MATH FAKULTÄT FÜR MATHEMATIK

Institut für Mathematische Optimierung

Beschleunigte Berechnungen gradbeschränkter aufspannender Bäume in planaren Graphen mittels erweiterter Formulierungen

Master-Arbeit zur Erlangung des akademischen Grades Master of Science

vorgelegt von

Stephan Sorgatz (181853)

am

31. August 2012

Erstgutachter: Prof. Dr. Volker Kaibel

Zweitgutachter: Prof. Dr. Sebastian Sager

Inhaltsverzeichnis

Symbolverzeichnis

$\binom{V}{2}$	Menge aller zweielementigen Knotenteilmengen der Knotenmenge V
$G = (V, E)$	ungerichteter Graph mit endlicher Knotenmenge V und Kantenmenge $E \subseteq \binom{V}{2}$
$K_n = (V_n, E_n)$	der vollständige Graph mit n Knoten
$K_{n,m}$	vollständiger bipartiter Graph mit n und m Knoten
$D = (V, A)$	gerichteter Graph mit endlicher Knotenmenge V und Bogenmenge $A \subseteq V \times V$
$\mathcal{C}(G)$	Menge der Kreise $C \subseteq E$ eines Graphen $G = (V, E)$
$\delta(v)$	Menge der zum Knoten $v \in V$ inzidenten Kanten $\{u, v\} \in E$ in einem Graphen $G = (V, E)$
$\delta(v)^{ein}$	Menge der in den Knoten $v \in V$ eingehenden Bögen $(u, v) \in A$ in einem Digraphen $D = (V, A)$
$\chi(S)$	charakteristischer Vektor der Menge $S \subseteq X$ für eine endliche Grundmenge X
$x(F)$	$\sum_{e \in F} x_e$ für eine endliche Menge M, $x \in \mathbb{R}^M$ und $F \subseteq M$
$\text{conv}(X)$	Konvexe Hülle aller Vektoren in der endlichen Menge X
$E(U)$	Menge der Kanten, die innerhalb der Knotenteilmenge $U \subseteq V$ verlaufen
$\delta(U)$	Menge der Kanten mit einem Endknoten in der Knotenteilmenge $U \subseteq V$ und dem anderen in $V \setminus U$
$G^* = (V^*, E^*)$	Dualgraph des planaren Graphen $G = (V, E)$
F	Menge der Gebiete eines planaren Graphen
$G(U)$	Untergraph $(U, E(U))$ des Graphen $G = (V, E)$ für die Knotenteilmenge $U \subseteq V$
$F(U)$	Menge der Gebiete des planaren Untergraphen $(U, E(U))$ des Graphen $G = (V, E)$ für die Knotenteilmenge $U \subseteq V$

1 Einführung

Die mathematische Optimierung ist ein Bereich der Mathematik mit starkem Bezug zur Anwendung. Insbesondere die Betrachtung von kombinatorischen und graphenbasierten Problemstellungen nimmt eine zentrale Rolle bei logistischen Prozessen ein – von der Planung von Produktionsprozessen, über Aufgaben an Drehkreuzen wie Flughäfen, bis hin zu Routenberechnungen und Ausfallplänen von Computernetzwerken und den Strecken des öffentlichen Nah- und Fernverkehrs. Dabei ist es in der alltäglichen Behandlung dieser Anwendungsgebiete oft sehr wichtig, flexibel und zeitnah auf auftretende Veränderungen zu reagieren bzw. in regelmäßigen, nah aufeinanderfolgenden Intervallen neue Lösungen produzieren zu können. Um diesen Vorgaben zu entsprechen, ist ein performanter Lösungsprozess notwendig.

Die vorliegende Arbeit beschäftigt sich mit den angesprochenen kombinatorischen Problemstellungen in Form von (gradbeschränkten) aufspannenden Bäumen in *planaren* Graphen. Dabei steht die Untersuchung und der Vergleich eines Separationsverfahrens und verschiedener erweiterter Formulierungen unterschiedlicher Größe im Mittelpunkt. Die Lösungszeiten und Solver-spezifische Parameter, wie die Anzahl hinzugefügter Schnittebenen bei der Separationsmethode und Branch-and-Bound-Verhalten des Solvers, sind die zentralen Vergleichskriterien. Hauptaussage wird die Überlegenheit der untersuchten erweiterten Formulierungen linearer Größe im Vergleich zu der Separationsmethode und der betrachteten erweiterten Formulierung kubischer Größe sein.

Als theoretische Einleitung zu den experimentellen Untersuchungen wird zunächst ein kurzer Überblick über grundlegende Begrifflichkeiten gegeben. Anschließend wenden wir uns der Formulierung der zu untersuchenden Problemstellung zu. Das darauffolgende Kapitel widmet sich den verschiedenen Lösungsmethoden und deren Korrektheit. Dabei nutzen wir unter anderem das Konzept des *Dualgraphen* für einen planaren Graphen und gehen hierfür auf die Dualität der *Delaunay-Triangulierung* und des *Voronoi-Diagrammes* einer Punktmenge in der Ebene ein. Nach einem Exkurs zur Nicht-Ganzzahligkeit des Polytops der erweiterten Formulierung kubischer Größe stehen die implementierungstechnischen Grundlagen im Mittelpunkt. Dafür werden die zu Grunde liegenden Testinstanzen und die Implementierung der Lösungsmethoden beleuchtet. Die detaillierte Darstellung der Testergebnisse und anschließende Auswertung der Experimentalläufe ist Inhalt des vorletzten Kapitels, worauf ein kurzes Fazit folgt.

2 Theoretische Grundlagen

2.1 Planare Graphen

Um mit planaren Graphen arbeiten zu können, betrachten wir in diesem Abschnitt die notwendigen Grundlagen.

Definition 2.1. *Ein Graph wird planar genannt, wenn es möglich ist, ihn so in der Ebene oder auf einer zweidimensionalen Sphäre zu zeichnen, dass sich seine Kanten (außerhalb der Endpunkte) nicht schneiden. Die konkrete Darstellung eines Graphen in der Ebene oder auf einer Sphäre wird Einbettung genannt.*

Es ist jedoch nicht immer sofort ersichtlich, ob sich ein gegebener Graph, dessen Kanten sich (außerhalb der Endpunkte) schneiden, nicht doch planar zeichnen lässt. In Abbildung 1 a) und b) sind exemplarisch zwei verschiedene Einbettungen eines Graphen dargestellt. Wir schauen uns daher eine andere Charakterisierung planarer Graphen an. Zunächst benötigen wir den Begriff des *Minors*.

Definition 2.2. *Ein Graph H wird Minor des Graphen G genannt, wenn sich H aus G durch eine Folge von Knoten- und Kantenlöschungen und Kontraktionen von Kanten erzeugen lässt. Wobei bei Kontraktion der Kante $e = \{u, v\} \in E$ diese und die anliegenden Knoten u und v inklusive ihrer inzidenten Kanten gelöscht werden. Anschließend wird ein neuer Knoten w erstellt, sowie Kanten zu allen Knoten, die adjazent zu u oder v waren.*

Damit können wir folgende Aussage von Kuratowski [17] und Wagner [27] festhalten:

Satz 2.3. *Ein ungerichteter Graph ist genau dann planar, wenn er weder K_5 noch $K_{3,3}$ als Minor enthält.*

Die Frage, ob es eine planare Einbettung für einen gegebenen Graphen gibt, lässt sich komplexitätstheoretisch effizient beantworten. Dazu formulieren wir folgenden Satz von Hopcroft und Trajan [12].

Satz 2.4. *Es gibt einen Algorithmus mit linearer Laufzeit, um eine planare Einbettung für einen Graphen zu finden oder zu entscheiden, dass er nicht planar ist.*

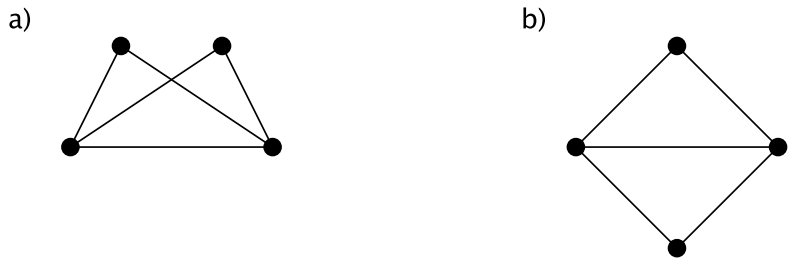

Abbildung 1: a) nicht-planare und b) planare Einbettung eines Graphen.

Im Laufe der Arbeit werden wir gelegentlich aus einem Graphen $G = (V, E)$ einen Digraphen konstruieren. Dieser ist – falls nicht anders bemerkt – definiert als $D = (V, A)$ mit:

$$A = \{(u, v) \in V \times V : \{u, v\} \in E\}.$$

2.2 Das (DC)MST-Problem

Im Folgenden sei $G = (V, E)$ ein Graph mit Knotenmenge V und Kantenmenge E. Eine Kantenteilmenge $W = \{\{v_0, v_1\}, ..., \{v_{l-1}, v_l\}\} \subseteq E$ heißt v_0-v_l-*Weg*, falls $v_0, ..., v_l$ paarweise verschieden sind. Wenn es für alle Knoten $u \neq v \in V$ einen u-v-Weg in der Kantenmenge E gibt, so heißt G *zusammenhängend*. Gilt für eine Kantenteilmenge $C = \{\{v_0, v_1\}, ..., \{v_{l-1}, v_l\}\} \subseteq E$, dass $v_0 = v_l$ ist und $v_0 = v_l, v_1, ..., v_{l-1}$ paarweise verschieden sind, so nennen wir C einen *Kreis*. Die Menge aller Kreise für einen Graphen $G = (V, E)$ bezeichnen wir mit $\mathcal{C}(G)$. Ist $\mathcal{C}(G)$ leer, so heißt G *kreisfrei*. Die Menge der mit dem Knoten $v \in V$ inzidenten Kanten wird als $\delta(v)$ definiert. Unter dem Grad eines Knotens $v \in V$ verstehen wir die Anzahl der mit v inzidenten Kanten, also $|\delta(v)|$. Im gerichteten Graphen erfolgt die Notation der Menge, der in einen Knoten $v \in V$ eingehenden Bögen, analog via $\delta^{ein}(v)$.

Als nächstes halten wir die Begriffe des *Waldes* und *Baumes* für einen Graphen als Definitionen fest.

Definition 2.5. *Ist die Kantenteilmenge $W \subseteq E$ eines ungerichteten Graphen $G = (V, E)$ kreisfrei, so nennen wir W einen Wald. Ist zusätzlich der Graph (V, W) zusammenhängend, so ist W ein Baum.*

Definition 2.6. *Für einen Graphen $G = (V, E)$ heißt die Kantenteilmenge $W \subseteq E$ aufspannender Wald, falls sie kreisfrei und der Grad jedes Knotens $v \in V$ in dem indizierten Untergraphen (V, W) echt größer als Null ist. Ist (V, W) zusammenhängend, so nennen wir W einen aufspannenden Baum.*

In diesem Kontext betrachten wir den Begriff der *Arboreszenz*, der später genutzt wird.

Definition 2.7. *Ein Digraph $D = (V, A)$ heißt Arboreszenz, wenn er keine antiparallelen Bögen hat, der D zugrunde liegende ungerichtete Graph ein aufspannender Baum ist und $|\delta(v)^{ein}| \leq 1$ für alle $v \in V$ gilt. In einer Arboreszenz gibt es genau einen Knoten $r \in V$ mit $|\delta(v)^{ein}| = 0$. Er heißt die Wurzel von D.*

Im Zentrum dieser Arbeit steht das *Minimum-Spanning-Tree-Problem* (MST-Problem). Es besteht darin, für einen zusammenhängenden Graphen $G = (V, E)$ einen aufspannenden Baum zu finden, der minimal bezüglich einer gegebenen Gewichtsfunktion $w : E \to \mathbb{R}$ ist. Um das Problem zu bearbeiten, werden wir unter anderem auf dem *Spanning-Tree-Polytop* des Graphen operieren, das als $P_{span}(G)$ bezeichnet wird, wobei

$$P_{span}(G) := \text{conv}\{\chi(F) \in \{0, 1\}^E : F \subseteq E \text{ ist aufspannender Baum von } G\}.$$

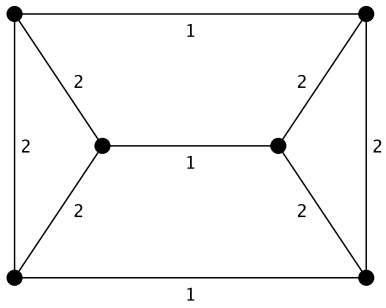

Abbildung 2: Planarer Graph mit gebrochenzahliger Optimallösung des DCMST-Problems bei Gradschranke 2 mit Wert 9 und ganzzahliger Optimallösung mit Wert 10.

Schränkt man die Problematik dahingehend ein, nur aufspannende Bäume als Lösung zuzulassen, deren Knoten einen bestimmten Maximalgrad $k \in \mathbb{N}$ nicht überschreiten, so geht man über zum *Degree-Constrained-Minimum-Spanning-Tree-Problem (DCMST-Problem)*. Auch hierfür einigen wir uns auf eine Notation für das zugehörige Polytop:

$$P^k_{tree}(G) := \{x \in \mathbb{R}^E : x \in P_{span}(G), x(\delta(v)) \leq k \ \ \forall\, v \in V\}.$$

Bei $P_{span}(G)$ handelt es sich als konvexe Hülle von charakteristischen Vektoren um ein ganzzahliges Polyeder, d.h. alle Ecken sind ganzzahlig. Bei der Definition von $P^k_{tree}(G)$ handelt es sich nicht mehr um eine konvexe Hülle charakteristischer Vektoren und die Ganzzahligkeitseigenschaft geht im allgemeinen verloren. Anderenfalls wäre für $k = 2$ das NP-schwere *Traveling-Salesman-Problem* (s. Abschnitt 2.3 und [14]) mit Hilfe der in Abschnitt 2.4 dargestellten Separationsmethode in polynomieller Zeit lösbar. Damit würde folglich $P = NP$ gelten. Betrachten wir als Ausruck der Nicht-Ganzzahligkeit den in Abbildung 2 dargestellten planaren Graphen von Korte und Vygen [3]. Die ganzzahlige Optimallösung des für diesen Graphen formulierten DCMST-Problems mit Gradschranke 2 nimmt den Wert 10 an. Setzt man jedoch $x_e = 0.5$ für alle Kanten e mit Gewicht 2 und $x_e = 1$ für alle Kanten mit Gewicht 1, so erhält man die gebrochenzahlige Optimallösung mit Wert 9. Wir schauen uns nun folgende äußere Beschreibung eines Polyeders $P_{tree}(G)$ für einen Graphen G nach Edmonds [7] an:

$$\begin{aligned} P_{tree}(G) := \{x \in \mathbb{R}^E : {} & x(E) = |V| - 1, && (2.1)\\ & x(E(U)) \leq |U| - 1 \ \ \forall\, \emptyset \neq U \subsetneq V, && (2.2)\\ & x \geq \mathbf{0}\}. \end{aligned}$$

Dabei gilt $P_{tree}(G) = P_{span}(G)$ (s. Edmonds [7]) für einen Graphen $G = (V, E)$, womit wir eine äußere Beschreibung für das Spanning-Tree-Polytop und ebenso für $P^k_{tree}(G)$ erhalten, die dann zusätzlich die Berücksichtigung der Gradschranken fordert. Mit diesen äußeren Beschreibungen können wir nun komfortabel lineare Programmierung betreiben. Es ist uns möglich dem System Ungleichungen in Form von Gradbeschrän-

kungen oder im Rahmen eines Schnittebenenverfahrens hinzuzufügen. Für den charakteristischen Vektor $x \in P_{tree}(G)$ eines aufspannenden Baumes $T \subseteq E$ fordert die Nebenbedingung (2.1) den Zusammenhang des Graphen (V, T), was aufgrund der Kreisfreiheit des Baumes schnell zu sehen ist. Sie wird daher als *Zusammenhangsbedingung* bezeichnet. Die Bedingungen (2.2) nennt man auch *Cycle-Elimination-Constraints*, da bei Verletzung für eine Knotenteilmenge $U \subseteq V$ $(x(E(U)) \geq |U|)$ die Existenz eines Kreises $C \subseteq U$ resultieren würde.

2.3 Das Traveling-Salesman-Problem

Dem DCMST-Problem ähnlich ist das Traveling-Salesman-Problem (TSP). Es besteht darin, in einem vollständigen Graphen K_n mit $n \geq 3$, bei gegebener Gewichtsfunktion $w : E \to \mathbb{R}$, einen *Hamilton-Kreis* w-minimaler Länge zu finden. Ein Hamilton-Kreis ist ein Kreis $C \subseteq E$, der jeden Knoten $v \in V$ genau einmal enthält. Als *Traveling-Salesman-Polytop* für K_n bezeichnet man die konvexe Hülle der charakteristischen Vektoren (in $\mathbb{R}^{E_n}$) der Hamilton-Kreise in K_n. Wir definieren analog Schrijver [26] die konvexe Hülle der charakteristischen Vektoren der Hamilton-Kreise in einem ungerichteten Graphen $G = (V, E)$ als das Traveling-Salesman-Polytop für G und notieren es via $P_{TSP}(G)$. Eine äußere Beschreibung polynomieller Größe von $P_{TSP}(G)$ ist nicht bekannt und auch nicht zu erwarten. Da das TSP NP-schwer ist, würde solch eine Beschreibung NP = co-NP bedeuten. Denn mit Hilfe des Farkas' Lemmas [8] hätte man ein polynomiell überprüfbares Zertifikat für die Erfüllbarkeit des komplementären TSP – also der Frage, ob es keinen Hamilton-Kreis in G gibt. Es sind jedoch Relaxierungen für das Traveling-Salesman-Polytop bekannt, wie beispielsweise das Ungleichungssystem

$$x(\delta(U)) \geq 2 \qquad \text{für alle } \emptyset \neq U \subsetneq V \tag{2.3}$$

$$x(\delta(v)) = 2 \qquad \text{für alle } v \in V \tag{2.4}$$

$$x \geq \mathbf{0}. \tag{2.5}$$

Wir können dabei die Ungleichungen (2.3) ersetzen durch

$$x(E(U)) \leq |U| - 1 \qquad \text{für alle } \emptyset \neq U \subsetneq V. \tag{2.6}$$

Es gilt durch Addition von (2.4) für alle $v \in U$

$$2x(E(U)) = \sum_{v \in U} x(\delta(v)) - x(\delta(U)) = 2|U| - x(\delta(U)).$$

Das Polytop, das durch die Ungleichungen (2.4), (2.5) und (2.6) definiert ist, nennen wir *Subtour-Eliminations-Polytop* $(P_{SEP}(G))$ für den Graphen $G = (V, E)$. Dabei heißen die Bedingungen (2.6) in diesem Kontext auch *Subtour-Eliminations-Bedingungen*, da sie eine kleinere, nicht alle Knoten besuchende Rundtour verhindern. Im Abschnitt 2.5.3 werden wir eine erweiterte Formulierung für $P_{SEP}(G)$ herleiten und später deren Lösungsverhalten im Rahmen experimenteller Untersuchungen betrachten.

2.4 Schnittebenenverfahren

2.4.1 Begriffserklärung

Bei Optimierungsproblemen kann es geschehen, dass sie durch sehr viele (meist heißt das exponentiell viele) Ungleichungen beschrieben werden. Dieses Merkmal hat oft einen negativen Einfluss auf die Laufzeit bei der Operation über dem Polyeder mittels LP-Techniken. Eine Möglichkeit diese Problematik zu umgehen, ist die Arbeit mit einem Schnittebenenverfahren. Da auch die Polyeder $P^{(k)}_{tree}(G)$ von exponentiell vielen Cycle-Elimination-Constraints beschrieben werden, ist solch ein Schnittebenenverfahren eine Methode dieser Arbeit das MST- bzw. DCMST-Problem zu bearbeiten. Dabei startet man mit einem Polyeder, bei dem Nebenbedingungen von $P^{(k)}_{tree}(G)$ weggelassen wurden. Das initiale lineare Programm sieht wie folgt aus:

$$\min\{\langle w, x\rangle : x(E) = |V| - 1,\ x_e \geq 0 \text{ für alle } e \in E,\ (x(\delta(v)) \leq k \text{ für alle } v \in V)\}. \tag{2.7}$$

Ist eine Optimallösung x^* des LPs gefunden, so tritt einer der drei Fälle auf:

(i) x^* ist Inzidenzvektor eines (gradbeschränkten) aufspannenden Baumes oder

(ii) x^* verletzt eine Nebenbedingung von $P^{(k)}_{tree}(G)$ oder

(iii) x^* ist nicht-ganzzahlige Lösung von $P^{(k)}_{tree}(G)$.

Mit dem Eintreten von (i) ist eine ganzzahlige Optimallösung für das (DC)MST-Problem gefunden und wir sind fertig. Bei (ii) werden eine oder mehrere solcher verletzter Ungleichungen zum System hinzugefügt und es wird ein neuer Optimierungslauf gestartet. Man versucht dabei, möglichst stark verletzte Ungleichungen hinzuzufügen, um – geometrisch gesprochen – einen möglichst großen Teil vom Polyeder abzuschneiden, inklusive der gefundenen Lösung. An Fall (iii) könnte sich die Ausführung eines Branch-and-Bound-Verfahrens mit unterer Schranke $w(x^*)$ anschließen oder es werden (auch in Kombination) Integer-Cuts zum LP hinzugefügt. Das in dieser Arbeit verwendete *SCIP-Framework* (Solving Constrained Integer Programs, s. Achterberg [1]) ist ein Branch-and-Cut-Framework, das selbstständig und auf nutzerdefinierte Arten Cuts zum LP hinzufügt, um ganzzahlige Lösungen zu finden – in Kombination mit einem Branch-and-Bound-Verfahren.

2.4.2 Eine Separationsmethode für das (gradbeschränkte) Spanning-Tree-Polytop

Eine Lösungsmethode für das (DC)MST-Problem ist es, mit der Optimierung über dem LP (2.7) zu beginnen und nach und nach verletzte Nebenbedingungen in Form von Schnittebenen hinzuzufügen. Da die (Grad-,) Zusammenhangs- und Nichtnegativitätsbedingungen schon im initialen Programm enthalten sind, erfolgt die Separation über (möglichst stark) verletzte Cycle-Elimination-Constraints. Um solche verletzten Nebenbedingungen zu finden, betrachten wir nach folgendem Hilfssatz eine Aussage von Schrijver [26], die dies darauf reduziert einen kapazitätsminimalen Schnitt zu finden.

Satz 2.8. *Es seien der Digraph $D = (V, A), x \in \mathbb{Q}_+^A, y \in \mathbb{Q}^V$ sowie disjunkte Mengen $S, T \subseteq V$ gegeben. Dann findet man in polynomieller Zeit eine Menge U mit $T \subseteq U \subseteq V \setminus S$, die $x(\delta^{in}(U)) + y(U)$ minimiert.*

Beweis: Wir erweitern den Digraphen D durch die Knoten s und t sowie Bögen (s, v) für alle $v \in V$ mit $y_v > 0$ und Bögen (v, t) für alle $v \in V$ mit $y_v < 0$. Somit erhalten wir den Digraphen $D' = (V \cup \{s, t\}, A')$. Wir definieren die Kapazitätsfunktion c auf A' mit:

$$\begin{aligned} c(u, v) &:= x(u, v) && \text{für } (u, v) \in A, \\ c(s, v) &:= y_v && \text{für } (s, v) \in A', \\ c(v, t) &:= -y_v && \text{für } (v, t) \in A'. \end{aligned}$$

Sei weiter $\kappa := -c(\delta_{A'}^{in}(t))$. Dann ist

$$\begin{aligned} c(\delta_{A'}^{in}(U \cup \{t\})) &= x(\delta_A^{in}(U)) + \sum_{\substack{v \in U \\ y_v > 0}} y_v - \sum_{\substack{v \in V \setminus U \\ y_v < 0}} y_v \\ &= x(\delta_A^{in}(U)) + \sum_{v \in U} y_v - \sum_{\substack{v \in V \\ y_v < 0}} y_v \\ &= x(\delta_A^{in}(U)) + y(U) - \kappa \end{aligned}$$

für jedes $U \subseteq V$. Somit ist das Minimieren von $x(\delta_A^{in}(U)) + y(U)$ reduziert auf das Finden eines kapazitätsminimalen $(S \cup \{s\}) - (T \cup \{t\})$-Schnittes in D' und daher in polynomieller Zeit realisierbar.

□

Damit erhalten wir folgendes Korollar, das uns das Finden einer maximal verletzten Ungleichung von $P_{tree}^{(k)}(G)$ sichert. Man beachte dabei den Übergang von einem ungerichteten Graphen zu einem Digraphen, der wie in Abschnitt 2.1 erfolgt.

Korollar 2.9. *Gegeben sei der Graph $G = (V, E)$ und der Vektor $x \in \mathbb{Q}_+^E$. Man kann in polynomieller Zeit entscheiden, ob $x \in P_{tree}^{(k)}(G)$ gilt oder eine maximal verletzte Cycle-Elimination-Constraint des Polyeders finden.*

Beweis: Definiere $y_v := 2 - x(\delta(v))$ für alle $v \in V$. Dann ist

$$2(x(E(U)) - |U|) = \sum_{v \in U} x(\delta(v)) - x(\delta(U)) - 2|U| = -x(\delta(U)) - y(U). \tag{2.8}$$

Jede Menge $U \subseteq V$, die $x(\delta(U)) + y(U)$ minimiert, maximiert also $x(E(U)) - |U|$. Nach Satz 2.8 finden wir solch ein U in polynomieller Zeit. Ist $x(E(U)) \leq |U| - 1$, so gilt $x \in P_{tree}^{(k)}(G)$. Ansonsten wird durch U diese maximal verletzte Ungleichung definiert, die wir als Nebenbedingung zum zu lösenden LP hinzufügen.

□

2.5 Erweiterte Formulierungen

Wir haben gesehen, wie man der Problematik (exponentiell) großer Beschreibungen eines Polyeders mit Hilfe eines Schnittebenenverfahrens begegnen kann. Eine andere Möglichkeit stellen *erweiterte Formulierungen* dar.

Definition 2.10. *Eine erweiterte Formulierung eines Polyeders $P \subseteq \mathbb{R}^n$ ist eine äußere Beschreibung $Q = \{y \in \mathbb{R}^d : Ay \leq b\}$ mit einer linearen Projektion $p : \mathbb{R}^d \to \mathbb{R}^n$, für die $p(Q) = P$ gilt.*

Die Idee ist es, den exponentiellen Charakter auf Kosten zusätzlicher Variablen zu eliminieren. Man bearbeitet dann das neue höherdimensionale Problem mittels LP-Techniken und erhält eine Lösung des Originalproblems via Projektion auf die Variablen des Originalproblems. Die Anzahl der Nebenbedingungen, die mit Ungleichheit formuliert sind, bezeichnen wir als *Größe der erweiterten Formulierung*. Im Folgenden werden drei erweiterte Formulierungen unterschiedlicher Größe vorgestellt.

2.5.1 Eine erweiterte Formulierung kubischer Größe für das (gradbeschränkte) Spanning-Tree-Polytop

Wir schauen uns in diesem Abschnitt eine erweiterte Formulierung für das (DC)MST-Problem nach Martin [18] und Kaibel et al. [13] an. Für die folgenden Ausführungen soll $G = K_n$ für $n \in \mathbb{N}$ gelten. Um die Überlegungen auf einen allgemeinen Graphen $G = (V, E)$ zu übertragen, kann man für $e \in E_n \setminus E$ die Variable x_e auf den Wert null fixieren und somit auf einer Seite des Polytops, das aus der unten stehenden erweiterten Formulierung resultiert, arbeiten.

Wir führen nun zusätzliche $z_{u,v,w}$-Variablen für alle paarweise verschiedenen $u, v, w \in V$ ein. Wobei jeder aufspannende Baum $T \in E_n$ durch einen Binärvektor $y = (x, z) \in \mathbb{R}^d$ mit $x = \chi(T)$ repräsentiert wird. Dabei gilt $z_{u,v,w} = 1$ für paarweise verschiedene $u, v, w \in V$, genau dann, wenn die Kante $\{u, v\}$ in T enthalten ist und w in der Zusammenhangskomponente von v in $T \setminus \{u, v\}$ liegt.

Wir schauen uns nun das Ungleichungssystem an, das gegeben ist durch

$$x_{\{u,v\}} - z_{u,v,w} - z_{v,u,w} = 0 \qquad \text{für alle } u, v, w \in V \text{ paarweise verschieden,} \tag{2.9}$$

$$x_{\{u,v\}} + \sum_{w \in V \setminus \{u,v\}} z_{u,w,v} = 1 \qquad \text{für alle } u, v \in V \text{ paarweise verschieden,} \tag{2.10}$$

$$x(E) = |V| - 1, \tag{2.11}$$

$$x \geq \mathbf{0} \quad \text{und} \quad z \geq \mathbf{0} \tag{2.12}$$

und definieren das Polytop

$$Q_{tree}(G) := \{(x, z) \in \mathbb{R}^d : (x, z) \text{ erfüllt (2.9) bis (2.12)}\}.$$

Satz 2.11. *Es sei $p : \mathbb{R}^d \to \mathbb{R}^E$ die Projektion auf die x-Variablen. Dann gilt $p(Q_{tree}(G)) = P_{tree}(G)$.*

Beweis: Es sei für einen aufspannenden Baum $T \subseteq E$ der Vektor $y = (x, z)$ mit $x = \chi(T)$ und z, wie oben konstruiert, gegeben. Bedingung (2.9) fordert für alle Knotenpaare $u, v \in V$ mit $u \neq v$, dass bei Existenz der Kante $\{u, v\} \in T$ jeder Knoten $w \notin \{u, v\}$ mit genau einem der beiden Knoten u, v in einer Zusammenhangskomponente in $T \setminus \{u, v\}$ liegt. Die Bedingung (2.10) sichert, dass für jedes Knotenpaar $u, v \in V$ mit $u \neq v$ genau ein u-v-Weg existiert, entweder direkt durch eine Kante oder über einen Weg $W = (u, w, \dots, v)$. Daher und aufgrund der Konstruktion von z und der Kantenanzahl des Baumes T erfüllt der Vektor y die Bedingungen (2.9) bis (2.12).

Um die Inklusion $p(Q_{tree}(G)) \subseteq P_{tree}(G)$ zu zeigen, genügt es, die Gültigkeit der Cycle-Elimination-Constraints für beliebiges $y = (x, z) \in Q_{tree}(G)$ nachzuweisen. Sei dazu $U \subseteq V$ Knotenteilmenge. Ist $|U| = 2$ (also $U = \{u, v\}$), dann folgt aus (2.10) und (2.12): $x(U) = x_{\{u,v\}} \leq 1 = |U| - 1$. Es sei also $|U| \geq 3$. Summiert man die Bedingungen (2.9) für alle paarweise verschiedenen $u, v, w \in U$, so erhält man

$$\begin{aligned}
&\sum_{\substack{u,v\in U\\ u\neq v}} \sum_{w\in U\setminus\{u,v\}} x_{\{u,v\}} = \sum_{\substack{u,v\in U\\ u\neq v}} \sum_{w\in U\setminus\{u,v\}} (z_{u,v,w} + z_{v,u,w}) \\
\Leftrightarrow &(|U|-2) \sum_{\substack{u,v\in U\\ u\neq v}} x_{\{u,v\}} = \sum_{\substack{u,v\in U\\ u\neq v}} \sum_{w\in U\setminus\{u,v\}} 2z_{u,v,w} \\
\Leftrightarrow\ &2(|U|-2)x(E(U)) = \sum_{\substack{u,v\in U\\ u\neq v}} \sum_{w\in U\setminus\{u,v\}} 2z_{u,v,w} \\
\Leftrightarrow\ &(|U|-2)x(E(U)) = \sum_{\substack{u,v\in U\\ u\neq v}} \sum_{w\in U\setminus\{u,v\}} z_{u,w,v}.
\end{aligned}$$

Wegen Bedingung (2.10) gilt für die rechte Seite der vorigen Zeile

$$\begin{aligned}
\sum_{\substack{u,v\in U\\ u\neq v}} \sum_{w\in U\setminus\{u,v\}} z_{u,w,v} &\leq \sum_{\substack{u,v\in U\\ u\neq v}} \sum_{w\in V\setminus\{u,v\}} z_{u,w,v} \\
&= \sum_{\substack{u,v\in U\\ u\neq v}} (1 - x_{\{u,v\}}) \\
&= |U|(|U|-1) - 2x(E(U)).
\end{aligned} \tag{2.13}$$

Zusammen gilt damit

$$\begin{aligned}
(|U|-2)x(E(U)) &= \sum_{u,v\in U, u\neq v} \sum_{w\in U\setminus\{u,v\}} z_{u,w,v} \\
\Leftrightarrow \qquad |U|x(E(U)) &\leq |U|(|U|-1) \\
\Leftrightarrow \qquad x(E(U)) &\leq |U|-1.
\end{aligned}$$

Demnach ist $x(E(U)) \leq |U| - 1$ für alle $U \subseteq V$ und damit ist die Behauptung gezeigt.

□

Wir haben eine erweiterte Formulierung der Größe $O(|V|^3)$ für $P_{tree}(G)$ und erhalten eine erweiterte Formulierung für $P^k_{tree}(G)$ mit dem Polytop

$$Q^k_{tree}(G) := \{(x,z) \in \mathbb{R}^d : (x,z) \in Q_{tree}(G) \text{ und } x(\delta(v)) \leq k \text{ für alle } v \in V\}$$

und wiederum der Projektion auf die x-Variablen. Im Folgenden werden wir uns auf diese erweiterte Formulierung mittels des Kürzels *CEFMST* (Cubic Extended Formulation Minimum Spanning Tree) beziehen. Wie oben erwähnt wird der Übergang zu einem allgemeinen Graphen $G = (V, E)$ vollzogen, indem für die Kanten $e \in E_n \setminus E$ die x-Variable mit $x_e = 0$ gesetzt wird, wodurch auch die der Kante zugehörigen z-Variablen laut (2.9) auf null fixiert werden. Dadurch arbeiten wir dann auf einer Seite des Polyeders für den vollständigen Graphen und es ändert sich die Komplexität der erweiterten Formulierung auf $O(|V||E|)$.

Bemerkung 2.12. *Die Zusammenhangsbedingung $x(E) = |V| - 1$ kann für $Q_{tree}(G)$ weggelassen werden. Wählt man nämlich im obigen Beweis $U = V$, so gilt in (2.13) Gleichheit und man erhält die Zusammenhangsbedingung aus den Bedingungen (2.9) und (2.10).*

2.5.2 Eine erweiterte Formulierung linearer Größe für das (gradbeschränkte) Spanning-Tree-Polytop von planaren Graphen

Um eine andere erweiterte Formulierung für das (DC)MST-Problem in planaren Graphen nach Williams [28] vorzubereiten, kommen wir zu dem Begriff des *Dualgraphen*. Dieses Konzept beruht auf der Beziehung zwischen einem Graphen (später in diesem Zusammenhang auch *Primalgraph* genannt) und seinem Dualgraphen.

Bei einem planaren Graphen unterteilen seine Ecken und Kanten die Ebene in f Gebiete. Dabei sind $f - 1$ Gebiete von Kanten und Ecken begrenzt, von denen wir sagen, dass sie in dem Gebiet enthalten sind. Genau eines, das sogenannte *äußere Gebiet* ist unbeschränkt. Zwei Gebiete heißen *benachbart*, wenn es eine gemeinsame Kante gibt, die beide Gebiete enthalten. Nach Eulers Formel gilt für jeden zusammenhängenden planaren Graphen $G = (V, E)$ mit seiner Menge von Gebieten F die Gleichung: $|V| + |F| - |E| = 2$.

Die Korrektheit ist mit einem einfachen Induktionsargument zu überprüfen. Man überlege sich, dass jeder zusammenhängende planare Graph durch eine der beiden folgenden Operationen aus dem Graphen, der nur aus einem Knoten besteht, gewonnen werden kann. Für diesen einelementigen Graphen gilt Eulers Formel offensichtlich.

(i) Hinzufügen eines Knotens, der über eine neue Kante mit einem bereits vorhandenen Knoten verbunden wird.

(i) Hinzufügen einer Kante, die zwei bereits vorhandene Knoten miteinander verbindet. Hier ist auch die Konstruktion einer Schleife, also einer Kante, die einen Knoten mit sich selbst verbindet, möglich.

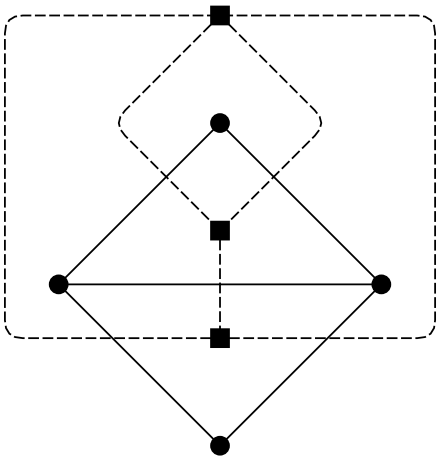

Abbildung 3: Planarer Graph mit runden Knoten und durchgezogenen Kanten und sein Dualgraph mit eckigen Knoten und gestrichelten Kanten.

Durch Schritt (i) wird die Anzahl der Knoten und die Anzahl der Kanten um eins erhöht, weshalb die Gleichung erfüllt bleibt. Nach Hinzufügen einer Kante in Schritt (ii) erhöht sich die Anzahl der Kanten und die der Gebiete um eins. Dadurch wird die Gleichung auch beibehalten.

Nun können wir für einen planaren Graphen $G = (V, E)$ seinen ebenfalls planaren Dualgraphen $G^* = (V^*, E^*)$ konstruieren (s. Abb. 3). Dafür halten wir die Einbettung des Graphen fest und erstellen für jedes Gebiet von G, inklusive dem äußeren, einen Dualknoten v^*. Die dualen Kanten erhalten wir, indem für jede Kante $e \in E$ genau eine Dualkante e^* erzeugt wird, die die Dualknoten, die aus den angrenzenden Gebieten hervorgegangen sind, verbindet, weshalb die Kantenanzahl in Primal- und Dualgraph gleich ist. Kanten, die inzident mit einen Knoten vom Grad eins sind, definieren eine Schleife im Dualgraphen. Es gilt nach Konstruktion $|V| = |F^*|$ und $|F| = |V^*|$ sowie $(G^*)^* = G$.

Betrachten wir nun für $G = (V, E)$ einen aufspannenden Baum $T_p \subseteq E$. Wir wählen eine Kantenteilmenge $T_d \subseteq E^*$ so, dass bei einem primal-dualen Kantenpaar $e \in E, e^* \in E^*$ genau dann $e^* \in T_d$ gilt, wenn $e \notin T_p$ ist. Die Graphen (V, T_p) und (V^*, T_d) nennen wir *komplementär*. Mit Eulers Formel in der Form $|E| = (|V| - 1) + (|F| - 1)$ folgt $|T_d| = |V^*| - 1$ und wir würden mit einer Kreisfreiheitseigenschaft von T_d feststellen, dass es sich dabei um einen aufspannenden Baum für G^* handelt.

Nun würde aber jeder Kreis in T_p bewirken, dass der Komplementärgraph nicht zusammenhängend ist. Anschaulich gäbe es sonst im Komplementärgraphen keinen Weg zwischen dem Knoten, der aus dem äußeren Gebiet resultiert, und den Knoten, die von den Gebieten innerhalb des Kreises stammen. Ein vollständiger Beweis ist durch Deo [5] gegeben. Fordern wir also Zusammenhang von T_p, so erhalten wir Kreisfreiheit in T_d. Also gibt es für jeden aufspannenden Baum im Primalgraphen einen komplementären aufspannenden Baum im Dualgraphen. Graphisch verdeutlicht wird dies in Abbildung 4.

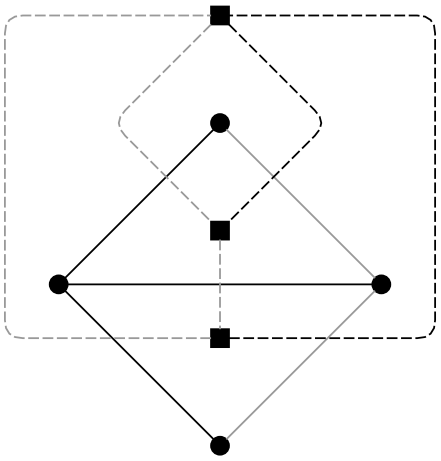

Abbildung 4: Aufspannender Baum T_p eines planaren Graphen und komplementärer aufspannender Baum T_d des Dualgraphen. Die schwarzen durchgezogenen und gestrichelten Linien sind die Kanten aus T_p bzw. T_d. Die grauen Linien sind die Kanten aus E bzw. E^*, die nicht für den jeweiligen Baum gewählt wurden.

Dieses Konzept der komplementären Bäume fließt in das folgende Ungleichungssystem nach Williams [28] und Pashkovich [24] ein. Dafür definieren wir bei gegebenem Graphen $G = (V, E)$ für jede Kante $e \in E$ mit $e = \{u, v\}$ Variablen $y_{e,v}$ und $y_{e,u}$. Für jede dieser Variablen werden korrespondierende Variablen im Dualgraphen z_{e^*,v^*} und z_{e^*,u^*} erstellt, wobei die Kanten $e \in E$ und $e^* = \{u^*, v^*\} \in E^*$ ein primal-duales Kantenpaar bilden. Geometrisch interpretiert ist der Bogen der Variable $y_{e,v}$ in Richtung des Knotens v gerichtet. Analoges gilt für z_{e^*,v^*}.

Es seien ein planarer Graph $G = (V, E)$ und sein Dualgraph $G^* = (V^*, E^*)$ gegeben. Ein Knoten $v^*_{inf} \in V^*$ und ein in seinem Gebiet enthaltener Knoten $v_{inf} \in V$ werden als duale bzw. primale Wurzel ausgezeichnet. Zudem gelten die linearen Bedingungen

$$x_e + z_{e^*,v^*} + z_{e^*,u^*} = 1 \quad \text{für } u^*, v^* \in V^* \text{ mit } e^* = \{u^*, v^*\} \in E^* \text{ Dualkante von } e \in E, \tag{2.14}$$

$$(1 - x_e) + y_{e,v} + y_{e,u} = 1 \quad \text{für } u, v \in V \text{ mit } e = \{u, v\} \in E, \tag{2.15}$$

$$\sum_{e \in \delta(v^*)} z_{e^*,v^*} = 1 \quad \text{für } v^* \in V^* \setminus \{v^*_{inf}\}, \tag{2.16}$$

$$\sum_{e \in \delta(v)} y_{e,v} = 1 \quad \text{für } v \in V \setminus \{v_{inf}\}, \tag{2.17}$$

$$\sum_{e \in \delta(v^*)} z_{e^*,v^*} = 0 \quad \text{für } v^* = v^*_{inf}, \tag{2.18}$$

$$\sum_{e \in \delta(v)} y_{e,v} = 0 \quad \text{für } v = v_{inf}, \tag{2.19}$$

$$z \geq \mathbf{0} \quad \text{und} \quad y \geq \mathbf{0}. \tag{2.20}$$

Dabei fließt jede Schleife nur jeweils einmal in die Summen (2.14), (2.16) und (2.18) ein, um die im Beweis genutzte Eigenschaft der totalen Unimodularität beizubehalten. Wir wollen nun folgenden Satz analog Pashkovich [24] beweisen.

Satz 2.13. *Für jeden planaren Graphen $G = (V, E)$ stellt das lineare System (2.14) bis (2.20) zusammen mit der Projektion auf die x-Variablen eine erweiterte Formulierung für $P_{tree}(G)$ dar.*

Beweis: Zunächst zeigen wir, dass für jedes $x \in P_{tree}(G)$, wobei x charakteristischer Vektor eines aufspannenden Baumes $T \subseteq E$ von G ist, Variablen z und y existieren, die das obige Ungleichungssystem erfüllen. Dafür betrachten wir die Arboreszenzen $N \subseteq A$ und $N^* \subseteq A^*$, die durch den Baum T und sein komplementäres Gegenstück mit den jeweiligen Wurzelknoten $v_{inf} \in V$ und $v^*_{inf} \in V^*$ definiert sind. Dabei ist der Digraph (V, A) die gerichtete Variante von G. Die Variable $y_{e,v}$ mit $e = \{u, v\}$ soll den Wert eins annehmen, wenn der Bogen (u, v) zur Arboreszenz N gehört, sonst ist sie gleich null. Analog ist z_{e^*,v^*} mit $e^* = \{u^*, v^*\}$ eins, wenn $(u^*, v^*) \in N^*$, ansonsten null. Die Erfüllung der Bedingungen (2.14) bis (2.20) sieht man schnell.

Für jeden Punkt (x, y, z) aus dem durch (2.14) bis (2.20) gegebenen Polytop, soll nun $x \in P_{tree}(G)$ gelten. Zunächst ist offensichtlich $(x, y, z) \in [0, 1]^{5|E|}$. Darüber hinaus stellen wir die totale Unimodularität der Matrix, die aus den Bedingungen (2.14) bis (2.20) resultiert, fest. Überprüfen lässt sich dies mittels des Kriteriums von Ghouila-Houri [11]. Dafür müssen wir für jede gegebene Zeilenindexteilmenge J eine Aufteilung $J = J^+ \cup J^-$ mit $J^+ \cap J^- = \emptyset$ finden, sodass die Summe der Zeilen aus J^+ minus der Summe der Zeilen in J^- ein Vektor mit $\{-1, 0, 1\}$ Einträgen ist. Man beachte, dass die z- und y-Variablen einmal in (2.14) und (2.15), sowie einmal in (2.16) bis (2.19) enthalten sind. Die x-Variablen treten in (2.14) und (2.15) jeweils einmal mit gegensätzlichen Koeffizienten auf. Wir wählen also alle Indices $j \in J$, die zu (2.14) und (2.15) gehören in J^+, die restlichen in J^-. Die Aufteilung der Zeilen der Nichtnegativitätsbedingungen ergibt sich daraus. Daher sind die Ecken des Polytops 0/1-wertig und es ist $x = \chi(T)$ für ein $T \subseteq E$. Es bleibt zu zeigen, dass es sich bei T um einen aufspannenden Baum handelt.

Dafür definieren wir Arboreszenzen $N \in E$ und $N^* \in E^*$ via

$$N = \{(u, v) \in V \times V : y_{e,v} = 1, e = \{u, v\}, e \in E\},$$
$$N^* = \{(u^*, v^*) \in V^* \times V^* : z_{e^*,v^*} = 1, e^* = \{u^*, v^*\}, e^* \in E^*\}.$$

Gemäß (2.16) und (2.17) gibt es für jeden Knoten $v \in V \backslash \{v_{inf}\}$ und $v^* \in V^* \backslash \{v^*_{inf}\}$ genau einen eingehenden Bogen in N bzw. N^*. Wegen der Bedingungen (2.18) und (2.19) existiert kein eingehender Bogen in den Arboreszenzen für v_{inf} und v^*_{inf}. Da also die Anzahl der Bögen stimmt, genügt es, die (gerichtete) Kreisfreiheit von N zu zeigen, um die gewünschte Spannbaumeigenschaft von T zu erhalten. Dafür definieren wir das *Innere* eines Kreises in G bzw. G^* als das Gebiet, das den Wurzelknoten v^*_{inf} bzw. v_{inf} nicht enthält. Wir halten an dieser Stelle die Einbettung des Graphen G in der Ebene fest, die G^* induziert.

Es enthalte N nun einen gerichteten Kreis C_1. Mit den Bedingungen (2.14) und (2.15) folgt, dass die z-Variablen der dualen Bögen des Kreises C_1 alle den Wert null annehmen. Das bedeutet, in N^* existieren

keine Bögen zwischen den Knoten im Inneren und denen im Äußeren des Kreises. Daraus folgt, N^* besitzt einen Kreis C_1^* im Inneren von C_1, da der Wurzelknoten v_{inf}^* nicht im Inneren des Kreises enthalten ist und somit wegen (2.16) für alle Knoten $v^* \in V^*$ im Inneren von C_1 ein eingehender Bogen existiert. Wiederum sind nun alle y-Variablen der primalen Bögen von C_1^* aufgrund von (2.14) und (2.15) auf null gesetzt. Iterativ erhalten wir eine unendliche Menge verschiedener Kreise $C_i \in \mathcal{C}(G), i \in \mathbb{N}$ in G (durch Identifizierung der Kanten in G mit den Bögen in N). Da jeder Kreis strikt im Inneren seines Vorgängers liegt, sind die Kreise paarweise kantendisjunkt. Im Widerspruch dazu ist jedoch die Anzahl von Kreisen in G endlich.

□

Wir erhalten also eine erweiterte Formulierung der Größe $4|E|$, auf die wir uns im weiteren Verlauf mittels des Kürzels *EFMST* (Extended Formulation Minimum Spanning Tree) beziehen. Den Übergang zum DCMST-Problem vollführen wir, indem wir zusätzlich die Einhaltung der Gradschranke fordern.

2.5.3 Eine erweiterte Formulierung linearer Größe für das Subtour-Elimination-Polytop von planaren Graphen

Wir wenden uns nun dem in Abschnitt 2.3 beschriebenen Subtour-Elimination-Polytop für $G = (V, E)$ mit $|V| \geq 3$ zu. Um uns die Ungleichungen für den folgenden Beweise noch einmal vor Augen zu führen, notieren wir sie an dieser Stelle erneut:

$$x(E(U)) \leq |U| - 1 \qquad \text{für alle } \emptyset \neq U \subsetneq V, \tag{2.21}$$

$$x(\delta(v)) = 2 \qquad \text{für alle } v \in V, \tag{2.22}$$

$$x \geq \mathbf{0}. \tag{2.23}$$

Für einen Graphen $G = (V, E), |V| \geq 3$ ist das Subtour-Eliminations-Polytop definiert via:

$$P_{SEP}(G) := \{x \in \mathbb{R}^E : x \text{ erfüllt (2.21) bis (2.23)}\}.$$

Um eine erweiterte Formulierung analog Pashkovich [24] zu entwickeln, nutzen wir für planare Graphen erneut das Konzept des Dualgraphen. Dafür definieren wir bei gegebenem planaren Graphen $G = (V, E)$ mit ausgezeichnetem Wurzelknoten $v_{inf} \in V$ Variablen z_{e^*,v^*} für jede Kante $e \in E$, $v_{inf} \notin e$ und Dualknoten $v^* \in V^*$, sodass $v^* \in e^*$. Dabei ist $e^* \in E^*$ Dualkante von $e \in E$. Wir betrachten das Ungleichungssystem

$$x_e + z_{e^*,v^*} + z_{e^*,u^*} = 1 \qquad \text{für } v_{inf} \notin e \text{ mit } e^* = \{u^*, v^*\} \in E^* \text{ Dualkante von } e \in E, \tag{2.24}$$

$$\sum_{e^* \in \delta(v^*)} z_{e^*,v^*} = 1 \qquad \text{für } v^* \in V^*, v_{inf} \notin v^*, \tag{2.25}$$

$$x(\delta(v)) = 2 \qquad \text{für alle } v \in V, \tag{2.26}$$

$$z \geq \mathbf{0}, \qquad x \geq \mathbf{0}. \tag{2.27}$$

Jede Schleife taucht nur einmal in den Summen (2.24) und (2.25) auf. Um zu zeigen, dass es sich bei dem Ungleichungssystem in Verbindung mit der Projektion auf die x-Variablen um eine erweiterte Formulierung für das Subtour-Elimination-Polytop handelt, benötigen wir zunächst die folgende Hilfsaussage.

Lemma 2.14. *Für jeden Graphen $G = (V, E)$ bilden die Ungleichungen (2.22), (2.23) und diejenigen Bedingungen (2.21), für deren Knotenteilmenge $U \subseteq V$ die Untergraphen $G(U)$ und $G(V \setminus U)$ zusammenhängend sind und U nicht den Knoten v_{inf} enthält, eine lineare Beschreibung von $P_{SEP}(G)$.*

Beweis: Aus Abschnitt 2.3 wissen wir bereits, dass die Bedingungen (2.21) für eine Knotenteilmenge $U \neq \emptyset$ äquivalent sind zu $x(\delta(U)) \geq 2$. Aufgrund der Symmetrie der Bedingung für die Mengen U und $V \setminus U$, können wir die Bedingungen weglassen, die den Knoten $v_{inf} \in V$ enthalten.

Ist der Graph $G(U)$ nicht zusammenhängend, existieren zwei Knotenteilmengen $U_1, U_2 \subseteq V$ mit

$$U = U_1 \cup U_2 \quad \text{mit } U_1, U_2 \neq \emptyset,\ U_1 \cap U_2 = \emptyset,\ \delta(U_1) \cap \delta(U_2) = \emptyset.$$

In dem Fall jedoch würden die Ungleichungen (2.21) für die Mengen U_1, U_2 die Ungleichung (2.21) für die ganze Menge U implizieren. Denn es ist

$$x(E(U)) = x(E(U_1)) + x(E(U_2)) \leq |U_1| - 1 + |U_2| - 1 = |U| - 2 < |U| - 1.$$

Analoges gilt, falls der Graph $G(V \setminus U)$ nicht zusammenhängend ist.

□

Mit dieser Aussage und der äußeren Beschreibung von $P_{tree}(G)$ für den Graphen $G = (V, E)$ aus Abschnitt 2.2 von Edmonds [7] lässt sich nachstehende Folgerung leicht überprüfen, die uns zusammen mit Satz 2.13 eine erweiterte Formulierung der Größe kleiner $4|E|$ für planare Graphen liefert.

Lemma 2.15. *Für jeden Graphen $G = (V, E)$ definiert das lineare System*

$$\begin{aligned} proj_{E'} x &\in P_{tree}(G'), \\ x(\delta(v)) &= 2 \text{ für alle } v \in V, \\ x &\geq \mathbf{0} \end{aligned}$$

das Subtour-Elimination-Polytops von G. Dabei ist $G' = (V', E')$ mit $V' = V \setminus \{v_{inf}\}$ und $E' = E(V')$.

Mit dem nachfolgenden Satz erhalten wir eine erweiterte Formulierung geringerer Größe für $P_{SEP}(G)$.

Satz 2.16. *Für jeden planaren Graphen $G = (V, E)$ bildet das lineare System (2.24) bis (2.27) in Verbindung mit der Projektion auf die x-Variablen eine erweiterte Formulierung für $P_{SEP}(G)$.*

Beweis: Zunächst wollen wir zeigen, dass jeder Punkt $x \in \mathbb{R}^E$, für den z-Variablen existieren, die (2.24) bis (2.27) erfüllen, die Ungleichungen (2.21) bis (2.23) respektiert und damit in $P_{SEP}(G)$ enthalten ist. Wir halten dafür die Einbettung von G in der Ebene fest.

Die Bedingungen (2.22) und (2.23) sind in beiden Systemen enthalten. Daher genügt es, die Gültigkeit der Subtour-Elimination-Constraints zu überprüfen. Nach obigem Lemma ist für eine Knotenteilmenge $U \subseteq V$, für die wir die Gültigkeit der Ungleichung (2.21) nachweisen wollen, der Knoten v_{inf} nicht in U enthalten und die Untergraphen $G(U)$ und $G(V \setminus U)$ sind zusammenhängend. Wir summieren nun die Bedingungen (2.24) über die Kanten in U

$$|E(U)| = \sum_{e \in E(U)} (x_e + \sum_{\substack{v^* \in V^*, \\ v^* \in e^*}} z_{e^*,v^*}) = x(E(U)) + \sum_{e \in E(U)} \sum_{\substack{v^* \in V^*, \\ v^* \in e^*}} z_{e^*,v^*}$$

und betrachten den Untergraphen $G(U)$ mit der Einbettung, die von G impliziert ist. Es sei $F^{'}$ die Menge der Gebiete von $G(U)$, die auch Gebiete von G sind. Also gilt

$$\begin{aligned} \sum_{e \in E(U)} \sum_{\substack{v^* \in V^*, \\ v^* \in e^*}} z_{e^*,v^*} &= \sum_{e \in E(U)} \sum_{\substack{v^* \in F', \\ v^* \in e^*}} z_{e^*,v^*} + \sum_{e \in E(U)} \sum_{\substack{v^* \in V^* \setminus F', \\ v^* \in e^*}} z_{e^*,v^*} \\ &\geq \sum_{e \in E(U)} \sum_{\substack{v^* \in F', \\ v^* \in e^*}} z_{e^*,v^*} = \sum_{v^* \in F'} \sum_{e^* \in \delta(v^*)} z_{e^*,v^*}. \end{aligned}$$

Aus Bedingung (2.25) folgt wegen $v_{inf} \notin U$:

$$\sum_{v^* \in F'} \sum_{e^* \in \delta(v^*)} z_{e^*,v^*} = |F^{'}|.$$

Wir bezeichnen die Menge der Gebiete des Untergraphen $G(U)$ mit $F(U)$ und bemerken, dass es höchstens ein Gebiet in $F(U)$ geben kann, das kein Gebiet von G ist. Anderenfalls wäre der Untergraph $G(V \setminus U)$ nicht zusammenhängend. Also gilt zusammen mit obigen Überlegungen

$$|E(U)| \geq x(E(U)) + |F^{'}| \geq x(E(U)) + |F(U)| - 1.$$

Da $G(U)$ zusammenhängend ist, können wir die Euler-Formel anwenden und erhalten

$$\begin{aligned} & |U| + |F(U)| - 2 \geq x(E(U)) + |F(U)| - 1 \\ \Leftrightarrow \quad & |U| - 1 \geq x(E(U)). \end{aligned}$$

Um die andere Inklusion zu zeigen, müssen für jedes $x \in P_{SEP}(G)$ z-Variablen existieren, die (2.24) bis (2.27) erfüllen. Dafür sei die Einbettung von G in der Ebene wieder fest.

Nach Satz 2.13 und Lemma 2.15 verwenden wir die z'_{e^*,v^*}-Variablen der erweiterten Formulierung von $P_{SEP}(G)$, die mit Hilfe der erweiterten Formulierung von $P_{tree}(G')$, mit $G' = G(V \setminus \{v_{inf}\})$, konstruiert wurde. Dabei wählen wir $v^{*'}_{inf}$ als Knoten, der aus dem Gebiet von $G^{'}$ hervorgeht, das den Knoten v_{inf} enthält. Wir fixieren anschließend z_{e^*,v^*} auf den Wert von $z^{'}_{e^*,v^{*'}_{inf}}$ der erweiterten Formulierung von $P_{tree}(G^{'})$, falls das Gebiet, aus dem v^* hervorgeht, den Knoten v_{inf} enthält. Anderenfalls setzen wir z_{e^*,v^*} auf den Wert der Variable $z^{'}_{e^*,v^*}$.

□

Wir haben demnach für jeden planaren Graphen $G = (V, E)$ eine erweiterte Formulierung von höchstens der Größe $3|E|$ gefunden. Im Folgenden wird sich auf diese erweiterte Formulierung der Ungleichungen (2.24) bis (2.27) mittels des Kürzels *EFSE* (Extended Formulation Subtour Elimination) bezogen.

2.6 Nichtganzzahligkeit einer erweiterten Formulierung

Wie in Abschnitt 2.2 erwähnt, handelt es sich bei $P_{tree}(G)$ für einen Graphen $G = (V, E)$ um ein ganzzahliges Polyeder. Wir wollen in diesem Kapitel die Ganzzahligkeit der oben betrachteten erweiterten Formulierung CEFMST betrachten, also untersuchen, ob das Polytop $Q_{tree}(G)$ nur ganzzahlige Ecken besitzt.

Dies geschah mit Hilfe experimenteller Untersuchungen. Dabei erzeugte ein C-Programm die äußeren Beschreibungen von $Q_{tree}(K_3)$ und $Q_{tree}(K_4)$, die an *Polymake* [10] übergeben wurden, das die Eckenmenge der Polytope zurücklieferte. Wir wollen später diese Eckenmenge graphisch betrachten. Dafür nutzen wir Illustrationen, wie in Abbildung 5, in der eine z-Variable in $Q_{tree}(K_4)$ dargestellt ist.

Für den Graphen K_3 traten in den Untersuchungen lediglich 3 Vektoren als Ecken auf, die alle ganzzahlig sind und deren Projektionen auf ihre x-Koordinaten genau den aufspannenden Bäumen im K_3 entsprechen. Im K_4 jedoch, wurden neben 16 ganzzahligen Ecken 10 nicht-ganzzahlige Ecken generiert. Die 0/1-Ecken repräsentieren wiederum die aufspannenden Bäume im K_4. Die gebrochenzahligen Ecken sind dergestalt, dass alle x-Koordinaten den Wert 0.5 annehmen, ebenso wie alle z-Variablen, die ungleich null sind, wovon jeweils 12 existieren. In den Abbildungen 6 und 8 sind die nicht-ganzzahligen Ecken des K_4 illustriert. Dabei wird auf die Darstellung der x-Variablen aufgrund ihrer einheitlichen Belegung mit dem Wert 0.5 verzichtet. Man erkennt, dass für einen festen Knoten $w \in V$ die $z_{u,v,w}$-Variablen mit $u, v \in V \backslash \{w\}$, $u \neq v$, die ungleich null sind, einen gerichteten Kreis bilden. Theoretisch sind demnach 2^4 verschiedene nicht-ganzzahlige Ecken möglich. In der Tat erhält man mit dieser Konstruktion 16 zulässige Punkte. Jedoch handelt es sich nur bei den abgebildeten 10 Vektoren um Ecken. In Abbildung 7 ist exemplarisch die Konstruktion einer der 6 Nicht-Ecken als Konvexkombination zweier ganzzahliger Ecken in $Q_{tree}(K_4)$ dargestellt.

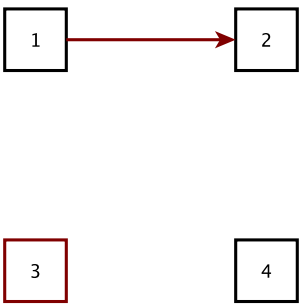

Abbildung 5: z-Variable am Beispiel $z_{1,2,3}$.

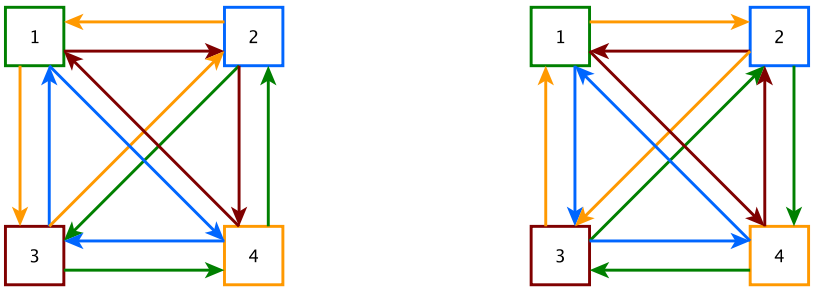

Abbildung 6: Nicht-ganzzahlige Ecken ohne parallele z-Variablen. Alle dargestellten Werte werden mit 0.5 angenommen. Die x-Variablen aller Kanten werden mit 0.5 angenommen.

Man kann zwei Typen nicht-ganzzahliger Ecken unterscheiden. Dazu betrachtet man für feste u, v die Anzahl der $z_{u,v,w}$-Variablen, die für verschiedene w ungleich null sind. In den Abbildungen äußert sich dies durch parallel verlaufende Pfeile. Bei den beiden Ecken in Abbildung 6 ist die Anzahl null. Die anderen Ecken weisen jeweils 3 parallele z-Variablenpaare auf. Für die in Abbildung 7 illustrierte Konvexkombination existieren 4 solcher Paare.

Nun sollte unabhängig von Polymake der Nachweis erbracht werden, dass die abgebildeten Punkte Ecken sind. Dafür wurden die Zeilen der äußeren Beschreibung, die die vermeintliche Ecke mit Gleichheit erfüllte, an *MATLAB* [19] übergeben und der Rang bestimmt. Ist dieser gleich der Spaltenanzahl, so handelt es sich um eine Ecke, was auf alle abgebildeten Ecken zutrifft.

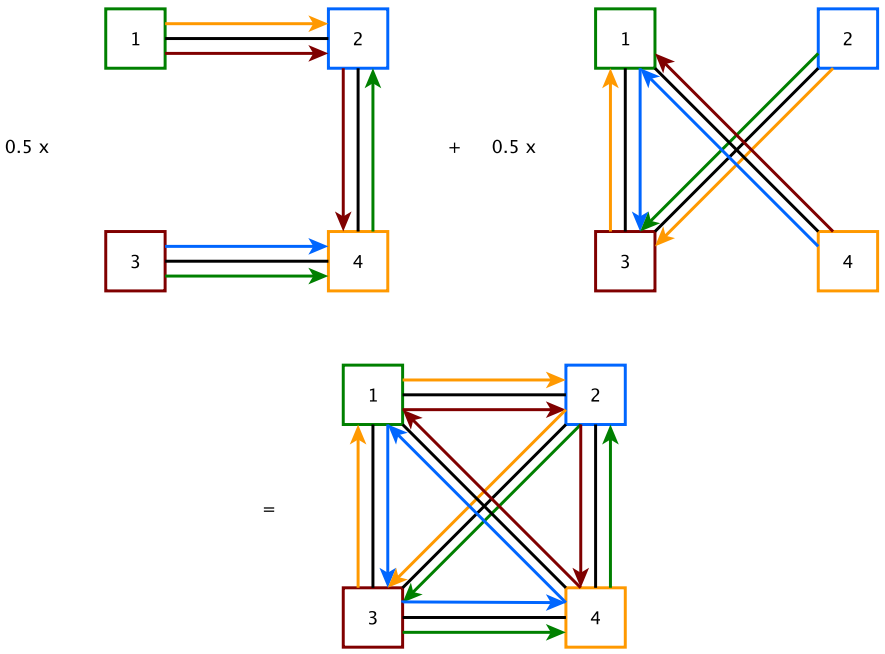

Abbildung 7: Konvexkombination eines nicht-ganzzahligen zulässigen Punktes, der keine Ecke ist, aus zwei ganzzahligen Ecken.

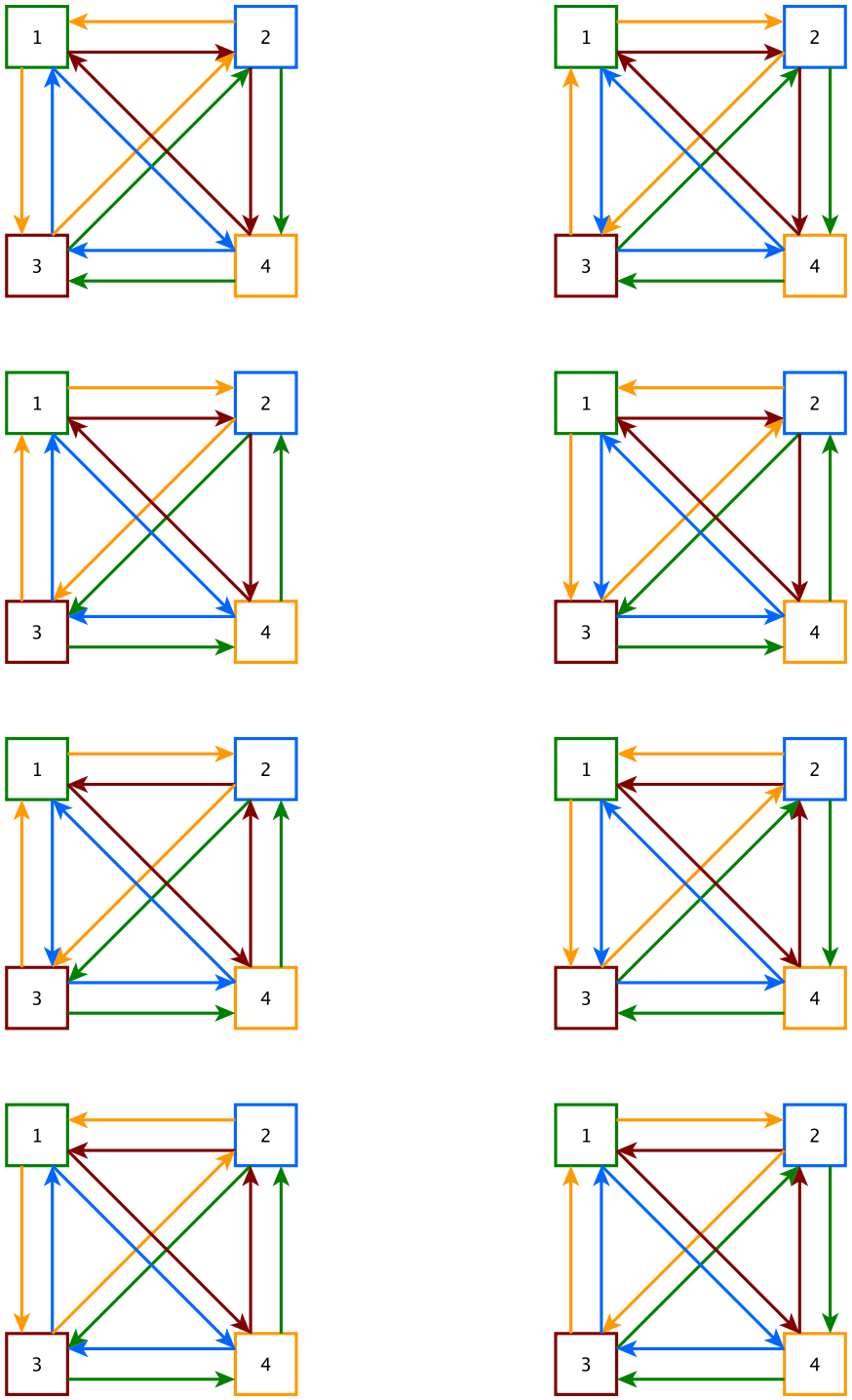

Abbildung 8: Nicht-ganzzahlige Ecken mit drei parallelen z-Variablen. Alle dargestellten Werte werden mit 0.5 angenommen. Die x-Variablen aller Kanten werden mit 0.5 angenommen.

2.7 Voronoi-Diagramme und Delaunay-Triangulierungen

Die erweiterten Formulierungen EFMST und EFSE beruhen auf dem Prinzip der Graphendualität bei planaren Graphen. In diesem Abschnitt wenden wir uns der Beantwortung der Frage zu, wie wir für einen gegebenen planaren Graphen seinen Dualgraphen finden und orientieren uns dabei an Klein [15]. Konstruktiv ließe sich das realisieren, indem man etwa durch Entlanglaufen der Kanten eines Graphen seine Gebiete absteckt und somit Dualknoten und daraus resultierend Dualkanten identifiziert. In der vorliegenden Arbeit wurde jedoch für eine gegebene Punktmenge in der Ebene die *Delaunay-Triangulierung* erstellt, die den planaren Graphen darstellt, auf dem die verschiedenen Optimierungsansätze ausgeführt wurden. Das *Voronoi-Diagramm* der Punktmenge ist der benötigte Dualgraph der Delaunay-Triangulierung. In dem folgenden Kapitel wollen wir die theoretischen Grundlagen der angesprochenen Strukturen und deren Dualitätseigenschaft beleuchten.

2.7.1 Voronoi-Diagramm

Zunächst kommen wir auf den Begriff des Voronoi-Diagrammes zu sprechen. Dabei bewegen wir uns im Raum $\mathbb{R}^2$ und betrachten eine n-elementige Menge von Punkten $V = \{v_1, ..., v_n\} \subset \mathbb{R}^2$. Wir notieren den euklidischen Abstand zwischen einem Punkt $v_i = (v_{i_1}, v_{i_2}) \in V$ und einem Punkt $x = (x_1, x_2) \in \mathbb{R}^2$ als

$$|v_i x| = \sqrt{|v_{i_1} - x_1|^2 + |v_{i_2} - x_2|^2}.$$

Anschließend definieren wir den *Bisektor* von zwei verschiedenen Punkten $v_i, v_j \in V$ via

$$B(v_i, v_j) := \{x \in \mathbb{R}^2 : |v_i x| = |v_j x|\}.$$

Der Bisektor $B(v_i, v_j)$ besteht also aus den Punkten $x \in \mathbb{R}^2$, die den gleichen euklidischen Abstand zu v_i und v_j mit $i \neq j$ haben. Geometrisch interpretiert handelt es sich um die Punkte auf der Mittelsenkrechten der Verbindungslinie $\overline{v_i v_j}$. Der $\mathbb{R}^2$ wird dabei von dem Bisektor $B(v_i, v_j)$ in zwei offene Halbebenen

$$D(v_i, v_j) := \{x \in \mathbb{R}^2 : |v_i x| < |v_j x|\}$$
$$D(v_j, v_i) := \{x \in \mathbb{R}^2 : |v_j x| < |v_i x|\}$$

zerlegt. Wir bezeichnen

$$VR(v_i, V) := \bigcap_{\substack{v_j \in V, \\ i \neq j}} D(v_i, v_j)$$

als die *Voronoi-Region* von v_i bezüglich V. Für zwei verschiedene Punkte $v_i, v_j \in V$ gilt offensichtlich, dass ihre Voronoi-Regionen disjunkt sind. Voronoi-Regionen sind als Durchschnitt endlich vieler offener Halbebenen selbst offen und konvex. Punkte aus $v_i \in V$, die auf dem Rand der konvexen Hülle von V liegen, erzeugen unbeschränkte Voronoi-Regionen. Die Differenz des $\mathbb{R}^2$ und der Vereinigung der Voronoi-Regionen heißt *Voronoi-Diagramm* von V. Wir notieren es als

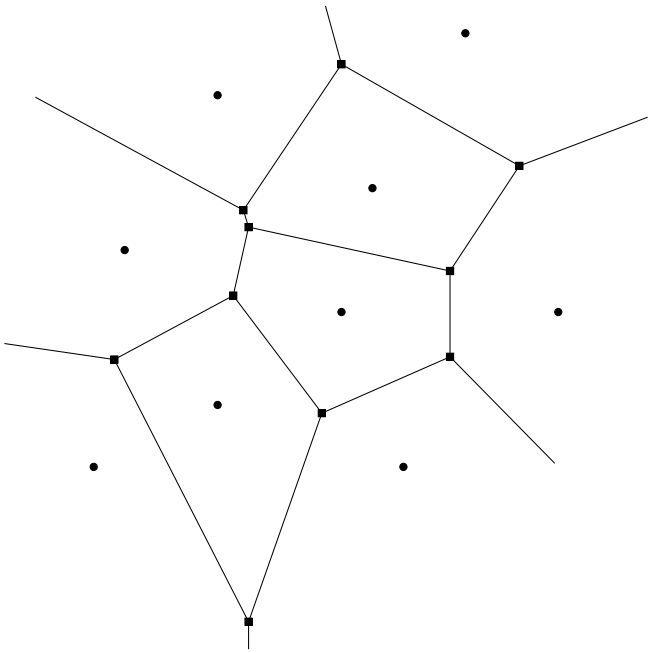

Abbildung 9: Voronoi-Diagramm mit eckigen Voronoi-Knoten und Voronoi-Kanten für neun runde Punkte in der Ebene.

$$VD(V) := \mathbb{R}^2 \setminus \bigcup_{i \in [n]} VR(v_i, V).$$

Für zwei verschiedene Punkte $v_i, v_j \in V$ heißt der Teil des Bisektors $B(v_i, v_j)$, der die beiden Voronoi-Regionen $VR(v_i, V)$ und $VR(v_j, V)$ begrenzt, *Voronoi-Kante*. Die Endpunkte der Voronoi-Kanten nennen wir *Voronoi-Knoten*. Nehmen wir an, die Voronoi-Kanten mit nur einem Endpunkt würden in einem gemeinsamen Knoten enden, dann kann das Voronoi-Diagramm einer endlichen Punktmenge als planarer Graph aufgefasst werden. In Abbildung 9 ist das Voronoi-Diagramm für eine neunelementige Punktmenge illustriert.

2.7.2 Delaunay-Triangulierung

Im Folgenden sei wieder $V \subset \mathbb{R}^2$ eine Menge von n Punkten in der Ebene und $VD(V)$ ihr Voronoi-Diagramm.

Es werden nun zwei verschiedene Punkte $v_i, v_j \in V$ mittels einer Kante verbunden, wenn ihre Voronoi-Regionen $VR(v_i, V)$ und $VR(v_j, V)$ an eine gemeinsame Voronoi-Kante in $VD(V)$ grenzen. Wir nennen die entstehende Kante *Delaunay-Kante*. Wählen wir für einen Graphen die Knotenmenge V und Kantenmenge als Menge aller Delaunay-Kanten, erhalten wir die *Delaunay-Zerlegung* $DT(V)$ von V. Jede Voronoi-Region enthält genau einen Punkt $v_i \in V$ und für jede Voronoi-Kante gibt es genau eine (kreuzende) Kante in $DT(V)$. Aufgrund der Konvexität der Voronoi-Regionen besitzen jeweils zwei benachbarte Regionen genau eine gemeinsame Kante. $DT(V)$ bildet daher den Dualgraphen von $VD(V)$. Für die Umkehrung der Aussage beachte man, dass die Voronoi-Kanten mit nur einem Voronoi-Knoten in einem gemeinsamen Knoten enden, den wir als Knoten des äußeren Gebietes von $DT(V)$ auffassen. Als dualer Graph eines planaren Graphen

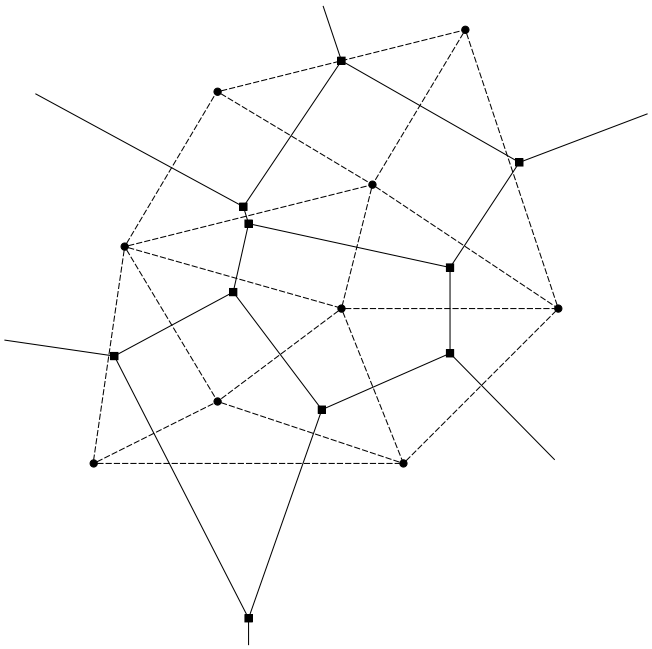

Abbildung 10: Voronoi-Diagramm mit eckigen Voronoi-Knoten und Voronoi-Kanten und Delaunay-Zerlegung mit gestrichelten Delaunay-Kanten für neun runde Punkte in der Ebene.

ist $DT(V)$ wiederum planar. Es folgt insbesondere, dass sich die Delaunay-Zerlegung einer n-elementigen Knotenmenge – bei geeigneter zugrunde liegender Datenstruktur – in linearer Zeit $O(n)$ aus dem Voronoi-Diagramm konstruieren lässt. In Abbildung 10 ist eine Delaunay-Zerlegung dargestellt.

Ist für alle Voronoi-Knoten die Menge der am nächsten gelegenen Punkte aus V dreielementig, so handelt es sich bei den Gebieten der Delaunay-Zerlegung um Dreiecke. Daher wird auch der Begriff *Delaunay-Triangulierung* genutzt.

Man beachte, dass das Konstrukt der Delaunay-Triangulierung einer Punktmenge V auch ohne Zuhilfenahme des Voronoi-Diagramms $VD(V)$ realisiert werden kann (s. Okabe et al. [22]). In den Experimenten, die im Rahmen dieser Arbeit stattfanden, wurde demzufolge für eine gegebene Punktmenge zuerst die Delaunay-Triangulierung erstellt. Daher handelte es sich bei den Testdaten immer um Triangulierungen. Erst im Anschluss folgte die Konstruktion des Voronoi-Diagramms.

Algorithmisch kann eine Delaunay-Triangulierung für eine gegebene n-elementige Punktmenge auf verschiedene Arten generiert werden (s. Fortune [9]). Komplexitätstheoretisch aufwändig ist der sogenannte *Flip-Algorithmus* mit einer Laufzeit von $\Omega(n^2)$. Dabei generiert man zunächst eine Triangulierung für die Punktmenge und tauscht (*flipt*) anschließend so lange Kanten, bis eine Delaunay-Triangulierung vorliegt. Laufzeittechnisch äquivalent mit einem Aufwand von $O(n\ log(n))$ sind ein *Divide-and-Conquer-Ansatz* und der *Sweepline-Algorithmus*.

3 Experimente

3.1 Implementierung und Testumgebung

Um die im Laufe der Arbeit vorgestellten Lösungsansätze für das (DC)MST-Problem zu testen und hinsichtlich verschiedener Parameter zu vergleichen, musste zunächst eine Testumgebung konstruiert und implementiert werden. Das Kernstück bildete dabei das SCIP-Framework. SCIP ist ein nicht-kommerzielles MIP- und Constraint-Programming-Framework, entwickelt am Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB). Es ist als Open-Source-Version verfügbar, gut dokumentiert und auch im Vergleich mit kommerziellen MIP-Solvern konkurrenzfähig (s. Mittelmann [21]). SCIP wurde bei den Tests in dieser Arbeit als Branch-and-Cut-Framework verwendet. Als LP-Solver diente der ebenfalls am Konrad-Zuse-Institut entwickelte und im SCIP-Framework standardmäßig verwendete *SoPlex* (Sequential Object-Orientated Simplex, s. [29]).

Um das systematische Testen zu erleichtern, wurde eine Testumgebung in C++ geschrieben, die die Grapheninstanzen erzeugte, einzelne Lösungsansätze ausführte und anschließend eine Auswertung bezüglich verschiedener Parameter vollzog. Nachfolgend betrachten wir die einzelnen Komponenten der Testumgebung sowie die Realisierung der verschiedenen Lösungsansätze: (i) Separation über $P^{(k)}_{tree}(G)$, (ii) Optimierung mittels EFMST, (iii) Optimierung mittels EFSE, (iv) Optimierung mittels CEFMST. Die Optimierungsprobleme wurden jeweils als IP gelöst.

3.1.1 Testdaten

Als Testdaten, auf denen die verschiedenen Lösungsverfahren ausgeführt wurden, dienten zum einen selbst generierte Zufallsgraphen, zum anderen Grapheninstanzen aus der *TSPLIB* [25]. Dies ist eine Bibliothek von Graphen, die zur Lösung des Traveling-Salesman-Problems (TSP) oder verwandter Problemstellungen herangezogen werden kann und eine externe Vergleichsmöglichkeit der hier erzielten Werte liefert. Die beiden genutzten Instanzarten unterscheiden sich in der Art und Weise, wie die Knoten der Graphen erstellt wurden.

Bei den selbst erstellten Graphinstanzen lieferte die in der *Boost Random Number Library* [20] zur Verfügung stehende *uniform_real-Methode* randomisiert Punkte im zweidimensionalen Einheitskreis. Dabei ist die Reproduzierbarkeit der Daten gewährleistet. Für die TSPLIB-Graphen dienten die TSPLIB-Dateien, die als Punkte im $\mathbb{R}^2$ gegeben sind, als Vorgabe für hier verwendete Probleminstanzen. Anschließend wurde mit Hilfe der *Computational Geometry Algorithms Library* (CGAL [4]) eine Delaunay-Triangulierung der vorliegenden Punkte erstellt und ein Voronoi-Diagramm erstellt. Die Dualität der beiden Konstrukte und ihre Rolle für die erweiterte Formulierung ist Thema in Kapitel 2.7. Die so erhaltenen Graphen, auf denen dann die Optimierungsroutinen abliefen, waren planar. Kantengewichte waren die euklidischen Abstände der inzidenten Knoten. Während sie die Graphinstanzen erstellte, bestimmte die Routine die benötigten primalen und dualen Wurzelknoten. Dabei war der mit dem äußeren Gebiet assoziierte Knoten immer die duale Wurzel und ein Knoten im Primalgraphen, der an das äußere Gebiet grenzt, die Primalwurzel. Die resultierenden Graphen

waren in der Form abgespeichert, dass nach Benennung der Wurzelknoten zeilenweise die benachbarten primalen Knoten zuzüglich ihres Abstandes und die mit der Dualkante inzidenten Dualknoten vermerkt waren.

Für die TSPLIB-Graphen ist noch zu beachten: Aufgrund der Funktionalität CGALs war es notwendig die gegebenen Knoten sehr minimal vor der Erstellung der Delaunay-Triangulierung und des Voronoi-Diagrammes zu verschieben. Trat der Fall ein, dass die vier begrenzenden Knoten zweier benachbarter Gebiete der Triangulierung auf einem Kreis lagen, so fielen die von CGAL erstellten Zentren der beiden Gebiete auf denselben Punkt und es wurde nur ein einzelner Voronoi-Knoten erzeugt. Dieses Verhalten führte zur Generierung eines Graphen, der für die linearen erweiterten Formulierungen nicht verwendbar war.

3.1.2 Realisierung der Lösungsmethoden

Separationsmethode

Eine zentrale Rolle bei der Realisierung des Separationsansatzes nahm die *Library for Efficient Modeling and Optimization in Networks* (LEMON [6]) ein. Sie ermöglichte eine komfortable und performante Arbeitsweise mit Graphen. So wurde für jede zu betrachtende Probleminstanz ein Objekt erzeugt, das einen LEMON-Graphen enthielt, wobei die Kanten des Graphen und die Problemvariablen mittels Maps verknüpft waren. Der LEMON-Graph war dabei die gerichtete Version des Problemgraphen, um Flussalgorithmen anwenden zu können. Um eine Verbindung zum SCIP-Framework herzustellen, war als weiteres Attribut ein Zeiger auf ein SCIP-Objekt eingepflegt. Durch einen Konstruktoraufruf des angesprochenen Instanz-Objekts wurde eine Grapheninstanz eingelesen und die entsprechenden Variablen, sowie die Zusammenhangsbedingung ($x(E) = |V| - 1$) und – falls vorhanden – die Gradschranke erzeugt.

Als weitere Nebenbedingung kam ein Constraint-Handler zum Einsatz. Dies ist eine Routine, die nach dem Lösen des Problem-LPs aufgerufen wird, um zu entscheiden, ob die gefundene Lösung zulässig für $P^{(k)}_{tree}(G)$ ist und falls nicht, dem LP eine oder mehrere Nebenbedingungen hinzufügt. Mittels des Resultats aus Abschnitt 2.4 ist das durch Lösen eines Max-Flow-Problems möglich, das verletzte Cycle-Elimination-Constraints liefert. An dieser Stelle kam wieder LEMON zum Einsatz, das eine performante Push-Relabel-Implementierung des Max-Flow-Problems bietet. Dazu wurden die künstlichen Knoten s und t zum LEMON-Graphen hinzugefügt, die Bogenkapazitäten berechnet und den Bögen des Digraphen zugewiesen. Die Push-Relabel-Methode berechnete einen Min-Cut. Auf diese Art fand sie eine maximal verletzte Ungleichung, die der Handler dem LP hinzufügte, wenn eine vermeintliche Lösung sie verletzte. Anderenfalls stellte die Routine an dieser Stelle die Zulässigkeit der Lösung fest.

Um die Performanz der Routine zu steigern, wurden bei den Experimentalläufen mehrere verletzte Ungleichungen pro LP-Iteration hinzugefügt. Da der LEMON-Algorithmus nur einen Min-Cut für einen vorliegenden Graphen berechnen kann, musste der Graph, der in dem jeweiligen Durchlauf vorlag, modifiziert werden. Dafür wurde der künstliche Knoten t entfernt und ein Knoten v, der sich auf der gleichen Seite des Min-Cuts

befand wie t, als neuer Senkenknoten ausgezeichnet. Die Kapazitäten der Bögen, die von einem Knoten u in t hineinflossen, wurden auf die Kapazitäten des Bogens (u, v) addiert bzw. wurde solch ein Bogen,falls es ihn vorher nicht gab, erstellt. Darauf folgte auf dem so veränderten Graphen die Berechnung eines neuen Min-Cuts mit Hilfe von LEMON.

Auf diese Art konnte der Constraint-Handler pro LP-Iteration 10 bis 200 Ungleichungen – je nach Knotenanzahl des Graphen – hinzufügen. Die Anzahl der auf einmal hinzuzufügenden Constraints war Inhalt einer Reihe von Vorexperimenten, die für jede Problemgröße der Zufallsgraphen und ohne Gradbeschränkung erfolgten. Die schnellste Strategie fand in den Experimenten Anwendung. Dabei kam für alle Gradschranken bei gleicher Graphengröße die gleiche Methode zum Einsatz. Zusätzlich zu der Steuerung, wie viele Bedingungen pro Iteration zum LP hinzukamen, war auch eine Abbruchschranke eingebaut. Es wurde nach Schrankengröße vielen erfolglosen Suchen nach einer verletzten Ungleichung die Suche abgebrochen und lediglich die bisher gefundenen Constraints dem linearen Programm hinzugefügt. Die Bestimmung des Abbruchkriteriums erfolgte analog durch Voruntersuchungen. Für Graphen mit einer Größe von 50 Knoten hieß das, dass 10 verletzte Ungleichungen pro Iteration hinzugefügt wurden, jedoch nach 10 erfolglosen Min-Cut-Auswertungen die bis dahin ermittelten Constraints in das LP einflossen. Bei Graphen mit einer Knotenkardinalität von 400 lagen diese Werte bei 200 und 120.

Durch das oben beschriebene Hinzufügen mehrerer verletzter Ungleichungen kann nicht garantiert werden, dass verschiedene Ungleichungen gefunden werden. Da es durchaus geschehen kann, dass LEMON den gleichen Min-Cut für verschiedene Graphen, die durch Modifizierung aus dem Ursprungsgraphen hervorgegangen sind, zurückliefert. Nach umfangreichen Vorexperimenten hatte sich diese Strategie jedoch als äußerst geschwindigkeitssteigernd erwiesen.

Um die Laufzeiten weiter zu verkürzen, wurde durch verschiedene Einstellungen das Branching von SCIP so weit wie möglich eingeschränkt. Voruntersuchungen hatten einen positiven Laufzeiteinfluss festgestellt, wenn der Lösungsprozess so lange wie möglich im Wurzelknoten blieb.

Im Zusammenhang mit der Implementierung der Separationsroutine stand auch das Vorgehen zur Lösung des TSP-Problems von Applegate, Bixby, Chvátal und Cook [2] im Fokus. Diese beginnen mit dem gleichen Initial-LP wie in dieser Arbeit und fügen auch Subtour-Elimination-Cuts während des Lösungsprozesses hinzu. Zusätzlich werden bei ihnen jedoch eine Reihe weiterer Cuts wie *Kamm- und Blüten-Ungleichungen*, die sie durch ausgeklügelte Heuristiken finden, generiert. Außerdem wird das Initial-LP mittels eines performanten Algorithmus als *2-Matching-Problem* gelöst. Der dort ebenfalls angewendete Mechanismus zur Spaltengenerierung anhand der reduzierten Kosten ist beim reinen TSP, das auf vollständigen Graphen arbeitet, wertvoller als auf den Graphen dieser Arbeit. Die Autoren arbeiten außerdem mit Shrinking-Techniken, wobei sie den Algorithmus von Padberg und Rinaldi [23] nutzen, um mehrere verletzte Subtour-Eliminations-Constraints in einem Durchlauf zu finden.

Tabelle 1: Laufzeiten für Zufallsgraphen einer Größe von 50 bis 25 000 Knoten unter den verschiedenen Lösungsverfahren und ohne anliegende Gradschranke bzw. Gradschranke 8

	ohne Gradschranke			Gradschranke 8		
# Knoten	Sep.[1]	EFMST	CEFMST	Sep.[1]	EFMST	CEFMST
50[2]	0.12	0.02	4.85[3]	0.11	0.02	4.83[3]
100[2]	1.03	0.05	103.54[4]	0.96	0.06	100.72[4]
200[2]	10.44[3]	0.10		10.66[3]	0.13	
300[2]	61.76[3]	0.16		60.11[3]	0.22	
400[2]	282.64[4]	0.24		276.69[4]	0.33	
500[2]		0.33			0.46	
750[3]		0.63			0.90	
1 000[3]		1.03			1.46	
2 000[3]		4.37			5.77	
5 000[3]		22.90			30.21	
7 500[3]		51.58			70.36	
10 000[3]		99.64			128.01	
25 000[5]		662.79			845.89	

Anmerkungen. dargestellte Zeiten sind geometrische Mittel in Sekunden

[1] Separation, [2] 500 getestete Graphen, [3] 100 getestete Graphen, [4] 50 getestete Graphen, [5] 25 getestete Graphen

Optimierung über den erweiterten Formulierungen

Die Optimierung über den erweiterten Formulierungen erfolgte so, dass die Routine pro Probleminstanz ein Objekt erzeugte. Dieses enthielt als Attribut einen Zeiger auf ein SCIP-Objekt und einen LEMON-Graph, bzw. zwei LEMON-Digraphen für die Primal- und Dualgraphen der linearen erweiterten Formulierungen. Ein Konstruktoraufruf des Objekts hatte wiederum zur Folge, dass die Graphdateien eingelesen, und das Optimierungsproblem inklusive aller nötigen Variablen und Nebenbedingungen erstellt wurde. Das vollständige Problem wurde dann an SCIP zur Lösung übergeben. Die Lösungsroutine führte – bis auf später näher erläuterte Ausnahmen – das vom SCIP-Framework bereitgestellte Presolving aus.

3.2 Tests und Testergebnisse

3.2.1 Lösungszeiten

Als Testplattform zur Untersuchung der Lösungsansätze diente ein System mit AMD Phenom II X6 2.8GHz Prozessor und Linux Betriebssystem. Eine Testroutine steuerte die Optimierung über den erzeugten Zufallsgraphen und Triangulierungen der TSPLIB-Knotendateien für die Gradschranken $k = 2, 3, 4, 8$ und ohne Gradbeschränkung mit Hilfe der oben beschriebenen Lösungsansätze. Die TSPLIB-Graphen waren für die Experimente in mehreren Gruppen zusammengefasst. Bei der Separation über den TSPLIB-Instanzen wurde

Tabelle 2: Laufzeiten für Zufallsgraphen einer Größe von 50 bis 25 000 Knoten unter den verschiedenen Lösungsverfahren und Gradschranke 2, 3 und 4

	Gradschranke 2				Gradschranke 3			Gradschranke 4		
# Knoten	Sep.[1]	EFMST	EFSE	CEFMST	Sep.[1]	EFMST	CEFMST	Sep.[1]	EFMST	CEFMST
50[2]	0.07	0.09	0.06	13.87[3]	0.12	0.02	5.24[3]	0.12	0.03	4.90[3]
100[2]	1.73	1.07	0.56	587.36[4]	0.99	0.06	103.61[4]	1.02	0.06	102.35[4]
200[2]	229.53[4]	15.88[3]	8.86[3]		10.66[3]	0.12		11.34[3]	0.13	
300[2]		148.29[4]	35.90[4]		61.54[3]	0.21		65.46[3]	0.21	
400[2]		661.67[5]	117.10[4]		287.10[4]	0.30		302.74[4]	0.30	
500[2]						0.43			0.42	
750[3]						0.81			0.80	
1 000[3]						1.34			1.34	
2 000[3]						5.26			5.28	
5 000[3]						27.97			27.13	
7 500[3]						68.29			63.08	
10 000[3]						129.41			112.62	
25 000[5]						910.37			750.95	

Anmerkungen. gemessene Zeiten sind geometrische Mittel in Sekunden

[1] Separation, [2] 500 getestete Graphen, [3] 100 getestete Graphen, [4] 50 getestete Graphen, [5] 25 getestete Graphen

diejenige Strategie zum Hinzufügen mehrerer Ungleichungen pro LP-Iteration gewählt, die auch Verwendung bei den Zufallsgraphen fand, die genauso groß waren wie der größte TSPLIB-Graph in der Testgruppe.

Die Tabellen 1 bis 4 zeigen die ermittelten Zeiten im geometrischen Mittel, die zur IP-Lösung aufgewendet werden mussten. Es ist dort auch die Anzahl der getesteten Zufallsgraphen pro Knotenanzahl und die Kardinalität der einzelnen TSPLIB-Gruppen notiert. Da bei der Separationsmethode einige wenige Graphen eine deutlich höhere Lösungszeit benötigten, sind bei den Zufallsinstanzen nur die Laufzeiten der schnellsten 50 % in die Wertermittlung eingeflossen. Der Laufzeitvergleich mit den erweiterten Formulierungen sollte nicht stark durch Ausreißer nach oben beeinflusst werden.

Bei Betrachtung der Zufallsgraphen-Laufzeiten in den Tabellen 1 und 2 fällt auf, dass die erweiterte Formulierung CEFMST, deutlich langsamer als die anderen Methoden war. Als möglicher Grund ist sicherlich die Größe der erweiterten Formulierung mit $O(|V||E|)$ vielen Ungleichheits-Nebenbedingungen zu erwähnen. Der kubische Charakter der CEFMST-Methode wird in der Laufzeitentwicklung bei Erhöhung der Knotenanzahl deutlich.

Die Separationsmethode benötigte im Vergleich deutlich geringere Zeiten zur Lösung der gestellten Optimierungsaufgaben, wobei ein relativ großer Laufzeitsprung bei den getesteten Zufallsgraphen von einer Größe von 300 Knoten auf 400 Knoten zu verzeichnen ist.

Tabelle 3: Laufzeiten für TSPLIB-Graphen einer Größe von unter 100 bis 18 512 Knoten unter den verschiedenen Lösungsverfahren und ohne anliegende Gradschranke bzw. Gradschranke 8

	ohne Gradschranke			Gradschranke 8		
# Knoten	Sep.[1]	EFMST	CEFMST	Sep.[1]	EFMST	CEFMST
≤ 100[2]	0.28	0.03	13.45	0.29	0.04	13.88
101 - 400[3]	11.03	0.09		9.71	0.12	
417 - 783[4]		0.38			0.55	
1 000 - 7 397[5]		3.34			4.40	
11 849 - 18 512[6]		187.75			228.18	

Anmerkungen. dargestellte Zeiten sind geometrische Mittel in Sekunden

[1] Separation, [2] 17 getestete Graphen, [3] 29 getestete Graphen, [4] 14 getestete Graphen, [5] 26 getestete Graphen, [6] 5 getestete Graphen

Tabelle 4: Laufzeiten für TSPLIB-Graphen einer Größe von unter 100 bis 18 512 Knoten unter den verschiedenen Lösungsverfahren und Gradschranke von 2, 3 und 4

	Gradschranke 2				Gradschranke 3			Gradschranke 4		
# Knoten	Sep.[1]	EFMST	EFSE	CEFMST	Sep.[1]	EFMST	CEFMST	Sep.[1]	EFMST	CEFMST
≤ 100[2]	0.62	0.26	0.16	46.43	0.24	0.04	14.31	0.27	0.04	14.56
101 - 400[3]	27.99[7]	9.32	2.85		10.83	0.11		11.11	0.11	
417 - 783[4]		1 941.14	180.62			0.52			0.49	
1 000 - 7 397[5]						6.06			3.92	
11 849 - 18 512[6]						396.13			205.41	

Anmerkungen. dargestellte Zeiten sind geometrische Mittel in Sekunden

[1] Separation, [2] 17 getestete Graphen, [3] 29 getestete Graphen, [4] 14 getestete Graphen, [5] 26 getestete Graphen, [6] 5 getestete Graphen, [7] 7 Graphen wurden nach 5 Stunden Rechenzeit abgebrochen und flossen nicht in den Durchschnittswert ein

Tabelle 5: Laufzeiten für Zufallsgraphen einer Größe von 1 000 bis 25 000 Knoten unter der EFMST-Methode und ohne anliegende Gradschranke bzw. Gradschranke 4 und 8 bei ausgeschaltetem Presolving

# Knoten	ohne Gradschranke	Gradschranke 8	Gradschranke 4
1 000[1]	0.85	0.97	1.00
2 000[1]	3.82	4.34	4.45
5 000[1]	20.15	22.79	23.95
7 500[1]	46.99	54.09	55.88
10 000[1]	92.13	97.82	102.60
25 000[2]	644.52	668.53	677.95

Anmerkungen. dargestellte Zeiten sind geometrische Mittel in Sekunden

[1] 100 getestete Graphen, [2] 25 getestete Graphen

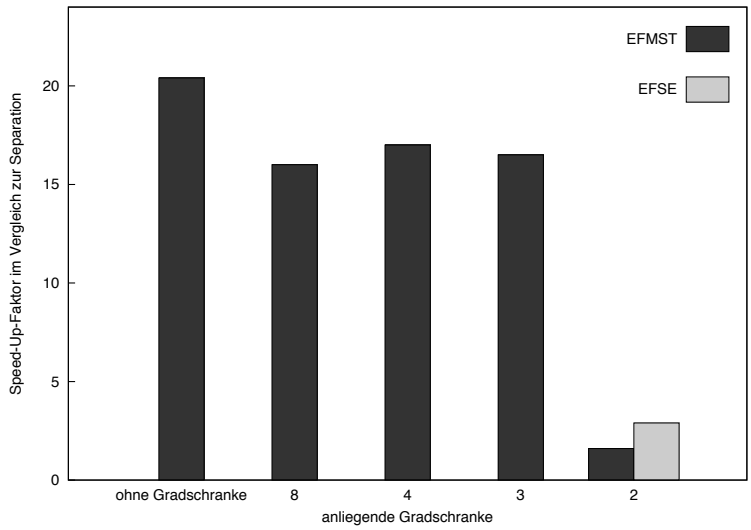

Abbildung 11: Speed-Up-Faktoren der erweiterten Formulierungen linearer Größe bei Zufallsgraphen einer Größe von 100 Knoten und ohne anliegende Gradschranke bzw. mit Gradschranke 2, 3, 4 und 8 im Vergleich zur Separationsmethode.

Auffällig ist der sehr große Vorteil der linearen erweiterten Formulierungen in puncto Laufzeit im Vergleich zu den anderen Methoden. Dieser Vorteil äußert sich im sogenannten *Speed-Up-Faktor*, der für ausgewählte Graphengrößen in den Abbildungen 11 und 12 dargestellt ist. Sicherlich spielte die Größe der Formulierungen dabei eine wichtige Rolle. Wobei zusätzlich zu den Lösungszeiten eine, für kleine Graphen sehr kurze, und für Graphen mit 25 000 Knoten etwa 100-sekündige Zeit hinzukam, die es zur Erstellung des Optimierungsproblems bedurfte.

Auch der Laufzeitunterschied für die Gradschranke 2 innerhalb der linearen Formulierungen ist bemerkenswert. Hier ist wohl erneut der Größenunterschied zwischen den Formulierungen zu nennen. Das EFSE-System verfügte über $2|E|$ weniger Variablen und Ungleichheits-Nebenbedingungen (inklusive Nichtnegativitätsbedingungen) als die EFMST-Methode.

Vergleicht man innerhalb einer Lösungsmethode die Laufzeitunterschiede bezüglich der verschiedenen Gradschranken, so kommt man zu einem konsistenten Ergebnis: Die Laufzeiten waren für $k = 2$ am höchsten und nahmen relativ stark beim Übergang zu $k > 2$ ab. Wobei kaum noch oder gar keine Unterschiede zwischen $k = 4, 8$ und ohne Gradschranke bestanden.

Die auftretenden Knotengrade der Optimallösungen für die Probleme ohne Gradschranke waren ebenfalls Inhalt der Untersuchungen. Dabei kamen fast ausschließlich Knotengrade mit Wert kleiner gleich 3 und selten bis maximal 5 vor – auch bei Graphengrößen von $n = 10\,000$. Das könnte ein Grund sein, weshalb es keine nennenswerten Laufzeitschübe bei Erhöhung der Gradschranke ab $k = 4$ gab, bzw. weshalb die Laufzeit für $k = 4$ im geometrischen Mittel geringer war als die von $k = 8$ bei der EFMST-Methode: Durch die Schranke

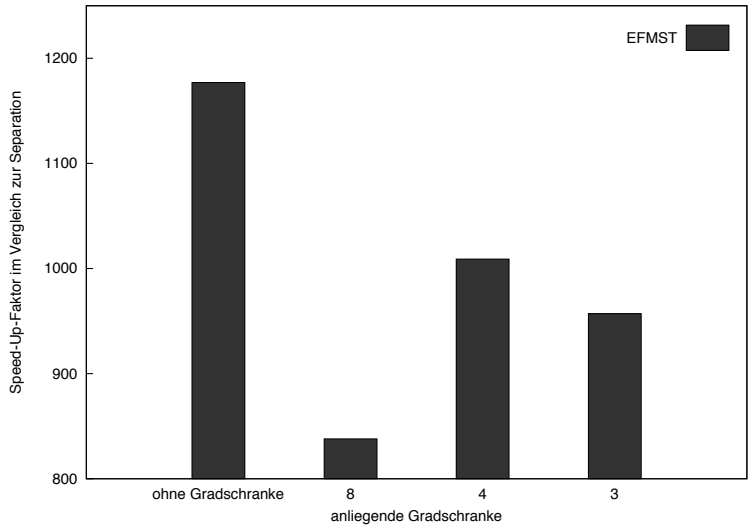

Abbildung 12: Speed-Up-Faktoren der EFMST-Methode bei Zufallsgraphen einer Größe von 400 Knoten und ohne anliegende Gradschranke bzw. mit Gradschranke 3, 4 und 8 im Vergleich zur Separationsmethode.

$k = 4$ wurde der Lösungsraum im Vergleich zu $k = 8$ so eingeschränkt, dass zulässige Lösungen, die als Optimallösung nicht in Frage kamen, schon von vornherein abgeschnitten wurden. Zusätzlich konnte auch der Presolver – mit Blick auf die Lösungsgrade ohne Schranke – effizienter arbeiten, wenn Gradschranke 4 vorgegeben war. In Tabelle 5 sind daher die Laufzeiten für Gradschranken 4 und 8 mit abgeschaltetem Presolving dargestellt.

Das Abschalten des Presolvings hatte zum einen den Effekt, für $k \neq 2$ die Lösungszeiten für die EFMST-Methode leicht zu verbessern und zum anderen die gerade beschriebene Laufzeiterhöhung bei Gradschrankenerhöhung nicht aufkommen zu lassen. Jedoch stiegen die Laufzeiten für die Gradschranke 2 sehr stark an, wenn das Presolving ausgeschaltet war. Aus diesem Grund und der Konsistenz wegen wurden alle Tests mit Presolving vorgenommen.

Die in den Tabellen 3 und 4 aufgelisteten Laufzeiten zeigen die eben diskutierten Verhaltensmuster in puncto Gradschrankenänderung, Knotenzahlerhöhung und Unterschiede zwischen den einzelnen Lösungsansätzen auch auf den TSPLIB-Graphen. Bezüglich der verschieden gearteten Testdaten ist also kein Unterschied auszumachen.

Zusätzlich zu den aufgelisteten Lösungsansätzen für das (DC)MST-Problem wurde eine von LEMON bereitgestellte Realisierung des Algorithmus von *Kruskal* [16] auf Graphen mit einer Größe von 25 000 Knoten und ohne Gradschranke bezüglich der benötigten Lösungszeiten untersucht. Diese erzielte eine durchschnittliche Laufzeit von 0.06 Sekunden. Es handelt sich also um einen Laufzeitvorteil von einem Faktor von etwa 10 000 im Vergleich zu der EFMST-Methode ohne Presolving auf den gleichen Graphinstanzen.

Tabelle 6: Auswertung für Zufallsgraphen einer Größe von 100 Knoten unter den verschiedenen Lösungsverfahren und ohne anliegende Gradschranke bzw. Gradschranke 8

	ohne Gradschranke			Gradschranke 8		
Parameter	Sep.[1]	EFMST	CEFMST	Sep.[1]	EFMST	CEFMST
Laufzeit	1.03	0.05	103.54	0.96	0.06	100.72
# Constraints Beginn	1.00	564	32 582	101	664	32 682
# Variablen	282	1 128	27 914	282	1 128	27 914
LP Lsg.-Zeit	0.13	0.01	100.26	0.13	0.01	97.42
# LP-Iterationen	2 948	730	41 105	2 942	730	40 731
# Wurzel-Kn. Iter.[2]	2 948	730	41 105	2 940	730	40 731
# B&B-Knoten	1.00	1.00	1.00	1.10	1.00	1.00
Wurzel-DB-Fehler[3]	0	0	0	0.95	0	0
# LP pro Knoten[4]	0	0	0	1.07	0	0
# Cuts Handler[5]	325	-	-	299	-	-
Zeit Handler[6]	0.63	-	-	0.56	-	-

Anmerkungen. dargestellte Parameter sind geometrische Mittel, dargestellte Zeiten sind geometrische Mittel in Sekunden
[1] Separation, [2] # Wurzelknoten Iterationen, [3] Fehler der Dual-Bound im Wurzelknoten in %, [4] ohne Wurzelknoten,
[5] # hinzugefügter Cuts des eigenen Constraint-Handlers, [6] vom Constraint-Handler beanspruchte Zeit

Tabelle 7: Auswertung für Zufallsgraphen einer Größe von 100 Knoten unter den verschiedenen Lösungsverfahren und Gradschranke 2 und 3

	Gradschranke 2				Gradschranke 3		
Parameter	Sep.[1]	EFMST	EFSE	CEFMST	Sep.[1]	EFMST	CEFMST
Laufzeit	1.73	1.07	0.56	587.36	0.99	0.06	103.61
# Constraints Beginn	101	664	556	32 682	101	664	32 682
# Variablen	282	1 128	846	27 914	282	1 128	27 914
LP Lsg.-Zeit	0.02	0.22	0.08	525.89	0.15	0.01	100.32
# LP-Iterationen	853	3 088	1 851	232 700	2 902	756	41 284
# Wurzel-Kn. Iter.[2]	617	2 636	1 684	157 506	2 899	756	41 284
# B&B-Knoten	16.54	2.79	2.47	4.85	1.12	1.00	1.00
Wurzel-DB-Fehler[3]	0.47	0.48	0.52	0.52	0.93	0	0
# LP pro Knoten[4]	3.00	52.86	23.35	1 724	1.09	0	0
# Cuts Handler[5]	643	-	-	-	289	-	-
Zeit Handler[6]	1.52	-	-	-	0.53	-	-

Anmerkungen. dargestellte Parameter sind geometrische Mittel, dargestellte Zeiten sind geometrische Mittel in Sekunden
[1] Separation, [2] # Wurzelknoten Iterationen, [3] Fehler der Dual-Bound im Wurzelknoten in %, [4] ohne Wurzelknoten,
[5] # hinzugefügter Cuts des eigenen Constraint-Handlers, [6] vom Constraint-Handler beanspruchte Zeit

3.2.2 Detaillierte Auswertung

Wir betrachten nun für einige Gradschranken und Testinstanzen einzelne Parameter der SCIP-Lösungsroutinen im geometrischen Mittel. In den Tabellen 6 bis 13 sind ausgewählte Parameter aus den Ausgabe-Files der SCIP-Routine dargestellt. Beschränken wir uns zunächst auf Zufallsgraphen-Instanzen mit einer Größe von 100 Knoten, die in den Tabellen 6 und 7 betrachtet werden.

Zuerst fallen die unterschiedlichen Größen der (initialen) LPs für die verschiedenen Lösungsansätze auf. Dabei wurden bei der Anzahl der Constraints alle Nebenbedingungen, inklusive der Gleichheits- aber exklusive der Nichtnegativitätsbedingungen berücksichtigt, weshalb beim Separationsverfahren (MST) nur so wenige initiale Ungleichungen aufgelistet sind: die Zusammenhangsbedingung und – falls vorhanden – die Gradbeschränkungen für jeden Knoten im Graphen. Es tritt deutlich der Größenunterschied zwischen den beiden erweiterten Formulierungen linearer Größe (EFMST und EFSE) und der Formulierung kubischer Größe (CEFMST) hervor. Auch der Größenvorteil der EFSE-Methode gegenüber der EFMST-Methode ist erkennbar. Bei den Experimenten unterschied sich die Anzahl der Variablen zwischen den erweiterten Formulierungen linearer Größe wie erwartet um einen Wert größer $|E|$. Dieser Wert war nicht genau $|E|$, da bei der EFSE-Formulierung nicht für alle Kanten z-Variablen existierten. Weiterhin wurden die Nebenbedingungen der EFMST-Methode anders als im Abschnitt 2.5.2 dargestellt in die Routine eingepflegt. Die ersten beiden Summen sind aufaddiert als eine Summe in das LP eingeflossen. Darum bestand hier auch nur der Unterschied von circa $|V|$ Constraints zu Beginn zwischen den erweiterten Formulierungen linearer Größe. Die Unterschiede in der Variablenanzahl zwischen den Testgruppen unterschiedlicher Gradbeschränkung bei gleicher Graphengröße der MST-Methode lassen sich damit erklären, dass lediglich die Hälfte der Testinstanzen mit der geringsten Rechenzeit für die Auswertung herangezogen wurde.

Bei den übrigen Parametern fällt zunächst ein deutlicher Unterschied zwischen den Testinstanzen mit $k = 2$ und den Gruppen mit $k \neq 2$ innerhalb der Lösungsmethoden auf. Wir konnten diese Diskrepanz auch bezüglich der Laufzeiten im vorangegangenen Abschnitt feststellen. Betrachten wir nun die LP-Lösungszeiten und die Anzahl der durchgeführten LP-Iterationen. Hier werden zunächst die erwartet höheren Werte der CEFMST- im Vergleich zur EFMST- und EFSE-Methode deutlich. Auch die Separation weißt hier höhere Werte als die lineare erweiterte Formulierung für $k \neq 2$ auf. Bei Gradschranke 2 änderte sich das Lösungsverhalten der Separationsroutine. Es wurden deutlich weniger LPs gelöst, dafür wurde stärker gebrancht. Auch die übrigen Methoden übten lediglich für $k = 2$ Branchingschritte aus. Jedoch erhöhte sich bei diesen die Anzahl der gelösten LPs und auch die dafür aufgewendete Zeit. Beim Vergleich der beiden erweiterten Formulierungen linearer Größe kann die EFSE-Methode mit leicht geringeren Werten aufwarten.

Werfen wir nun einen Blick auf das Branchingverhalten der verschiedenen Lösungsansätze. Die Werte des Wurzel-Dual-Bound-Fehler und der Anzahl der gelösten LPs pro Knoten sind so zu verstehen, dass nur diejenigen Testinstanzen in den Durchschnittswert einflossen, bei denen Branching durchgeführt wurde. Bis auf wenige Ausnahmen bei der Separation wurde erst für $k = 2$ gebrancht. Dabei war der Fehler der Dual Bound im Wurzelknoten von etwa gleicher Güte für alle Methoden für Gradschranke 2. Bei der Separation für

Gradbeschränkung ungleich 2 war der Fehler am Wurzelknoten etwas höher. Die erweiterten Formulierungen branchten in geringem Maße, wobei die kubische Variante etwa doppelt so viele Knoten pro Testinstanz erstellte wie die linearen Ansätze und auch deutlich mehr LPs pro Knoten löste. Bei den linearen Methoden wird ein großer Unterschied in der Anzahl der gelösten LPs pro Knoten deutlich. Dieser Wert ist bei der EFMST-Methode größer. Auffällig ist hier auch, dass die Separationsmethode für die Gradschranke 2 zwar mehr Knoten erstellte als die linearen erweiterten Formulierungen, sie jedoch weniger LPs pro Knoten löste.

Die ermittelten Parameter des für die Separation verwendeten Constraint-Handlers zeigen den zuvor notierten Unterschied zwischen Gradbeschränkung gleich und ungleich 2 auf. Wenn man die Anzahl der hinzugefügten Cuts mit der Anzahl der Constraints zu Beginn der linearen erweiterten Formulierungen vergleicht, so liegt hier der Vorteil bei der Separationsmethode – außer für Gradschranke 2. Die benötigte Zeit des Constraint-Handlers erscheint mit etwa der Hälfte der ermittelten Lösungszeit etwas hoch, weshalb hier noch Verbesserungsmöglichkeiten bestehen. Wurde jedoch im Rahmen der Voruntersuchungen der Einfluss des Handlers eingeschränkt – indem weniger Cuts pro Iteration gesucht und hinzugefügt wurden – so erhöhte sich die Gesamtlösungszeit deutlich.

Wir blicken nun auf die soeben betrachteten Parameter für Graphen mit 400 Knoten, die in den Tabellen 8 und 9 aufgelistet sind. Aufgrund der hohen Lösungszeiten wurden die CEFMST-Methode und die Separation für Gradschranke 2 nicht getestet. Auch hier startete die Separation mit sehr wenigen Bedingungen und die EFSE-Methode war im Vergleich zur EFMST-Methode etwas kleiner.

Das Verhältnis bezüglich der Anzahl der LP-Iterationen und der aufgewendeten LP-Lösungszeit ist zwischen der Separation und den erweiterten Formulierungen tendenziell wie bei einer Graphengröße von 100 Knoten. Der Faktor, um den die MST-Methode mehr LPs löste als die EFMST-Methode, hat sich jedoch von etwa 4 auf etwa 10 erhöht. Dementsprechend erhöhte sich auch der Faktor bezüglich der LP-Lösungszeit. Ebenso verhält es sich bei dem Vergleich der beiden erweiterten Formulierungen. Deutlich wird erneut der Unterschied zwischen Tests mit Gradschranke ungleich und gleich 2.

Beim Branchingverhalten der Methoden ist erkennbar, dass die Separation schon für Testinstanzen mit Gradschranke ungleich 2 in geringem Maße branchte. Im Gegensatz dazu nutzte die EFMST-Methode erst für Gradschranke 2 das Branch-and-Bound Verfahren. Dann wurden im Vergleich zu den kleineren Testinstanzen deutlich mehr Knoten betrachtet und mehr LPs pro Knoten gelöst. Im Vergleich zur EFSE-Methode, die auch stärker branchte als noch bei den kleineren Graphen, fallen beide Werte etwa doppelt so groß aus. Der Dual-Bound-Fehler im Wurzelknoten liegt für beide erweiterten Formulierungen auf dem gleichen Niveau und ist leicht besser als bei den 100-Knoten Graphinstanzen.

Tabelle 8: Auswertung für Zufallsgraphen einer Größe von 400 Knoten unter den verschiedenen Lösungsverfahren und ohne anliegende Gradschranke bzw. Gradschranke 8

	ohne Gradschranke		Gradschranke 8	
Parameter	Separation	EFMST	Separation	EFMST
Laufzeit	282.64	0.24	276.69	0.33
# Constraints Beginn	1.00	2 345	401	2 745
# Variablen	1 173	4 690	1 172	4 690
LP Lsg.-Zeit	172.69	0.12	170.45	0.15
# LP-Iterationen	386 062	3 123	378 879	3 262
# Wurzel-Kn. Iter.[1]	385 570	3 123	378 129	3 262
# B&B-Knoten	1.92	1.00	2.34	1.00
Wurzel-DB-Fehler[2]	0.54	0	0.48	0
# LP pro Knoten[3]	2.73	0	3.13	0
# Cuts Handler[4]	9 302	-	9 116	-
Zeit Handler[5]	97.55	-	97.01	-

Anmerkungen. dargestellte Parameter sind geometrische Mittel, dargestellte Zeiten sind geometrische Mittel in Sekunden [1] # Wurzelknoten Iterationen, [2] Fehler der Dual-Bound im Wurzelknoten in %, [3] ohne Wurzelknoten, [4] # hinzugefügter Cuts des eigenen Constraint-Handlers, [5] vom Constraint-Handler beanspruchte Zeit

Tabelle 9: Auswertung für Zufallsgraphen einer Größe von 400 Knoten unter den verschiedenen Lösungsverfahren und Gradschranke 2 und 3

	Gradschranke 2		Gradschranke 3	
Parameter	EFMST	EFSE	Separation	EFMST
Laufzeit	661.67	117.10	287.10	0.30
# Constraints Beginn	2 745	2 336	401	2 745
# Variablen	4 689	3 518	1 173	4 690
LP Lsg.-Zeit	362.26	38.87	178.84	0.15
# LP-Iterationen	884 827	130 790	378 748	3 232
# Wurzel-Kn. Iter.[1]	16 635	9 658	378 381	3 232
# B&B-Knoten	5 724	2 113	1.70	1.00
Wurzel-DB-Fehler[2]	0.13	0.21	0.40	0
# LP pro Knoten3	135	51.97	2.50	0
# Cuts Handler[4]	-	-	8 967	-
Zeit Handler[5]	-	-	94.73	-

Anmerkungen. dargestellte Parameter sind geometrische Mittel, dargestellte Zeiten sind geometrische Mittel in Sekunden [1] # Wurzelknoten Iterationen, [2] Fehler der Dual-Bound im Wurzelknoten in %, [3] ohne Wurzelknoten, [4] # hinzugefügter Cuts des eigenen Constraint-Handlers, [5] vom Constraint-Handler beanspruchte Zeit

Im Hinblick auf das Constraint-Handler-Verhalten bemerken wir einen erwarteten Anstieg der hinzugefügten Cuts und der aufgewendeten Zeit des Constraint-Handlers. Beide Parameter nehmen für die betrachteten Lösungen mit Gradschranke ungleich 2 etwa die gleichen Werte an. Das Verhältnis von Zeit des Constraint-Handlers und insgesamt aufgewendeter Lösungszeit hat sich leicht derart verändert, dass der Constraint-Handler nun weniger als die Hälfte der Gesamtzeit benötigte. Eine wie oben beschriebene Erhöhung der Aktivität des Handlers hatte in den Voruntersuchungen jedoch keinen positiven Laufzeiteinfluss.

In Tabelle 10 sind die Parameter für Graphen mit einer Größe von 25 000 Knoten, die mittels der EFMST-Methode bei verschiedenen Gradschranken bearbeitet wurden, dargestellt. Bis auf den Parameter LP-Lösungszeit sind alle Werte für die unterschiedlichen Gradschranken auf dem gleichen Niveau. Die LP-Lösungszeit war etwas geringer für die Testfälle ohne Gradbeschränkung, was auch die dort erzielte geringste Rechenzeit widerspiegelt. Bemerkenswert ist, dass gar kein Branching ausgeführt wurde – selbst bei großen Testinstanzen. Um den oben angesprochenen Einfluss des Presolvings bei der EFMST-Methode auch hier aufzugreifen, erfolgt in Tabelle 11 eine Auflistung der gemessenen Parameter für 25 000 Knoten-Graphen bei ausgeschaltetem Presolving. Es wird der Anstieg der Rechen- und LP-Lösungszeit bei Verringerung der anliegenden Gradschranke deutlich. Insgesamt sinken die beiden Werte und die Anzahl der hinzugefügten Cuts, außer bei $k = 3$, im Vergleich zu den Testläufen mit eingeschaltetem Presolving. Wie oben erwähnt erfolgten die Tests jedoch mit Presolving, da dies einen positiven Effekt für $k = 2$ hatte. Erkennbar ist diese Tendenz schon für $k = 3$.

Um Nichtkonsistenz der angestellten Beobachtungen für die TSPLIB-Graphen auszuschließen, betrachten wir die Parameter für TSPLIB-Graphen mit höchstens 100 Knoten unter den verschiedenen Lösungsmethoden bei unterschiedlichen Gradbeschränkungen. Diese sind in die Tabellen 12 und 13 dargestellt. Alles in allem können wir die gleichen Verhältnisse innerhalb der Lösungsmethoden bei Änderung der Gradschranke – insbesondere von ungleich auf gleich 2 – sowie zwischen den verschiedenen Lösungsansätzen bei gleicher Gradschranke feststellen.

Tabelle 10: Auswertung für Zufallsgraphen einer Größe von 25 000 Knoten unter der EFMST-Methode und ohne anliegende Gradschranke bzw. Gradschranke 3, 4 und 8 mit eingeschaltetem Presolving

	ohne Gradschranke	Gradschranke 8	Gradschranke 4	Gradschranke 3
Laufzeit	662.79	845.89	750.95	910.37
# Constraints Beginn	149 793	174 793	174 793	174 793
# Variablen	299 586	299 586	299 586	299 586
LP Lsg.-Zeit	590.81	798.15	710.59	787.37
# LP-Iterationen	195 724	198 126	196 735	202 368
# Wurzel-Kn. Iter.[1]	195 724	198 126	196 735	202 368
# B&B-Knoten	1.00	1.00	1.00	1.00
Wurzel-DB-Fehler[2]	0	0	0	0
# LP pro Knoten[3]	0	0	0	0

Anmerkungen. dargestellte Parameter sind geometrische Mittel, dargestellte Zeiten sind geometrische Mittel in Sekunden
[1] # Wurzelknoten Iterationen, [2] Fehler der Dual-Bound im Wurzelknoten in %, [3] ohne Wurzelknoten

Tabelle 11: Auswertung für Zufallsgraphen einer Größe von 25 000 Knoten unter der EFMST-Methode und ohne anliegende Gradschranke bzw. Gradschranke 3, 4 und 8 mit ausgeschaltetem Presolving

	ohne Gradschranke	Gradschranke 8	Gradschranke 4	Gradschranke 3
Laufzeit	644.52	668.53	677.95	923.93
# Constraints Beginn	149 793	174 793	174 793	174 793
# Variablen	299 586	299 586	299 586	299 586
LP Lsg.-Zeit	571.04	628.33	640.31	800.92
# LP-Iterationen	194 293	193 807	193 574	201 767
# Wurzel-Kn. Iter.[1]	194 293	193 807	193 574	201 767
# B&B-Knoten	1.00	1.00	1.00	1.00
Wurzel-DB-Fehler[2]	0	0	0	0
# LP pro Knoten[3]	0	0	0	0

Anmerkungen. dargestellte Parameter sind geometrische Mittel, dargestellte Zeiten sind geometrische Mittel in Sekunden
[1] # Wurzelknoten Iterationen, [2] Fehler der Dual-Bound im Wurzelknoten in %, [3] ohne Wurzelknoten

Tabelle 12: Auswertung für TSPLIB-Graphen einer Größe von höchstens 100 Knoten unter den verschiedenen Lösungsverfahren und ohne anliegende Gradschranke bzw. Gradschranke 8

	ohne Gradschranke			Gradschranke 8		
Parameter	Sep.[1]	EFMST	CEFMST	Sep.[1]	EFMST	CEFMST
Laufzeit	0.28	0.03	13.45	0.29	0.04	13.88
# Constraints Beginn	1.00	338	11 734	64	399	11 820
# Variablen	169	675	10 118	187	675	10 118
LP Lsg.-Zeit	0.04	0.01	12.34	0.04	0.01	12.58
# LP-Iterationen	964	444	16 623	964	444	16 590
# Wurzel-Kn. Iter.[2]	939	444	16 623	939	444	16 590
# B&B-Knoten	1.48	1.00	1.00	1.48	1.00	1.00
Wurzel-DB-Fehler[3]	0	0	0	0	0	0
# LP pro Knoten[4]	3.71	0	0	3.71	0	0
# Cuts Handler[5]	144	-	-	144	-	-
Zeit Handler[6]	0.003	-	-	0.05	-	-

Anmerkungen. dargestellte Parameter sind geometrische Mittel, dargestellte Zeiten sind geometrische Mittel in Sekunden
[1] Separation, [2] # Wurzelknoten Iterationen, [3] Fehler der Dual-Bound im Wurzelknoten in %, [4] ohne Wurzelknoten,
[5] # hinzugefügter Cuts des eigenen Constraint-Handlers, [6] vom Constraint-Handler beanspruchte Zeit

Tabelle 13: Auswertung für TSPLIB-Graphen einer Größe von höchstens 100 Knoten unter den verschiedenen Lösungsverfahren und Gradschranke 2 und 3

	Gradschranke 2				Gradschranke 3		
Parameter	Sep.[1]	EFMST	EFSE	CEFMST	Sep.[1]	EFMST	CEFMST
Laufzeit	0.62	0.26	0.16	46.43	0.24	0.04	14.31
# Constraints Beginn	64	399	327	11 820	64	399	11 820
# Variablen	169	675	506	10 118	169	675	10 118
LP Lsg.-Zeit	0.03	0.06	0.03	40.85	0.05	0.01	12.74
# LP-Iterationen	581	1 139	781	54 438	857	453	16 677
# Wurzel-Kn. Iter.[2]	300	997	754	46 112	851	453	16 677
# B&B-Knoten	12.79	3.82	1.69	2.44	1.19	1.00	1.00
Wurzel-DB-Fehler[3]	0.54	0.37	0.46	0.51	0.97	0	0
# LP pro Knoten[4]	6.04	33.53	20.35	2 193	22.39	0	0
# Cuts Handler[5]	390	-	-	-	108	-	-
Zeit Handler[6]	0.13	-	-	-	0.003	-	-

Anmerkungen. dargestellte Parameter sind geometrische Mittel, dargestellte Zeiten sind geometrische Mittel in Sekunden
[1] Separation, [2] # Wurzelknoten Iterationen, [3] Fehler der Dual-Bound im Wurzelknoten in %, [4] ohne Wurzelknoten,
[5] # hinzugefügter Cuts des eigenen Constraint-Handlers, [6] vom Constraint-Handler beanspruchte Zeit

4 Fazit

Wir halten als Hauptergebnis dieser Arbeit die deutliche Laufzeitüberlegenheit der beiden erweiterten Formulierungen linearer Größe im Vergleich zu der implementierten Separationsmethode fest. Die detaillierte Auswertung einzelner Parameter hat zudem aufgezeigt, dass die EFMST- und EFSE-Methoden weniger LP-Iterationen ausführten und weniger Zeit für das LP-Lösen aufwendeten – außer wenn bei Gradschranke 2 die Separation verstärkt branchte und somit LP-Zeit einsparte. Im Branchingverhalten wiesen die einzelnen Ansätze in etwa das gleiche Muster auf: Bei $k \neq 2$ wurde kaum (Separation) bis gar nicht (erweiterte Formulierungen) gebrancht. Für $k = 2$ haben alle Ansätze Branch-and-Bound ausgeführt. Wenn man als Ausdruck der Größe der Separations-Problembeschreibungen die Anzahl der hinzugefügten Cuts mit der Anzahl der Constraints der erweiterten Formulierungen vergleicht, so lag hier der Vorteil meist bei den EFMST- und EFSE-Methoden. Die erweiterte Formulierung kubischer Größe benötigte die erwarteten größten Laufzeiten, den größten LP-Lösungsaufwand und die größten Problembeschreibungen. Im Vergleich der beiden Formulierungen linearer Größe konnte die etwas kompaktere EFSE-Methode mit den besseren Werten aufwarten. Auf diese Art kann man in der vorliegenden Arbeit den Einfluss, den die Komplexität der erweiterten Formulierung auf das Lösungsverhalten hat, ausmachen.

Um verzerrende Effekte auf den selbst generierten Grapheninstanzen auszuschließen, wurden alle durchgeführten Tests auch auf TSPLIB-Graphen ausgeführt. Dabei konnten wir tendenziell die gleichen Verhältnisse auf beiden Testinstanzarten feststellen.

Trotz der positiven Resultate, die die erweiterten Formulierungen erzielt haben, sollte man darauf hinweisen, dass die Separationsmethode Raum für Verbesserungen lässt. In Abschnitt 3.1.2 wurden kurz mögliche Ansätze dafür angesprochen. Der Fokus dieser Arbeit lag jedoch auf der Untersuchung der erweiterten Formulierungen. Im Hinblick auf praktische Anwendungen sind die EFMST- und EFSE-Ansätze etwas eingeschränkt, da sie die Planarität der Graphen, auf denen sie operieren sollen, voraussetzen. Es wurde auch der immer noch sehr große Laufzeitnachteil im Vergleich zu einer Implementierung des Kruskal-Algorithmus deutlich. Durch die hier angestellten Untersuchungen haben sich erweiterte Formulierungen generell jedoch als praktisch interessante Optimierungsmethoden und als vielversprechende Themen künftiger Forschungsarbeiten herausgestellt.

Literatur

[1] Achterberg, T.: *SCIP: Solving Constraint Integer Programs*. Mathematical Programming Computation, 1(1):1–41, 2009.

[2] Applegate, D. L., Bixby, R. E., Chvátal, V. und Cook, W. J.: *The Traveling Salesman Problem. A Computational Study*. Princeton University Press, Princeton, 2007.

[3] B.Korte und Vygen, J.: *Kombinatorische Optimierung. Theorie und Algorithmen*. Springer-Verlag, Berlin, 2008.

[4] Computational Geometry Algorithms Library. `http://www.cgal.org`, letzter Zugriff: 05/08/2012.

[5] Deo, N.: *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall of India, Neu Delhi, 2004.

[6] Dezső, B., Jüttner, A. und Kovács, P.: *LEMON - An Open Source C++ Graph Template Library*. Electronic Notes in Theoretical Computer Science, 264(5):23–45, 2011.

[7] Edmonds, J.: *Submodular Functions, Matroids and Certain Polyhedra*. In: Guy, R., Hanani, H., Sauer, N. und Schönheim, J. (Herausgeber): *Combinatorial Structures and Their Applications; Proceedings of the Calgary International on Combinatorial Structures and Their Applications 1969*, Seiten 69–87. Gordon and Breach, New York, 1970.

[8] Farkas, G.: *A Fourier-féle mechanikai elv alkalmazásai*. Mathematikai és Természettudományi Értesitö, 12:457–472, 1894.

[9] Fortune, S.: *Voronoi-Diagrams and Delaunay-Triangulations*. In: Du, D.Z. und Hwang, F. (Herausgeber): *Computing in Euclidean Geometry*, Band 4 der Reihe *Lecture Notes on Computing*, Seiten 225–265. World Scientific Publishing, Singapur, 1995.

[10] Gawrilow, E. und Joswig, M.: *Polymake: A Framework for Analyzing Convex Polytopes*. In: Kalai, G. und Ziegler, G. M. (Herausgeber): *Polytopes. Combinatorics and Computation*, Seiten 43–74. Birkhäuser, Basel, 2000.

[11] Ghouila-Houri, A.: *Caractérisation des Matrices Totalement Unimodulaires*. Comptes Rendus de l'Académie des Sciences, 254:1192–1194, 1962.

[12] Hopcroft, J.E. und Tarjan, R.E.: *Efficient Planarity Testing*. Journal of the ACM, 21:549–568, 1974.

[13] Kaibel, V., Pashkovich, K. und Theis, D.O.: *Symmetry Matters for Sizes of Extended Formulations*. arXiv:0911.3712v1 [math.CO], November 2009.

[14] Karp, R. M.: *Reducibility Among Combinatorial Problems*. In: Miller, R. E. und Thatcher, J. W. (Herausgeber): *Complexity of Computer Computations; Proceedings of a Symposium on the Complexity of Computer Computations*, Seiten 85–103, New York, 1972.

[15] Klein, R.: *Algorithmische Geometrie*. Addison-Wesley-Longman, Bonn, 1997.

[16] Kruskal, J. B.: *On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem*. Proceedings of the American Mathematical Society, 7:48–50, 1956.

[17] Kuratowski, K.: *Sur le Problème des Courbes Gauches en Topologie*. Fundamenta Mathematicae, 15:271–283, 1930.

[18] Martin, R. K.: *Using Separation Algorithms to Generate Mixed Integer Model Reformulations*. Technischer Bericht, University of Chicago, 1987.

[19] MATLAB: *Version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, 2010.

[20] Maurer, J. und Watanabe, S.: *Boost Random Number Library*. `http://www.boost.org/doc/libs/1_50_0/doc/html/boost_random.html`, letzter Zugriff: 05/08/2012.

[21] Mittelmann, H.: *Mixed Integer Linear Programming Benchmark*. `http://plato.asu.edu/ftp/milpc.html`, letzter Zugriff: 05/08/2012.

[22] Okabe, A., Boots, B. und Sugihara, K.: *Spatial Tessellations. Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, 1992.

[23] Padberg, M. und Rinaldi, G.: *An Efficient Algorithm for the Minimum Capacity Cut Problem*. Mathematical Programming, 47:19–36, 1990.

[24] Pashkovich, K.: *Extended Formulations for Combinatorial Polytopes*. Dissertation, Otto-von-Guericke-Universität Magdeburg, 2012.

[25] Reinelt, G.: *TSPLIP - A Traveling Salesman Problem Library*. INFORMS Journal on Computing, 3(4):376–384, 1991. `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95`, letzter Zugriff: 05/08/2012.

[26] Schrijver, A.: *Combinatorial Optimization. Polyhedra and Efficiency*. Springer, Berlin, 2003.

[27] Wagner, K.: *Über eine Eigenschaft der komplexen Ebene*. Mathematische Annalen, 114:570–590, 1937.

[28] Williams, J. C.: *A Linear-Size Zero-One Programming Model for the Minimum Spanning Tree Problem in Planar Graphs*. Networks, 39(1):53–60, 2001.

[29] Wunderling, R.: *Paralleler und objektorientierter Simplex-Algorithmus*. Dissertation, Technische Universität Berlin, 1996.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Magdeburg, 31. August 2012

Stephan Sorgatz