomialität des Problems zu erhalten, genügt es zu zeigen, dass die Matrix Q in der Beschreibung als quadratisches Programm durch $Q = L^\top DL$ mit einer Diagonalmatrix D und einer vollständig unimodularen Matrix $L \in \mathbb{Z}^{N\times n}$ entsteht, die die Eigenschaft hat, dass die Matrix

$$\begin{pmatrix} A & 0 \\ L & -E_N \\ -L & E_N \end{pmatrix}$$

ebenfalls vollständig unimodular ist oder zumindest eine polynomial beschränkte Kegelnorm (Def. 2.3) aufweist. Dabei bezeichnet E_N die Einheitsmatrix der Größe N.

Das Verfahren wird detaillierter im nächsten Abschnitt ausgeführt.

2.3.4 Konvexe quadratische ganzzahlige Programmierung

Dieser Abschnitt folgt [Bal95b].

Problem 10 (Unimodularly constrained Integer Quadratic Programming(UIQP)).

Instanz:	$n, m \in \mathbb{Z}^+$,$Q \in \mathbb{Q}^{n\times n}$, $d \in \mathbb{Q}^n$
	$b \in \mathbb{Z}^m$, $A \in \mathbb{Z}^{m\times n}$, A vollständig unimodular
Gesucht:	$x = \arg\min\{f(x) \mid Ax \leqslant b, x \in \mathbb{Z}^n\}$
Zielfunktion:	$f(x) = x^\top Qx - d^\top x$

Lemma 2.10. *[Mag59] Sei* $G = (V, E)$ *Graph und* $A = (a_{ij})$ *seine Inzidenzmatrix. Sei* $S \subseteq V$ *und* $(x_1, \ldots, x_n)$ *der charakteristische Vektor von* S*. Der von* S *induzierte Subgraph von* G *ist stabil genau dann, wenn*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j = 0 \tag{2.8}$$

Beweis. Sei zunächst S stabil. Aufgrund der Stabilität ist für $a_{ij} = 1$ stets $x_i = 0$ oder $x_j = 0$ und damit $x_ix_j = 0$.

Gelte nun 2.8. Weil $a_{ij}x_ix_j >= 0$, folgt $a_{ij}x_ix_j = 0 \forall i, j \in \{1, \ldots, n\}$. Seien $i, j \in V$, es ist zu zeigen, dass dann keine Kante zwischen i und j existiert. Aus $a_{ij}x_ix_j = 0$ folgt mit $x_i = x_j = 1$ aber $a_{ij} = 0$. □

Definition 2.2. *Sei* $\alpha(G)$ *die Knotenanzahl einer kardinalitätsmaximalen stabilen Menge in* G. $\alpha(G)$ *heißt die* ***Stabilitätszahl*** *oder* ***Unabhängigskeitszahl****.*

Für allgemeine Graphen ist es $\mathcal{NP}$-schwer, $\alpha(G)$ zu berechnen, denn damit kann die Entscheidungsvariante des $\mathcal{NP}$-schweren Problems `INDEPENDENT SET` entschieden werden. [GJ79]

Theorem 2.11. *[HR68] Unimodularly constrained Integer Quadratic Programming ist* $\mathcal{NP}$*-schwer.*

Beweis. Betrachte das Problem, 2.8 zu maximieren. Dann entspricht einer optimalen Lösung eine maximale stabile Menge.

Das folgende Problem ist dazu äquivalent:

Maximiere

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i - (n+1)\sum_{i=1}^{n}\sum_{i=1}^{n} a_{ij}x_ix_j \tag{2.9}$$

mit binären Variablen x_i.

Da die einzigen Nebenbedingungen die $\{0, 1\}$—Bedingungen sind, ist die Nebenbedingungsmatrix A vollständig unimodular. Auf diese Weise kann also $\alpha(G)$ berechnet werden. Damit folgt die Behauptung. □

Aufgrund der Schwere des Problems zieht man sich auf den Fall mit konvexer Zielfunktion zurück:

Problem 11 (Unimodularly constrained Convex Integer Quadratic Programming(UCIQP))**.**

Instanz:	$n, m \in \mathbb{Z}^+$,$Q \in \mathbb{Q}^{n\times n}$, $d \in \mathbb{Q}^n$, Q **positiv semidefinit** $b \in \mathbb{Z}^m$, $A \in \mathbb{Z}^{m\times n}$, A vollständig unimodular
Gesucht:	$x = \arg\min\{f(x) \mid Ax \leqslant b, x \in \mathbb{Z}^n\}$
Zielfunktion:	$f(x) = x^\top Qx - d^\top x$

Man kann zeigen: Die Zielfunktion g ist genau dann konvex, wenn Q positiv semidefinit ist.
Zur Lösung des UCIQP betrachtet man zunächst den Spezialfall mit separabler Zielfunktion:

Problem 12 (Unimodularly constrained Separable Convex Integer Quadratic Programming(USCIQP))**.**

Instanz:	$n, m, N \in \mathbb{Z}^+$, $L \in \mathbb{Z}^{N\times n}$, $D \in \mathbb{Q}^{N\times N}$, D diagonal, $d \in \mathbb{Q}^n$ $b \in \mathbb{Z}^m$, $A \in \mathbb{Z}^{m\times n}$, A vollständig unimodular
Gesucht:	$x = \arg\min\{g(x,y) \mid Ax \leqslant b, y = Lx, y \in \mathbb{Z}^N\}$
Zielfunktion:	$g(x,y) = y^\top Dy - d^\top x$, g separabel

Unter bestimmten Bedingungen kann das USCIQP in Polynomialzeit gelöst werden.
Sei dazu

$$\bar{A} = \begin{bmatrix} A & 0 \\ L & -I \\ -L & I \end{bmatrix}$$

Theorem 2.12. *Ist* $\bar{A}$ *vollständig unimodular und ist die zulässige Region* $Ax \leqslant b$ *beschränkt, so kann das Problem in einer Zeit polynomial in* n, m *und* $\log_2 \|b\|_\infty$ *gelöst werden.*

Die Aussage folgt aus Theorem 4.3 in [HS90]. Das Theorem bezieht sich auf den in [HS90] vorgestellten Algorithmus zur Lösung von Problemen dieser Art.

Ist b also zum Beispiel ein Vektor aus $\{0,1\}^n$, so ist das Problem stark polynomial lösbar.

Wir betrachten nun den Fall, dass $\bar{A}$ nicht vollständig unimodular ist. Auch hier wurde bereits ein polynomial lösbarer Spezialfall gefunden.

Zur Beschreibung wird zunächst die Kegelnorm von ganzzahligen Matrizen definiert.

Definition 2.3 (Kegel)**.** *Eine Menge* $X \subseteq \mathbb{R}^n$ *heißt ein* ***Kegel****, wenn*

$$X = \text{cone}(X) := \{\lambda_1 x_1 + \cdots + \lambda_k x_k \,|\{x_1, \ldots, x_k\} \subseteq X, \lambda_i \geqslant 0\}$$

Definition 2.4 (Matrixkegel)**.** *Sei* $A \in \mathbb{Z}^{m\times n}$, $S_- \uplus S_+ = \{1, \ldots, n\}, T_- \uplus T_+ = \{1, \ldots, m\}$.
Der zu diesen Teilmengen gehörige ***Kegel*** *sei definiert durch durch*

$$\begin{aligned} \mathfrak{C}(S_-, S_+, T_-, T_+, A) \quad := \quad & \{x \in \mathbb{R}^n \mid x_i \leqslant 0 \forall i \in S_-; x_i \geqslant 0 \forall i \in S_+; \\ & (Ax)_j \leqslant 0 \forall j \in T_-; (Ax)_j \geqslant 0 \forall j \in T_+\} \end{aligned}$$

Definition 2.5 (Erzeugendensystem eines Kegels)**.** *Eine endliche Teilmenge* $\{x_1, \ldots, x_k\}$ *eines Kegels* X *heißt ein* ***Erzeugendensystem*** *oder ein* ***Generatorsystem****, wenn*

$$X = \{\lambda_1 x_1 + \cdots + \lambda_k x_k \mid \lambda_1, \ldots, \lambda_k \geqslant 0\}.$$

Das Erzeugendensystem heißt ***ganzzahlig****, wenn alle darin enthaltenen Vektoren nur ganzzahlige Einträge haben.*

Definition 2.6 (Kegelnorm)**.** *Sei* C *durch endlich viele, ganzzahlige Vektoren erzeugt. Dann sei*

$$\|C\| := \min_U\{\max_{v \in U}\|u\| \mid U \text{ Erzeugendensystem von } C\}$$

Allgemein gilt, dass jeder Kegel, der als endlicher Schnitt von Halbräumen entsteht (jeder **polyhedrale** Kegel) durch endlich viele Vektoren erzeugt wird (siehe z.B. [Zie95]). Der oben definierte Matrixkegel hat in unserem Fall insbesondere auch ein ganzzahliges Erzeugendensystem, wie direkt aus der vollständigen Unimodularität der Matrix A folgt. Für rationale Matrizen folgt die Existenz eines rationalen Erzeugendensystems aus der Cramerschen Regel und dann kann man mit Skalieren ganzzahlig machen [Bal95a].

Damit kann nun die Laufzeit des bekannten Lösungsalgorithmus angegeben werden:

Definition 2.7. *Sei* $A \in \mathbb{Z}^{m \times n}$.

$$\delta(A) := \max_{S_-, S_+, T_-, T_+}\{\|(S_-, S_+, T_-, T_+)\| \mid S_- \uplus S_+ = \{1, \ldots, n\}, T_- \uplus T_+ = \{1, \ldots, m\}\}$$

Theorem 2.13. *Sei* $\bar{A} \in \{0, 1, -1\}^{m \times n}$, $\delta(\bar{A})$ *polynomial in* m *und* n *und* $\log_2 \|b\|_\infty$ *polynomial in* n *und* m*. Dann kann das Sub-Problem in stark polynomialer Zeit gelöst werden.*

Sei $\Delta(A)$ das Maximum der Absolutwerte der Subdeterminanten von A. Dann gilt folgendes Lemma:

Lemma 2.14. *Es gilt stets:* $\delta(A) \leqslant \Delta(A)$.

Für schwache Polynomialität würde es also ausreichen, wenn man zeigen könnte, dass die Subdeterminanten der Matrix $\bar{A}$ polynomial in n und m beschränkt sind. Dann kann das Problem mit dem Algorithmus von Hochbaum und Shanthikumar gelöst werden.

In bestimmten Spezialfällen kann $\delta(A)$ generell beschränkt werden. Für Matrizen mit höchstens zwei Nichtnulleinträgen pro Zeile folgt beispielweise $\delta(A) = 1$.

2.3.5 Anwendung der Resultate

In diesem Abschnitt werden Schritte dahin unternommen, die im vorherigen Abschnitt vorgestellten Resultate auf das MTCP-Problem anzuwenden und dabei möglichst eine polynomiale Lösung des Problems zu finden.

Diese Schritte werden anhand eines Beispiels illustriert. Sei G der kleinste zulässige Graph in L-Form für das MTCP-Problem auf 8 Knoten und 10 Kanten (Bild). Die Kanten seien zeilenweise nummeriert.

Die Zielfunktion für das quadratische Programm ist dann

$$x_4 \cdot (x_1 + x_2 + x_6 + x_7) + x_6 \cdot (x_3 + x_8 + x_9) + x_1 \cdot x_3 + x_5(x_2 + x_7) + x_9 \cdot (x_7 + x_{10}) + x_8 \cdot x_{10}.$$

Um einfacher eine symmetrische Matrix benutzen zu können, wird die Zielfunktion nun durch

$$2x_4 \cdot (x_1 + x_2 + x_6 + x_7) + 2x_6 \cdot (x_3 + x_8 + x_9) + 2x_1 \cdot x_3 + 2x_5(x_2 + x_7) + 2x_9 \cdot (x_7 + x_{10}) + 2x_8 \cdot x_{10}$$

ersetzt, was an der Optimalität von Lösungen nichts ändert.
Die Q-Matrix hat dann die Gestalt

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

In dieser Form ergibt sich ein Problem mit der Konvexität, denn diese Matrix ist nicht positiv semidefinit. Da die Matrix symmetrisch und reell ist, würde es genügen, für positive Semidefinitheit schwache Diagonaldominanz herbeizuführen.
Es gilt nämlich:

Theorem 2.15. *[Ger31][Kreissatz von Gerschgorin] Sei A komplexe $n \times n$-Matrix mit Einträgen a_{ij}. Sei*

$$r_i = \sum_{j \neq i} |a_{ij}|.$$

*Sei $B(a_{ii}, r_i)$ die abgeschlossene Kreisscheibe um a_{ii} in der komplexen Ebene (eine **Gerschgorin-Scheibe**). Dann gilt:*
Jeder Eigenwert von A liegt in mindestens einer der Gerschgorin-Scheiben.

Korollar 2.16. *Reelle symmetrische Matrizen mit schwacher Diagonaldominanz sind positiv semidefinit.*

Beweis. Reelle, symmetrische Matrizen haben nur reelle Eigenwerte. Aus der schwachen Diagonaldominanz folgt, dass die Gerschgorin-Scheiben alle in $\mathbb{R}_0^+$ vollständig enthalten sind. Damit sind alle Eigenwerte größer gleich 0 und es folgt die Behauptung. □

Gesucht ist nun also zunächst eine Matrix L, die vollständig unimodular ist und die Eigenschaft hat, dass mit einer geeigneten Matrix D $Q = L^T D L$ gilt. Die Matrix D wird hier nicht benötigt. Offenbar kann die Matrix Q nicht aus einer Matrix L durch $Q = L^T L$ hervorgehen, da Q nicht positiv semidefinit ist. Man kann aber eine Matrix L finden, die die Matrix Q zumindest abseits der Hauptdiagonalen erzeugt.

Die Matrix L hat soviele Zeilen wie es Knicke im Ausgangsgraphen gibt, die Kosten verursachen. Die Anzahl der Spalten entspricht der Anzahl der Kanten im Originalgraphen. Man könnte die Matrix als Knicke-Kanten-Inzidenzmatrix bezeichnen. Sie entsteht wie eine Inzidenzmatrix: Ein Eintrag ist gleich 1, wenn der Knick in Zeile i die Kante in Spalte j benutzt, sonst 0.

Im vorliegenden Fall ergibt sich also folgende Matrix:

$$L := \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Durch Multiplikation erhält man

$$L \cdot L^T = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 4 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 4 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \end{pmatrix}$$

Die Diagonaleinträge geben dabei für jede Kante an, an wieviel kostenden Knicken sie beteiligt ist.

Da die Anzahl der Kanten in jeder Lösung durch die Kreisstruktur vorgegeben ist (sie ist stets gleich der Anzahl der Knoten), verändern Nichtnulleinträge auf der Diagonalen die optimale Lösung nicht, solange sie alle gleich sind. Dann entsprechen sie nämlich nur der Addition der stets gleichen Konstanten auf den Zielfunktionswert.

Offenbar sind in der Matrix $L^T \cdot L$ die Diagonaleinträge aber nicht von gleichem Wert. In dieser Form wird eine Optimallösung Kanten vermeiden, die an vielen kostenden Knicken beteiligt sind.

Da nur binäre Werte zugelassen sind, lässt sich diese Abweichung durch den linearen Anteil wieder ausgleichen.

Bezeichne $\overline{A}$ die Matrix

$$\overline{A} = \begin{pmatrix} A & 0 \\ L & -I \\ -L & I \end{pmatrix}.$$

Dabei ist I die quadratische Einheitsmatrix der Größe N. Könnte man nun zeigen, dass $\delta(\overline{A})$ polynomial in n und m beschränkt ist, erhielte man direkt die Polynomialität des Problems durch Anwendung der Resultate aus [Bal95a] und Benutzung des Algorithmus von Hochbaum und Shantikumar in [HS90].

Um herauszufinden, ob die Matrix $\overline{A}$ aus dem letzten Abschnitt vollständig unimodular ist, kann man zunächst bei sehr kleinen Instanzen explizit alle Submatrizen generieren und überprüfen. Bei diesen Versuchen traten vollständig unimodulare Matrizen auf.

Um größere Beispiele untersuchen zu können, werden nun Netzwerkmatrizen eingeführt. Diese bilden die Grundlage des Algorithmus von Seymour zur polynomialen Erkennung von vollständig unimodularen Matrizen [Sey80].

2.3.6 Netzwerkmatrizen

Definition 2.8. *Sei* $G = (V, E)$ *ein gerichteter Graph und sei* $T = (V, E')$ *ein spannender Baum auf* G. *Sei* M $m \times n$*–Matrix mit* $m = |E'|$ *und* $n = |E|$. *Für* $e = (u, v) \in E$ *und* $e' \in E$ *sei*

$$M_{e',e} := \begin{cases} +1 & \text{der } (u,v)\text{-Pfad in } T \text{ benutzt } e' \text{ vorwärts} \\ -1 & \text{der } (u,v)\text{-Pfad in } T \text{ benutzt } e' \text{ rückwärts} \\ 0 & \text{der } (u,v)\text{-Pfad in } T \text{ benutzt } e' \text{ nicht} \end{cases}$$

Dann heißt M *eine* ***Netzwerkmatrix****.*

Da T spannend ist, muss G zusammenhängend sein. Jede Spalte der Matrix M identifiziert den in T eindeutigen Weg zwischen den Endpunkten der zu der Spalte gehörenden Graphkante.

Theorem 2.17. *[CCPS98] Netzwerkmatrizen sind vollständig unimodular.*

Könnte man also zeigen, dass es sich bei der Matrix $\overline{A}$ aus dem letzten Abschnitt um eine Netzwerkmatrix handelt, erhielte man die polynomiale Lösbarkeit des MTCP-Problems. Es wird nun zunächst ein polynomialer Algorithmus zur Erkennung von Netzwerkmatrizen beschrieben. Die Beschreibung folgt [Sch98].

In dem Fall, dass die Matrix pro Spalte nur höchstens eine 1 und eine -1 enthält (sonst 0), ist die Matrix genau dann vollständig unimodular, wenn sie eine Netzwerkmatrix ist. Dies kann mit dem Kriterium von Ghouila-Houri entschieden werden:

Theorem 2.18 (Ghouila-Houri-Kriterium)**.** *[GH62] Sei* M *Matrix mit allen Einträgen aus* $\{0, 1, -1\}$. *Dann ist* M *genau dann vollständig unimodular, wenn sich jede Familie von Spalten von* M *in zwei Teile zerlegen lässt, so dass die Summe der Spalten in einem Teil minus die Summe der Spalten in dem anderen Teil ein Vektor mit Einträgen ausschließlich in* $\{0, 1, -1\}$ *ist.*

Dazu definiert man einen Graphen G, die Knoten von G, $v_1, \dots, v_m$, seien mit den Zeilen von M identifiziert. Zwei Knoten i und j seien durch eine Kante verbunden, wenn es in M eine Spalte gibt, die Nichtnulleinträge in den Zeilen i und j mit dem gleichen Vorzeichen hat. Zusätzlich seien zwei Knoten durch einen Weg der Länge 2 (mit einem zusätzlichen neuen Knoten) verbunden, wenn es eine M eine Spalte mit zwei Nichtnulleinträgen in den Zeilen i und j hat, diesmal mit verschiedenen Vorzeichen. Dann gilt: Die Zerlegung aus dem Kriterium von Ghouila-Houri existiert genau dann, wenn der Graph G bipartit ist.

Man kann nun also annehmen, dass es mindestens eine Spalte mit mindestens drei Nichtnulleinträgen gibt. Für jedes $i \in \{1, \dots, m\}$ wird dann ein ungerichteter Graph $G_i = (V_i, E_i)$ mit der Knotenmenge $V_i\{1, \dots, m\} \setminus \{i\}$ definiert. Setze

$$E_i := \left\{\{k, \ell\} \in (\{1, \dots, m\} \setminus \{i\})^2 \mid \exists r \in \{1, \dots, n\} : M_{i,r} = 0 \wedge M_{k,r} \neq 0 \neq M_{\ell,r} \wedge k \neq \ell\right\}$$

Zwei Knoten k, ℓ sind also genau dann in E_i durch eine Kante verbunden, wenn es eine Spalte r gibt, die in der i-ten Zeile eine 0 und in der k-ten und ℓ-ten Zeile Nichtnulleinträge hat. Dann gilt:

Lemma 2.19. *Sei* M $m \times n$*-Matrix mit Einträgen aus* $\{0, 1, -1\}$. M *habe mindestens eine Spalte mit mindestens drei Nichtnulleinträgen. Dann gilt:*

Ist M *eine Netzwerkmatrix, so ist einer der Graphen* G_i *unzusammenhängend.*

Beweis. Sei M Netzwerkmatrix, M komme vom Graphen G und dem Baum T her. Dann hat T einen (ungerichteten) Pfad der Länge 3 (aufgrund der Spalte mit mindestens drei Nichtnulleinträgen und der Definition von Netzwerkmatrizen). Sei e_j die mittlere Kante eines solchen Pfades. Da T ein Baum ist, gibt es zwei Kanten e_k und e_ℓ, die in verschiedenen Zusammenhangskomponenten von $T \setminus \{e_j\}$ liegen. Dann folgt: Jeder Pfad, der die Kanten e_k und e_ℓ benutzt, muss auch e_j benutzen, insbesondere hat jede Spalte Nichtnulleinträgen in der k-ten und ℓ-ten Zeile einen Nichtnulleintrag in der k-ten Zeile. Daher folgt nach Definition der G_i, dass G_j unzusammenhängend ist, denn die Endpunkte der Kanten e_k und e_ℓ können in G_j nicht verbunden werden. □

Greift das Ghouila-Houri-Kriterium also nicht, besteht der nächsten Schritt darin, den Zusammenhang der Graphen G_i zu prüfen. Findet man keinen unzusammenhängenden Graphen, so ist M keine Netzwerkmatrix. Sonst sei ohne Einschränkung G_1 unzusammenhängend. Seien $C_1, \dots, C_p$ die Zusammenhangskomponenten von G_1. Bezeichne mit M_i die i-te Zeile von M.

Setze $W := \operatorname{supp} M_1$ und $W_i := W \cap \operatorname{supp}(M_i)$ für $i \geqslant 2$. Sei U_k die Vereinigung der W_i über die Knoten i in der k-ten Zusammenhangskomponente.

Definiere den Graphen H. Seine Knoten entsprechen den Zusammenhangskomponenten $C_1, \dots, C_p$. Es seien C_k und C_ℓ genau dann durch eine Kante verbunden, wenn gilt:

$$\exists i \in C_k : U_\ell \not\subseteq W_i \wedge U_\ell \cap W_i \neq \emptyset \wedge \exists j \in C_\ell : U_k \not\subseteq W_j \wedge U_k \cap W_j \neq \emptyset$$

Für jede Komponente C_k sei M_k die Submatrix bestehend aus der ersten Zeile von M und den Zeilen von M mit Index in C_k. Dann gilt:

Theorem 2.20. *[Sch98]* M *ist Netzwerkmatrix genau dann, wenn* H *bipartit ist und* M_k *für alle* $k = 1, \ldots, p$ *Netzwerkmatrizen sind.*

Für den Beweis siehe die angegebene Referenz. Das Theorem impliziert, dass die Eigenschaft, eine Netzwerkmatrix zu sein, in Polynomialzeit mit Algorithmus 1 überprüft werden kann.

```
Input : m × n-Matrix M
Result : True, wenn M Netzwerkmatrix, sonst False
if es gibt Eintrag M_{i,j} mit M_{ij} ∉ {0, 1, −1} then
    Return False;
end
if jede Spalte von M hat höchstens zwei Nichtnulleinträge then
    konstruiere Graphen G;
    Return istBipartit(G);
end
existsDisconnected := False;
for i = 1, ..., m do
    Konstruiere Graphen G_i;
    if G_i ist unzusammenhängend then
        existsDisconnected := True;
        break;
    end
end
if ¬ existsDisconnected then
    Return False;
end
Konstruiere Graphen H aus unzusammenhängendem G_i;
if ¬ H bipartit then
    return False;
end
for C_k Zusammenhangskomponente von G_i do
    Konstruiere Submatrix M_k;
    if ¬ NetzwerkMatrix(M_k) then
        Return False;
    end
end
Return True;
```

Algorithmus 1 : Polynomiale Erkennung von Netzwerkmatrizen

Zur Analyse genügt es zunächst, die Matrix

$$\begin{pmatrix} A \\ L \end{pmatrix}$$

zu untersuchen. Kann man von dieser zeigen, dass sie nicht vollständig unimodular ist, so kann auch $\overline{A}$ nicht vollständig unimodular sein.

Dazu wurde zunächst die Matrix des vollständigen 4×2-Gitters aus der obigen Konstruktion untersucht. Diese hat (bei Nummerierung entsprechend der Abbildung) die Gestalt aus Tabelle 2.3.

$$L_1 := \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0
\end{pmatrix}$$

Tabelle 2.3: AL-Matrix des 4×2-Gitters

Mit Hilfe des Erkennungsalgorithmus für Netzwerkmatrizen konnte gezeigt werden, dass es sich bei Matrix L_1 um eine Netzwerkmatrix handelt. Dann wurde ein vollständiges 4×3-Gitter ausprobiert mit Übernahme der bisherigen Nummerierung, siehe Abbildung.

Man erhält die Matrix in Tabelle 2.4.

Dabei wurde festgestellt:

Bemerkung 2.21. L_2 *ist keine Netzwerkmatrix.*

So kann schon nach einem Rekursionsschritt kein unzusammenhängendes G_i mehr gefunden werden.

Dies schließt die vollständige Unimodularität der Matrix natürlich nicht aus. Mit Hilfe einer vollständigen Implementierung des Seymourschen Dekompositionsalgorithmus von Matthias Walter konnte jedoch eine verletzende Submatrix identifiziert werden.

Die Submatrix mit den Zeilenindizes

$$(11, 20, 21, 23, 26, 31, 34)$$

und den Spaltenindizes

$$(5, 6, 10, 12, 14, 16, 17),$$

$$L_2 := \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}$$

Tabelle 2.4: AL-Matrix des 4×3-Gitters

gegeben durch Tabelle 2.5, hat Determinante 2.

$$M := \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Tabelle 2.5: Verletzende Submatrix der AL-Matrix des 4×3-Gitters

2.4 Problemangepasste Lösungsansätze für MTCP

Während im letzten Abschnitt allgemeine Lösungsansätze für das MTCP-Problem betrachtet wurden, folgen nun Ansätze, die direkt auf das Problem zielen und für andere nur mit Anpassungen zu gebrauchen sind.s

2.4.1 Polynomialität des MTCP-Problems

In diesem Abschnitt soll gezeigt werden, dass das MTCP in Polynomialzeit gelöst werden kann. Dazu werden zwei verschiedene Ansätze gezeigt, von denen einer am Ende zum Ziel führt.

Ursprüngliche Idee Zunächst wird in diesem Abschnitt eine Idee vorgestellt, die sich als für das MTCP-Problem zumindest ohne weitere Ideen nicht als passend herausgestellt hat. Diese kann aber möglicherweise zur Reduktion von ähnlich gearteten Problemen und mit Modifikationen eventuell sogar für Cover-Probleme benutzt werden, daher wird sie zumindest erwähnt.

Das Knotengadget (s. Abbildung 2.5) steht für jeweils einen Knoten des originalen Gitters. Die dick eingezeichneten Linien dienen zur Verbindung der Knotengadgets untereinander entsprechend der Verbindung der originalen Knoten. In der Reduktion entspräche die Auswahl einer dicken Kante der Nichtauswahl dieser Kante im Originalgraphen. In dieser Form war das Gagdet für das Partitionierungsproblem gedacht. Die Probleme, die hierbei auftreten, sind zweierlei. Zuerst wäre zu nennen, dass in dieser Form keine Partition erzwungen wird. Denn es können einerseits "gegenüberliegende" Knicke, also z.B. NW und SE, beide benutzt werden zu Kosten 2. Das wird aus Kostengründen nicht auftreten, denn es können stattdessen ebensogut zwei gerade Knicke ausgewählt werden, also NS und EW.

Das andere Problem, das noch schwerwiegender auffällt und den Reduktionsversuch auf diese Art und Weise zum Scheitern gebracht hat, besteht darin, dass der Knoten nicht überdeckt werden muss. Benutzt man das Gadget also so, wie es in der Abbildung zu erkennen ist, besteht eine Optimallösung darin, dass kein Knotengadget überdeckt wird, also alle dicken Kanten benutzt werden. Dann haben alle ausgewählten Kanten Kosten von 0. Dies lässt sich auf verschiedene Arten beheben. So kann zum Beispiel das Gewicht der Verbindungskanten erhöht werden oder es

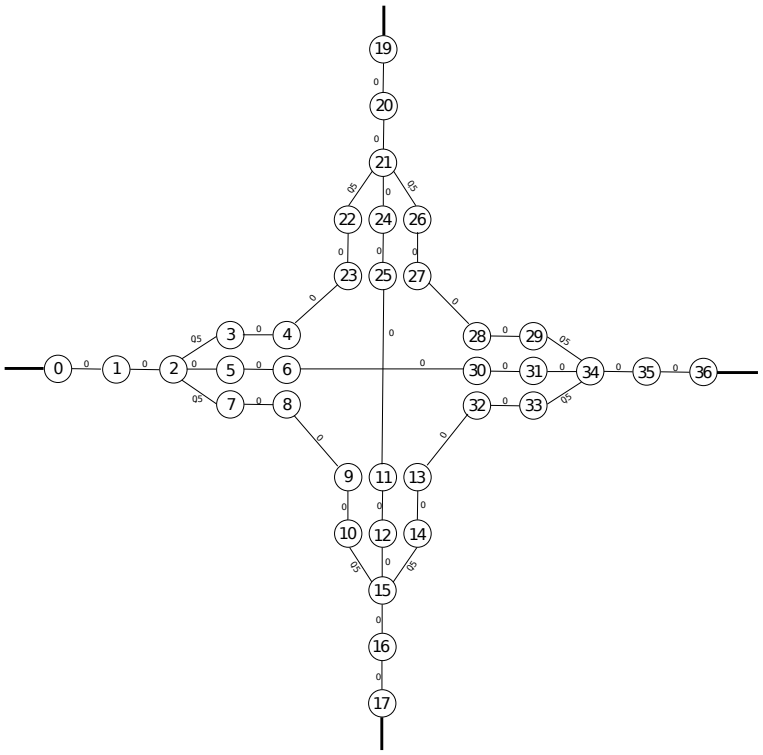

Abbildung 2.5: Knotengadget

können auch gewisse Änderungen in dem Gadget durchgeführt werden wie das Einführen von zwei zusätzlichen Knoten, die beide durch Knoten aus dem ursprünglichen Gadget saturiert werden müssen und dadurch eine Überdeckung erzwingen. Alle diese Möglichkeiten bringen jedoch immer auch eigene Schwierigkeiten mit sich, so ließen sich zum Beispiel mit letzterer Methode nach Einführung der zusätzlichen Knoten die Abbiegekosten nicht mehr für alle Abbiegearten erzwingen.

Die Reduktion Nun wird die eigentliche Reduktion erklärt, diese benutzt das Knotengadget aus Abbildung 2.6.

Das für die Reduktion benutzte Knotengadget ist in Abbildung 2.6 dargestellt. In einem perfekten Matching müssen in dem Gadget die Knoten 1 und 2 mit zwei der Knoten N, E, S, W verbunden werden. Offenbar bleiben stets zwei Knoten übrig. Diese müssen durch die Verbindungen zu den anderen Knotengadgets gematcht werden.

Die horizontalen und vertikalen Verbindungen zwischen den Knotengadgets sind in den Abbildungen 2.7 und 2.8 dargestellt. Um Schwierigkeiten bei der Kostenerhaltung aus dem Weg zu gehen, wird definiert:

Definition 2.22. *Sei* $\tilde{G}$ *der Graph, der entsteht, wenn man die Knoten von* G *durch die Knotengadgets aus Abbildung 2.6 und die horizontalen und vertikalen Kanten durch die in den Abbildungen 2.7 und 2.8 zu sehenden Verbindungsgraphen ersetzt.*

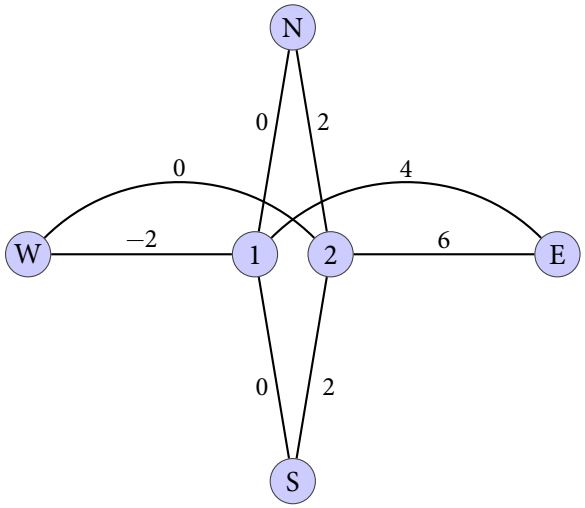

Abbildung 2.6: Knotengadget für MTCP-Reduktion

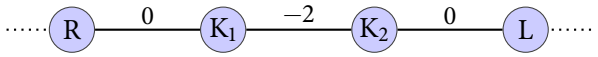

Abbildung 2.7: horizontale Knotenverbindung für MTCP-Reduktion

*Ein perfektes Matching im Graphen $\tilde{M}$ heiße **zulässig**, wenn folgende Regeln berücksichtigt werden:*

- *Werden in einem Knotengadget die Zentralknoten* 1 *und* 2 *mit den Knoten* S *und* E *verbunden, so ist stets* S *mit dem Knoten* 2 *und* E *mit dem Knoten* 1 *verbunden.*
- *Werden in einem Knotengadget die Zentralknoten* 1 *und* 2 *mit den Knoten* S *und* N *verbunden, so ist stets* S *mit dem Knoten* 2 *und* N *mit dem Knoten* 1 *verbunden.*
- *Werden in einem Knotengadget die Zentralknoten* 1 *und* 2 *mit den Knoten* S *und* W *verbunden, so ist stets* S *mit dem Knoten* 1 *und* W *mit dem Knoten* 2 *verbunden.*
- *Werden in einem Knotengadget die Zentralknoten* 1 *und* 2 *mit den Knoten* N *und* W *verbunden, so ist stets* N *mit dem Knoten* 2 *und* W *mit dem Knoten* 1 *verbunden.*
- *Werden in einem Knotengadget die Zentralknoten* 1 *und* 2 *mit den Knoten* N *und* E *verbunden, so ist stets* N *mit dem Knoten* 2 *und* E *mit dem Knoten* 1 *verbunden.*
- *Werden in einem Knotengadget die Zentralknoten* 1 *und* 2 *mit den Knoten* E *und* W *verbunden, so ist stets* E *mit dem Knoten* 1 *und* W *mit dem Knoten* 2 *verbunden.*

Ziel dieses Abschnitt ist das Theorem:

Theorem 2.23. *Sei* $G = (V, E)$ *Gitter. Sei* $\tilde{G}$ *der Graph, der entsteht, wenn man die Knoten von* G *durch die Knotengadgets aus Abbildung 2.6 und die horizontalen und vertikalen Kanten durch die*

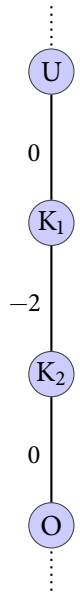

Abbildung 2.8: vertikale Knotenverbindung für MTCP-Reduktion

in den Abbildungen 2.7 und 2.8 zu sehenden Verbindungsgraphen ersetzt. Für ein Matching in $\tilde{G}$ *sei die Summe der Kantengewichte, für eine Lösung des MTCP-Problems in* G *seien die Knickkosten die Zielfunktion. Dann gilt: Es gibt eine bijektive, kostenerhaltende Abbildung von der Menge der zulässigen perfekten Matchings in* $\tilde{G}$ *in die Menge der Lösungen des MTCP-Problems in* G.

Korollar 2.24. *Das MTCP-Problem kann in* $\mathcal{O}(n^3)$ *mit dem ungarischen Algorithmus nach Kuhn & Munkres gelöst werden.*

Sei M ein perfektes Matching in $\tilde{G}$. Jedem Knoten in G entspricht ein Knotengadget in $\tilde{G}$ und jeder Kante eine horizontale bzw. vertikale Verbindung in $\tilde{G}$. Sei $R(v)$ für einen Knoten $v \in G$ bzw. $R(e)$ für eine Kante in G der entsprechende Teilgraph des Graphen $\tilde{G}$ (die Verbindungskanten sollen zu den Kantengadgets gehören).

Die Lösung für das MTCP-Problem wird nun für jeden Knoten und jede Kante einzeln konstruiert. Zu jeder Kante e müssen in $R(e)$ die Knoten K_1 und K_2 überdeckt werden. Das kann auf genau zwei Arten geschehen: Entweder wird die Kante zwischen K_1 und K_2 ausgewählt oder es werden die beiden anderen Kanten ausgewählt. Ersterem entspricht die Auswahl der Kante in G, letzterem die Nichtauswahl der Kante.

Jeder Knoten in G muss durch genau einen Knick einer der Arten NS, EW, NE, SE, NW, EW überdeckt werden. Betrachte für einen Knoten $v \in V$ das Gadget $R(v)$. In $R(v)$ müssen die Knoten 1 und 2 überdeckt werden. Auf diese Weise werden genau zwei Richtungen durch Kanten mit 1 und 2 verbunden und zwei nicht. Seien D_1 und D_2 diese Richtungen. Dann wird im Knoten v ein Knick in die Richtungen D_1 und D_2 ausgewählt. Offenbar ist dann jeder Knoten in G durch genau einen Knick überdeckt und die Graderhaltungsbedingungen in G sind erfüllt, weil sie auch in $\tilde{G}$ über die Verbindungsgadgets hinweg erfüllt werden müssen (sonst blieben Knoten unüberdeckt).

Lemma 2.25. *Sei* $G = (V, E)$ *Gitter mit einer Lösung des MTCP-Problems. Bezeichne* SE *die Anzahl der Südost-,* SW *die Anzahl der Südwest-,* NW *die Anzahl der Nordwest-,* NE *die Anzahl der Nordost-,* NS *die Anzahl der Nordsüd- und* EW *die Anzahl der Ostwestknicke. Dann gilt:*

$$SE + SW + NW + NE = 2 \cdot (NE + SE)$$

Zum Beweis des Lemmas wird ein weiteres Lemma benutzt:

Lemma 2.26. *[KV00] Für jeden Graphen* $G = (V, E)$ *gilt*

$$\sum_{v \in V} |\delta(v)| = 2|E| \tag{2.10}$$

Lemma 2.27. *Sei* $G = (V, E)$ *Gitter mit einer Lösung des MTCP-Problems. Bezeichne* SE *die Anzahl der Südost-,* SW *die Anzahl der Südwest-,* NW *die Anzahl der Nordwest-,* NE *die Anzahl der Nordost-,* NS *die Anzahl der Nordsüd- und* EW *die Anzahl der Ostwestknicke. Sei* $E_L = H_L \dot\cup S_L$ *disjunkte Vereinigung der horizontalen Kanten und der vertikalen Kanten. Dann gilt*

$$SE + NE + NW + SW + 2 \cdot EW = 2 \cdot (|H_L|) \tag{2.11}$$

Beweis. Die Knoten von L lassen sich in drei Klassen unterteilen: Die Knoten, an denen keine horizontale Kante ausgewählt wird, die, an denen eine horizontale Kante ausgewählt wird und schließlich die Knoten mit Knick EW. Der Horizontalgrad der Knoten des erstes Typs ist gleich 0, des zweiten Typs gleich 1 und des dritten Typs gleich 2. Da die Koeffizienten auf der linken Seite von Gleichung 2.11 genau dem durch die Knicke entstehenden Horizontalgrad entsprechen, folgt die Behauptung.

□

In der Formel steht auf der rechten Seite also eine 2, weil jede horizontale Kante einmal auf ihrer linken Seite durch einen Knick der Form SE,NE der EW und einmal auf ihrer rechten Seite durch einen Knick der Form SW,NW oder EW gezählt wird. Aus dieser Überlegung erhält man:

Lemma 2.28. *Die Bezeichnungen seien wie im letzten Lemma. Dann gilt*

$$SE + NE + EW = |H_L|$$

und ebenso gilt

$$SW + NW + EW = |H_L|$$

Man erhält also:

$$SW + NW + EW = SE + NE + EW$$

Durch Subtraktion von EW auf beiden Seiten ergibt sich

$$SW + NW = SE + NE$$

So erhält man Lemma 2.25.

Es folgt die Analyse des Gadgets aus Abbildung 2.6. Wie die Auswahl eines Knicks der Auswahl eines partiellen Matchings in dem Gadget entspricht, wurde bereits oben erläutert. Es wurden nun die minimalen Kosten der getroffenen Kantenauswahl gelistet:

NS	2
EW	2
SE	6
SW	0
NE	6
NW	0

Ein Knick heiße **echt**, wenn er nicht von der Form NS oder EW ist.
Folgendes Lemma wird sich im Weiteren als nützlich erweisen:

Lemma 2.29. *Ein **Geradenstück** auf einem Kreis C sei eine inklusionsweise maximale Abfolge von unechten Knicken zusammen mit den echten Knicken an dem Ende der Abfolge. Dann gilt: Die Anzahl der Knicke in C ist gleich der Anzahl der Geradenstücke in C.*

Die Aussage folgt dann auch direkt für Kreispartitionierungen.
Ziel ist nun zu zeigen, dass die von dem Matching verursachten Kosten den Abbiegekosten entsprechen. Dazu wird zunächst gezeigt, dass diese Aussage für jedes Geradenstück gilt. Dann kann man die Aussage auf Kreise und damit auf Kreispartitionierungen hochziehen.

Lemma 2.30. *Geradenstücke verursachen Kosten von* 1.

Beweis. Sei L ein Geradenstück mit $k+2$ Knoten. Dann enthält L $k+1$ Kanten. Diese entsprechen Verbindungsgadgets mit Gesamtkosten $-2 \cdot (k+1) = -2k-2$.
Nach Lemma 2.25 kann man annehmen, dass jeder echte Knick Kosten von 3 verursacht. Weiterhin gibt es auf dem Geradenstück L k Knoten mit unechtem Knick, diese verursachen Kosten von $2 \cdot k$. Da es sich um Kreise handelt, jedes Geradenstück also zu genau zwei echten Knicken und zu jedem echtem Knick genau zwei Geradenstücke gehören, zählt man zur Vermeidung von Doppelung für jedes Geradenstück nur einen der beiden echten Knicke. Man erhält Gesamtkosten von

$$3 + 2 \cdot k - 2k - 2 = 1$$

Jedes Geradenstück verursacht also Kosten von 1. □

Damit folgt mit der oben angegebenen Konstruktion die Behauptung aus Theorem 2.23.

2.5 Untersuchungen zu weiteren Problemen

In diesem Abschnitt werden weitere Probleme im Rahmen der Klassifikation untersucht.

2.5.1 Achsenparallele Überdeckung mit Linkkosten [KACL]

Es wurde in [AKMS97] gezeigt, dass KNCW auf vollständigen Graphen $\mathcal{NP}-$schwer ist. Die dort benutzte Reduktion benutzt von sich aus nur bei der Überdeckung der sogenannten U-Turn-Gadgets nichtachsenparallele Verbindungen. Daher liegt die Idee nahe, dass man durch eine geeignete Modifikation dieser Gadgets die Reduktion auf den achsenparallelen Fall übertragen kann und dann auch in diesem die Schwere des Problems erhält.

Wie man diese Reduktion jedoch übernehmen kann, ist nicht offensichtlich. Wie sich herausstellt, werden die nichtachsenparallelen Verbindungen auf entscheidende Weise benutzt. So besteht bei der Überdeckung dort in den zu einer Variablen gehörigen Zeile Wahlfreiheit zwischen der ersten und der zweiten Zeile. Diese werden dort mit den gleichen Kosten überdeckt (bis auf einen vernachlässigbar kleinen Anteil). Dieser Teil der Reduktion lässt sich achsenparallel nur schwer simulieren, da bei einfachem Ersetzen der U-Turn-Gadgets durch von Knoten gebildete Rechtecke beim Überdecken der oberen Zeile Kosten von 4, beim Überdecken der unteren Zeile jedoch Kosten von 8 aufträten. Dieser Unterschied ist aber nicht mehr vernachlässigbar klein.

Eine Idee, wie man dieser Schwierigkeit ausweicht, besteht darin, dass man erzwingt, dass die Zusatzkosten statt bei keiner der beiden Überdeckungsmöglichkeiten wie im nichtachsenparallelen Fall hier bei beiden Überdeckungen erzwungen werden, indem man das obere Ende des U-Turn-Gadgets eine Zeile über die obere der drei Variablenzeilen legt, so dass die Zusatzknicke stets nötig werden, siehe Abbildung 2.9.

Nun hat man aber das Problem, dass, anstatt bei der Überdeckung der U-Turn-Gadgets auf eine der freien Zeilen zu knicken und wieder weg, was Kosten von 8 verursacht 2.10, man auch beide U-Turn-Gadgets mit Kosten 4 überdecken kann und dann mit weiteren Kosten von 4 beide freien Variablenzeilen. Dadurch werden also bis auf die vertikalen U-Turn-Gadgets schon alle Knoten überdeckt (Abbildung 2.11).

Um dieses Problem wiederum zu vermeiden, kann man die U-Turn-Gadgets auf der rechten Seite so modifizieren, dass ihr oberes Ende noch eine Zeile weiter oben liegt. Dann entstehen bei der eben skizzierten Triviallösung Kosten von 10, während bei der gewollten Lösung mit einer freien Variablenzeile nur Kosten von 8 entstehen.

Nun muss man sich noch Gedanken darüber machen, wie die vertikalen U-Turn-Gadgets konstruiert werden. Im nichtachsenparallelen Fall hat man die Wahlfreiheit, welche der drei Zeilen man freilässt, dort entstehen bis auf vernachlässigbar kleine Anteile stets die gleichen Kosten.

Es wird wiederum von der Idee ausgegangen, die Kosten für die verschiedenen Überdeckungsmöglichkeiten dadurch zu egalisieren, dass eventuell auftretende Zusatzkosten bei einer der Überdeckungsmöglichkeiten durch geeignete Konstruktion bei allen Möglichkeiten auftreten sollen.

Das alles entscheidende Problem, das hierbei auftritt, ist die Wahlmöglichkeit zwischen nicht nur zwei, sondern drei Überdeckungsarten. Man konstruiert also wie gehabt rechteckige U-Turn-Gadgets und zwar so, dass die oberen Gadgets ihren vertikalen Anteil nicht auf der selben Geraden haben wie die unteren Gadgets. Für das Überdecken von zwei U-Turn-Gadgets und zwei Spalten werden dann also Kosten von 12 benötigt.

Die Triviallösung, die keine der Spalten überdeckt, benötigt jedoch nur Kosten von 8. Kombiniert man diese mit der Lösung für die Zeilen, die pro Zeile zwei teurer ist als die gewünschte Lösung, erhält man eine Lösung, die bei hoher Spaltenanzahl billiger ist als die gewünschte (bei $m > \frac{n}{2}$).

Eine weitere Idee, die man dann einbringen kann, ist durch Vervielfachung der Zeilen die gewünschte Lösung zu favorisieren (zum Beispiel jede Zeile $m-$mal).

Das funktioniert aber so einfach nicht, weil dann die Lösung aus Abbildung 2.12 billiger wird.

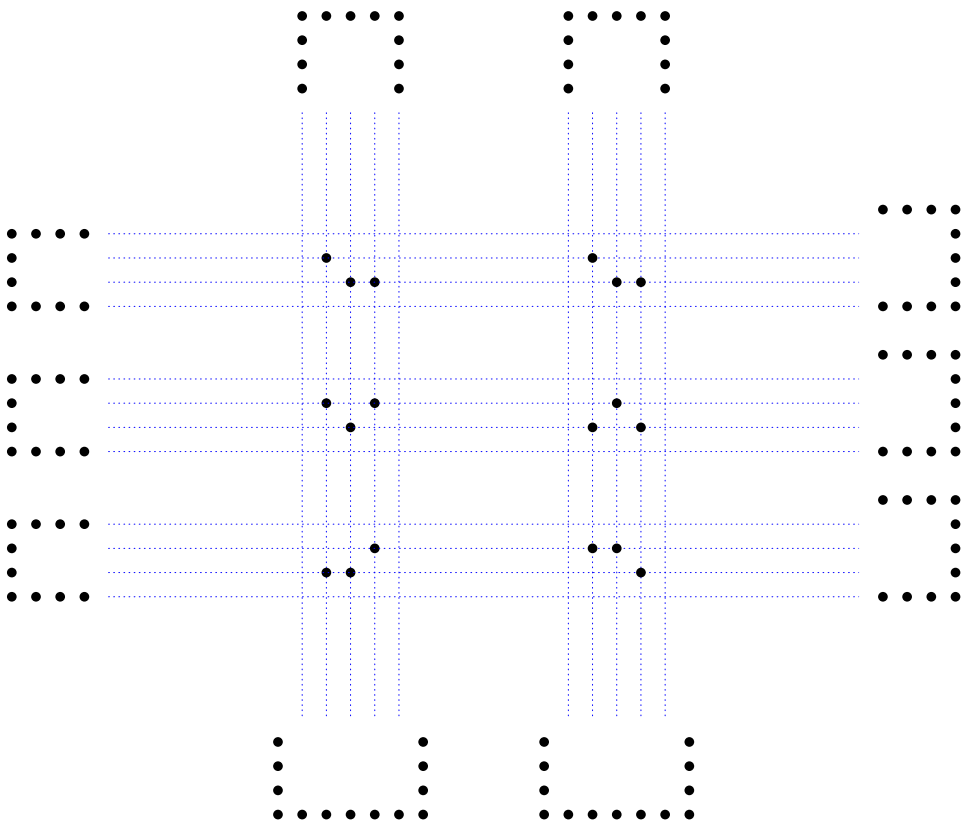

Abbildung 2.9: Erste Abbildung zur fehlgeschlagenen HACL-Reduktion

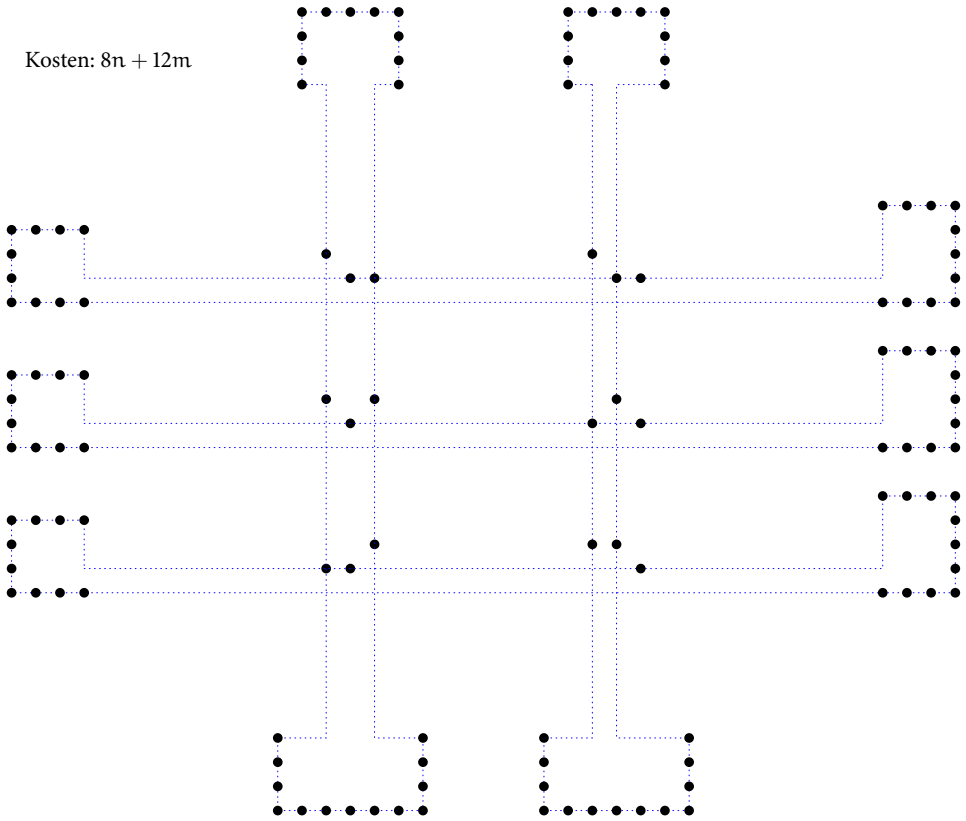

Abbildung 2.10: Zweite Abbildung zur fehlgeschlagenen HACL-Reduktion

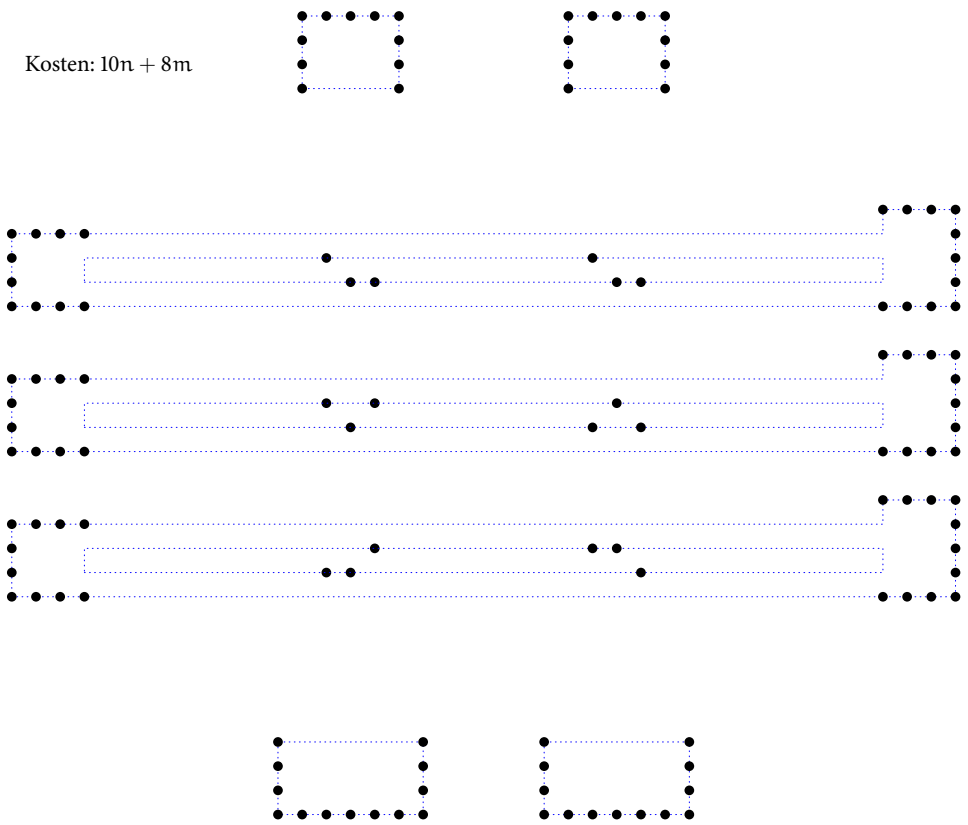

Abbildung 2.11: Dritte Abbildung zur fehlgeschlagenen HACL-Reduktion

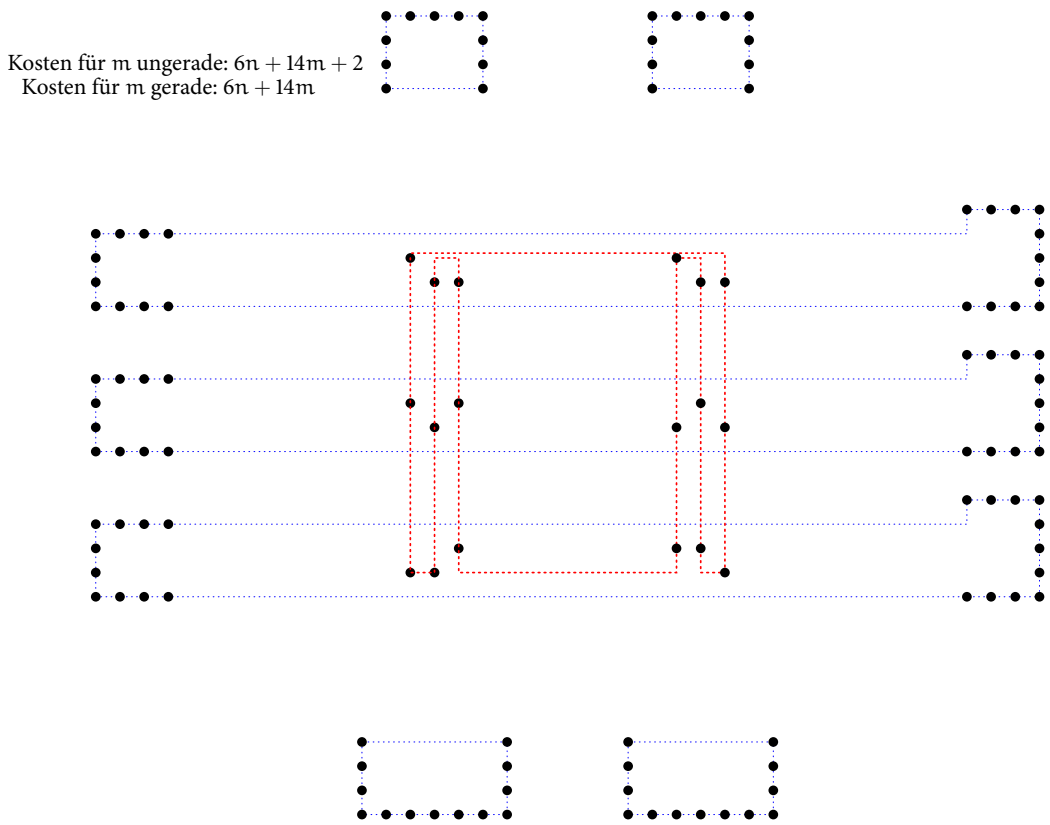

Abbildung 2.12: Vierte Abbildung zur fehlgeschlagenen HACL-Reduktion

2.5.2 Partitionierung mit beliebigen Knicken und Linkkosten [KBPL]

Bei diesem Problem wird eine Familie von Kreisen gesucht, Knicke sind überall erlaubt. Es wird eine Partition der Knoten gesucht und es zählt die Anzahl der Knicke.

Sieht man sich das Problem an, kann man auf die Idee kommen, dass Knicke außerhalb von Knoten keinen Vorteil bringen. Das gilt zwar nicht, man kann aber die Zielfunktionswerte gegeneinander abschätzen:

Theorem 2.31. *Zu jeder Lösung von KBPL vom Wert OPT gibt es eine Lösung von KNPL vom Wert höchstens* $2 \cdot OPT$.

Beweis. Man kann annehmen, dass jeder Knick außerhalb eines Knotens in der Lösung von KBPL zwei zu überdeckende Knoten als Nachbarn hat. Hat ein Knick nämlich einen Knick als Nachbarn, kann man durch geeignetes Verschieben des Knickortes einen der Knicke einsparen, dadurch wird die Schranke höchstens besser.

Jeder Knick außerhalb eines Knotens kann in einer Lösung von KNPL durch Knicke in seinen beiden Nachbarn ersetzt werden. Aufgrund der Annahme, dass Knicke stets nur Knoten als Nachbarn haben, verdoppelt sich auf diese Weise der Zielfunktionswert höchstens.

□

Wie eben gezeigt wurde, unterscheiden sich die optimalen Zielfunktionswerte von KBPL und KNPL höchstens um einen Faktor 2. Es soll nun gezeigt werden, dass dieser Wert auch angenommen wird.

Sei dazu P ein regelmäßiges $n - \mathrm{Eck}$. Auf jeder Kante des n-Ecks werden 5 Knoten platziert, zwei auf den Ecken und drei im Inneren. Entferne nun aus P die Ecken.

Eine minimale Lösung von KBPL benötigt dann offensichtlich nur n Knicke, indem sie in den Ecken knickt, die entfernt wurden.

Betrachte nun das KNPL-Problem auf P. Knicke in den Ecken sind nun nicht mehr möglich, man muss auf Knicke in den Nachbarn der Ecken ausweichen. Diese Lösung hat genau $2n$ Knicke.

2.5.3 Minimum-Turn Cycle Cover

Für das MTCC sind genau die Gitter zulässig, die keine isolierten Punkte enthalten. U-Turns sind erlaubt.

Beim MTCC-Problem haben die Optimallösungen der LP-Relaxation mitunter einen echt besseren Zielfunktionswert als die Lösungen der zugehörigen ganzzahligen Programme.

So ist eine Optimallösung des MTCC für das vollständige 3×3-Gitter in Abbildung 2.13 zu sehen. Normal dicke Knicke werden mit Wert 1, dicker eingezeichnete Knicke mit Wert 2 benutzt.

Im Gegensatz dazu muss die LP-Relaxation in der Mitte nicht mit Wert 2 überdecken. Eine optimale Lösung der LP-Relaxation ist in Abbildung 2.14 zu sehen. Die durchgezogenen Linien bedeuten einen Wert von 1, die gestrichelten Linien einen Wert von $\frac{1}{2}$. Der Zielfunktionswert beträgt hier 6.

Diese Beobachtung wirft die Frage auf, wie weit (als Faktor) der optimale Zielfunktionswert der LP-Relaxation von dem optimalen Zielfunktionswert des IPs abweichen kann (die sogenannte **Ganzzahligkeitslücke**). Die Frage konnte im Rahmen dieser Arbeit nicht geklärt werden. Man kann jedoch zeigen, dass die Lücke mindestens 2 beträgt.

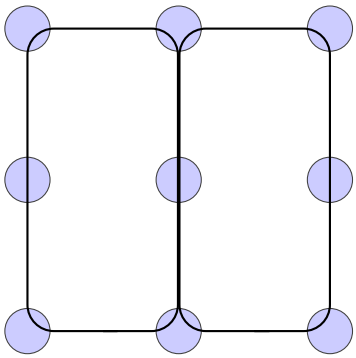

Abbildung 2.13: Ganzzahlige Optimallösung des MTCC-Problems für das 3×3-Gitter

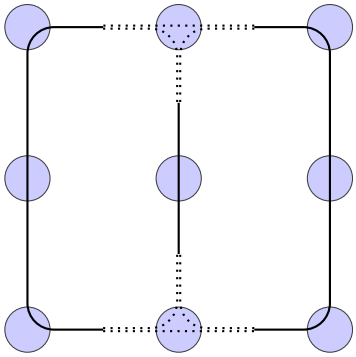

Abbildung 2.14: LP-Optimallösung des MTCC-Problems für das 3×3-Gitter

Das Beispiel für die Ganzzahligskeitslücke von 2 sind die k-**Treppen**. Diese sind Untergraphen von Gittern. k bezeichnet die Anzahl der verschiedenen Zeilen. Optimallösungen für die 25-Treppe und für deren LP-Relaxation sind in den Abbildungen 2.15 und 2.16 zu sehen. Die rot eingezeichneten Knicken in der ersten der beiden Abbildungen werden mit Wert $\frac{1}{2}$ genommen, die schwarzen dünnen Knicke mit Wert 1 und die fett eingezeichneten Knicke mit Wert 2.

Bei diesem Beispiel tritt eine Ganzzahligkeitslücke von $\frac{100}{54} \cong 1.85$ auf. Allgemein ist der optimale Zielfunktionswert für das LP gleich $2k + 4$ und für das IP $4k$. Dass das LP immer diesen Wert erreicht, ist leicht durch die Konstruktion zu sehen. Man muss argumentieren, warum das IP nicht mit steigendem k vielleicht eine bessere Lösung produziert. Da jeder Kreis auf der Treppe aber wieder zum Anfang zurückkehren muss und es keine andere Möglichkeit gibt, als wieder über die Treppe zurückzukehren (mit Ausnahme der kleinen Kreise an den Enden der Treppe)), muss jeder der Knoten auf der Treppen (bis auf die an den Enden) zweimal überdeckt werden. Zumindest von den Knoten, die nicht am Ende liegen, verursacht jeder Kosten von 2. Das liefert eine untere Schranke von $4 \cdot (k - 2)$. Für $k \to \infty$ ist die Lücke also für kein $\epsilon > 0$ durch $2 - \epsilon$

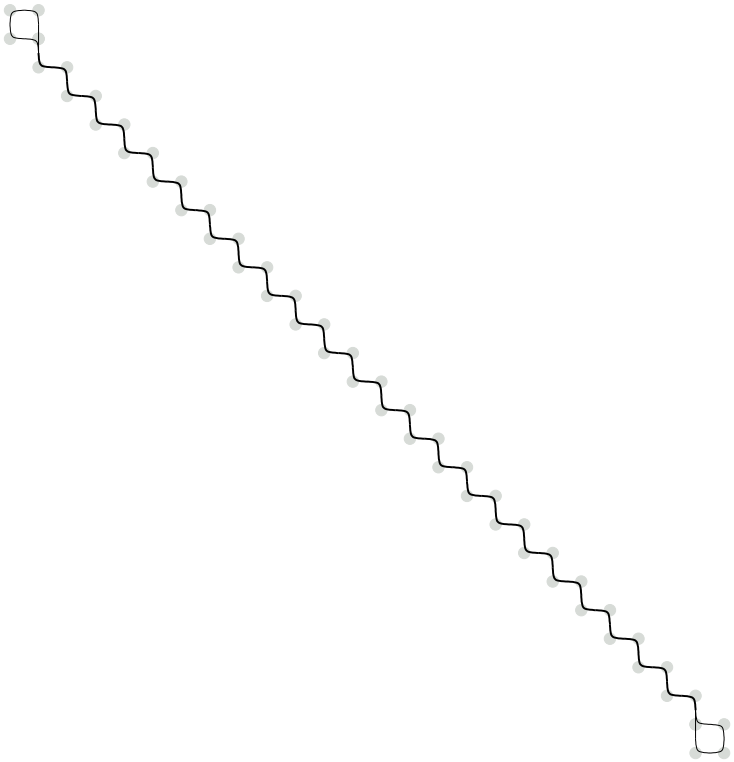

Abbildung 2.15: Optimallösung der 25-Treppe, $c_{opt} = 100$, dicke Kanten mit Gewicht 2

beschränkbar.

In Abbildung 2.17 sieht man eine LP-Optimallösung, die nicht halbzahlig bleibt. Alleine durch Lösen der LP-Relaxation und Multiplikation mit dem kleinsten natürlichen k, das alle Werte ganzzahlig macht, ist ohne weitere Schritte also bestenfalls eine $4-$Approximation erreichbar.

Gomory-Chvátal-Schnitte [CF96] Betrachte das Polytop $P = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$, $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$. Der schwierige Teil, eine lineare Zielfunktion über die ganzzahligen Punkte innerhalb des Polytops P zu minimieren, besteht darin, dass man (zumindest teilweise und implizit) die im Inneren des Polytops liegendenden ganzzahligen Punkte ermitteln muss. Die **ganzzahlige Hülle** P_I von P ist definiert durch $P_I := \text{conv}\{x \in \mathbb{Z}^n \mid Ax \leqslant b\}$.

Eine Methode, über die ganzzahlige Hülle von P zu optimieren, besteht darin, über die LP-Relaxation zu minimieren. Ist die gefundene Optimallösung nicht ganzzahlig (beim Simplex genügt es, "nicht ganzzahlig" zu sagen. Allgemein müsste man wohl sagen: nicht in P_I, denn ohne Angabe eines Lösungsverfahrens für LPs kann man ja nicht von einer optimalen Ecklösung ausgehen), kann ein **Schnitt** (eine separierende Hyperebene) gesucht werden, der die gefundene Lösung unzulässig macht. Eine zugehörige Ungleichung wird dann zum aktuellen linearen Programm hinzugefügt und es wird erneut gelöst. Diese Schritte werden solange iteriert, bis eine gefundene Optimallösung ganzzahlig ist. Dann gehört sie zu P_I und ist eine optimale Lösung des ganzzahligen Programms. Dieser Algorithmus führt jedoch, da nicht spezifiert wurde, wie die se-

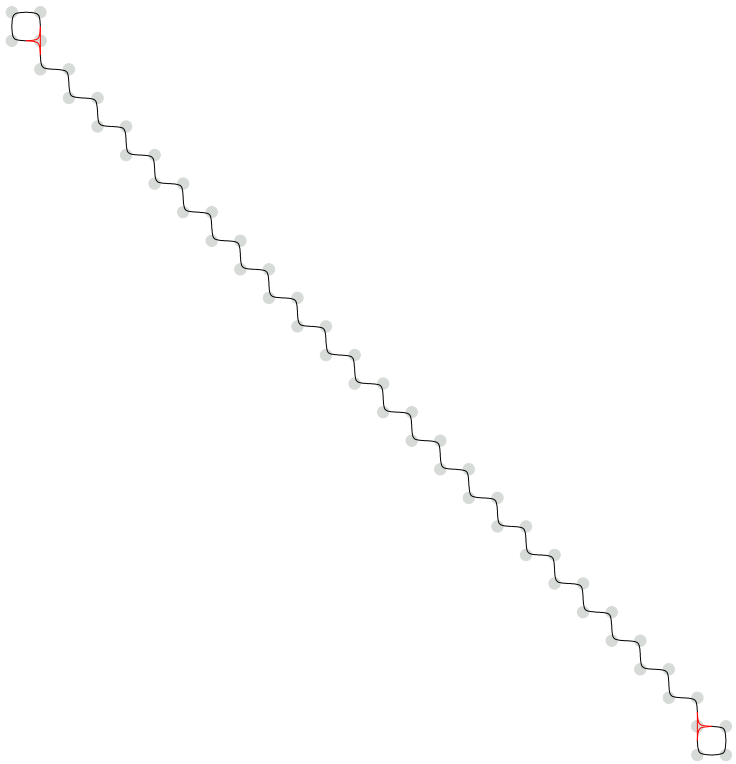

Abbildung 2.16: LP-Optimallösung der 25-Treppe, $c_{opt} = 54$

parierende Hyperebene liegen soll, zunächst nicht notwendig in endlich vielen Schritten zu einer Lösung.

Eine Möglichkeit, die Endlichkeit dieses Verfahrens zu erzwingen, besteht darin, die Optimallösungen der linearen Programme von der ganzzahligen Hülle durch Gomory-Chvátal-Schnitte zu separieren.

Definition 2.32. *Seien* P *und* P_I *wie oben definiert. Ein* ***Gomory-Chvátal-Schnitt*** *ist eine gültige Ungleichung der Form*

$$\lambda^T A x \leqslant \lfloor \lambda^T b \rfloor$$

für $\lambda \in \mathbb{R}^m_+$, $\lambda^T A \in \mathbb{Z}^n$. *Dabei genügt es, sich auf* $\lambda \in [0,1[^m$ *zu beschränken.*

Der **Rang-1-Abschluss** von P ist definiert durch

$$P_1 = \{x \in P \mid \lambda^T A x \leqslant \lfloor \lambda^T b \rfloor, \lambda \in [0,1[^m, \lambda^T A \in \mathbb{Z}^m\}.$$

2.6 Kürzeste Wege in der Ebene

Definition 2.9 (Angular Shortest-Path-Problem)**.** *Gegeben sei ein Gitter* $G = (V, E)$ *und zwei Knoten* $s, t \in V$. *Wieviele Knicke hat ein Weg von* s *nach* t *mindestens?*

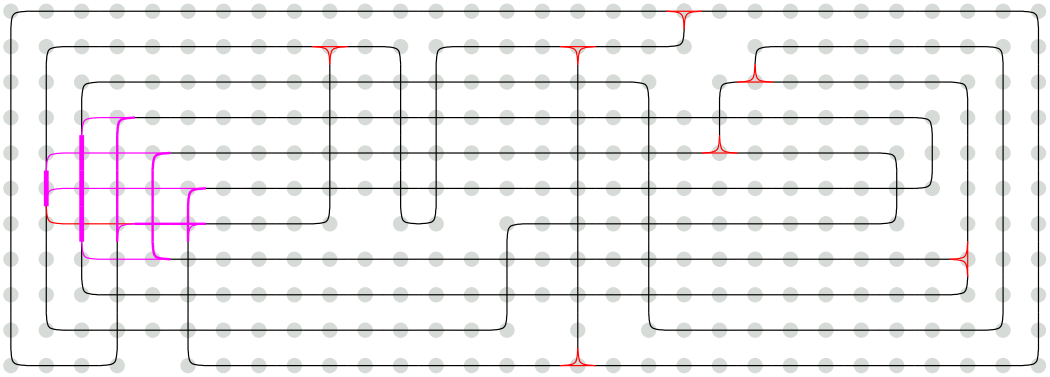

Abbildung 2.17: Eine nicht halbzahlige LP-Optimallösung des MTCC, rot=$\frac{1}{2}$, violett=$\frac{1}{4}$, violett fett=$\frac{3}{4}$, $c_{opt} = 42$

Im Rahmen der algorithmischen Geometrie gibt es eine Reihe von Untersuchen zur Berechnung von kürzesten Wegen in Polygonen mit der sogenannten *Link-Distance-Metrik*. Diese bezieht sich auf die Anzahl an linearen Teilstücken in einem Weg und entspricht damit der Knickmetrik.

Die Berechnung von kürzesten Wegen in Gittern mit Knickkosten lässt sich als Berechnung von Wegen mit minimaler Link-Distance in Polygonen mit polygonalen Hindernissen auffassen, wenn keine U-Turns zugelassen sind. Für dieses Problem existiert ein Algorithmus mit Laufzeit $\mathcal{O}(m \log^2 n\alpha(n))$.[MRW90]

Für Gitter lässt sich das Problem auf einfachem Weg durch einen modifizierten Labeling-Algorithmus lösen. Dieser Ansatz wird hier beschrieben.

Sei $D := \{N, E, S, W\}$ die Menge der Richtungen. Seien $c(d_1, d_2)$ die Kosten, in einem Knoten mit Richtung d_1 in Richtung d_2 zu drehen. Diese Kosten sind also Element von $\{0, 1, 2\}$. Sei außerdem $l(d)$ die Richtung, die aus d durch Linksrehung, $r(d)$ die Richtung, die aus d durch Rechtsdrehung entsteht.

Der Algorithmus endet spätestens nach n Schleifendurchläufen, da ein abbiegeminimaler Weg jeden Knoten im Graphen höchstens einmal besucht.

Eine andere Möglichkeit für die Suche nach knickminimalen Wegen in einem Gitter, die zu den gleichen Ergebnissen wie der vorherige Algorithmus führt, ist eine Anpassung des Graphens. So kann man jeden Knoten durch einen K_4 ersetzen. Die Knoten seien quadratisch angeordnet und die Ecken seien entsprechend der Himmelsrichtungen ausgerichtet. Horizontale und vertikale Kanten innerhalb des K_4 bekommen ein Gewicht von 0, diagonale Kanten ein Gewicht von 1. Die Verbindungskanten zwischen den K_4-Graphen entsprechen den Kanten im ursprünglichen Graphen und haben eni Gewicht von 0. Die Auswahl der Startrichtung und der Endrichtung spezifiziert dann in den beiden K_4-Graphen zwei Knoten entsprechend der Himmelsrichtungen. Zwischen diesen kann man dann einen kantenbasierten kürzesten Weg in einem Graphen mit nichtnegativen Kantengewichten suchen. Wegen des Maximalgrades von r ist $m \in \mathcal{O}(n)$. Mit Fibonacci-Heaps erhält man dann eine Laufzeit von $\mathcal{O}(n \log n)$. [FT87].

Input : zusammenhängendes Gitter $G = (V, E)$, Startknoten s, Zielknoten t, Startrichtung d
Result : Ein im Knoten s in Richtung d beginnender, in t endender abbiegeminimaler Weg
for Knoten $v \in V$ **do**
  for Knoten $w \in V$ **do**
    for Richtungspaar (d_1, d_2) **do**
      Initialisiere Label $\text{pred}_v(w, d_1, d_2) := \emptyset$;
      Initialisiere Label $\text{dist}_v(w, d_1, d_2) := \infty$;
    end
  end
end
for Knoten $v \in V$ **do**
  for Richtungspaar (d_1, d_2) **mit** $d_1 \neq l(l(d_2))$ **do**
    $\text{pred}_v(v, d_1, d_2) := (v, d_1)$;
    $\text{dist}_v(v, d_1, d_2) := c(d_1, d_2)$;
  end
end
while $\text{dist}_t(s, d_i, d_j) = \infty \forall d_i, d_j \in D$ **do**
  for Knoten $v \in V$ **do**
    for Label $L = (w, d_1, d_2)$ **im Knoten** v **do**
      for Knoten x**, der von Knoten** v **in Richtung** d_2 **erreichbar ist do**
        if $\text{dist}_x(w, d_1, d_2) > \text{dist}_v(w, d_1, d_2)$ **then**
          $\text{dist}_x(w, d_1, d_2) := \text{dist}_v(w, d_1, d_2)$;
          $\text{pred}_x(w, d_1, d_2) := v$;
        end
      end
      if $\text{dist}_v(w, d_1, l(d_2)) > \text{dist}_v((w, d_1, d_2) + 1$ **then**
        $\text{dist}_v(w, d_1, l(d_2)) := \text{dist}_v((w, d_1, d_2) + 1$;
        $\text{pred}_v(w, d_1, l(d_2)) := (v, d_2)$;
      end
      if $\text{dist}_v(w, d_1, r(d_2)) > \text{dist}_v((w, d_1, d_2) + 1$ **then**
        $\text{dist}_v(w, d_1, l(d_2)) := \text{dist}_v((w, d_1, d_2) + 1$;
        $\text{pred}_v(w, d_1, l(d_2)) := (v, d_2)$;
      end
    end
  end
end
Sei $\text{dist}_t(s, d_i, d_j) \neq \infty$;
preds := $\emptyset$;
while $\text{pred}_t(s, d_i, d_j)! = (s, d_k)$ **für alle** $d_k \in D$ **do**
  preds += $\text{pred}_t(s, d_i, d_j)$;
  prev := $\text{pred}_t(s, d_i, d_j)$;
  $(t, d_j) :=$ prev;
end
reverse(preds);
return preds;

Algorithmus 2 : Suche nach einem abbiegeminimalen Weges in einem Gitter

3 Praktisches Lösen

Während im letzten Kapitel die Probleme gemäß der Klassifizierung eher theoretisch untersucht werden, soll es nun in diesem Kapitel um Ansätze gehen, die die Probleme möglichst bis zur Optimalität lösen. Dazu werden neben selbstimplementierten Algorithmen der Lösungscode für nichtbipartites Matching von Ed Rothberg (`http://elib.zib.de/pub/Packages/mathprog/matching/weighted/`) und die Software ILOG CPLEX 11.210 benutzt.

3.1 Generieren von zufälligen Instanzen

In diesem Abschnitt werden die Vorgehensweisen für das Erzeugen einer Testumgebung für die verschiedenen Probleme beschrieben.

3.1.1 Erzeugen von zufälligen Kreisen

Input : Graph $G = (V, E)$ mit n Knoten, nummeriert als $1, \ldots, n$
Result : Ein zufälliger Kreis in G, $\emptyset$ falls Versuch gescheitert
$\sigma :=$ zufällige Permutation von $\{1, \ldots, n\}$;
Weise dem Knoten i den Index $\sigma(i)$ zu;
Sei s zufälliger Knoten aus V;
$\overline{V} := V \cap \{x \in V \mid \sigma(v) \geqslant \sigma(s)\}$;
if es gibt Kreis C **in** $\overline{V}$**, der** s **enthält then**
  return C;
else
  return False;
end

Algorithmus 3 : Zufälliges Erzeugen eines Kreises

Die Suche nach einem Kreis in $\overline{V}$ kann dann durch eine Tiefensuche durchgeführt werden.

3.1.2 Minimum-Turn Cycle-Partition [MTCP]

Für MTCP ist die Zulässigkeitsbedingung komplex (Theorem 1.18). Man kann sich aber behelfen, indem man zufällige Kreise erzeugt und sie unter der Bedingung der Disjunktheit zu den bisherigen Kreisen zufügt. Nach Definition des Problems ist ein Graph genau dann zulässig, wenn er als disjunkte Vereinigung von Kreisen geschrieben werden kann.

```
Input : Höhe h, Breite b, Anzahl von Kreisen c
Result : Ein zufälliges MTCP-zulässiges h × b-Gitter, das durch zufälliges Erzeugen von
         c Kreisen konstruiert wurde, oder ∅, falls Versuch fehlgeschlagen
used := ∅;
num := 0;
circ := ∅;
while num < c do
    G := vollständiges h × b-Gitter;
    G := G−used;
    if G ist kreisfrei then
        return False;
    end
    C := erzeugezufälligenKreis(G);
    for v Knoten in C do
        used := used + v;
    end
    num := num + 1;
    circ := circ + C;
end
return circ;
```

Algorithmus 4 : Zufälliges Erzeugen einer MTCP-Instanz mit vorgegebenen Parametern

3.1.3 Minimum-Turn Cycle-Cover [MTCC]

Für MTCC ist jedes Gitter ohne isolierte Knoten zulässig. Daher können zufällige Instanzen für das MTCC-Problem recht einfach erzeugt werden:

3.1.4 Minimum-Turn Cycle-Cover No U-Turns

Für MTCCNU ist jedes Gitter mit Minimalgrad 2 zulässig. Der Algorithmus zur Erzeugung von Instanzen dieses Problems ist dem Algorithmus zur Erzeugung von MTCC-Instanzen sehr ähnlich (Algorithmus 6).

3.1.5 Angular und MinBends TSP

Für TSP-Probleme gibt es keine Zulässigkeitsbeschränkungen, zufällige Instanzen können also einfach durch Erzeugung von zufälligen Koordinatenpaaren in einer umgebenden Box erzeugt werden.

3.2 Lösungsmethoden

In diesem Abschnitt werden verschiedene Heuristiken für das MTCC-Problem mit und ohne U-Turns vorgestellt.

Input : Höhe h, Breite b, Wahrscheinlichkeit $p \in [0,1]$
Result : Eine zufälliges $h \times b$-Gitter ohne isolierte Knoten, oder $\emptyset$, falls Versuch fehlgeschlaegen
G := leeres $h \times b$-Gitter;
For$1 \leqslant i \leqslant h, 1 \leqslant j \leqslant b$ k := Zufallszahl aus $[0,1]$;
if $k \leqslant p$ **then**
  Erzeuge Knoten in G an Position (i,j);
end
while $\exists$ **isolierter Knoten** v **in** G **do**
  w := zufälliger potentieller Nachbar von v;
  füge Knoten w zum Graphen G hinzu;
end
return G;

Algorithmus 5 : Zufälliges Erzeugen einer MTCC-Instanz mit vorgegebenen Parametern

Input : Höhe h, Breite b, Wahrscheinlichkeit $p \in [0,1]$
Result : Eine zufälliges $h \times b$-Gitter ohne isolierte Knoten
G := leeres $h \times b$-Gitter;
for $1 \leqslant i \leqslant h, 1 \leqslant j \leqslant b$ **do**
  k := Zufallszahl aus $[0,1]$;
  if $k \leqslant p$ **then**
    Erzeuge Knoten in G an Position (i,j);
  end
end
while $\exists$ **Knoten** v **in** G **mit** $\deg(v) < 2$ **do**
  w := zufälliger zusätzlicher Nachbar von v;
  füge Knoten w zum Graphen G hinzu;
end
return G;

Algorithmus 6 : Zufälliges Erzeugen einer MTCCNU-Instanz mit vorgegebenen Parametern

3.2.1 Minimum-Turn Cycle-Partition

Die Polynomialität des MTCP-Problems wurde oben gezeigt. Man könnte nun den dort beschriebenen Algorithmus zur Lösung des Problems benutzen. Bei der Analyse des ganzzahligen Problems zeigt sich jedoch ein interessanter Effekt. In allen getesteten Instanzen war bereits die Lösung der LP-Relaxation ganzzahlig. Daher wurde auf die Implementierung des Algorithmus verzichtet.

3.2.2 Minimum-Turn Cycle-Cover

Mit der IP-Formulierung werden durch CPLEX Instanzen etwa bis zur Größe 50×50 optimal gelöst. Für größere Instanzen passiert es oft, dass vor Erschöpfen des Arbeitsspeichers keine zulässige Lösung gefunden wird. Hier hilft eine Startheuristik.

3.2.3 Startheuristik für das MTCC-Problem

Input : Gitter $G = (V, E)$
Result : zulässige Lösung des MTCC, falls existent
$\mathcal{C} = \emptyset$;
$T = V$;
while es gibt Knoten $v \in T$ **do**
  H = {K Kreise C, die v überdecken};
  foreach Kreis C **in** H **do**
    ratio(C) = #{von C erstmals überdeckte Knoten}/Knickkosten(C)
  end
  $C = \arg\min \text{ratio}(C)$;
  $\mathcal{C} = \mathcal{C} \cup C$;
  $T = T \setminus \{v \mid v \in C\}$;
end

Algorithmus 7 : Startheuristik für das MTCC

Die Menge der alles überdeckenden Kreise findet sich dann in $\mathcal{C}$.

3.2.4 Minimum-Turn Cycle-Cover No U-Turns

Schnittebenen nach Fischetti und Lodi Gomory-Chvátal-Schnitte wurden in Abschnitt 2.5.3 definiert.

Das Paper von Fischetti und Lodi[FL07] beschreibt ein Verfahren, möglichst gute Gomory-Chvátal-Schnitte mit Hilfe von gemischt-ganzzahliger Programmierung zu finden.

Gesucht ist also ein Schnitt $a^\top x \leqslant \alpha_0$ mit $\alpha = \lfloor u^\top A \rfloor$ und $\alpha_0 = \lfloor u^\top b \rfloor$ für ein $u \in \mathbb{R}^+$. Dabei möchte man den Unterschied zwischen der linken Seite und der rechten Seite möglichst groß machen.

Das kann man also MIP mit einer echten Ungleichung schreiben:

$$\begin{aligned}
\max \quad & a^\top x^* - \alpha_0 && (3.1)\\
& \alpha_j \leqslant u^\top A_j, j = 1, \ldots, n && (3.2)\\
& \alpha_0 > u^\top b - 1 && (3.3)\\
& u_i \geqslant 0, i = 1, \ldots, n && (3.4)\\
& \alpha, \alpha_0 \text{ ganzzahlig} && (3.5)
\end{aligned}$$

Weil Variablen mit $x_j^* = 0$ nichts zur Verletzung beitragen, kann man sie im Modell weglassen und dann als $\alpha_j := u^\top A_j$ wieder rekonstruieren. Außerdem kann man $u_i < 1$ annehmen, weil die Cuts sonst dominiert werden (man kann u durch den fraktionalen Anteil ersetzen).

Das von Fischetti und Lodi vorgeschlagene MIP-Modell sieht folgendermaßen aus:

$$\begin{aligned}
\max \quad & \sum_{j \in J(x^*)} \alpha_j x_j^+ - \alpha_0 && (3.6)\\
& f_j = u^\top A_j - \alpha_j, j \in J(x^*) && (3.7)\\
& f_0 = u^\top b - \alpha_0 && (3.8)\\
& 0 \leqslant f_j \leqslant 1 - \delta, j \in J(x^*) \cup \{0\} && (3.9)\\
& 0 \leqslant u_i \leqslant 1 - \delta, i = 1, \ldots, m && (3.10)\\
& \alpha_j \text{ganzzahlig}, j \in J(x^*) \cup \{0\} && (3.11)
\end{aligned}$$

wobei $J(x^*) := \{j \in \{1, \ldots, n\} : x_j^* > 0\}$

Dabei sind die Slackvariablen $f_j = u^\top A_j - \lfloor u^\top A_j \rfloor$ explizit gemacht. Die Wahl von δ kann für numerische Probleme sorgen, in [FL07] wird $\delta = 0.01$ vorgeschlagen.

Um die Auswahl gleichwertiger Schnitte im MIP zu beeinflussen, wird noch vorgeschlagen, die Zielfunktion zum Ausdruck

$$\max \left(\sum_{j \in J(x^*)} \alpha_j x_j^* - \alpha_0 \right) - \sum_{i=1}^{m} \omega_i u_i$$

zu ändern. Dabei wird $\omega_i := 10^{-4}$ gesetzt. Dies führt dazu, dass im gefundenen Cut möglichst wenige Variablen einen Wert ungleich 0 besitzen.

Iteriertes Runden Es wurde ein Verfahren entwickelt, um das IP mit Hilfe einer Reihe von LPs zu lösen, deren Ergebnisse gerundet werden und in das nächste LP einfließen.

Ein dabei zu lösendes Subproblem ist die Zerlegung einer gefundenen LP-Lösung in eine Summe von Kreisen mit konstantem Gewicht.

Lemma 3.1. *Algorithmus 8 terminiert in Polynomialzeit.*

Beweis. Jeder Schleifendurchlauf benötigt soviel Zeit wie die Bestimmung der Kreisfreiheit eines Graphen, das geht zum Beispiel in $\mathcal{O}(m \cdot \log n)$. In jedem Schritt wird eine Knickvariable auf 0

Input : LP-Lösung des MTCC-Problems
Result : Menge C von Kreisen, die die Variablenwerte aus der LP-Lösung induzieren
circ := $\emptyset$;
while es gibt in LP-Lösung Variable mit Wert größer 0 **do**
  Sei δ kleinster positiver Wert aus der LP-Lösung;
  Suche Kreis C mit Gewicht δ der Knick mit diesem Wert enthält; Entferne Kreis aus LP-Lösung;
end
return circ;

Algorithmus 8 : Kreiszerlegung für LP-Lösung

gesetzt, die Schleife wird also höchstens $10n$ mal durchlaufen (die Anzahl der Knicke pro Knoten, 4 U-Turns und 6 andere). Da die Graderhaltung durch die entsprechende Bedingung im LP gilt und bei jedem Schleifendurchlauf durch das Entfernen eines Kreises erhalten wird, lassen sich solange Knicke finden, bis alle Variablen auf 0 gesetzt sind. □

Input : Instanz G des MTCC/MTCCNU-Problems
Result : Lösung des MTCC/MTCCNU-Problems auf G
Für jeden Knick K sei $C(K) := 0$;
Sei L := LP-Relaxation des Ausgangsproblems;
Sei L_{opt} := optimale Lösung der LP-Relaxation;
while L_{opt} **ist nicht ganzzahlig do**
  for Knick K **in** L_{opt} **do**
    $L_{opt}(K) := L_{opt}(K) - C(K)$;
  end
  Berechne Kreiszerlegung D von L_{opt} mit Algorithmus 8;
  Sei C_{max} der Kreis in D mit größtem Wert; **foreach Knick** K **in** C_{max} **do**
    $C(K) := C(K) + C_{max}(K)$;
  end
  foreach Knick K **do**
    L := L + „$x_K >= C(K)$“
  end
  Sei L_{opt} := optimale Lösung von L;
end

Algorithmus 9 : Iteriertes Runden für das MTCC/MTCCNU-Problem

Algorithmus 9 wurde hier mit aufgenommen, auch wenn die Terminierung nicht klar ist.

Heuristik durch Matching von maximal kreisunabhängigen Mengen Zwei Knoten im Gitter G heißen **rechteckunabhängig**, wenn sie nicht auf einem Rechteck, also einem Kreis mit genau 4 Knicken, liegen.

```
Input : Instanz G des MTCC/MTCCNU-Problems
Result : Lösung des MTCC/MTCCNU-Problems auf G
Sei I := ∅;
used := ∅;
circ := ∅;
while es gibt Knoten ν in G−used, der von I rechteckunabhängig ist do
    I := I + ν;
    Sei K vollständiger Graph auf I;
    Setze c(ν, w) :=Knickkosten eines minimalen Weges von ν nach w für ν, w ∈ I;
    Berechne kostenminimimales 2-Matching M von K;
    foreach Kreis C in M do
        Berechne knickminimale Wege in G zwischen den Endpunkten von Kanten in C;
        Füge die knickminimalen Wege zu einem Kreis C_min zusammen;
        Setze circ := circ + C_min;
    end
    foreach Knoten ν in C_min do
        used := used + ν;
    end
end
```

3.3 Rechenexperimente

Minimum-Turn Cycle-Partition

Mit Hilfe des oben beschriebenen Algorithmus wurde eine Testumgebung mit 1248 Instanzen erzeugt. Diese sind allesamt Teilmengen des 50×50-Gitters.

Es folgt eine Tabelle mit den gefundenen Eigenschaften der Testumgebung:

Anzahl	min(Höhe)	max(Höhe)	min(Breite)	max(Breite)
1248	4	50	4	50
min(Kreisanzahl)	max(Kreisanzahl)	$\overline{c}_{opt}$	$\sigma_{c_{opt}}$	
1	132	$110\frac{1}{6}$	83.86	

Die Zielfunktionswerte lagen zwischen 4 und 478.

3.4 Lösungsgalerie

In diesem Abschnitt werden einige der berechneten Lösungen gezeigt.

Alle beschriebenen Algorithmen wurden zu Testzwecken implementiert, es war aber keiner der Algorithmen kompetitiv zum Branch and Bound-Verfahren von CPLEX. Daher wurde auf die Aufnahme von Ergebnissen verzichtet. Dafür folgen nun einige Bilder von Lösungen.

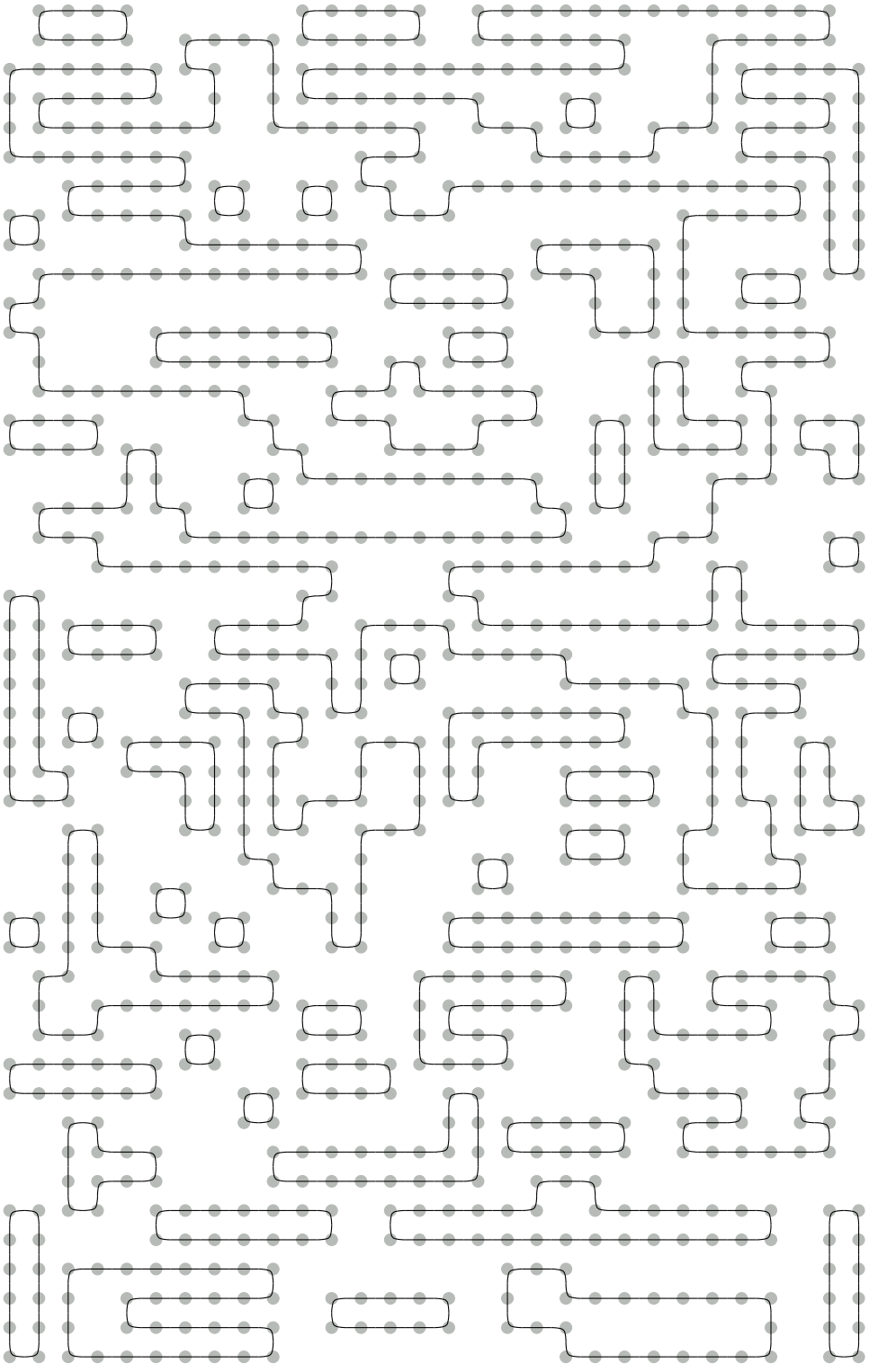

Abbildung 3.1: Optimallösung des MTCP-Problems für ein 47×30-Gitter mit $c_{opt} = 432$

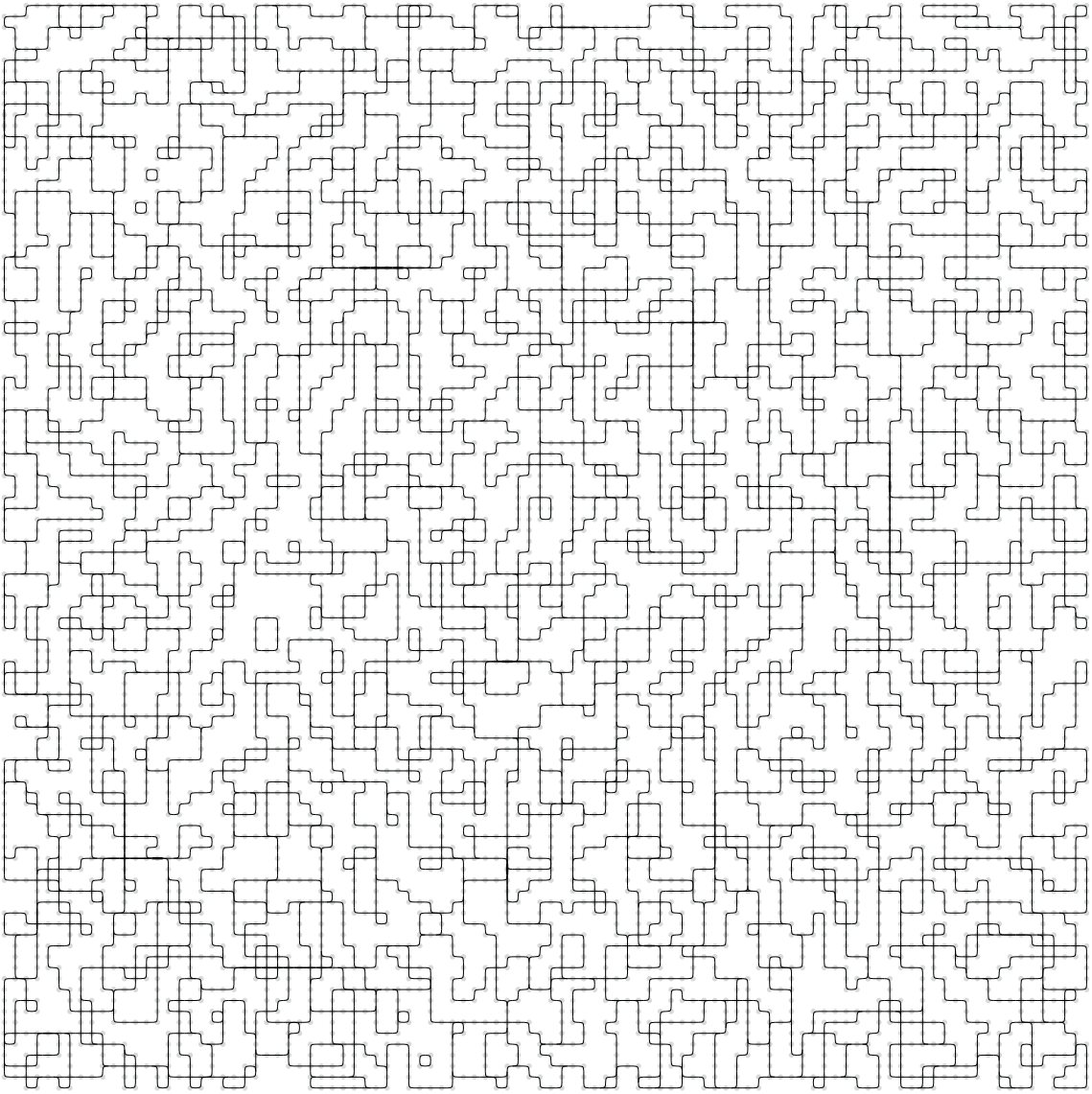

Abbildung 3.2: Lösung einer MTCCNU-Instanz, 100×100-Gitter, $c = 3538$, $LB = 3536$, Rechenzeit 1 Stunde.

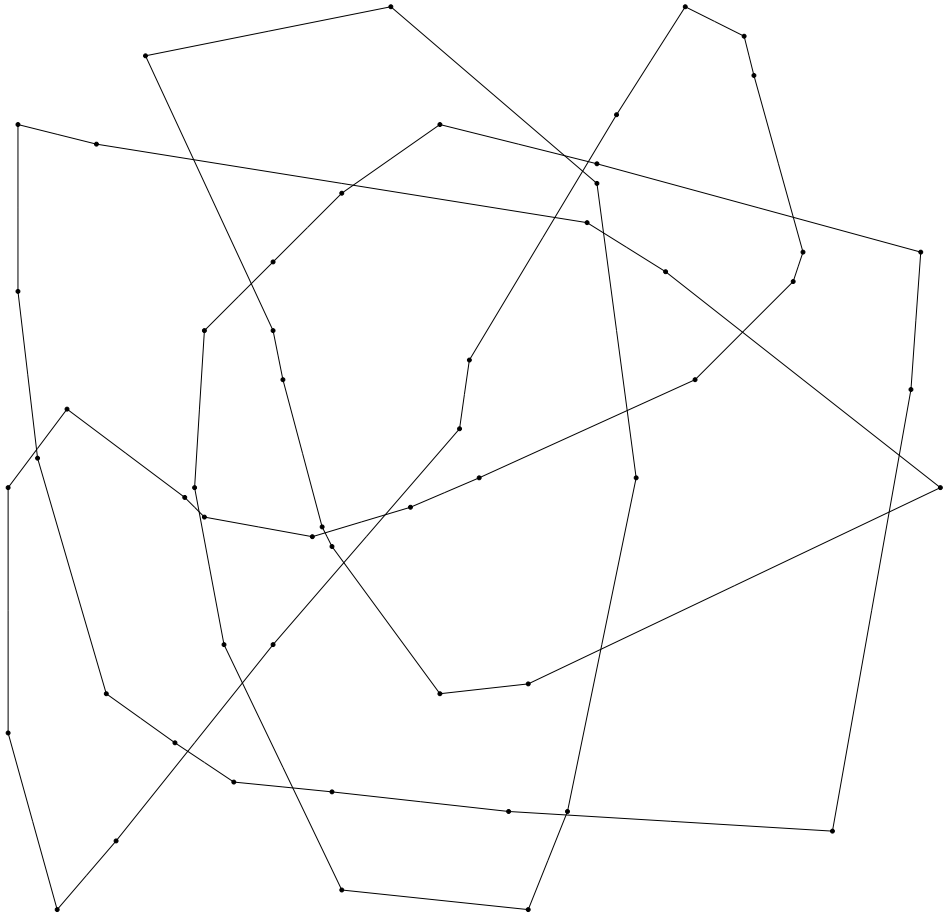

Abbildung 3.3: Lösung einer Angular CCP-Instanz, 55 Punkte, c $\cong$ 1739.37, LB $\cong$ 1636.15, Rechenzeit 1 Stunde, GAP $\cong$ 3.22%

4 Zusammenfassung und Ausblick

In der Arbeit wurden Überdeckungsprobleme mit auf Winkeln basierenden Kosten untersucht. Im ersten Kapitel wurde einen Überblick über die bereits vorhandene Literatur gegeben und die Probleme im Rahmen einer Klassifikation eingeordnet. Im zweiten Kapitel wurden eigene Untersuchungen durchgeführt. So wurde beispielsweise erläutert, wie die klassifizierten Probleme als ganzzahlige Programme modelliert werden können. Nach einer Reihe von Versuchen, die letztendlich nicht zum Ziel führten, konnte gezeigt werden, dass das MTCP-Problem einen polynomialen Algorithmus erlaubt. Unter dem Gesichtspunkt, dass das das lineare Programm zu diesem Problem in allen Versuchen bereits eine ganzzahlige Lösung liefert, ist das Resultat nicht überraschend. Schließlich konnten kleinere Resultate zu einigen der untersuchten Probleme erzielt werden. Für das letzte Kapitel wurde einige Bilder von Lösungen angefertigt, die teilweise optimal sind, teilweise zumindest von beweisbarer Güte.

Viele Fragen in Bezug auf Winkelkosten mussten offen bleiben. Die bedeutendsten werden nun aufgelistet:

- Die Komplexität des MTCC- sowie des MTCCNU-Problems sind weiterhin uneingeordnet.
- Es ist nicht bekannt, ob diese Probleme ein polynomialzeitliches Approximationsschema erlauben. Ein solches ist nur für eine Kombination von Winkel- und Längenkosten bekannt.
- Bei Überdeckungsproblemen für Punkte in der Ebene sind keine Approximationsalgorithmen mit konstanten Faktoren bekannt, alle bekannten Algorithmen liefern logarithmische Faktoren.
- Viele weitere Problemvarianten sind bisher gar nicht untersucht worden.

> *Wir stehen selbst enttäuscht und sehn betroffen*
> *Den Vorhang zu und alle Fragen offen.*
>
> — B.Brecht

Symbolverzeichnis

$\cup$	Vereinigung von Mengen
$\dot{\cup}$	disjunkte Vereinigung von Mengen
Σ_n	Menge aller Permutationen von n Elementen
$\subset$	Echte Teilmenge
$\subseteq$	Teilmenge oder gleich
$G[X]$	von der Knotenmenge X induzierter Subgraph von G

Literaturverzeichnis

[ABD+05] Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005. http://arxiv.org/pdf/cs/0309014v2.

[AKMS97] Alok Aggarwal, Sanjeev Khanna, Rajeev Motwani, and Baruch Schieber. The angular-metric traveling salesman problem. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 221–229, 1997. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.2692&rep=rep1&type=pdf.

[AL93] Muhammad H. Alsuwaiyel and D. T. Lee. Minimal link visibility paths inside a simple polygon. *Comput. Geom. Theory Appl.*, 3(1):1–25, 1993. http://www.sciencedirect.com/science/journal/09257721.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, February 1993.

[AMP03] E.M. Arkin, J.S.B. Mitchell, and C.D. Piatko. Minimum-link watchman tours. *Information Processing Letters*, 86:203–207(5), 31 May 2003. http://dx.doi.org/10.1016/S0020-0190(02)00502-1.

[Bal95a] Ross Baldick. Refined proximity and sensitivity results in linearly constrained convex separable integer programming. *Linear Algebra and its Applications*, 226-228:389 – 407, 1995.

[Bal95b] Ross Baldick. A unified approach to polynomially solvable cases of integer "non-separable" quadratic optimization. *Discrete Appl. Math.*, 61(3):195–212, 1995. http://www.sciencedirect.com/science/journal/0166218X.

[Bar86] Francisco Barahona. A solvable case of quadratic 0-1 programming. *Discrete Appl. Math.*, 13(1):23–26, 1986. http://www.sciencedirect.com/science/journal/0166218X.

[BBD+08] Sergey Bereg, Prosenjit Bose, Adrian Dumitrescu, Ferran Hurtado, and Pavel Valtr. Traversing a set of points with a minimum number of turns. *Discrete and Computational Geometry*, 2008. http://dx.doi.org/10.1007/s00454-008-9127-1.

[Bel50] Hans-Boris Belck. Reguläre Faktoren von Graphen. *Journal für die reine und angewandte Mathematik*, 188:228–252, 1950. http://www.digizeitschriften.de/index.php?id=loader&tx_jkDigiTools_pi1[IDDOC]=507134.

[Blo91] Johannes Blömer. Computing sums of radicals in polynomial time. pages 670–677, 1991. http://dx.doi.org/10.1109/SFCS.1991.185434.

[CCPS98] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. John Wiley and Sons, New York, 1998.

[CF96] A. Caprara and M. Fischetti. {0-1/2}-Chvátal-Gomory cuts. *Mathematical Programming*, (74):221–223, 1996.

[CFN85] Gérard Cornuéjols, Jean Fonlupt, and Denis Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33(1):1–27, 1985. http://www.springerlink.com/content/qp7040036335w104/.

[CMMS02] Angel Corberán, Rafael Martí, Eulalia Martínez, and David Soler. The rural postman problem on mixed graphs with turn penalties. *Comput. Oper. Res.*, 29(7):887–903, 2002. http://linkinghub.elsevier.com/retrieve/pii/S0305054800000915.

[CN86] Wei-Pang Chin and Simeon Ntafos. Optimum watchman routes. In *SCG '86: Proceedings of the second annual symposium on Computational geometry*, pages 24–33, New York, NY, USA, 1986. ACM. http://doi.acm.org/10.1145/10515.10518.

[Coo71] Stephen Arthur Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press. http://dx.doi.org/10.1145/800157.805047.

[Coo03] Stephen Arthur Cook. The importance of the P versus NP question. *Journal of the ACM*, 50(1):27–29, 2003. http://doi.acm.org/10.1145/602382.602398.

[Die05] Reinhard Diestel. *Graph Theory*. Springer, 2005.

[FL07] Matteo Fischetti and Andrea Lodi. Optimizing over the first chvátal closure. *Math. Program.*, 110(1):3–20, 2007.

[FT87] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. http://doi.acm.org/10.1145/28869.28874.

[Ger31] Semyon Aranovich Gershgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk. USSR Otd. Fiz.-Mat. Nauk*, (7):749–754, 1931.

[GH62] A. Ghouila-Houri. Caractérisation des matrices totalement unimodulaires. *C. R. Acad. Sci. Paris*, 254:1192–1194, 1962.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[Hel91] Martin Held. *On the computational geometry of pocket machining*. Springer-Verlag New York, Inc., New York, NY, USA, 1991. http://www.springerlink.com/content/h6k007p63577/.

[HR68] Peter L. Hammer and Sergiu Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Springer, 1968.

[HS90] Dorit S. Hochbaum and J. George Shanthikumar. Convex separable optimization is not much harder than linear optimization. *J. ACM*, 37(4):843–862, 1990. http://doi.acm.org/10.1145/96559.96597.

[HSS92] Dorit S. Hochbaum, Ron Shamir, and J. George Shanthikumar. A polynomial algorithm for an integer quadratic non-separable transportation problem. *Math. Program.*, 55:359–371, 1992. http://www.springerlink.com/content/r1721238413v0311/.

[IPS82] Alon Itai, Christos H. Papadimitriou, and Jayme L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982. http://dx.doi.org/10.1137/0211056.

[KKM94] E. Kranakis, Krizanc, and D. Meertens. Link length of rectilinear hamiltonian tours in grids. *Ars Combinatoria*, 38:177–192, 1994. http://www.scs.carleton.ca/~kranakis/Papers/guard.pdf.

[KV00] Bernhard Korte and Jens Vygen. *Combinatorial Optimization - Theory and Algorithms*. Springer, 2000.

[LU97] William Lenhart and Christopher Umans. Hamiltonian cycles in solid grid graphs. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 496, Washington, DC, USA, 1997. IEEE Computer Society.

[MA89] Renato D. Monteiro and Ilan Adler. Interior path following primal-dual algorithms. part ii: Convex quadratic programming. *Mathematical Programming*, 44(1):43–66, May 1989.

[Mag59] Kh. Maghout. Sur la determination des nombres de stabilité et du hombres chromatiques d'un graph. *C. R. Acad. Sci.*, (25):3522–3523, 1959.

[Mcb82] Richard Mcbride. Controlling left and u-turns in the routing of refuse collection vehicles. *Computers & Operations Research*, 9(2):145 – 152, 1982. http://dx.doi.org/10.1016/0305-0548(82)90013-2.

[Mit99] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[MRW90] Joseph S. B. Mitchell, Günter Rote, and Gerhard Woeginger. Minimum-link paths among obstacles in the plane. *Algorithmica*, 6:308–317, 1990. http://doi.acm.org/10.1145/98524.98537.

[MT82] Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1:194–197, 1982. http://theory.stanford.edu/~megiddo/pdf/complexity%20of%20locating%20linear%20facilities.pdf.

[Onn07] Shmuel Onn. Convex discrete optimization. *Encyclopedia of Optimization*, 2009:513, 2007. http://arxiv.org/abs/math/0703575.

[Pap95] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1995.

[Rei79] John H. Reif. Complexity of the mover's problem and generalizations. In *SFCS '79: Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 421–427, Washington, DC, USA, 1979. IEEE Computer Society. http://www.cs.duke.edu/~reif/paper/FOCSmovers.pdf.

[Sch98] Alexander Schrjiver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998.

[Sch03] Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*, volume A. Springer, 2003.

[Sey80] P. D. Seymour. Decomposition of regular matroids. *Journal of Combinatorial Theory*, 28:305–359, 1980. http://dx.doi.org/10.1016/0095-8956(80)90075-1.

[Sha75] Michael Ian Shamos. Geometric complexity. In *STOC '75: Proceedings of seventh annual ACM symposium on Theory of computing*, pages 224–233, New York, NY, USA, 1975. ACM. http://doi.acm.org/10.1145/800116.803772.

[Sur86] Subhash Suri. A linear time algorithm for minimum link paths inside a simple polygon. *Comput. Vision Graph. Image Process.*, 35(1):99–110, 1986. http://www.sciencedirect.com/science/journal/0734189X.

[SW00] Cliff Stein and David P. Wagner. Approximation algorithms for the minimum bends traveling salesman problem. In *Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization*, 2000. http://www.springerlink.com/content/l6wej0t4gew4bjrb/fulltext.pdf.

[WN88] Laurence A. Wolsey and George L. Nemhauser. *Integer and Combinatorial Optimization*. John Wiley and Sons, 1988.

[Zie95] Günter M. Ziegler. *Lectures on Polytopes*. Springer, 1995.