Fakultät für Mathematik und Informatik
Lehrgebiet Diskrete Mathematik und Optimierung

Ein effizienter Algorithmus für Min-Cost-Flow in regulären Matroiden

Diplomarbeit

Betreuer: Prof. Dr. Winfried Hochstättler
Bearbeiter: Dieter Weninger
Linden-Hauptstr. 17
91459 Markt Erlbach
Matrikelnummer 6275281
Studiengang: Diplom II Informatik mit Nebenfach Mathematik
Datum: 10. März 2014

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Linden, 10. März 2014

(Dieter Weninger)

Danksagung

Zuerst möchte ich mich bei meinem Betreuer Prof. Dr. Winfried Hochstättler für die Unterstützung bei der Anfertigung der Diplomarbeit herzlich bedanken. Er hat nicht nur mein Interesse für das Forschungsthema *Flüsse in regulären Matroiden* geweckt, sondern mir in Problemsituationen per Email oder Telefon stets geholfen.

Ein weiterer Dank geht an Maximilian Merkert und meine Tochter Mira. Beide haben Teile der Arbeit Korrektur gelesen.

Abschließend danke ich meiner Frau Ute, meinen Kindern Mira, Naima, Jonathan, meinen Eltern und meinen Schwiegereltern. Gleichzeitig Familienvater zu sein, einen Vollzeitberuf zu haben, ein Fernstudium zu absolvieren und ein Haus zu renovieren ist mir schwer gefallen, aber ihr habt immer zu mir gehalten und mich motiviert.

Kurzreferat

In der vorliegenden Arbeit wird, basierend auf einigen Grundlagen zu Graphen und Matroiden, ein Algorithmus für das Min-Cost-Flow Problem in regulären Matroiden entwickelt. Ausgehend von einem maximalen Matroidfluss werden Kreise mit negativen durchschnittlichen Kosten in einem Residualmatroid bestimmt und entsprechend der Matroidfluss augmentiert, um einen optimalen Matroidfluss zu erhalten. Wir betrachten verschiedene Kostenfunktionen und analysieren die entsprechenden Laufzeiten des Algorithmus. Abschließend wird eine Implementierung des Algorithmus vorgestellt.

Inhaltsverzeichnis

1. Einleitung **1**

2. Grundlagen **3**
2.1. Kodierungslänge, Speicherbedarf und Laufzeit 3
2.2. Lineare Optimierung 4
2.2.1. Dualität 4
2.2.2. Streng polynomialer Algorithmus 6
2.3. Graphen 7
2.3.1. Grundlegende Definitionen 7
2.3.2. Netzwerke 10
2.3.3. Kürzester Pfad 13
2.3.4. Maximaler Fluss 14
2.3.5. Min-Cost-Flow 17
2.4. Matroide 21
2.4.1. Grundlegende Definitionen und Axiome 21
2.4.2. Dualität und Minoren 26
2.4.3. Reguläre Matroide 30
2.4.4. Orakel 39
2.4.5. Greedy-Algorithmus 40

3. Herleitung des Algorithmus **42**
3.1. Min-Cost-Flow mit konvexer separabler Kostenfunktion 42
3.2. Min-Mean-Circuit-Canceling Methode 45
3.2.1. Konvergenz 45
3.2.2. Konvexe separable Kostenfunktionen 46
3.3. Negative Kreise 50
3.4. Maximaler Matroidfluss 53
3.5. Bestimmung eines Min-Mean-Circuit 57
3.6. Algorithmus für Min-Cost-Flow in regulären Matroiden 64

4. Implementierung des Algorithmus **67**
4.1. Entwickeltes Programm 67
4.2. Signierung 71
4.3. Testinstanzen 72

5. Zusammenfassung **81**

A. Testinstanzen **86**
A.1. Graphisches Matroid 86
A.2. Graphisches Matroid $M(K_{3,3})$ 88
A.3. Kographisches Matroid $M(K_{3,3})^*$ 89
A.4. Kographisches Matroid $M(K_5)^* \oplus_2 M(K_{3,3})^*$ 91
A.5. Reguläres Matroid $M(K_5) \oplus_2 R_{10} \oplus_2 M(K_{3,3})$ 93
A.6. Reguläres Matroid $M(K_{3,3})^* \oplus_3 M(K_{3,3})$ 95
A.7. Reguläres Matroid $M(K_5) \oplus_3 M(K_5)^*$ 97

B. Inhalt CD **99**

1. Einleitung

Die *Kombinatorische Optimierung* ist ein Teilgebiet der diskreten Mathematik. Sie spielt in vielen Bereichen wie z.B. Operations Research eine wichtige Rolle. In der kombinatorischen Optimierung geht es darum, aus einer Menge von diskreten Elementen, eine Teilmenge zu bestimmen, die bestimmen Anforderungen entspricht und bezüglich einer Bewertungsfunktion optimal ist. Solche Fragestellungen treten in der Praxis sehr häufig auf. Beispielsweise ist das Finden einer kürzesten Route zwischen zwei Städten ein Problem, welchem in der kombinatorischen Optimierung eine zentrale Rolle zukommt.

Wir werden in dieser Arbeit mit den beiden Strukturen *Graphen* und *Matroide* arbeiten. Ein Graph ist eine abstrakte Struktur, die eine Menge von Objekten zusammen mit Verbindungen zwischen diesen Objekten darstellt. Die Objekte werden üblicherweise durch Knoten und die Verbindungen durch Kanten dargestellt. Die Kanten können sowohl ungerichtet als auch gerichtet sein. Hingegen ist ein Matroid eine mathematische Struktur, mit deren Hilfe der Begriff der Unabhängigkeit aus der linearen Algebra verallgemeinert wird. Die von Graphen induzierten Matroide stellen die Klasse der graphischen Matroide dar. Somit können wir Matroide auch als eine Verallgemeinerung von Graphen auffassen. Sowohl Graphen, als auch Matroide stellen wichtige Strukturen der kombinatorischen Optimierung dar.

Das *Min-Cost-Flow Problem* ist ein Flussproblem. Hier geht es darum, einen bestimmten Fluss so zu bestimmen, dass unter Einhaltung von Flussbedingungen und Kapazitätsbeschränkungen die Kosten für den Fluss minimal sind. Typische Anwendungen sind Transport- oder Distributionsaufgaben. Das Min-Cost-Flow Problem wurde in der Vergangenheit eingehend in Graphen untersucht und es wurden mehrere Algorithmen mit Laufzeitanalysen dafür entwickelt (siehe Schrijver [32] Seiten 177-197). In Tardos [37] wurde erstmals ein streng polynomialer Algorithmus zum Lösen des Min-Cost-Flow Problems in Graphen angegeben. Ein Jahr später veröffentlichte Tardos [38] einen streng polynomialen Algorithmus zur Berechnung kombinatorischer linearer Programme. Damit war es möglich ein, als lineares Programm formuliertes, Min-Cost-Flow Problem in streng polynomialer Zeit zu lösen. Auch wurde das Min-Cost-Flow Problem innerhalb regulärer Matroide betrachtet und Algorithmen dafür entwickelt (siehe Burkard/Hamacher [5]). Unter den entwickelten Algorithmen für reguläre Matroide befindet sich aber kein Algorithmus mit Laufzeitanalyse. Hauptgegenstand der Arbeit ist es deshalb, einen Algorithmus für Min-Cost-Flow in regulären Matroiden mit Laufzeitanalyse anzugeben. Der von uns entwickelte Algorithmus ist nicht rein kombinatorischer Natur. Da er aber sukzessive lineare Programme löst, um den optimalen Fluss zu bestimmen, ist er strukturell einem rein kombinatorischen Algorithmus sehr ähnlich.

Die Arbeit gliedert sich in drei Teile. Im ersten Teil werden die Grundlagen besprochen, die zum Verständis des entwickelten Algorithmus notwendig sind. Hier gehen wir zuerst auf Basiswissen der Komplexitätstheorie ein. Dies benötigen wir im Verlauf der Arbeit, um Laufzeitanalysen von Algorithmen durchführen zu können. Dann gehen wir auf die Dualität und den eben schon erwähnten Algorithmus der Linearen Optimierung von Tardos [38] ein. Anschließend zeigen wir, dass dieser Algorithmus streng polynomial ist, falls die Nebenbedingungsmatrix des linearen Programms eine total unimodulare Matrix ist. Es folgt ein größerer Abschnitt über Graphen, in dem wir zuerst eine Verbindung zwischen der Graphentheorie und der linearen Algebra aufzeigen und dann auf die drei bekannten kombinatorischen Optimierungsprobleme *Kürzester Pfad*, *Maximaler Fluss* und *Min-Cost-Flow* eingehen. Nach dem Abschnitt über Graphen folgt ein Abschnitt über Matroide. Neben den gängigen Axiomen werden wir darin auf die Dualität, das Konzept der Minoren und die Klasse der reguläre Matroide zu sprechen kommen. Da uns im Kontext dieser Arbeit besonders die regulären Matroide interessieren, werden wir noch auf deren Signierung, Komposition und Dekomposition eingehen. Auch besprechen wir Grundlagen zu Flüssen in regulären Matroiden. Zudem wollen wir das Orakelkonzept und den Greedy-Algorithmus im Kontext von Matroiden ansprechen. Im zweiten Teil der Arbeit entwickeln wir, basierend auf den Grundlagen, einen Algorithmus für Min-Cost-Flow in regulären Matroiden. Zuerst werden wir dabei die Kostenfunktion verallgemeinern und eine konvexe separable Kostenfunktion betrachten. Dann wollen wir auf die *Min-Mean-Circuit-Canceling Methode* eingehen. Hier geht es darum, ausgehend von einer zulässigen Zirkulation, Kreise mit negativen minimalen durchschnittlichen Kosten zu finden und entlang dieser Kreise den Fluss zu augmentieren. Anschließend folgt ein Abschnitt über *negative Kreise*, welcher uns ein Argument liefert, wann ein Fluss, bezüglich einer Kostenfunktion, optimal ist. Wie wir anfänglich eine zulässige Zirkulation finden, soll in Abschnitt 3.4 gezeigt werden. Die Bestimmung eines *Min-Mean-Circuit* ist Gegenstand von Abschnitt 3.5. Grundlage für die Bestimmung eines Min-Mean-Circuit bildet ein Algorithmus von Karzanov [21]. Letztlich geben wir in Abschnitt 3.6 unseren entwickelten Algorithmus für Min-Cost-Flow in regulären Matroiden an. Die Entwicklung des Algorithmus findet anhand von Definitionen, Lemmata, Sätzen und Korollaren statt. Damit der Leser den Zusammenhang der mathematischen Aussagen leichter nachvollziehen kann, ist der Aufbau der Beweisführung in Abschnitt 3.6 als Abbildung 3.10 explizit dargestellt. Im dritten Teil der Arbeit gehen wir auf die Implementierung des entwickelten Algorithmus ein. Unter Verwendung von Kompositionsmethoden binärer Matroide ist es uns gelungen Testinstanzen zu erstellen, welche reguläre Matroide, aber weder graphisch noch kographisch sind. Daneben haben wir ein Programm entwickelt, mit dem man Repräsentationsmatrizen regulärer Matroide signieren kann. Dieses Programm haben wir genutzt um reguläre Matroide mit einer Orientierung zu versehen. Um die Korrektheit der Berechnungen auf den Testinstanzen des Algorithmus belegen zu können, haben wir im Anhang A äquivalente lineare Programme aufgestellt und mit einem kommerziellen Löser gegengerechnet. Der gesamte Quellcode und alle Testinstanzen liegen der Arbeit auf einer CD bei (siehe Anhang B).

2. Grundlagen

2.1. Kodierungslänge, Speicherbedarf und Laufzeit

Wir sagen, dass ein Algorithmus Problem Π löst, wenn er für jedes Problembeispiel $\mathcal{I} \in \Pi$ eine Lösung findet. Das Ziel des Algorithmenentwurfs ist es, möglichst effiziente Verfahren zur Lösung von Problemen zu finden. Um die Effizienz eines Algorithmus messen zu können, beschäftigen wir uns im Folgenden mit den Begriffen Kodierungslänge, Speicherbedarf und Laufzeit. Als Grundlage für dieses Kapitel verwenden wir Grötschel [13].

Die Laufzeit eines Algorithmus hängt zunächst von der Größe der Eingabedaten ab. Bevor wir also Laufzeitanalysen anstellen können, müssen wir eine Aussage darüber treffen, wie wir unsere Problembeispiele kodieren wollen. Die *Kodierungslänge* $\langle n \rangle$ einer ganzen Zahl $n \in \mathbb{Z}$ ist die Anzahl der zu ihrer Darstellung notwendigen Bits, d.h.

$$\langle n \rangle := \lceil \log_2(|n| + 1) \rceil + 1. \tag{2.1}$$

Jede rationale Zahl r hat eine Darstellung $r = p/q$ mit $p, q \in \mathbb{Z}$. Dabei sind p und q teilerfremd und $q > 0$. Die Kodierungslänge einer rationalen Zahl $r \in \mathbb{Q}$ ist

$$\langle r \rangle := \langle p \rangle + \langle q \rangle. \tag{2.2}$$

Abgeleitet von (2.1) und (2.2) ergibt sich für einen Vektor $x \in \mathbb{Q}^n$ die Kodierungslänge $\langle x \rangle := \sum_{i=1}^{n} \langle x_i \rangle$ und für eine Matrix $A \in \mathbb{Q}^{m \times n}$ die Kodierungslänge $\langle A \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} \langle x_{ij} \rangle$.

Die Anzahl der Bits, die während der Ausführung von Algorithmus A mindestens einmal benutzt wurden, wollen wir als den *Speicherbedarf* von A zur Lösung von $\mathcal{I}$ bezeichnen.

Die *Laufzeit* von A zur Lösung von $\mathcal{I}$ ist die Anzahl der elementaren Rechenoperationen, die A benötigt, um eine Lösung von $\mathcal{I}$ zu finden, multipliziert mit der Kodierungslänge der größten auftretenden Zahl. Dabei verstehen wir als elementare Rechenoperationen Lesen, Schreiben, Löschen, Addieren, Subtrahieren, Multiplizieren, Dividieren und Vergleichen.

Definition 2.1.1. *Sei A ein Algorithmus zur Lösung eines Problems Π.*

(i) Die Funktion $f_A : \mathbb{N} \to \mathbb{N}$, definiert durch

$$f_A(n) := \max\{\textit{Laufzeit von } A \textit{ zur Lösung von } \mathcal{I} \mid \mathcal{I} \in \Pi \textit{ und } \langle \mathcal{I} \rangle \leq n\},$$

heißt Laufzeitfunktion von A.

(ii) Die Funktion $s_A : \mathbb{N} \to \mathbb{N}$*, definiert durch*

$$s_A(n) := \max\{\textit{Speicherbedarf von } A \textit{ zur Lösung von } \mathcal{I} \mid \mathcal{I} \in \Pi \textit{ und } \langle \mathcal{I} \rangle \leq n\},$$

heißt Speicherplatzfunktion von A.

(iii) Der Algorithmus A hat eine polynomiale Laufzeit, wenn es ein Polynom $p : \mathbb{N} \to \mathbb{N}$ *gibt mit*

$$f_A(n) \leq p(n) \qquad \forall n \in \mathbb{N}.$$

Wir sagen f_A *ist von der Ordnung höchstens* n^k *(geschrieben* $f_A = O(n^k)$*), falls das Polynom p den Grad k hat.*

(iv) Der Algorithmus A hat polynomialen Speicherplatzbedarf, falls es ein Polynom $q : \mathbb{N} \to \mathbb{N}$ *gibt mit*

$$s_A(n) \leq q(n) \qquad \forall n \in \mathbb{N}.$$

Wir wollen noch anmerken, dass ein Algorithmus keine polynomiale Laufzeit haben kann, falls dessen Speicherplatzfunktion nicht durch ein Polynom beschränkbar ist. Dies ergibt sich daraus, dass jede Benutzung eines Speicherplatzes implizit in die Berechnung der Laufzeitfunktion eingeht. Im Folgenden betrachten wir deshalb ausschließlich die Laufzeit von Algorithmen.

2.2. Lineare Optimierung

Die lineare Optimierung ist ein mathematisches Verfahren, bei dem eine lineare Zielfunktion über einer Menge, welche durch lineare Gleichungen bzw. Ungleichungen eingeschränkt ist, optimiert wird. Da die von den linearen Gleichungen bzw. Ungleichungen definierte Menge konvex ist, kann die lineare Optimierung auch als ein Spezialfall der konvexen Optimierung angesehen werden. Die konvexe Menge kann man als Polyeder auffassen und dadurch Erkenntnisse der Polyedertheorie innerhalb der linearen Optimierung verwenden.

2.2.1. Dualität

In diesem Abschnitt möchten wir auf ein sehr mächtiges Werkzeug innerhalb der linearen Optimierung, die Dualität, eingehen. Im Wesentlichen wollen wir die Sätze der schwachen und starken Dualität, den Satz vom komplementären Schlupf und ein Optimalitätskriterium aufzeigen. Wir verwenden als Grundlage für die Ausführungen Hochstättler [17]. Begriffe wie *zulässig*, *beschränkt*, usw. setzen wir als bekannt vorraus.

Gegeben ist ein lineares Programm (2.3), welches wir auch als das *primale Programm* bezeichnen wollen.

$$\begin{array}{lrcl} \max & c^T x & & \\ \text{u.d.N.} & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{2.3}$$

$\mathbb{K}$ sei dabei ein Körper mit $A \in \mathbb{K}^{m\times n}, b \in \mathbb{K}^m$ und $c \in \mathbb{K}^n$. $x \in \mathbb{K}^n$ bezeichnen wir als *Primalvariablen*. Das lineare Programm

$$\begin{array}{lll} \min & u^T b & \\ \text{u.d.N.} & u^T A & \geq \quad c^T \end{array} \tag{2.4}$$

heißt das *duale Programm* zu (2.3). $u \in \mathbb{K}^m$ sind die *Dualvariablen*. Für den Zielfunktionswert der beiden linearen Programme gilt folgende Beziehung, die auch als *schwache Dualität* bezeichnet wird.

Lemma 2.2.1. *Ist $x \geq 0$ zulässig für das primale Programm (2.3) und u für das duale Programm (2.4), so gilt $c^T x \leq u^T b$ (Beweis Hochstättler [17]).*

Das Lemma 2.2.1 gibt es auch in einer verschärften Form.

Satz 2.2.1. *Ist das primale Programm zulässig und beschränkt, so gibt es für das primale und das duale Programm Optimallösungen x^* und u^* und es gilt: $c^T x^* = u^{*T} b$ (Beweis Hochstättler [17]).*

Haben wir eine duales Paar linearer Programme, bei dem Satz 2.2.1 gilt, dann können wir daraus auf den Satz vom *komplementären Schlupf* schließen.

Satz 2.2.2. *Seien $x^* \in \mathbb{K}^n_+$ mit $Ax^* = b$ und $u^* \in \mathbb{K}^m$ mit $u^{*T} A \geq c^T$. Dann sind x^*, u^* genau dann ein Paar von Optimallösungen des primalen bzw. dualen Programmes, wenn gilt:*

i) $x^*_i \neq 0 \Rightarrow c_i = (u^{*T} A)_i$ *bzw. äquivalent dazu*

ii) $c_i < (u^{*T} A)_i \Rightarrow x^*_i = 0$

(Beweis Hochstättler [17]).

Abschließend schauen wir uns ein Kriterium an, welches uns eine Aussage liefert, ob x Optimallösung von (2.3) ist. Hier wollen wir den Beweis explizit ausführen, denn dieser liefert uns eine Methode, um aus einer Optimallösung von (2.3), eine Optimallösung für (2.4) zu berechnen. In Abschnitt 3.5 werden wir dieses Resultat benutzen.

Satz 2.2.3. *Sei $A \in \mathbb{K}^{m\times n}$ eine Matrix mit vollem Zeilenrang und seien $c \in \mathbb{K}^n$, $b \in \mathbb{K}^m$. Wir betrachten das lineare Programm (2.3). Sei B eine zulässige Basis von (2.3). Dann ist die zugehörige zulässige Basislösung x eine Optimallösung des linearen Programms, wenn $c^T - c_B^T A_{.B}^{-1} A \leq 0$ gilt.*

Beweis: Wir bestimmen eine Optimallösung von (2.4) und setzen dazu $u^T = c_B^T A_{.B}^{-1}$. Dann gilt nach Voraussetzung $u^T A \geq c^T$, womit u für (2.4) zulässig ist und $u^T b = c_B^T A_{.B}^{-1} b = c_B^T x_B = c^T x$. Damit folgt die Behauptung aus Lemma 2.2.1. Aus Satz 2.2.1 folgt, dass u Optimallösung von (2.4) ist. □

Wenn B eine zulässige Basis ist, dann wird $c^T - c_B^T A_{.B}^{-1} A$ auch als die *reduzierten Kosten* bezeichnet.

2.2.2. Streng polynomialer Algorithmus

In Tardos [38] wurde ein streng polynomialer Algorithmus zum Lösen von kombinatorischen linearen Programmen veröffentlicht. Ein Algorithmus ist *streng polynomial*, wenn er die folgende Definition erfüllt. Dabei verstehen wir unter der Dimension der Eingabe die Anzahl der Eingabezahlen.

Definition 2.2.1. *Ein Algorithmus ist streng polynomial, falls*

(i) er nur elementare arithmetische Operationen wie Addition, Vergleich, Muliplikation, Division verwendet,

(ii) die Anzahl elementarer arithmetischer Operationen in der Dimension der Eingabe polynomial beschränkt ist und

(iii) wenn die Größe der Zahlen, die während des Algorithmus auftreten, polynomial in der Dimension der Eingabe und der Kodierungslänge (siehe Abschnitt 2.1) der Eingabe beschränkt ist.

Als nächstes wollen wir einen Satz von Tardos angeben, der in Schrijver [31] veröffentlicht ist. Wir setzen dabei die Notation bezüglich der Kodierungslänge aus Abschnitt 2.1 voraus.

Satz 2.2.4. *Es existiert ein Algorithmus, welcher ein gegebenes rationales lineares Programm* $\max\{cx \mid Ax \geq b\}$ *in höchstens* $p(\langle A\rangle)$ *elementaren arithmetischen Operationen auf durch* $\langle A\rangle + \langle b\rangle + \langle c\rangle$ *polynomial beschränkten Zahlen für ein Polynom* p *löst (Beweis Schrijver [31]).*

Der in Tardos [38] angegebene Algorithmus zum Lösen eines rationalen linearen Programms hat, unter Annahme von $n \geq m$, eine Laufzeit von

$$O(n^5 + n^3 \log \Delta(A) + n^2 p(\langle A\rangle)).$$

Dabei ist $\Delta(A)$ für $A \in \mathbb{K}^{m\times n}$ eine ganze Zahl mit $\Delta(A) \geq \max(|\det B|)$ für alle Submatrizen B von A.

Da wir im Folgenden reguläre Matroide betrachten, sind wir besonders an dem Fall interessiert, bei dem die Nebenbedingungsmatrix eine Form wie in der anschließenden Definition hat.

Definition 2.2.2. *Eine Matrix* A *über* $\mathbb{K}$ *ist total unimodular, wenn die Determinante jeder Submatrix von* A *den Wert 0, 1 oder* -1 *hat.*

Für eine total unimodulare Matrix folgt mit Satz 2.2.4 das folgende Korollar.

Korollar 2.2.1. *Für ein rationales, lineares Programm mit einer total unimodularen Nebenbedingungsmatrix existiert ein streng polynomialer Algorithmus (Beweis Schrijver [31]).*

Wenn die Nebenbedingunsmatrix A eine total unimodulare Matrix ist, dann ist $\Delta(A) = 1$. Zudem liegt $\langle A \rangle$ in $O(n^2)$. Damit ergibt sich für ein lineares Programm mit einer total unimodularen Nebenbedingungsmatrix für den Algorithmus von Tardos eine Laufzeit von $O(n^5 + n^4)$. Dieses Resultat werden wir später an einigen Stellen benutzen.

2.3. Graphen

In diesem Abschnitt gehen wir zunächst auf grundlegende Definitionen der Graphentheorie ein, da wir diese an einigen Stellen im Verlauf der Arbeit benötigen. Dann zeigen wir die Verbindung zwischen dem Modell der Graphen und der linearen Algebra auf. Anschließend beschreiben wir drei klassische Probleme der kombinatorischen Optimierung. Alle drei Probleme haben Entsprechungen in der Matroidtheorie und werden uns später wieder begegnen.

2.3.1. Grundlegende Definitionen

Ein *ungerichteter Graph* ist ein Tripel $G = (V, E, \Psi)$, wobei V und E endliche Mengen sind und $\psi : E \to \{X \subseteq V \mid |X| = 2\}$. Ein *gerichteter Graph* ist ebenfalls ein Tripel $D = (V, E, \Psi)$. V und E sind wieder endlich und $\Psi : E \to \{(v, w) \in V \times V \mid v \neq w\}$. Die Elemente von V heißen *Knoten* und diejenigen von E *Kanten*. Einen gerichteten Graphen bezeichnet man auch als *Digraph*. Meist lassen wir Ψ weg und schreiben einfach $G = (V, E)$ bzw. $D = (V, E)$. Ist v ein Endknoten einer Kante $e \in E$, dann ist v *inzident* mit e. Im gerichteten Fall sagt man, dass (v, w) in v *beginnt* und in w *endet*.

Wir wollen nun die Begriffe Kreis, Baum und Kokreis definieren. Später werden wir sehen, dass diese Begriffe eine zentrale Rolle bei der Darstellung von Matroiden spielen. Sei $G = (V, E)$ ein Graph. Eine *Kantenfolge* F in G ist eine Folge $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$, wobei $k \geq 0$ und $e_i = \{v_i, v_{i+1}\} \in E$ bzw. $e_i = (v_i, v_{i+1}) \in E$ für $i = 1, \ldots, k$. Gilt zusätzlich $e_i \neq e_j$ für alle $1 \leq i < j \leq k$, so heißt F ein *Kantenzug* in G. Gilt auch noch $v_1 = v_{k+1}$, so nennen wir den Kantenzug F *geschlossen*. Ein *Kantenweg* (engl. Path) ist ein Graph $P = (\{v_1, \ldots, v_{k+1}\}, \{e_1, \ldots, e_k\})$ mit der Eigenschaft, dass $v_i \neq v_j$ für $1 \leq i < j \leq k + 1$ und die Folge $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$ ein Kantenzug ist. Wir nennen $v, w \in V$ *verbindbar*, wenn eine Kantenfolge von v nach w existiert. Sind je zwei Knoten $v, w \in V$ verbindbar, so heißt G *zusammenhängend*.

Definition 2.3.1. *Ein Kreis ist ein Graph $G = (V, E)$ mit der Eigenschaft, dass die Folge $v_1, e_1, v_2, \ldots, v_k, e_k, v_1$ ein geschlossener Kantenzug mit $k \geq 2$ und $v_i \neq v_j$ für $1 \leq i < j \leq k$ ist.*

Definition 2.3.2. *Ein Graph $G = (V, E)$ heißt Baum, wenn je zwei Knoten durch genau einen Kantenweg verbindbar sind.*

Wenn ein Graph ein Baum ist, dann ist er minimal zusammmenhängend und kreislos (siehe Seip [33] Seiten 117-118). Ein ungerichteter Graph ohne Kreise heißt *Wald*. Ein

Untergraph B von G heißt G *aufspannender Baum*, wenn B ein Baum ist und es eine Kantenteilmenge $E^* \subset E$ gibt mit $B = G \cap E^* := (V, E^*)$. Wenn G zusammenhängend ist, dann existiert immer auch ein G aufspannender Baum (siehe Seip [33] Seite 121). Der Graph $G \setminus V' = G \cap (V \setminus V')$ heißt der *Kograph* von $G \cap V'$ bzw. V'.

Definition 2.3.3. *Sei $G = (V, E)$ ein zusammenhängender Graph und $T \subset E$ eine Teilmenge der Kanten von G. T heißt T trennende Kantenmenge von G, wenn der Kograph $G \setminus T$ unzusammenhängend ist. Eine minimale trennende Kantenmenge heißt ein Kokreis (Minimalschnitt) von G.*

Definition 2.3.4. *Es sei $G = (V, E)$ ein zusammenhängender Graph und $B^* = (V, E^*) = G \cap E^*$ sei ein G aufspannender Baum mit der Kantenmenge $E^* \subset E$. Weiter sei $E^{**} = E \setminus E^*$ die komplementäre Kantenmenge von E^* bezüglich E. Dann heißt der Kograph $G \setminus E^* = (V, E^{**}) = G \cap E^{**}$ der Kobaum zu B^* und wird mit B^{**} bezeichnet.*

Wir zeigen nun, dass jede Kante des Kobaums einen Kreis bestimmt, der sie als einzige Kante des Kobaums enthält. Umgekehrt bestimmt jede Kante des aufspannenden Baums einen Kokreis, der sie als einzige Kante des aufspannenden Baums enthält.

Satz 2.3.1. *Es sei $G = (V, E)$ ein zusammenhängender Graph mit $|V| > 1$. Weiter sei $B^* = G \cap E^*$ ein G aufspannender Baum und $B^{**} = G \cap E^{**}$ der Kobaum zu B^*, dann gelten folgende Aussagen:*

(i) Für jedes $e^{**} \in E^{**}$ enthält $B^* + e^{**}$ genau einen Kreis von G und

(ii) für jedes $e^* \in E^*$ enthält $B^{**} + e^*$ genau einen Kokreis von G.

(Beweis Seip [33] Seite 149).

Beispiel 2.3.1. Wir wollen Satz 2.3.1 an einem Beispiel aufzeigen (siehe Abb. 2.1).

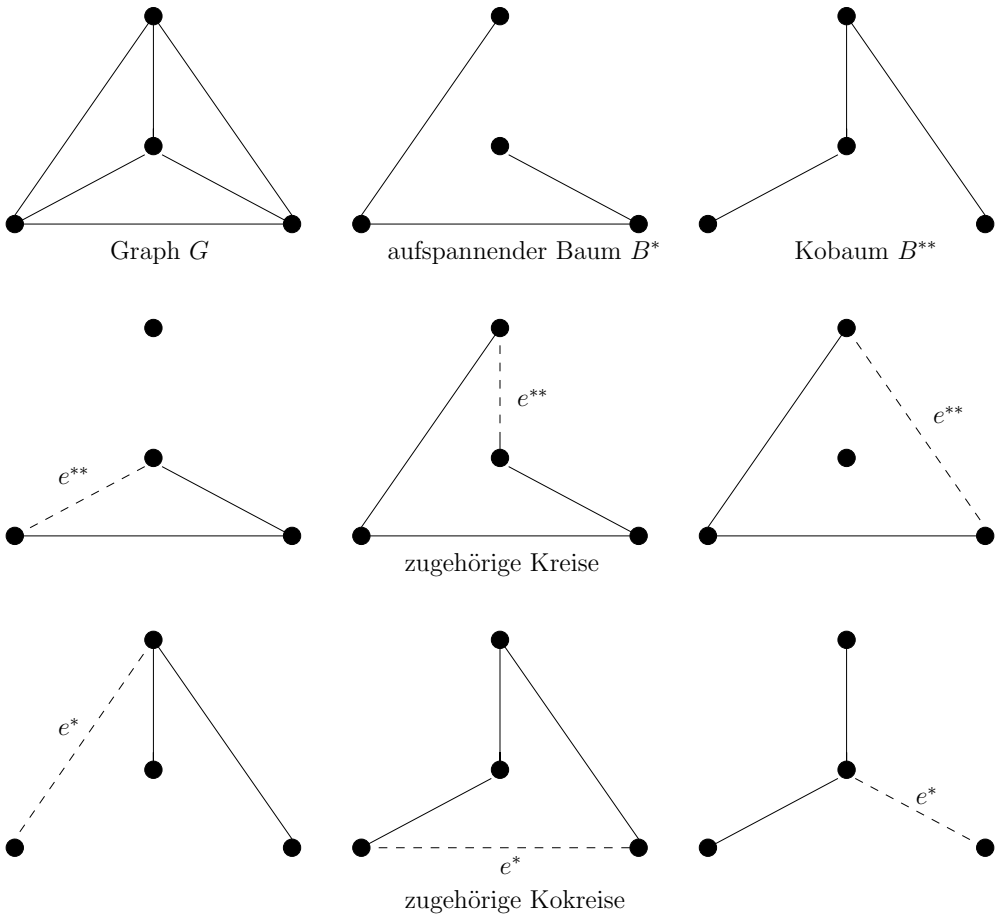


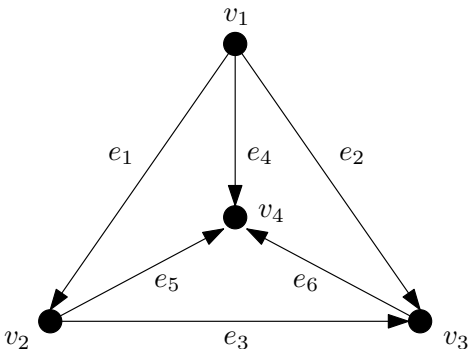
Abb. 2.1.: Beispiel zu Satz 2.3.1.

Häufig wird man mit dem Problem konfrontiert einen Graphen geeignet zu beschreiben. Eine Möglichkeit besteht darin, den Graphen durch eine *Knoten-Kanten-Inzidenzmatrix* (kurz Inzidenzmatrix) darzustellen. Wir wollen uns die Inzidenzmatrix im gerichteten Fall näher anschauen. Der ungerichtete Fall verläuft analog, indem lediglich das negative Vorzeichen entfernt wird. Dabei setzen wir eine geeignete Nummerierung der Knoten und Kanten voraus.

Definition 2.3.5. *Sei $G = (V, E)$ ein nummerierter gerichteter Graph, dann definieren wir seine Inzidenzmatrix durch*

$$A := (\pi_{ij})_{\substack{1 \le i \le |V| \\ 1 \le j \le |E|}} \quad \textit{mit } \pi_{ij} := \begin{cases} 1, & \textit{falls } e_j \in E \textit{ in } v_i \textit{ beginnt}, \\ -1, & \textit{falls } e_j \in E \textit{ in } v_i \textit{ endet}, \\ 0, & \textit{sonst}. \end{cases}$$

Beispiel 2.3.2. Abbildung 2.2 zeigt den Graphen aus Beispiel 2.3.1 mit einer Orientierung. Die entsprechende Inzidenzmatrix ist A.



$$A = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 & -1 \end{pmatrix}$$

Abb. 2.2.: Gerichteter Graph mit Inzidenzmatrix.

2.3.2. Netzwerke

Hier betrachten wir Flüsse in Netzwerken und schlagen dabei eine Brücke zwischen der linearen Algebra und dem Modell der Graphen. Eine Instanz ist gegeben durch einen gerichteten Graphen $D = (V, A)$ mit Kapazitätsfunktion $k : A \to \mathbb{Q}_+$, Quelle $s \in V$ und Senke $t \in V \setminus s$. Das Quadrupel (D, k, s, t) wird als *Netzwerk* bezeichnet. Indem wir eine Kante von t nach s hinzufügen erhalten wir ein erweitertes Netzwerk. Gilt für dieses erweiterte Netzwerk und einer Flussfunktion $f : E \to \mathbb{R}$ auf den Kanten, dass der aus einem Knoten herausfließende Fluss gleich dem hineinfließenden Fluss entspricht, also $f^+(v) = f^-(v)\ \forall v \in V$, dann sprechen wir von einer *Zirkulation*. Hier wird uns die Kapazitätsfunktion noch nicht interessieren, später aber, z.B. in Abschnitt 2.3.4, spielt sie eine zentrale Rolle bei der Ermittlung eines zulässigen Flusses. Zudem hat die Kapazitätsfunktion maßgeblichen Einfluss auf die Ganzzahligkeit eines Flusses.

Immer wenn man ein zusammenhängendes System hat, bei dem jeder Teil von anderen Teilen abhängt, kann man das System durch eine Matrix beschreiben. Interagierende Systeme lassen sich mit den Methoden der linearen Algebra beschreiben, falls die zugrunde liegenden Gesetze linearer Natur sind. Wie wir in Definition 2.3.5 gesehen haben, lassen sich Graphen anhand einer $(m \times n)$-Inzidenzmatrix A darstellen. Für jede Matrix über dem Körper $\mathbb{R}$ gibt es zwei Unterräume des $\mathbb{R}^n$ und zwei Unterräume des $\mathbb{R}^m$. Die Spalten von A sind Vektoren im $\mathbb{R}^m$, deren Linearkombinationen den *Spaltenraum* $S(A)$ erzeugen, ein Unterraum des $\mathbb{R}^m$. Die Zeilen von A sind Vektoren im $\mathbb{R}^n$. Deren Linearkombinationen erzeugen den *Zeilenraum* $S(A^T)$. Der *Kern* von A, kurz $\text{Kern}\,(A)$, enthält jeden Vektor x, für den $Ax = 0$ gilt. $\text{Kern}\,(A)$ ist ein Unterraum des $\mathbb{R}^n$. Der Kern der transponierten Matrix A^T enthält alle Lösungen der Gleichung $A^T y = 0$ bzw. $y^T A = 0^T$. $\text{Kern}\,(A^T)$ ist ein Unterraum von $\mathbb{R}^m$. Satz 2.3.2 liefert eine Aussage über die Dimension und Orthogonalität der Unterräume (siehe Strang [36] Seiten 181-203).

Satz 2.3.2. *Sei eine $(m \times n)$-Matrix A vom Rang r über $\mathbb{R}$ gegeben.*

(i) Der Spaltenraum $S(A)$ und der Zeilenraum $S(A^T)$ von A haben beide die Dimension r. $\text{Kern}\,(A)$ *hat die Dimension $n-r$ und* $\text{Kern}\,(A^T)$ *hat die Dimension*

$m - r$.

(ii) $\mathrm{Kern}\,(A)$ *und Zeilenraum* $S(A^T)$ *von* A *sind orthogonale Komplemente im* $\mathbb{R}^n$. $\mathrm{Kern}\,(A^T)$ *ist das orthogonale Komplement des Spaltenraums* $S(A)$ *im* $\mathbb{R}^m$.

Wir wollen uns nun ein Beispiel anschauen, welches Satz 2.3.2 illustriert (siehe Strang [36] Seiten 419-429). Es ist der Graph aus Beispiel 2.3.2 gegeben. Wir formen die entsprechende Inzidenzmatrix

$$A = \begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \left(\begin{array}{cccccc} 1 & 1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 & -1 \end{array} \right) \end{array}$$

wie folgt um. Zuerst addieren wir die erste Zeile zur zweiten Zeile und dann addieren wir die zweite Zeile zur dritten Zeile. In der dabei entstandenen Matrix unterscheiden sich die dritte und die vierte Zeile nur durch ihr Vorzeichen. Lassen wir die vierte Zeile weg und permutieren die Spalten, dann erhalten wir die Matrix

$$A' = \begin{array}{c} \begin{array}{cccccc} e_1 & e_3 & e_6 & e_2 & e_4 & e_5 \end{array} \\ \left(\begin{array}{cccccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{array} \right) \end{array}.$$

Die drei Spalten e_1, e_3 und e_6 sind linear unabhängig und bilden daher einen aufspannenden Baum (siehe Abschnitt 2.3.1) bzgl. G. Damit sind die Spalten e_2, e_4 und e_5 linear abhängig, woraus die Kreise $\{e_1, e_3, e_2\}$, $\{e_1, e_3, e_6, e_4\}$ und $\{e_3, e_6, e_5\}$ (vgl. Beispiel 2.3.1) folgen. In den Zeilen von A' stehen die Kokreise $\{e_1, e_2, e_4\}$, $\{e_3, e_2, e_4, e_5\}$ und $\{e_6, e_4, e_5\}$ (vgl. ebenfalls Beispiel 2.3.1). Wir wollen nun die vier Unterräume näher betrachten:

1. $\mathrm{Kern}\,(A)$ enthält die Lösungen der Gleichung $Ay = 0$. Wie aus A' ersichtlich ist, ist die Dimension $n - r = 6 - 3 = 3$. Diese Gleichungen stellen die Flusserhaltung dar. Der Fluss in einen Knoten hinein ist gleich dem Fluss hinaus. Man spricht auch von der *Kirchhoff'schen Knotenregel*.

2. Für den Zeilenraum $S(A^T)$ betrachten wir den Vektor $A^T x$, welcher aus Differenzen besteht. Seine Dimension ist $r = 3$. Die Unbekannten x_1, x_2, x_3 und x_4 stellen *Potentiale* an den Knoten dar und $A^T x$ liefert die *Potentialdifferenzen* auf den Kanten. Die Komponenten von $A^T x$ bilden entlang eines Kreises die Summe 0. Man spricht hier auch von der *Kirchhoff'schen Maschenregel*.

3. $\mathrm{Kern}\,(A^T)$ enthält die Lösungen der Gleichung $A^T x = 0$. Für derartige Vektoren sind alle sechs Potentialdifferenzen 0. Jedes x im $\mathrm{Kern}\,(A^T)$ ist ein konstanter Vektor (c, c, c, c). Der Kern von A^T ist eine Gerade im $\mathbb{R}^m$ mit Dimension $m - r = 1$. Wir können die Potentiale um denselben Betrag erhöhen oder erniedrigen, ohne die Potentialdifferenzen zu verändern.

4. Für den Spaltenraum $S(A)$ betrachten wir den Vektor Ay. Seine Dimension ist $r = 3$. Treten mehr als drei Kanten auf bilden sich Kreise. Ein Vektor liegt in $S(A)$ genau dann, wenn er orthogonal zum Vektor (c, c, c, c) vom Kern (A^T) ist.

Da das Zusammenwirken von Fluss und den Potentialdifferenzen wichtig zum Verständnis von Abschnitt 3.5 ist, wollen wir im Folgenden diesbezüglich explizit ein Beispiel ausführen. In einem Netzwerk ergibt sich der Strom y entlang einer Kante aus dem Produkt der Potentialdifferenzen und der Leitfähigkeit. Dies bezeichnet man auch als das *Ohm'sche Gesetz*. Dabei beschreibt die Leitfähigkeit wie leicht ein Fluss durch eine Kante fließt. Bezüglich elektronischer Schaltungen beschreibt die Leitfähigkeit den Kehrbruch des Widerstandes. Wir wollen im Folgenden eine Leitfähigkeitsmatrix $C = I$ verwenden. Kombiniert man die Potentialdifferenzen, die Leitfähigkeit und die Kirchhoff'sche Knotenregel, dann erhalten wir die Gleichung

$$ACA^T x = 0. \tag{2.5}$$

Da aber auf der rechten Seite von (2.5) ein Nullvektor steht, kommt kein Fluss zustande. Um einen Fluss im Netzwerk zu erhalten, legen wir deshalb eine Spannungsquelle zwischen v_1 und v_4 an. Dabei ist v_1 die Quelle und v_4 die Senke. Damit wir eine Zirkulation erhalten, fließt der Strom von v_4 innerhalb der Spannungsquelle wieder zurück nach v_1. Die Spannungsquelle stellt also eine Kante von v_4 nach v_1 dar. Damit erhalten wir die Gleichung

$$ACA^T x = f. \tag{2.6}$$

f beschreibt quasi den Fluss aus der Quelle v_1 heraus. Da $C = I$ ist, erhalten wir als Gleichung $AA^T x = f$, wobei

$$AA^T = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix} = P$$

gilt. Da P nicht invertierbar ist, kann man nicht nach den Potentialen auflösen. Dies liegt daran, dass der Vektor (c, c, c, c) im Kern (A^T) liegt. Wir „erden" deshalb den vierten Knoten v_4 mit $x_4 = 0$ und entfernen dadurch die vierte Spalte und vierte Zeile von P. Lösen wir nun $AA^T x = f$ nach den unbekannten Potentialen x_1, x_2 und x_3 unter Annahme eines Quellstromes von S im Knoten v_1, dann bekommen wir durch

$$\begin{pmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} S \\ 0 \\ 0 \end{pmatrix} \qquad \text{die Potentiale} \qquad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} S/2 \\ S/4 \\ S/4 \end{pmatrix}.$$

Über das Ohm'sche Gesetz $y = -CA^T x$ erhalten wir schließlich die Ströme bzw. den Fluss

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = - \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} S/2 \\ S/4 \\ S/4 \\ 0 \end{pmatrix} = \begin{pmatrix} S/4 \\ S/4 \\ 0 \\ S/2 \\ S/4 \\ S/4 \end{pmatrix}.$$

2.3.3. Kürzester Pfad

Eines der bekanntesten Probleme der kombinatorischen Optimierung ist vermutlich die Bestimmung eines kürzesten Pfades, bzgl. einer Gewichtsfunktion, zwischen zwei Knoten in einem gerichteten Graphen. Es kommt häufig als Teilproblem in schwierigeren Optimierungsproblemen vor. Zudem hat dieses Problem viele Anwendungen in der Praxis. Algorithmen, die einen kürzesten Pfad berechnen, werden z.B. für die Berechnung von Reiserouten eingesetzt. In einem gerichteten Graphen stellt sich das Problem wie folgt dar:

Problem 2.3.1 Kürzester Pfad

Instanz: Gerichteter Graph $D = (V, E)$, Gewichte $c : E \to \mathbb{R}$ und zwei Knoten $s, t \in V$.

Aufgabe: Finde einen kürzesten s-t-Pfad P, d.h. $c(P)$ ist minimal, oder entscheide, dass t von s aus nicht erreicht werden kann.

Es gibt einige Varianten von Problem 2.3.1. Neben der Suche eines kürzesten Pfades von s nach t, ist es auch möglich, kürzeste Pfade von s zu allen anderen Knoten $v \in V$ zu bestimmen. Auch existiert eine Variante, welche kürzeste Pfade zwischen allen Knotenpaaren $u, v \in V$ berechnet.

Wie Problem 2.3.1 zu entnehmen ist, ist es möglich negative Kosten auf den Kanten zu haben. Dies stellt kein grundlegendes Problem dar solange keine negativen Kreise auftreten. Treten negative Kreise (siehe Definition 2.3.1) auf, dann ist die Bestimmung eines kürzesten Pfades bzw. Kantenweges $\mathcal{NP}$-vollständig (siehe Hochstättler/Schliep [18] Seite 53 und Garey/Johnson [10]). Man unterscheidet drei Fälle:

1. Alle Gewichte sind nicht negativ.
2. Es treten keine negativen Kreise auf.
3. Es sind negative Kreise vorhanden.

Der bekannteste Algorithmus zum Berechnen eines kürzesten Pfades stammt von *Dijkstra*. Der Algorithmus setzt für eine korrekte Arbeitsweise voraus, dass alle Gewichte auf den Kanten nicht negativ sind. Bei einer einfachen Implementierung hat

er eine Laufzeit von $O(|V|^2)$. Durch die Benutzung eines Fibonacci-Heaps kann man eine Laufzeit von $O(|E| + |V| \log |V|)$ erreichen. Ein Algorithmus, der mit negativen Gewichten umgehen kann, solange keine negativen Kreise vorhanden sind, ist der Algorithmus von *Bellman-Ford*. Dieser hat eine Laufzeit von $O(|V||E|)$. Zudem ist es möglich den Algorithmus von Bellmann-Ford so zu modifizeren, dass er negative Kreise detektieren kann. Der Algorithmus von *Floyd-Warshall* berechnet die kürzesten Pfade zwischen allen Paaren von Knoten in $O(|V|^3)$. Wir wollen hier nicht weiter auf die Details dieser Algorithmen eingehen und verweisen stattdessen auf [1], [24] und [18].

Abschließend wollen wir anschauen, wie ein kürzester Pfad durch das Lösen eines linearen Programm bestimmt werden kann. Sei ein gerichteter Graph $D = (V, E)$ mit Kostenfunktion $c : E \to \mathbb{R}$ und entsprechender Inzidenzmatrix B (siehe Definition 2.3.5) gegeben. Weiter definieren wir die *charakteristische Funktion* χ über V auf folgende Art (siehe Hochstättler/Schliep [18] Seite 33).

Definition 2.3.6. *Sei V eine Menge und $T \subseteq V$. Wir definieren die charakteristische Funktion $\chi_T : V \to \{0, 1\}$ durch*

$$\chi_T(v) = \begin{cases} 1, & \text{falls } v \in T \\ 0, & \text{falls } v \notin T \end{cases}$$

Falls $|V|$ endlich ist, nennen wir χ_T einen Inzidenzvektor.

Für die Bestimmung eines kürzesten Pfades von s nach t erhält man schließlich folgendes lineares Programm:

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{u.d.N.} & (-B)x & = & \big(\chi(t) - \chi(s)\big) \\ & x & \geq & 0. \end{array} \tag{2.7}$$

Falls ein s-t-Pfad im gegebenen Graphen existiert, dann hat das Programm eine zulässige Lösung. Das Programm ist unbeschränkt, wenn Kreise mit negativen Kosten vorhanden sind. Andernfalls hat das Programm eine Optimallösung $x \in \{0, 1\}^{|E|}$. Die Menge $\{e \in E \mid x_e = 1\}$ beschreibt den kürzesten s-t-Pfad. Der Zielfunktionswert stellt die Länge des Pfades dar. Die Dualisierung des linearen Programmes (2.7) ist in (2.8) dargestellt.

$$\begin{array}{lll} \max & d^T\big(\chi(t) - \chi(s)\big) & \\ \text{u.d.N.} & d^T(-B) & \leq \quad c \end{array} \tag{2.8}$$

Die Lösung $d \in \mathbb{Z}^{|V|}$ des dualen Programmes liefert die Potentiale (vgl. Abschnitt 2.3.2) der Knoten von s aus. Es ist leicht zu sehen, dass für jede Lösung d auch $d + \delta$ mit $\delta \in \mathbb{R}$ eine zulässige Lösung ist. Man setzt in aller Regel den Wert von δ so, dass $d_s = 0$ gilt.

2.3.4. Maximaler Fluss

Ein Problem, welches in vielen praktischen Fragestellungen auftritt, ist die Bestimmung eines maximalen Flusses. Eine Instanz ist gegeben durch das Quadrupel

(D, k, s, t) (siehe Abschnitt 2.3.2). Die Aufgabe besteht darin, einen maximalen Fluss durch das Netzwerk von s nach t zu senden und dabei die Kapazitätsbeschränkungen einzuhalten. Ein zulässiger Fluss hat dabei die Eigenschaft der Flusserhaltung zu erfüllen, d.h. für jeden Knoten $v \in V \setminus \{s, t\}$ gilt, dass der Gesamtfluss der in v hineinfließt dem Fluss entspricht, welcher aus v herausfließt. Der *Nettofluss* ist dabei der Fluss der s verlässt minus dem Fluss, der in s eintritt. Aufgrund der Flusserhaltungseigenschaft entspricht der Nettofluss dem Fluss, der in t eintritt minus dem, der aus t austritt.

Problem 2.3.2 Maximaler Fluss

Instanz: Ein Netzwerk (D, k, s, t).
Aufgabe: Bestimmung eines maximalen Flusses von s nach t.

Ein erster kombinatorischer Algorithmus zur Bestimmung eines maximalen Flusses in einem Netzwerk wurde in [8] veröffentlicht. Dieser Algorithmus wird nach seinen Erfindern auch *Ford-Fulkerson* Algorithmus genannt. Die Terminierung des Algorithmus ist dann gesichert, wenn das Netzwerk rationale Kapazitäten besitzt. Wesentlich für die Arbeitsweise des Algorithmus ist die Bestimmung eines augmentierenden Pfades im *Residualnetzwerk*. Wenn kein augmentierender Pfad mehr existiert, ist der Fluss maximal. Das Residualnetzwerk ist wie folgt definiert (siehe Schrijver [32] Seiten 148-150).

Definition 2.3.7. *Sei ein gerichteter Graph $D = (V, E)$ mit Kapazitätsfunktion $k : E \to \mathbb{Q}_+$ und s-t-Fluss $f : E \to \mathbb{Q}_+$ gegeben. Für eine Kante $e \in E$ bezeichnen wir mit e^{-1} die Kante bei dem die Orientierung umgekehrt wurde und setzen $E^{-1} := \{e^{-1} \mid e \in E\}$. Das Residualnetzwerk $D_f = (V, E_f)$ von f ist durch*

$$\begin{aligned} E_1 &:= \{e \mid e \in E, f(e) < k(e)\} \\ E_2 &:= \{e^{-1} \mid e \in E, f(e) > 0\} \\ E_f &:= E_1 \cup E_2 \end{aligned}$$

mit Kapazitätsfunktion $k^f : E_f \to \mathbb{Q}_+$

$$k^f(e) = \begin{cases} k(e) - f(e), & \forall e \in E_1 \\ f(-e), & \forall e \in E_2 \end{cases}$$

definiert.

Die Wahl des augmentierenden Pfades ist entscheident für die Laufzeit. Anstatt einen beliebigen, augmentierenden Pfad zu wählen, sollte man einen kürzesten verwenden, d.h. einen augmentierenden Pfad mit einer minimalen Anzahl von Kanten. Diese einfache Idee liegt dem *Edmonds-Karp* Algorithmus zugrunde. Er stellt den ersten polynomialen Algorithmus zur Bestimmung eines maximalen Flusses dar. Seine Laufzeit beträgt $O(|V| \cdot |E|^2)$. Wir gehen nicht näher auf die beiden Algorithmen ein und verweisen auf [1], [24] und [18].

Solange die Kapazitäten ganzzahlig sind, gibt es ein einfaches Kriterium, dass die erwähnten Algorithmen terminieren. In jedem Schritt wird der Fluss um mindestens 1 erhöht und zudem stellen Schnitte (siehe Definition 2.3.3) natürliche, obere Schranken eines Flusses dar. Dabei ist die Kapazität eines s-t-Schnittes gleich der Summe der Kapazitäten seiner Kanten. Ein minimaler s-t-Schnitt ist ein s-t-Schnitt mit minimaler Kapazität. Der folgende Satz ist eines der bekanntesten Resultate der kombinatorischen Optimierung und zeigt die Verbindung zwischen Flüssen und Schnitten auf.

Satz 2.3.3. *In einem Netzwerk ist der maximale s-t-Fluss gleich der minimalen Kapazität eines s-t-Schnittes (siehe Korte/Vygen [24] Seite 187).*

Eine weitere wichtige Eigenschaft, die aus ganzzahligen Kapazitäten folgt, ist im folgenden Korollar festgehalten.

Korollar 2.3.1. *Sind die Kapazitäten eines Netzwerks ganzzahlig, so gibt es einen ganzzahligen maximalen s-t-Fluss (siehe Korte/Vygen [24] Seite 187).*

Wir geben nun ein lineares Programm an, welches einen maximalen s-t-Fluss bestimmt. Wir verwenden folgende Notation. Sei $W \subseteq V$, dann bezeichnen wir mit $\delta^-(W)$ die Menge der Kanten, die in W eintreten und entsprechend mit $\delta^+(W)$ die Menge der Kanten, die von W herausführen. Für $\delta^+(\{v\})$ schreiben wir kurz $\delta^+(v)$. Unter Verwendung dieser Notation erhalten wir das folgende lineare Programm (siehe Grötschel et al. [14] Seite 197).

$$\begin{array}{lrl} \max & x(\delta^+(s)) - x(\delta^-(s)) & \\ \text{u.d.N.} & x(\delta^-(v)) - x(\delta^+(v)) = 0 & \forall v \in V \setminus \{s,t\} \\ & 0 \le x_e \le c_e & \forall e \in E \end{array} \tag{2.9}$$

In $x \in \mathbb{R}^E$ stehen die Flüsse auf $e \in E$. $x(\delta^+(s)) - x(\delta^-(s))$ stellt den Wert des s-t-Flusses dar. Aus (2.9) ist ersichtlich, dass der Nullfluss immer zulässig ist. Abschließend wollen wir das duale lineare Programm zu (2.9) erwähnen (siehe Grötschel et al. [14] Seite 198). Dabei führen wir Dualvariablen y_e für alle $e \in E$ und z_v für alle $v \in V$ ein.

$$\begin{array}{lrcll} \min & \sum_{e \in E} c_e y_e & & & \\ \text{u.d.N.} & z_w - z_v + y_e & \ge & 0 & \forall e = (v,w) \in E \\ & z_s & = & 1 & \\ & z_t & = & 0 & \\ & y_e & \ge & 0 & \forall e \in E \end{array} \tag{2.10}$$

Als Ergebnis liefert das duale Programm den minimalen s-t-Schnitt mit Wert $\sum_{e \in E} c_e y_e$. Die beteiligten Kanten werden durch die Werte der Dualvariablen $y \in \{0,1\}^{|E|}$ spezifiziert.

2.3.5. Min-Cost-Flow

Einfache Kostenfunktion

Häufig tritt das Problem auf, einen Fluss mit einem vorgegebenen Wert durch ein Netzwerk zu senden und dabei minimale Kosten zu verursachen. Vielen Transport- oder Distributionsaufgaben liegt ein solches Problem zugrunde. Man nennt dieses Problem auch das *Min-Cost-Flow Problem*.

Sei ein gerichteter Graph $D = (V, E)$ mit Kostenfunktion $c : E \to \mathbb{Q}$ gegeben. Die Kosten, die ein Fluss $f : E \to \mathbb{Q}$ verursacht, sind $\sum_{e \in E} c(e)f(e)$. Man unterscheidet zwei Varianten des Min-Cost-Flow Problems (siehe Schrijver [32] Seite 177). Beim *Min-Cost-s-t-Flow Problem* ist ein gerichteter Graph $D = (V, E)$, zwei ausgezeichnete Knoten $s, t \in V$, eine Kapazitätsfunktion $k : E \to \mathbb{Q}_+$, eine Kostenfunktion $c : E \to \mathbb{Q}$ und ein Wert $\phi \in \mathbb{Q}_+$ gegeben. Ziel ist es einen s-t Fluss $f \leq k$ mit Wert ϕ zu bestimmen, welcher minimale Kosten verursacht. Eng verwandt ist das *Min-Cost-Circulation Problem*. Hier ist ein gerichteter Graph $D = (V, E)$, eine Bedarfsfunktion $d : E \to \mathbb{Q}$, eine Kapazitätsfunktion $k : E \to \mathbb{Q}_+$ und eine Kostenfunktion $c : E \to \mathbb{Q}$ gegeben. Ziel ist es hier eine Zirkulation f mit $d \leq f \leq k$ zu finden, die minimale Kosten hat. Man kann einfach das Min-Cost-s-t-Flow Problem in das Min-Cost-Circulation Problem umwandeln, indem man eine zusätzliche Kante e_0 von t nach s mit $d(e_0) := k(e_0) = \phi$ und $c(e_0) := 0$ einfügt. Zudem sei $d(e) := 0 \ \forall e \neq e_0$. Eine Zirkulation mit minimalen Kosten im, um e_0, erweiterten Graphen, liefert einen Fluss mit Wert ϕ mit minimalen Kosten in D. Wenn wir später Flüsse mit minimalen Kosten in regulären Matroiden betrachten, werden wir ausschließlich mit Zirkulationen arbeiten. Wir wollen noch kurz das Min-Cost-Flow Problem explizit herausstellen. Dabei verwenden wir allerdings eine Bedarfsfunktion $b : V \to \mathbb{Q}$ auf den Knoten mit $\sum_{v \in V} b(v) = 0$ (siehe Hochstättler/Schliep [18] Seite 89).

Problem 2.3.3 Min-Cost-Flow

Instanz: Gerichteter Graph $D = (V, A)$ mit Kapazitätsfunktion $k : E \to \mathbb{Q}_+$, Kostenfunktion $c : E \to \mathbb{Q}$ und Bedarfsfunktion $b : V \to \mathbb{Q}$.
Aufgabe: Bestimmung eines Flusses $f \leq k$ mit minimalen Kosten der b erfüllt.

Wie bei der Bestimmung eines maximalen Flusses, benötigen wir das Residualnetzwerk (siehe Definition 2.3.7). Dabei erweitern wir die Kostenfunktion c um die Kanten aus E_2 und definieren

$$c^f(e) := \begin{cases} c(e), \text{für } e \in E_1 \\ -c(e), \text{für } e^{-1} \in E_2. \end{cases}$$

Zudem verwenden wir die folgende Notation. Sei C ein gerichteter Kreis in D_f, dann definieren wir $\chi^C \in \mathbb{R}^{|E|}$ mit

$$\chi^C(e) := \begin{cases} 1, & \text{falls } C \text{ Kante } e \text{ traversiert,} \\ -1, & \text{falls } C \text{ Kante } e^{-1} \text{ traversiert,} \\ 0, & \text{sonst.} \end{cases}$$

Damit sind wir nun in der Lage ein Optimalitätskriterium für das Min-Cost-Flow Problem anzugeben (siehe Klein [23]).

Satz 2.3.4. *Sei $D = (V, E)$ ein gerichteter Graph, eine Bedarfsfunktion $d : E \to \mathbb{Q}$, eine Kapazitätsfunktion $k : E \to \mathbb{Q}_+$, eine Kostenfunktion $c : E \to \mathbb{Q}$ und $f : E \to \mathbb{Q}$ eine zulässige Zirkulation. f hat unter allen zulässigen Zirkulationen minimale Kosten genau dann, wenn jeder gerichtete Kreis in D_f nichtnegative Kosten hat (Beweis Schrijver [32] Seiten 178-179).*

Satz 2.3.4 liefert einen Algorithmus um eine Zirkulation mit minimalen Kosten zu bestimmen. Wähle einen gerichteten Kreis C mit negativen Kosten in D_f und passe den Fluss entsprechend an (siehe Klein [23]). Falls kein solcher gerichteter Kreis mehr existiert, ist f eine Zirkulation mit minimalen Kosten. Für rationale Daten terminiert dieser Algorithmus. Wählt man allerdings die negativen Kreise beliebig, dann kann dieser Algorithmus exponentielle Laufzeit haben. Goldberg und Tarjan [11] ist es gelungen zu beweisen, dass man einen streng polynomialen Algorithmus für das Min-Cost-Flow Problem erhält, wenn man statt beliebiger negativer Kreise, negative Kreise mit minimalen durchschnittlichen Kosten $\frac{c^f(C)}{|C|}$ (siehe auch Karp [20]) wählt. Man nennt diesen Ansatz auch *Min-Mean-Cycle-Canceling*.

Neben Satz 2.3.4 gibt es weitere Optimalitätskriterien. Zudem existieren weitere Ansätze für Algorithmen um einen Fluss mit minimalen Kosten zu bestimmen. Darauf möchten wir aber nicht weiter eingehen und verweisen auf [1], [24] und [18].

Abschließend wollen wir uns noch anschauen, wie man das Min-Cost-Flow Problem mit einem linearen Programm lösen kann (siehe Hochstättler/Schliep [18] Seite 89).

$$\begin{array}{lrcl} \min & c^T f & & \\ \text{u.d.N.} & Bf & = & b \\ & f & \leq & k \\ & 0 & \leq & f \end{array} \tag{2.11}$$

Dabei stellt B die Inzidenzmatrix, $b : V \to \mathbb{Q}$ die Bedarfsfunktion, $c : A \to \mathbb{Q}$ die Kostenfunktion und f den Fluss auf den Kanten dar. Eine Dualisierung von (2.11) liefert (2.12).

$$\begin{array}{lrcl} \max & -y^T k - \pi^T b & & \\ \text{u.d.N.} & -y^T - \pi^T B & \leq & c^T \\ & y & \geq & 0 \end{array} \tag{2.12}$$

π stellt die Potentiale (siehe Abschnitt 2.3.2) dar.

Konvexe separable Kostenfunktion

Wir haben im vorhergehenden Abschnitt gezeigt, wie die Kanten eines Graphen mit einer einfachen Kostenfunktion versehen werden können. Wir wollen hier eine naheliegende Erweiterung der Kostenfunktion vornehmen.

Zuerst definieren wir, was wir unter einer konvexen Menge (siehe Reemtsen [29] Seite 44) und einer konvexen Funktion verstehen (siehe Reemtsen [29] Seite 45).

Definition 2.3.8. *Eine Menge $K \subseteq \mathbb{R}^n$ heißt konvex, wenn gilt:*

$$x, y \in K, t \in [0,1] \Rightarrow tx + (1-t)y \in K.$$

Definition 2.3.9. *Sei $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ eine Funktion und $K \subseteq D$ eine konvexe Menge. f heißt konvex auf K, falls gilt:*

$$x, y \in K, t \in [0,1] \Rightarrow f(tx + (1-t)y) \leq tf(x) + (1-t)f(y). \tag{2.13}$$

Ist eine Funktion als Summe von Funktionen f_i darstellbar, welche jeweils von Variablen abhängen, die nur in ihr und in keiner weiteren f_j mit $j \neq i$ vorkommen, dann nennen wir eine solche Funktion *separabel* (siehe [16], [9]).

Definition 2.3.10. *Seien $f_i : \mathbb{R} \to \mathbb{R}$ mit $i = 1, \ldots, n$ Funktionen, dann nennen wir die Funktion*

$$F(x) := \sum_{i=1}^{n} f_i(x_i) \quad x \in \mathbb{R}^n \tag{2.14}$$

separabel.

Setzt sich eine Funktion F aus Teilfunktionen zusammen, welche konvex sind, dann ist auch die separable Funktion konvex, wie das folgende Lemma zeigt.

Lemma 2.3.1. *Für $i = 1, \ldots, r$ seien $a_i > 0$ und seien $f_i : D \subseteq \mathbb{R}^n \to \mathbb{R}$ konvexe Funktionen auf einer konvexen Menge $K \subseteq D$. Dann ist auch die Funktion*

$$F(x) := \sum_{i=1}^{r} a_i f_i(x), \quad x \in D, \tag{2.15}$$

auf K konvex (Beweis Reemtsen [29] Seite 46).

Mit Lemma 2.3.1 folgt unmittelbar, dass für konvexe Funktionen f_i auch die Funktion aus Definition 2.3.10 konvex ist.

Wir unterscheiden zwei verschiedene Modelle einer konvexen separablen Kostenfunktion (siehe Ahuja et al. [1] Seite 544). Dabei bezeichne C_{ij} die konvexe Kostenfunktion auf dem Element $(i,j) \in E$ eines gerichteten Graphen $D = (V, E)$.

1. *Stückweises lineares Modell*: Jede Kostenfunktion $C_{ij}(x_{ij})$ hat höchstens p lineare Segmente. Dabei bezeichnen $0 = d_{ij}^0 < d_{ij}^1 < \ldots$ die *Knickstellen* (engl. Breakpoints) der Funktion. Die Kosten variieren linear im Intervall $[d_{ij}^{k-1}, d_{ij}^k]$ und c_{ij}^k stellt den entsprechenden linearen Kostenkoeffizienten dar.

2. *Kurzgefasstes Funktionsmodell*: $C_{ij}(x_{ij})$ ist in funktionaler Form, wie z.B. x_{ij}^4, angegeben.

Im Verlauf der Arbeit beschränken wir uns auf den 1. Fall.

Somit können wir das lineare Programm (2.11) mit einer konvexen separablen Kostenfunktion in der folgenden Form schreiben. Dabei stellt $C_{ij}(x_{ij})$ die stückweise

lineare konvexe Kostenfunktion und k_{ij} die Kapazität von Kante $(i,j) \in E$ dar.

$$\begin{array}{lll} \min & \sum_{(i,j)\in E} C_{ij}(x_{ij}) & \\ \text{u.d.N.} & \sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = b(i) & \forall i \in V \\ & 0 \le x_{ij} \le k_{ij} & \forall (i,j) \in E \end{array} \tag{2.16}$$

(2.16) können wir so transformieren, dass wir ein Min-Cost-Flow Problem mit einer Kostenfunktion wie in (2.11) erhalten (siehe Ahuja et al. [1] Seite 552). Wir gehen der Einfachkeit halber davon aus, dass jede stückweise lineare konvexe Funktion p lineare Segmente hat. Sei x_{ij} der Fluss auf Kante $(i,j) \in E$. Wir teilen dabei den Fluss x_{ij} durch

$$y_{ij}^k = \begin{cases} 0, & \text{falls } x_{ij} \le d_{ij}^{k-1} \\ x_{ij} - d_{ij}^{k-1}, & \text{falls } d_{ij}^{k-1} \le x_{ij} \le d_{ij}^k \\ d_{ij}^k - d_{ij}^{k-1}, & \text{falls } x_{ij} \ge d_{ij}^k \end{cases}$$

auf die k Segmente auf. Es gilt $x_{ij} = \sum_{k=1}^p y_{ij}^k$ und $C_{ij}(x_{ij}) = \sum_{k=1}^p c_{ij}^k y_{ij}^k$. Indem wir $\sum_{k=1}^p y_{ij}^k$ für x_{ij} in (2.16) einsetzen, erhalten wir das folgende lineare Programm:

$$\begin{array}{lll} \min & \sum_{(i,j)\in E} \sum_{k=1}^p c_{ij}^k y_{ij}^k & \\ \text{u.d.N.} & \sum_{\{j|(i,j)\in E\}} \sum_{k=1}^p y_{ij}^k - \sum_{\{j|(j,i)\in E\}} \sum_{k=1}^p y_{ji}^k = b(i) & \forall i \in V \\ & 0 \le y_{ij}^k \le d_{ij}^k - d_{ij}^{k-1} & \forall (i,j) \in E, k = 1,\ldots,p \end{array} \tag{2.17}$$

(2.17) stellt das Min-Cost-Flow Problem auf dem expandierten Graphen $G' = (V, E')$ mit p parallelen Kanten für jede Kante $(i,j) \in E$ dar. In [1] wird auf Seite 553 die Äquivalenz von (2.16) und (2.17) gezeigt. Ein Nachteil der Transformation besteht darin, dass meist $|E| << |E|'$ gilt.

Wir wollen uns jetzt anschauen, welche Eigenschaften (2.16) haben muss, damit die Optimallösung ganzzahlig ist. Zunächst definieren wir, was wir unter einem Polyeder verstehen (siehe Hochstättler [17] Seite 77).

Definition 2.3.11. *Eine Menge $P \subset \mathbb{K}^n$ heißt Polyeder, wenn es ein $m \in \mathbb{N}$, eine $(m \times n)$-Matrix A über $\mathbb{K}$ und ein $b \in \mathbb{K}^m$ gibt mit $P := \{x \in \mathbb{K}^n \mid Ax \le b\}$.*

Ist die Nebenbedingungsmatrix total unimodular (siehe Definition 2.2.2) und die rechte Seite ganzzahlig, dann hat das Polyeder eine bestimmte Gestalt. Dabei ist eine *Ecke* eine nulldimensionale Seitenfläche des Polyeders.

Satz 2.3.5. *Sei A eine total unimodulare Matrix und b ein ganzzahliger Vektor, dann hat das Polyeder $P := \{x \mid Ax \le b\}$ ganzzahlige Ecken (Beweis Schrijver [31] Seite 266).*

Satz 2.3.5 versetzt uns in die Lage eine Aussage darüber zu treffen, wann (2.16) eine ganzzahlige Optimallösung besitzt.

Korollar 2.3.2. *Sei ein Min-Cost-Flow Problem anhand* (2.16) *gegeben. Die Optimallösung ist dann ganzzahlig, wenn die Nebenbedingungsmatrix total unimodular, die rechte Seite b und die Kapazitätsfunktion ganzzahlig sind und wenn die Breakpoints der Kostenfunktion ganzzahlig sind.*

Beweis: Durch die ganzzahlige Kapazitätsfunktion und die ganzzahligen Breakpoints der Kostenfunktion, lässt sich (2.16) in (2.17) transformieren. Die dabei entstehende Nebenbedingunsmatrix ist weiterhin total unimodular, weil dem Graphen lediglich parallele Kanten hinzugefügt wurden. Da b nach Voraussetzung ganzzahlig ist, folgt mit Satz 2.3.5 die Behauptung. □

2.4. Matroide

In diesem Abschnitt sollen zunächst grundlegende Definitionen und Axiomatisierungen von Matroiden aufgezeigt werden. Danach gehen wir auf die Dualität und Minoren von Matroiden ein. Anschließend widmen wir uns den regulären Matroiden. Neben der Charakterisierung von regulären Matroiden sind wir an deren Signierbarkeit, Flusseigenschaften und Komposition/Dekomposition interessiert. Abschließend gehen wir auf das Orakelkonzept von Matroiden ein und sprechen zur Abrundung noch kurz den Greedy-Algorithmus an.

2.4.1. Grundlegende Definitionen und Axiome

Bei einem Matroid handelt es sich um eine mathematische Struktur, mit deren Hilfe der Begriff der Unabhängigkeit aus der linearen Algebra verallgemeinert wird. Grundlegend für die Matroidtheorie war eine Veröffentlichung im Jahr 1935 von Hassler Whitney [44]. Matroide besitzen eine Vielzahl von Anwendungen in der kombinatorischen Optimierung und der Graphentheorie.

Definition 2.4.1. *Ein Matroid M ist ein geordnetes Paar $(E,\mathcal{I})$, bestehend aus einer endlichen Menge E und einer Mengenfamilie $\mathcal{I} \subseteq 2^E$ mit*

(I1) $\varnothing \in \mathcal{I}$,

(I2) $I \in \mathcal{I}$ *und* $I' \subseteq I \Rightarrow I' \in \mathcal{I}$,

(I3) $I_1, I_2 \in \mathcal{I}$ *und* $|I_1| < |I_2| \Rightarrow \exists e \in I_2 \setminus I_1 : I_1 \cup e \in \mathcal{I}$.

Bedingung (I3) wird auch *Basisergänzungsaxiom* genannt. Die Mengen in $\mathcal{I}(M)$ bzw. $\mathcal{I}$ sind die unabhängigen Mengen von M und E bzw. $E(M)$ ist die Grundmenge von M. Die Mengen in $2^E \setminus \mathcal{I}$ heißen abhängig.

Der Name *Matroid* stammt daher, dass man endlich viele Vektoren als Spalten in einer Matrix zusammenfassen kann. Dies führt direkt zu folgendem Satz.

Satz 2.4.1. *Sei E die Menge der Spaltenlabel einer $(m \times n)$-Matrix A über einem Körper $\mathbb{K}$, und sei $\mathcal{I}$ die Menge der Teilmengen I von E, bei denen die Multimenge der mit Elementen aus I gekennzeichneten Spalten linear unabhängig in dem Vektorraum $\mathbb{K}^m$ ist. Dann ist $(E, \mathcal{I})$ ein Matroid (Beweis Oxley [28] Seite 8).*

Das Matroid, welches wir aus der Matrix A erhalten, bezeichnen wir mit $M[A]$ und nennen es *Vektormatroid* von A.

Beispiel 2.4.1. Sei $A \in \mathbb{R}^{2\times 4}$ die folgende Matrix:

$$\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left(\begin{array}{ccccc} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{array} \right) \end{array}$$

Dann ist $E = \{1, 2, 3, 4, 5\}$ und

$$\mathcal{I} = \{\varnothing, \{1\}, \{2\}, \{3\}, \{5\}, \{1, 2\}, \{1, 3\}, \{1, 5\}, \{2, 5\}, \{3, 5\}\}.$$

Die Familie der abhängigen Mengen dieses Matroids ist

$$\{\{4\}, \{1, 4\}, \{2, 4\}, \{3, 4\}, \{4, 5\}, \{2, 3\}\} \cup \{X \subseteq E \mid |X| \geq 3\}.$$

Also sind

$$\{\{4\}, \{2, 3\}, \{1, 2, 5\}, \{1, 3, 5\}\}$$

die minimal abhängigen Mengen, sind die abhängigen Mengen, deren Teilmengen alle unabhängig sind.

Definition 2.4.2. *Ein Kreis (engl. circuit) in einem Matroid $M = (E, \mathcal{I})$ ist eine minimal abhängige Teilmenge $C \subseteq E$. Die Menge aller Kreise in M bezeichnen wir mit $\mathcal{C}$. Ist C ein Kreis mit $|C| = n$, so nennen wir n die Länge von C (siehe Oxley [28] Seite 9).*

Wie aus Beispiel 2.4.1 ersichtlich ist, lässt sich aus der Familie der unabhängigen Mengen $\mathcal{I}$ die Familie der Kreise $\mathcal{C}$ eines Matroids bestimmen. Selbiges gilt auch für den umgekehrten Fall. Also ist ein Matroid durch die Familie der Kreise $\mathcal{C}$ eindeutig bestimmt.

Lemma 2.4.1. *Sei $\mathcal{C}$ die Familie der Kreise eines Matroids $M = (E, \mathcal{I})$. Dann gilt*

(C1) $\varnothing \notin \mathcal{C}$,

(C2) $C_1, C_2 \in \mathcal{C}$ *und* $C_1 \subseteq C_2 \Rightarrow C_1 = C_2$,

(C3) $\forall C_1 \neq C_2 \in \mathcal{C}\ \forall e \in C_1 \cap C_2\ \exists C_3 \in \mathcal{C} : C_3 \subseteq (C_1 \cup C_2) \setminus e$

(Beweis Oxley [28] Seite 9).

Bedingung (C3) wird auch als *(schwaches) Kreiseliminationsaxiom* bezeichnet. (C1), (C2) und (C3) charakterisieren die Familie der Kreise eines Matroids.

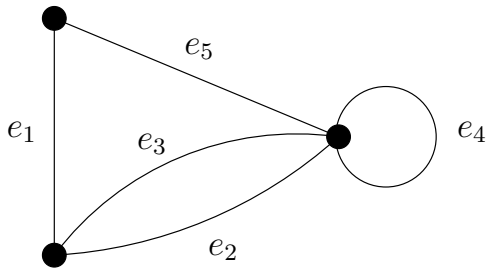


Abb. 2.3.: Kreismatroid $M(G)$.

Satz 2.4.2. *Sei E eine endliche Menge und $\mathcal{C} \subseteq 2^E$ eine Familie von Teilmengen von E, die (C1), (C2) und (C3) erfüllt. Sei $\mathcal{I} \subseteq 2^E$ die Familie derjenigen Teilmengen I von E, die kein Element C von $\mathcal{C}$ als Teilmenge enthalten. Dann ist $M = (E, \mathcal{I})$ ein Matroid und $\mathcal{C}$ die Menge der Kreise (Beweis Oxley [28] Seite 10).*

Aus Lemma 2.4.1 und Satz 2.4.2 erhalten wir folgendes Korollar.

Korollar 2.4.1. *Sei $\mathcal{C} \subseteq 2^E$. Dann ist $\mathcal{C}$ die Menge der Kreise eines Matroids auf E genau dann, wenn $\mathcal{C}$ die Bedingungen (C1), (C2) und (C3) erfüllt.*

Ein Kreis, welcher aus einer unabhängigen Mengen durch Hinzunahme eines Elementes entsteht, ist eindeutig (vgl. Satz 2.3.1).

Satz 2.4.3. *Sei $M = (E, \mathcal{I})$ ein Matroid, $I \in \mathcal{I}, e \in E$ und $I \cup e \notin \mathcal{I}$. Dann gibt es genau einen Kreis $C(I, e) \subseteq I \cup e$ und es gilt $e \in C(I, e)$ (Beweis Oxley [28] Seite 11).*

Die Betrachtung der Familie aller Kreise ist aus der Graphentheorie motiviert.

Satz 2.4.4. *Sei E die Menge der Kanten eines Graphen $G = (V, E)$ und $\mathcal{C}$ die Menge der Kantenmengen von Kreisen des Graphen. Dann ist $\mathcal{C}$ die Menge der Kreise eines Matroids (Beweis Oxley [28] Seite 11).*

Dieses Matroid nennen wir *Kreismatroid* oder *Polygonmatroid* $M(G)$ von G. Dabei sind die unabhängigen Mengen von $M(G)$ die Kantenmengen von G, welche Wälder sind.

Beispiel 2.4.2. Sei G der Graph in Abbildung 2.3 und $M(G)$ das entsprechende Kreismatroid. Dann gilt $E(M) = \{e_1, e_2, e_3, e_4, e_5\}$ und $\mathcal{C}(M) = \{\{e_4\}, \{e_2, e_3\}, \{e_1, e_2, e_5\}, \{e_1, e_3, e_5\}\}$.

Vergleichen wir das Vektormatroid $M[A]$ aus Beispiel 2.4.1 und das Kreismatroid $M(G)$ aus Beispiel 2.4.2, dann ist ersichtlich, dass die minimal abhängigen Mengen von $M[A]$ den Kreisen von $M(G)$ und die unabhängigen Mengen von $M[A]$ den unabhängigen Mengen von $M(G)$ entsprechen. Dies führt uns auf den Begriff der Isomorphie von Matroiden.

Definition 2.4.3. *Zwei Matroide $M_1 = (E_1, \mathcal{I}_1)$ und $M_2 = (E_2, \mathcal{I}_2)$ heißen isomorph, in Zeichen $M_1 \cong M_2$, wenn es eine Bijektion $\psi : E_1 \to E_2$ gibt mit*

$$I \in \mathcal{I}_1 \Leftrightarrow \psi(I) := \{\psi(e) \in E_2 \mid e \in I\} \in \mathcal{I}_2$$

Ein Matroid, das isomorph zu einem Kreismatroid ist, heißt *graphisch.*

Ist ein Matroid M isomorph zu einem Vektormatroid $M[A]$ einer Matrix A über einem Körper $\mathbb{K}$, dann ist M *repräsentierbar* über $\mathbb{K}$.

Definition 2.4.4. *Sei $M = (E, \mathcal{I})$ ein Matroid, dann sagen wir*

(i) M ist binär, wenn es über $\mathrm{GF}(2)$ repräsentierbar ist,

(ii) M ist ternär, wenn es über $\mathrm{GF}(3)$ repräsentierbar ist und

(iii) M ist regulär, wenn es über jedem Körper $\mathbb{K}$ repräsentierbar ist.

Dabei bezeichnet $\mathrm{GF}(2)$ den *Galoiskörper* mit zwei Elementen $\{0, 1\}$ und $\mathrm{GF}(3)$ den Körper mit drei Elementen $\{-1, 0, 1\}$. Für weitere Informationen zu den beiden Körpern verweisen wir auf Trümper [40] Seiten 18-24. Es sei noch erwähnt, dass nicht jedes Matroid über einem Körper repräsentierbar ist (siehe Oxley [28] Seiten 170-173).

Eine Liste von maximal unabhängigen Mengen in einem Matroid M anzugeben, liefert eine weitere Axiomatisierung über Basen.

Definition 2.4.5. *Sei $(E, \mathcal{I})$ ein Matroid, dann heißt $B \in \mathcal{I}$ Basis, wenn es kein $I \in \mathcal{I}$ gibt mit $B \subset I$. Die Familie der Basen bezeichnen wir mit $\mathcal{B}$.*

Die Basen eines Matroids haben viele Gemeinsamkeiten mit Basen in Vektorräumen wie folgendes Lemma zeigt.

Lemma 2.4.2. *Sind B_1 und B_2 Basen eines Matroids $M = (E, \mathcal{I})$, dann ist $|B_1| = |B_2|$ (Beweis Oxley [28] Seite 16).*

Die Axiomatisierung der Familie der Basen gestaltet sich wie folgt.

Lemma 2.4.3. *Sei $\mathcal{B}$ die Familie der Basen eines Matroids $M = (E, \mathcal{I})$, dann gilt*

(B1) $\mathcal{B} \neq \varnothing$,

(B2) $\forall B_1, B_2 \in \mathcal{B}\ \forall e \in B_1 \setminus B_2\ \exists f \in B_2 \setminus B_1 : (B_1 \setminus e) \cup f \in \mathcal{B}$

(Beweis Oxley [28] Seite 17).

Der folgende Satz zeigt, dass (B1) und (B2) die Basen eines Matroids eindeutig charakterisieren.

Satz 2.4.5. *Sei $\mathcal{B} \subseteq 2^E$ mit (B1) und (B2). Zudem sei $\mathcal{I} := \{I \in 2^E \mid \exists B \in \mathcal{B} : I \subseteq B\}$. Dann ist $(E, \mathcal{I})$ ein Matroid und $\mathcal{B}$ die Menge seiner Basen (Beweis Oxley [28] Seite 17).*

Sie M eine Matrix $(E,\mathcal{I})$ mit $X \subseteq E$ und $\mathcal{I}|X$ die Menge $\{I \subseteq X \mid I \in \mathcal{I}\}$, dann ist das Paar $(X,\mathcal{I}|X)$ ebenfalls ein Matroid, welches wir als *Restriktion* von M auf X bezeichnen. Man kann die Restriktion von M auf X mit $X \subseteq E$ auch als das Löschen der Elemente $E \setminus X$ von M interpretieren.

Mit den Begriffen Basis und Restriktion können wir die Rangfunktion eines Matroids definieren.

Definition 2.4.6. *Sei $M = (E,\mathcal{I})$ ein Matroid, $X \subseteq E$ und B_X eine Basis von X. Dann bezeichnen wir die Funktion* $\mathrm{Rang} : 2^E \to \mathbb{N}_0$ *mit*

$$\mathrm{Rang}(X) := |B_X| \text{ für ein } B_X \in \mathcal{B}(M_{|X})$$

als Rang von X.

Auch die Rangfunktion liefert ein Axiomensystem. Darauf möchten wir hier aber nicht weiter eingehen, denn im weiteren Verlauf der Arbeit finden ausschließlich Axiomatisierungen über unabhängige Mengen und Kreise Anwendung. Der interessierte Leser sei diesbezüglich auf Oxley [28] Seiten 22-28 verwiesen.

Wie wir in Definition 2.4.4 gesehen haben, können manche Matroide anhand eines Körpers repräsentiert werden. Eine kompakte Darstellung durch eine Matrix führt uns auf folgende Defintion.

Definition 2.4.7. *Sei $M = (E,\mathcal{I})$ ein Matroid mit n Elementen und* $\mathrm{Rang}(M) = k$. *Sei $A \in \mathbb{K}^{m\times n}$ eine Matrix mit $k \leq m$ Zeilen und n Spalten. Weiter sei eine Funktion $\psi : E \to \{1,\dots,n\}$, welche jedem Element $e \in E$ eine Spalte $\psi(e)$ der Matrix A zuordnet, gegeben. Falls für genau jede unabhängige Menge $I \in \mathcal{I}$ von M die entsprechenden Spalten $\{\psi(e) \mid e \in I\}$ von A linear unabhängig über dem Körper $\mathbb{K}$ sind, bezeichnen wir A als Repräsentationsmatrix von M.*

Im Allgemeinen existiert keine eindeutige Repräsentationsmatrix für ein Matroid, denn jede Repräsentationsmatrix kann durch elementare Zeilenumformungen in eine andere Matrix umgeformt werden, welche das gleiche Matroid repräsentiert.

Definition 2.4.8. *Sei $M = (E,\mathcal{I})$ ein Matroid und $A \in \mathbb{K}^{m\times n}$ eine Repräsentationsmatrix von M und $r = \mathrm{Rang}(M)$. Hat A die Form $(I_r|D)$ mit $D \in \mathbb{K}^{r\times n-r}$, dann nennen wir A Standardrepräsentationsmatrix von M.*

Wir werden im Folgenden den Begriff Repräsentationsmatrix und Standardrepräsentationsmatrix äquivalent verwenden.

Eine schöne Eigenschaft von graphischen Matroiden ist deren Darstellbarkeit als Graphen. Wir wollen zeigen, dass es für alle Matroide mit kleinem Rang eine geometrische Darstellungsmöglichkeit gibt, die vergleichbar nützlich ist. Durch Übergang von linearer zu affiner Abhängigkeit kann man in der geometrischen Darstellung eine Dimension einsparen.

Definition 2.4.9. *Seien $a_1,\dots,a_k \in \mathbb{K}^n$. Eine Linearkombination $a = \sum_{i=1}^k \lambda_i a_i$ mit $\lambda_i \in \mathbb{K}$ heißt affine Kombination von $a_1,\dots,a_k$, falls $\sum_{i=1}^k \lambda_i = 1$ gilt. $a_1,\dots,a_k$ sind*

affin unabhängig, wenn aus $\sum_{i=1}^{k} \lambda_i a_i = 0$ *und* $\sum_{i=1}^{k} \lambda_i = 0$ *notwendig folgt, dass* $\lambda_i = 0$ *für alle* $i = 1, \ldots, k$ *ist. Andernfalls heißen die Vektoren affin abhängig.*

Satz 2.4.6. *Sei* E *eine endliche Menge eines* $\mathbb{K}$*-Vektorraums und* $\mathcal{I}$ *die Menge der affin unabhängigen Teilmengen von* E*. Dann ist* $(E, \mathcal{I})$ *ein Matroid (Beweis Oxley [28] Seite 36).*

Wir nennen das Matroid aus Satz 2.4.6 auch *affines Matroid*. Falls M ein affines Matroid über $\mathbb{R}$ mit Rang $m+1$ und $m \leq 3$ ist, dann ist $X \subseteq E$ abhängig in M, falls X durch zwei identische Punkte, drei kolineare Punkte, vier koplanare Punkte oder fünf Punkte im Raum $\mathbb{R}^m$ dargestellt wird.

Beispiel 2.4.3. Das Matroid aus Beispiel 2.4.1 bzw. 2.4.2 hat die in Abbildung 2.4 aufgezeigte geometrische Darstellung. Der zweielementige Kreis wird dabei als zwei sich berührende Punkte und der dreielementige Kreis wird als Linie, durch die entsprechenden Punkte, dargestellt. Schleifen, welche in einem affinen Matroid nicht auftreten können, werden separat dargestellt und eingerahmt.

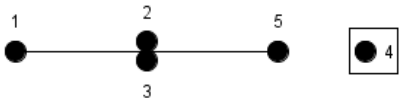


Abb. 2.4.: Geometrische Darstellung des Vektormatroids aus Beispiel 2.4.1.

Es soll hier erwähnt werden, dass die geometrische Darstellung nicht mit der Darstellung von Matroiden als Graphen verwechselt werden darf.

2.4.2. Dualität und Minoren

Wie in der linearen Optimierung (siehe Abschnitt 2.2.1) existiert auch in der Matroidtheorie eine Dualität. Auch hier stellt die Dualität ein sehr mächtiges Werkzeug dar. Wir werden zunächst zeigen, was wir unter dem dualen Matroid verstehen. Dann sprechen wir einige besondere Graphen an, von denen kein dualer Graph, aber ein induziertes duales Matroid existiert. Letztlich gehen wir auf Minoren und das Konzept der verbotenen Minoren, zur Klassifizierung von Matroiden, ein.

Satz 2.4.7. *Sei* $M = (E, \mathcal{I})$ *ein Matroid und* $\mathcal{B}(M)^* := \{E \setminus B \mid B \in \mathcal{B}(M)\}$*. Dann ist* $\mathcal{B}(M)^*$ *die Familie der Basen eines Matroids (Beweis Oxley [28] Seiten 68-69).*

Das Matroid aus Satz 2.4.7 heißt das *duale Matroid* von M und wird notiert als M^*. Die Basen von M^* heißen Kobasen von M und die Kreise von M^* heißen Kokreise von M. Zudem gilt offensichtlich $(M^*)^* = M$. Die Repräsentationsmatrix des dualen Matroid leitet sich direkt von Definition 2.4.8 ab.

Satz 2.4.8. *Ist M das Vektormatroid von $(I_r|D)$, dann ist M^* das Vektormatroid von $(-D^T|I_{n-r})$ (Beweis Oxley [28] Seiten 81-82).*

Wir wollen uns die Dualität beispielhaft am Matroid F_7 anschauen. Dabei handelt es sich um das *Fano Matroid*. F_7 ist die kleinste projektive Ebene und über dem Körper GF(2) repräsentierbar.

Beispiel 2.4.4. Links ist die Repräsentationsmatrix des Fano Matroid F_7 angegeben und rechts steht die Repräsentationsmatrix des dualen Fano Matroid F_7^*.

$$\left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right) \qquad \left(\begin{array}{ccc|cccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right)$$

Wir wollen nun einige duale Eigenschaften von graphischen Matroiden aufzeigen. Wir definieren zunächst, was wir unter dem dualen Matroid, was durch einen Graphen G induziert wird, verstehen.

Definition 2.4.10. *Sei $G = (V, E)$ ein Graph. Das duale Matroid $M(G)^*$ von $M(G)$ nennen wir Kokreismatroid. Ein Matroid, das isomorph zu einem Kokreismatroid ist, heißt kographisch.*

Ein Graph wird *planar* genannt, wenn er in der Ebene so gezeichnet werden kann, so dass sich Kanten höchstens an deren Enden schneiden. Zwei besondere Graphen sind die beiden nicht planaren Graphen K_5 und $K_{3,3}$ (siehe Abb. 2.5).

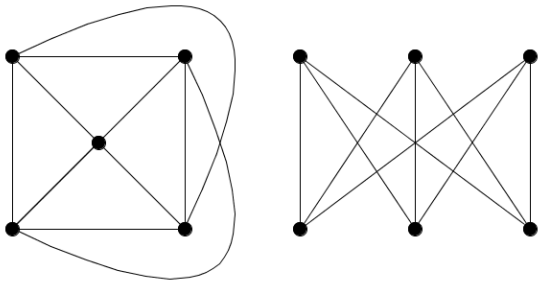


Abb. 2.5.: Die beiden Graphen K_5 (links) und $K_{3,3}$ (rechts).

Nun ist es möglich bei diesen beiden Graphen so Knoten einzufügen, dass Kanten entsprechend unterteilt werden.

Definition 2.4.11. *Jeder Graph G' der von einem Graphen G durch eine Sequenz von Unterteilungen von Kanten durch Knoten abgeleitet ist, wird eine Subdivision von G genannt.*

In Abbildung 2.6 sind Subdivisionen von K_5 und $K_{3,3}$ dargestellt.

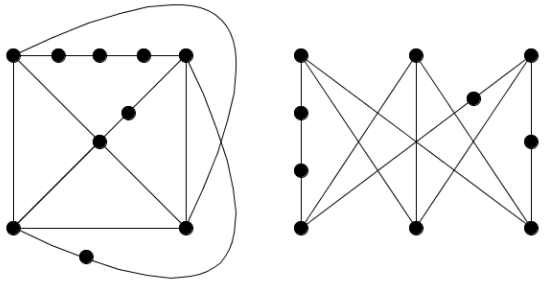


Abb. 2.6.: Eine Subdivision von K_5 (links) und eine Subdivision von $K_{3,3}$ (rechts).

Über die Subdivision ist es möglich eine Charakterisierung planarer Graphen anzugeben.

Satz 2.4.9. *Ein Graph G ist planar genau dann, wenn er keine Subdivision von K_5 und $K_{3,3}$ enthält (siehe Bondy/Murty [3] Seite 268).*

Anhand der Planarität ist es möglich die folgenden beiden Aussagen über duale Matroide zu treffen.

Satz 2.4.10. *Weder $M(K_5)^*$ noch $M(K_{3,3})^*$ sind graphisch (Beweis Oxley [28] Seite 90).*

Satz 2.4.11. *Ist $G = (V, E)$ planar, dann ist $M(G)^*$ graphisch (Beweis Oxley [28] Seite 91).*

Zu jedem regulären Matroid existiert ein duales Matroid. Wie der folgende Satz zeigt, ist das duale Matroid eines regulären Matroids auch wieder regulär.

Satz 2.4.12. *Sei M ein reguläres Matroid, dann ist auch das duale Matroid M^* regulär (Beweis Oxley [28] Seite 85).*

Als nächstes wollen wir den, in der Matroidtheorie, gebrauchten Begriff *Minor* beleuchten. Dazu benötigen wir die Operation der *Kontraktion*. Die Kontraktion ist die duale Operation des Löschens bzw. der Restriktion (siehe Abschnitt 2.4.1) von Elementen eines Matroids.

Definition 2.4.12. *Sei $M = (E, \mathcal{I})$ ein Matroid und $T \subseteq E$. Die Kontraktion T in M ist dann das Matroid $M/T = (M^* \setminus T)^*$ mit der Grundmenge $E \setminus T$.*

Die Kontraktionsoperation ist assoziativ und kommutativ. Selbiges gilt auch für die Löschoperation.

Satz 2.4.13. *Sei $M = (E, \mathcal{I})$ ein Matroid und $T_1, T_2 \subseteq E$ mit $T_1 \cap T_2 = \varnothing$. Dann gilt*

(i) $(M \setminus T_1) \setminus T_2 = M \setminus (T_1 \cup T_2) = (M \setminus T_2) \setminus T_1$,

(ii) $(M/T_1)/T_2 = M/(T_1 \cup T_2) = (M/T_2)/T_1$ *und*

(iii) $(M \setminus T_1)/T_2 = (M/T_2) \setminus T_1$

(Beweis Oxley [28] Seite 109).

Eine wichtige Konsequenz aus Satz 2.4.13 ist, dass jede Abfolge von Lösch- und Kontraktionsoperationen über einem Matroid M in der Form $M \setminus X/Y$ mit $X \cap Y = \varnothing$ geschrieben werden kann.

Definition 2.4.13. *Sei $M = (E, \mathcal{I})$ ein Matroid und M' durch eine Abfolge von Lösch- und Kontraktionsoperationen aus M entstanden, dann ist M' ein Matroid welches wir als einen Minor von M bezeichnen.*

Das Konzept der *verbotenen Minoren* spielt in der Theorie der Matroide eine sehr wichtige Rolle, denn darüber ist es möglich Klassen von Matroiden zu charakterisieren. Neben den Matroiden F_7, F_7^*, $M(K_5)$, $M(K_5)^*$, $M(K_{3,3})$ und $M(K_{3,3})^*$, benötigen wir dazu noch das *uniforme Matroid*.

Definition 2.4.14. *Seien $m, n \in \mathbb{N}$ mit $0 \leq m \leq n$. Weiter sei E eine Menge mit $|E| = n$ und $\mathcal{B}$ die Menge aller m-elementigen Teilmengen von E. $\mathcal{B}$ erfüllt (B1) und (B2). Das zugehörige Matroid nennen wir das uniforme Matroid $U_{m,n}$ von Rang m mit n Elementen.*

In Abbildung 2.7 sind verschiedene Klassen von Matroiden (planar, graphisch, kographisch, regulär, binär, ternär, repräsentierbar) und deren Beziehungen über verbotene Minoren dargestellt. Ein Pfeil von einer Klasse K_1 zu einer Klasse K_2 impliziert, dass die Klasse K_1 in K_2 enthalten ist. Die Beschriftung auf den Pfeilen gibt die verbotenen Minoren der Klasse K_1 in K_2 an.

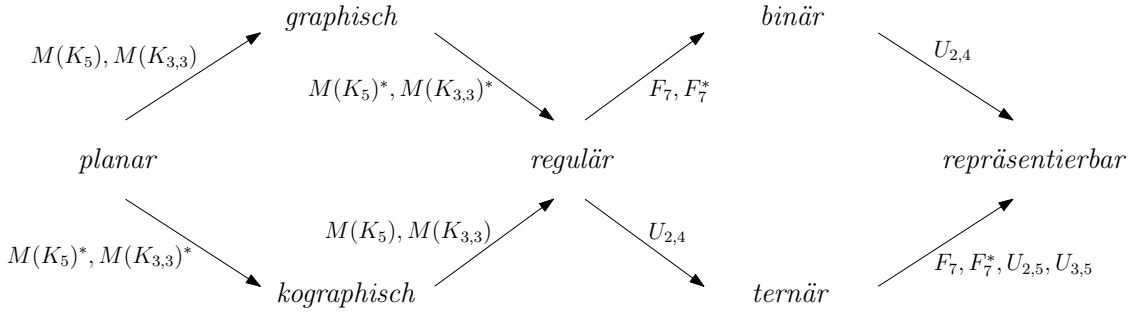


Abb. 2.7.: Beziehungen zwischen verschiedenen Klassen von Matroiden unter Ausschluss von Minoren.

Einige der bereits erwähnten Klassen von Matroiden haben die Eigenschaft, dass alle konstruierbaren Minoren ebenfalls in dieser Klasse liegen. Man spricht in diesem Fall davon, dass die Klasse *unter Minorenbildung abgeschlossen* ist. Auf die graphischen und regulären Matroide trifft dies zu.

Satz 2.4.14. *Jeder Minor eines graphischen Matroids ist graphisch (siehe Oxley [28] Seite 112).*

Satz 2.4.15. *Jeder Minor eines regulären Matroids ist regulär (siehe Oxley [28] Seite 112).*

2.4.3. Reguläre Matroide

Reguläre Matroide stellen eine Klasse von Matroiden dar, welche herausragende Eigenschaften besitzen. Einige dieser Eigenschaften wurden erstmals von Tutte [41] veröffentlicht. Das besondere Interesse an regulären Matroiden hat mehrere Gründe. Einerseits sind graphische und kographische Matroide in der Klasse der regulären Matroide enthalten (siehe Abb. 2.7). Andererseits gibt es Matroide, wie wir später sehen werden, die weder graphisch, noch kographisch, aber regulär sind. Zudem ist es möglich eine Vielzahl von Eigenschaften von Graphen in regulären Matroiden zu verallgemeinern. Der Beweis der graphentheoretischen Eigenschaften setzt meist die Existenz von Knoten voraus. Im Kontext von regulären Matroiden existieren aber keine Knoten und deshalb sind diese Beweise gewöhnlich anders zu führen. Man gewinnt dabei allerdings häufig ein tiefer gehendes Verständnis der zugrunde liegenden, kombinatorischen Struktur. In diesem Abschnitt werden wir neben Definition 2.4.4 noch drei weitere Charakterisierungen für reguläre Matroide aufzeigen. Wir orientieren uns inhaltlich an Hamacher [15].

Eine erste Charakterisierung von regulären Matroiden geht zurück auf Tutte [41].

Satz 2.4.16. *Ein Matroid $M = (E, \mathcal{I})$ wird regulär genannt genau dann, wenn es durch eine total unimodulare Matrix (siehe Definition 2.2.2) repräsentiert werden kann (siehe Hamacher [15] Seite 8).*

Da jede Inzidenzmatrix eines Graphen G ohne Schleifen eine total unimodulare Matrix ist, folgt dass $M(G)$ regulär ist. Zudem ist $M(G)^*$ regulär (siehe Satz 2.4.12).

Eine weitere Charakterisierung regulärer Matroide wurde in Form einer Kardinalitätsaussage von Minty [27] gegeben.

Satz 2.4.17. *Das Matroid $M = (E, \mathcal{C})$ mit der Menge der Kreise $\mathcal{C}$ und der Menge der Kokreise $\mathcal{D}$ wird regulär genannt genau dann, wenn jeder Kreis $C \in \mathcal{C}$ und jeder Kokreis $D \in \mathcal{D}$ mit $C = C^+ \cup C^-$ and $D = D^+ \cup D^-$ so partitioniert werden kann, dass*

$$|C^+ \cap D^+| + |C^- \cap D^-| = |C^+ \cap D^-| + |C^- \cap D^+| \tag{2.18}$$

für jedes Paar $(C, D) \in \mathcal{C} \times \mathcal{D}$ gilt (siehe Hamacher [15] Seite 9).

Die Kardinalitätsaussage (2.18) wird als *Partitionierungseigenschaft* eines regulären Matroid bezeichnet. Die Elemente $e \in C^+$ und $e \in C^-$ nennen wir die *positiven* bzw. *negativen Elemente* von C. Selbiges gilt für $e \in D^+$ und $e \in D^-$. Im weiteren Verlauf

verwenden wir zudem die folgenden Notationen:

$$\begin{aligned} \mathcal{C}_e &:= \{C \in \mathcal{C} \mid e \in C\}, \\ \mathcal{C}_e^+ &:= \{C \in \mathcal{C} \mid e \in C^+\} \text{ und} \\ \mathcal{C}_e^- &:= \{C \in \mathcal{C} \mid e \in C^-\}. \end{aligned}$$

Analog verwenden wir Notationen für $\mathcal{D}_e, \mathcal{D}_e^+$ und $\mathcal{D}_e^-$. Die Partitionierung von $C \in \mathcal{C}$ kann auf folgende Art mit Inzidenzvektoren $x_C := (x_C(e))_{e \in E} \in \{0, 1, -1\}^{|E|}$ in Verbindung gebracht werden:

$$\begin{aligned} X_C(e) = 1 &\Leftrightarrow e \in C^+, \\ X_C(e) = 0 &\Leftrightarrow e \notin C \text{ und} \\ X_C(e) = -1 &\Leftrightarrow e \in C^-. \end{aligned}$$

Analoges gilt für $D \in \mathcal{D}$. Satz 2.4.17 kann damit auch in einer Art dargestellt werden, aus welcher die Orthogonalität der Kreis- und Kokreisinzidenzvektoren ersichtlich ist.

Satz 2.4.18. *M ist ein reguläres Matroid genau dann, wenn jedem $C \in \mathcal{C}$ und jedem $D \in \mathcal{D}$ ein entsprechender Inzidenzvektor $x_C, x_D \in \{0, 1, -1\}^{|E|}$ zugeordnet werden kann, so dass*

$$x_C^T x_D := \sum_{e \in E} x_C(e) x_D(e) = 0 \qquad \forall (C, D) \in \mathcal{C} \times \mathcal{D} \tag{2.19}$$

gilt (siehe Hamacher [15] Seite 10).

Eine Verallgemeinerung von Satz 2.4.18 führt zur Klasse der *orientierten Matroide*. Innerhalb der Klasse der orientierten Matroide ist es möglich Konzepte der gerichteten Graphen oder linearen Programmierung zu verallgemeinern. Wir gehen hier nicht weiter auf die Theorie der orientierten Matroide ein und verweisen auf [2].

Als nächstes wollen wir zeigen, dass einfache Änderungen der Partitionierung keinen Einfluss auf die Kardinalitätsaussage 2.18 haben. Wir definieren zunächst was wir unter einer *umgekehrten Orientierung* in einem Element $e \in E$ verstehen.

Definition 2.4.15. *Sei $M = (E, \mathcal{C})$ ein reguläres Matroid mit $e \in E$. M kehrt seine Orientierung in e genau dann um, wenn die Partitionen $C = C^+ \cup C^-$ und $D = D^+ \cup D^-$ durch $C = \hat{C}^+ \cup \hat{C}^-$ und $D = \hat{D}^+ \cup \hat{D}^-$ mit*

$$\begin{aligned} \hat{C}^+ &:= (C^+ \setminus \{e\}) \cup (C^- \cap \{e\}), \\ \hat{C}^- &:= (C^- \setminus \{e\}) \cup (C^+ \cap \{e\}), \end{aligned} \tag{2.20}$$

und

$$\begin{aligned} \hat{D}^+ &:= (D^+ \setminus \{e\}) \cup (D^- \cap \{e\}), \\ \hat{D}^- &:= (D^- \setminus \{e\}) \cup (D^+ \cap \{e\}), \end{aligned} \tag{2.21}$$

ersetzt werden.

Das zu e umgekehrte Element bezeichnen wir mit e^{-1}. Entsprechend bezeichnen wir für $T \subseteq E$ mit T^{-1} eine Teilmenge von Elementen der Grundmenge, die alle in der Orientierung umgekehrt wurden, d.h. $T^{-1} := \{e^{-1} \mid e \in E\}$. Nun kann man zeigen, dass die Eigenschaft ein reguläres Matroid zu sein nicht dadurch verloren geht, wenn die Orientierung von einigen Elementen umgekehrt wird.

Satz 2.4.19. *Sei $E' \subseteq E$ und $M = (E, \mathcal{C})$ ein reguläres Matroid mit Partitionen $C = C^+ \cup C^-$ und $D = D^+ \cup D^-$. Falls M seine Orientierung in jedem $e \in E'$ umkehrt, dann bleibt die Partitionierungseigenschaft* (2.18) *weiterhin gültig (Beweis Hamacher [15] Seite 13).*

Abschließend wollen wir uns eine Charakterisierung regulärer Matroide über binäre Matroide von Seymour [34] anschauen. Dafür benötigen wir zuerst die Begriffe der *symmetrischen Differenz*, *1-Summe*, *2-Summe* und *3-Summe*.

Definition 2.4.16. *Gegeben zwei binäre Matroide $M_1 = (E_1, \mathcal{C}_1)$ und $M_2 = (E_2, \mathcal{C}_2)$. Mit $M_1 \triangle M_2$ bezeichnen wir die symmetrische Differenz von M_1 und M_2 mit Grundmenge $E_1 \triangle E_2 = (E_1 \setminus E_2) \cup (E_2 \setminus E_1)$.*

Sei $\mathcal{C}_i$ die Familie aller Teilmengen von E_i die als disjunkte Vereinigung von Kreisen von M_i mit $i = 1, 2$ darstellbar sind. Dann ist die Menge der Kreise $\mathcal{C}_1 \triangle \mathcal{C}_2$ von $M_1 \triangle M_2$ definiert als $\mathcal{C}_1 \triangle \mathcal{C}_2 := \min(\{C_1 \triangle C_2 \mid C_i \in \mathcal{C}_i, i = 1, 2\})$.

Definition 2.4.17. *Seien $M = (E_1, \mathcal{C}_1)$ und $M_2 = (E_2, \mathcal{C}_2)$ binäre Matroide mit $\mathcal{C}_1, \mathcal{C}_2$ als Mengen der Kreise und $\mathcal{D}_1, \mathcal{D}_2$ als Mengen der Kokreise.*

(i) *$M_1 \triangle M_2$ ist die 1-Summe (Schreibweise $M_1 \oplus_1 M_2$) von M_1 und M_2, falls $E_1 \cap E_2 = \varnothing$ und $|E_1|, |E_2| < |E_1 \triangle E_2|$.*

(ii) *$M_1 \triangle M_2$ ist die 2-Summe (Schreibweise $M_1 \oplus_2 M_2$) von M_1 und M_2, falls $E_1 \cap E_2 = \{e\}, \{e\} \notin (\mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{D}_1 \cup \mathcal{D}_2)$ und $|E_1|, |E_2| < |E_1 \triangle E_2|$.*

(iii) *$M_1 \triangle M_2$ ist die 3-Summe (Schreibweise $M_1 \oplus_3 M_2$) von M_1 und M_2, falls $E_1 \cap E_2 = T$, mit $|T| = 3$, $T \in \mathcal{C}_1 \cap \mathcal{C}_2, D \subsetneq T \ \forall D \in \mathcal{D}_i, i = 1, 2$ und $|E_1|, |E_2| < |E_1 \triangle E_2|$.*

Seymour hat gezeigt, dass jedes reguläre Matroid durch 1-, 2- und 3-Summen Kombinationen von graphischen, kographischen Matroiden und einem speziellen Matroid R_{10} erzeugt werden kann. Wir sprechen in diesem Fall auch von der *Komposition* von Matroiden. Der Satz gilt auch für den gegensätzlichen Fall der *Dekomposition*. Bevor wir den Satz von Seymour angeben, zeigen wir noch zwei unterschiedliche Repräsentationsmatrizen des R_{10} Matroids über GF(2). Dieses Matroid ist regulär, aber weder graphisch noch kographisch. $R_{10} \cong M[(I_5|B^{10.1})]$, $R_{10} \cong M[(I_5|B^{10.2})]$

mit

$$B^{10.1} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{und} \quad B^{10.2} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Satz 2.4.20. *Jedes reguläre Matroid kann durch 1-Summen, 2-Summen und 3-Summen Kombinationen von graphischen, kographischen Matroiden und dem R_{10} Matroid konstruiert werden.*

In Trümper [39] ist ein Algorithmus angegeben, der den obigen Satz 2.4.20 nutzt um in Polynomialzeit $O((m+n)^3)$ zu entscheiden, ob eine $(m \times n)$-Matrix total unimodular ist. Details zur Implementierung des Algorithmus wurden in [43] veröffentlicht.

Signierbarkeit

Reguläre Matroide werden häufig durch Representationsmatrizen dargestellt, die nur $\{0,1\}$ Einträge haben. Wie eine solche Matrix in eine Matrix, die $\{0,1,-1\}$ Einträge enthält, umgewandelt werden kann, soll in diesem Abschnitt gezeigt werden. Die Umwandlung wird auch als *Signierung* bezeichnet. Durch die Signierung wird eine Partitionierung (siehe Definition 2.4.17) bzw. Orientierung der Elemente des Matroids festgelegt.

Definition 2.4.18. *Wir bezeichnen ein Matroid als ungerichtet, falls eine Partitionierung der Kreise und Kokreise existiert und* (2.18) *gilt. Ein Matroid mit einer speziellen Partitionierung, die* (2.18) *erfüllt, wird als gerichtet bezeichnet.*

Damit die signierte Matrix total unimodular ist, muss die $\{0,1\}$-Matrix gewisse Eigenschaften erfüllen.

Definition 2.4.19. *Eine $\{0,1\}$-Matrix A ist regulär, wenn einige 1-Einträge durch -1-Einträge ersetzt werden können, sodass die resultierende Matrix total unimodular ist.*

Camion [6] hat die folgenden beiden Resultate gezeigt.

Satz 2.4.21. *Falls eine $\{0,1\}$-Matrix zu einer total unimodularen Matrix signiert werden kann, dann ist diese Signierung bis auf Multiplikation der Zeilen oder Spalten mit -1 eindeutig.*

Satz 2.4.22. *Eine $\{0,\pm 1\}$-Matrix A ist genau dann total unimodular, wenn für jede quadratische Submatrix A' von A mit geraden Zeilen- und Spaltensummen die Summen aller Einträge von A' durch 4 teilbar ist.*

Es sei erwähnt, dass man auch eine $\{0,1\}$-Repräsentationsmatrix des Fano Matroids F_7 (siehe Beispiel 2.4.4) signieren kann. Das Ergebnis ist allerdings keine total unimodulare Matrix.

Wir wollen nun den Algorithmus 2.4.1 zum Signieren von regulären Matrizen von Camion näher betrachten (siehe Cornuéjols [7] Seiten 67-68). In einer erweiterten Form ist dieser Algorithmus wesentlicher Bestandteil eines Algorithmus, der Matrizen auf totale Unimodularität testet. Details dazu wurden in [39], [40] und [43] veröffentlicht.

Algorithmus 2.4.1 Signierung

Eingabe: Eine reguläre $\{0,1\}$-Matrix A und ein bipartiter Graph G. Die Knoten von G entsprechen den Zeilen und Spalten von A. Eine Zeile r und eine Spalte c sind adjazent, gdw. $A_{rc} = 1$ gilt. Zudem ist ein maximaler Wald F in G mit einer beliebigen Signierung gegeben.

Ausgabe: Eindeutige total unimodulare Signierung von G. Wobei die Kanten von F wie vorgegeben signiert sind.

Indiziere die Kanten von G durch $e_1, \ldots, e_n$, so dass die Kanten von F zuerst kommen und jede Kante e_j mit $j \geq |F|+1$ zusammen mit den Kanten, die kleinere Indizes haben, einen sehnenlosen kleinsten Kreis H_j in G bilden. Für $j = |F| + 1, \ldots, n$ signiere e_j so, dass H_j total unimodular ist.

Flüsse

In diesem Abschnitt sollen Matroidflüsse in gerichteten regulären Matroiden (siehe Definitionen 2.4.17 und 2.4.18) beschrieben werden. Matroidflüsse verallgemeinern das Konzept von Flüssen in gerichteten Graphen. Im Falle graphischer Matroide entsprechen Matroidflüsse den Flüssen in Graphen und im Falle von kographischen Matroiden entsprechen Matroidflüsse den Spannungsproblemen (engl. Tensions) in Netzwerken. Minty [27] war einer der ersten, welcher eine Flusstheorie in regulären Matroiden entwickelte. Wir orientieren uns bei der Darstellung dieses Kapitels an Burkard/Hamacher [5] und Hamacher [15].

Bei der Definition von Matroidflüssen geht man gewöhnlich davon aus, dass ein ausgezeichnetes Element $e_1 \in E$ existiert. Dieses Element kann man in graphischen Matroiden als eine Rückwärtskante verstehen, welches die Senke mit der Quelle verbindet (siehe Abschnitt 2.3.5). $\tilde{E} = E \setminus e_1$ ist dabei die Menge der Elemente ohne e_1. Wir gehen im weiteren Verlauf der Arbeit stets davon aus, dass ein solches Element existiert. Zudem sei auf $\tilde{E}$ eine Kapazitätsfunktion $k : \tilde{E} \to \mathbb{N}_0$ definiert.

Definition 2.4.20. *Sei ein gerichtetes reguläres Matroid $M = (E, \mathcal{C})$ mit Kapazitätsfunktion k gegeben. Eine Funktion $f : E \to \mathbb{N}_0$ wird Matroidfluss genannt, falls die*

(i) Kapazitätseigenschaft

$$0 \leq f(e) \leq k(e) \qquad \forall e \in \tilde{E} \tag{2.22}$$

(ii) und die Kokreiseigenschaft

$$f(D^+) := \sum_{e \in D^+} f(e) = \sum_{e \in D^-} f(e) =: f(D^-) \qquad \forall D \in \mathcal{D} \tag{2.23}$$

erfüllt sind. $f(e_1)$ *ist der Wert des Matroidflusses.* f *wird maximaler Matroidfluss genannt, wenn der Wert* $f(e_1)$ *maximal ist.*

Es ist möglich in (2.22) anstelle von 0 eine zusätzliche Kapazitätsfunktion für die unteren Schranken des Matroidflusses anzugeben. Darauf wollen wir hier nicht weiter eingehen und diesbezüglich auf [15] verweisen.

Wir schauen uns nun Kriterien an, welche uns Aussagen lieferen, wann ein Matroidfluss maximal ist.

Lemma 2.4.4. *Sei* f *ein Matroidfluss und* $D \in \mathcal{D}^-_{e_1}$. *Dann gilt* $f(e_1) \leq k(D^+)$ *(Beweis Hamacher [15] Seite 34).*

Nun definieren wir, was wir unter einem Fluss f augmentierenden Kreis verstehen.

Definition 2.4.21. *Sei ein reguläres Matroid* $M = (E, \mathcal{C})$ *mit Fluss* f *gegeben. Wir nennen* $C \in \mathcal{C}^+_{e_1}$ *einen* f *augmentierenden Kreis, wenn* $f(e) > 0$ *für* $e \in C^-$ *und* $f(e) < k(e)$ *für* $e \in C^+ \setminus \{e_1\}$ *gilt.*

Solange f nicht maximal ist, kann man immer einen f augmentierenden Kreis $C \in \mathcal{C}^+_{e_1}$ finden und entsprechend den Fluss f mit

$$\varepsilon(C) := \min(\min_{e \in C^-}(f(e)), \min_{e \in C^+ \setminus \{e_1\}}(k(e) - f(e))) \tag{2.24}$$

und $\chi_C \in \mathbb{R}^E$ mit

$$\chi_C(e) := \begin{cases} 1, & \text{falls } e \in C^+ \\ -1, & \text{falls } e \in C^- \\ 0, & \text{sonst} \end{cases} \tag{2.25}$$

durch

$$f' := f + \varepsilon(C)\chi_C \tag{2.26}$$

anpassen.

Lemma 2.4.5. *Ein Matroidfluss* f *ist genau dann maximal, wenn es keinen* f *augmentierenden Kreis* C *mehr gibt (Beweis Hamacher [15] Seiten 36-37).*

Abschließend wollen wir noch kurz auf das Min-Cost-Flow Problem in regulären Matroiden eingehen. Sei eine Kostenfunktion $C : E \to \mathbb{N}_0$ mit $C(e_1) = 0$ gegeben.

Definition 2.4.22. *Ein Matroidfluss* f *wird extrem genannt, wenn seine Kosten*

$$C(f) := \sum_{e \in E} f(e) \cdot c(e)$$

über alle Matroidflüsse mit Wert $f(e_1)$ *minimal sind. Ein Minimalkostenmatroidfluss ist ein maximaler, extremer Matroidfluss (siehe Burkard/Hamacher [5]).*

In [5] sind drei verschiedene Algorithmen angegeben die das Min-Cost-Flow Problem in regulären Matroiden lösen. Einer der Algorithmen startet mit einem maximalen Matroidfluss und reduziert dann die Kosten über das Finden negativer Kreise in einem inkrementellen Matroid (vgl. Abschnitt 2.3.5).

Komposition und Dekomposition

Wir wollen uns in diesem Abschnitt anschauen, wie sich auf der Ebene von Repräsentationsmatrizen die Komposition und Dekomposition von binären Matroiden darstellt. Dies interessiert uns deswegen, weil die regulären Matroide eine Teilmenge der binären Matroide sind (siehe Abb. 2.7) und wir später einige reguläre Matroide zum Testen, des in Abschnitt 3 entwickelten Algorithmus, selbst erstellen wollen.

In Definition 2.4.8 haben wir spezifiziert, dass eine Standardrepräsentationsmatrix eines Matroids $M = (E, \mathcal{I})$ mit $r = \text{Rang}(M)$ die Form $(I_r|D)$ hat. Dabei ist jeder Spalte ein Element von E zugeordnet. Den Zeilen können wir die gleichen Elemente zuordnen, welche in den Spalten der Einheitsmatrix I_r stehen (siehe Matrix A Abb. 2.8). Ohne Informationen zu verlieren, können wir die Einheitsmatrix weglassen und erhalten dadurch eine *partielle Standardrepräsentationsmatrix* (siehe Matrix B Abb. 2.8). Oft sprechen wir im Folgenden von einer Matrix bzw. Repräsentationsmatrix und meinen dabei eine partielle Standardrepräsentationsmatrix. Aus dem Zusammenhang sollte ersichtlich sein, welcher Fall gemeint ist. Im Graphenfall entsprechen die Elemente von X einem aufspannenden Baum und die Elemente von Y dem entsprechenden Kobaum.

$A =$

	e_1	e_3	e_6	e_2	e_4	e_5
e_1	1	0	0	1	1	0
e_3	0	1	0	1	1	1
e_6	0	0	1	0	1	1

$B =$

		Y		
		e_2	e_4	e_5
X	e_1	1	1	0
	e_3	1	1	1
	e_6	0	1	1

Abb. 2.8.: Partielle Standardrepräsentationsmatrix

Als nächstes soll definiert werden, was unter *Separation* und *Zusammenhang* von Matroiden zu verstehen ist (siehe Seymour [35]).

Definition 2.4.23. *Sei $M = (E, \mathcal{I})$ ein Matroid und $k \in \mathbb{N}$. Eine Partition (U_1, U_2) von E wird eine k-Separation von M genannt, falls $|U_1|, |U_2| \geq k$ und*

$$r(U_1) + r(U_2) \leq r(E) + k - 1. \tag{2.27}$$

Gilt in 2.27 Gleichheit, dann spricht man auch von einer *exakten k-Separation.*

Definition 2.4.24. *Ein Matroid $M = (E, \mathcal{I})$ ist k-zusammenhängend, wenn keine k'-Separation mit $k' < k$ existiert.*

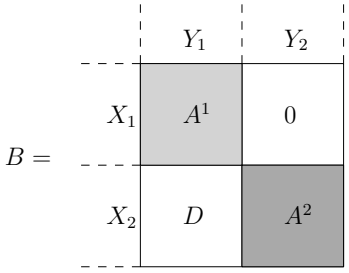


Abb. 2.9.: Matrix B mit exakter k-Separation

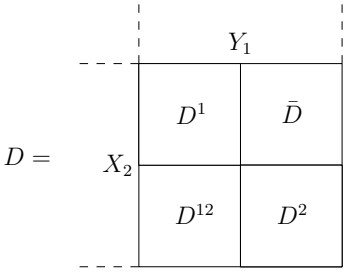


Abb. 2.10.: Unterteilung von D

Im Kontext von Matroiden bezeichnet man ein 2-zusammenhängendes Matroid auch einfach als *zusammenhängend* (vgl. Graphenfall Abschnitt 2.3.1).

Wir wollen nun die k-Separation mit partiellen Standardrepräsentationsmatrizen in Verbindung bringen. Gegeben sei ein Matroid M und eine Partition (E_1, E_2) von E mit exakter k-Separation. X_2 sei eine maximale unabhängige Teilmenge von E_2. Wir vergrößeren nun X_2 durch eine Teilmenge von X_1 zu einer Basis von M. Zudem ist $Y_i = E_i - X_i$ für $i = 1, 2$. Die partielle Standardrepräsentationsmatrix B von M zur Basis $X_1 \cup X_2$ stellt sich dann wie in Abbildung 2.9 dar, wobei $|X_1 \cup Y_1|, |X_2 \cup Y_2| \geq k$ und der GF(2)-Rang von D gleich $k-1$ sind (siehe Trümper [40] Seite 169). Die Matrix D wollen wir weiter unterteilen, nämlich in vier Submatrizen (siehe Abb. 2.10). Wir nehmen an, dass die beiden Matrizen bestehend aus D^1 und $\bar{D}$ bzw. $\bar{D}$ und D^2 den GF(2)-Rang $k-1$ haben, dann kann man zeigen, dass $\bar{D}$ wie D auch den GF(2)-Rang $k-1$ hat (siehe Trümper [40] Seite 174). Sei weiter $\bar{D}$ eine invertierbare quadratische Matrix, dann existiert eine Matrix $F = D^2 \cdot \bar{D}^{-1}$ mit $D^2 = F \cdot \bar{D}$ und $D^{12} = F \cdot D^1$. Somit können wir D^{12} über die anderen drei Matrizen berechnen:

$$D^{12} = F \cdot D^1 = D^2 \cdot \bar{D}^{-1} \cdot D^1. \tag{2.28}$$

Mit den geleisteten Vorarbeiten sind wir nun in der Lage die k-Summe von binären Matroiden zu behandeln. Uns interessieren aber, wegen Satz 2.4.20, nur die Fälle $k = 1, 2, 3$. In Definition 2.4.17 sind wir bereits auf die 1-, 2- und 3-Summe von binären Matroiden eingegangen. Hier wollen wir die 1-, 2- und 3-Summe durch partielle Standardrepräsentationsmatrizen darstellen. Gegeben sei ein binäres Matroid M mit Matrix B und 1-Separation. Folglich ist $D = 0$ und die beiden Matrizen B_1 und B_2 stellen die entsprechenden partiellen Standardrepräsentationsmatrizen der binären Matroide M_1 und M_2 einer *1-Summen Dekomposition* dar (siehe Abb. 2.11). Die umgekehrte Operation wird *1-Summen Komposition* genannt.

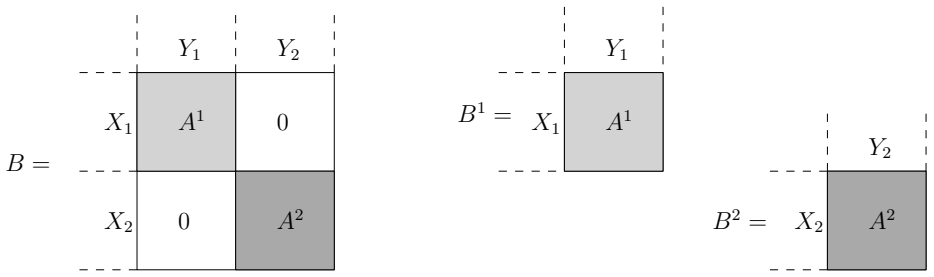


Abb. 2.11.: 1-Summe

Interessanter als der Fall der 1-Summe ist der Fall der 2-Summe. Gegeben sei ein zusammenhängendes binäres Matroid M mit Matrix B und 2-Separation. Weil M zusammenhängend ist, muss die 2-Separation exakt sein. Da der GF(2)-Rang von D gleich $k - 1 = 1$ ist, gestaltet sich eine *2-Summen Dekomposition* wie in Abbildung 2.12. Dabei stellen B_1 und B_2 die partiellen Standardrepräsentationsmatrizen der dekomponierten binären Matroide dar. Die umgekehrte Operation wird *2-Summen Komposition* genannt. Die Matrix $\bar{D}$ im Überlappungsbereich besteht hier nur aus 1-Einträgen und hat den GF(2)-Rang $k - 1 = 2 - 1 = 1$.

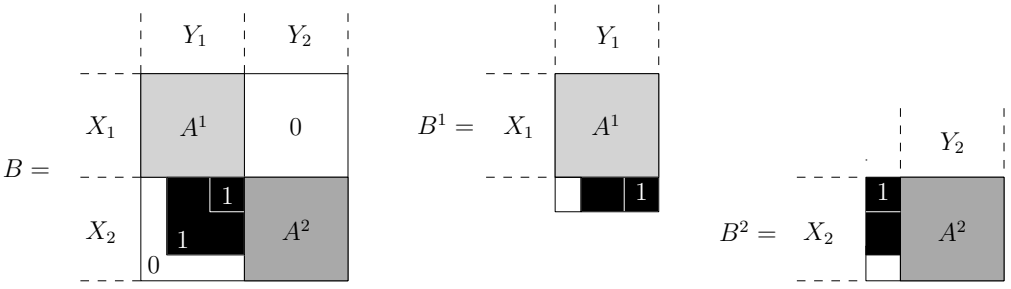


Abb. 2.12.: 2-Summe

Um alle Fälle von Satz 2.4.20 behandeln zu können, benötigen wir noch die 3-Summe. Sei ein 3-zusammenhängendes Matroid M mit Matrix B und 3-Separation gegeben, dann gestaltet sich eine *3-Summen Dekomposition* wie in Abbildung 2.13. Am einfachsten ist es hier für $\bar{D}$ im Überlappungsbereich eine 2×2-Matrix mit vollem GF(2)-Rang anzunehmen. An den entsprechenden, zu $\bar{D}$ angrenzenden Einträgen in A^1 und A^2 werden 1-Einträge angenommen. B_1 und B_2 stellen wieder die partiellen Standardrepräsentationsmatrizen der dekomponierten binären Matroide dar. Den umgekehrten Fall bezeichnen wir als *3-Summen Komposition*. Die Berechnung der Matrix D^{12} kann mit (2.28) erfolgen.

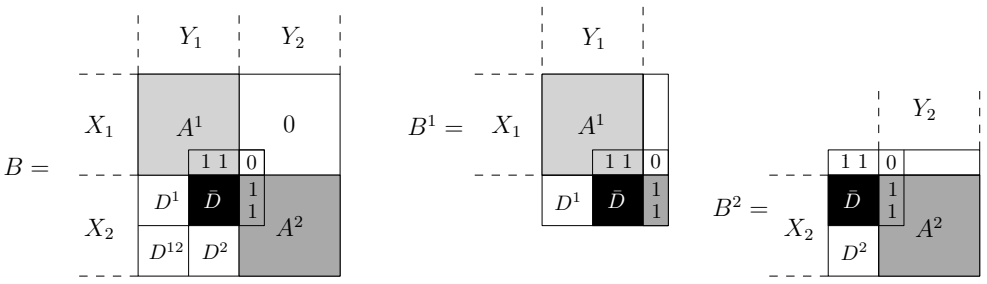


Abb. 2.13.: 3-Summe

2.4.4. Orakel

Um Eigenschaften von Matroiden überprüfen zu können, stellt sich die Frage, wie man Matroide geeignet darstellt, um z.B. ein Computerprogramm schreiben zu können, welches Matroide als Eingabe akzeptiert. Da es mindestens 2^{2^n} verschiedene Matroide mit n Elementen gibt (siehe Lovász/Recski [26]), folgt daraus, dass bei jeder beliebigen Kodierungsart immer Matroide existieren, deren Kodierung eine exponentielle Länge hat. Aufgrund dieser Tatsache hat sich ein anderes Konzept der Repräsentation von Matroiden etabliert. Matroide werden durch *Orakel* dargestellt.

Wir verwenden als Kodierungslänge eines Matroids $M = (E, \mathcal{I})$ die Anzahl der Elemente von E. Dabei ist das Matroid selbst in dem Orakel verborgen. Das Orakel kann man als eine Art Subroutine interpretieren, welches Fragen eines speziellen Typs beantwortet (siehe Grötschel [12]).

Beispiel 2.4.5. Sei ein Matroid $M = (E, \mathcal{I})$ mit bekannter Grundmenge E gegeben.

Orakelname	Eingabe	Frage	Ausgabe
Unabhängigkeitsorakel	$F \subseteq E$	Ist F unabhängig?	ja/nein
Basisorakel	$F \subseteq E$	Ist F eine Basis von E?	ja/nein
Kreisorakel	$F \subseteq E$	Ist F ein Kreis?	ja/nein
Rangorakel	$F \subseteq E$	Wie groß ist der Rang von F?	Rang(F)

Wenn wir einen Algorithmus betrachten, welcher Eigenschaften von Matroiden überprüft, so gehen wir stets davon aus, dass ein Matroid durch ein Orakel und die Grundmenge E gegeben ist. Bei der Komplexitätsanalyse von Algorithmen für Matroide wird dann ein Orakelaufruf nur als ein Schritt gezählt. Man nennt Verfahren, die Orakel aufrufen können, *Orakelalgorithmen*. Ist ihre Laufzeit in dem eben festgelegten Sinne polynomial in $|E|$, dann sagt man, dass die Verfahren *orakelpolynomial* sind.

Hat man einen Algorithmus, dessen Laufzeit orakelpolynomial in $|E|$ ist, so ist der Algorithmus für alle Matroide im üblichen Sinne polynomial, für die das Orakel durch einen in $|E|$ polynomiellen Algorithmus realisiert werden kann. Beispielsweise ist dies bei Vektormatroiden $M[A]$ für das Unabhängigkeitsorakel der Fall, denn bei gegebener Matrix A kann man die Rangbestimmung durch Gauß-Elimination

realisieren.

2.4.5. Greedy-Algorithmus

Es soll in diesem Abschnitt untersucht werden, wie gut ein Optimierungsproblem der Form

$$\max\{w(I) \mid I \in \mathcal{I}\}, \tag{2.29}$$

gelöst werden kann. Dabei ist $\mathcal{I} \subseteq 2^E$ eine Mengenfamilie, die (I1) und (I2) aus Definition 2.4.1 erfüllt. Somit ist $\mathcal{I}$ ein Unabhängigkeitssystem auf einer Grundmenge E. $w : E \to \mathbb{R}$ ist eine Gewichtsfunktion und wir definieren für $X \subseteq E$ das Gewicht von X als $w(X) := \sum_{e \in X} w(e)$.

Wir betrachten den folgenden Algorihmus, welcher auch als Greedy-Algorithmus (engl. greedy = gefräßig) bezeichnet wird. Dessen Idee besteht darin, eine von der leeren Menge ausgehende Lösung aufzubauen und dabei in einer Iteration immer dasjenige Element zu wählen, das den meisten Fortschritt verspricht (siehe Grötschel [12]).

Algorithmus 2.4.2 Greedy

Eingabe: Grundmenge $E = \{e_1, e_2, \ldots, e_n\}$ mit Gewichten $w(e_i) \in \mathbb{R}\ \forall i = 1, \ldots, n$. Zudem ist ein Unabhängigkeitssystem $\mathcal{I} \subseteq 2^E$ durch ein Unabhängigkeitsorakel (siehe Beispiel 2.4.5) gegeben.

Ausgabe: Eine unabhängige Menge $I_g \in \mathcal{I}$.

1: Sortiere E, so daß $w(e_1) \geq w(e_2) \geq \ldots \geq w(e_n)$ gilt.
2: Setze $I := \varnothing$.
3: **for** $i = 1, \ldots, n$ **do**
4: Ist $w(e_i) \leq 0$, gehe zu 7.
5: Ist $I \cup e_i$ unabhängig (Orakelaufruf), dann setze $I := I \cup e_i$.
6: **end for**
7: Setze $I_g := I$ und gib I_g aus.

Der folgende Satz wurde unabhängig von verschiedenen Autoren bewiesen.

Satz 2.4.23. *Ein Unabhängigkeitssystem $(E, \mathcal{I})$ ist genau dann ein Matroid, wenn der Greedy Algorithmus 2.4.2 eine Optimallösung für das Problem* (2.29) *bestimmt (Beweis siehe Korte/Vygen [24] Seite 345).*

Matroide sind also genau die Unabhängigkeitssysteme, wo der Greedy-Algorithmus für eine beliebige Gewichtsfunktion die Optimallösung liefert. Somit kann man in Definition 2.4.1 das Basisergänzungsaxiom durch ein Greedy-Optimalitätsaxiom ersetzen.

Satz 2.4.24. *Sei $\mathcal{I} \subseteq 2^E$ eine Mengenfamilie. Dann ist $\mathcal{I}$ die Familie der unabhängigen Mengen eines Matroids genau dann, wenn*

(I1) $\varnothing \in \mathcal{I}$,

(I2) $I \in \mathcal{I}$ *und* $I' \subseteq I \Rightarrow I' \in \mathcal{I}$ *und*

(I3) für alle Gewichtsfunktionen $w : E \to \mathbb{R}$ *liefert der Greedy-Algorithmus eine maximale Menge in* $\mathcal{I}$ *von maximalem Gewicht*

(Beweis Oxley [28] Seiten 64-65).

Das Maximierungsproblem (2.29) kann man anhand einer polynomialen Transformation in ein Minimierungsproblem

$$\min_{B \text{ Basis}} w(B) \tag{2.30}$$

überführen und aus Algorithmus 2.4.2 einen Minimierungsalgorithmus über Basissysteme ableiten. Eine typische Anwendung ist hier z.B. die Berechnung eines minimal aufspannenden Baums eines zusammenhängenden Graphen.

3. Herleitung des Algorithmus

3.1. Min-Cost-Flow mit konvexer separabler Kostenfunktion

Im Abschnitt 2.4.3 sind wir bereits auf das Min-Cost-Flow Problem in regulären Matroiden eingegangen. Dort haben wir den Fall einer einfachen Kostenfunktion betrachtet. Hier wollen wir die Kostenfunktion erweitern. Indem wir die Kapazitäten auf den Elementen des Matroids mit in die Kostenfunktion modellieren, erhalten wir einen einfachen Fall einer konvexen separablen Kostenfunktion (siehe Abschnitt 2.3.5). Anschließend zeigen wir einige Eigenschaften die aus der Konvexität der Kostenfunktion folgen. Für die Darstellung der folgenden beiden Abschnitte 3.1 und 3.2 nutzen wir Ergebnisse aus Karzanov/McCormick [22].

Für jedes Element sind Schranken bzw. Kapazitäten $\ell, u \in \mathbb{R}^E$ mit $\ell \leq u$ und Kosten $c \in \mathbb{R}^E$ gegeben. Indem wir die Schranken über eine Funktion $w_e : \mathbb{R} \to \mathbb{R}$

$$w_e(r) = \begin{cases} c_e u_e + \alpha(r - u_e), & \text{falls } r > u_e, \\ c_e r, & \text{falls } \ell_e \leq r \leq u_e, \\ c_e \ell_e + \alpha(\ell_e - r), & \text{falls } r < \ell_e \end{cases} \tag{3.1}$$

abbilden, erhalten wir eine konvexe separable Kostenfunktion (siehe Abb. 3.1). Dabei ist α eine ausreichend große, positive Zahl und r der aktuelle Flusswert auf dem entsprechenden Element. Dabei dürfen durchaus mehrere Segmente c_e vorhanden sein. Wir fordern nur, dass

$$w(x) = \sum_{e \in E} w_e(x_e) \tag{3.2}$$

eine konvexe separable Kostenfunktion ist. x ist eine Zirkulation bzw. ein Matroidfluss.

Sei L ein Untervektorraum von $\mathbb{R}^E$ mit Koordinaten, welche durch die Elemente von E indiziert werden. $L^\perp$ ist das orthogonale Komplement von L. Das Min-Cost-Flow Problem stellt sich dann so dar:

$$\text{Bestimme eine Zirkulation } x \in L \text{ die } w(x) \text{ minimiert.} \tag{3.3}$$

Wir nehmen an, dass L nicht nur den Nullpunkt enthält und (3.3) eine endliche Optimallösung besitzt. Zudem treffen wir die Einschränkung, dass es sich bei L um einen *unimodularen Raum* handelt. Somit kann L durch eine total unimodulare $(m \times n)$-Matrix M (siehe Definition 2.2.2) dargestellt werden. Wir nehmen auch an, dass M vollen Zeilenrang hat. Die Anzahl der Spalten entspricht der Anzahl der Elemente des Matroids $n = |E|$. Wir können die Spalten von M permutieren und

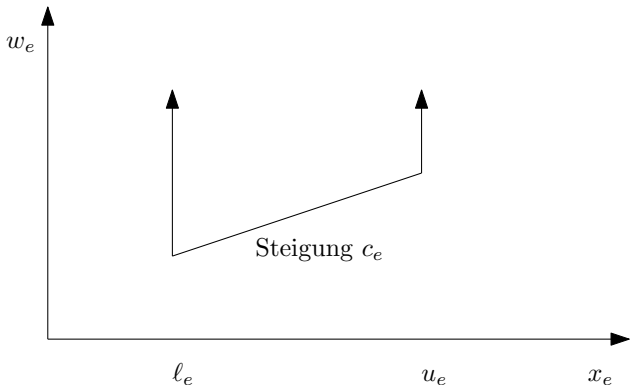


Abb. 3.1.: Konvexe separable Kostenfunktionen mit unterer Schranke ℓ_e, oberer Schranke u_e und Steigung c_e.

erhalten eine Matrix $(B\ N)$, wobei B eine nicht-singuläre $(m \times m)$-Matrix ist. Da Pivots die totale Unimodularität der Matrix erhalten (siehe Trümper [40] Seiten 21-22), stellt $(I\ B^{-1}N)$ eine Standardrepräsentationsmatrix (siehe Definition 2.4.8) von L dar. Die $((n-m) \times n)$-Matrix $K = ((B^{-1}N)^T\ -I)$ hat vollen Zeilenrang und ist ebenfalls eine total unimodulare Matrix. Da $\{y \in \mathbb{R}^E : Ky = 0\} = L^\perp$ gilt, ist auch $L^\perp$ ein unimodularer Raum und K repräsentiert das duale Matroid (siehe Satz 2.4.8). Die Vektoren aus L nennen wir Zirkulationen bzw. Matroidflüsse und die Vektoren aus $L^\perp$ bezeichnen wir als Kozirkulationen.

Ein nicht-null Vektor $\xi \in L$ ($\xi \in L^\perp$), dessen Träger $\operatorname{supp}(\xi) = \{e \in E \mid \xi_e \neq 0\}$ minimal bzgl. Inklusion ist, wird als Kreis von L bzw. als Kokreis von $L^\perp$ bezeichnet. Ein unimodulares System ist dadurch gekennzeichnet, dass alle Kreise bzw. Kokreise multiplikative Vielfache von $(0, \pm 1)$-Vektoren sind. Die Größe eines Kreises bzw. Kokreises ist die Anzahl der nicht-null Einträge des entsprechenden Vektors. Mit ϕ bzw. $\phi^\perp$ bezeichnen wir die Größe des größten Kreises bzw. größten Kokreises. Im Folgenden verwenden wir zudem die Variablen $\rho, \rho^\perp$ mit

$$\rho = \operatorname{Rang}(M) + 1 = m + 1 \text{ und} \tag{3.4}$$

$$\rho^\perp = \operatorname{Rang}(K) + 1 = n - m + 1. \tag{3.5}$$

Wir erhalten damit folgenden Zusammenhang:

$$\phi \leq \rho = m + 1 \text{ und} \tag{3.6}$$

$$\phi^\perp \leq \rho^\perp = n - m + 1. \tag{3.7}$$

Die Konvexität von w_e sichert uns die Existenz einer *rechtsseitigen Ableitung* $c_e^\vdash(r)$ und einer *linksseitigen Ableitung* $c_e^\dashv(r)$ für $r \in \mathbb{R}$. Dabei ist $c_e^\dashv(r) < c_e^\vdash(r)$ an einem Breakpoint (vgl. Abschnitt 2.3.5) möglich. Das Intervall $[c_e^\dashv(r), c_e^\vdash(r)]$ wird als *Subdifferential* von $w_e(r)$ und $y \in [c_e^\dashv(r), c_e^\vdash(r)]$ wird als *Subgradient* von w_e bei r bezeichnet (siehe Rockafellar [30] Seiten 213-226). Zusätzlich sichert uns die Konvextät von w_e die folgende Eigenschaft:

$$c_e^\dashv(r) \leq c_e^\vdash(r) \leq c_e^\dashv(r') \leq c_e^\vdash(r') \text{ für } r < r'. \tag{3.8}$$

Wir nehmen im Folgenden an, dass w_e durch zwei Orakel 3.1.1 und 3.1.2 (siehe Abschnitt 2.4.4) gegeben ist. Dem Orakel 3.1.1 übergibt man einen Punkt und bekommt die Kosten in diesem Punkt.

Orakel 3.1.1 Kosten

Eingabe: Punkt $r \in \mathbb{R}$.
Ausgabe: Kosten $c_e^{\vdash}(r)$ und $c_e^{\dashv}(r)$.

Das zweite Orakel 3.1.2 liefert nach Eingabe einer Steigung p einen Punkt r mit $c_e^{\dashv}(r) \le p \le c_e^{\vdash}(r)$.

Orakel 3.1.2 Punkt

Eingabe: Steigung $p \in \mathbb{R}$.
Ausgabe: Punkt r mit $c_e^{\dashv}(r) \le p \le c_e^{\vdash}(r)$.

Jede Zirkulation $x \in L$ kann als konische Kombination $\alpha_1\xi^1 + \ldots + \alpha_k\xi^k$ mit $k \le |E|$ von Kreisinzidenzvektoren $\xi^1, \ldots, \xi^k \in L$ dargestellt werden. Später werden wir dies in Lemma 3.3.4 explizit beweisen. Für einen $(0, \pm 1)$-Vektor $\xi \in \mathbb{R}^E$ führen wir das Paar $F_\xi = (A_\xi, B_\xi)$ mit $A_\xi = \{e \mid \xi_i = 1\}$ und $B_\xi = \{e \mid \xi_e = -1\}$ ein. Ist ξ ein Kreis, dann bezeichnen wir F_ξ auch als Kreis. Die Menge der Kreise bezeichnen wir mit $\mathcal{F}$. Die Größe von F ist $|F| = |A| + |B|$. Zudem verwenden wir äquivalent $c^{\dashv}(A) := \sum_{e\in A} c_e^{\dashv}(x_e)$ und $c^{\vdash}(B) := \sum_{e\in B} c_e^{\vdash}(x_e)$.

Wenn wir einen Kreis $F = (A, B)$ und eine Zirkulation $x \in L$ haben und wollen x entlang F augmentieren, dann vergrößern wir den Fluss x entlang A um $\varepsilon > 0$ und verkleinern den Fluss entlang B um ε. Die Änderung der Zielfunktion pro Einheit ε ist $c^{\vdash}(A) - c^{\dashv}(B)$. Die Kosten von F bei x bezeichnen wir mit $c(x, F)$. F ist ein negativer Kreis falls $c(x, F) < 0$. Die durchschnittlichen Kosten von F bei x sind $\bar{c}(x, F) = c(x, F)/|F|$. Einen Kreis F mit minimalen Kosten $\bar{c}(x, F)$ bezeichnen wir als *Min-Mean-Circuit*. Anhand $\bar{c}(x, F)$ definieren wir

$$\lambda(x) = \max\{0, \min_{F\in\mathcal{F}} \bar{c}(x, F)\}. \tag{3.9}$$

Sei x eine Zirkulation und y eine Kozirkulation. Dann bezeichnen wir $c_e^{\vdash}(x_e) - y_e$ als *positive reduzierte Kosten* und $y_e - c_e^{\dashv}(x_e)$ als *negative reduzierte Kosten*. Das nächste Lemma von Karzanov [22] zeigt, dass $-\lambda(x)$ die größte untere Schranke der reduzierten Kosten für $y \in L^\perp$ ist.

Lemma 3.1.1. *$\lambda \in \mathbb{R}$ ist eine obere Schranke von $\lambda(x)$ genau dann, wenn ein $y \in L^\perp$ existiert und für die reduzierten Kosten*

$$c_e^{\vdash}(x_e) - y_e \ge -\lambda \quad \text{und} \quad y_e - c_e^{\dashv}(x_e) \ge -\lambda \quad \forall e \in E \tag{3.10}$$

gilt. Falls $\lambda = \lambda(x) > 0$, $F = (A, B)$ ein Min-Mean-Circuit ist und $y \in L^\perp$ (3.10) erfüllt, gilt zudem $c_e^{\vdash}(x_e) - y_e = -\lambda$ für alle $e \in A$ und $y_e - c_e^{\dashv}(x_e) = -\lambda$ für alle $e \in B$ (Beweis Karzanov [22]).

Lemma 3.1.1 liefert uns ein Optimalitätskriterium für eine Zirkulation. Sei $x^* \in L$ eine Zirkulation ohne negative Kreise (siehe Satz 3.3.1), d.h. $\lambda(x^*) = 0$. Zusammen mit (3.10) folgt

$$c_e^{\dashv}(x_e^*) \leq y_e \leq c_e^{\vdash}(x_e^*) \quad \forall e \in E \tag{3.11}$$

und $w(x) \geq w(x^*)$ für alle $x \in L$.

3.2. Min-Mean-Circuit-Canceling Methode

Im Abschnitt 3.1 haben wir beschrieben, was wir unter einem Min-Mean-Circuit verstehen. Hier wollen wir nun ein Verfahren aufzeigen, welches sukzessive negative Min-Mean-Circuits bestimmt und entlang dieser Kreise den Matroidfluss entsprechend augmentiert. Wir bezeichnen dieses Vorgehen als *Min-Mean-Circuit-Canceling Methode*. Dabei gehen wir davon aus, dass wir ein weiteres Orakel 3.2.1 besitzen. Diesem Orakel übergeben wir einen Matroidfluss und bekommen einen entsprechenden Min-Mean-Circuit. Wie man dieses Orakel durch einen polynomialen Algorithmus realisieren kann, zeigen wir in Abschnitt 3.5.

Orakel 3.2.1 Min-Mean-Circuit

Eingabe: $x \in L$.
Ausgabe: Berechne $\lambda = \lambda(x)$ und falls $\lambda > 0$, finde einen Min-Mean-Circuit $F = (A, B)$ in x und $y \in L^{\perp}$ damit (3.10) erfüllt ist.

Das Verfahren startet mit einem zulässigen $x \in L$ und berechnet $\lambda = \lambda(x)$. Falls $\lambda = 0$ ist kein negativer Kreis vorhanden und damit x nach Satz 3.3.1 optimal. Andernfalls finde F und y anhand Orakel 3.2.1. Für jedes Element $e \in F$ verwenden wir Ausgabe 2 von Orakel 3.1.1 für w_e um die lokale Schrittweite

$$\varepsilon_e = \Delta \text{ mit } \begin{cases} c_e^{\dashv}(x_e + \Delta) \leq c_e^{\vdash}(x_e) + \lambda = y_e \leq c_e^{\vdash}(x_e + \Delta), & \text{falls } e \in A \\ c_e^{\dashv}(x_e - \Delta) \leq c_e^{\dashv}(x_e) - \lambda = y_e \leq c_e^{\vdash}(x_e + \Delta), & \text{falls } e \in B \end{cases} \tag{3.12}$$

zu bestimmen. Die Gesamtschrittweite ergibt sich mit $\varepsilon = \min\{\varepsilon_e \mid e \in A \cup B\}$. Wir erhöhen ε so lange, bis reduzierte Kosten in F nicht negativ werden. Dann augmentieren wir den Fluss entlang F um ε durch

$$x'_e = \begin{cases} x_e + \varepsilon, & \text{für } e \in A \\ x_e - \varepsilon, & \text{für } e \in B \\ x_e, & \text{sonst.} \end{cases} \tag{3.13}$$

Anschließend setzen wir $x = x'$ und wiederholen das Vorgehen bis λ klein genug ist.

3.2.1. Konvergenz

Wir wollen hier auf die Konvergenz der oben erwähnten Min-Mean-Circuit-Canceling Methode eingehen. Der Konvergenzbeweis nutzt meist aus, dass λ nach einer gewis-

sen Anzahl von Iterationen um einen bestimmen Faktor verkleinert wurde. Entsprechende Beweise kann man in Korte/Vygen [24] Seiten 224-227 und Hochstättler/Schliep [18] Seiten 99-103 finden. Das folgende Lemma stammt von Karzanov/McCormick [22].

Lemma 3.2.1. *Nach $\rho^\perp$ Iterationen gilt $\lambda(x)^{i+\rho^\perp} \leq \lambda^i(1-\frac{1}{2\phi})$ (Beweis Karzanov/McCormick [22]).*

Seien β und ε im Folgenden zwei Konstanten. β sei die Abweichung einer anfänglichen Lösung für das Min-Cost-Flow Problem von der Optimallösung und ε entspricht der maximalen Abweichung der Finallösung von der Optimallösung. Mit Lemma 3.2.1 sind wir nun in der Lage die Anzahl der Iterationen der Min-Mean-Circuit-Canceling Methode, in Abhängigkeit von β und ε, abzuschätzen.

Satz 3.2.1. *Seien $\varepsilon, \beta \in \mathbb{R}, 0 < \varepsilon < \beta, \lambda(x)$ ein anfänglicher Wert für $x \in L$ und $\lambda(x) \leq \beta$. Nach $O(\rho^\perp \phi \lceil \log(\beta/\varepsilon) \rceil)$ Iterationen wird $\lambda(x)$ auf einen Wert echt kleiner als ε reduziert.*

Beweis: Mit Lemma 3.2.1 wissen wir, dass nach $\rho^\perp$ Iterationen der Wert von $\lambda(x)$ um mindestens den Faktor $(1-1/(2\phi))$ reduziert wurde. Da $\beta \cdot (e^{-1})^{\log(\beta)} = 1$ und $(1-1/(2\phi))^{2\phi} < e^{-1}$ folgt

$$\begin{aligned}
\beta\Big(\big(1-\frac{1}{2\phi}\big)^{2\phi}\Big)^{\log(\beta)} &< 1 \\
\Big(\big(1-\frac{1}{2\phi}\big)^{2\phi}\Big)^{\log(\beta)} &< \frac{1}{\beta} \\
\varepsilon\Big(\big(1-\frac{1}{2\phi}\big)^{2\phi}\Big)^{\log(\beta)} = \Big(\big(1-\frac{1}{2\phi}\big)^{2\phi}\Big)^{\log(\frac{\beta}{\varepsilon})} = \big(1-\frac{1}{2\phi}\big)^{2\phi\log(\frac{\beta}{\varepsilon})} &< \frac{\varepsilon}{\beta} \\
\big(1-\frac{1}{2\phi}\big)^{2\phi\lceil\log(\frac{\beta}{\varepsilon})\rceil} \leq \big(1-\frac{1}{2\phi}\big)^{2\phi\log(\frac{\beta}{\varepsilon})} &< \frac{\varepsilon}{\beta}.
\end{aligned}$$

Also benötigen wir insgesamt höchstens $\rho^\perp 2\phi\lceil\log(\beta/\varepsilon)\rceil$ Iterationen, um für $\lambda(x)$ einen Wert echt kleiner als ε zu erreichen. Unter Vernachlässigung von Konstanten folgt die Behauptung. □

3.2.2. Konvexe separable Kostenfunktionen

Stückweise lineare Kostenfunktion mit ganzzahligen Steigungen

Wir untersuchen nun, wieviele Iterationen die Min-Mean-Circuit-Canceling Methode benötigt, falls eine stückweise lineare konvexe separable Kostenfunktionen mit ganzzahligen Steigungen vorliegt. Dabei sei S die Anzahl der linearen Stücke, in allen w_e, mit $e \in E$ und C die größte, absolute Steigung.

Lemma 3.2.2. *Gegeben eine stückweise lineare konvexe separable Kostenfunktion mit ganzzahligen Steigungen und $x \in L$ eine Zirkulation. Falls $\lambda(x) < 1/\phi$ gilt, ist x*

optimal.

Beweis: Sei $y \in L^{\perp}$ ein dualer Vektor, für den $\lambda(x) < 1/\phi$ gilt. Anhand Lemma 3.1.1 folgt dann, dass die positiven und negativen reduzierten Kosten größer als $-1/\phi$ sind. Insbesondere gilt dies für alle Elemente im Min-Mean-Circuit $F = (A, B)$ und dies impliziert für die Kosten von F

$$c(x, F) = (c^{\vdash} - y)(A) + (y - c^{\dashv})(B) > -\frac{|F|}{\phi} \geq -1.$$

Da aber nach Korollar 2.3.2 $c(x, F)$ ganzzahlig ist, muss $c(x, F) \geq 0$ gelten. Also ist kein negativer Kreis mehr vorhanden und x ist damit, nach den Sätzen 2.3.4 und 3.3.1, optimal. □

Lemma 3.2.3. *Sei $x^0 \in L$ eine anfängliche Zirkulation und C die größte absolute Steigung von w_e mit $e \in E$, dann gilt $\lambda(x^0) \leq C$.*

Beweis: Angenommen $\lambda(x^0) > C$, dann muss es einen Min-Mean-Circuit $F = (A, B)$ mit

$$\lambda(x^0) = -\frac{c(x^0, F)}{|F|} = \frac{c^{\dashv}(B) - c^{\vdash}(A)}{|F|} > C \geq \frac{|c^{\dashv}(B)| + |c^{\vdash}(A)|}{|F|} \geq \frac{c^{\dashv}(B) - c^{\vdash}(A)}{|F|}$$

geben. Was uns auf einen Widerspruch führt. □

Mit Lemma 3.2.2 und Lemma 3.2.3 sind wir nun in der Lage die Anzahl der Iterationen der Min-Mean-Circuit-Canceling Methode abzuschätzen.

Satz 3.2.2. *Sei eine stückweise lineare konvexe separable Kostenfunktionen mit ganzzahligen Steigungen gegeben. Dann benötigt die Min-Mean-Circuit-Canceling Methode $O(\rho^{\perp}\phi\lceil\log(\phi C)\rceil)$ Iterationen.*

Beweis: Aus Lemma 3.2.3 wissen wir, dass $\lambda(x) \leq C$ für alle $x \in L$ gilt. Zudem haben wir in Lemma 3.2.2 gezeigt, dass x optimal ist wenn $\lambda(x) < 1/\phi$ gilt. Substituieren wir die beiden Schranken in Satz 3.2.1 β durch C und $1/\varepsilon$ durch ϕ, dann folgt die Behauptung. □

Stückweise lineare Kostenfunktion mit rationalen Steigungen

$\lambda(x)$ mit $x \in L$ ist ein Maß für die duale Konvergenz, da es nach Lemma 3.1.1 die größte untere Schranke der reduzierten Kosten über alle $y \in L^{\perp}$ darstellt. Für eine Laufzeitbetrachtung einer stückweisen linearen konvexen separablen Kostenfunktion mit rationalen Steigungen hilft uns $\lambda(x)$ aber nur bedingt. Deshalb führen wir ein weiteres Konvergenzmaß ein.

Definition 3.2.1. *Gegeben $x \in L$ und $y \in L^\perp$. Für jedes $e \in E$ definieren wir*

$$\langle x_e, y_e \rangle = \begin{cases} y_e - c_e^{\vdash}(x_e), & \text{falls } c_e^{\vdash}(x_e) < y_e, \\ c_e^{\dashv}(x_e) - y_e, & \text{falls } c_e^{\dashv}(x_e) > y_e, \\ 0, & \text{falls } c_e^{\dashv}(x_e) \leq y_e \leq c_e^{\vdash}(x_e) \end{cases} \tag{3.14}$$

und

$$\langle x, y \rangle = \max_{e \in E} \langle x_e, y_e \rangle. \tag{3.15}$$

Wir nennen $\langle x, y \rangle$ die Kostendistanz zwischen x und y.

$\langle x, y \rangle$ bezeichnet den Absolutwert der negativsten reduzierten Kosten für x und y. Es gilt $\langle x, y \rangle \geq 0$ und falls $\langle x, y \rangle = 0$ ist, ist x optimal, da sich (3.14) auf (3.11) reduziert.

Als erstes wollen wir zeigen, dass aus einer positiven Kostendistanz die Existenz eines Kreises mit negativen Kosten folgt. $x^* \in L$ stellt im Folgenden für uns eine optimale Zirkulation und $y^* \in L^\perp$ eine optimale Kozirkulation dar.

Lemma 3.2.4. *Sei x^* eine Optimallösung für* (3.3) *und $y^* \in L^\perp$. x^* und y^* erfüllen zusammen das Optimalitätskriterium* (3.11)*. Falls $x \in L$ eine Zirkulation mit $\delta = \langle x, y^* \rangle > 0$ ist, dann existiert ein Kreis $F = (A, B) \in \mathcal{F}$ mit $c(x, F) \leq -\delta$.*

Beweis: Sei $u \in E$ ein Element mit $\langle x_u, y_u^* \rangle = \delta > 0$. Wir dürfen $x_u \neq x_u^*$ annehmen, sonst wäre $\delta = 0$. Zudem behandeln wir den Fall $x_u < x_u^*$. Der umgekehrte Fall $x_u > x_u^*$ kann analog bewiesen werden und wird deshalb hier nicht weiter betrachtet. Sei $z = x^* - x$ die Differenz zweier Matroidflüsse, dann wissen wir mit Lemma 3.3.4, dass z als konische Kombination von Kreisen $\xi^1, \ldots, \xi^k \in L$ dargestellt werden kann. Da $z_u > 0$, gibt es ein ξ^i für $1 \leq i \leq k$ mit $\xi_u^i = 1$. Sei $F = (A, B) \in \mathcal{F}$ der ξ^i entsprechende Kreis. Da $x_u < x_u^*$ ist $u \in A$. Für die weiteren Elemente des Kreises F gilt $x_e^* > x_e$ für $e \in A$ und $x_e^* < x_e$ für $e \in B$. Anhand (3.8) und (3.11) erhalten wir die Ungleichungen $c_e^{\vdash}(x_e) \leq c_e^{\dashv}(x_e^*) \leq y_e^*$ für $e \in A$ und $c_e^{\dashv}(x_e) \geq c_e^{\vdash}(x_e^*) \geq y_e^*$ für $e \in B$. Da $\langle x_u, y_u^* \rangle = \delta > 0$, folgt zudem $\delta = y_u^* - c_u^{\vdash}(x_u)$ bzw. $c_u^{\vdash}(x_u) = y_u^* - \delta$. Letztlich erhalten wir mit

$$c(x, F) = c_u^{\vdash}(x_u) + \sum_{e \in A \setminus \{u\}} c_e^{\vdash}(x_e) - \sum_{e \in B} c_e^{\dashv}(x_e) \leq -\delta + \underbrace{y_u^* + y^*(A \setminus \{u\}) - y^*(B)}_{=0} = -\delta$$

die Behauptung. □

Wir wollen nun untersuchen welche Beziehung zwischen den beiden Maßen $\lambda(x)$ und der Kostendistanz besteht. Wir zeigen deshalb im Folgenden anhand zweier Lemmata, dass mit der Länge des größten Kreises ϕ die Abschätzung $\lambda(x) \leq \langle x, y^* \rangle \leq \phi\lambda(x)$ folgt.

Lemma 3.2.5. $\langle x, y^* \rangle \leq \phi\lambda(x)$.

Beweis: Angenommen es gilt $\langle x, y^* \rangle > \phi\lambda(x)$, dann gibt es nach Lemma 3.2.4 einen Kreis F mit $c(x, F) < -\phi\lambda(x)$ und folglich $-\lambda(x) \leq c(x, F)/|F| < -\phi\lambda(x)/|F|$. Da

aber $|F| \leq \phi$ ist, gilt $-\phi\lambda(x)/|F| \leq -\lambda(x)$, was uns auf einen Widerspruch führt. □

Lemma 3.2.6. $\lambda(x) \leq \langle x, y^* \rangle$.

Beweis: Angenommen $\lambda(x) > \langle x, y^* \rangle$ gilt. Nach Lemma 3.1.1 wissen wir, dass $\lambda(x) \leq \lambda$, falls ein $y \in L^\perp$ existiert, so dass (3.10) erfüllt ist. y^* ist ein solches y, für das $\lambda = \langle x, y^* \rangle = \max_e \left\{ \max\{y^* - c_e^{\vdash}(x_e), c_e^{\dashv}(x_e) - y^*, 0\} \right\}$ gilt und somit zum Widerspruch $\lambda(x) > \langle x, y^* \rangle = \lambda \geq \lambda(x)$ führt. □

Nach den Vorüberlegungen sind wir nun in der Lage zu beweisen wie viele Iterationen die Min-Mean-Circuit-Canceling Methode benötigt, um eine optimale Zirkulation $x^* \in L$ zu berechnen. Vorher zeigen wir noch, dass es eine direkte Verbindung zwischen der Kostenfunktion w und der Anzahl der möglichen Werte für die Kostendistanz gibt.

Lemma 3.2.7. *Sei $x \in L$ eine Zirkulation, $y^* \in L^\perp$ eine optimale Kozirkulation und w eine stückweise lineare konvexe separable Kostenfunktionen mit Anzahl S Steigungen, dann nimmt $\langle x, y^* \rangle$ höchstens $O(S)$ verschiedene Werte an.*

Beweis: Wir dürfen annehmen, dass y^* während aller Iterationen konstant bleibt (siehe Lemma 3.2.4). Da $c_e^{\vdash}(x_e)$ und $c_e^{\dashv}(x_e)$ einer der S Steigungen entspricht, folgt mit (3.14) und (3.15), dass $\langle x, y^* \rangle$ nur die Werte $y_e^* - c_e^{\vdash}(x_e), c_e^{\dashv} - y_e^*$ oder 0 für $e \in E$ annimmt. Damit kann aber $\langle x, y^* \rangle$ nur $O(S)$ verschiedene Werte annehmen. □

Satz 3.2.3. *Im Falle einer stückweise linearen konvexen separablen Kostenfunktion w mit Anzahl S Steigungen benötigt die Min-Mean-Circuit-Canceling Methode $O(S\rho^\perp\phi\lceil\log\phi\rceil)$ Iterationen um eine optimale Zirkulation $x^* \in L$ zu berechnen.*

Beweis: Indem wir in Satz 3.2.1 $\beta = 1$ und $\varepsilon = 1/\phi$ setzen, wissen wir, dass nach $O(\rho^\perp\phi\lceil\log\phi\rceil)$ Iterationen $\lambda(x)$ um einen Faktor von mehr als $1/\phi$ reduziert wird. Eine solche Sequenz von Iterationen wollen wir als *Großiteration* bezeichnen. Sei $x \in L$ eine Zirkulation vor einer Großiteration und $x' \in L$ die Zirkulation nach der Großiteration, dann gilt

$$\begin{aligned} \langle x', y^* \rangle &\leq \phi\lambda(x') && \text{(nach Lemma 3.2.5)} \\ &< \lambda(x) && \text{(Eigenschaft der Großiteration)} \\ &\leq \langle x, y^* \rangle. && \text{(nach Lemma 3.2.6)} \end{aligned}$$

Also ist $\langle x, y^* \rangle$ nach einer Großitertation echt kleiner geworden. Nach höchstens S Großiterationen hat sich $\langle x, y^* \rangle$ somit S-mal echt verkleinert. Da aber $\langle x, y^* \rangle$ höchstens $O(S)$ verschiedene Werte annehmen kann (siehe Lemma 3.2.7), gilt $\langle x, y^* \rangle = 0$ und damit ist x optimal. Insgesamt benötigen wir also $O(S\rho^\perp\phi\lceil\log\phi\rceil)$ Iterationen um eine optimale Zirkulation x^* zu berechnen. □

3.3. Negative Kreise

Wir wollen hier die Aussage von Satz 2.3.4 nochmal herausstellen und einen Beweis für reguläre Matroide angeben. Uns ist dies an dieser Stelle deshalb so wichtig, weil es sich dabei um das zentrale Resultat handelt, anhand welchem man entscheiden kann, ob ein Matroidfluss bzgl. einer Kostenfunktion optimal ist. Zuerst betrachten wir aber zwei besondere Matroide. Für beide Matroide gibt es Entsprechungen als Graphen (siehe Hochstättler/Schliep [18] Seiten 71 und 90). Wir wollen zudem beweisen, dass beide Matroide regulär sind.

Wir gehen im Folgenden davon aus, dass ein ausgezeichnetes Element $e_1 \in E$ wie in Abschnitt 2.4.3 existiert. Die Grundmenge ohne dieses ausgezeichnete Element definieren wir durch $\tilde{E} := E \setminus \{e_1\}$. Wie in Definition 2.4.20, ist ein Matroidfluss durch eine Kapazitätsfunktion $k : \tilde{E} \to \mathbb{Z}_+$ von oben und durch 0 von unten beschränkt. Weiter gibt es eine Kostenfunktionen auf den Elementen des Matroids $c : \tilde{E} \to \mathbb{Z}$.

Zur Notation merken wir drei Dinge an. Eine Tilde verwenden wir immer dann, wenn das Element e_1 gesondert betrachtet wird. M^f deutet auf ein, von einem Fluss f, induziertes Matroid hin. M_U beschreibt ein Matroid was, um eine Menge U von Elementen erweitert ist.

Definition 3.3.1. *Sei $M = (E, \mathcal{C})$ ein gerichtetes reguläres Matroid mit $F \subseteq E$. Zudem sei F^{-1} eine Menge von Elementen die aus F hervorgeht, indem jedes Element in seiner Orientierung umgekehrt wird (siehe Definition 2.4.15). $\mathcal{C}_{F^{-1}}$ ist die Menge von Kreisen über $E_{F^{-1}} = E \cup F^{-1}$. Wir nennen das Matroid $M_{F^{-1}} = (E_{F^{-1}}, \mathcal{C}_{F^{-1}})$ dass um F^{-1} erweiterte Matroid.*

Lemma 3.3.1. *Das aus einem gerichteten regulären Matroid $M = (E, \mathcal{C})$ mit $F \subseteq E$ hervorgegangene erweiterte Matroid $M_{F^{-1}} = (E_{F^{-1}}, \mathcal{C}_{F^{-1}})$ ist ein gerichtetes reguläres Matroid.*

Beweis: Da M ein gerichtetes reguläres Matroid ist, existiert eine total unimodulare Matrix A die M repräsentiert (siehe Satz 2.4.16). Wir kopieren A und mit -1 multiplizierte Spalten von A, welche F^{-1} entsprechen, in eine neue Matrix A'. Jede quadratische Untermatix H von A' enthält nun zwei entgegengesetzte Spalten, also $\det(H) = 0$ oder ist eine Untermatrix von A, bei der einige Spalten mit -1 multipliziert wurden und womit $\det(H) = \{0, \pm 1\}$ gilt. Folglich ist A' eine total unimodulare Matrix. Da A' aber eine Repräsentationsmarix von $M_{F^{-1}}$ ist, ist $M_{F^{-1}}$ ein gerichtetes reguläres Matroid. □

Wir betrachten als nächstes ein Matroid, welches durch einen Matroidfluss f (siehe Definition 2.4.20) induziert wird. Dieses Matroid wollen wir als *Residualmatroid* bezeichnen.

Definition 3.3.2. *Sei ein gerichtetes reguläres Matroid $M = (E, \mathcal{C})$ mit Fluss f und Kapazitätsfunktion $k : E \to \mathbb{Z}_+$ gegeben. Für jeden Fluss f mit $0 \leq f \leq k$ definieren*

wir

$$\begin{aligned}\tilde{E}_1^f &:= \{e \mid e \in \tilde{E}, f(e) < k(a)\},\\ \tilde{E}_2^f &:= \{e^{-1} \mid e \in \tilde{E}, f(e) > 0\} \text{ und}\\ E^f &:= \tilde{E}_1^f \cup \tilde{E}_2^f \cup \{e_1\}.\end{aligned}$$

Ferner sei $\mathcal{C}^f$ die Menge von Kreisen die durch E^f bedingt ist. Das Matroid $R^f = (E^f, \mathcal{C}^f)$ wird als Residualmatroid bezeichnet.

Den Elementen des Residualmatroids wird eine Kapazitätsfunktion $k^f : E^f \to \mathbb{Z}_+$ mit

$$k^f(e) := \begin{cases} k(e) - f(e), & \text{für } f(e) < k(e) \text{ und } e \in \tilde{E}_1^f \\ f(e^{-1}), & \text{für } 0 < f(e^{-1}) \text{ und } e \in \tilde{E}_2^f \end{cases}$$

und Kostenfunktion $c^f : E^f \to \mathbb{Z}$ mit

$$c^f(e) := \begin{cases} c(e), & \text{für } f(e) < k(e) \text{ und } e \in \tilde{E}_1^f \\ -c(e^{-1}), & \text{für } 0 < f(e^{-1}) \text{ und } e \in \tilde{E}_2^f \end{cases}$$

zugeordnet.

Lemma 3.3.2. *Sei $M = (E, \mathcal{C})$ ein gerichtetes reguläres Matroid mit Kapazitätsfunktion k und Matroidfluss f. Das entsprechende Residualmatroid $R^f = (E^f, \mathcal{C}^f)$ ist ein reguläres Matroid.*

Beweis: Wir erweiteren das Matroid M um $\tilde{E}^{-1}$ und erhalten das Matroid $M_{\tilde{E}^{-1}} = (E_{\tilde{E}^{-1}}, \mathcal{C}_{\tilde{E}^{-1}})$. Anhand von Lemma 3.3.1 wissen wir, dass $M_{\tilde{E}^{-1}}$ ein reguläres Matroid ist. Da das Residualmatroid R^f ein Minor (siehe Definition 2.4.13) von $M_{\tilde{E}^{-1}}$ ist, folgt mit Satz 2.4.15 die Behauptung. □

Nun wollen wir uns ein weiteres Matroid, das *Trägermatroid*, anschauen. An dieser Stelle weichen wir von der Definition 2.4.20 des Matroidflusses ab und erlauben rein formal auch einen negativen Fluss auf den Elementen.

Definition 3.3.3. *Sei $M = (E, \mathcal{C})$ ein reguläres Matroid und f ein Matroidfluss. Sei weiter $S^f := \{e \in E \mid f(e) > 0\} \cup \{e^{-1} \in E^{-1} \mid f(e) < 0\}$ und $\mathcal{J}^f$ die Menge von Kreisen über S, dann bezeichen wir das Matroid $T^f = (S^f, \mathcal{J}^f)$ als Trägermatroid.*

Lemma 3.3.3. *Sei f ein Matroidfluss in einem regulären Matroid, dann ist das Trägermatroid $T^f = (S^f, \mathcal{J}^f)$ ein reguläres Matroid.*

Beweis: Sei $R^f = (E^f, \mathcal{C}^f)$ das Residualmatroid bezüglich f. Das Residualmatroid ist nach Lemma 3.3.2 regulär. Das Trägermatroid $T^f = (S^f, \mathcal{J}^f)$ ist aber ein Minor von R^f. Anhand Satz 2.4.15 ist damit das Trägermatroid auch regulär. □

Das Residual- als auch der Trägermatroid werden von Matroidflüssen induziert. Damit im Folgenden keine Verwechslung der beiden Matroide auftritt, wird das Residualmatroid mit R^f und das Trägermatroid mit T^f bezeichnet.

Nun zeigen wir, dass sich ein Matroidfluss bzw. eine Zirkulation als konische Kombination von Inzidenzvektoren von gerichteten Kreisen darstellen lässt (vgl. Hochstättler/Schliep [18] Seite 90).

Lemma 3.3.4. *Seien f, f' zwei zulässige Matroidflüsse für das Min-Cost-Flow Problem in regulären Matroiden und $T^{f-f'} = (E, \mathcal{C})$ das Trägermatroid (siehe Definition 3.3.3) bezüglich $f - f'$. Dann existieren gerichtete Kreise $C_1, \ldots, C_r$ mit $r \leq |E|$ in $T^{f-f'}$ und $\lambda_1, \ldots, \lambda_r \geq 0$, so dass $\tilde{f} = |f - f'| = \sum_{i=1}^{r} \lambda_i \chi_{C_i}$.*

Beweis: Wir führen den Beweis per Induktion über die Anzahl der Elemente $|E|$ des Trägermatroids $T^{f-f'} = (E, \mathcal{C})$. Wir dürfen $|E| > 0$ annehmen, ansonsten gibt es nichts zu zeigen. Sei $\tilde{f}$ ein Matroidfluss und $e_1 \in E$ mit $\tilde{f}(e_1) > 0$. Dann gibt es ein Element $e_2 \in E$ mit $\tilde{f}(e_2) > 0$, weil $\tilde{f}$ eine Zirkulation ist. Mit dem gleichen Argument folgt aus $\tilde{f}(e_2) > 0$ die Existenz eines $e_3 \in E$ mit $\tilde{f}(e_3) > 0$. Dies können wir induktiv fortsetzen. Da aber $|E|$ endlich ist, kommen wir wieder zu e_1 zurück und finden einen Kreis C_1 im Residualmatroid $R^{f'}$. Sei $\lambda_1 = \min_{e \in C_1} \tilde{f}(e)$, dann ist $\tilde{f}' := \tilde{f} - \lambda_1 \chi_{C_1}$ auch eine Zirkulation. Zudem gilt für $T^{\tilde{f}-\lambda_1\chi_{C_1}} = (E', \mathcal{C}')$ und $T^{f-f'} = (E, \mathcal{C})$ bezüglich der Elemente die Inklusionsbeziehung $E' \subset E$. Aufgrund der Induktionsannahme, existieren dann $\lambda_2, \ldots, \lambda_r \geq 0$ mit $r \leq |E|$ und gerichtete Kreise $C_2, \ldots, C_r$, so dass $\tilde{f}' = \sum_{i=2}^{r} \lambda_i \chi_{C_i}$ gilt. □

Der folgende Satz liefert uns ein Optimalitätskriterium, wann ein zulässiger Fluss für das Min-Cost-Flow Problem, in einem regulären Matroid, bezüglich der Kostenfunktion optimal ist (vgl. Hochstättler/Schliep [18] Seite 91).

Satz 3.3.1. *Sei f ein zulässiger Matroidfluss mit Wert $f(e_1)$ für $M = (E, \mathcal{C})$ mit Kapazitätsfunktion $k : E \to \mathbb{Z}_+$ und Kostenfunktion $c : E \to \mathbb{Z}$. f hat minimale Kosten genau dann, wenn es im Residualmatroid $R^f = (E^f, \mathcal{C}^f)$ mit Kapazitätsfunktion $k^f : E^f \to \mathbb{Z}_+$ und Kostenfunktion $c^f : E^f \to \mathbb{Z}$ keinen gerichteten Kreis C mit $c^f(C) < 0$ gibt.*

Beweis: Wir zeigen zuerst die eine Richtung. Falls ein Kreis $C \in \mathcal{C}^f$ mit $c^f(C) < 0$ existiert, dann gibt es einen zulässigen Matroidfluss f' mit $c^T f' < c^T f$. Wenn es einen gerichteten Kreis C mit $c^f(C) < 0$ in $R^f = (E^f, \mathcal{C}^f)$ gibt, dann passen wir den Fluss f in $M = (E, \mathscr{C})$ entlang C anhand (2.24) mit $f' = f + \varepsilon(C)\chi_C$ an. Da

$$\begin{aligned} f'(e) &\leq f(e) + k^f(e)\chi(C) = k(e) && \text{für } e \in E \\ f'(e) &\geq f(e) - f(e^{-1}) = 0 && \text{für } e^{-1} \in E \end{aligned}$$

gilt, werden die Kapazitäten eingehalten. Zudem findet die Anpassung des Flusses entlang eines Kreises statt, damit ist f' auch eine Zirkulation. Nach Definition 2.4.22 gilt $f(e_1) = f'(e_1)$. Somit ist f' ein zulässiger Matroidfluss. Zudem verbessert sich die Zielfunktion mit $c^T f' = c^T f + \varepsilon(C) c^T \chi(C) < c^T f$.
Wir zeigen nun die andere Richtung. Seien f', f zulässige Matroidflüsse für das Min-Cost-Flow Problem in regulären Matroiden mit $f'(e_1) = f(e_1)$ und $c^T f_1 < c^T f$. Nach Lemma 3.3.4 können wir $|f' - f|$ als konische Kombination von Kreisen des Trägermatroids $T^f = (S^f, \mathcal{J}^f)$ schreiben. Da aber nach Voraussetzung $c^T f_1 + c^T f = c^t(f_1 - f) < 0$

gegeben ist, muss mindestens einer dieser Kreise einem Kreis $C \in \mathcal{C}^f$ im Residualmatroid $R^f = (E^f, \mathcal{C}^f)$ mit $c^f(C) < 0$ entsprechen. □

3.4. Maximaler Matroidfluss

Die Min-Mean-Circuit-Canceling Methode setzt einen zulässigen Matroidfluss voraus. Wie man einen solchen Matroidfluss bestimmt, wollen wir in diesem Abschnitt zeigen. Wir entwickeln einen Algorithmus, welcher einen maximalen Matroidfluss (siehe Definition 2.4.20) in einem gerichteten regulären Matroid berechnet. Das M ein reguläres Matroid mit einer entsprechenden Partitionierung (siehe Definition 2.4.17) und Signierung (siehe Abschnitt 2.4.3) ist, wird dadurch Rechnung getragen, dass wir eine total unimodulare $\{0, 1, -1\}$-Repräsentationsmatrix als gegeben voraussetzen.

Unser Algorithmus geht zur Bestimmung eines maximalen Flusses ähnlich vor wie der Algorithmus von *Edmonds-Karp* (siehe Abschnitt 2.3.4). Es werden sukzessive lineare Programme (3.16) gelöst, um einen kürzesten augmentierenden Kreis $C \in \mathcal{C}^+_{e_1}$ zu berechnen. Entlang C wird dann der Fluss entsprechend angepasst.

$$\begin{array}{rrcl} \min & \sum_{i=1}^{|E|} x_i & & \\ & Bx & = & 0 \\ & x_1 & = & 1 \\ & x & \geq & 0 \end{array} \tag{3.16}$$

Durch die Nebenbedingung $x_1 = 1$ wird sichergestellt, dass $e_1 \in C$ gilt. Unter einem kürzesten Kreis verstehen wir einen Kreis mit einer minimalen Anzahl von Elementen aus E. Der Wert der Zielfunktion $\sum_{i=1}^{|E|} x_i$ entspricht der Anzahl der Elemente von C. B stellt die Repräsentationsmatrix für diese Iteration dar. In der ersten Iteration entspricht B der Repräsentationsmatrix von M. Entlang C wird der Fluss durch (2.24) über (2.26) angepasst. Der neue Fluss bestimmt das Residualmatroid (siehe Definiton 3.3.2). Die Repräsentationsmatrix des Residualmatroids wird dann in der nächsten Iteration anstelle von B zum Lösen von 3.16 verwendet. Anhand Satz 2.3.5 folgt direkt, dass (3.16) eine ganzzahlige Lösung mit $x \in \{0, 1\}^{|E|}$ besitzt. Dabei entsprechen die 1-Einträge den Elementen des kürzesten Kreises C.

Mit den oben ausgeführten Vorüberlegungen sind wir nun in der Lage einen Algorithmus anzugeben, welcher einen maximalen Fluss in einem gerichteten regulären Matroid bestimmt.

Algorithmus 3.4.1 Maximaler Matroidfluss

Eingabe: Matroid $M = (E, \mathcal{C})$ mit total unimodularer Repräsentationsmatrix B, Kapazitätsfunktion $k : \tilde{E} \to \mathbb{Z}_+$ und ausgezeichnetes Element e_1.
Ausgabe: Ein maximaler Matroidfluss f in M mit Wert $f(e_1)$.

1: Berechne durch (3.16) einen kürzesten augmentierenden Kreis C. Falls unzulässig STOP.
2: Bestimme $\varepsilon(C)$ durch (2.24) und passe $f' := f + \varepsilon(C)\chi^C$ (2.26) an.
3: Erstelle Residualmatroid nach Definition 3.3.2 abhängig von f' und gehe zu 1.

Der Algorithmus 3.4.1 berechnet in Zeile 1 kürzeste augmentierende Kreise. Falls keine Lösung existiert, ist kein augmentierender Kreis mehr vorhanden und Algorithmus 3.4.1 terminiert mit einem maximalen Matroidfluss. Möglicherweise ist dies auch der Nullfluss. Entlang des, in Zeile 1, bestimmten kürzesten Kreises wird der Fluss in Zeile 2 entsprechend augmentiert. Abhängig von dem dadurch entstandenen Fluss wird das entsprechende Residualmatroid in Zeile 3 bestimmt. Anschließend beginnt in Zeile 1 eine neue Iteration.

Satz 3.4.1. *Algorithmus 3.4.1 terminiert und berechnet einen ganzzahligen maximalen Matroidfluss.*

Beweis: Dazu führen wir einen Induktionsbeweis über die Anzahl der augmentierenden Kreise. Wir beginnen mit dem Nullfluss. Haben wir einen ganzzahligen Matroidfluss, der nicht maximal ist, dann bestimmen wir in Zeile 1 einen augmentierenden Kreis C und abhängig davon in Zeile 2 daraus $\varepsilon(C)$. Da die Kapazitätsfunktion nach Voraussetzung ganzzahlig ist, ist auch $\varepsilon(C)$ und damit der Fluss auf jedem Element ganzzahlig. Ein ganzzahliger Fluss hat wieder eine ganzzahlige Kapazitätsfunktion k^f in sukzessiven Iterationen zur Folge. Bei jeder Augmentierung erhöhen wir den Matroidfluss um mindestens den Wert 1. Da der maximale Matroidfluss durch Lemma 2.4.4 beschränkt ist, folgen die Behauptungen aus Lemma 2.4.5. □

Wir wollen als nächstes untersuchen welche Laufzeit Algorithmus 3.4.1 hat. Dazu verwenden wir zwei Funktionen wie in Schrijver [32] auf Seite 153. Sei $\mathcal{M}$ die Menge von gerichteten regulären Matroiden. $\mu : \mathcal{M} \to \mathbb{Z}_+$ ist eine Funktion, welche die Länge eines kürzesten augmentierenden Kreises eines gerichteten regulären Matroids M liefert. Die zweite Funktion $\alpha : \mathcal{M} \to 2^E$ bestimmt die Menge von Elementen, die zu einem kürzesten augmentierenden Kreis eines gerichteten regulären Matroids M gehören. Abgeleitet von den beiden Funktionen μ und α verwenden wir weitergehend Funktionen $\tilde{\mu}(M) := \mu(M) - 1$ und $\tilde{\alpha}(M) := \alpha(M) \setminus \{e_1\}$, die das ausgezeichnete Element e_1 geeignet berücksichtigen.

Lemma 3.4.1. *Sei $M = (E, \mathcal{C})$ ein gerichtetes reguläres Matroid und K ein kürzester augmentierender Kreis, dann gilt für das durch die Elemente $\tilde{\alpha}(K)^{-1}$ erweiterte Matroid $\tilde{\mu}(M) = \tilde{\mu}(M_{\tilde{\alpha}(K)^{-1}})$ und $\tilde{\alpha}(M) = \tilde{\alpha}(M_{\tilde{\alpha}(K)^{-1}})$.*

Beweis: Zunächst gilt, dass $M_{\tilde{\alpha}(K)^{-1}}$ ein Minor des regulären Matroids $M_{\tilde{E}^{-1}}$ ist. Mit Satz 2.4.15 ist somit $M_{\tilde{\alpha}(K)^{-1}}$ auch ein reguläres Matroid. Wir zeigen, dass $\tilde{\mu}(M)$ und $\tilde{\alpha}(M)$ unter Hinzunahme eines Elementes e^{-1} mit $e \in \tilde{\alpha}(M)$ invariant sind. Angenommen dies wäre nicht der Fall, dann gäbe es einen kürzesten augmentierenden Kreis C' der e^{-1} mit weniger als $\tilde{\mu}(M)$ traversiert. Da $e \in \tilde{\alpha}(M)$, gibt es einen kürzesten augmentierenden Kreis C'' der e mit $\tilde{\mu}(M)$ traversiert. Damit muss es aber einen kürzesten augmentierenden Kreis $C' \cup C'' \setminus \{e, e^{-1}\}$ mit weniger als $\tilde{\mu}(M)$ geben. Damit wäre aber $\tilde{\mu}(M)$ nicht minimal gewesen. Widerspruch! □

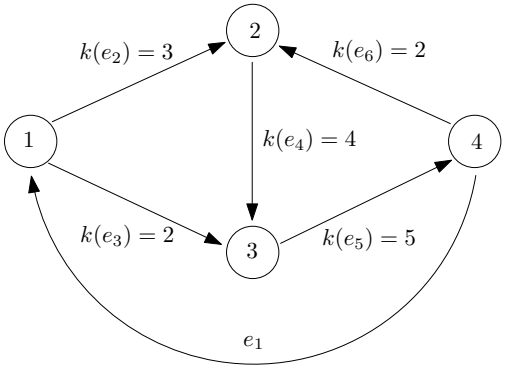
Wir betrachten nun die Laufzeit von Algorithmus 3.4.1.

Satz 3.4.2. *Algorithmus 3.4.1 bestimmt einen maximalen Matroidfluss in einem gerichteten regulären Matroid in* $O(n^7 + n^6)$.

Beweis: Wir zeigen zuerst die Anzahl der Iterationen in Abhängigkeit der Anzahl der Elemente n. Wenn wir einen Matroidfluss f entlang eines kürzesten augmentierenden Kreises C in M^f zu f' anpassen, dann ist $M^{f'}$ ein Minor von $M^f_{\tilde{\alpha}(M^f)^{-1}}$. Mit Lemma 3.4.1 folgt $\tilde{\mu}(M^{f'}) \geq \tilde{\mu}(M^f_{\tilde{\alpha}(M^f)^{-1}}) = \tilde{\mu}(M^f)$. Weiter folgt mit Lemma 3.4.1, falls $\tilde{\mu}(M^{f'}) = \tilde{\mu}(M^f)$, dass $\tilde{\alpha}(D^{f'}) \subseteq \tilde{\alpha}(M^f_{\tilde{\alpha}(M^f)^{-1}}) = \tilde{\alpha}(M^f)$. Da mindestens ein Element von C zu M^f aber nicht zu $M^{f'}$ gehört, erhalten wir eine echte Inklusion. Weil sich $\tilde{\alpha}(M^f)$ höchstens n mal erhöht und $\tilde{\alpha}(M^f)$ höchstens n mal verkleinert falls sich $\tilde{\mu}(M^f)$ nicht ändert, gibt es maximal n^2 Iterationen. In Zeile 1 wird in jeder Iteration ein lineares Programm (3.16) berechnet, um einen kürzesten Kreis zu bestimmen. Aus Korollar 2.2.1 wissen wir, dass (3.16) in $O(n^5 + n^4)$ berechnet werden kann. Somit benötigt Algorithmus 3.4.1 $O(n^7 + n^6)$ Iterationen, um einen maximalen Matroidfluss zu bestimmen. □

Beispiel 3.4.1. Wir wenden Algorithmus 3.4.1 auf das graphische Matroid $M(G_1)$ in Abbildung 3.2) an. Die den Elementen entsprechenden Kapazitäten sind den Kanten zugeordnet. Das ausgezeichnete Element ist e_1.

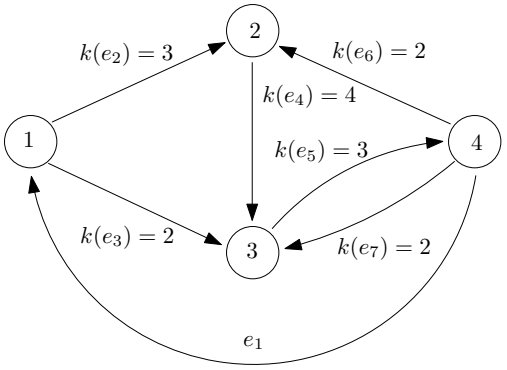
1. Iteration: Durch elementare Umformungen der Inzidenzmatrix von G_1 erhalten wir eine Repräsentationsmatrix B_1 mit $M(G_1) \cong M[B_1]$. Lösen wir das lineare Programm über B_1, dann erhalten wir als Lösung $x = (1,0,1,0,1,0)^T$ und damit $\{e_1, e_3, e_5\}$ als kürzesten augmentierenden Kreis C. $\varepsilon(C)$ entlang dieses Kreises ist 2. Damit berechnen wir auf den Elementen von $M(G_1)$ den Fluss $f_1 = (2,0,2,0,2,0)^T$.



$$B_1 = \begin{array}{c} \begin{matrix} e_1 & e_3 & e_4 & e_2 & e_5 & e_6 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & -1 & 0 & -1 \end{pmatrix} \end{array}$$

Abb. 3.2.: Graphisches Matroid $M(G_1)$.

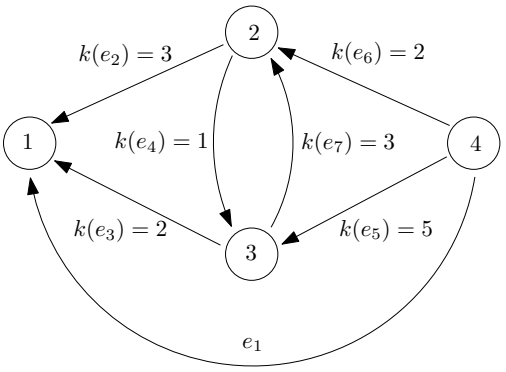
2. Iteration: Anhand von f_1 generieren wir das Residualmatroid mit den entsprechenden Kapazitäten. Dem Residualmatroid liegt der Graph G_2 (siehe Abb. 3.3) mit Repräsentationsmatrix B_2 zugrunde. Wir erhalten nun als Lösung $x = (1,1,0,1,1,0,0)^T$ und damit $\{e_1, e_2, e_4, e_5\}$ als kürzesten augmentierenden Kreis C. Die minimale Kapazität der Kreislemente $\varepsilon(C)$ ist 3 und somit berechnen wir den Fluss $f_2 = (5,3,2,3,5,0)^T$.



$$B_2 = \begin{array}{c} \begin{matrix} e_1 & e_3 & e_4 & e_2 & e_5 & e_6 & e_7 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 1 & -1 & -1 \\ 0 & 0 & 1 & -1 & 0 & -1 & 0 \end{pmatrix} \end{array}$$

Abb. 3.3.: Residualmatroid $M(G_2)$.

3. Iteration: In der dritten und letzten Iteration leiten wir ein Residualmatroid mit Kapazitäten vom aktuellen Fluss f_2 ab. Der entsprechende Graph G_3 ist in Abb. 3.4 dargestellt. Lösen wir das lineare Programm mit B_3, dann ist dieses unzulässig, da kein augmentierender Kreis mit e_1 mehr existiert. Somit erhalten wir auf den Elementen von $M(G_1)$ den Fluss f_2 mit Wert $f_2(e_1) = 5$.



$$B_3 = \begin{array}{c} \begin{matrix} e_1 & e_3 & e_4 & e_2 & e_5 & e_6 & e_7 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & -1 \end{pmatrix} \end{array}$$

Abb. 3.4.: Residualmatroid $M(G_3)$.

3.5. Bestimmung eines Min-Mean-Circuit

Wir werden in diesem Abschnitt einen Algorithmus herleiten, welcher in einem Residualmatroid $M^f = (E^f, \mathcal{C}^f)$ (siehe Definition 3.3.2) das *Min-Mean-Circuit Problem* löst. Der Algorithmus basiert auf einer Veröffentlichung von Karzanov [21].

Sei L ein linearer Unterraum von $\mathbb{R}^E$, gegeben durch $Bx = 0$. Dabei ist B eine total unimodulare $(m \times n)$-Repräsentationsmatrix mit $n = |E|$. x steht für einen Kreisinzidenzvektor. Weiter sei ein Kegel $\mathcal{K} = L \cap \mathbb{R}^E_+ = \{x \in L \mid x \geq 0\}$ und eine Kostenfunktion $c : E \to \mathbb{R}$ gegeben. $|x|$ bezeichne die ℓ_1-Norm $\sum_{e \in E} |x_e|$. Für $x \in \mathcal{K} \setminus \{0\}$ ist der durchschnittliche Minimalwert durch $m_c(x) = cx/|x|$ gegeben. Ziel ist es ein x zu finden, welches $m_c(x)$ minimiert:

$$m^* := \min_{x \in \mathcal{K} \setminus \{0\}} m_c(x).$$

Im Folgenden unterscheiden wir nicht zwischen reellwertigen Funktionen auf E und dem linearen Raum $\mathbb{R}^{|E|}$, dessen Koordinaten den Elementen von E entsprechen. Wir definieren Mengen $S(c) := \{e \mid c(e) \neq 0\}, S^+(c) := \{e \mid c(e) > 0\}, S^-(c) := \{e \mid c(e) < 0\}$ und $S^0(c) := \{e \mid c(e) = 0\}$ auf der Kostenfunktion. Weiter wollen wir durch c^+ bzw. c^- ausdrücken, dass wir Vektoren mit den Einträgen $\max\{c(e), 0\}$ bzw. $\min\{c(e), 0\}$ betrachten. Wir nehmen an, dass jedes $e \in E$ im *Support* $S(c)$ von mindestens einem Kreis in $\mathcal{K}$ enthalten ist:

$$\bigcup_{x \in \mathcal{K} \setminus \{0\}} S(x) = E. \tag{3.17}$$

Der folgende Algorithmus besitzt zwei Möglichkeiten die Kostenfunktion anzupassen. Die erste Operation addiert zur Kostenfunktion einen Vektor $y \in L^\perp$, aus dem orthogonalen Komplement von L durch

$$c' := c + y. \tag{3.18}$$

Dabei verändert sich der Wert von m^* nicht. Die zweite Operationen addiert zur Kostenfunktionen einen Skalar

$$c := c + a \tag{3.19}$$

mit $a \in \mathbb{R}$ und $a := -cx/|x|$. Wir wollen diese Operation als *Shift* bezeichnen. Die Shift-Operation vergrößert m^* auf $m^* + a$. In jeder Iteration des Algorithmus wird ein Element $r \in S^-(c)$ gewählt und das folgende Paar dualer linearer Programme, wie noch gezeigt wird, gelöst:

$$\begin{array}{lrcl} \min & c^+x & & \\ \text{u.d.N.} & x & \in & \mathcal{K} \\ & x(r) & = & 1, \end{array} \tag{3.20}$$

$$\begin{array}{lrcl} \max & y(r) & & \\ \text{u.d.N.} & y & \in & L^\perp \\ & c^+(e) + y(e) & \geq & 0, \quad \text{für } e \in E \setminus \{r\}. \end{array} \tag{3.21}$$

$y = 0$ ist eine zulässige Lösung für (3.21) und (3.20) hat eine zulässige Lösung wegen (3.17). Da die Zielfunktion von (3.20) durch Null von unten beschränkt ist, besitzt (3.20) eine Optimallösung. Folglich existiert nach Satz 2.2.1 auch eine Optimallösung für (3.21). Mit Satz 2.2.2 folgt für $e \in E \setminus \{r\}$

$$c^+(e) + y(e) > 0 \Rightarrow x(e) = 0.$$

Daraus können wir für $S(x) \setminus \{r\}$

$$\begin{array}{lcll} c'(e) & = & 0 & \text{falls } c(e) \geq 0 \text{ und} \\ c'(e) & = & c(e) & \text{falls } c(e) < 0 \end{array} \tag{3.22}$$

folgern. Falls $c'(r) \geq 0$, aktualisieren wir c durch $c := c'$. Für den Fall $c'(r) < 0$, berechnen wir a, führen (3.19) mit $c := (c')^a$ durch und merken uns den mit (3.20) berechneten Inzidenzvektor $x^* := x$. Dabei gilt wegen (3.22) $0 < a$. Dann beginnt die nächste Iteration wenn $S^-(c) \neq \varnothing$. Falls in dieser Iteration $c(r) < 0$ für ein r gilt, welches schon in der letzten Iteration gewählt wurde, muss r auch in dieser Iteration wieder selektiert werden.

Wir wollen uns nun anschauen, wie sich die Dualität der linearen Programme (3.20) und (3.21) genau gestaltet. Wir ersetzen dabei die Mengeschreibweisen $x \in \mathcal{K}$ und $y \in L^\perp$ durch einen Ausdruck mit einer total unimodularen Repräsentationsmatrix B eines Matroids M. (3.20) lässt sich damit auch in der folgenden Form schreiben:

$$\begin{array}{lrcl} \min & c^+x & & \\ \text{u.d.N} & Bx & = & 0 \\ & x(r) & = & 1 \\ & x & \geq & 0. \end{array} \tag{3.23}$$

Das dazu duale Programm ist

$$\begin{array}{lrcl} \max & y(r) & & \\ \text{u.d.N.} & \gamma B + y & = & 0 \\ & c^+(e) + y(e) & \geq & 0, \quad \text{für } e \in E \setminus \{r\}. \end{array} \tag{3.24}$$

Wir wollen die Dualitätsbeziehung zwischen (3.23) und (3.24) formal herleiten und definieren dazu eine neue Matrix B'. Diese besteht aus B und einer zusätzlichen

Zeile, die nur in der Spalte r eine 1 enthält und damit der Nebenbedingung $x(r) = 1$ aus (3.23) Rechnung trägt. Zudem führen wir Dualvariablen γ bzw. σ für die Nebenbedingungen $Bx = 0$ bzw. $x(r) = 1$ ein. Dualisieren wir (3.23), dann erhalten wir

$$\begin{array}{lll} \max & \gamma 0 + \sigma & \\ \text{u.d.N.} & (\gamma, \sigma)B' & \leq \quad c^+. \end{array}$$

Dieses lineare Programm können wir mit Schlupfvariablen s zu

$$\begin{array}{ll} \max & \sigma \\ \text{u.d.N.} & (\gamma, \sigma)B' - c^+ + s = 0 \\ & s \geq 0 \end{array}$$

umformen. Da $x(r) = 1$ ist, ist in der Optimallösung die entsprechende Nebenbedingung im dualen Programm durch Satz 2.2.2 mit Gleichheit zu $c^+(r) = 0$ erfüllt. Damit gilt auch $s(r) = 0$. Wir definieren Variablen $y := s - c^+$. Mit $c^+(r) = 0$ und $s(r) = 0$ folgt $y(r) = 0$. Somit erhalten wir

$$\begin{array}{lll} \max & \sigma & \\ \text{u.d.N.} & (\gamma, \sigma)B' + y & = \quad 0. \\ & c^+(e) + y(e) & \geq \quad 0, \quad \text{für } e \in E \setminus \{r\}. \end{array}$$

Letztlich substituieren wir σ durch $y(r)$ und erhalten (3.24). Wenn wir eine Optimallösung für (3.23) bestimmt haben, dann brauchen wir (3.24) nicht mehr explizit zu lösen. Da beim Lösen von (3.23) die inverse Basismatrix der Optimallösung bestimmt wurde, liefert uns der Beweis von Satz 2.2.3 eine Methode, wie wir die Duallösung y einfach bestimmen können.

Durch (3.23) wird also ein kürzester gerichteter Kreis in einem regulären Matroid berechnet, der Element r enthält. Dabei repräsentiert B nicht das Matroid, dass dem Algorithmus zur Berechnung übergeben wird. Dem Algorithmus wird ein Matroid übergeben, welches in allen Elementen in der Orientierung umgekehrt ist (siehe Definition 2.4.15). Die übergebene Repräsentationsmatrix ist also $-B$. Die Bestimmung eines kürzesten gerichteten Kreises liefert einem damit im Dualen einen Vektor y, den wir als Potentialdifferenzen (siehe Abschnitt 2.3.2) auffassen können.

Mit all den Vorüberlegungen sind wir nun in der Lage den Algorithmus von Karzanov, übertragen auf reguläre Matroide, explizit in Algorithmus 3.5.1 anzugeben.

Algorithmus 3.5.1 Min-Mean-Circuit

Eingabe: Residualmatroid $M^f = (E^f, \mathcal{C}^f)$ mit Kostenfunktion $c : E \to \mathbb{R}$.
Ausgabe: Ein Min-Mean-Circuit C, oder die Aussage, dass kein solcher Kreis existiert.

```
1:  x* = 0
2:  r̃ = ∅
3:  while S⁻(c) ≠ ∅ do
4:      if S⁻(c) ∩ r̃ ≠ ∅ then
5:          r = r̃
6:      else
7:          Wähle r ∈ S⁻(c) beliebig.
8:      end if
9:      Löse lineares Programm (3.20) um x ∈ K zu erhalten.
10:     Bestimme anhand der Optimallösung von (3.20) die Duallösung y ∈ L⊥.
11:     c' = c + y
12:     if c'(r) ≥ 0 then
13:         c = c'
14:     else
15:         Bestimme a := −c'x/|x| und setze c' = c' + a.
16:         Speichere gefundenen Kreisinzidenzvektor x* = x.
17:         c = c'
18:     end if
19:     r̃ = r
20: end while
21: if x* ≠ 0 then
22:     C = {e ∈ E | x*(e) ≠ 0}
23: end if
```

Da wir nun die Arbeitsweise des Algorithmus beschrieben haben, soll bewiesen werden, dass der Algorithmus tatsächlich auch einen Min-Mean-Circuit berechnet.

Satz 3.5.1. *Algorithmus 3.5.1 berechnet in $M^f = (E^f, \mathcal{C}^f)$ einen Min-Mean-Circuit.*

Beweis: Zu Beginn verschieben wir c anhand einer ausreichend großen Zahl, dass $m^* \leq 0$ gilt. In jeder Iteration des Algorithmus wird c anhand (3.18) mit $y \in L^\perp$ durch $c' = c + y$ angepasst. Dabei bleibt m^* unverändert. Falls $c'(r) < 0$ wird c' zudem einer Shift-Operation (3.19) unterzogen, wobei mit (3.22) $c'(e) \leq 0$ für $e \in S(x)$ und daher $a > 0$ gilt. Diese Transformation vergrößert m^* um a. (3.22), $y(r) = c^+x \geq 0$ und $a > 0$ sind ursächlich für die monotonen Eigenschaften

$$S^-(c') \subseteq S^-(c) \text{ und } c'(e) \geq c(e) \text{ für alle } e \in S^-(c'). \tag{3.25}$$

Der Algorithmus terminiert, falls $c'(e) \geq 0$ für alle $e \in E$, d.h. $S^-(c) = \varnothing$ gilt. Für den in der letzten Iteration berechneten Kreisinzidenzvektor x^*, bei dem eine Shift-Operation stattgefunden hat, gilt $c'x^* = 0$. Nach dieser Iteration wurde die Kostenfunktion nur noch durch die Addition von Vektoren aus $L^\perp$ angepasst. Dies ändert

aber das Skalarprodukt von c' und x^* nicht, deshalb muss $c'x^* = 0$ und folglich $c'(e) = 0$ für alle $e \in S(x^*)$ sein. Zusammen mit der Voraussetzung $m^* \leq 0$ impliziert dies, dass x^* eine Optimallösung für das Min-Mean-Circuit Problem ist. □

Wir benötigen noch eine Aussage über die Laufzeit von Algorithmus 3.5.1. Dazu müssen wir etwas vorarbeiten und definieren zunächst zwei verschiedene Arten von Iterationen.

Definition 3.5.1. *Wir bezeichnen eine Iteration als essentiell, falls die Menge $S^-(c)$ echt kleiner wird. Wird die Menge $S^-(c)$ nicht echt kleiner, dann bezeichnen wir diese Iteration als nicht-essentiell.*

Aufgrund der monotonen Eigenschaft (3.25) gibt es damit höchstens $|E|$ essentielle Iterationen. Wir müssen also lediglich die Anzahl der nicht-essentiellen Iterationen näher analysieren. Wir wollen nun zwei aufeinander folgende, nicht-essentielle Iterationen näher betrachten. Dabei sind c_i, c_i', x_i, y_i und a_i für $i = 1, 2$ die entsprechenden Vektoren bzw. Werte. Der tiefgestellte Index stellt hier also nicht ein Element eines Vektors, sondern die jeweilige Iteration dar. Da die zwei Iterationen nicht-essentiell sind, gilt $S^-(c_1) = S^-(c_2)$. Folglich bleibt auch die Menge $D := S^+(c_1) \cup S^0(c_1)$ während der beiden Iterationen unverändert. Im Folgenden verwenden wir die Notation $x_1(D) := \sum_{e \in D} x(e)$.

Lemma 3.5.1. *Seien zwei aufeinander folgende nicht-essentielle Iterationen mit c_i, c_i', x_i, y_i und a_i für $i = 1, 2$ gegeben, dann gilt $x_1(D) > x_2(D)$.*

Beweis: Wir dürfen annehmen, dass während der beiden nicht-essentiellen Iteration immer das gleiche Element $r \in S^-(c)$ selektiert wird und es gilt wegen der Konsekutivität der Iterationen $c_2 = (c_1')^{a_1}$. Aufgrund der Shift-Operation gilt $c_2(e) \geq a_1$ für alle $e \in D$. (3.22) impliziert $c_2(e) = a_1$ für $e \in S(x_1) \cap D$. Aus $c_2 x_1 = c_2^+ x_1 + c_2^- x_1 = 0$ folgt $|c_2^- x_1| = c_2^+ x_1 = a_1 x_1(D)$ und zusammen mit $x_1(r) = 1$ ergibt sich $|c_2(r)| \leq a_1 x_1(D)$. Da die zweite Iteration aber eine nicht-essentielle Iteration ist, gilt andererseits $|c_2(r)| > y_2(r) = c_2^+ x_2 \geq a_1 x_2(D)$. Setzen wir die Ungleichungen zusammen, erhalten wir $a_1 x_1(D) \geq |c_2(r)| > a_1 x_2(D)$. Da aber $a_1 > 0$ ist, folgt daraus die Behauptung. □

Anhand Lemma 3.5.1 haben wir also gezeigt, dass die Größe der kürzesten Kreise während einer Sequenz von nicht-essentiellen Iterationen echt kleiner wird. Als nächstes zeigen wir wieviele Iterationen Algorithmus 3.5.1 benötigt. Dabei sei η die Anzahl und q der maximale Eintrag der Extremalstrahlen von $\mathcal{K}$. Weiter sei ϕ wie in (3.6) und ρ wie in (3.4).

Satz 3.5.2. *Die Anzahl der Iterationen von Algorithmus ist höchstens $\rho \min\{q^2\phi, \eta\}$.*

Beweis: Sei $x_i = \lambda_i^1 z_i^1 + \ldots + \lambda_i^{m(i)} z_i^{m(i)}$ und $i = 1, 2$. Dabei sind alle λ_i^j positiv und z_j^i Extremvektoren von $\mathcal{K}$ mit $z_i^j(r) = 1$ für das selektierte Element $r \in E$. Anhand Lemma 3.5.1 wissen wir, dass $x_1(D) > x_2(D)$ für zwei konsekutive, nicht-essentielle Iterationen gilt, also kann die Anzahl konsekutiver, nicht-essentieller Iterationen

nicht die Anzahl verschiedener Werte von $z_1^j(D)$ überschreiten. Falls z_2^j ein ganzzahliger Extremvektor von $\mathcal{K}$ ist, dann ist $z_2^j(D)$ eine ganze Zahl zwischen 1 und $q\phi$ und es gibt höchstens q Möglichkeiten z_2^j so zu z_1^j zu normalisieren, dass $z_1^j(r) = 1$ gilt. Da es maximal ρ viele essentielle Iterationen geben kann, braucht der Algorithmus somit höchstens $\rho \min\{q^2\phi, \eta\}$ Iterationen. □

Satz 3.5.2 versetzt uns nun in die Lage die Laufzeit von Algorithmus 3.5.1 anzugeben.

Korollar 3.5.1. *Algorithmus 3.5.1 berechnet einen Min-Mean-Circuit in* $O(\rho\phi(n^5 + n^4))$.

Beweis: Das Algorithmus 3.5.1 einen Min-Mean-Circuit in einem regulären Matroid berechnet, ergibt sich aus Satz 3.5.1. Der maximale Eintrag der Elementarvektoren von $\mathcal{K}$ ist in unserem Fall 1. Zudem können wir $\phi \leq \eta$ annehmen. Damit benötigt der Algorithmus nach Satz 3.5.2 $O(\rho\phi)$ Iterationen. Aus Korollar 2.2.1 wissen wir, dass wir für (3.23) eine Laufzeit von $O(n^5 + n^4)$ ansetzen dürfen. Insgesamt erhalten wir damit als Laufzeit $O(\rho\phi(n^5 + n^4))$. □

Abschließend wollen wir uns noch ein Beispiel von Algorithmus 3.5.1 anschauen.

Beispiel 3.5.1. Wir wollen Algorithmus 3.5.1 auf das graphische Matroid in Abbildung 3.5 anwenden. Die anfänglichen Werte der Kostenfunktion sind an den Kanten in Abbildung 3.5 dargestellt.

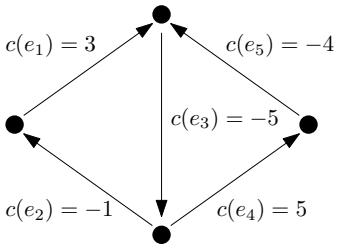


Abb. 3.5.: Graphisches Matroid $M(G)$.

1. Iteration: In der ersten Iteration wird das Element e_3 als r gewählt und mit (3.23) wird der kürzeste Kreis $\{e_1, e_2, e_3\}$ berechnet. Wir erhalten $y = (-3, 0, 3, -3, 0)$, $c' = (0, -1, -2, 2, -4)$ und $c = (1, 0, -1, 3, -3)$ durch $c = c'^a$ mit $a = -(0 - 1 - 2)/3 = 1$ (siehe Abb. 3.6). Wir merken uns $x^* = (1, 1, 1, 0, 0)$ als Kreisinzidenzvektor.

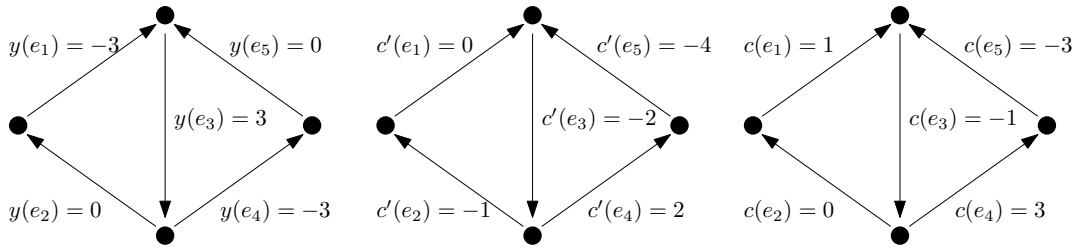


Abb. 3.6.: 1. Iteration: y, $c' = c + y$ und $c = c'^a$.

2. Iteration: Da wir in der ersten Iteration e_3 als r gewählt haben und weiterhin $c(e_3) < 0$ gilt, müssen wir e_3 wieder als r wählen. (3.23) berechnet $\{e_1, e_2, e_3\}$ als kürzesten Kreis. Wir erhalten $y = (-1, 0, 1, -1, 0)$ und $c' = (0, 0, 0, 2, -3)$ (siehe Abb. 3.7). In dieser Iteration wird keine Shift-Operation durchgeführt.

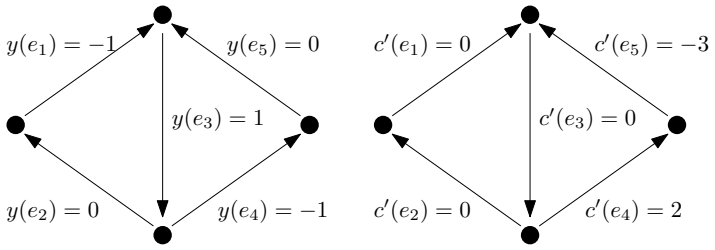


Abb. 3.7.: 2. Iteration: y und $c' = c + y$.

3. Iteration: Nun wird e_5 als r gewählt, weil dies das einzige verbliebene Element mit negativen Kosten ist. Als kürzesten Kreis, der e_5 enthält, bekommen wir $\{e_3, e_4, e_5\}$. Wir erhalten $y = (0, 0, 0, -2, 2)$, $c' = (0, 0, 0, 0, -1)$ und $c = (1/3, 1/3, 1/3, 1/3, -2/3)$ durch $c = c'^a$ mit $a = -(0 - 0 - 1)/3 = 1/3$ (siehe Abb. 3.8). Wir merken uns $x^* = (0, 0, 1, 1, 1)$ als Kreisinzidenzvektor.

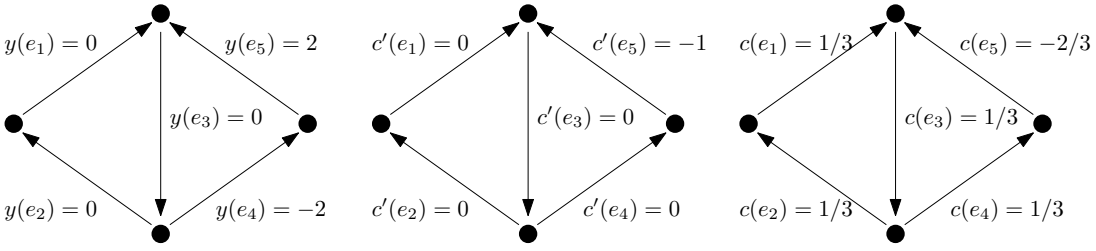


Abb. 3.8.: 3. Iteration: y, $c' = c + y$ und $c = c'^a$.

4. Iteration: Wir müssen wieder e_5 als r wählen. (3.23) berechnet $\{e_3, e_4, e_5\}$ als kürzesten Kreis. Wir erhalten $y = (-1/3, 2/3, -1/3, -1/3, 2/3)$ und $c' = (0, 1, 0, 0, 0,)$ (siehe Abb. 3.9). In dieser Iteration wird keine Shift-Operation durchgeführt. Da

nun $S^-(c) = \varnothing$ terminiert der Algorithmus. Der letzte bestimmte Inzidenzvektor $x^* = (0,0,1,1,1)$ kennzeichnet $\{e_3, e_4, e_5\}$ als Min-Mean-Circuit mit $m^* = -4/3$.

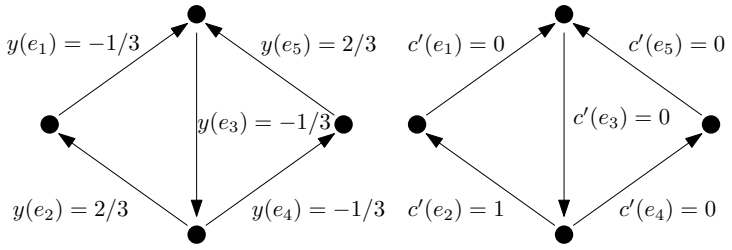


Abb. 3.9.: 4. Iteration: y und $c' = c + y$.

3.6. Algorithmus für Min-Cost-Flow in regulären Matroiden

Die, in den Abschnitten 3.1, 3.2, 3.3, 3.4 und 3.5, geleisteten Vorarbeiten versetzen uns nun in die Lage den Algorithmus 3.6.1 für das Min-Cost-Flow in regulären Matroiden anzugeben.

Algorithmus 3.6.1 Min-Cost-Flow in regulären Matroiden

Eingabe: Ein gerichtetes reguläres Matroid $M = (E, \mathcal{C})$ mit Kapazitätsfunktion $k : E \to \mathbb{Z}_+$, Kostenfunktion $c : E \to \mathbb{Q}$ und ein ausgezeichnetes Element e_1.
Ausgabe: Ein maximaler Matroidfluss $0 \le f \le k$ mit minimalen Kosten.

1: Bestimme mit Algorithmus 3.4.1 einen maximalen Matroidfluss f.
2: Bestimme mit Algorithmus 3.5.1 einen Min-Mean-Circuit C in M^f. Falls C nicht-negative Gesamtkosten hat STOP.
3: Berechne $\tau := \min_{e \in C} k^f(e)$ und augmentiere Fluss mit $f := f + \tau\chi^C$.
4: Gehe zu 2.

Lemma 3.6.1. *Algorithmus 3.6.1 berechnet einen ganzzahligen Matroidfluss.*

Beweis: Aus Satz 3.4.1 folgt, dass anfänglich ein ganzzahliger maximaler Matroidfluss bestimmt wird. Wir unterscheiden nun zwischen einer einfachen Kostenfunktion und einer konvexen separablen Kostenfunktion. Im Falle einer einfachen Kostenfunktion folgt aus der ganzzahligen Kapazitätsfunktion, dass τ immer ganzzahlig ist. Damit bleibt der Fluss im Verlauf des Algorithmus ganzzahlig. Im Falle einer konvexen separablen Kostenfunktion ergibt sich die Behauptung durch Korollar 2.3.2. □

Bevor wir nun zu den beiden zentralen Aussagen der Arbeit kommen, wollen wir in Abbildung 3.10 explizit den Zusammenhang der verwendeten Definitionen,

Lemmata, Sätze und Korollare aufzeigen und somit die Beweisführung transparent machen.

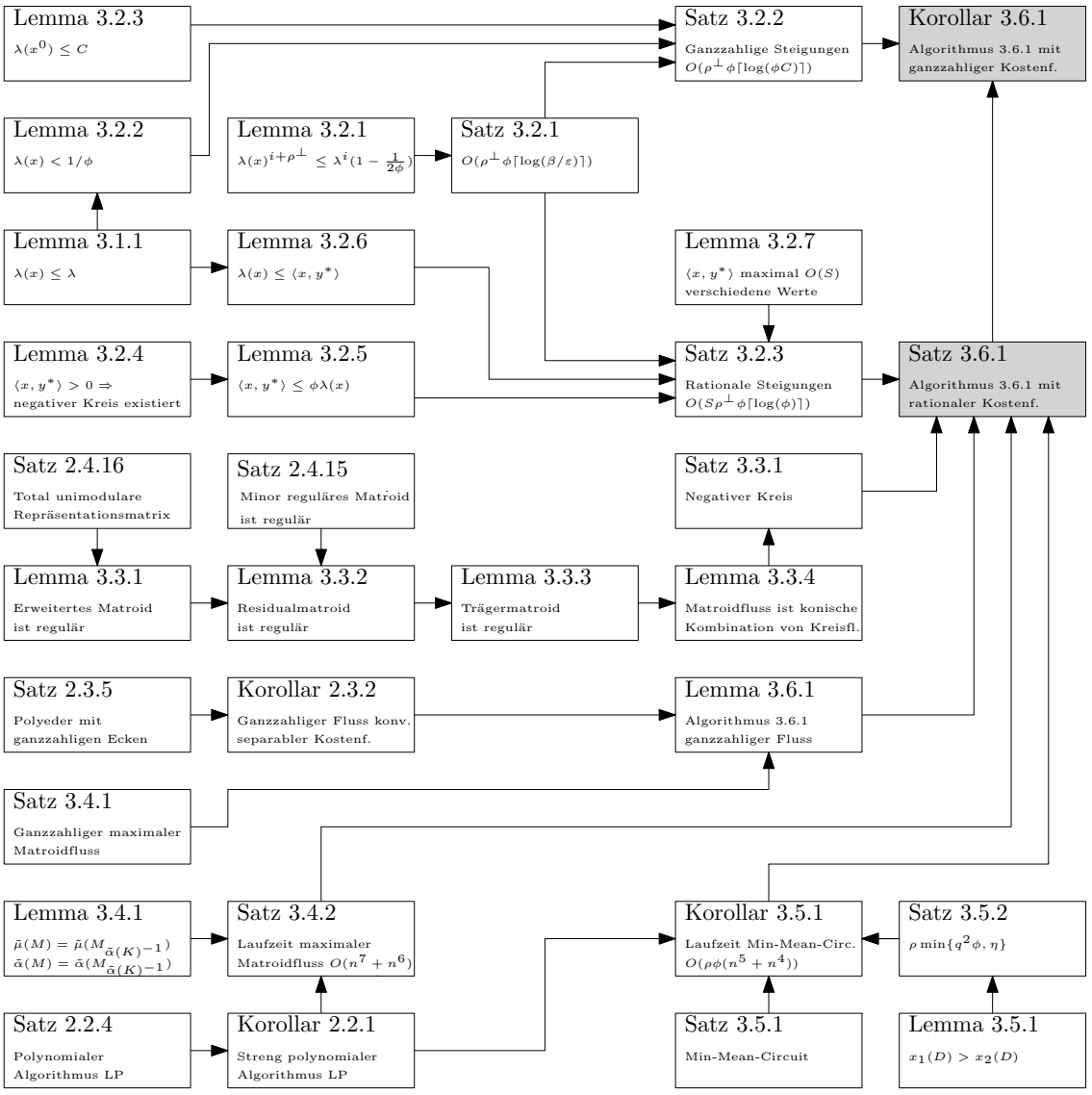


Abb. 3.10.: Beweisführung der Arbeit.

Wir wollen nun die Laufzeit von Algorithmus 3.6.1 explizit in einem Satz formulieren. Die Bedeutungen der verwendeten Bezeichner sind den Abschnitten 2.2.2, 3.1 und 3.2.2 zu entnehmen.

Satz 3.6.1. *Algorithmus 3.6.1 bestimmt einen ganzzahligen maximalen Matroidfluss mit minimalen Kosten in einem regulären Matroid in*

$$O(n^7 + n^6 + S\rho\rho^\perp\phi^2(n^5 + n^4)\lceil\log(\phi)\rceil).$$

Beweis: Die Ganzzahligkeit des Matroidflusses folgt aus Lemma 3.6.1. Der Algorithmus terminiert falls kein negativer Min-Mean-Circuit und damit auch kein negativer Kreis mehr vorhanden ist. Nach Satz 3.3.1 ist damit der Matroidfluss bezüglich der Kostenfunktion optimal. Algorithmus 3.4.1 bestimmt einen maximalen Matroidfluss in $O(n^7 + n^6)$ (siehe Satz 3.4.2) und Algorithmus 3.5.1 bestimmt einen Min-Mean-Circuit in $O(\rho\phi(n^5 + n^4))$ (siehe Korollar 3.5.1). Indem wir die Kapazitätsfunktion in die Kostenfunktion modellieren (siehe Abschnitt 3.1) erhalten wir eine konvexe separable Kostenfunktion mit rationalen Steigungen und ganzzahligen Breakpoints. Anhand Satz 3.2.3 wissen wir, dass die Min-Mean-Circuit-Canceling Methode dafür $O(S\rho^\perp\phi\lceil\log(\phi)\rceil)$ Iterationen benötigt. Insgesamt erhalten wir damit eine Laufzeit von $O(n^7 + n^6 + S\rho\rho^\perp\phi^2(n^5 + n^4)\lceil\log(\phi)\rceil)$. □

Korollar 3.6.1. *Falls $c : E \to \mathbb{Z}_+$ ist, bestimmt Algorithmus 3.6.1 einen maximalen Matroidfluss mit minimalen Kosten in einem regulären Matroid in*

$$O(n^7 + n^6 + \rho\rho^\perp\phi^2(n^5 + n^4)\lceil\log(\phi C)\rceil).$$

Beweis: Verwenden wir im Beweis von Satz 3.6.1 $O(\rho^\perp\phi\lceil\log(\phi C)\rceil)$ aus Satz 3.2.2 für die Anzahl der Iterationen der Min-Mean-Circuit-Canceling Methode, dann erhalten wir eine Gesamtlaufzeit von $O(n^7 + n^6 + \rho\rho^\perp\phi^2(n^5 + n^4)\lceil\log(\phi C)\rceil)$. □

4. Implementierung des Algorithmus

4.1. Entwickeltes Programm

In diesem Abschnitt gehen wir auf Details zur Implementierung von Algorithmus 3.6.1 ein. Als Programmiersprache haben wir *C++* verwendet. Zudem wurden die Softwarepakete *boost* [4] und *CPLEX* [19] benutzt. Bei boost handelt es sich um eine freie C++-Bibliothek, welche aus einer großen Anzahl von portablen Unterbibliotheken besteht. Im wesentlichen verwenden wir von boost die Unterbibliothek *uBLAS*. uBLAS liefert Matrix- und Vektorklassen, sowie grundlegende Routinen der linearen Algebra. CPLEX ist eine kommerzielle Software mit der man verschiedene Optimierungsprobleme lösen kann. CPLEX kann sowohl alleinstehend, als auch als Bibliothek verwendet werden.

Bei der Implementierung von Algorithmus 3.6.1 beschränken wir uns auf eine Kapazitätsfunktion $k : E \to \mathbb{Z}_+$ und auf eine Kostenfunktion $c : E \to \mathbb{Z}$. Zudem verwenden wir zum Lösen der linearen Programme nicht den Algorithmus von Tardos aus Abschnitt 2.2.2, sonderen benutzen den Simplex Algorithmus (siehe Hochstättler [17]) von CPLEX.

Im Wesentlichen wurde der Algorithmus durch fünf Funktionen implementiert. Zuerst wird mit `readinfile()` die Testinstanz (siehe Abschnitt 4.3) gelesen und die entsprechenden Datenstrukturen `M`, `cap`, `cost` und `distinguished` mit den Eingabewerten initialisiert. In `M` steht das Eingabematroid, in `cap` die Kapazitätswerte, in `cost` die Kostenwerte und in `distinguished` das ausgezeichnete Element. Anschließend wird mit `getmaxmatroidflow()` ein maximaler Matroidfluss berechnet. Das Ergebnis wird in dem Vektor `flow` gespeichert. Dann folgt eine Endlosschleife die erst dann abgebrochen wird, wenn kein negativer Min-Mean-Circuit mehr gefunden wird. Mit `getresidualmatroid()` wird das Residualmatroid (siehe Abschnitt 3.3) `R` mit entsprechenden Kapazitäten `rescap` und Kosten `rescost` berechnet. Die Verbindung zwischen Residualmatroid und Eingabematroid wird in `mapping` festgehalten. Nach der Erstellung des Residualmatroids wird, falls vorhanden, durch die Funktion `getminmeancircuit()` ein negativer Min-Mean-Circuit `minmeancircuit` bestimmt. Mit der Funktion `augmentmatroidflow()` wird dann entsprechend der Matroidfluss `flow` augmentiert. Ist kein negativer Min-Mean-Circuit mehr vorhanden, wird die Endlosschleife beendet. Die eben aufgezeigte Arbeitsweise des Programms ist in gekürzter Form im folgenden Quellcode nochmal dargestellt:

```
int main(...)
{
   integer_matrix M;
   vector<int> cap;
   vector<int> cost;
   int distinguished;
   vector<int> flow;

   readinfile(inputfile,M,cap,cost,distinguished);
   getmaxmatroidflow(M,cap,distinguished,flow);

   integer_matrix R;
   vector<int> rescap;
   vector<int> rescost;
   map<int,pair<int,int> > mapping;
   vector<int> minmeancircuit;

   while( true )
   {
      getresidualmatroid(M,flow,cap,
                         cost,distinguished,R,
                         rescap,rescost,mapping);

      getminmeancircuit(R,rescost,minmeancircuit);
      if( minmeancircuit.size() == 0 )
         break;

      augmentmatroidflow(rescap,minmeancircuit,
                         mapping,distinguished,flow);
   }

   return 0;
}
```

Wir wollen nun auf vier der oben beschriebenen Funktionen näher eingehen. Über

```
int getmaxmatroidflow(const integer_matrix& M,
                      const vector<int>& cap,
                      int distinguished,
                      vector<int>& flow);
```

wird ein maximaler Matroidfluss berechnet. Dazu werden innerhalb einer Endlosschleife, wie in Abschnitt 3.4 beschrieben, sukzessive kleinste augmentierende Kreise innerhalb des entsprechenden Residualmatroids berechnet. Im Gegensatz zum Residualmatroid, welches für die Berechnung von Min-Mean-Circuits verwendet wird, spielen hier aber nur die Kapazitäten `cap` und nicht die Kosten eine Rolle. Die

Berechnung der kleinsten augmentierenden Kreise geschieht, indem in jeder Iteration ein lineares Programm (3.16) aufgestellt und mit `cplex.solve()` gelöst wird. Dabei muss sichergestellt werden, dass diese Kreise immer das ausgezeichente Element `distinguished` enthalten. Anschließend wird die maximale Flusserhöhung ε berechnet und der Fluss `flow` entsprechend augmentiert. Ist kein augmentierender Kreis mehr vorhanden, d.h. das lineare Programm ist unzulässig, dann wird die Endlossschleife abgebrochen und der finale Matroidfluss ist ein maximaler Matroidfluss bzgl. des ausgezeichneten Elements. Der Wert des Matroidflusses wird durch das ausgezeichnete Element bestimmt. Die Funktion kann als Statuswert einen `int` liefern. Ein Wert von -1 deutet auf einen Fehler hin. Ein Wert von 0 zeigt eine erfolgreiche Ausführung der Funktion an. Selbiges gilt auch für die folgenden Funktionen. In der Funktion

```
int getresidualmatroid(const integer_matrix& M,
                       const vector<int>& flow,
                       const vector<int>& cap,
                       const vector<int>& cost,
                       int distinguished,
                       integer_matrix& R,
                       vector<int>& rescap,
                       vector<int>& rescost,
                       map<int,pair<int,int> >& mapping);
```

wird abhängig vom Fluss und den Kapazitäten bestimmt, welche Elemente das Residualmatroid enthält und wie sich die neuen Kapazitäten `rescap` zusammensetzen. Bei entgegengerichteten Elementen wird zudem der Kostenwert mit einem negativen Vorzeichen versehen. Ist das Element gleichgerichtet, wie im Eingabematroid, dann bleiben die Kostenwerte unverändert. Die angepassten Kostenwerte werden in `rescost` abgespeichert. Schließlich wird der Zusammenhang `mapping` zwischen den beiden Matroiden aktualisiert. Der Algorithmus 3.5.1 zur Bestimmung eines Min-Mean-Circuits ist innerhalb der Funktion

```
int getminmeancircuit(const integer_matrix& M,
                      const vector<int>& cost,
                      vector<int>& minmeancircuit);
```

implementiert. In `M` wird die Standardrepräsentationsmatrix des Residualmatroids und in `cost` die zu den Elementen des Matroids gehörigen Kosten übergeben. In `minmeancircuit` wird der Inzidenzvektor geliefert, welcher dem negativen Min-Mean-Circuit des übergebenen Matroids entspricht. Zunächst werden Kopien von `R=-M` und `c=cost` angelegt. Dabei ist `R` im Vorzeichen gedreht, weil in jeder Iteration ein kürzester Kreis gesucht wird, welcher das Element `r` mit `c(r)<0` in umgekehrter Orientierung enthält. Von `c` wird eine Kopie erstellt, weil die Kostenfunktion durch (3.19) und (3.18) verändert wird. Die Bestimmung des Min-Mean-Circuit findet in einer Endlosschleife statt. In jeder Iteration wird ein Element `r` mit `c(r)<0` gewählt. Ist kein solches `r` mehr vorhanden, wird die Endlosschleife abgebrochen.

Dabei wählen wir das `r`, bei dem `c(r)` am kleinsten ist. Wurde in einer Iteration ein Element `r` gewählt, welches in der darauffolgenden Iteration auch gewählt werden kann, ist dieses Element wieder zu verwenden (vgl. Abschnitt 3.5). Anschließend wird ein lineares Programm als CPLEX Model wie in (3.20) aufgebaut und durch `cplex.solve()` gelöst. Neben der Primallösung `x` holen wir uns von CPLEX die Duallösung `d`. Anhand der Duallösung können wir `y` bestimmen. Dann aktualisieren wir die Kostenfunktion durch `c=c+y` und führen im Falle von `c(r)<0` einen zusätzlichen Shift der Kostenfunktion mit `c=c+a` durch. Als kritisch sind in der Implementierung von Algorithmus 3.5.1 Vergleiche numerischer Werte zu sehen, um ein korrektes Verhalten zu gewährleisten. Besonders der Test, ob `c(r)<0` gilt, ist problematisch. Wir verwenden dafür ein Makro `IsLT(val1,val2)` mit `val1=c(r)` und `val2=0`. Dieses Makro testet, ob `val1-val2` echt kleiner als $-\varepsilon$ mit $\varepsilon = 10^{-9}$ ist. Ein ähnliches Vorgehen brauchen wir auch noch an anderen Stellen und haben deshalb entsprechende Makros in der Datei `basetypes.hpp` implementiert:

```
#define DEFAULT_EPSILON   1e-09
#define REALABS(x)        (fabs(x))
#define EPSGE(x,y,eps)    ((x)-(y) >= -(eps))
#define EPSLT(x,y,eps)    ((x)-(y) < -(eps))
#define EPSLE(x,y,eps)    ((x)-(y) <= (eps))
#define EPSGT(x,y,eps)    ((x)-(y) > (eps))
#define EPSEQ(x,y,eps)    (REALABS((x)-(y)) <= (eps))
#define IsGE(val1, val2)  ( EPSGE(val1, val2, DEFAULT_EPSILON) )
#define IsEQ(val1, val2)  ( EPSEQ(val1, val2, DEFAULT_EPSILON) )
#define IsLT(val1, val2)  ( EPSLT(val1, val2, DEFAULT_EPSILON) )
#define IsLE(val1, val2)  ( EPSLE(val1, val2, DEFAULT_EPSILON) )
#define IsGT(val1, val2)  ( EPSGT(val1, val2, DEFAULT_EPSILON) )
```

In der Funktion

```
int augmentmatroidflow(const vector<int>& rescap,
                       const vector<int>& minmeancircuit,
                       map<int,pair<int,int> >& mapping,
                       vector<int>& flow,
                       vector<int>& augment);
```

wird anhand `rescap` der Wert τ bestimmt. Dann gehen wir über alle Elemente des berechneten Min-Mean-Circuit `minmeancircuit` und entscheiden anhand der Matroidzuordnung `mapping` ob wir den Fluss auf dem entsprechenden Element vergrößern oder verkleinern. Der Vektor `augment` dient uns lediglich dazu die lokale Flussveränderung anzuzeigen.

Schließlich wurde ein `Makefile` implementiert. Es definiert wie die einzelnen Quelldateien

```
algorithm.cpp, augment.cpp, maxmatroidflow.cpp,
minmeancircuit.cpp, readinput.cpp, residualmatroid.cpp
```

miteinander zu kompilieren sind und welche zusätzlichen Bibliotheken zu linken sind, damit ein ausführbares Programm entsteht. Insgesamt hat das Programm einen Umfang von 517 NCLOC[1]. Dabei wurden nur die `.cpp`-Dateien gezählt. Header-Dateien sind folgende vorhanden:

```
basetypes.hpp, augment.hpp, maxmatroidflow.hpp,
minmeancircuit.hpp, readinput.hpp, residualmatroid.hpp
```

Ruft man das Programm ohne Angabe von Parametern auf, dann erscheint eine Ausgabe zu dessen Bedienung:

```
Usage: algorithm <matroid-instance.txt>
```

Abschließend wollen wir noch auf die Ausgabe des Programms eingehen. Diese könnte z.B. wie folgt aussehen:

```
### max matroid flow: 5 2 2 4 3 2 4 1 0 2 4 1 (7)
### cost: 68
### augmenting matroid flow: 0 0 0 0 0 3 -3 0 3 0 0 0 0
### augmenting matroid flow: 0 0 1 1 -1 0 0 0 0 0 0 0 0
### number negativ min-mean-circuits: 2
### optimal matroid flow: 5 2 3 5 2 5 1 1 3 2 4 1 (7)
### optimal cost: 61
```

Zuerst wird der berechnete maximale Matroidfluss $[5, 2, 2, 4, 3, 2, 4, 1, 0, 2, 4, 1, (7)]$ ausgegeben. Der Wert der Matroidflusses wird durch das ausgezeichnete Element bestimmt. Der entsprechende Wert ist mit runden Klammern gekennzeichnet und in diesem Fall 7. Danach folgt der Kostenwert 68 des anfänglich bestimmten maximalen Matroidflusses. Es folgen n viele Ausgaben der augmentierenden Kreisflüsse. Dabei ist n die Anzahl der gefundenen negativen Min-Mean-Circuits. In diesem Beispiel wurden 2 Min-Mean-Circuits gefunden. Letzlich erfolgt die Ausgabe des optimalen Matroidflusses $[5, 2, 3, 5, 2, 5, 1, 1, 3, 2, 4, 1, (7)]$ und dessen Kostenwert 61.

4.2. Signierung

Um die Implementierung von Algorithmus 3.6.1 testen zu können, benötigen wir als Eingabe ein reguläres Matroid, welches mit einer Orientierung versehen ist. Dies könnten wir dadurch erreichen, indem wir Algorithmus 2.4.1 implementieren. Wir haben uns aber entschieden, die für [43] entwickelte Software *Unimodularity Library* [42] dafür zu verwenden, denn diese Bibliothek ist umfangreich getestet und der Signierungsalgorithmus ist sehr durchdacht implementiert. Dabei übergeben wir der Bibliothek die vorher eingelesene Matrix durch folgenden Funktionsaufruf:

```
bool sign_matrix(integer_matrix& matrix);
```

[1] Non-Comment Lines of Code

Anschließend erhalten wir die signierte Matrix zurück. Allerdings ergibt sich dabei folgendes Problem. Die Bibliothek entscheidet selbst wie sie die Matrix signiert. Wir möchten aber, ohne die Bibliothek explizit anzupassen, Einfluss auf die Signierung nehmen können. Die Lösung besteht darin, wie im Algorithmus 2.4.1, zuerst einen bipartiten Graphen zu erzeugen, welcher für jede Zeile und Spalte der Eingabematrix einen Knoten enthält. Ein Zeilenknoten wird mit einem Spaltenknoten verbunden, wenn der entsprechende Matrixeintrag eine 1 ist. Dann wird mit dem Algorithmus von *Kruskal* ein aufspannender Baum auf dem Graphen berechnet. Anhand einer wählbaren *seed* wird dann der Anteil der Matrix, welcher dem aufspannenden Baum entspricht, zufällig signiert. Anschließend wird die teilsignierte Matrix an [42] übergeben und die Signierung wird vervollständigt. Für die Erstellung des Graphen verwenden wir die *LEMON Graph Library* [25]. Am Ende der Signierung wird mit [42] getestet ob die signierte Matrix total unimodular ist. Als Eingabeformat wählen wir das Format der Testinstanzen von [42]. In der ersten Zeile steht, durch ein Leerzeichen getrennt, die Anzahl der Zeilen und Spalten der Matrix. Nach einer Leerzeile folgt die Darstellung der Matrix. Als Beispiel sei im Folgenden die Instanz `net-10x10.mat` von [42] angegeben.

```
10 10

 0  0  1  1  1  1  0  0  1 -1
 1  0  0  0  0  0  1  1  0  1
 1  0  1  1  1  1  1  1  1  0
 0  0  1  1  0  1  0  0  0 -1
 0  1  1  1  1  0  1  1  1  0
 1  0  0  0  0  1  0  0  0  0
 0  0  0  0  0  0  0  0  0  0
 0  0  0  1  0  0  0  0  0  0
 0  1  1  1  1  0  1  1  1  0
 0  0  1  0  1  0  0  1  1  0
```

Um ein ausführbares Progamm der Signierung erzeugen zu können, kann man das mitgelieferte `Makefile` verwenden. Darin ist spezifiziert wie der Quellcode `singing.cpp` zu kompilieren und mit den beiden Bibliotheken [25] und [42] zu verlinken ist. Das ausführbare Programm erwartet einen Startwert für den Zufallsgenerator, die Eingabedatei und die Ausgabedatei. In der Ausgabedatei ist nach einer erfolgreichen Ausführung des Programms die signierte Matrix enthalten. Ohne Angabe von Parametern zeigt das ausführbare Programm `signing` seine Benutzung an:

```
Usage: signing seed <input-matrix.txt> <signed-matrix.txt>
```

4.3. Testinstanzen

Wir wollen nun einige Instanzen generieren mit denen wir die Funktionalität der Implementierung von Algorithmus 3.6.1 testen können.

Zuerst legen wir ein einfaches Format fest, welches der Algorithmus einlesen kann. Dazu verwenden wir einfach das Format aus [42] erweitert um drei Zeilen. In der ersten zusätzlichen Zeile wird jedem Element ein Kapazitätswert zugeordnet und in der zweiten, zusätzlichen Zeile wird jedem Element ein Kostenwert zugeteilt. Letztlich folgt eine Zahl welche das ausgezeichnete Element definiert. Für das ausgezeichnete Element legen wir zudem den Kapazitätswert `Inf` und den Kostenwert 0 fest. Die Eingabe einer (5×9)-Matrix mit Kapazitäts-/Kostenwerten und ausgezeichnetem Element e_1 sieht dann z.B. wie folgt aus:

```
5 9

0  0  1  0  0 -1  0  0  1
0  0  0  1 -1 -1  0  0  0
0  0  0  0  0  0  1 -1  1
1  0  0  0 -1 -1  0 -1  1
0  1  0  0 -1  0  0 -1  0

Inf 10 10 8 5 7 3 9 14

0 1 10 10 12 2 3 7 1

1
```

Der einfachste Fall Testinstanzen regulärer Matroide zu erzeugen besteht darin, graphische Matroide zu verwenden (siehe Abb. 2.7). In Abb. 4.1 ist ein graphisches Matriod mit 13 Elementen dargestellt.

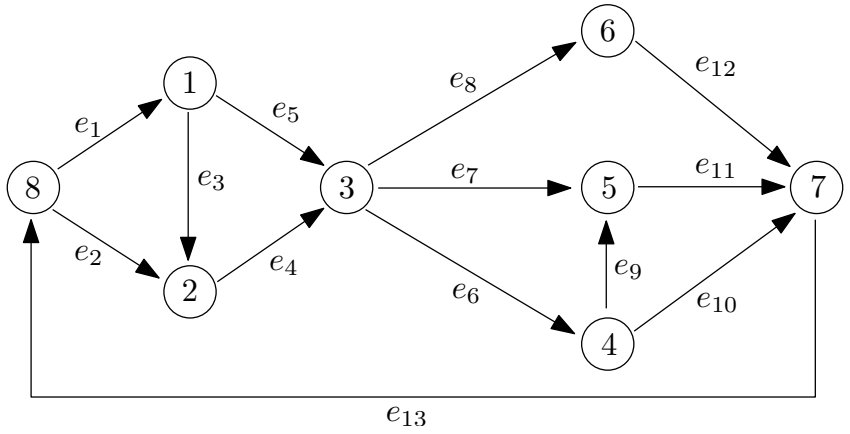


Abb. 4.1.: Graphisches Matroid

Die dem graphischen Matroid entsprechende Inzidenzmatrix ist

$$\begin{array}{c} \begin{array}{ccccccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} & e_{12} & e_{13} \end{array} \\ \left(\begin{array}{ccccccccccccc} -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{array} \right). \end{array}$$

Durch Zeilenumformungen erhalten wir die folgende Repräsentationsmatrix:

$$\begin{array}{c} \begin{array}{ccccccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} & e_{12} & e_{13} \end{array} \\ \left(\begin{array}{ccccccccccccc} -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right). \end{array}$$

Entsprechend der Zuordnung der Spalten zu den Elementen des Matroids verwenden wir als Kapazitäten

$$[5, 2, 3, 7, 3, 5, 7, 1, 3, 2, 4, 1, \text{Inf}]$$

und als Kosten

$$[1, 1, 1, 2, 4, 2, 5, 8, 1, 1, 1, 1, 0].$$

Das ausgezeichnete Element soll e_{13} sein. In A.1 ist zuerst die gesamte Testinstanz dargestellt, dann folgt die Ausgabe des entwickelten Programms (siehe Abschnitt 4.1). Zu Beginn wird ein maximaler Matroidfluss $[5, 2, 2, 4, 3, 2, 4, 1, 0, 2, 4, 1, (7)]$ berechnet. Der Fluss über e_{13} ist 7. Es werden zwei Min-Mean-Circuits gefunden und entsprechend der Matroidfluss augmentiert. Der optimale Matroidfluss ist

$$[5, 2, 3, 5, 2, 5, 1, 1, 3, 2, 4, 1, (7)].$$

Die Kosten des optimalen Matroidflusses sind 61. Anschließend ist das entsprechende lineare Programm und die Lösung mit CPLEX dargestellt. Man sieht, dass die beiden Ergebnisse übereinstimmen.

Wir wollen noch ein zweites, graphisches Matroid als Testinstanz generieren. Wir wählen, dass durch den Graphen $K_{3,3}$ induzierte Matroid $M(K_{3,3})$ mit einer bestimmten Orientierung (siehe Abb. 4.2). Die entsprechende Inzidenzmatrix ist

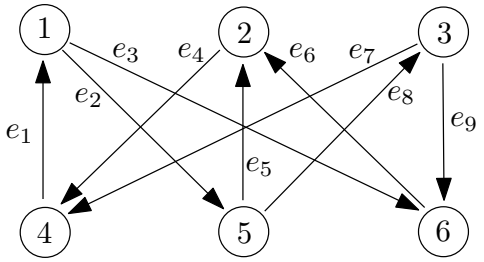


Abb. 4.2.: Graphisches Matroid $M(K_{3,3})$ mit Orientierung

$$\begin{array}{c} \begin{array}{ccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{array} \\ \left(\begin{array}{ccccccccc} -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 \end{array} \right). \end{array}$$

Die Inzidenzmatrix können wir zu einer Repräsentationsmatrix

$$\begin{array}{c} \begin{array}{ccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{array} \\ \left(\begin{array}{ccccccccc} 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \end{array} \right) \end{array}$$

umformen. Als Kapazitäten legen wir

$$[\text{Inf}, 10, 10, 8, 5, 7, 3, 9, 14]$$

und als Kosten

$$[0, 1, 10, 10, 12, 2, 3, 7, 1]$$

fest. Das ausgezeichnete Element ist e_1. In A.2 ist wieder zuerst die Testinstanz dargestellt. Anschließend folgt die Ausgabe des implementierten Programms. Der Fluss über das ausgezeichnete Element ist 11 und als optimalen Fluss erhalten wir

$$[(11), 10, 1, 8, 1, 7, 3, 9, 6]$$

mit Kostenwert 204. Zur Verifikation ist auch hier wieder anschließend das entsprechende lineare Programm und dessen Lösung mit CPLEX dargestellt.

Wesentlich interessanter wird es, wenn wir anstelle von graphischen Matroiden, nun kographische Matroide als Testinstanzen betrachten. In [40] ist auf Seite 50 die Repräsentationsmatrix

$$\left(\begin{array}{ccccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{array} \right)$$

für $M(K_{3,3})^*$ dargestellt. Nun verwenden wir das in Abschnitt 4.2 implementierte Programm um die Matrix zu signieren und erhalten

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

Wir legen die Kapazitäten

$$[3, 1, 12, 1, 3, 2, 4, \text{Inf}, 5]$$

und Kosten

$$[2, 3, 2, 5, 2, 8, 2, 0, 10]$$

fest. Das ausgezeichnete Elemente soll e_8 sein. Die entsprechende Testinstanz ist unter A.3 dargestellt. Angewandt auf den entwickelten Algorithmus können wir einen optimalen Matroidfluss von

$$[3, 1, 4, 0, 0, 0, 3, (4), 1]$$

mit Kostenwert 33 bestimmen. Die Lösung des linearen Programms mit CPLEX bestätigt die Richtigkeit.

Nun wollen wir, die in Abschnitt 2.4.3 entwickelten Kompositionsmethoden für binäre Matroide verwenden, um uns komponierte reguläre Matroide zu generieren. Nicht dargestellte Einträge in den folgenden Matrizen entsprechen 0-Einträgen. Wir wollen zuerst $M(K_5)^* \oplus_2 M(K_{3,3})^*$ erstellen. Oben haben wir bereits die Repräsentationsmatrix von Matroid $M(K_{3,3})^*$ gezeigt. Eine Repräsentationsmatix für $M(K_5)^*$ finden wir unter [40] auf Seite 50:

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

Indem wir zuerst die Einheitsmatrizen weglassen, erhalten wir die partiellen Standardrepräsentationsmatrizen. Diese komponieren wir, wie in Abschnitt 2.4.3 gezeigt, durch eine 2-Summe (siehe Abb. 4.3). Letztlich hängen wir eine (9×9)-Einheitsmatrix an und signieren die gesamte Matrix. Die dabei entstandene Matrix ist unter A.4 dargestellt. Auch hier fügen wir Kapazitäten

$$[2, 4, 5, 6, 5, 7, 7, 3, 5, 2, 3, 4, 3, \text{Inf}, 10, 4, 4]$$

und Kosten

$$[2, 1, 3, 10, 11, 3, 8, 4, 2, 6, 3, 4, 4, 0, 7, 12, 3]$$

hinzu. Zudem definieren wir e_{14} als das ausgezeichnete Element. Unser Algorithmus berechnet auf dieser Instanz einen optimalen Matroidfluss von

$$[0,0,0,3,3,7,0,3,4,0,0,0,3,(7),0,0,4]$$

mit Kostenwert 128. Dabei wurde ein negativer Min-Mean-Circuit gefunden. Die Lösung des linearen Programms bestätigt die Korrektheit des optimalen Matroidflusses.

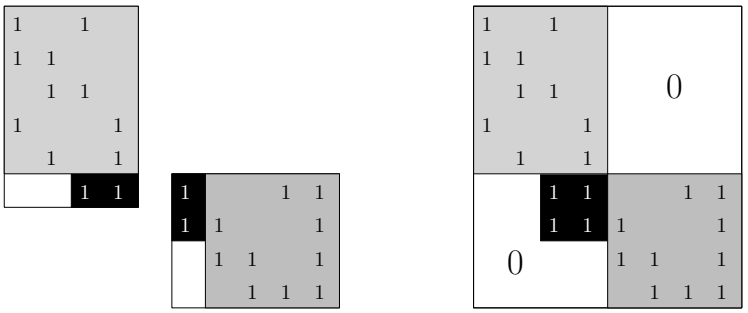


Abb. 4.3.: $M(K_5)^* \oplus_2 M(K_{3,3})^*$

Wir wollen eine weitere Testinstanz über eine 2-Summen Komposition erstellen. Diesmal komponieren wir aber drei Matroide: $M(K_5) \oplus_2 R_{10} \oplus_2 M(K_{3,3})$. Dabei dürfen wir die partiellen Standardrepräsentationsmatrizen durch Permutation der Zeilen und Spalten und durch Pivots (siehe [40] Seite 20) geeignet umformen. Zuerst benötigen wir die partiellen Standardrepräsentationsmatrizen in ganz bestimmten Formen. Die partielle Standardrepräsentationsmatrix von $M(K_5)$ (siehe [40] Seite 48)

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \quad \text{formen wir zu} \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

um. Die Repräsentationsmatrix von R_{10} formen wir von

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{zu} \quad \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

um. Letztlich nehmen wir eine Umformung der Matrix von $M(K_{3,3})$ (siehe [40] Seite 49) von

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{zu} \quad \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

vor. Dann fügen wir die drei durch Umformungen entstandenen Matrizen durch 2-Summen Kompositionen zusammen (siehe Abb. 4.4), stellen eine Einheitsmatrix davor und signieren die gesamte Matrix. Anschließend definieren wir Kapazitäten

$$[10, 12, 9, 14, 15, 29, 3, 29, 38, 19, 8, 29, 27, 26, 14, 11, 19, \text{Inf}, 28, 29, 19, 3, 5, 5, 9],$$

Kosten

$$[9, 7, 6, 20, 4, 9, 6, 14, 9, 8, 9, 7, 10, 11, 12, 8, 9, 0, 8, 9, 7, 3, 3, 2, 5]$$

und das ausgezeichnete Element e_{18}. Die dadurch entstandene Testinstanz ist unter A.5 dargestellt. Unser entwickelter Algorithmus, angewandt auf diese Testinstanz, berechnet einen optimalen Fluss von

$$[10, 12, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, (12), 2, 0, 0, 0, 0, 0, 0]$$

mit Gesamtkosten 310. Die Lösung ist korrekt, wie aus der Verifikation mit CPLEX ersichtlich ist.

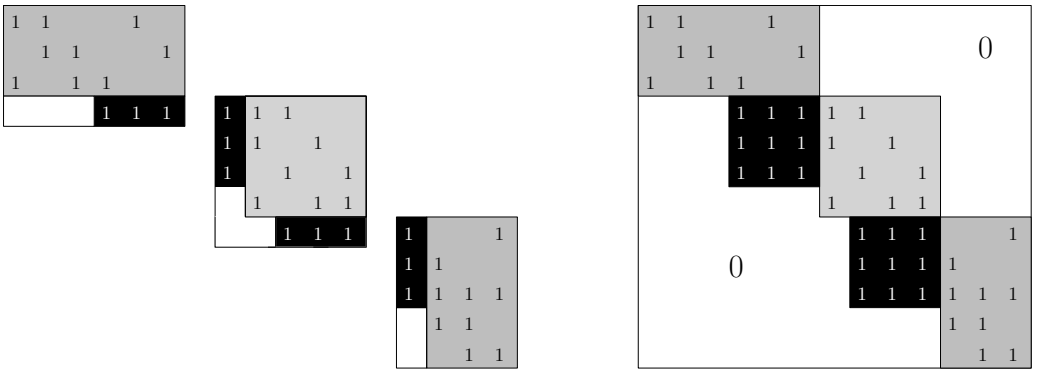


Abb. 4.4.: $M(K_5) \oplus_2 R_{10} \oplus_2 M(K_{3,3})$

Jetzt wollen wir zwei Matroide durch eine 3-Summe $M(K_{3,3})^* \oplus_3 M(K_{3,3})$ komponieren. Wir führen die notwendigen Umformungen der Matrix von $M(K_{3,3})^*$

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{zu} \quad \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

und der Matrix von $M(K_{3,3})$

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{zu} \quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

durch. Die partielle Standardrepräsentationsmatrix der 3-Summe ist in Abb. 4.5 dargestellt. Im Überlappungsbereich erhalten wir die folgenden Matrizen (vgl. Abschnitt 2.4.3):

$$\bar{D} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, D^1 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, D^2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

$\bar{D}$ hat dabei vollen GF(2)-Rang. Um die gesamte Matrix D zu erhalten müssen wir aber noch D^{12} mit $D^{12} = D^2 \cdot \bar{D}^{-1} \cdot D^1$ berechnen und erhalten

$$D^{12} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

Auch hier signieren wir die enstandene Matrix, fügen Kapazitäten

$$[2, 4, 6, 7, \text{inf}, 2, 3, 8, 5, 4, 10, 2],$$

Kosten

$$[4, 5, 3, 3, 0, 2, 10, 20, 2, 4, 8, 1]$$

hinzu und definieren als ausgezeichnetes Element e_5. Die generierte Testinstanz ist unter A.6 dargestellt. Unser Algorithmus findet darin zwei negative Min-Mean-Circuits und augmentiert entsprechend den Matroidfluss. Letztlich erhalten wir einen optimalen Fluss

$$[2, 2, 2, 0, (9), 2, 3, 5, 3, 4, 7, 2]$$

mit Kosten 238. Auch dieses Ergebnis konnten wir mit einem linearen Programm und einer Lösung von CPLEX bestätigen.

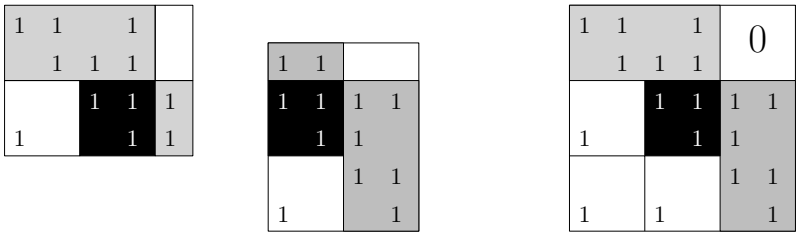


Abb. 4.5.: $M(K_{3,3})^* \oplus_3 M(K_{3,3})$

Abschließend wollen wir eine letzte Testinstanz $M(K_5) \oplus_3 M(K_5)^*$ erzeugen. Wir formen zunächst die Matrix von $M(K_5)$

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \quad \text{zu} \quad \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

um und dann formen wir die Matrix von $M(K_5)^*$

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{zu} \quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

um. Die resultierende 3-Summen Komposition ist in Abb. 4.6 dargestellt. Wir erhalten im Überlappungsbereich die Matrizen

$$\bar{D} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, D^1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, D^2 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \text{ und berechnen } D^{12} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Da wir nun die komplette Matrix bestimmt haben, fügen wir eine Einheitsmatrix hinzu und signieren die gesamte Matrix. Abermals legen wir Kapazitäten

$$[2, 3, 8, 5, 4, 9, 10, 5, 5, 2, 4, 8, 9, \text{inf}]$$

Kosten

$$[1, 1, 3, 2, 3, 4, 5, 9, 10, 2, 10, 2, 10, 0]$$

fest und definieren als ausgezeichnetes Element e_{14}. Die dabei entstandene Testinstanz ist unter A.7 aufgelistet. Wir finden mit unserem Algorithmus darin einen negativen Min-Mean-Circuit und berechnen einen optimalen Matroidfluss von

$$[2, 0, 3, 5, 3, 0, 5, 0, 0, 2, 0, 0, 5, (5)]$$

mit Kostenwert 109. Die Korrektheit wird abermals durch die Lösung des entsprechenden linearen Programms mit CPLEX bestätigt.

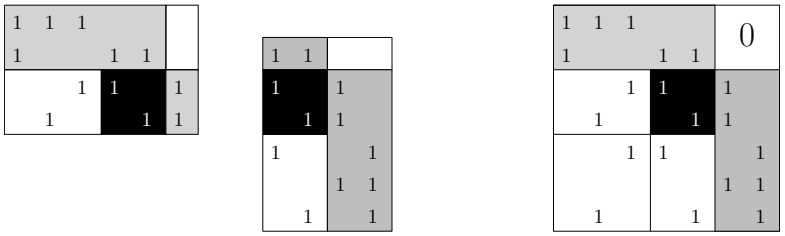


Abb. 4.6.: $M(K_5) \oplus_3 M(K_5)^*$

5. Zusammenfassung

In der vorliegenden Arbeit haben wir uns zuerst Grundlagen der Laufzeitanalyse von Algorithmen, der linearen Optimierung, der Graphentheorie und der Matroidtheorie erarbeitet. Darauf aufbauend haben wir Algorithmen zur Bestimmung eines maximalen Matroidflusses und zur Lösung des Min-Cost-Flow Problems in regulären Matroiden entwickelt. Für beide Algorithmen wurden Analysen zur Laufzeit angegeben. Die Laufzeitanalyse des Algorithmus für das Min-Cost-Flow Problem in regulären Matroiden betrachtet konvexe separable Kostenfunktionen mit ganzzahligen und rationalen Steigungen. Liegen ganzzahlige Steigungen vor, dann können wir einen maximalen Matroidfluss mit minimalen Kosten in $O(n^7 + n^6 + \rho\rho^{\perp}\phi^2(n^5 + n^4)\lceil\log(\phi C)\rceil)$ bestimmen. Eine Laufzeit von $O(n^7 + n^6 + S\rho\rho^{\perp}\phi^2(n^5 + n^4)\lceil\log(\phi)\rceil)$ konnten wir im Falle von rationalen Steigungen zeigen.

Nach der Herleitung und Analyse des Algorithmus haben wir eine Implementierung davon vorgestellt. Um das entwickelte Programm effektiv testen zu können, wurden verschiedene Testinstanzen von regulären Matroiden erstellt. Dabei haben wir uns die Komposition von binären Matroiden (siehe Trümper [40] Seiten 168-187) zu Nutze gemacht. Dadurch konnten wir reguläre Matroide erstellen, welche weder graphisch noch kographisch sind. Um den dabei entwickelten regulären Matroiden eine Orientierung zuordnen zu können, haben wir ein Programm implementiert, mit dem wir zufallsgesteuert reguläre Matroide signieren können. Dabei sind sieben verschiedene Testinstanzen entstanden. Damit wurde der entwickelte Algorithmus getestet. Zur Verifikation der Berechnungen des Algorithmus wurden äquivalente, lineare Programme aufgestellt und mit einem kommerziellen Löser gegengerechnet. In allen Fällen wurde die Äquivalenz der optimalen Lösung gezeigt.

Der entwickelte Algorithmus löst sukzessive lineare Programme um zuerst einen maximalen Matroidfluss und danach negative Min-Mean-Circuits zu bestimmen. Damit ist der Algorithmus nicht rein kombinatorischer Natur. Allerdings ist der Algorithmus durch seinen strukturellen Aufbau einem kombinatorischen Algorithmus sehr ähnlich. Würde es uns gelingen, einen kombinatorischen Algorithmus für das *Shortest-Circuit* Problem in regulären Matroiden mit einer streng polynomialen Laufzeit zu entwickeln, dann würde der hier vorgestellte Algorithmus sofort rein kombinatorische Algorithmen für die Bestimmung eines maximalen Matroidflusses und eines maximalen Matroidflusses mit minimalen Kosten liefern. Und mehr noch, wir würden sogar einen rein kombinatorischen streng polynomialen Algorithmus für das Min-Cost-Flow Problem in regulären Matroiden mit einer stückweise linearen konvexen separablen Kostenfunktion erhalten. Es bleibt somit die Frage offen, wie man einen kombinatorischen Algorithmus für das Shortest-Circuit Problem in regulären Matroiden mathematisch herleiten kann. Dabei könnte es hilfreich sein das Konzept

der Adjazenz in binären Matroiden näher zu untersuchen. Seymour [35] könnte dafür als Grundlage dienen. Eine weitere Herangehensweise könnte die Entwicklung einer Fallunterscheidung anhand von Satz 2.4.20 bzw. [34] sein. Letztlich wäre es sinnvoll weitere konvexe separable Kostenfunktionen mit entsprechender Laufzeitanalyse zu betrachten. Als Ausgangspunkt dafür bietet sich Karzanov/McCormick [22] an.

Literaturverzeichnis

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. Ziegler. *Oriented matroids*. Cambridge University Press, 1993.

[3] J.-A. Bondy and U. S. R. Murty. *Graph theory*. Graduate texts in mathematics. Springer, New York, London, 2007. OHX.

[4] Boost C++ Libraries. `http://www.boost.org`, 2013.

[5] R. E. Burkard and H. Hamacher. Minimal cost flows in regular matroids. *Mathematical Programming Study*, 14:32–47, 1981.

[6] P. Camion. Characterization of totally unimodular matrices. *Proceedings of the American Mathematical Society*, 16(5):1068–1073, 1965.

[7] G. Cornuéjols. *Combinatorial Optimization: Packing and Covering*. Society for Industrial and Applied Mathematics, 2001.

[8] L. R. Ford and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. *Canadian Journal of Mathematics*, 9:210–218, 1957.

[9] N. Gaffke and F. Pukelsheim. Vector and matrix apportionment problems and separable convex integer optimization. *Math. Meth. of OR*, 67(1):133–159, 2008.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[11] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *J. ACM*, 36(4):873–886, 1989.

[12] M. Grötschel. *Diskrete Optimierung*. Technische Universität Berlin, Vorlesungsskript, 2013.

[13] M. Grötschel. *Einführung in die Lineare und Kombinatorische Optimierung*. Technische Universität Berlin, Vorlesungsskript, 2013.

[14] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[15] H. Hamacher. *Flows in regular matroids*. Oelgeschlager, Gunn and Hain Königstein, 1981.

[16] D. S. Hochbaum and J. G. Shanthikumar. Convex separable optimization is not much harder than linear optimization. *J. ACM*, 37(4):843–862, October 1990.

[17] W. Hochstättler. *Lineare Optimierung*. FernUni Hagen, Vorlesungsskript, 2013.

[18] W. Hochstättler and A. Schliep. *CATBox: An Interactive Course in Combinatorial Optimization*. Springer, 1st edition. edition, Feb. 2010.

[19] IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html`, 2013.

[20] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, Sept. 1978.

[21] A. V. Karzanov. Minimal mean weight cuts and cycles in directed graphs. *Providence, RI: American Mathematical Society*, 1985.

[22] A. V. Karzanov and S. T. McCormick. Polynomial methods for separable convex optimization in unimodular linear spaces with applications. *SIAM J. Comput.*, 26(4):1245–1275, Aug. 1997.

[23] M. Klein. A primal method for minimal cost flows, with applications to the assignment and transportation problems, 1967.

[24] B. Korte and J. Vygen. *Kombinatorische Optimierung: Theorie und Algorithmen*. Springer, 2008.

[25] LEMON Graph Library 1.3. `http://lemon.cs.elte.hu/trac/lemon`, 2013.

[26] L. Lovász and A. Recski. Selected topics of matroid theory and its applications. pages [171]–185, 1982.

[27] G. J. Minty. On the axiomatic foundations of the theories of directed linear graphs, electrical networks and network-programming. *J. Math. Mech.*, 15:485–520, 1966.

[28] J. G. Oxley. *Matroid theory*. Oxford University Press, New York, 1992.

[29] R. Reemtsen. *Einführung in die nichtlineare Optimierung*. FernUni Hagen, Vorlesungsskript, 2008.

[30] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, New York, NY, USA, 1970.

[31] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[32] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

[33] U. Seip. *Graphentheorie*. FernUni Hagen, 2001.

[34] P. D. Seymour. Decomposition of regular matroids. *J. Comb. Theory, Ser. B*, 28(3):305–359, 1980.

[35] P. D. Seymour. Adjacency in binary matroids. *European Journal of Combinatorics*, 7:171–176, 1986.

[36] G. Strang. *Lineare Algebra*. Springer-Lehrbuch. Springer, 2003.

[37] E. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, July 1985.

[38] E. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Oper. Res.*, 34(2):250–256, Mar. 1986.

[39] K. Trümper. A decomposition theory for matroids. v. testing of matrix total unimodularity. *Journal of Combinatorial Theory, Series B 49*, pages 241–281, 1990.

[40] K. Trümper. *Matroid decomposition.* Academic Press, 1992.

[41] W. T. Tutte. A homotopy theorem for matroids, I. *Transactions of the American Mathematical Society*, 88(1):pp. 144–160, 1958.

[42] Unimodularity Library 1.2b. `http://www.utdallas.edu/~klaus/TUtest`, 2012.

[43] M. Walter and K. Trümper. Implementation of a unimodularity test. *Math. Program. Ser. C*, 5(1):57–73, 2013.

[44] H. Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57:509–533, 1935.

A. Testinstanzen

A.1. Graphisches Matroid

Testinstanz

```
7 13

-1  0  1  0  1  0  0  0  0  0  0  0  0
-1 -1  0  1  1  0  0  0  0  0  0  0  0
-1 -1  0  0  0  1  1  1  0  0  0  0  0
-1 -1  0  0  0  0  1  1  1  1  0  0  0
 0  0  0  0  0  0 -1  0 -1  0  1  0  0
 0  0  0  0  0  0  0 -1  0  0  0  1  0
-1 -1  0  0  0  0  0  0  0  0  0  0  1

5 2 3 7 3 5 7 1 3 2 4 1 Inf

1 1 1 2 4 2 5 8 1 1 1 1 0

13
```

Ausgabe Algorithmus

```
### max matroid flow: 5 2 2 4 3 2 4 1 0 2 4 1 (7)
### cost: 68
### augmenting matroid flow: 0 0 0 0 0 3 -3 0 3 0 0 0 0
### augmenting matroid flow: 0 0 1 1 -1 0 0 0 0 0 0 0 0
### number negativ min-mean-circuits: 2
### optimal matroid flow: 5 2 3 5 2 5 1 1 3 2 4 1 (7)
### optimal cost: 61
```

Lineares Programm

```
min
   +1 f1 +1 f2 +1 f3 +2 f4 +4 f5
   +2 f6 +5 f7 +8 f8 +1 f9 +1 f10
   +1 f11 + 1 f12
s.t.
   f3 - f1 + f5 = 0
   f4 - f1 -f2 + f5 = 0
   f6 -f1 -f2 +f7 + f8 = 0
   f10 -f1 -f2 +f7 +f8 +f9 = 0
   f11 -f7 - f9 = 0
   f12 -f8 = 0
   f13 -f1 -f2 = 0
   f13 = 7
bounds
   0 <= f1 <= 5
   0 <= f2 <= 2
   0 <= f3 <= 3
   0 <= f4 <= 7
   0 <= f5 <= 3
   0 <= f6 <= 5
   0 <= f7 <= 7
   0 <= f8 <= 1
   0 <= f9 <= 3
   0 <= f10 <= 2
   0 <= f11 <= 4
   0 <= f12 <= 1
   0 <= f13
end
```

Lösung des linearen Programms mit CPLEX

```
Dual simplex - Optimal:  Objective =  6.1000000000e+01

Variable Name         Solution Value
f1                           5.000000
f2                           2.000000
f3                           3.000000
f4                           5.000000
f5                           2.000000
f6                           5.000000
f7                           1.000000
f8                           1.000000
f9                           3.000000
f10                          2.000000
f11                          4.000000
f12                          1.000000
f13                          7.000000
```

A.2. Graphisches Matroid $M(K_{3,3})$

Testinstanz

```
5 9

0  0  1  0  0 -1  0  0  1
0  0  0  1 -1 -1  0  0  0
0  0  0  0  0  0  1 -1  1
1  0  0  0 -1 -1  0 -1  1
0  1  0  0 -1  0  0 -1  0

Inf 10 10 8 5 7 3 9 14

0 1 10 10 12 2 3 7 1

1
```

Ausgabe Algorithmus

```
### max matroid flow: (11) 8 3 8 5 3 3 3 0
### cost: 214
### augmenting matroid flow: 0 0 0 0 -4 4 0 4 4
### augmenting matroid flow: 0 2 -2 0 0 0 0 2 2
### number negativ min-mean-circuits: 2
### optimal matroid flow: (11) 10 1 8 1 7 3 9 6
### optimal cost: 204
```

Lineares Programm

```
min
   +0 f1 +1 f2 +10 f3
   +10 f4 +12 f5 +2 f6
   +3 f7 +7 f8 +1 f9
s.t.
   f3 -f6 +f9 = 0
   f4 -f5 -f6 = 0
   f7 -f8 +f9 = 0
   f1 -f5 -f6 -f8 +f9 = 0
   f2 -f5 -f8 = 0
   f1 = 11
bounds
   0 <= f1
   0 <= f2 <= 10
   0 <= f3 <= 10
   0 <= f4 <= 8
   0 <= f5 <= 5
   0 <= f6 <= 7
   0 <= f7 <= 3
   0 <= f8 <= 9
   0 <= f9 <= 14
end
```

Lösung des linearen Programms mit CPLEX

```
Dual simplex - Optimal:  Objective =  2.0400000000e+02

Variable Name           Solution Value
f1                            11.000000
f2                            10.000000
f3                             1.000000
f4                             8.000000
f5                             1.000000
f6                             7.000000
f7                             3.000000
f8                             9.000000
f9                             6.000000
```

A.3. Kographisches Matroid $M(K_{3,3})^*$

Testinstanz

```
4 9

-1  0  0  0  1  0  0  1 -1
 0 -1  0  0 -1  1  0  0  1
 0  0 -1  0  0  1  1  0  1
 0  0  0  1  0  0  1 -1  1

3 1 12 1 3 2 4 Inf 5

2 3 2 5 2 8 2 0 10

8
```

Ausgabe Algorithmus

```
### max matroid flow: 3 1 3 1 0 0 2 (4) 1
### cost: 34
### augmenting matroid flow: 0 0 1 -1 0 0 1 0 0
### number negativ min-mean-circuits: 1
### optimal matroid flow: 3 1 4 0 0 0 3 (4) 1
### optimal cost: 33
```

Lineares Programm

```
min
   +2 f1 +3 f2 +2 f3 +5 f4
   +2 f5 +8 f6 +2 f7 +0 f8 +10 f9
s.t.
   -f1 +f5 +f8 -f9 = 0
   -f2 -f5 +f6 +f9 = 0
   -f3 +f6 +f7 +f9 = 0
    f4 +f7 -f8 +f9 = 0
    f8 = 4
bounds
   0 <= f1 <= 3
   0 <= f2 <= 1
   0 <= f3 <= 12
   0 <= f4 <= 1
   0 <= f5 <= 3
   0 <= f6 <= 2
   0 <= f7 <= 4
   0 <= f8
   0 <= f9 <= 5
end
```

Lösung des linearen Programms mit CPLEX

```
Dual simplex - Optimal:  Objective =  3.3000000000e+01

Variable Name           Solution Value
f1                            3.000000
f2                            1.000000
f3                            4.000000
f7                            3.000000
f8                            4.000000
f9                            1.000000
All other variables in the range 1-9 are 0.
```

A.4. Kographisches Matroid $M(K_5)^* \oplus_2 M(K_{3,3})^*$

Testinstanz

```
9 17

1  0  0  0  0  0  0  0  0 -1  0 -1  0  0  0  0  0
0 -1  0  0  0  0  0  0  0 -1 -1  0  0  0  0  0  0
0  0  1  0  0  0  0  0  0  0 -1  1  0  0  0  0  0
0  0  0  1  0  0  0  0  0 -1  0  0 -1  0  0  0  0
0  0  0  0 -1  0  0  0  0  0 -1  0  1  0  0  0  0
0  0  0  0  0  1  0  0  0  0  0  1 -1  0  0  1 -1
0  0  0  0  0  0  1  0  0  0  0  1 -1  1  0  0 -1
0  0  0  0  0  0  0 -1  0  0  0  0  0  1  1  0 -1
0  0  0  0  0  0  0  0 -1  0  0  0  0  0 -1 -1  1

2 4 5 6 5 7 7 3 5 2 3 4 3 Inf 10 4 4

2 1 3 10 11 3 8 4 2 6 3 4 4 0 7 12 3

14
```

Ausgabe Algorithmus

```
### max matroid flow: 0 0 0 3 3 3 0 3 0 0 0 0 3 (7) 0 4 4
### cost: 156
### augmenting matroid flow: 0 0 0 0 0 4 0 0 4 0 0 0 0 0 0 -4 0
### number negativ min-mean-circuits: 1
### optimal matroid flow: 0 0 0 3 3 7 0 3 4 0 0 0 3 (7) 0 0 4
### optimal cost: 128
```

Lineares Programm

```
min
   +2 f1 +1 f2 +3 f3 +10 f4 +11 f5 +3 f6
   +8 f7 +4 f8 +2 f9 +6 f10 +3 f11 +4 f12
   +4 f13 +0 f14 +7 f15 +12 f16 +3 f17
s.t.
    f1 -f10 -f12 = 0
   -f2 -f10 -f11 = 0
    f3 -f11 +f12 = 0
    f4 -f10 -f13 = 0
   -f5 -f11 +f13 = 0
    f6 +f12 -f13 +f16 -f17 = 0
    f7 +f12 -f13 +f14 -f17 = 0
   -f8 +f14 +f15 -f17 = 0
   -f9 -f15 -f16 +f17 = 0
     f14 = 7
bounds
   0 <= f1 <= 2
   0 <= f2 <= 4
   0 <= f3 <= 5
   0 <= f4 <= 6
   0 <= f5 <= 5
   0 <= f6 <= 7
   0 <= f7 <= 7
   0 <= f8 <= 3
   0 <= f9 <= 5
   0 <= f10 <= 2
   0 <= f11 <= 3
   0 <= f12 <= 4
   0 <= f13 <= 3
   0 <= f14
   0 <= f15 <= 10
   0 <= f16 <= 4
   0 <= f17 <= 4
end
```

Lösung des linearen Programms mit CPLEX

```
Dual simplex - Optimal:  Objective =  1.2800000000e+02

Variable Name           Solution Value
f4                            3.000000
f5                            3.000000
f6                            7.000000
f8                            3.000000
f9                            4.000000
f13                           3.000000
f14                           7.000000
f17                           4.000000
All other variables in the range 1-17 are 0.
```

A.5. Reguläres Matroid $M(K_5) \oplus_2 R_{10} \oplus_2 M(K_{3,3})$

Testinstanz

```
12 25

-1  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  0  0  0  0  0  0  0  0
 0 -1  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  0  0  0  0  0  0  0
 0  0  1  0  0  0  0  0  0  0  0  0  1  0 -1  1  0  0  0  0  0  0  0  0  0
 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 -1  1 -1  1 -1  0  0  0  0  0
 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0 -1  1 -1  1  0 -1  0  0  0  0
 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0 -1  1 -1  0 -1  0  1  0  0  0
 0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0  1  0 -1 -1  0  0  0
 0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0  1 -1 -1  0  0 -1
 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1 -1 -1 -1  0  0
 0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  1 -1 -1 -1  1 -1
 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1 -1  0
 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0 -1  1

10 12 9 14 15 29 3 29 38 19 8 29 27 26 14 11 19 Inf 28 29 19 3 5 5 9

9 7 6 20 4 9 6 14 9 8 9 7 10 11 12 8 9 0 8 9 7 3 3 2 5

18
```

Ausgabe Algorithmus

```
### max matroid flow: 10 12 0 2 2 2 0 0 0 0 0 0 0 0 0 0 10 (12) 0 0 0 0 0 0 0
### cost: 330
### augmenting matroid flow: 0 0 0 -2 -2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0
### number negativ min-mean-circuits: 1
### optimal matroid flow: 10 12 0 0 0 2 2 0 0 0 0 0 0 0 0 0 10 (12) 2 0 0 0 0 0 0
### optimal cost: 310
```

Lineares Programm

```
min
   +9 f1 +7 f2 +6 f3 +20 f4 +4 f5 +9 f6
   +6 f7 +14 f8 +9 f9 +8 f10 +9 f11 +7 f12
   +10 f13 +11 f14 +12 f15 +8 f16 +9 f17 +0 f18
   +8 f19 +9 f20 +7 f21 +3 f22 +3 f23 +2 f24 +5 f25
s.t.
   -f1 +f13 +f14 +f17 = 0
   -f2 +f14 +f15 +f18 = 0
    f3 +f13 -f15 +f16 = 0
    f4 -f16 +f17 -f18 +f19 -f20 = 0
    f5 -f16 +f17 -f18 +f19 -f21 = 0
    f6 -f16 +f17 -f18 -f20 +f22 = 0
   -f7 +f19 -f21 -f22 = 0
   -f8 +f20 -f21 -f22 -f25 = 0
    f9 +f20 -f21 -f22 -f23 = 0
   -f10 +f20 -f21 -f22 -f23 +f24 -f25 = 0
    f11 +f23 -f24 = 0
    f12 -f24 + f25 = 0
    f18 = 12
bounds
   0 <= f1 <= 10
   0 <= f2 <= 12
   0 <= f3 <= 9
   0 <= f4 <= 14
   0 <= f5 <= 15
   0 <= f6 <= 29
   0 <= f7 <= 3
   0 <= f8 <= 29
   0 <= f9 <= 38
   0 <= f10 <= 19
   0 <= f11 <= 8
   0 <= f12 <= 29
   0 <= f13 <= 27
   0 <= f14 <= 26
   0 <= f15 <= 14
   0 <= f16 <= 11
   0 <= f17 <= 19
   0 <= f18
   0 <= f19 <= 28
   0 <= f20 <= 29
   0 <= f21 <= 19
   0 <= f22 <= 3
   0 <= f23 <= 5
   0 <= f24 <= 5
   0 <= f25 <= 9
end
```

Lösung des linearen Programms mit CPLEX

```
Dual simplex - Optimal:  Objective =  3.1000000000e+02

Variable Name           Solution Value
f1                            10.000000
f2                            12.000000
f6                             2.000000
f7                             2.000000
f17                           10.000000
f18                           12.000000
f19                            2.000000
All other variables in the range 1-25 are 0.
```

A.6. Reguläres Matroid $M(K_{3,3})^* \oplus_3 M(K_{3,3})$

Testinstanz

```
6 12

-1  0  0  0  0  0  1 -1  0  1  0  0
 0 -1  0  0  0  0  0 -1  1  1  0  0
 0  0  1  0  0  0  0  0  1  1 -1 -1
 0  0  0 -1  0  0  1  0  0  1 -1  0
 0  0  0  0 -1  0  0  0  0  0  1  1
 0  0  0  0  0 -1  1  0 -1  0  0  1

2 4 6 7 Inf 2 3 8 5 4 10 2

4 5 3 3 0 2 10 20 2 4 8 1

5
```

Ausgabe Algorithmus

```
### max matroid flow: 0 2 0 0 (9) 0 3 7 5 4 7 2
### cost: 264
### augmenting matroid flow: 2 2 0 0 0 0 0 -2 0 0 0 0
### augmenting matroid flow: 0 -2 2 0 0 2 0 0 -2 0 0 0
### number negativ min-mean-circuits: 2
### optimal matroid flow: 2 2 2 0 (9) 2 3 5 3 4 7 2
### optimal cost: 238
```

Lineares Programm

```
min
   +4 f1 +5 f2 +3 f3 +3 f4 +0 f5 +2 f6
   +10 f7 +20 f8 +2 f9 +4 f10 +8 f11 +1 f12
s.t.
   -f1 +f7 -f8 +f10 = 0
   -f2 -f8 +f9 +f10 = 0
    f3 +f9 +f10 -f11 -f12 = 0
   -f4 +f7 +f10 -f11 = 0
   -f5 +f11 +f12 = 0
   -f6 +f7 -f9 +f12 = 0
    f5 = 9
bounds
   0 <= f1 <= 2
   0 <= f2 <= 4
   0 <= f3 <= 6
   0 <= f4 <= 7
   0 <= f5
   0 <= f6 <= 2
   0 <= f7 <= 3
   0 <= f8 <= 8
   0 <= f9 <= 5
   0 <= f10 <= 4
   0 <= f11 <= 10
   0 <= f12 <= 2
end
```

Lösung des linearen Programms mit CPLEX

```
Dual simplex - Optimal:  Objective =  2.3800000000e+02

Variable Name           Solution Value
f1                            2.000000
f2                            2.000000
f3                            2.000000
f5                            9.000000
f6                            2.000000
f7                            3.000000
f8                            5.000000
f9                            3.000000
f10                           4.000000
f11                           7.000000
f12                           2.000000
All other variables in the range 1-12 are 0.
```

A.7. Reguläres Matroid $M(K_5) \oplus_3 M(K_5)^*$

Testinstanz

```
7 14

1  0  0  0  0  0  0 -1 -1 -1  0  0  0  0
0 -1  0  0  0  0  0 -1  0  0 -1  1  0  0
0  0 -1  0  0  0  0  0  0 -1  1  0  1  0
0  0  0 -1  0  0  0  0  1  0  0  1  1  0
0  0  0  0  1  0  0  0  0  1 -1  0  0 -1
0  0  0  0  0  1  0  0  0  0  0  0 -1  1
0  0  0  0  0  0  1  0 -1  0  0 -1  0 -1

2 3 8 5 4 9 10 5 5 2 4 8 9 Inf

1 1 3 2 3 4 5 9 10 2 10 2 10 0

14
```

Ausgabe Algorithmus

```
### max matroid flow: 1 0 4 5 4 0 5 0 0 1 0 0 5 (5)
### cost: 112
### augmenting matroid flow: 1 0 -1 0 -1 0 0 0 0 1 0 0 0 0
### number negativ min-mean-circuits: 1
### optimal matroid flow: 2 0 3 5 3 0 5 0 0 2 0 0 5 (5)
### optimal cost: 109
```

Lineares Programm

```
min
   +1 f1 +1 f2 +3 f3 +2 f4 +3 f5 +4 f6
   +5 f7 +9 f8 +10 f9 +2 f10
   +10 f11 +2 f12 +10 f13 +0 f14
s.t.
    f1 -f8 -f9 -f10 = 0
   -f2 -f8 -f11 +f12 = 0
   -f3 -f10 +f11 +f13 = 0
   -f4 +f9 +f12 +f13 = 0
    f5 +f10 -f11 -f14 = 0
    f6 -f13 +f14 = 0
    f7 -f9 -f12 -f14 = 0
    f14 = 5
bounds
   0 <= f1 <= 2
   0 <= f2 <= 3
   0 <= f3 <= 8
   0 <= f4 <= 5
   0 <= f5 <= 4
   0 <= f6 <= 9
   0 <= f7 <= 10
   0 <= f8 <= 5
   0 <= f9 <= 5
   0 <= f10 <= 2
   0 <= f11 <= 4
   0 <= f12 <= 8
   0 <= f13 <= 9
   0 <= f14
end
```

Lösung des linearen Programms mit CPLEX

```
Dual simplex - Optimal:  Objective =  1.0900000000e+02

Variable Name           Solution Value
f1                            2.000000
f3                            3.000000
f4                            5.000000
f5                            3.000000
f7                            5.000000
f10                           2.000000
f13                           5.000000
f14                           5.000000
All other variables in the range 1-14 are 0.
```

B. Inhalt CD

Auf der CD befindet sich auf der obersten Ebene die Diplomarbeit in elektronischer Form. Die entsprechende Datei ist `Diplomarbeit.pdf`. Dann existieren drei Verzeichnisse. Im Verzeichnis `Algorithmus` befindet sich der Quellcode des in der Arbeit entwickelten Algorithmus. Der Quellcode zum Signieren von regulären Matroiden ist im Verzeichnis `Signing` zu finden. Letztlich gibt es noch einen Ordner `Testinstanzen` in welchem die sieben generierten Testinstanzen und die entsprechenden linearen Programme zur Verifikation abgespeichert sind.