Dissertation zur Erlangung des
Doktorgrades der Technischen Fakultät
der Albert-Ludwigs-Universität
Freiburg im Breisgau.

Dekan:	Prof.Dr Zappe
Vorsitz:	Prof. Lausen
Beisitz:	Prof. Ottmann
Betreuer:	Prof. Albers
Prüfer:	Prof. Srivastav
Abgabe der Promotion:	02.04.2008
Datum der Promotion:	15.05.2009

Approximative Nash-Gleichgewichte in Netzwerk-Spielen

Stefan Eilts

Freiburg, 2. April 2008

In Gedenken an meine Großeltern:

Marga Gesine Eilts, geb. Polter
⋆20.09.1927
†19.08.1999

und

Friedrich Eilts
⋆06.10.1921
†04.11.1994

Danksagung

Ich danke dem Lehrstuhl für tatkräftige Unterstützung dieser Arbeit. Insbesondere im technischen Bereich

- Wolfgang Paulat

und im bürokratischen Bereich

- Birgit Stadel.

Weiterhin danke ich den Mitglieder des Lehrstuhls, die mich immer bei Fragen und Problemen unterstützt haben. In alphabetischer Reihenfolge:

- Hiroshi Fujiwara
- Christian Gunia
- Tobias Jacobs
- Sonja Lauer
- Tim Nonner
- Fabian Schiller
- Markus Schmidt
- Swen Schmelzer
- Sylva Scholz
- Alexander Souza

Für das seelische und leibliche Wohl sorgten meine Familie

- Renate und Harm Eilts
- Bettina Kruse und Thomas Eilts
- Bianca und Hendrik Baumann
- Jutta und Hermann Eilts

Weiterhin danke ich meiner geliebten Freundin

- Birka Schunter und Tochter
- Isabel Schunter

Zusammenfassung

In dieser Doktorarbeit wollen wir Einsicht in die Entstehung großer Netzwerke gewinnen. Das Internet ist ein Produkt vieler Spieler und ist entstanden durch verteilte unkoordinierte Aktionen dieser Spieler. Wir stellen uns die Frage, wie kritisch ist das Fehlen einer zentralen Koordination? Hierfür werden wir das Netzwerk-Erstellungs-Spiel und einige Netzwerk-Spiele betrachten.
In Kapitel 1 werden wir benötigte Grundlagen aus der endlichen Geometrie und aus der Komplexitätstheorie angeben. Dieses Kapitel ist sehr knapp gehalten und der interessierte Leser kann durch die dort angegebene Literatur einen tieferen Einblick erhalten.
Wir werden in Kapitel 2 die wichtige Klasse der Primal-Dual-Algorithmen vorstellen. Diese Klasse enthält für viele Probleme die besten bekannten Approximations-Algorithmen. Weiterhin sind viele bekannte und optimale Greedy-Algorithmen, wie der Algorithmus von Dijkstra, von Edmonds oder von Kruskal Primal-Dual-Algorithmen. Wir werden hauptsächlich Primal-Dual-Algorithmen für Netzwerk-Probleme betrachten. Insbesondere werden wir die Berechnung des kürzesten Weges, des minimalen Spannbaumes, des minimalen Steiner-Baumes und des allgemeineren Steiner-Waldes behandeln. Die Primal-Dual-Algorithmen brauchen wir für die späteren Kapitel 3 und 5.
In Kapitel 3 werden wir einen Gegner für die Prioritäts- und Stapel-Algorithmen einführen, welcher eine untere Schranke für die Approximationsgüte der Algorithmen bestimmt. Neben den im vorherigen Kapitel betrachten Algorithmen gehören auch die Greedy- und Online-Algorithmen zu den Prioritäts- und Stapel-Algorithmen. Wir werden einige kürzeste Weg-Probleme und Spannbaum-Probleme betrachten. Speziell für das Steiner-Wald-Problem werden wir mit Hilfe des Gegners eine untere Schranke von 2 zeigen, welche die in Kapitel 2 gezeigte Approximationsgüte trifft.
Wir werden in Kapitel 4 das Netzwerk-Erstellungs-Spiel von Fabrikant und andere betrachten. Jeder Spieler kann Kanten mit Kosten α bauen und möchte die Distanz zu allen anderen Spielern minimieren. Die Autoren konnten nur für kleine α Nash-Gleichgewichte finden, welche keine Bäume sind. Wir werden auch für große α Nash-Gleichgewichte angeben. Weiter stellten die Autoren die Vermutung auf, dass für große α jedes nichttransitive Nash-Gleichgewicht ein Baum ist. Ein transitives Nash-Gleichgewicht ist ein schwaches Nash-Gleichgewicht, welches eine Folge von Strategiewechsel zulässt, die zu einem nicht Nash-Gleichgewichtszustand führt. Wir werden zeigen, dass die Baumvermutung falsch ist und das für große α starke Nash-Gleichgewichte existieren, die keine Strategiewechsel zulassen.
In Kapitel 5 werden wir einige Netzwerk-Spiele betrachten. Die Autoren Anshelevich und andere haben für das Teilspiel des Steiner-Waldes gezeigt, dass die globale optimale Lösung ein 3-approximatives Nash-Gleichgewicht ist. Weiterhin wurde gezeigt, dass in diesem Teilspiel ein 4.65-approximatives Nash-Gleichgewicht existiert, welches eine globale 2-Approximation ist. Wir werden für ein allgemeineres Spiel zeigen, dass die globale 2-Approximation des Primal-Dual-Algorithmuses ein 3-approximatives Nash-Gleichgewicht ist und einen Algorithmus zur Berechnung angeben. Unser Ergebnis ist viel allgemeiner und zeigt für Netzwerk-Spiele, in denen der Primal-Dual-Algorithmus aus Kapitel 2 eine α-Approximation berechnet, die Existenz eines $2\alpha - 1$-approximatives Nash-Gleichgewichts.
In der Umkehrung werden wir für das Steiner-Wald-Spiel zeigen, dass der Primal-Dual-Algorithmus im Allgemeinen kein α-approximatives Nash-Gleichgewicht für $\alpha < 3$ berechnet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Grundlagen

1.1 Endliche Geometrie

In diesem Kapitel betrachten wir kurz einige Grundlagen, die wir später benötigen. Zunächst werden wir Grundlagen aus der endlichen Geometrie betrachten. Wir haben die Quellen [Beu82], [Bro95], [BCN89], [Jun94], [MS78], [Ore62] und [Sca86] benutzt, die dieses Thema ausführlicher betrachten.

Definition 1.1 (Inzidenzstruktur)**.** *Das Tripel $(X, \mathcal{B}, I)$ ist genau dann eine **Inzidenzstruktur**, wenn X,$\mathcal{B}$ zwei disjunkte Mengen sind und $I \subseteq X \times \mathcal{B}$ ist. Wir benutzen für die Relation I die Infix-Notation. Die Elemente von X heißen je nach Situation **Punkte** oder **Knoten** und die Elemente von $\mathcal{B}$ heißen je nach Situation **Blöcke**, **Geraden** oder **Kanten**. Falls X und $\mathcal{B}$ endlich sind, so nennen wir auch die Inzidenzstruktur **endlich**.*

Definition 1.2 (kollinear, adjazent, inzident)**.** *Es sei $(X, \mathcal{B}, I)$ eine Inzidenzstruktur. Die Menge $Y \subseteq X$ ist genau dann **kollinear**, wenn ein Block B mit*

$$YIB$$

existiert. Es ist genau dann YIB, wenn

$$\forall y \in Y : yIB$$

*ist. Zwei verschiedene Punkte x, y heißen **adjazent**, wenn die Menge $\{x, y\}$ kollinear ist. Ein Punkt x und ein Block B heißen **inzident**, wenn xIB ist. Zwei verschiedene Blöcke B, B' heißen **adjazent**, falls beide Blöcke einen gemeinsamen inzidenten Punkt besitzen. Für eine Menge $Y \subseteq X$ bezeichne (Y) die Menge aller Blöcke, die zu jedem $y \in Y$ inzident sind und für eine Menge von Blöcken $\mathcal{A} \subseteq \mathcal{B}$ bezeichne $(\mathcal{A})$ die Menge aller Punkte, die zu jedem $A \in \mathcal{A}$ inzident sind.*

Falls x und B inzident sind, so sagen wir, dass der Punkt x auf der Geraden B liegt, dass die Gerade B den Punkt x enthält oder ähnliches.

Definition 1.3 (Homomorphismus)**.** *Es seien $(X, \mathcal{B}, I)$ und $(X', \mathcal{B}', I')$ Inzidenzstrukturen. Eine Abbildung*

$$f : X \cup \mathcal{B} \to X' \cup \mathcal{B}'$$

*heißt **Homomorphismus**, wir benutzen die Schreibweise*

$$f : (X, \mathcal{B}, I) \to (X', \mathcal{B}', I'),$$

falls

$$f|_X \colon X \to X' \text{ und } f|_{\mathcal{B}} \colon \mathcal{B} \to \mathcal{B}'$$

ist und für jeden Punkt $x \in X$ und jeden Block $B \in \mathcal{B}$ gilt

$$xIB \Rightarrow f(x)I'f(B).$$

*Falls f zusätzlich bijektiv und f^{-1} auch ein Homomorphismus ist, so nennen wir f einen **Isomorphismus** von $(X, \mathcal{B}, I)$ auf $(X', \mathcal{B}', I')$. Jeder Isomorphismus f mit $(X, \mathcal{B}, I) = (X', \mathcal{B}', I')$ ist ein **Automorphismus**. Die Menge der Automorphismen der Inzidenzstruktur bezeichnen wir mit* $\mathrm{Aut}((X, \mathcal{B}, I))$.

Definition 1.4 (knotenautomorph)**.** *Eine Inzidenzstruktur $(X, \mathcal{B}, I)$ heißt* ***knotenautomorph****, falls zu je zwei Knoten u und v ein $f \in \mathrm{Aut}((X, \mathcal{B}, I))$ mit $f(u) = v$ existiert.*

Definition 1.5 (Design)**.** *Das Paar $(X, \mathcal{B})$ ist genau dann ein* ***Design****, wenn $(X, \mathcal{B}, \in)$ eine Inzidenzstruktur mit $\mathcal{B} \subseteq \mathcal{P}(X)$ ist.*

Definition 1.6 (Graph)**.** *Ein endliches Design $G = (V, E)$ ist genau dann ein* ***ungerichteter Graph*** *(kurz* ***Graph****), wenn für alle Kanten e die Bedingung*

$$1 \leq |e| \leq 2$$

gilt. Im Fall $|e| = 2$ für alle Kanten e nennen wir G ***schlingenfrei****.*

Beispiel 1.7 (vollständiger Graph)**.** *Es sei V eine Menge und $E = \{\{u, v\} \mid u, v \in V\}$. Das Paar (V, E) ist ein schlingenfreier Graph. Da alle möglichen Kanten mit zwei Punkten existieren, nennen wir den Graphen* ***vollständig****. Alle vollständigen Graphen mit $|V| = n$ vielen Knoten sind zueinander isomorph. Wir schreiben K_n für den vollständigen Graphen mit n Knoten. K_n ist knotenautomorph.*

Definition 1.8 (Kantenfolge, Kantenzug, Weg)**.** *Es sei $G = (V, E)$ ein Graph und u, v Knoten. Eine* ***Kantenfolge*** *von u nach v ist eine endliche Folge von Kanten $(\{u_0, u_1\}, \{u_1, u_2\}, \ldots, \{u_{n-1}, u_n\})$ mit $u = u_0$ und $v = u_n$. Ein* ***Kantenzug*** *von u nach v ist eine Kantenfolge von u nach v, bei der alle Kanten paarweise verschieden sind. Ein* ***Weg*** *von u nach v ist ein Kantenzug von u nach v, in dem die Knoten $u_0, \ldots, u_n$ paarweise verschieden sind.*

Wir werden einen Weg $(\{u_0, u_1\}, \{u_1, u_2\}, \ldots, \{u_{n-1}, u_n\})$ häufig durch den Vektor aller besuchten Knoten $(u_0, u_1, \ldots, u_n)$ angeben.

Definition 1.9 (zusammenhängend, Zusammenhangskomponente)**.** *Es sei $G = (V, E)$ ein Graph. Die Knoten u und v heißen* ***zusammenhängend****, wenn es in G einen Weg von u nach v gibt. Eine Knotenmenge $V' \subseteq V$ heißt* ***zusammenhängend****, falls jedes Knotenpaar $u, v \in V'$ zusammenhängend ist. Die Knotenmenge C ist eine* ***Zusammenhangskomponente*** *von G, wenn C zusammenhängend ist und für alle Knoten $v \notin C$ ist $C \cup \{v\}$ nicht zusammenhängend. Der Graph G heißt* ***zusammenhängend****, wenn V eine Zusammenhangskomponente ist.*

Definition 1.10 (gerichteter Graph)**.** *Das Tupel (V, E) ist ein* ***gerichteter Graph****, falls $E \subseteq V \times V$ und V endlich ist. Für eine Kante $e = (v, w) \in E$ heißt $v = \pi_1(e)$ der* ***Anfangs-*** *und $w = \pi_2(e)$ der* ***Endpunkt*** *von e. (π_i ist die i-te Koordinatenprojektion von $V \times V$ auf V.)*

Ein gerichteter Graph besteht aus den beiden Inzidenzstrukturen (V, E, I_i) (i=1,2), mit xI_ie genau dann, wenn $x = \pi_i(e)$ ist. Für gerichtete Graphen ist eine **Kantenfolge** von u nach v ein Kantenvektor $((u_0, u_1), (u_1, u_2), \ldots, (u_{n-1}, u_n))$ mit $u = u_0$ und $v = u_n$. Ein **Kantenzug** und ein **Weg** im gerichteten Graphen definieren wir analog zur Definition 1.8. Eine **starke Zusammenhangskomponente** C ist ein Menge von Knoten, so dass jedes Knotenpaar u, v in C ein Weg von u nach v besitzt und C ist bzgl. der Inklusion maximal mit dieser Eigenschaft.

Definition 1.11 (Schnittkante)**.** *Für einen ungerichteten Graph $G = (V, E)$ und eine Menge $S \subseteq V$ ist eine Kante e mit $|e \cap S| = 1$ eine* ***Schnittkante*** *von S. Die Menge der Schnittkanten von S bezeichnen wir mit $\delta_G(S)$. Für $F \subseteq E$ definieren wir $\delta_F(S) := \delta_{(V,F)}(S)$.*
Für einen gerichteten Graph $G = (V, E)$ und eine Knotenmenge S definieren wir

$$\delta_G^+(S) := \{e = (u, v) \in E \mid u \in S, v \notin S\}$$

und

$$\delta_G^-(S) := \{e = (u, v) \in E \mid u \notin S, v \in S\}.$$

Falls klar ist, welchen Graph G wir betrachten, kürzen wir $\delta_G(S)$, $\delta_G^+(S)$ und $\delta_G^-(S)$ mit $\delta(S)$, $\delta^+(S)$ und $\delta^-(S)$ ab. Weiterhin kürzen wir für einelementig Mengen $S = \{v\}$, $\delta_G(\{v\})$, $\delta_G^+(\{v\})$ und $\delta_G^-(\{v\})$ mit $\delta_G(v)$, $\delta_G^+(v)$ und $\delta_G^-(v)$ ab. Wir streichen den Index G, falls klar ist, welchen Graphen wir betrachten.

Definition 1.12 (Netzwerk)**.** *Das Tupel $G = (V, E, c)$ ist genau dann ein* ***Netzwerk****, wenn (V, E) ein Graph und c eine nicht negative Funktion der Kanten in den Raum der reellen Zahlen ist.*

Beispiel 1.13 (kanonisches Netzwerk). *Es sei $G = (V, E)$ ein Graph. Durch*

$$c : E \to \mathbb{R}_0^+, e \mapsto 1$$

wird ein Netzwerk $G = (V, E, c)$ definiert. Wir nennen es das ***kanonische Netzwerk*** *zu G.*

Definition 1.14 (Distanz, kürzester Weg). *Für zwei Knoten u, v eines Netzwerkes $G = (V, E, c)$ definieren wir die Distanzfunktion*

$$d(u,v) := \begin{cases} \min\{\sum_{i=1}^{n} c_{e_i} \mid (e_1, \ldots, e_n) \text{ ist ein Weg von } u \text{ nach } v\} & \text{falls } u, v \text{ zusammenhängend sind} \\ \infty & \text{sonst.} \end{cases}$$

Ein Weg $(e_1, \ldots, e_n)$ von u nach v mit $\sum_{i=1}^{n} c_{e_i} = d(u, v)$ heißt ***kürzester Weg****.*

Definition 1.15 (metrisches Netzwerk). *Ein vollständiges Netzwerk $G = (V, E, c)$ ist genau dann* ***metrisch****, wenn (V, d) ein metrischer Raum für die obige Distanzfunktion d ist.*

Definition 1.16 (Durchmesser). *Es sei $G = (V, E)$ ein Graph und (V, E, c) das zugehörige kanonische Netzwerk. Die Distanz des längsten kürzesten Weg ist der* ***Durchmesser*** *des Graphen.*

Definition 1.17 ((partieller) Parallelismus). *Es sei $(X, \mathcal{B}, I)$ eine Inzidenzstruktur. Eine reflexive und symmetrische Relation $\parallel$ auf $\mathcal{B} \times \mathcal{B}$ ist genau dann ein* ***partieller Parallelismus****, wenn es zu jedem Block B und jedem Punkt x genau ein Block $B' \parallel B$ existiert, der mit x inzident ist. Wir schreiben $(x \parallel B)$ für diesen Block B'. Die Blöcke B_1, B_2 heißen* ***parallel*** *, falls $B_1 \parallel B_2$ ist.*
Ein transitiver partieller Parallelismus heißt ***Parallelismus****. Mit*

$$[B] := \{B' \in \mathcal{B} \mid B' \parallel B\}$$

bezeichnen wir in diesem Fall die Äquivalenzklasse des Blockes B.

Lemma 1.18. *Es sei $(X, \mathcal{B}, I)$ eine Inzidenzstruktur mit partiellem Parallelismus $\parallel$. Zwei parallele Böcke B und B' sind entweder gleich oder* ***disjunkt****, das heißt sie haben keinen gemeinsamen Punkt. Wir schreiben oft $B \cap B' = \emptyset$ für disjunkte Blöcke B, B'. Zwei disjunkte, nicht parallele Blöcke heißen* ***windschief****.*

Beweis: Es seien die beiden Blöcke B und B' parallel. Falls B und B' den gemeinsamen Punkt x enthalten, so ist $B = (x \parallel B) = (x \parallel B') = B'$. □

Definition 1.19 (affine Ebene). *Ein Design $(X, \mathcal{B})$ ist eine* ***affine Ebene*** *genau dann, wenn die folgenden drei Punkte erfüllt sind*

1. *Je zwei Punkte liegen auf genau einer Geraden.*
2. *Die Relation $B \parallel B'$ gelte genau dann, wenn $B \cap B' = \emptyset$ oder $B = B'$ ist. $\parallel$ ist ein Parallelismus.*
3. *Es existiert ein* ***Dreieck****, das heißt drei Punkte, die nicht kollinear sind. Jede Gerade hat mindestens zwei Punkte.*

Wie in Definition 1.2 bezeichnen wir die eindeutige Gerade durch die zwei Punkte x, y mit (x, y).

Definition 1.20 (Steiner-System). *Ein Design $(X, \mathcal{B})$ ist ein $\mathcal{S}(t, k, v)$-****Steiner-System****, wenn die folgenden drei Bedingungen gelten.*

1. *Je t paarweise verschiedene Punkte sind in einem eindeutigen Block.*
2. *Jeder Block enthält genau k viele Punkte.*
3. *Es existieren genau n viele Punkte.*

Definition 1.21 (Blockzerlegungszahl)**.** *Es sei $(X, \mathcal{B})$ ein $\mathcal{S}(t,k,v)$-Steiner-System. Für paarweise verschiedene Punkte $x_1, ..., x_t$ sei die* ***Blockzerlegungszahl*** *λ_i die Anzahl der Blöcke, die die Punkte $x_1, \ldots, x_i$ enthalten ($i = 0, \ldots, t$). Für $t \leq i \leq k$ sei $\lambda_i := 1$.*
Weiter sei für einen Block $\{x_1, \ldots, x_k\}$ und $0 \leq i, j \leq k$ die ***Blockzerlegungszahl***

$$\lambda_{i,j} = |\{B \in \mathcal{B} \mid x_1, ..., x_j \in B \text{ und } x_{j+1}, ..., x_i \notin B\}|.$$

Lemma 1.22. *Es sei $(X, \mathcal{B})$ ein $\mathcal{S}(t,k,v)$-Steiner-System. Die Blockzerlegungszahl λ_i ist für $1 \leq i \leq t$ unabhängig von der Wahl und Reihenfolge der Punkte und es gilt*

$$\lambda_i = \lambda_{i,i} = \frac{\binom{v-i}{t-i}}{\binom{k-i}{t-i}}.$$

Es ist $\lambda_i = \lambda_{i,i} = 1$ für $t \leq i \leq k$. Weiterhin ist die Blockzerlegungszahl $\lambda_{i,j}$ wohldefiniert, das heißt unabhängig von der Wahl des Blockes und der Reihenfolge der Punkte im Block. $\lambda_{i,j}$ erfüllt die rekursive Gleichung:

$$\lambda_{i,j} = \lambda_{i+1,j} + \lambda_{i+1,j+1}.$$

Beweis: Durch Induktion über i. Nach Definition des Steiner-Systems ist

$$\lambda_t = \lambda_{t,t} = 1 = \frac{\binom{v-t}{0}}{\binom{k-t}{0}}.$$

Wir zeigen nun den Induktionsschritt von $i+1$ auf i. Zu den gegebenen Punkten $x_1, \ldots, x_i$ können wir $v - i$ viele verschiedene Punkte x_{i+1} wählen. Die Anzahl der Blöcke, die die Punkte $x_1, .., x_i, x_{i+1}$ enthalten, beträgt λ_{i+1} Für einen Block $B = \{x_1, \ldots, x_i, x_{i+1}, \ldots, x_k\}$ würde auch die Wahl eines der Knoten $x_{i+2}, \ldots, x_k$ den Block B ergeben. Somit ist

$$\lambda_i = \lambda_{i,i} = \frac{v-i}{k-i}\lambda_{i+1} = \frac{v-i}{k-i}\frac{(v-i-1)!(t-i-1)!(k-t)!}{(t-i-1)!(v-t)!(k-i-1)!} = \frac{\binom{v-i}{t-i}}{\binom{k-i}{t-i}}$$

Es sei nun $\{x_1, ..., x_k\}$ ein Block. Es ist

$$\begin{aligned}\lambda_{i,j} &= |\{B \in \mathcal{B} \mid x_1, ..., x_j \in B, x_{j+1}, .., x_i \notin B\}| \\ &= |\{B \in \mathcal{B} \mid x_1, ..., x_j \in B, x_{j+1}, .., x_i, x_{i+1} \notin B\}| + |\{B \in \mathcal{B} \mid x_1, ..., x_j, x_{i+1} \in B, x_{j+1}, .., x_i \notin B\}| \\ &= \lambda_{i+1,j} + \lambda_{i+1,j+1}.\end{aligned}$$

Aus den Formel folgt die Wohldefiniertheit der λ_i und $\lambda_{i,j}$. □

Die folgenden Blockzerlegungszahlen sind von speziellem Interesse.

- λ_0 ist die Anzahl der Blöcke.
- λ_1 ist die Anzahl der Blöcke, die einen bestimmten Punkt enthalten.
- $\lambda_{1,0}$ ist die Anzahl der Blöcke, die einen bestimmten Punkt nicht enthalten.
- $\lambda_{k,0}$ ist die Anzahl der Blöcke, die zu einem bestimmten Block disjunkt sind.

Wir werden uns hauptsächlich für $\mathcal{S}(2, q, q^2)$-Steiner-Systeme interessieren.

Beispiel 1.23. *Es sei $q \geq 2$ Die Blockzerlegungszahlen des $\mathcal{S}(2, q, q^2)$-Steiner-Systems sind*

$$\begin{array}{llll} \lambda_0 = q(q+1) & & & \\ \lambda_{1,0} = (q-1)(q+1) & \lambda_1 = q+1 & & \\ \lambda_{2,0} = q^2 - q - 1 & \lambda_{2,1} = q & \lambda_2 = 1 & \\ \cdots & \cdots & \cdots & \\ \lambda_{q,0} = q-1 & \lambda_{q,1} = q & \vdots & \lambda_{q,q} = 1. \end{array}$$

Satz 1.24. *A ist genau dann eine endliche affine Ebene, wenn A ein $\mathcal{S}(2, q, q^2)$-Steiner-System für $q \geq 2$ ist. Wir nennen q die* ***Ordnung*** *der affinen Ebene.*

Beweis: Der Beweis ist umfangreich und kann in [MS78] oder [Beu82] nachgelesen werden. □

Die folgende Tabelle zeigt die Anzahl der affinen Ebenen der Ordnung q für kleine q. Sie stammt von [Bro95].

Ordnung q	2	3	4	5	6	7	8	9	10	11	12
Anzahl	1	1	1	1	-	1	1	≥ 7	-	≥ 1	?

Wir werden später affine Ebenen mit größeren Ordnungen benötigen.

Lemma 1.25 (affine Koordinatenebene)**.** *Es sei K ein endlicher Körper. Die Mengen $X := K^2$ und $\mathcal{B} := \{a + bK \mid a, b \in X, b \neq 0\}$ definieren eine affine Ebene $(X, \mathcal{B})$.*

Beweis: Wir zeigen, dass $(X, \mathcal{B})$ ein $(2, q, q^2)$-Steiner-System mit $q := |K|$ ist. Es seien $x, y \in X$ zwei verschiedene Punkte. Die Punkte liegen auf der Gerade $x + (y - x)K$ mit $y - x \neq 0$. Es sei $a + bK$ eine weitere Gerade, die x und y enthält. Wir schreiben $x = a + \alpha b$ und $y = a + \beta b$ mit $\alpha, \beta \in K$. Es ist

$$y - x = a + \beta b - (a + \alpha b) = (\beta - \alpha)b.$$

Da $\alpha \neq \beta$ ist, folgt

$$(y - x)K = bK.$$

Weiterhin ist

$$x + (y - x)K = x + bK = a + b\alpha + bK = a + bK.$$

Somit ist die Gerade, die die Punkte x und y enthält, eindeutig.
Da $|K| = q$ ist, ist auch $|a + bK| = q$ für alle $b \neq 0$. Weiterhin ist $|X| = |K^2| = q^2$. □

Für alle Primzahlen p und $n \in \mathbb{N}$ existiert genau ein Körper $\mathbb{F}_q$ mit $q = p^n$ vielen Elementen. Daher existiert auch eine affine Ebene der Ordnung q.

Definition 1.26 (π-Raum)**.** *Ein π-Raum ist ein Tupel $(X, \mathcal{B}, I, \|)$ mit den folgenden Eigenschaften:*

- *$(X, \mathcal{B}, I)$ ist eine Inzidenzstruktur.*
- *Je zwei Punkte liegen auf genau einer Geraden.*
- *$\|$ ist ein partieller Parallelismus.*
- *Es seien die beiden Geraden B und B' nicht parallel. Entweder sind B und B' mit einem Punkt inzident oder es existiert genau eine Gerade, welche zu beiden Geraden parallel ist.*
- *Es existiert ein Dreieck und jede Gerade ist mindesten zu einer anderen Geraden parallel.*

Beispiel 1.27. *Jede affine Ebene ist ein π-Raum.*

Beweis: Es sei $(X, \mathcal{B})$ eine affine Ebene. Falls die zwei Geraden B und B' nicht parallel sind, dann ist $B \cap B' \neq \emptyset$ und die beiden Geraden besitzen einen gemeinsamen Punkt. Da ein Dreieck existiert, ist jede Gerade mindestens zu einer anderen Geraden parallel. Die restlichen Eigenschaften sind trivial. □

Definition 1.28 (π-Graph)**.** *Es sei $(X, \mathcal{B}, I, \|)$ ein π-Raum. $G = (V, E)$ mit $V = X \cup \mathcal{B}$ und*

$$E = \{\{x, B\} \mid x \in X, B \in \mathcal{B}, xIB\} \cup \{\{B, B'\} \mid B, B' \in \mathcal{B}, B \parallel B', B \neq B'\}$$

*ist ein Graph und heißt der π-****Graph*** *des π-Raumes $(X, \mathcal{B}, I, \|)$.*

Definition 1.29 (geodätisch)**.** *Ein ungerichteter Graph G ist genau dann* ***geodätisch****, wenn jedes Punktepaar genau einen kürzesten Weg besitzt.*

Lemma 1.30. *Der π-Graph G des π-Raumes $(X, \mathcal{B}, I, \|)$ ist geodätisch und hat den Durchmesser 2.*

Beweis: Wir betrachten zwei verschiedene Knoten des Graphens und bestimmen den kürzesten Weg.

1. Es seien zwei Punkte $x, y \in X$ gegeben. Die beiden Punkte x und y sind in G nicht direkt verbunden. Es existiert genau eine Gerade $B = (x, y)$ mit xIB und yIB. Der eindeutige kürzeste Weg ist (x, B, y).

2. Es sei nun ein Punkt $x \in X$ und eine Gerade $B \in \mathcal{B}$ gegeben. Fall $x \in B$ ist, dann sind x und B in G direkt verbunden. Im anderen Fall sind x und B über die eindeutige Gerade $(x \parallel B)$ in G verbunden.

3. Es seien nun die Geraden B und B' gegeben. Falls $B \parallel B'$ ist, dann sind B und B' direkt verbunden. Im anderen Fall haben B und B' entweder einen gemeinsamen Punkt x oder es existiert genau eine Gerade, die zu beiden Geraden parallel ist.

   Da B und B' verschieden sind, können sie nicht zwei gemeinsame Punkte besitzen.

Da ein Dreieck existiert, gibt es auch wirklich Knoten in G, die die Distanz 2 haben. □

1.2 Optimierungsprobleme und Komplexitätsklassen

In diesem Abschnitt betrachten wie Optimierungsprobleme und deren zugehörigen Komplexitätsklassen. Es wird davon ausgegangen, dass der Leser Grundkenntnisse der theoretischen Informatik besitzt und Begriffe wie Algorithmus, berechenbar, $\mathcal{P}$, $\mathcal{NP}$ usw. kennt. Als Literatur haben wir [ACG+99], [Hro03] und [BEY98] verwendet.

Definition 1.31 (Optimierungsproblem)**.** *Ein Optimierungsproblem P ist ein Tupel $(\mathcal{I}, \text{Lsg}, c, \text{ziel})$, wobei*

1. *$\mathcal{I}$ ein Menge von **Instanzen** ist.*
2. *Lsg ordnet jeder Instanz $I \in \mathcal{I}$ eine Menge von **zulässigen Lösungen** $\text{Lsg}(I)$ zu.*
3. *c ist eine **Wertfunktion** oder **Maßfunktion**, die jedem Paar (x, I) mit $I \in \mathcal{I}$ und $x \in \text{Lsg}(I)$ eine reelle Zahl zuordnet.*
4. *$\text{ziel} \in \{\min, \max\}$.*

Im Fall $\text{ziel} = \min$ *heißt P **Minimierungsproblem** und im Fall* $\text{ziel} = \max$ *heißt P **Maximierungsproblem**.*

Falls klar ist, welche Instanz I wir meinen, kürzen wir $\text{Lsg}(I)$ und $c(\cdot, I)$ mit Lsg und $c(\cdot)$ ab. Weiterhin kürzen wir oft zulässige Lösung mit Lösung ab. Ein Algorithmus ALG zum Optimierungsproblem berechnet zu einer Instanz I eine zulässige Lösung $ALG(I) \in \text{Lsg}(I)$.

Definition 1.32 (optimale Lösung)**.** *Es sei ein Optimierungsproblem $P = (\mathcal{I}, \text{Lsg}, c, \text{ziel})$ gegeben. Eine zulässige Lösung $x \in \text{Lsg}(I)$ der Instanz $I \in \mathcal{I}$ heißt **optimal**, falls*

$$c(x, I) = \text{opt} := \text{opt}(I) := \inf_{y \in Sol(I)} c(y, I)$$

für ein Minimierungsproblem bzw.

$$c(x, I) = \text{opt} := \text{opt}(I) := \sup_{y \in Sol(I)} c(y, I)$$

für ein Maximierungsproblem ist. Wir bezeichnen die optimale Lösung mit $\text{OPT}(I)$*. Falls ein Algorithmus existiert, der zu jeder Instanz I eine optimale Lösung* $\text{OPT}(I)$ *berechnet, so bezeichnen wir diesen Algorithmus mit* OPT*.*

Zu einem Optimierungsproblem $P = (\mathcal{I}, \text{Lsg}, c, \text{ziel})$ gehören die drei folgenden Probleme.

Definition 1.33 (Konstruktionsproblem P_K)**.** *Finde zu einer Instanz $I \in \mathcal{I}$ eine optimale Lösung* $\text{OPT}(I)$ *und den optimalen Wert* $\text{opt}(I)$*, falls eine optimale Lösung existiert.*

Definition 1.34 (Wertproblem P_W)**.** *Finde zu einer Instanz $I \in \mathcal{I}$ den optimalen Wert* $\text{opt}(I)$*.*

Definition 1.35 (Entscheidungsproblem P_E)**.** *Entscheide zu einer Instanz I und einer Zahl a, ob es eine Lösung $x \in \mathrm{Lsg}(I)$ mit*

$$c(x, I) \leq a$$

für Minimierungsprobleme bzw.

$$c(x, I) \geq a$$

für Maximierungsprobleme existiert.

Definition 1.36 (nichtdeterministisches Optimierungsproblem)**.** *Ein gegebenes Optimierungsproblem $(\mathcal{I}, \mathrm{Lsg}, c, \mathrm{ziel})$ ist ein **nichtdeterministisches Optimierungsproblem**, falls die folgenden Bedingungen gelten.*

1. *Die Menge der Instanzen $\mathcal{I}$ ist in Polynomialzeit entscheidbar.*
2. *Es existiert ein Polynom p, so dass für jede Instanz $I \in \mathcal{I}$*

   $$x \in \mathrm{Lsg}(I) \Rightarrow |x| \leq p(|I|)$$

   gilt, wobei $|\cdot|$ die Länge der Kodierung ist. Weiterhin ist für alle y mit $|y| \leq p(|I|)$ in Polynomialzeit entscheidbar, ob $y \in \mathrm{Lsg}(I)$.
3. *Die Wertefunktion c ist in Polynomialzeit berechenbar.*

Wir bezeichnen die Klasse der nichtdeterministische Optimierungsprobleme mit $\mathcal{NPO}$.

Die Namensgebung $\mathcal{NPO}$ stammt von dem folgenden Lemma:

Lemma 1.37. *Es sei $P = (\mathcal{I}, \mathrm{Lsg}, c, \mathrm{ziel})$ ein nichtdeterministisches Optimierungsproblem. Das zugehörige Entscheidungsproblem P_E ist in $\mathcal{NP}$.*

Beweis: Wir zeigen es nur für Minimierungsprobleme. Es sei (I, a) eine Instanz des Entscheidungsproblems P_E. Wir konstruieren einen nichtdeterministischen Algorithmus in Polynomialzeit. Der Algorithmus prüft zuerst in Polynomialzeit ob I eine Instanz des Optimierungsproblems und a eine reelle Zahl ist. Falls mindestens eine Antwort Nein ist, dann ist die Instanz (I, a) unzulässig.
Im Ja-Fall ist die Instanz (I, a) für P_E zulässig. Der Algorithmus wählt nun zufällig ein y mit $|y| \leq p(|I|)$ und entscheidet in Polynomialzeit, ob $y \in \mathrm{Lsg}(I)$. Falls $y \notin \mathrm{Lsg}(I)$, dann gibt der Algorithmus Nein aus. Im anderen Fall berechnet der Algorithmus den Wert $c(y)$ in Polynomialzeit. Falls $c(y) \leq a$ gibt der Algorithmus Ja, sonst Nein aus.
Falls eine zulässige Lösung $x \in \mathrm{Lsg}(I)$ mit $c(x) \leq a$ existiert, dann gibt es einen Berechnungspfad, der Ja ausgibt und somit gibt es einen akzeptierenden Berechnungspfad. □

Lemma 1.38. *Es sei $P = (\mathcal{I}, \mathrm{Lsg}, c, \mathrm{ziel}) \in \mathcal{NPO}$. Dann existiert der optimale Algorithmus* OPT.

Beweis: Für jede Instanz I hat jede zulässige Lösung $x \in \mathrm{Lsg}(I)$ ein maximale Länge von $p(|I|)$. Der Algorithmus OPT kann für die endlich vielen Wörter y mit $|y| \leq p(|I|)$ entscheiden, ob $y \in \mathrm{Lsg}(I)$ und im Ja-Fall den Wert $c(y)$ berechnen. Anschließend wählt OPT die zulässige Lösung x mit dem optimalen Wert $c(x) = \mathrm{opt}(I)$ aus. □

Beispiel 1.39 (Ganzzahlige Programmierung IP)**.** *Es sei p ein Polynom. Eine Instanz I des ganzzahligen linearen Programms* IP *ist*

$$\begin{aligned} c \cdot x &\to \min \\ Ax &\geq b \\ x &\in \mathbb{N}_0^n \end{aligned}$$

mit einer ganzzahligen Matrix A und ganzzahligen Vektoren b und c. Weiterhin betrachten wir nur Lösungen x des obigen Programms mit

$$|x| \leq p(|I|). \tag{1.1}$$

Dieses Problem ist in $\mathcal{NPO}$.

Beweis: Wir zeigen, dass es ein nichtdeterministisches Optimierungsproblem $(\mathcal{I}, \text{Lsg}, d, \text{ziel})$ ist. Es ist

$$\mathcal{I} = \{(A, b, c) \mid A \text{ ist eine ganzzahlige Matrix und } b, c \text{ sind ganzzahlige Vektoren}\}.$$

Die Menge $\mathcal{I}$ ist offensichtlich in Polynomialzeit entscheidbar. Für $I = (A, b, c)$ ist

$$\text{Lsg}(I) = \{x \mid Ax \geq b, x \in \mathbb{N}_0^n\} \cap \{x \mid |x| \leq p(|I|)\}.$$

Für $x \in \text{Lsg}(I)$ ist $|x| \leq p(|I|)$ und für alle y mit $|y| \leq p(|I|)$ können wir in Polynomialzeit prüfen, ob $Ay \geq b$, $y \in \mathbb{N}_0^n$ gilt. Die Wertfunktion $d(x, I) = c \cdot x$ ist in Polynomialzeit berechenbar und es ist ziel = min. □

Analoges kann man für Maximierungsprobleme IP zeigen. In der Praxis wird häufig durch die Vorgabe einer Genauigkeit oder durch einen endlichen Zahlenbereich die Bedingung 1.1 definiert. Die Autoren Orponen und Mannila [OM87] haben gezeigt, das obiges Problem sogar $\mathcal{NPO}$-vollständig ist. Wir werden Optimierungsprobleme nicht durch explizite Auflistung des Tupels $(\mathcal{I}, \text{Lsg}, c, \text{ziel})$ angeben. Wir werden oft das folgende Format verwenden, welches wir am Beispiel des IP demonstrieren.

Problem 1.40 (IP)**.**
Instanz*: Eine ganzzahlige Matrix A und zwei ganzzahlige Vektoren b, c.*
Lösung*: Ein x mit $Ax \geq b$, $x \in \mathbb{N}_0^n$.*
Maß*: $c \cdot x$.*
Ziel*:* min.

Das Wort Instanz beschreibt eine typische Instanz $I \in \mathcal{I}$. Das Wort Lösung beschreibt eine typische Lösung der Instanz I, Maß ist die Wertfunktion und Ziel ist gleich ziel.

Definition 1.41 (Approximation)**.** *Es sei $P = (\mathcal{I}, \text{Lsg}, c, \text{ziel})$ ein nichtdeterministisches Optimierungsproblem mit $c \geq 0$ und es sei $\alpha \geq 1$. Die Lösung x der Instanz I ist eine α-**Approximation**, falls eine reelle Zahl $\alpha \geq 1$ mit*

$$c(x) \leq \alpha \operatorname{opt}(I)$$

für Minimierungsprobleme bzw.

$$c(x) \geq \frac{1}{\alpha} \operatorname{opt}(I)$$

für Maximierungsprobleme existiert. Ein Algorithmus ALG *heißt α-**Approximationsalgorithmus**, falls ein Funktion*

$$\alpha : \mathcal{I} \to \mathbb{R}$$

und ein konstantes β mit

$$c(\text{ALG}) \leq \alpha \operatorname{opt} + \beta$$

für Minimierungsprobleme bzw.

$$c(\text{ALG}) \geq \frac{1}{\alpha} \operatorname{opt} - \beta$$

*für Maximierungsprobleme existiert. Dabei ist α der **Approximationsfaktor**. Falls α durch eine Konstante $C \geq 1$ beschränkt ist, ist das Optimierungsproblem **konstant approximierbar**. Wir bezeichnen die Klasse der konstant approximierbaren Optimierungsprobleme mit $\mathcal{APX}$.*

Ein 1-Approximationsalgorithmus ist ein optimaler Algorithmus.

Definition 1.42 (Polynomielles Optimierungsproblem)**.** *Ein nichtdeterministisches Optimierungsproblem P ist in Polynomialzeit, wenn ein Polynomialzeit-Algorithmus existiert, der für jede Instanz $I \in \mathcal{I}$ die optimale Lösung* $\text{OPT}(I)$ *und den optimalen Wert* $\operatorname{opt}(I)$ *berechnet. Die Klasse der Optimierungsprobleme in Polynomialzeit wird mit $\mathcal{PO}$ bezeichnet.*

Beispiel 1.43 (lineare Programmierung LP)**.** *Es sei p ein Polynom. Eine Instanz I des linearen Programms* LP *ist*

$$\begin{aligned} c \cdot x &\to \min \\ Ax &\geq b \\ x &\geq 0 \end{aligned}$$

mit einer Matrix A und Vektoren b und c. Weiterhin betrachten wir nur Lösungen x des obigen linearen Programms mit

$$|x| \leq p(|I|). \tag{1.2}$$

Dieses Problem ist in $\mathcal{PO}$.

Beweis: Der Beweis ist zu umfangreich, um ihn hier zu führen. Er beruht auf der Ellipsoid-Methode und ist zum Beispiel bei Schrijver [Sch86] nachlesbar. □

Kapitel 2

Primal-Dual-Algorithmen

In diesem Kapitel behandeln wir Primal-Dual-Algorithmen, die wir später benötigen werden. Als Literatur wurde [BYR06], [GGP$^+$94], [GW95], [GW97], [Jai01], [WGMV95] und [Wil02] verwendet.

2.1 Der Primal-Dual-Algorithmus für Hitting Set

In diesem ersten Abschnitt betrachten wir Primal-Dual-Algorithmen und Vorläufer für das Hitting Set-Problem. Wir betrachten zunächst das folgende allgemein formulierte **ganzzahlige Problem** IP

$$c \cdot x \to \min \tag{2.1}$$
$$Ax \geq b \tag{2.2}$$
$$x \in \mathbb{N}_0^n. \tag{2.3}$$

A, b und c sind Teil der Eingabe und x ist die gesuchte Variable. Da dieses Problem $\mathcal{NPO}$-hart, können wir es nicht durch ein Polynomialzeit-Algorithmus lösen. (Wir setzen $\mathcal{NP} \neq \mathcal{P}$ voraus.) Wenn wir die Bedingung $x \in \mathbb{N}_0^n$ zu $x \geq 0$ relaxieren erhalten wir ein lineares Programm, welches wir das **primale Programm** LP nennen:

$$c \cdot x \to \min \tag{2.4}$$
$$Ax \geq b \tag{2.5}$$
$$x \geq 0. \tag{2.6}$$

Das dazu **duale Programm** DP lautet:

$$b \cdot y \to \max \tag{2.7}$$
$$A^t y \leq c \tag{2.8}$$
$$y \geq 0. \tag{2.9}$$

Aus der linearen Optimierung ist der folgende Satz bekannt.

Satz 2.1 (Starke Dualität)**.**

$$\min\{c \cdot x | Ax \geq b, x \geq 0\} = \max\{b \cdot y \mid A^t y \leq c, y \geq 0\}$$

Korollar 2.2. *Es ist*

$$\mathrm{OPT(IP)} \geq \mathrm{OPT(LP)} = \mathrm{OPT(DP)}.$$

Definition 2.3 (zulässige Lösung, unzulässige Lösung)**.** *Ein Vektor $x \in \mathbb{N}_0^n$ heißt **(zulässige) Lösung** des ganzzahligen Programms* IP*, falls $Ax \geq b$ ist. Wir schreiben $x \in$ IP. Im anderen Fall heißt x **unzulässig**.*
*Ein Vektor $x \geq 0$ (bzw. $y \geq 0$) heißt **zulässige Lösung** des linearen (bzw. dualen) Programms* LP *(bzw.* DP*), falls $Ax \geq b$ (bzw. $A^t y \leq c$) ist. Wir schreiben $x \in$ LP (bzw. $y \in$ DP). Im anderen Fall heißt x (bzw. y) **unzulässig**.*

Für eine zulässige primale Lösung x und duale Lösung y schreiben wir $(x, y) \in \mathrm{LP} \times \mathrm{DP}$ (bzw. $(x, y) \in \mathrm{IP} \times \mathrm{DP}$, falls x ganzzahlig ist). Weiterhin ist (x, y) optimal, wenn beide Lösungen x und y optimal sind.

Definition 2.4 (Schlupfbedingungen)**.** *Es sei $(x, y) \in \mathrm{LP} \times \mathrm{DP}$. Das Paar (x, y) erfüllt die **primale Schlupfbedingung**, wenn*

$$x_i > 0 \Rightarrow (A^t y)_i = c_i$$

*für alle i ist. Das Paar (x, y) erfüllt die **duale Schlupfbedingung**, wenn*

$$y_j > 0 \Rightarrow (Ax)_j = b_j$$

für alle j ist.

Satz 2.5. *Es sei $(x, y) \in \mathrm{LP} \times \mathrm{DP}$. Das Paar (x, y) ist genau dann optimal, wenn beide Schlupfbedingungen erfüllt sind.*

Beweis: Es seien x und y optimal. Da die beiden Vektoren x und y nicht negativ sind, gilt

$$c \cdot x = \sum_i c_i x_i \geq \sum_i \left(A^t y\right)_i x_i = \sum_i \sum_j a_{ji} y_j x_i = \sum_j \left(\sum_i a_{ji} x_i \right) y_j = \sum_j \left(Ax\right)_j y_j \geq \sum_j b_j y_j = b \cdot y.$$

Aus der Optimalität folgt $c \cdot x = b \cdot y$, und die Gleichheit in obiger Zeile. Es folgt

$$\sum_i \left(c_i - (A^t y)_i\right) x_i = 0 = \sum_j \left((Ax)_j - b_j\right) y_j.$$

Aus der Nichtnegativität der Terme $c_i - (A^t y)_i$ und $(Ax)_j - b_j$ folgen die beiden Schlupfbedingungen. Umgekehrt folgt aus den Schlupfbedingungen und der Nichtnegativität von x und y

$$c \cdot x = \sum_i c_i x_i = \sum_i \left(A^t y\right)_i x_i = \sum_i \sum_j a_{ji} y_j x_i = \sum_j \left(\sum_i a_{ji} x_i \right) y_j = \sum_j \left(Ax\right)_j y_j = \sum_j b_j y_j = b \cdot y.$$

Aus dem starken Dualitätssatz folgt die Behauptung. □

Definition 2.6 (relaxierte Schlupfbedingungen)**.** *Es sei $(x, y) \in \mathrm{LP} \times \mathrm{DP}$ und $\alpha, \beta \geq 1$. Das Paar (x, y) erfüllt die **primale relaxierte Schlupfbedingung**, wenn*

$$x_i > 0 \Rightarrow \frac{c_i}{\alpha} \leq (A^t y)_i \leq c_i$$

*für alle i ist. Das Paar (x, y) erfüllt die **duale relaxierte Schlupfbedingung**, wenn*

$$y_j > 0 \Rightarrow \beta b_j \geq (Ax)_j \geq b_j$$

für alle j ist.

Im Fall $\alpha = 1$ oder $\beta = 1$ wird die relaxierte Schlupfbedingung zur normalen Schlupfbedingung.

Satz 2.7. *Es erfülle $(x, y) \in \mathrm{IP} \times \mathrm{DP}$ die beiden relaxierten Schlupfbedingungen mit den Parametern $\alpha, \beta \geq 1$. Dann gilt*

$$c \cdot x \leq \alpha\beta b \cdot y.$$

Beweis: Aus der Nichtnegativität der Lösungen x und y folgt

$$c \cdot x = \sum_i c_i x_i \leq \alpha \sum_i (A^t y)_i x_i = \alpha \sum_i \sum_j a_{ji} y_j x_i = \alpha \sum_j \left(\sum_i a_{ji} x_i \right) y_j \leq \alpha\beta \sum_j b_j y_j = \alpha\beta b \cdot y.$$

□

Korollar 2.8. *Es sei* ALG *ein Algorithmus, der eine Lösung x des ganzzahligen Programms* IP *berechnet. Weiterhin existiere eine duale Lösung y des zugehörigen relaxierten dualen Programms, so dass das Paar (x, y) die beiden relaxierten Schlupfbedingungen mit den Parametern $\alpha, \beta \geq 1$ erfülle, dann berechnet* ALG *eine $\alpha\beta$-Approximation.*

Beweis: Es ist nach dem Satz

$$c(\mathrm{ALG}) = c \cdot x \leq \alpha\beta c \cdot y \leq \alpha\beta \,\mathrm{OPT(DP)} \leq \alpha\beta \,\mathrm{OPT(IP)}.$$

□

Wir werden die Primal-Dual Algorithmen nur für Überdeckungsprobleme benutzen.

Definition 2.9 (Überdeckungsprobleme)**.** *Ein Minimierungsproblem* IP *der Gestalt*

$$\begin{aligned} c \cdot x &\to \min \\ Ax &\geq b \\ x &\in \{0,1\}^n \end{aligned}$$

ist ein ***Überdeckungsproblem*** CP*.*

Bei einem Überdeckungsproblem geben wir eine Lösung x oft durch die Menge

$$X = \{i \mid x_i = 1\}$$

an. Es ist $x = \mathbb{1}_X$. Der Vorläufer der Primal-Dual-Algorithmen ist der Algorithmus von Hochbaum [Hoc82] für das Set Cover-Problem.

Problem 2.10 (Set Cover)**.**
Instanz*: Eine Menge U, $\mathcal{S} \subseteq \mathcal{P}(U)$ und eine Funktion $c : \mathcal{S} \to \mathbb{R}_0^+$.*
Lösung*: Eine Familie von Mengen $\mathcal{S}' \subseteq \mathcal{S}$, so dass*

$$\bigcup \{S \mid S \in \mathcal{S}'\} = U$$

ist.
Maß*:* $\sum_{S \in \mathcal{S}'} c_S$
Ziel*:* min*.*
Wir nennen U das ***Universum*** *und c die* ***Kostenfunktion****. Eine Menge $S \in \mathcal{S}$* ***überdeckt*** *alle Elemente $e \in S$. Gesucht ist also eine gewichtete kleinste Teilmenge von $\mathcal{S}$, die das Universum U überdeckt.*

Da wir später Unter-Probleme des Hitting-Set-Problems betrachten, formulieren wir das Set-Cover-Problem als ein Hitting-Set-Problem um. Durch Vertauschen der Mengen und der Elemente sieht man leicht, dass beide Probleme äquivalent sind.

Problem 2.11 (Hitting Set)**.**
Instanz*: Eine Menge U, $\mathcal{S} \subseteq \mathcal{P}(U)$ und eine Funktion $c : U \to \mathbb{R}_0^+$.*
Lösung*: Eine Menge $U' \subseteq U$ so dass*

$$U' \cap S \neq \emptyset$$

für alles $S \in \mathcal{S}$ ist.
Maß*:* $\sum_{e \in U'} c_e \to \min$
Ziel*:* min*.*
Wir nennen U wieder das ***Universum*** *und c die* ***Kostenfunktion****. Eine Menge $U' \subset U$* ***trifft*** *die Menge $S \in \mathcal{S}$, falls*

$$U' \cap S \neq \emptyset$$

ist. Ein Element $e \in U$ ***trifft*** *die Menge S, falls die Menge $\{e\}$ die Menge S trifft. Gesucht ist also ein gewichtetes kleinstes Teiluniversum von U, welches alle Mengen $S \in \mathcal{S}$ trifft.*

Wir können das Hitting-Set-Problem durch folgendes CP beschreiben

$$\begin{aligned} c \cdot x &\to \min \\ \sum_{e \in S} x_e &\geq 1, \forall S \\ x_e &\in \{0,1\}, \forall e \in U. \end{aligned}$$

Wir relaxieren $x_e \in \{0,1\}$ zu $x_e \geq 0$. Auf die Bedingung $x_e \leq 1$ können wir verzichten. Wir erhalten das folgende duale Programm

$$\sum_{S \in \mathcal{S}} y_S \to \max$$
$$\sum_{S: e \in S} y_S \leq c_e, \forall e \in U$$
$$y_S \geq 0, \forall S.$$

Das duale (und auch das primale) Programm ist nach Beispiel 1.43 in Polynomialzeit lösbar. Der Algorithmus von Hochbaum wählt eine optimale duale Lösung y^{OPT} und die primale Lösung

$$U' := \{e \mid \sum_{S: e \in S} y_S^{\mathrm{OPT}} = c_e\}.$$

Satz 2.12. *Der Algorithmus von Hochbaum ist ein $s_{\max}$-Approximationsalgorithmus für das Hitting Set-Problem mit*

$$s_{\max} := \max_{S \in \mathcal{S}} |S|.$$

Beweis: Wir zeigen zuerst, dass die primale Lösung $x = \mathbb{1}_{U'}$ zulässig ist. Falls sie nicht zulässig ist, dann existiert ein S mit $\sum_{e \in S} x_e = 0$. Für alle $e \in S$ folgt nach Definition der primalen Lösung $\sum_{S': e \in S'} y_{S'}^{\mathrm{OPT}} < c_e$. Da y_S^{OPT} nur in diesen Bedingungen vorkommt, kann y_S^{OPT} erhöht werden. Dies ist ein Widerspruch zur Optimalität von y^{OPT}.
Nach der Definition von U' ist die primale Schlupfbedingung erfüllt. Für alle S gilt offensichtlich

$$1 \leq \sum_{e \in S} x_e \leq s_{\max}$$

und somit gilt die relaxierte duale Schlupfbedingung mit dem Faktor $\beta = s_{\max}$. Mit Korollar 2.8 folgt die Approximationsgüte. Wir haben schon gesehen, dass der Algorithmus in Polynomialzeit ist. □

Falls jede Menge maximal zwei Elemente hat, erhalten wir ein 2-Approximationsalgorithmus. Dies im Vertex Cover-Problem der Fall.

Problem 2.13 (Vertex Cover)**.** *Es sei $G = (V, E)$ ein Graph und $c : V \to \mathbb{R}_0^+$ eine Kostenfunktion auf den Knoten. Gesucht ist eine Knotenmenge $V' \subseteq V$, so dass jede Kante abgedeckt ist, dass heißt für alle Kanten e ist $e \cap V' \neq \emptyset$. Die Kosten*

$$c(V') := \sum_{v \in V'} c_v$$

sollen minimiert werden.

Korollar 2.14. *Der Algorithmus von Hochbaum ist ein 2-Approximationsalgorithmus für das Vertex Cover-Problem.*

Der Beweis von Satz 2.12 zeigt weiterhin, dass jede duale Lösung y, die nicht in einer dualen Variable erhöht werden kann, eine $s_{\max}$-Approximation für das Set-Cover-Problem liefert. Wir machen deshalb die folgende Definition.

Definition 2.15 (minimale, maximale Lösung)**.** *Eine primale Lösung x eines* IP *(bzw.* LP*) heißt **minimal**, falls für alle i und alle $\epsilon > 0$ der Vektor $x - \epsilon e_i$ keine zulässige Lösung ist.*
Eine duale Lösung y eines DP *heißt **maximal**, falls für alle i und all $\epsilon > 0$ der Vektor $y + \epsilon e_i$ keine zulässige Lösung ist.*

Bar-Yehuda und Even [BE81] haben aus dieser Idee den ersten Primal-Dual-Algorithmus in [BE81] entwickelt. Dazu benötigen wir noch die folgenden Definitionen.

Definition 2.16 (verletzt)**.** *Eine unzulässige Lösung $x \geq 0$ von* LP *besitzt mindestens ein Index j mit*

$$(Ax)_j < b_j.$$

*Wir nennen die Bedingung j **verletzt** (bzgl. x).*

```
01 X ← ∅
02 y ← 0
03 l ← 0
04 SOLANGE X unzulässig ist:
05    l ← l + 1
06    Wähle V ⊆ {S | die Menge S ist verletzt bzgl. X}
07    Erhöhe y_j für alle j ∈ V bis ein dichtes i_l existiert.
08    X ← X ∪ {i_l}
09 FÜR k = l BIS 1
10    FALLS X\{i_k} zulässige Lösung ist DANN X ← X\{i_k}
11 AUSGABE X
```

Abbildung 2.1: Der allgemeine Primal-Dual-Algorithmus PD mit einem inversen Löschschritt.

```
01 U' ← ∅
02 y ← 0
03 l ← 0
04 SOLANGE U' unzulässig ist:
05    l ← l + 1
06    Wähle eine Menge S mit Σ_{e ∈ S} x_e = 0
07    Erhöhe y_S bis ein e_l ∈ U mit Σ_{S':e_l ∈ S'} y_{S'} = c_{e_l} existiert.
08    U' ← U' ∪ {e_l}
09 FÜR k = l BIS 1
10    FALLS U'\{e_k} zulässige Lösung ist DANN U' ← U'\{e_k}
11 AUSGABE U'
```

Abbildung 2.2: Der Primal-Dual-Algorithmus PDHS für das Hitting Set-Problem.

Definition 2.17 (dicht)**.** *Es sei* $(x, y) \in \mathrm{LP} \times \mathrm{DP}$*. Wir nennen die Bedingung* i ***dicht****, wenn die Bedingung*

$$(A^t y)_i \leq c_i$$

mit Gleichheit erfüllt ist. Analog nennen wir die Bedingung j ***dicht****, wenn die Bedingung*

$$(Ax)_j \geq b_j$$

mit Gleichheit erülllt ist.

Der Primal-Dual-Algorithmus PD mit einem inversen Löschschritt für allgemeine CP-Probleme ist in Abbildung 2.1 dargestellt. Bei einem Primal-Dual-Algorithmus ohne Löschschritt sind die Zeilen 08-10 wegzulassen. Mit welcher Geschwindigkeit die dualen Variablen y_j in Zeile 07 erhöht werden, ist vom jeweiligen Algorithmus abhängig. Der charakteristische Vektor $x = \mathbb{1}_X$ ist dann eine Lösung des ganzzahligen Programms. Das Lösungspaar (x, y) erfüllt die erste Schlupfbedingung. Der Algorithmus ist im Allgemeinen nicht in Polynomialzeit. Für einen Polynomialzeit-Algorithmus muss die Prüfung in Zeile 04 und 10, ob eine Lösung zulässig bzw. unzulässig in Polynomialzeit berechenbar sein. Weiterhin muss in Zeile 06 das Finden einer Familie von verletzten Mengen in Polynomialzeit sein.
Speziell für das Hitting Set Problem ist der Primal-Dual-Algorithmus in Abbildung 2.2 dargestellt. Im folgenden Satz betrachten wir den Algorithmus von Bar-Yehuda und Even ([BE81]).

Satz 2.18 (Bar-Yehuda, Even)**.** *Der Algorithmus* PDHS *ist ein* $s_{\max}$*-Approximationsalgorithmus.*

Beweis: Wir haben zu zeigen, dass PDHS in Polynomialzeit ist. Die Schleife 04 wird maximal $|U|$-Mal durchlaufen. Da alle Mengen S in der Eingabe von Hitting-Set kodiert sind, kann in Zeile 06 eine verletzte Menge in Polynomialzeit gefunden werden. Falls keine verletzte Menge existiert, so ist die Lösung zulässig und die Zeilen 04 und 10 können somit auch in Polynomialzeit berechnet werden. Die Approximationsgüte wird wie in Satz 2.12 gezeigt. □

Für das Hitting Set-Problem existiert ein Greedy-Algorithmus mit der Approximationsgüte $1 + \ln(|U|)$ (siehe [Chv79], [Joh74], [Lov75] und [Sla97] für eine genauere Analyse). Feige [Fei98] hat unter der Voraussetzung $\mathcal{NP} \not\subseteq \text{DTIME}(2^{\log\log(n)})$ gezeigt, dass kein Approximationsalgorithmus eine bessere Güte als $(1 - o(1)) \ln(|U|)$ hat. Raz und Safra [RS84] haben unter der Voraussetzung $\mathcal{NP} \not\subseteq \mathcal{P}$ gezeigt, dass kein Approximationsalgorithmus eine bessere Güte als $c \ln(n)$ für eine Konstante c hat.
Der Greedy-Algorithmus ist ein Dual-Fitting-Algorithmus. Dies ist eine Variante der Primal-Dual-Algorithmen, bei denen die dualen Variablen y die Kosten $c \cdot x$ des linearen Programms überdecken und nicht zulässig für das duale Programm DP sind. Durch die Bestimmung eines Faktors α wird die Lösung $\frac{1}{\alpha} y$ zu einer zulässigen Lösung für DP gemacht und man erhält dadurch eine α-Approximation. Dies kann man in [FR04] und [JMM+03] nachlesen.

2.2 Primal-Dual-Algorithmen für Netzwerk-Probleme

Wir wollen nun die Primal-Dual-Algorithmen für einige Netzwerk-Probleme betrachten. Es sei $G = (V, E, c)$ ein Netzwerk und $f : \mathcal{P}(V) \to \mathbb{Z}$ eine in Polynomialzeit berechenbare Funktion. Wir betrachten das folgende **Netzwerk-Problem** CP(f)

$$\begin{aligned} \sum_e c_e &\to \min \\ \sum_{e \in \delta(S)} x_e &\geq f(S), \forall S \subseteq V \\ x_e &\in \{0,1\}, \forall e \in E. \end{aligned}$$

Wir nehmen ohne Einschränkung der Allgemeinheit $f(V) \leq 0$ und $f(\emptyset) \leq 0$ an, da CP(f) sonst keine Lösung besitzt. Bei diesem Problem nennen wir die Menge S bzgl. x **verletzt** , falls die Bedingung mit dem Index $\delta(S)$ verletzt ist. Das heißt, es ist

$$\sum_{e \in \delta(S)} x_e < f(s).$$

Das relaxierte primale Programm LP(f) lautet

$$\begin{aligned} c \cdot x &\to \min \\ \sum_{e \in \delta(S)} x_e &\geq f(S), \forall S \subseteq V \\ x &\in [0,1]^E \end{aligned}$$

und das duale Programm DP(f) lautet

$$\sum_{S \subseteq V} f(S) y_S - \sum_{e \in E} z_e \to \max \tag{2.10}$$

$$\sum_{S : e \in \delta(S)} y_S \leq c_e + z_e, \forall e \in E \tag{2.11}$$

$$y, z \geq 0. \tag{2.12}$$

$$\tag{2.13}$$

Wir betrachten zunächst Netzwerk-Probleme für Funktionen mit Bildbereich $\{0,1\}$. Dies sind Unterprobleme des Hitting Set-Problems.

Definition 2.19 (binäre Funktion)**.** *Eine Funktion*

$$f : \mathcal{P}(V) \to \{0,1\}$$

heißt ***binäre*** *Funktionen*

```
01 X ← ∅
02 y ← 0
03 l ← 0
04 SOLANGE X unzulässig ist:
05    l ← l + 1
06    Wähle eine verletzte Menge S.
07    Erhöhe y_S bis ein dichtes e_l existiert.
08    X ← X ∪ {e_l}
09 FÜR k = l BIS 1
10    FALLS X\{e_k} zulässige Lösung ist DANN X ← X\{e_k}
11 AUSGABE X
```

Abbildung 2.3: Der Primal-Dual-Algorithmus PD1

Da die Bedingung $x_e \leq 1$ für binäre Funktionen redundant ist, vereinfacht sich das primale Programm LP(f) zu

$$c \cdot x \to \min$$
$$\sum_{e \in \delta(S)} x_e \geq f(S), \forall S \subseteq V$$
$$x \geq 0.$$

Das duale Programm DP(f) vereinfacht sich zu

$$\sum_{S \subseteq V} f(S) y_S \to \max$$
$$\sum_{S: e \in \delta(S)} y_S \leq c_e, \forall e \in E$$
$$y \geq 0.$$

Definition 2.20 (dicht)**.** *Es sei* $(x, y) \in \mathrm{LP}(f) \times \mathrm{DP}(f)$*. Die Kante* e *heißt* ***dicht****, falls*

$$\sum_{S: e \in \delta(S)} y_S = c_e$$

ist und die Menge S *heißt* ***dicht****, falls*

$$\sum_{e \in \delta(S)} x_e = f(S)$$

ist.

Zuerst betrachten wir die Primal-Dual-Algorithmen, bei denen in jeder Iteration genau eine duale Variable erhöht wird. Mit einem inversen Löschschritt ist der Prima-Dual-Algorithmus $PD1$ in Abbildung 2.3 dargestellt

Lemma 2.21. *Für binäre Funktionen hat der Algorithmus* PD1 *nach dem Schleifendurchlauf 04-08 eine zulässige, zyklenfreie Lösung berechnet.*

Beweis: Wir nehmen an, dass die Kante $e = \{u, v\}$ in der (unzulässigen) zyklenfreien Lösung X einen Zyklus schließen würde. Es sei S eine Menge mit der Schnittkante e. Ohne Einschränkung ist $u \in S$ und $v \notin S$. Wenn wir den Weg von u nach v in X verfolgen, gelangen wir zu einer Kante e', die inzident zu einem Knoten in S und inzident zu einem Knoten in S^C ist. Es ist $e' \in \delta(S)$ und die Bedingung S

$$\sum_{e \in \delta(S)} x_e \geq x_{e'} = 1$$

ist nicht verletzt. Der Algorithmus würde also die Variable y_S nicht erhöhen und die Kante e nicht hinzufügen.
Da nach dem Schleifendurchlauf keine Bedingung verletzt ist, ist X zulässig. □

2.2.1 Der kürzeste Weg

Wir betrachten nun das Problem der Bestimmung des kürzesten Weges.

Problem 2.22 (kürzesterWeg)**.**
Instanz: *Ein Netzwerk $G = (V, E, c)$ und zwei Knoten s, t.*
Lösung: *Ein Weg von s nach t.*
Maß: *Summe der Kantengewichte des Weges.*
Ziel: min.

Dieses Problem kann durch die binäre Funktion

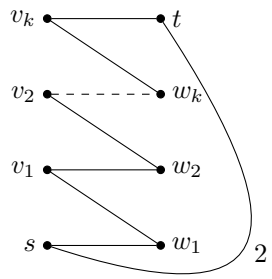
$$f(S) = \begin{cases} 1 & \text{falls } s \in S, t \notin S \\ 0 & \text{sonst} \end{cases} \tag{2.14}$$

oder die Funktion

$$f(S) = \begin{cases} 1 & \text{falls } |S \cap \{s,t\}| = 1 \\ 0 & \text{sonst} \end{cases} \tag{2.15}$$

als Netzwerk-Problem $\mathrm{CP}(f)$ formuliert werden.
Das folgende Beispiel zeigt, dass die richtige Wahl der verletzten Menge entscheidend ist.

Beispiel 2.23. *Wir betrachten das folgende Netzwerk:*



Alle Kanten bis auf die Kante $\{s,t\}$ haben das Gewicht 1. Die Menge $S = \{s, v_1, v_2, \ldots, v_k\}$ trennt s und t und ist somit verletzt. Wir können y_S auf 1 erhöhen, bis alle Kanten mit Gewicht 1 dicht sind. Nun können wir kein andere duale Variable $y_{S'}$ für ein verletztes S' erhöhen. Die Kante $\{s,t\}$ kann nicht mehr dicht werden. Der Algorithmus konstruiert also die Lösung $(s, w_1, v_1, w_2, v_2, \ldots, w_k, v_k, t)$ mit den Kosten $2k+1$. Die optimale Lösung hat den Wert 2 und der Algorithmus hat eine duale Lösung mit dem Wert 1 konstruiert.

Die Lösung kann also bei beliebiger Wahl der verletzen Mengen beliebig schlecht werden. Wir werden daher die folgenden Mengen betrachten.

Definition 2.24 (minimal verletzt)**.** *Es sei X eine nicht zulässige Lösung und $x = \mathbb{1}_X$ der zugehörige charakteristische Vektor des Netzwerk-Problems* $\mathrm{CP}(f)$*. Die Menge $S \subseteq V$ heißt **minimal verletzt**, falls*

- *die Bedingung*

$$\sum_{e \in \delta(S)} x_e \geq f(S)$$

  verletzt ist und

- *für alle echten Teilmengen $S' \subset S$, $S' \in \mathcal{S}$ die Bedingung*

$$\sum_{e \in \delta(S')} x_e \geq f(S')$$

nicht verletzt ist.

S ist also verletzt und bezüglich der Inklusion minimal mit dieser Eigenschaft.

Definition 2.25 (Erweiterung)**.** *Es sei X eine nicht zulässige Lösung des Netzwerk-Problems mit einer binären Funktion f. Eine zulässige Lösung Y heißt **Erweiterung** von X, falls*

$$X \subseteq Y$$

*ist. Eine Erweiterung Y ist eine **minimale Erweiterung** von Y, falls für alle $e \in Y \backslash X$ die Menge $Y \backslash \{e\}$ keine zulässige Lösung ist.*

Wir können jetzt den folgenden Satz formulieren.

Satz 2.26. *Es sei* PD1 *in Polynomialzeit für das Netzwerk-Problem* CP(f) *und*

$$\alpha := \max_{\substack{X \text{ ist eine unzulässige} \\ \text{Lösung}}} \max_{\substack{Y \text{ ist eine minimale} \\ \text{Erweiterung von } X}} |Y \cap \delta(S(X))|.$$

Dabei bezeichnet $S(X)$ die von PD1 *gewählte verletze Menge bei der gegebenen Teillösung X. Dann ist* PD1 *ein α-Approximationsalgorithmus.*

Beweis: Aus der Dichtheit der Kanten in der Lösung folgt

$$c(\text{PDg}) = \sum_{e \in \text{PD1}} c_e = \sum_{e \in \text{PD1}} \sum_{S : e \in \delta(S)} y_S = \sum_S \sum_{\substack{e \in \delta(S) \\ e \in \text{PD1}}} y_S = \sum_S |\,\text{PD1} \cap \delta(S)| y_S.$$

Es sei e_i und $S_i = S(\{e_1, \ldots, e_{i-1}\})$ die von PD1 im i-ten Schritt gewählte Kante und verletzte Menge. Es folgt

$$c(\text{PDg}) = \sum_{i=1}^{l} |\,\text{PD1} \cap \delta(S_i)| y_{S_i}$$

Durch den den Löschschritt in Iteration k ist PD1 $\cup \{e_1, \ldots, e_{k-1}\}$ eine minimale Erweiterung von $\{e_1, \ldots, e_{k-1}\}$. Es folgt

$$|\,\text{PD1} \cap \delta(S_i)| \leq \max_{\substack{Y \text{ ist eine minimale} \\ \text{Erweiterung von} \{e_1, \ldots e_{i-1}\}}} |Y \cap \delta(S_i)| \leq \alpha.$$

Insgesamt folgt

$$c(\text{PD1}) \leq \alpha \sum_S y_S.$$

Aus $\sum_{S \subseteq V} y_S \leq$ opt folgt die Behauptung. □

Definition 2.27 (disjunkt dominiert)**.** *Eine Funktion $f : \mathcal{P} \to \mathbb{Z}$ heißt **disjunkt dominiert**, falls*

$$f(S \cup T) \leq \max\{f(S), f(T)\}$$

für alle disjunkten Mengen S und T ist.

Beispiel 2.28. *Die Funktion (2.14) für das kürzeste Weg-Problem der Knoten s und t*

$$f(S) = \begin{cases} 1 & \text{falls } s \in S, t \notin S \\ 0 & \text{sonst} \end{cases} \tag{2.16}$$

ist disjunkt dominiert.

Beweis: Für $\max\{f(S), f(T)\} = 1$ ist nichts zu zeigen. Es sei $f(S) = f(T) = 0$ für die beiden Mengen $S, T \subseteq V$. Somit ist $s \in S, T$ und $t \notin S, T$. Es folgt $s \in S \cup T$ und $t \notin S \cup T$ und $f(S \cup T) = 0$. □

Lemma 2.29. *Eine binäre Funktion ist genau dann disjunkt dominiert, wenn*

$$f(S) = f(T) = 0 \Rightarrow f(S \cup T) = 0$$

für alle disjunkten Mengen S, T gilt.

Lemma 2.30. *Es sei ein Netzwerk-Problem* $\mathrm{CP}(f)$ *mit einer binären disjunkt dominierten Funktion f gegeben.*

1. *Eine Lösung X ist genau dann zulässig, wenn für jede Zusammenhangskomponente C von X, die Bedingung $f(C) = 0$ erfüllt ist.*
2. *Die minimalen verletzen Mengen einer unzulässigen Lösung X sind die Zusammenhangskomponenten C von X mit $f(C) = 1$.*

Beweis: Es sei S eine verletzte Menge. Dann ist $|\delta(S) \cap X| = 0$ und $f(S) = 1$. Wir können S als disjunkte Vereinigung von Zusammenhangskomponenten C_i von X schreiben. Da f disjunkt dominiert ist, folgt $f(C_i) = 1$ für ein i und die Menge C_i ist verletzt.
Da jede echte Teilmenge einer Zusammenhangskomponente von X eine Schnittkante in X enthält, kann keine Untermenge einer Zusammenhangskomponente von X verletzt sein. □

Wir können die Zusammenhangskomponenten in jeder Iteration in Polynomialzeit berechnen oder die Familie der Zusammenhangskomponente in jeder Iteration in Polynomialzeit aktualisieren.

Definition 2.31 (proper)**.** *Eine Funktion $f : \mathcal{P} \to \mathbb{Z}$ heißt **proper**, falls f disjunkt dominiert und **symmetrisch**, das heißt es ist*

$$f(S) = f(S^C)$$

für alle $S \subseteq V$.

Beispiel 2.32. *Die Funktion (2.15) des kürzesten Weges-Problem*

$$f(S) = \begin{cases} 1 & \textit{falls } |S \cap \{s,t\}| = 1 \\ 0 & \textit{sonst} \end{cases} \tag{2.17}$$

ist proper.

Beweis: Wir zeigen zuerst, dass f disjunkt dominiert ist. Es seien $S, T \subseteq V$ zwei disjunkte Mengen mit $f(S) = f(T) = 0$. Wir folgern

$$|S \cap \{s,t\}| \equiv |T \cap \{s,t\}| \equiv 0 \pmod 2.$$

Aus der Disjunktheit von S und T folgt

$$|(S \cup T) \cap \{s,t\}| \equiv |S \cap \{s,t\}| + |T \cap \{s,t\}| \equiv 0 \pmod 2$$

und somit ist $f(S \cup T) = 0$.
Wir zeigen nun die Symmetrie. Aus $|(S \cup S^C) \cap \{s,t\}| = 2$ folgt $f(S) + f(S^C) \equiv 0 \pmod 2$ und $f(S) = f(S^C)$. □

Lemma 2.33. *Eine binäre symmetrische Funktion ist genau dann proper, wenn für alle Mengen S, T mit $S \subseteq T$*

$$f(S) = f(T) = 0 \Rightarrow f(T \backslash S) = 0$$

gilt.

Beweis: Es sei f proper und $S \subseteq T$ zwei Mengen mit $f(S) = f(T) = 0$. Es folgt

$$f(T \backslash S) = f((T \backslash S)^C) \leq \max\{f(T^C), f(S)\} = \max\{f(T), f(S)\} = 0.$$

Es seien nun umgekehrt zwei disjunkte Mengen S und T gegeben. Für $\max\{f(S), f(T)\} = 1$ ist nichts zu zeigen, also sei $f(S) = f(T) = 0$. Es folgt $f(S^C) = f(T^C) = 0$. Aus $f(T^C) = 0$, $f(S) = 0$ und $S \subseteq T^C$ folgt

$$0 = f(T^C \backslash S) = f((S \cup T)^C) = f(S \cup T).$$

□

Im folgenden Satz werden wir den Algorithmus von Dijkstra [Dij59] angegeben.

Satz 2.34 (Dijkstra)**.** *Es sei ein Netzwerk $G = (V, E, c)$ und zwei Knoten s, t gegeben. Der Algorithmus* PD1 *bzgl. der Funktion (2.14)*

$$f(S) = \begin{cases} 1 & \text{falls } s \in S, t \notin S \\ 0 & \text{sonst} \end{cases}$$

ist ein Primal-Dual-Algorithmus mit der Approximationsgüte 1 *für das kürzeste Weg-Problem.*

Beweis: Da die minimale verletzte Menge eine Zusammenhangskomponente ist, hat jede minimale Erweiterung genau eine Schnittkante. Der Algorithmus PD1 führt maximal $|V|$ Iterationen und in jeder Iteration können wir in Polynomialzeit die Zusammenhangskomponente berechnen oder aktualisieren und auf Zulässigkeit prüfen. □

Bei diesem Problem ergibt jede beliebige Lösch-Reihenfolge eine optimale Lösung. Wir betrachten nun ein Problem, bei dem dies nicht so ist.

2.2.2 Der minimale gerichtete Spannbaum

Problem 2.35 (gerichteter Spannbaum)**.**
***Instanz**: Ein gerichtetes Netzwerk $G = (V, E, c)$ und ein Knoten s.*
***Lösung**: Ein Teilnetzwerk, so dass für alle Knoten t ein Weg von s nach t existiert.*
***Maß**: Summe der Kantengewichte des Teilnetzwerks.*
***Ziel**:* min.

Wir können das Problem durch das folgende gerichtete Netzwerk-Problem CP(f)

$$c \cdot x \to \min \tag{2.18}$$

$$\sum_{e \in \delta^-(S)} x_e \geq f(S), \forall S \tag{2.19}$$

$$x \in \{0, 1\}^E \tag{2.20}$$

mit der Funktion

$$f(S) := \begin{cases} 1 & \text{falls } \emptyset \subset S \subset V, s \notin S \\ 0 & \text{sonst} \end{cases}$$

beschreiben.Wir haben das folgende Lemma.

Lemma 2.36. *Es sei das gerichtete Netzwerk-Problem mit obiger Funktion f gegeben.*

1. *Eine Lösung X ist genau dann zulässig, wenn jede starke Zusammmenhangskomponente C ohne den Knoten s eine Eingangsschnittkante hat.*
2. *Die minimalen verletzten Mengen einer unzulässigen Lösung X sind die starken Zusammenhangskomponenten ohne den Knoten s, die keine Eingangsschnittkante haben.*

Beweis: Es sei S eine verletzte Menge. Dann ist $|\delta^-(S) \cap X| = 0$ und $\emptyset \subset S \subset V$, $s \notin S$. Die Menge S besteht aus starken Zusammenhangskomponenten C_i. Wir betrachten den Graphen H, der für jede starke Zusammenhangskomponente C_i einen Knoten i besitzt. Die Kante (i, j) exisitert in H, falls es eine Kante von C_i nach C_j existiert. Falls jede Zusammenhangskomponente eine Eingangskante hat, existiert ein Zyklus in H. Dieser Zyklus bildet eine starke Zusammenhangskomponente in H und somit auch in G. Dies ist ein Widerspruch. Also hat eine dieser Zusammenhangskomponenten C_j keine Eingangskante. Aus $s \notin C_j$ folgt $f(C_j) = 1$ und die Menge C_j ist verletzt. Jede Teilmenge einer starken Zusammenhangskomponente C_i enthält eine Eingangskante und ist daher nicht verletzt.
Wir nehmen nun an, dass jede starke Zusammenhangskomponente ohne s eine Eingangskante hat. Wir bilden wieder den zyklenfreien Graph H mit den starken Zusammenhangskomponente C_i als Knoten. Es sei t ein Knoten. t entspricht einem Knoten C_1 in H. Da C_1 eine Eingangskante hat, können wir rückwärts von C_1 zu einem anderen Knoten C_2 gelangen. Dieser Knoten führt wieder zu einem Knoten C_3, der wegen der Zyklenfreiheit nicht einer der vorherigen Knoten ist. Somit gelangen wir schließlich in die starke Zusammenhangskomponente, welche den Knoten s enthält. Somit existiert ein gerichteter Weg von s nach t in (V, X). □

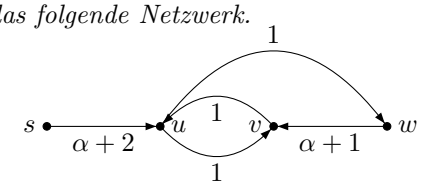
Wir können die starken Zusammenhangskomponenten in jeder Iteration in Polynomialzeit berechnen oder diese in jeder Iteration in Polynomialzeit aktualisieren. Im nächsten Satz betrachten wir den Algorithmus von Edmonds ([Edm67]).

Satz 2.37 (Edmonds)**.** *Es sei das gerichtete Netzwerk-Problem mit obiger Funktion f gegeben. Der Primal-Dual-Algorithmus $PD1$ ist ein Approximationsalgorithmus der Güte 1.*

Beweis: Die minimalen verletzten Mengen sind die starken Zusammenhangskomponenten ohne s, die keine Eingangskante haben. Somit hat jede minimale Erweiterung genau eine Eingangskante und die Behauptung für die Approximationsgüte folgt mit Satz 2.26. Der Algorithmus führt maximal $|E|$ Iterationen und in jeder Iteration können wir in Polynomialzeit die starken Zusammenhangskomponenten berechnen oder aktualisieren. □

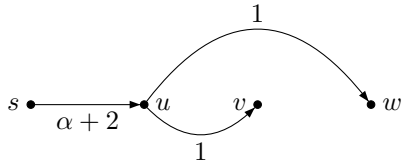
Wir zeigen nun, dass die Reihenfolge des Löschschrittes für dieses Problem wesentlich ist.

Beispiel 2.38. *Wir betrachten das folgende Netzwerk.*



Die Mengen $\{u\}$, $\{v\}$ und $\{w\}$ sind minimal verletzt. Wir können im ersten Schritt $y_{\{u\}} = 1$ und die Kante (v, u) wählen. Im zweiten Schritt wählen wir die Kante (u, v) mit $y_{\{v\}} = 1$. Nun ist die Menge $\{u, v\}$ minimal verletzt. Wir setzen $y_{\{u,v\}} = \alpha$ und die Kante (w, v) wird dicht. Im vorletzten Schritt erhöhen wir $y_{\{w\}} = 1$ und fügen die Kante (u, w) zur Lösung hinzu. Zuletzt setzen wir $y_{\{u,v,w\}} = 1$ und fügen die letzte Kante (s, u) hinzu.

- *Wir betrachten nun zuerst den inversen Löschschritt. Die Kanten (s, u) und (u, w) werden in jeder Lösung gebraucht und können nicht gelöscht werden. Die erste mögliche löschbare Kante ist (w, v). Diese kann gelöscht werden. Die Kante (u, v) kann nicht gelöscht werden Weiter kann die Kante (v, u) gelöscht werden. Wir erhalten die folgende Lösung*

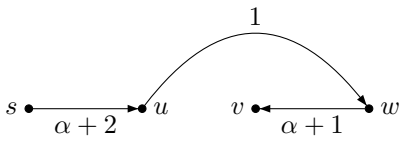


  mit Wert $\alpha + 4$. Die Wert der dualen Lösung ist $\sum_S y_S = y + 4$ und es folgt die Optimalität dieser Lösung.

- *Nun betrachten wir den folgenden Löschschritt. Wir betrachten zuerst die Kanten (u, v) und (v, u) und löschen diese. Wir erhalten die Lösung*

```
01 X ← ∅
02 y ← 0
03 l ← 0
04 SOLANGE X unzulässig ist:
05    l ← l + 1
06    V ⊆ {S | S ist verletzt bzgl. X}
07    Erhöhe gleichmäßig y_S für alle S ∈ V bis ein dichtes e_l existiert.
08    X ← X ∪ {e_l}
09 FÜR k ← l BIS 1
10    FALLS X\{e_k} zulässige Lösung ist DANN X ← X\{e_k}
11 AUSGABE X
```

Abbildung 2.4: Der Primal-Dual-Algorithmus PDg mit gleichmäßiger Variablenerhöhung



mit Wert $2\alpha + 4$.

Der Algorithmus von Prim [Pri57] zur Berechnung eines ungerichteten minimalen Spannbaumes ist kein Primal-Dual-Algorithmus und wird daher hier nicht betrachtet. Dieser Algorithmus ist ein Dual-Fitting-Algorithmus, eine Variante der Primal-Dual-Algorithmen. Den Algorithmus von Kruskal zur gleichen Problemstellung werden wir später betrachten.

2.2.3 Der Algorithmus von Nicholson

Wir betrachten nun Primal-Dual-Algorithmen, bei denen im Allgemeinen mehr als eine duale Variable von verletzten Mengen erhöht werden. Dabei erhöhen wir die dualen Variablen gleichmäßig. In Abbildung 2.4 ist der Algorithmus PDg dargestellt. Wie in Lemma 2.21 ist die berechnete Lösung nach dem Schleifendurchlauf 04-08 zyklenfrei (und zulässig).

Satz 2.39. *Es sei* PDg *in Polynomialzeit für das Netzwerk-Problem* $CP(f)$ *Weiterhin gelte für alle unzulässigen Lösungen* X *und minimalen Erweiterungen* Y *von* X

$$\sum_{S \in V(X)} |Y \cap \delta(S)| \leq \alpha |V(X)|.$$

Dabei bezeichne $V(X)$ *die Familie von verletzten Mengen, die von* PDg *bezüglich der unzulässigen Teillösung* X *gewählt wird. Dann ist* PDg *ein* α*-Approximationsalgorithmus.*

Beweis: Es bezeichne V_i die in der i-ten Iteration von PDg gewählten verletzten Mengen, ϵ_i den Wert der Erhöhung der zugehörigen duale Variablen in Iteration i und e_i die gewählte dichte Kante. Es ist $y_S = \sum_{i:S \in V_i} \epsilon_i$ und es folgt $\sum_S y_S = \sum_i |V_i| \epsilon_i$. Weiterhin ist $V_i = V(\{e_1, \ldots, e_{i-1}\})$. Wie im Beweis von Satz 2.26 ist

$$c(\text{PDg}) = \sum_S |\,\text{PDg} \cap \delta(S)| y_S.$$

Es folgt

$$c(\text{PDg}) = \sum_S \sum_{i:S \in V_i} |\,\text{PDg} \cap \delta(S)| \epsilon_i = \sum_{i=1}^{l} \sum_{S \in V_i} |\,\text{PDg} \cap \delta(S)| \epsilon_i$$

Durch den inversen Löschschritt ist PDg$\cup\{e_1,\dots e_{i-1}\}$ eine minimale Erweiterung von $\{e_1,\dots,e_{i-1}\}$ und es folgt

$$\sum_{S\in V_i} |\,\mathrm{PDg}\cap\delta(S)| \le \alpha|V_i|.$$

Es ist

$$c(\mathrm{PDg}) \le \alpha\sum_{i=1}^{l}|V_i|\epsilon_i = \alpha\sum_S y_S \le \alpha\,\mathrm{opt}$$

und die Behauptung folgt. □

Wenn nicht anders angegeben, werden wir in Zeile 06 für V alle minimalen verletzten Mengen wählen. Wir werden nun wieder das kürzeste Weg-Problem betrachten. Im folgenden Satz ist der Algorithmus von Nicholson [Nic66] angegeben.

Satz 2.40 (Nicholson)**.** *Es sei das Netzwerk $G=(V,E,c)$ und die zwei Knoten s,t gegeben. Der Algorithmus* PDg *bzgl der Funktion (2.15)*

$$f(S)=\begin{cases}1 & \textit{falls } |S\cap\{s,t\}|=1\\ 0 & \textit{sonst}\end{cases}$$

ist ein Primal-Dual-Algorithmus mit der Approximationsgüte 1 *für das kürzeste Weg-Problem.*

Beweis: Es sei X eine unzulässige Lösung. Die beiden minimalen verletzten Mengen sind die Zusammenhangskomponente von s und die Zusammenhangskomponente von t. Jede minimale Erweiterung Y hat genau eine Schnittkante ín der Zusammenhangskomponente von s und eine in der Zusammenhangskomponente von t. (Es kann die gleiche Kante sein.) Somit ist

$$\sum_{S\in V(X)}|Y\cap\delta(S)|\le 2=|V(X)|,$$

wobei $V(X)$ die beiden von PDg gewählten Mengen bezeichnet. Die Behauptung folgt mit Satz 2.39. □

2.2.4 Der minimale Spannbaum

Wir betrachten nun das Problem der Berechnung eines minimalen (ungerichteten) Spannbaumes.

Problem 2.41 (minimaler Spannbaum)**.**
Instanz*: Ein Netzwerk $G=(V,E,c)$.*
Lösung*: Ein zusammenhängender Teilgraph.*
Maß*: Summe der Gewichte des Teilgraphens.*
Ziel*:* min

Lemma 2.42. *Das minimale Spannbaum-Problem ist ein Netzwerkproblem mit der Funkion*

$$f(S):=\begin{cases}1 & \textit{für alle } \emptyset\neq S\neq V\\ 0 & \textit{sonst}\end{cases}.$$

Beweis: Leicht. □

Im nächsten Satz werden wir den Algorithmus von Kruskal ([Kru56]) betrachten.

Satz 2.43 (Kruskal)**.** *Es sei ein Netzwerk $G=(V,E,c)$ gegeben. Der Algorithmus* PDg *ist ein Primal-Dual-Algorithmus mit der Approximationsgüte* 1 *für das minimale Spannbaum-Problem.*

Beweis: Die minimal verletzten Mengen sind die Zusammenhangskomponenten. Wir werden die Erhöhung der dualen Variablen sprachlich als Verstreichen von Zeit ansehen. Da für jede Kante $e = \{u, v\}$ mit Kosten c_e die dualen Variablen der Zusammenhangskomponenten beider inzidenten Knoten u und v wachsen, wird die Kante zum Zeitpunkt $\frac{c_e}{2}$ betrachtet. Falls die Kante e einen Zyklus in der Teillösung schließen würde, sind die beiden Knoten u und v vor dem Zeitpunkt $\frac{c_e}{2}$ in der gleichen Zusammenhangskomponente und die duale Variablen der Kante e wachsen nicht auf c_e. Die Kante e würde nicht dicht werden und somit nicht zur Lösung hinzugefügt werden. Im anderen Fall wird die Kante zum Zeitpunkt $\frac{c_e}{2}$ dicht und zur Lösung hinzugefügt. Der Algorithmus sortiert also die Kanten aufsteigend und fügt die aktuell günstigste Kante hinzu, welche keinen Zyklus schließen würde. Dies ist genau der Greedy-Algorithmus von Kruskal. □

Leider können wir in diesem Fall nicht die Optimalität mit dem Satz 2.39 zeigen, da die IP-Lücke 2 ist, wie das folgende Lemma zeigt.

Lemma 2.44. *Die* IP*-Lücke des ganzzahligen Programms*

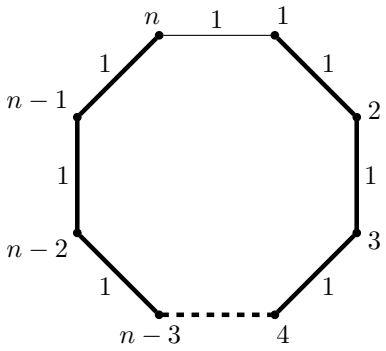
$$\begin{aligned} c \cdot x &\to \min \\ \sum_{e:e\in\delta(S)} x_e &\geq 1, \forall \emptyset \subset S \subset V \\ x &\in \{0,1\}^E \end{aligned}$$

zum relaxierten linearen Programm

$$\begin{aligned} c \cdot x &\to \min \\ \sum_{e:e\in\delta(S)} x_e &\geq 1, \forall \emptyset \subset S \subset V \\ x &\geq 0 \end{aligned}$$

beträgt 2*.*

Beweis: Wir betrachten den Graphen C_n mit Kosten 1 für jede Kante.



C_n ist ein Kreis mit n Knoten. Die optimale ganzzahlige Lösung muss alle Kanten bis auf eine wählen. Die Kosten sind $\text{opt} = n - 1$. Die optimale fraktionale Lösung A ist $x_e = \frac{1}{2}$ für alle Kanten e und hat die Kosten $\text{a} = \frac{n}{2}$. Somit beträgt die IP-Lücke mindestens

$$\frac{\text{opt}}{\text{a}} = \frac{n-1}{\frac{n}{2}} \to 2.$$

Der Algorithmus für das allgemeinere Problem des Steiner-Walds in Satz 2.51 hat einen Approximationsfaktor von 2 und zeigt damit die Umkehrung. □

Wir werden mit einem anderen CP die Optimalität zeigen. Dafür benötigen wir die folgende Definition.

Definition 2.45 (Partition, Schnittkante)**.** *Es sei $G = (V, E)$ ein Graph und $\mathcal{S} = \{S_1, \dots S_k\}$ eine* ***Partition*** *von V, das heißt es ist*

$$V = \bigcup_i^k S_i$$

für nichtleere, disjunkte Mengen S_i, $i = 1, \dots, k$. Wir schreiben kurz $V = \dot{\bigcup} S_i$. Die Menge der ***Schnittkanten*** *der Partition $\mathcal{S} = \{S_1, \dots, S_k\}$ ist*

$$\delta(\mathcal{S}) := \delta(S_1, \dots, S_k) := \{e = \{u, v\} \mid \exists 1 \leq i \neq j \leq k, u \in S_i, v \in S_j\}.$$

Für eine Menge S schreiben wir weiterhin $\delta(S)$, statt $\delta(S, S^C)$. Wir können das Problem des minimalen Spannbaumes für das Netzwerk $G = (V, E, c)$ mit dem folgenden CP(mSB) beschreiben

$$\begin{aligned} c \cdot x &\to \min \\ \sum_{e \in \delta(\mathcal{S})} x_e &\geq |\mathcal{S}| - 1, \forall \text{ Partitionen } \mathcal{S} \\ x &\in \{0, 1\}^E \end{aligned}$$

beschreiben, da jede Partition der Größe k genau k Zusammenhangskomponenten hat und somit mindestens $k - 1$ viele Schnittkanten enthalten muss. Das relaxierte Programm hat die folgende Gestalt

$$c \cdot x \to \min \tag{2.21}$$

$$\sum_{e \in \delta(\mathcal{S})} x_e \geq |\mathcal{S}| - 1, \forall \text{ Partitionen } \mathcal{S} \tag{2.22}$$

$$x \leq 1 \tag{2.23}$$

$$x \geq 0. \tag{2.24}$$

$$\tag{2.25}$$

Lemma 2.46. *Es sei x eine minimale zulässige Lösung von*

$$\begin{aligned} c \cdot x &\to \min \\ \sum_{e \in \delta(\mathcal{S})} x_e &\geq |\mathcal{S}| - 1, \forall \textit{ Partitionen } \mathcal{S} \\ x &\geq 0 \end{aligned}$$

Dann ist

$$x \leq 1.$$

Beweis: Wir nehmen an, dass es eine Kante $e = \{u, v\}$ mit $x_e > 1$ existiert. Dann existiert eine Partition $\mathcal{S}$, die die Schnittkante e enthält und de Bedingung 2.22 mit Gleichheit erfüllt. (Sonst könnten wir x_e verringern.) Wir schreiben $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$, wobei ohne Einschränkung der Allgemeinheit durch Umsortierung $e \in \delta(S_1) \cap \delta(S_2)$ ist. Falls $|\mathcal{S}| = 2$ ist, ist $|\mathcal{S}| - 1 = 1$ und $x_e = 1$. Wir können also $|\mathcal{S}| > 2$ annehmen. Die Menge

$$\mathcal{S}' := \{S_1 \cup S_2, S_3, \dots, S_k\}$$

ist dann eine Partition, die nicht die Schnittkante e enthält. Es ist

$$\sum_{e' \in \delta(\mathcal{S}')} x_{e'} \leq \sum_{e' \in \delta(\mathcal{S})} x_{e'} - x_e = |\mathcal{S}| - 1 - x_e < |\mathcal{S}'| - 1$$

und die Bedinung $\mathcal{S}'$ ist verletzt. Dies ist ein Widerspruch. □

Wir können also das obige lineare Programm zum folgenden linearen Programm LP(mSB)

$$\begin{aligned} c \cdot x &\to \min \\ \sum_{e \in \delta(\mathcal{S})} x_e &\geq |\mathcal{S}| - 1, \forall \text{ Partitionen } \mathcal{S} \\ x &\geq 0 \end{aligned}$$

vereinfachen. Das zugehörige duale Programm DP(mSB) hat die folgende Gestalt

$$\begin{aligned} \sum_{\mathcal{S}} (|\mathcal{S}| - 1) y_{\mathcal{S}} &\to \max \\ \sum_{\mathcal{S}: e \in \delta(\mathcal{S})} y_{\mathcal{S}} &\leq c_e, \forall e \in E \\ y &\geq 0. \end{aligned}$$

Der Algorithmus von Kruskal in Satz 2.43 erhöht in jeder Iteration alle dualen Variablen $S_1, \ldots, S_k$ einer Partition $\mathcal{S}$. Wir erhöhen stattdessen die duale Variable der Partition. Statt zum Zeitpunkt $\frac{c_e}{2}$ werden die Kanten nun zum Zeitpunkt c_e betrachtet. Der Algorithmus wählt also die gleichen Kanten.

Satz 2.47 (Kruskal)**.** *Es sei $G = (V, E, c)$. Der Algorithmus von Kruskal berechnet ein optimale primale und duale Lösung zum* CP(mSB) *und* DP(mSB) *für das minimale Spannbaum-Problem.*

Beweis: Es sei ALG der Algorithmus von Kruskal. Wie im Satz 2.26 ist

$$c(\text{ALG}) = \sum_{\mathcal{S}} |\text{ ALG} \cap \delta(\mathcal{S})| y_{\mathcal{S}}.$$

Es bezeichne $\mathcal{S}_i$ die in der i-ten Iteration gewählten Partition und e_i die gewählte Kante. In der Iteration i hat die Teillösung $X_i = \{e_1, \ldots, e_{i-1}\}$ genau $i-1$ viele Kanten, die keinen Zyklus bilden. Somit existieren genau $n - i + 1$ Zusammenhangskomponenten und es ist $|\mathcal{S}_i| = n - 1 + 1$. Die Teillösung X_i hat die minimale Erweiterung ALG $\cup \{e_1, \ldots, e_{i-1}\}$ und jede minimale Erweiterung Y hat $n - i$ Schnittkanten, das heißt es ist

$$|Y \cap \delta(\mathcal{S}_i)| = n - i.$$

Es folgt

$$c(\text{ALG}) = \sum_{i=1}^{l} |ALG \cap \delta(\mathcal{S}_i)| y_{\mathcal{S}_i} = \sum_{i=1}^{l} (n - i) y_{\mathcal{S}_i} = \sum_{\mathcal{S}} (|\mathcal{S}| - 1) y_{\mathcal{S}}$$

und wir haben die Optimalität beider Lösungen gezeigt. □

Der schon erwähnte Dual-Fitting-Algorithmus von Prim [Pri57] löst das minimale Spannbaum-Problem auch optimal.

2.2.5 Der minimale Steiner-Wald

Wir betrachten nun das Problem der Berechnung eines minimalen Steiner-Baumes.

Problem 2.48 (Steiner-Baum)**.**
***Instanz**: Ein Netzwerk $G = (V, E, c)$ und eine Menge von Knoten T.*
***Lösung**: Ein Teilnetzwerk, so dass alle Knoten in T zusammenhängend sind.*
***Maß**: Summe der Kantengewichte des Teilnetzwerkes.*
***Ziel**:* min.

Die Knoten in T heißen **Terminals**. Robins und Zelikovsky [RZ00] haben eine 1.55-Approximation für dieses Problem angegeben. Unter der Voraussetzung, dass $co-\mathcal{RP} \neq \mathcal{NP}$ ist, besitzt dieses Problem keinen Aproximationsalgorithmus mit einer Güte besser als 1.007 (siehe [Thi01]). Wir werden einen Primal-Dual-Algorithmus mit Approximationsgüte von 2 für das allgemeinere Problem des Steiner-Waldes angeben.

Problem 2.49 (Steiner-Wald)**.**
Instanz*: Ein Netzwerk $G = (V, E, c)$ und eine Familie $\mathcal{T} = \{T_1, \ldots, T_m\}$ von Knotenmengen T_i.*
Lösung*: Ein Netzwerk, so dass für alle i die Menge T_i zusammenhängend ist.*
Maß*: Summe der Kantengewichte des Teilnetzwerkes.*
Ziel*:* min*.*

Die Knoten der Menge $V \setminus \bigcup_i T_i$ heißen **Steiner-Knoten**. Agrawal, Klein und Ravi ([AKR91]) haben für dieses Problem einen impliciten Primal-Dual-Algorithmus mit der Approximationsgüte 2 angegeben.

Lemma 2.50. *Das Steiner-Wald-Problem ist ein Netzwerk-Problem* CP(f) *mit der folgenden Funktion*

$$f(S) := \begin{cases} 1 & \text{falls } \exists i \text{ mit } \emptyset \neq S \cap T_i \neq T_i \\ 0 & \text{sonst} \end{cases}$$

Beweis: Leicht. □

Satz 2.51 (Goemans, Williamson)**.** *Es sei ein Netzwerk-Problem* CP(f) *mit einer binären, properen Funktion f gegeben. Der Algorithmus* PDg *ist ein Primal-Dual-Algorithmus mit der Approximationsgüte* 2*.*

Beweis: Wir wissen schon, dass PDg für binäre, propere Funktionen in Polynomialzeit ist. Wir wollen Satz 2.39 benutzen und müssen für jede unzulässige Lösung X und minimale Erweiterung Y von X die Ungleichung

$$\sum_{S \in V(X)} |Y \cap \delta(S)| \leq 2|V(X)|$$

zeigen. Da f proper ist, folgt nach Lemma 2.30, dass

$$V(X) = \{C \mid C \text{ ist eine Zusammenhangskomponente von } (V, X) \text{ mit } f(C) = 1\}$$

ist. Wir definieren einen Graphen $H = (V_H, E_H)$. Zu jeder Zusammenhangskomponente C_v in (V, X) korrespondiert genau ein Knoten v in H. Zwei Knoten u und v sind in H adjazent, falls die zugehörigen Zusammenhangskomponenten C_v und C_w in (V, Y) eine gemeinsame Kante haben.
Wir löschen in H alle Knoten mit Grad 0. Wir behaupten, dass für jedes Blatt v in H die Bedingung $f(C_v) = 1$ gilt. Denn sei v ein Blatt in H mit $f(C_v) = 0$ und e die zu v inzidente Kante in H. Es sei D die Zusammenhangskomponente in H, die den Knoten v enthält. Da Y zulässig ist, folgt $f(D) = 0$. Aus Lemma 2.33 folgt $f(D \setminus C_v) = 0$. Da Y eine minimale Erweiterung ist, ist $Y \setminus \{e\}$ keine gültige Lösung und die Zusammenhangskomponente C_v oder $D \setminus C_v$ ist verletzt. Dies ist ein Widerspruch und wir haben die Behauptung gezeigt.
Da H maximal $|V_H| - 1$ viele Kanten hat und jeder Knoten $v \in V_H$ mit $f(C_v) = 0$ einen Grad größer gleich 2 hat, folgt

$$\begin{aligned} \sum_{S \in V(X)} |Y \cap \delta(S)| &= \sum_{v \in V_H : f(C_v) = 1} |\delta_H(v)| = \sum_{v \in V_H} |\delta_H(v)| - \sum_{v \in V_H : f(C_v) = 0} |\delta_H(v)| \\ &\leq 2(|V_H| - 1) - 2(|V_H| - |V(X)|) = 2|V(X)| - 2. \end{aligned}$$

□

Wir zeigen, dass für binäre, propere Funktionen die Reihenfolge der Löschung egal ist.

Lemma 2.52. *Es sei X eine zulässige und zyklenfreie Lösung für* CP(f) *mit einer binären, properen Funktion f. Weiterhin sei*

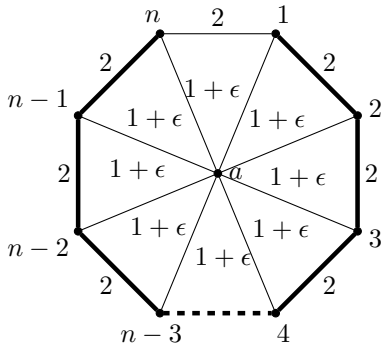
$$Y := \{e \in X \mid X \setminus \{e\} \text{ ist zulässig}\}.$$

Dann ist $X \setminus Y$ zulässig.

Beweis: Es sei D eine Zusammenhangskomponenten von $(V, X\backslash Y)$. Es ist $D \subseteq C$ für eine Zusammenhangskomponente von (V, X). Wir bezeichnen die Schnittkanten von D in X mit $e_1, \ldots, e_k$. Durch das Löschen der Kante i zerfällt C in die zwei Zusammenhangskomponenten D_i und $C\backslash D_i$. Ohne Einschränkung der Allgemeinheit ist $D \subseteq C\backslash D_i$. Aus $e_i \in Y$ folgt $f(D_i) = 0$ (und $f(C\backslash D_i) = 0$). $D, D_1, \ldots, D_k$ ist eine Partition von C. Es folgt $f(C\backslash D) = f(D_1 \cup \ldots \cup D_k) = 0$. Da auch $f(C) = 0$ ist, folgt mit Lemma 2.33 $f(D) = 0$. Da für alle Zusammenhangskomponente D von $(V, X\backslash Y)$ die Bedingung $f(D) = 0$ gilt, ist die Lösung $X\backslash Y$ zulässig. □

Das nächste Beispiel zeigt, dass sogar im Steiner-Wald-Problem die Approximationsgüte von PDg im Allgemeinen mindestens 2 ist.

Beispiel 2.53. *Wir betrachten das folgende Steiner-Wald-Problem.*



Die Menge der Terminals sind alle Knoten, bis auf den mittleren Knoten a.
Der Algorithmus PDg *erhöht alle dualen Variablen der Terminals um* 1. *Nun sind alle äußeren Kanten dicht und es werden* $n-1$ *viele äußere Kanten zur Lösung hinzugefügt. Wir haben sie im Schaubild dick markiert. Die Kosten dieser Lösung betragen* $\text{pdg} = 2(n-1)$.
Die optimale Lösung benutzt alle zu a inzidenten Kanten und hat die Kosten

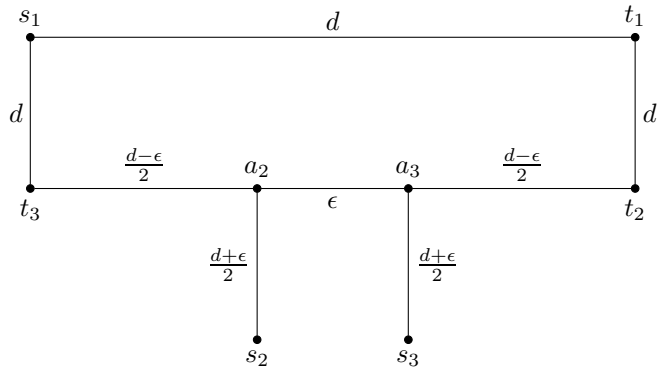
$$\text{opt} = n(1+\epsilon) \to n \text{ für } \epsilon \to 0.$$

Es ist

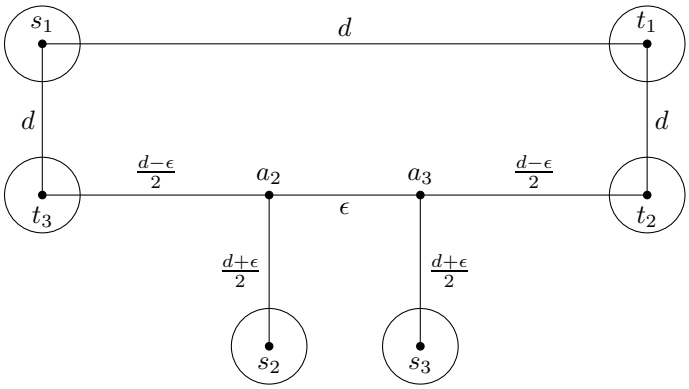
$$\frac{2(n-1)}{n} \to 2 \text{ für } n \to \infty.$$

Wir betrachten jetzt ein Beispiel, welches wir später benötigen werden.

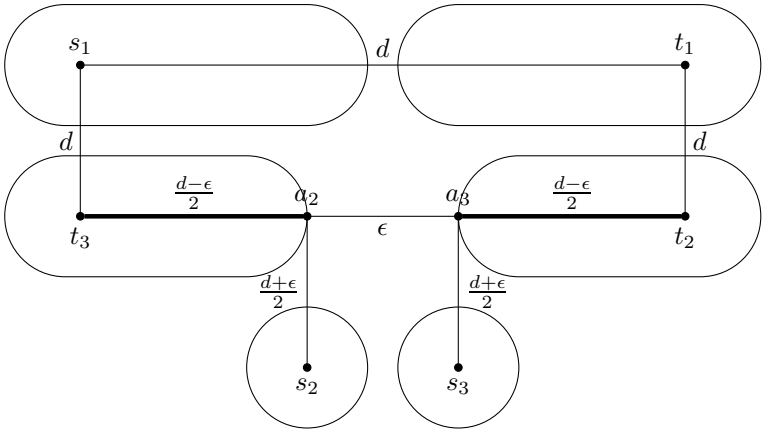
Beispiel 2.54. *Es sei der folgende Graph gegeben.*



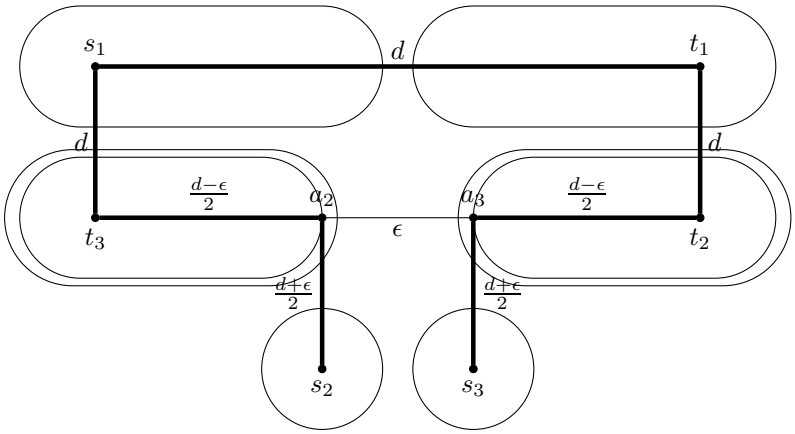
Die Paare (s_i, t_i), $i = 1, 2, 3$ *müssen verbunden werden und* a_1, a_2 *sind Steiner-Knoten. Zum Zeitpunkt* $t = 0$ *sind die Mengen* $\{s_i\}, \{t_i\}$, $i = 1, 2, 3$ *minimal verletzt.*



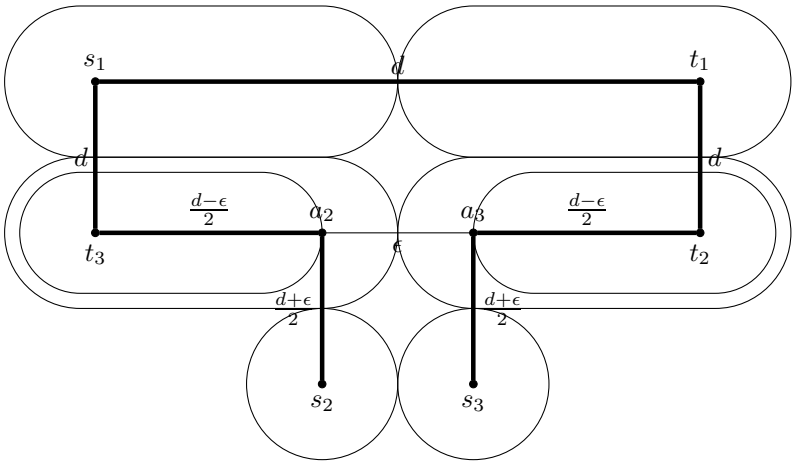
Zum Zeitpunkt $t = \frac{d-\epsilon}{2}$ sind die Kanten (t_3, a_2) und (a_3, t_2) dicht und werden zur Lösung hinzugefügt.



Die neuen minimal verletzten Mengen sind nun $\{s_1\}$, $\{t_1\}$, $\{t_3, a_2\}$, $\{a_3, t_2\}$, $\{s_2\}$ und $\{s_3\}$:



Zum Zeitpunkt $t = \frac{d}{2}$ sind alle Kanten dicht. Ohne Einschränkung wird die Kante $\{a_2, a_3\}$ mit Gewicht ϵ nicht gewählt. (Dies können wir durch eine Gewichtsänderung auf $\epsilon + \epsilon'$ für ein kleines $\epsilon' > 0$ erreichen.) In der Lösung kann keine Kante gelöscht werden.



Der Wert der Lösung ist

$$\text{pdg} = \sum_e c_e x_e = 5d$$

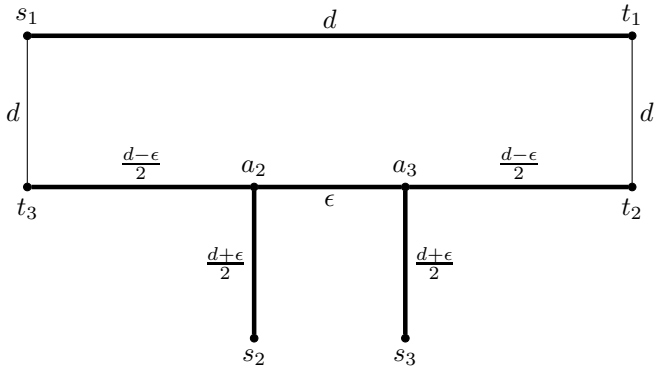
und der Wert der dualen Variablen beträgt

$$\sum_S f(S) y_S = 3d.$$

Es ist

$$3d \leq 5d \leq 2 \cdot 3d = 6d$$

und PDg *hat eine* 2*-Approximation berechnet. Die optimale Lösung hat Kosten* $\text{opt} = 3d + \epsilon$ *und ist im folgenden Schaubild dargestellt.*



2.2.6 Allgemeine propere Funktionen

Es sei nun der Bildbereich von f wieder $\mathbb{Z}$. Da unter Umständen

$$f_{\max} := \max_S f(S)$$

nicht in Polynomialzeit berechenbar ist, nehmen wir an, dass $f_{\max}$ in der Eingabe kodiert ist. Wir lösen das Netzwerk-Problem $\text{CP}(f)$ in $f_{\max}$ vielen Phasen für schwach-supermodulare Funktionen f.

Definition 2.55 (schwach-supermodular)**.** *Eine Funktion* $f : V \to \mathbb{Z}$ *ist* ***schwach-supermodular****, falls für alle Mengen* S, T

1. $f(S) + f(T) \leq f(S \cup T) + f(S \cap T)$ *oder*
2. $f(S) + f(T) \leq f(S \backslash T) + f(T \backslash S)$

ist.

Die schwach supermodularen Funktionen enthalten die wichtige Familie der properen Funktionen.

Lemma 2.56. *Jede propere Funktion f ist schwach-supermodular.*

Beweis: Für die beiden Mengen S, T ist

$$\begin{aligned}
&f(S) \leq \max\{f(S\backslash T), f(S \cap T)\} \\
&f(T) \leq \max\{f(T\backslash S), f(S \cap T)\} \\
&f(S) = f(S^C) \leq \max\{f(T\backslash S), f((S \cup T)^C)\} = \max\{f(T\backslash S), f(S \cup T)\} \text{ und} \\
&f(T) = f(T^C) \leq \max\{f(S\backslash T), f(S \cup T)\}.
\end{aligned}$$

Die Summation der beiden Gleichungen, die das Minimum von $f(S\backslash T)$, $f(T\backslash S)$, $f(S \cap T)$ und $f(S \cup T)$ enthalten, liefert die Behauptung. □

Problem 2.57 (Überlebensfähige Netzwerke)**.**
***Instanz**: Ein Netzwerk $G = (V, E, c)$ und eine Funktion $r : \{\{u, v\} \mid u, v \in V, u \neq v\} \to \mathbb{N}_0$.*
***Lösung**: Ein Teilnetzwerk, so dass für alle Knoten u, v, $u \neq v$ mindestens $r_{uv} := r(\{u, v\})$ viele kantendisjunkte Weg von u nach v existieren.*
***Maß**: Summe der Kantengewichte des Teilnetzwerkes.*
***Ziel**:* min.

Lemma 2.58. *Mit dem Satz von Menger können wir das Problem des minimalen überlebensfähigen Netzwerkes als Netzwerk-Problem* CP(f) *mit der properen Funktion*

$$f(S) := \max_{u \in S, v \notin S} r_{uv}$$

beschreiben.

Beweis: Aus $r_{uv} = r_{vu}$ folgt die Symmetrie von f.
Es seien S und T zwei disjunkte Mengen. Es ist

$$f(S \cup T) = \max_{u \in S \cup T, v \notin S \cup T} r_{uv}.$$

Aus Symmetriegründen könne wir ohne Einschränkung der Allgemeinheit

$$\max_{u \in S \cup T, v \notin S \cup T} r_{uv} = \max_{u \in S, v \notin S \cup T} r_{uv}$$

annehmen. Es folgt

$$f(S \cup T) = \max_{u \in S, v \notin S \cup T} r_{uv} \leq \max_{u \in S, v \notin S} r_{uv} = f(S).$$

Damit haben wir gezeigt, dass f proper ist. □

In jeder Phase wird eine Lösung für die Mengen gebildet, deren Bedarf an Schnittkanten am größten ist. Nach $f_{\max} := \max_S f(S)$ vielen Phasen haben wir dann eine zulässige Lösung gefunden. In der Phase 1 wird eine Teillösung X_1 für die Mengen S mit

$$f(S) = f_{\max} := \max_S f(S)$$

gesucht. Das Problem reduziert sich dann auf das Netzwerk-Problem CP(f_2) mit

$$f_2 := f - |\delta_{X_1}(S)|.$$

und wir können die Suche iterieren.
In Phase i haben wir dann eine Lösung X_i gefunden, die das Netzwerk-Problem CP(f_i) mit der Funktion $f(S) - f_{\max} + i$ erfüllt. Beginnend mit einer properen oder schwachen-supermodularen Funktion werden zeigen, dass in jeder Phase das zugehörige Netzwerk-Problem mit einer schwach-supermodularen Funktion beschrieben werden kann. Insgesamt wird $X_{f_{\max}}$ eine $2\mathcal{H}(f_{\max})$-Approximation werden.

Definition 2.59 (submodular)**.** *Eine Funktion $g : \mathcal{P}(V) \to \mathbb{Z}$ heißt **submodular**, falls für alle Mengen S und T*

1. *$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ und*
2. *$f(S) + f(T) \geq f(S\backslash T) + f(T\backslash S)$ ist.*

Beispiel 2.60. *Es sei $G = (V, E)$ ein Graph und $x \in \mathbb{Z}^E$ eine nicht negative Kantenfunktion. Dann ist*

$$f : \mathcal{P}(V) \to \mathbb{Z}, S \mapsto \sum_{e \in \delta(S)} x_e$$

submodular.

Beweis: Wir definieren

$$\delta(S, T) := \{e = \{u, v\} \mid u \in S\backslash T, v \in T\backslash S\}$$

und die Funktion

$$f : \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}, (S, T) \mapsto \sum_{e \in \delta(S,T)} x_e$$

für die Mengen $S, T \subseteq V$. Es ist

$$\begin{aligned}
f(S) &= f(S\backslash T, T\backslash S) + f(S\backslash T, (S \cup T)^C) + f(S \cap T, T\backslash S) + f(S \cap T, (S \cup T)^C) \\
f(T) &= f(T\backslash S, S\backslash T) + f(T\backslash S, (S \cup T)^C) + f(S \cap T, S\backslash T) + f(S \cap T, (S \cup T)^C) \\
f(S \cap T) &= f(S \cap T, S\backslash T) + f(S \cap T, T\backslash S) + f(S \cap T, (S \cup T)^C) \\
f(S \cup T) &= f(S\backslash T, (S \cup T)^C) + f(T\backslash S, (S \cup T)^C) + f(S \cap T, (S \cup T)^C) \\
f(S\backslash T) &= f(S\backslash T, T\backslash S) + f(S\backslash T, S \cap T) + f(S\backslash T, (S \cup T)^C) \text{ und} \\
f(T\backslash S) &= f(T\backslash S, S\backslash T) + f(T\backslash S, S \cap T) + f(T\backslash S, (S \cup T)^C).
\end{aligned}$$

Es folgt

$$\begin{aligned}
f(S) + f(T) &= f(S \cap T) + f(S \cup T) + 2f(S\backslash T, T\backslash S) \\
f(S) + f(T) &= f(S\backslash T) + f(T\backslash S) + 2f(S \cap T, (S \cup T)^C)
\end{aligned}$$

Aus

$$f(S\backslash T, T\backslash S), f(S \cap T, (S \cup T)^C) \geq 0$$

folgt die Behauptung. □

Insbesondere ist für jeden Untergraphen G' die Funktion $|\delta_{G'}(\cdot)| = \sum_{e \in \delta_{G'}(\cdot)} 1$ submodular.

Lemma 2.61. *Es sei f schwach-supermodular und g submodular. Dann ist $f - g$ schwach supermodular.*

Beweis: Es seien die beiden Mengen $S, T \subseteq V$ gegeben.

1. Im Fall $f(S)+f(T) \leq f(S\cap T)+f(S\cup T)$ ist $(f-g)(S)+(f-g)(T) \leq (f-g)(S\cap T)+(f-g)(S\cup T)$ und
2. im Fall $f(S)+f(T) \leq f(S\backslash T)+f(T\backslash S)$ ist $(f-g)(S)+(f-g)(T) \leq (f-g)(S\backslash T)+(f-g)(T\backslash T)$.

□

In jeder Phase werden wir ein Netzwerk-Problem CP(h) für eine binäre unkreuzbare Funktion lösen.

Definition 2.62 (unkreuzbar)**.** *Eine Funktion $h : \mathcal{P}(V) \to \{0, 1\}$ heißt **unkreuzbar**, falls die folgende Bedingung erfüllt ist. Aus $h(S) = h(T) = 1$ für zwei Mengen S und T folgt*

1. *$h(S \cap T) = h(S \cup T) = 1$ oder*
2. *$h(S\backslash T) = h(T\backslash S) = 1$.*

Wie bei den vorherigen Funktionen für Netzwerk-Probleme setzen wir immer $h(V), h(\emptyset) \leq 0$ voraus.

Beispiel 2.63. *Es sei ein Netzwerk $G = (V, E, c)$ und zwei Knoten s, t gegeben. Die Funktion (2.14)*

$$h(S) := \begin{cases} 1 & \text{falls } s \in S, t \notin S \\ 0 & \text{sonst} \end{cases},$$

die das Problem des kürzesten Weges formuliert, ist unkreuzbar.

Beweis: Es sei $h(S) = h(T) = 1$ für zwei Mengen S und T. Somit ist $s \in S, T$ und es ist $t \notin S, T$. Es folgt $s \in S \cap T, S \cup T$ und $t \notin S \cap T, S \cup T$ und $h(S \cap T) = h(S \cup T) = 1$. □

Lemma 2.64. *Jede binäre propere Funktion f ist unkreuzbar. Aus disjunkt-dominiert folgt im Allgemeinen nicht unkreuzbar.*

Beweis: Es sei $f(S) = f(T) = 1$. Es ist

$$\begin{aligned} 1 = f(S) &\leq \max\{f(S\backslash T), f(S \cap T)\} \\ 1 = f(T) &\leq \max\{f(T\backslash S), f(S \cap T)\} \\ 1 = f(S^C) &\leq \max\{f(S^C\backslash T^C), f(S^C \cap T^C)\} = \max\{f(T\backslash S), f(S \cup T)\} \\ 1 = f(T^C) &\leq \max\{f(T^C\backslash S^C), f(S^C \cap T^C)\} = \max\{f(S\backslash T), f(S \cup T)\}. \end{aligned}$$

Durch Betrachtung der beiden Gleichungen, die das Minimum von $f(S \cap T)$, $f(S \cup T)$, $f(S\backslash T)$ und $f(T\backslash S)$ enthält, folgt die erste Behauptung.
Es sei $V = \{1, 2, 3\}$ und $f(1, 2) = f(2, 3) = f(2) = 1$ und 0 sonst. Diese Funktion ist disjunkt dominiert, aber nicht unkreuzbar für die beiden Mengen $S = \{1, 2\}$ und $T = \{2, 3\}$. □

Lemma 2.65. *Die minimalen verletzten Menge einer unkreuzbaren Funktion h einer unzulässigen Teillösung X sind disjunkt.*

Beweis: Annahme es existieren zwei nicht disjunkte minimal verletzte Mengen S und T. Es ist $h(S) = h(T) = 1$ und $\delta_X(S) = \delta_X(S) = \emptyset$. Aus der Submodularität von $|\delta_X(\cdot)|$ folgt

1. $\delta_X(S \cap T) = \delta_X(S \cap T) = \emptyset$ und
2. $\delta_X(S\backslash T) = \delta_X(T\backslash S) = \emptyset$.

Aus der Unkreuzbarkeit von h folgt mindestens einer der beiden folgenden Fälle.

1. Im Fall $h(S \cap T) = h(S \cup T) = 1$ ist die Menge $S \cap T$ eine echte Teilmenge von S und T, die verletzt ist.
2. Im Fall $h(S\backslash T) = h(T\backslash S) = 1$ ist die Menge $S\backslash T$ eine echte Teilmenge von S, die verletzt ist.

Beide Fälle erzeugen einen Widerspruch und somit ist die Behauptung bewiesen. □

Dieses Lemma hat gezeigt, dass es in jeder Phase nur polynomiell viele minimale verletzte Mengen geben kann. Leider sind die minimalen Mengen im Allgemeinen nur schwer zu berechnen. So ist zum Beispiel die Funktion, die nur für eine Menge S den Wert 1 hat, unkreuzbar. Das Suchen dieser Menge S durch Probieren würde exponentiell lange dauern. Wir nehmen daher an, dass ein Orakel für CP(f) existiert, welches in einem Zeittakt alle minimalen verletzten Mengen mit $f(S) = f_{\max}$ zu einer (unzulässigen) Lösung X ausgibt. Für propere Funktion ist ein Orakel in Polynomialzeit mit Hilfe Gomory-Hu-Schnitt-Berechnungen gegeben (siehe [GGW98]).

Lemma 2.66. *Es sei $f \geq 0$ schwach supermodular und $f_{\max} := \max_S f(S)$. Die Funktion*

$$h : \mathcal{P}(V) \to \mathbb{Z},\ h(S) := \begin{cases} 1 & \text{falls } f(S) = f_{\max} \\ 0 & \text{sonst} \end{cases}$$

ist unkreuzbar.

Beweis: Es sei $h(S) = h(T) = 1$ für die beiden Mengen S und T. Es folgt $f(S) = f(T) = f_{\max}$. Im Fall $f(S)+f(T) \leq f(S\cap T)+f(S\cup T)$ folgt $f(S\cap T)+f(S\cup T) \geq 2f_{\max}$. Somit ist $f(S\cap T) = f(S\cup T) = f_{\max}$ und $h(S \cap T) = h(S \cup T) = 1$. Der andere Fall ist analog. (Da $f(V)$, $f(\emptyset) \leq 0$ ist, folgt auch $h(V)$, $h(\emptyset) \leq 0$.) □

Wir zeigen nun, dass ein Primal-Dual-Algorithmus mit Approximationsgüte 2 für unkreuzbare Funktionen h existiert.

Definition 2.67 (Zeuge, Zeugenfamilie)**.** *Es sei ein Netzwerk-Problem* CP(h) *mit einer unkreuzbaren Funktion h gegeben. Weiterhin sei X eine unzulässige Lösung, $V(X)$ die Familie der minimalen verletzten Mengen und Y eine minimale Erweiterung von X. Eine Menge $S \subseteq V$ ist ein* ***Zeuge*** *der Kante*

$$e \in \bigcup_{C \in V(X)} \delta_Y(C),$$

falls

1. $h(S) = 1$,
2. $\delta_Y(S) = \{e\}$ *und*
3. *für alle* $C \in V(X)$ *ist* $C \subseteq S$ *oder* $C \cap S = \emptyset$

ist. $\mathcal{S} \subseteq \mathcal{P}(V)$ *ist eine* ***Zeugenfamilie****, falls* $\mathcal{S}$ *aus Zeugen besteht und für jede Kante*

$$e \in \bigcup_{C \in V(X)} \delta_Y(C)$$

genau ein Zeuge $S_e \in \mathcal{S}$ *existiert.*

Lemma 2.68. *Es sei ein Netzwerk-Problem* CP(h) *mit einer unkreuzbaren Funktion h gegeben. Weiterhin sei X eine unzulässige Lösung, $V(X)$ die Familie der minimalen verletzten Mengen und Y eine minimale Erweiterung von X. Dann existiert eine Zeugenfamilie.*

Beweis: Da für alle $e \in X$ und S mit $e \in \delta(S)$ die Bedingung S nicht verletzt ist, ist jede Kante $e \in \bigcup_{C \in V(X)} \delta_Y(C)$ in $Y \backslash X$. Da Y eine minimale Erweiterung von X ist, würde das Löschen einer Kante $e \in Y \backslash X$ eine Bedingung S_e verletzen. Es ist somit $h(S_e) = 1$ und $\delta_Y(S_e) = \{e\}$. Da jedes $C \in V(X)$ minimal verletzt bzgl. X ist, folgt $C \subseteq S_e$ oder $C \cap S = \emptyset$. □

Definition 2.69 (laminar, kreuzen)**.** *Eine Familie $\mathcal{S}$ von Mengen heißt* ***laminar****, falls für alle Mengen* $S, T \in \mathcal{S}$

1. $S \subseteq T$,
2. $T \subseteq S$ *oder*
3. $S \cap T = \emptyset$

ist. Falls $\mathcal{S}$ nicht laminar ist, dann existieren zwei Mengen S und T, die sich ***kreuzen****, das heißt es ist* $S \not\subseteq T$, $T \not\subseteq S$ *und* $S \cap T \neq \emptyset$.

Lemma 2.70. *Es sei ein Netzwerk-Problem* CP(h) *mit einer unkreuzbaren Funktion h gegeben. Weiterhin sei X eine unzulässige Lösung, $V(X)$ die Familie der minimalen verletzten Mengen und Y eine minimale Erweiterung von X. Dann existiert eine laminare Zeugenfamilie.*

Beweis: Nach Lemma 2.68 existiert eine Zeugenfamilie $\mathcal{S}$. Falls in $\mathcal{S}$ zwei Mengen S und T existieren, die sich kreuzen, entkreuzen wir diese. Aus $h(S) = h(T) = 1$ folgt

$$h(S \cap T) = h(S \cup T) = 1 \text{ oder } h(S \backslash T) = h(T \backslash S) = 1.$$

Im ersten Fall ersetzen wir S und T durch $S \cap T$ und $S \cup T$ und im zweiten Fall durch $S \backslash T$ und $T \backslash S$. Dadurch verringert sich die Anzahl der sich kreuzenden Mengen. Denn falls die Menge A die Menge

S und T kreuzt, dann kreuzt A maximal zwei der neuen Mengen. Falls A die Menge S kreuzt und $A \not\subseteq T$ ist, dann kreuzt A nicht die Menge $S \cap T$ und $T \backslash S$. Falls A die Menge S kreuzt und $A \subseteq T$ ist, dann kreuzt A nicht die Menge $S \cup T$ und $S \backslash T$. Durch das Entkreuzen der Mengen S und T verringert sich die Anzahl der kreuzenden Mengen mindestens um eins. Wir müssen noch zeigen, dass die Zeugen-Eigenschaft erhalten bleibt. Die Eigenschaft $h(S) = 1$ bleibt klarerweise erhalten. Es seien zwei Kanten e_1 und e_2 mit den zwei sich kreuzenden zugehörigen Zeugen S_{e_1} und S_{e_2} gegeben. Wir nehmen an, dass wir uns im Fall

$$h(S_{e_1} \cap S_{e_2}) = h(S_{e_1} \cup S_{e_2}) = 1$$

befinden. Aus der Submodularität von $|\delta_Y(\cdot)|$ folgt

$$2 = |\delta_Y(S_{e_1}) + \delta_Y(S_{e_2}) \geq |\delta_Y(S_{e_1} \cap S_{e_2})| + |\delta_Y(S_{e_1} \cup S_{e_2})|.$$

Da Y zulässig ist folgt $|\delta_Y(S_{e_1} \cap S_{e_2})| \geq 1$ und $|\delta_Y(S_{e_1} \cup S_{e_2})| \geq 1$. Insgesamt ist $|\delta_Y(S_{e_1} \cap S_{e_2})| = 1$ und $|\delta_Y(S_{e_1} \cup S_{e_2})| = 1$ Die Mengen $S_{e_1} \cap S_{e_2}$ und $S_{e_1} \cup S_{e_2}$ sind verletzt und somit ist jede minimale verletzte Menge eine Teilmenge oder disjunkt von den Mengen. Der Fall $h(S_{e_1} \backslash S_{e_2}) = h(S_{e_1} \backslash S_{e_1}) = 1$ ist analog. Durch Iterieren erhalten wir eine laminare Zeugenfamilie. □

Wir fügen zu einer laminaren Zeugenfamilie $\mathcal{S}$ die Menge V hinzu und können den folgenden Baum $B = (V_B, E_B)$ definieren. Die Knotenmenge ist

$$V_B := \{v_S \mid S \in \mathcal{S} \cup \{V\}\}$$

Falls T die minimale echte Obermenge von S ist, so sind die beiden Knoten v_S und v_T durch die ungerichtete Kante in $e = \{v_S, v_T\} \in E_B$ verbunden. Für jede minimal verletzte Menge C existiert eine minimale Menge S in $\mathcal{S} \cup \{V\}$ mit $C \subseteq S$. Wir sagen, dass C zu dem Knoten v_S **korrespondiert** und nennen den Knoten v_S **aktiv**.

Lemma 2.71. *Der Baum B hat höchstens ein nicht aktives Blatt.*

Beweis: Nur V und minimale Mengen von $\mathcal{S}$ können Blätter definieren. Jede minimale Menge S von $\mathcal{S}$ ist verletzt und enthält deswegen eine minimal verletzte Menge C und ist somit aktiv. Der Knoten v_V könnte nicht aktiv sein. □

Beispiel 2.72. *Entgegen der Behauptung in [WGMV95] kann v_V ein Blatt und inaktiv sein. Es sei der Graph $G = (V, \{e\})$ mit der Knotenmenge $V = \{s, t\}$ und einer einzigen Kante $e = \{s, t\}$ mit den Kantenkosten $c_e = 1$ gegeben. Die Funktion*

$$f(S) := \begin{cases} 1 & \text{falls } S = \{s\} \\ 0 & \text{sonst} \end{cases}$$

ist unkreuzbar (siehe Beispiel 2.63). Die einzige verletzte Menge bezüglich $X = \emptyset$ ist $C = \{s\}$. Die einzige minimale Erweiterung ist $Y = \{e\}$ und $\mathcal{S} = \{C\}$ ist eine laminare Zeugenfamilie. Der Baum B besteht nur aus den beiden Blättern y_C und y_V. Der Knoten y_C ist aktiv und der Knoten y_V ist nicht aktiv.
Wir zeigen auch, dass V ein aktives Blatt sein kann. Wir betrachten wieder den obigen Graphen G. Die Funktion

$$h(S) := \begin{cases} 1 & \text{falls } |S \cap \{s, t\}| = 1 \\ 0 & \text{sonst} \end{cases}$$

ist unkreuzbar. Zu $X = \emptyset$ ist $Y = \{e\}$ die einzige minimale Erweiterung. Die beiden Mengen $\{s\}$ und $\{t\}$ sind minimal verletzt und $\mathcal{S} = \{\{s\}\}$ ist eine laminare Zeugenfamilie. Das Blatt y_V ist durch die minimale verletzte Menge $\{t\}$ aktiv.

Lemma 2.73. *Es sei v_S aktiv und C die zugehörige korrespondierende aktive Menge. Es ist*

$$|\delta_B(v_S)| \geq |\delta_Y(C)|.$$

01 $X_0 \leftarrow \emptyset$
02 FÜR $p \leftarrow 1$ BIS $f_{\max}$
03 $\quad g_p \leftarrow f - |\delta_{X_{p-1}}(\cdot)|$
04 $\quad h_p(S) \leftarrow \begin{cases} 1 & \text{falls } g_p(S) = \max\limits_T g_p(T) \\ 0 & \text{sonst} \end{cases}$
05 $\quad E_p \leftarrow E \setminus X_{p-1}$
06 $\quad$ Es sei X die von PDg berechnete Lösung bezüglich des Graphen (V, E_p) und der unkreuzbaren Funktion h_p.
07 $\quad X_p \leftarrow X_{p-1} \cup X$
08 Ausgabe $X_{f_{\max}}$

Abbildung 2.5: Der Algorithmus GW

Beweis: Wir betrachten die folgende bijektive Abbildung

$$\begin{array}{lllll} \delta_Y(C) & \rightarrow & \mathcal{S} & \rightarrow & E_B \\ e & \mapsto & S_e & \mapsto & (v_{S_e}, v_T), \end{array}$$

wobei S_e den Zeugen zu e in $\mathcal{S}$ und T die echte minimale Obermenge von S_e im $\mathcal{S}$ bezeichne. Es sei $e \in \delta_Y(C)$. Wir befinden uns in einem der beiden Fälle.

1. Die Menge S_e enthält die Menge C. Falls C zur Menge $S' \neq S$ korrespondiert, folgt $S' \subset S$ und $e \in \delta(S')$. Die Menge S' wäre auch ein Zeuge für e. Dies ist ein Widerspruch. Also korrespondiert C zu S_e.

2. Die Menge S_e enthält nicht die Menge C. Da T kein Zeuge von e ist, folgt $C \subseteq T$. Falls C zur Menge $T' \neq T$ korrespondiert, folgt $S \subseteq T'$, da T' kein Zeuge für die Kante e ist. Weiterhin ist $T' \subset T$ und T' wäre die minimale Obermenge von S. Also korrespondiert C zu T.

Da C zu S_e oder T korrespondiert, ist $S = S_e$ oder $S = T$ und es folgt die Behauptung. □

Satz 2.74. *Es sei ein Netzwerk-Problem* CP(h) *mit einer unkreuzbaren Funktion h gegeben. Der Algorithmus* PDg *ist eine* 2*-Approximation.*

Beweis: Es sei X eine unzulässige Lösung und Y eine minimale Erweiterung von X. Nach Lemma 2.71 haben alle nicht aktiven Mengen mindestens den Grad 2, bis auf ein Blatt. Es ist

$$\sum_{v_S \in V_B : v_S \text{ ist nicht aktiv}} |\delta(v_S)| \geq 2(|V_B| - |V(X)| - 1) + 1.$$

Es folgt

$$\begin{aligned} \sum_{C \in V(X)} |\delta_Y(C)| &\leq \sum_{v_S \in V_B : v_S \text{ ist aktiv}} |\delta_B(v_S)| = \sum_{v_S \in V_B} |\delta(v_S)| - \sum_{v_S \in V_B : v_S \text{ ist nicht aktiv}} |\delta(v_S)| \\ &\leq 2(|V_B| - 1) - 2(|V_B| - |V(X)| - 1) - 1 = 2|V(X)| - 1 \end{aligned}$$

und mit Satz 2.39 die Behauptung. □

Wir kehren jetzt zu einer properen Funktion f zurück. Wir lösen CP(f) durch den folgenden Algorithmus GW in Abbildung 2.5. Da die unkreuzbaren Funktionen eine Teilfamilie der schwach-supermodularen Funktionen sind, können wir die Mengen S mit $f(S) = f_{\max}$ im Allgemeinen nicht in Polynomialzeit berechnen. Wir gehen davon aus, dass ein Orakel existiert, welches zu einer (unzulässigen) Lösung alle minimalen verletzten Mengen S mit $f(S) = f_{\max}$ ausgibt. Für propere Funktionen existiert solch ein Orakel in Polynomialzeit (siehe [GGW98]). $\max\limits_T g_p(T)$ ist zu Beginn $f_{\max}$ und wird in jeder Phase genau um eins dekrementiert.

Lemma 2.75. *Es sei X_{p-1}, die von GW in Phase $p-1$ berechnete Teillösung und (x,y) die von* PDg *bzgl. h_p und E_p berechnete Lösung. Weiterhin sei*

$$z_e := \begin{cases} \sum\limits_{S:e\in\delta(S)} y_S & \text{für } e \in X_{p-1} \\ 0 & \text{sonst} \end{cases}$$

Dann ist (y,z) eine zulässige Lösung von DP(f) *mit*

$$\sum_{e\in E} z_e = \sum_S (f(S) - f_{\max} + p - 1) y_S.$$

Beweis: Zur Erinnerung es ist DP(f)

$$\begin{aligned} \sum_{S\subseteq V} f(S) y_S - \sum_{e\in E} z_e &\to \max \\ \sum_{S:e\in\delta(S)} y_S &\le c_e + z_e, \forall e \in E \\ y, z &\ge 0. \end{aligned}$$

und der Algorithmus PDg berechnet eine Lösung von DP(h_p)

$$\begin{aligned} \sum_{S\subseteq V} h_p(S) y_S &\to \max \\ \sum_{S:e\in\delta(S)} y_S &\le c_e, \forall e \in E_p \\ y &\ge 0. \end{aligned}$$

Für $e \in X_{p-1}$ ist nach Definition von z_e

$$\sum_{S:e\in\delta(S)} y_S \le c_e + z_e$$

und für $e \in E\backslash X_{p-1} = E_p$ ist

$$\sum_{S:e\in\delta(S)} y_S \le c_e.$$

Es ist

$$\sum_{e\in E} z_e = \sum_{e\in X_{p-1}} \sum_{S:e\in\delta(S)} y_S = \sum_S |\delta_{X_{p-1}}(S)| y_S$$

Aus $y_S > 0$ folgt $h_p(S) = 1$ und $g_p(S) = f_{\max} - p + 1$. Da auch $g_p(S) = f(S) - |\delta_{X_{p-1}}(S)|$ ist, folgt $|\delta_{X_{p-1}}(S)| = f(S) - f_{\max} + p - 1$ und die Behauptung. □

Satz 2.76. *Der Algorithmus GW berechnet für schwach-supermodulare Funktionen f eine $2\mathcal{H}(f_{\max})$-Approximation, wobei $\mathcal{H}(n)$ die n-te harmonische Zahl ist.*

Beweis: Es sei y_S die von PDg in Phase p berechnete duale Lösung. Nach Lemma 2.75 ist

$$\text{opt}(\text{DP}(f)) \ge \sum_S f(S) y_S - \sum_e z_e = \sum_S (f(S) - f(S) + f_{\max} - p + 1) y_S = \sum_S (f_{\max} - p + 1) \sum_S y_S.$$

Durch Summation über alle Phasen und mit Satz 2.74 folgt

$$\sum_{e\in X_{f_{\max}}} c_e \le 2 \sum_{p=1}^{f_{\max}} \frac{1}{f_{\max} - p + 1} \,\text{opt}(\text{DP}(f)) = 2\mathcal{H}(f_{\max}) \,\text{opt}(\text{DP}(f))$$

und die Behauptung. □

```
01 X ← ∅
02 g ← f
03 SOLANGE g ≱ 0:
04     F ← E\ X
05     Finde eine optimale Punkt-Lösung x für LP(g) bzgl. G = (V, F).
06     E_{1/2} = {e | x_e ≥ 1/2 }
07     X ← X ∪ E_{1/2}
08     g ← g − |δ_{E_{1/2}}(·)|
09 AUSGABE X
```

Abbildung 2.6: Der Algorithmus von Jain

2.3 Der Algorithmus von Jain

Wir betrachten nun den Algorithmus von Jain [Jai01] für das gleiche Problem. Dieser Algorithmus berechnet eine optimale Lösung des relaxierten primalen Problems. In dieser Lösung werden alle e mit $x_e \geq \frac{1}{2}$ auf 1 gerundet. Dieser Vorgang wird dann auf dem Restnetzwerk fortgesetzt. Der Algorithmus ist in der Abbildung 2.6 angegeben. Um in Zeile 05 eine optimale Lösung zu finden, benutzen wir die Ellipsoid-Methode. Daher nehmen wir an, dass ein Separationsorakel für CP(f) existiert. Für propere Funktionen f existiert ein Orakel in Polynomialzeit. Die Schleife in 03 wird verlassen, wenn die Lösung zulässig ist und kann somit mit Hilfe des Separationsorakels überprüft werden. Eine optimale Lösung ist eine optimale Punkt-Lösung, wenn die Lösung $|E|$ linear unabhängige Nebenbedingungen mit Gleichheit erfüllt. Daher können wir mit dem folgenden Algorithmus eine optimale Lösung in eine optimale Punkt-Lösung wandeln.

```
01 SOLANGE x keine Punkt-Lösung ist:
02     Es sei U der affine Raum der Nebenbedingungen die mit Gleichheit bzgl. der
         Lösung x erfüllt sind.
03     Wähle eine Gerade aus U die durch x geht.
04     Finde einen der Endpunkte y der Gerade, die das Polyhedron schneiden.
05     x ← y
```

Wir zeigen, dass sich das Separationsorakel für f auf g überträgt.

Lemma 2.77. *Es sei f schwach supermodular und ein Separationsorakel für* LP(f) *des Netzwerkes $G = (V, E, C)$ gegeben. Für $X \subseteq E$ existiert ein Separationsorakel für* LP($f - |\delta_X(\cdot)|$) *des Netzwerkes $G = (V, E\backslash X, c)$.*

Beweis: Es sei $\overline{x} \in \mathbb{R}^{E\backslash X}$ eine (unzulässige) Lösung von CP($f - |\delta_X(\cdot)|$). Weiterhin sei $x \in [0,1]^E$ der Vektor $\overline{x}$, der in allen Koordinaten $e \in X\backslash E$ mit 1 erweitert wurde. Es ist $\overline{x}$ das Bild von x bezüglich der Projektion $\overline{\ }: \mathbb{R}^E \to \mathbb{R}^{E\backslash X}$. Der Vektor $\overline{x}$ erfüllt genau dann

$$\sum_{e \in \delta_{E\backslash X}(S)} \overline{x}_e \geq f(S) - |\delta_X(S)|, \forall S \subseteq V$$

wenn der Vektor x

$$\sum_{e \in \delta_E(S)} x_e \geq f(S), \forall S \subseteq V$$

erfüllt. Falls das Orakel von f eine x und den Lösungspolyeder von CP(f) trennende Hyperebene ausgibt, dann trennt die Projektion der Hyperebene auf $\mathbb{R}^{E\backslash X}$ den Vektor $\overline{x}$ und den Lösungspolyeder von CP($f - |\delta_X(\cdot)|$). Die projektierte Hyperebene ist wieder eine Hyperebene. □

Satz 2.78. *Es sei x eine optimale Lösung von* LP(f) *und $E_{\frac{1}{2}} = \{e \mid x_e \geq \frac{1}{2}\}$. Falls Y eine ganzzahlige Lösung von* LP($f - |\delta_{E_{\frac{1}{2}}}(\cdot)|$) *mit Wert kleiner gleich* $2\,\mathrm{opt}(\mathrm{LP}(f - |\delta_{E_{\frac{1}{2}}}(\cdot)|))$ *ist, dann ist $Y \cup E_{\frac{1}{2}}$ eine ganzzahlige Lösung von* LP(f) *mit Wert kleiner gleich* $2\,\mathrm{opt}(\mathrm{LP}(f))$.

Beweis: Es ist leicht zu sehen, dass $Y \cup E_{\frac{1}{2}}$ eine ganzzahlige Lösung von $\mathrm{LP}(f)$ ist. x eingeschränkt auf $\mathbb{R}^{E \setminus E_{\frac{1}{2}}}$ ist auch eine Lösung von $\mathrm{LP}(f - |\delta_{E_{\frac{1}{2}}}(\cdot)|)$ und es folgt

$$\mathrm{opt}(\mathrm{LP}(f - |\delta_{E_{\frac{1}{2}}}(\cdot)|) \leq \mathrm{opt}(\mathrm{LP}(f)) - \sum_{e \in E_{\frac{1}{2}}} c_e x_e$$

und

$$2\,\mathrm{opt}(\mathrm{LP}(f)) \geq 2\,\mathrm{opt}(\mathrm{LP}(f - |\delta_{E_{\frac{1}{2}}}(\cdot)|) + \sum_{e \in E_{\frac{1}{2}}} c_e 2x_e.$$

Noch Voraussetzung ist $2\,\mathrm{opt}(\mathrm{LP}(f - |\delta_{E_{\frac{1}{2}}}(\cdot)|)) \geq \sum_{e \in Y} c_e$ und für $e \in E_{\frac{1}{2}}$ ist $2x_e \geq 1$. Deshalb folgt

$$2\,\mathrm{opt}(\mathrm{LP}(f)) \geq \sum_{e \in Y} c_e + \sum_{e \in E_{\frac{1}{2}}} c_e$$

und die Behauptung. □

Korollar 2.79. *Die* IP*-Lücke von* $\mathrm{CP}(f)$ *beträgt für schwach supermodulare Funktionen* f *genau* 2.

Beweis: Der obige Satz zeigt, dass die IP-Lücke höchstens 2 ist. Lemma 2.44 zeigt die andere Richtung. □

Es bleibt noch zu zeigen, dass der obige Algorithmus terminiert. Dafür müssen wir sicherstellen, dass jede optimale Punkt-Lösung x mindestens eine Kante e mit $x_e \geq \frac{1}{2}$ besitzt. Wir verweisen hierfür auf [Jai01].

Kapitel 3

Prioritäts- und Stapel-Algorithmen

3.1 Einführung

In diesem Kapitel betrachten wir das Modell der Prioritäts- und Stapel-Algorithmen. Durch die Einführung eines Gegners erhalten wir untere Schranken für den kompetitiven Faktor dieser Algorithmenklassen. Viele bekannte Greedy-Algorithmen wie die Algorithmen von Kruskal, Prim oder Dijkstra sind Stapel-Algorithmen. Weiterhin gehören die Primal-Dual [GW95], [GW97], [Wil02] die Local-Ratio-Algorithmen [BYBFR04] und Dual-Fitting-Algorithmen [FR04], [JMM$^+$03] zu den Stapel-Algorithmen. Das Modell der Prioritäts-Algorithmen wurden von Borodin, Nielson und Rackoff [BNR02] und Angelopoulos und Borodin [AB04] entwickelt und von Davis und Impagliazzo [DI05a], [DI05b] auf Graphen erweitert. Borodin, Cashman und Magen [BCM05] haben die Prioritäts-Algorithmen zu den Stapel-Algorithmen erweitert. Wir werden hier beide Modelle und einige Resultate aus den oben genannten Artikeln vorstellen und erweitern. Die Prioritäts/Stapel-Algorithmen erfüllen die folgenden Punkte:

- Die Instanz oder Teile der Instanz werden in eine Menge von Eingabedaten zerlegt.
- Die Ausgabe des Algorithmuses ist eine Menge von Entscheidungen. Jede Entscheidung wird über ein einzelnes Eingabedatum gemacht und jedes Eingabedatum der Instanz wird entschieden.
- Der Algorithmus definiert eine totale Ordnung auf allen möglichen Eingabedaten und
- betrachtet die Eingabedaten in der Reihenfolge dieser Ordnung.
- Beim Treffen der Entscheidung kennt der Algorithmus nur das aktuellen Eingabedatum und alle vorherigen Eingabedaten mit den zugehörigen Entscheidungen.

Bei Greedy-Algorithmen ist die Ordnung durch die Kosten der Eingabedaten definiert. Bis auf Punkt drei treffen auch alle Punkte für Online-Algorithmen zu. Bei Online-Algorithmen ist die Ordnung kanonisch durch die Ankunftszeiten der Eingabedaten definiert. Daher stammen viele der folgenden Begriffe aus diesem Bereich (siehe [BEY98] für einen guten Überblick). Ein fester Prioritäts/Stapel-Algorithmus definiert zu Beginn des Algorithmuses die Ordnung und verwendet in jeder Iteration die gleich Ordnung. Ein adaptiver Prioritäts/Stapel-Algorithmus kann in jeder Iteration eine neue Ordnung definieren. Ein Stapel-Algorithmus führt zusätzlich in der End-Phase einen inversen Löschschritt durch, welcher sicherstellt, dass die Lösung minimal bzgl. der Inklusion ist.
Wie die Instanz in Eingabedaten zerlegt ist problemabhängig und es gibt Probleme, in denen es mehre Möglichkeiten der Definition der Eingabedaten gibt. Wir werden daher bei den Problemen folgende Notation verwenden, die wir am Beispiel des Steiner-Waldes-Problems verdeutlichen.

Problem 3.1 (Steiner Wald)**.**
***Instanz**: Hier wird die Instanz beschrieben. Beim Steiner-Wald-Problem ist es ein Netzwerk $G = (V, E, c)$ und eine Familie von Terminalmengen $T_i \subseteq V$.*
***Eingabe**: Die Eingabedaten sind dem Algorithmus nicht bekannt und werden in Reihenfolge der vom Algorithmus gewählten Ordnung präsentiert. So sind die Eingabedaten beim Steiner-Wald-Problem die Kanten e mit den zugehörigen Kosten c_e.*

```
EINGABE: I = {γ_1, ..., γ_n} ⊆ Γ
AUSGABE: Lösung S = {(γ_i, σ_i) | i = 1, ..., n}
01 S ← ∅
02 l ← 0
03 Wähle eine totale Ordnung ≤ auf Γ.
04 Ordne I entsprechender totalen Ordnung I = {γ_π(l) ≤ γ_π(l+1) ≤ ... ≤ γ_π(n)}.
05 SOLANGE I ≠ ∅:
06 l ← l + 1
07 Wähle σ_π(l) für das kleinste unentschiedene Eingabedatum γ_π(i) basierend
08    auf γ_π(l) und den vorherigen Eingabedaten und Entscheidungen (γ_π(i), σ_π(i)), i < l.
09    S ← S ∪ {(γ_π(l), σ_π(l))}
10    I ← I\{γ_π(l)}
11 AUSGABE: S
```

Abbildung 3.1: Der feste Prioritäts-Algorithmus

```
EINGABE: I = {γ_1, ..., γ_n} ⊆ Γ
AUSGABE: Lösung S = {(γ_i, σ_i) | i = 1, ..., n}
01 S ← ∅
02 l ← 1
03 SOLANGE I ≠ ∅:
04    l ← l + 1
05    Wähle eine totale Ordnung ≤_l auf Γ basierend auf den vorherigen betrachteten
06    Eingabedaten und Entscheidungen (γ_π_i(i), σ_π_i(i)), i < l.
07    Ordne I entsprechend der Ordnung ≤_l: I = {γ_π_l(l) ≤_l ... ≤_l γ_π_l(n)}
08    Wähle σ_π_l(l) für das kleinste unentschiedene Eingabedatum γ_π_l(l) basierend auf
09    γ_π_l(l) und den vorherigen Eingabedaten und Entscheidungen (γ_π_i(i), σ_π_i(i)), i < l.
10    S ← S ∪ {(γ_π_l(l), σ_π_l(l))}
11    I ← I\{γ_π_l(l)}
12 AUSGABE: S
```

Abbildung 3.2: Der adaptive Prioritäts-Algorithmus

Information*: Unter diesem Punkt werden die Teile der Instanz angegeben, die dem Algorithmus bekannt sind. Beim Steiner-Wald-Problem ist es die Familie der Terminalmengen.*
Entscheidung*: Im Steiner-Wald-Problem gibt die Entscheidung σ_e an, ob eine Kante e akzeptiert oder abgelehnt wird.*
Lösung*: Ist eine Teilgraph von G, in dem für alle i die Terminalmenge T_i zusammenhängend ist.*
Maß*: Ist in diesem Fall, die Summe der Kantengewichte des Teilgraphens.*
Ziel*:* min

Falls die Entscheidung Akzeptieren und Ablehnen ist, werden wir sie meistens nicht angeben. In den Abbildungen 3.1 und 3.4 sind die Prioritäts-Algorithmen dargestellt.

Bei den Prioritäts-Algorithmen haben wir die Entscheidungsmöglichkeiten nicht eingeschränkt, um Probleme wie zum Beispiel Vertex Coloring behandeln zu können. Für die Stapel-Algorithmen müssen wir leider die Entscheidungen $\sigma_i \in \{0, 1\}$ auf Ablehnen und Annehmen einschränken. Die Stapel-Algorithmen löschen vor der Ausgabe der Lösung noch überflüssige Elemente in umgekehrter Reihenfolge mit einem inversen Löschschritt. Sie sind in Abbildung 3.3 und 3.4 dargestellt.

Satz 3.2. *Die Primal-Dual-Algorithmen sind adaptive Prioritäts-Algorithmen und die Primal-Dual-Algorithmen mit einem inversen Löschschritt sind adaptive Stapel-Algorithmen.*

Beweis: Wir zeigen die Behauptung nur für Primal-Dual-Algorithmen mit einem inversen Löschschritt.

```
EINGABE: I = {γ_1, ..., γ_n} ⊆ Γ
AUSGABE: Lösung S ⊆ I
   01 S ← ∅
   02 l ← 1
   03 Wähle eine totale Ordnung ≤ auf Γ.
   04 Ordne I entsprechend der totalen Ordnung I = {γ_π(l) ≤ γ_π(l+1) ≤ ... ≤ γ_π(n)}.
   05 SOLANGE I ≠ ∅:
   06    l ← l + 1
   07    Wähle σ_π(l) für das kleinste unentschiedene Eingabedatum γ_π(l) basierend
   08    auf γ_π(l) und den vorherigen Eingabedaten und Entscheidungen (γ_π(i), σ_π(i)), i < l.
   09    FALLS σ_π(l) = 1 DANN S ← S ∪ {γ_π(l)}
   10    I ← I\{γ_π(l)}
   11 FÜR l = n BIS 1
   12    FALLS S\{γ_π_l(l)} zulässige Lösung ist S ← S\{γ_π_l(l)}
   13 AUSGABE S
```

Abbildung 3.3: Der feste Stapel-Algorithmus

```
   EINGABE: I = {γ_1, ..., γ_n} ⊆ Γ
   AUSGABE: Lösung S = {(γ_i, σ_i) | i = 1, ..., n}
   01 S ← ∅
   02 l ← 1
   03 SOLANGE I ≠ ∅:
   04    l ← l + 1
   05    Wähle eine totale Ordnung ≤_l auf Γ basierend auf den vorherigen betrachteten
   06    Eingabedaten und Entscheidungen (γ_π_i(i), σ_π_i(i)), i < l.
   07    Ordne I entsprechend der Ordnung ≤_l: I = {γ_π_l(l) ≤_l ... ≤_l γ_π_l(n)}
   08    Wähle σ_π_l(l) für das kleinste unentschiedene Eingabedatum γ_π_l(l) basierend auf
   09    γ_π_l(l) und den vorherigen Eingabedaten und Entscheidungen (γ_π_i(i), σ_π_i(i)), i < l.
   10    FALLS σ_π_l(l) = 1 DANN S ← S ∪ {γ_π_l(l)}
   11    I ← I\{γ_π_l(l)}
   12 FÜR l = n BIS 1
   13    FALLS S\{γ_π_l(l)} zulässige Lösung ist S ← S\{γ_π_l(l)}
   14 AUSGABE: S
```

Abbildung 3.4: Der adaptive Stapel-Algorithmus

```
01 S ← ∅
02 y ← 0
03 l ← 0
04 SOLANGE S unzulässig ist:
05    l ← l + 1
06    V ← {i | (Ax)_i < b_i} (die Menge der verletzten Bedingungen)
07    Erhöhe y_i für alle i ∈ V bis ein j mit (A^t y)_j = c_j existiert.
08    S ← S ∪ {j}
09 FÜR j = l BIS 1
10    FALLS S\{x_j} zulässige Lösung ist DANN S ← S\{x_j}
11 AUSGABE S
```

Abbildung 3.5: Der Primal-Dual-Algorithmus mit einem inversen Löschschritt für CP.

Der andere Fall folgt analog. Wir wollen folgendes allgemein formulierte Covering-Problem CP lösen:

$$\begin{aligned} c \cdot x &\to \min \\ Ax &\geq b \\ x &\in \{0,1\}^n \end{aligned}$$

Im Kontext eines Stapel-Algorithmuses lautet es:
Instanz: Eine nicht negative Matrix A und nicht negative Vektoren b und c.
Eingabe: (x_j, c_j)
Informationen: Der Algorithmus kennt die Matrix A und den Vektor b.
Entscheidung: Annehmen oder Ablehnen von x_j.
Lösung: $\{x \in \{0,1\}^n \mid Ax \geq b\}$.
Maß: $c \cdot x$.
Wenn wir die Bedingung $x \in \{0,1\}^n$ zu $x \geq 0$ relaxieren, erhalten wir das folgende duale Programm DP.

$$\begin{aligned} b \cdot y &\to \max \\ A^t y &\leq c \\ y &\geq 0 \end{aligned}$$

In Abbildung 3.5 ist der Primal-Dual-Algorithmus mit einem inversen Löschschritt dargestellt. Mit welcher Geschwindigkeit die y_i in Zeile 07 erhöht werden, ist vom jeweiligen Algorithmus abhängig und es können auch y_i existieren, die die Geschwindigkeit null haben und somit nicht erhöht werden. Der charakteristische Vektor $x = \mathbb{1}_S$ zu S ist dann eine Lösung des Integer-Programms.
In jedem Schleifen-Durchgang wird durch die Erhöhung der y_i in Zeile 07 durch den Zeitpunkt, wann $(A^t y)_j = c_j$ gilt, eine Ordnung auf den Eingabedaten (x_j, c_j) definiert. (Bei Gleichheit des Zeitpunktes ist durch die Wahl des Algorithmuses die Ordnung definiert.) Das kleinste (x_j, c_j) wird dann akzeptiert und sobald die Lösung zulässig wird, werden die verbleibenden unentschiedenen x_j implizit abgelehnt. Die letzen Schritte des Primal-Dual-Algorithmuses sind offensichtlich ein inverser Löschschritt und es folgt damit die Behauptung. □

Analog können wir zeigen, dass die Dual-Fitting-Algorithmen (siehe [JMM+03] und [FR04]) auch adaptive Stack-Algorithmen sind. Wir definieren nun das adaptive Spiel zwischen einem Stapel-Algorithmus ALG und einem Gegner ADV für Minimierungsprobleme. Es ist in Abbildung 3.6 dargestellt. Die Indices an den Mengen dienen nur für spätere Beweiszwecke und können weggelassen werden. Analog werden feste/adaptive Prioritäts/Stapel-Spiele definiert.

Satz 3.3. *Wenn ein adaptiven Stapel-Algorithmus mit Approximationsfaktor α existiert, dann existiert eine Gewinnstrategie A im obigen Spiel.*

Beweis: Dieser Beweis erfolgt analog zum Beweis in [DI05a] für ein Spiel um eine fest vorgegebene endliche Menge von Instanzen. Dort wird auch die Umkehrung bewiesen, die wir in unserem Spiel nicht beweisen können. Es sei ADV ein Gegner mit partieller Startinstanz Γ_1. Wir zeigen die Existenz einer

```
01 I_1 ← ∅ (I wird die Instanz)
02 S_1 ← ∅ (S wird die berechnete Lösung von A)
03 l ← 0
04 ADV wählt ein endliches Γ_1 ⊆ Γ
05 SOLANGE Γ_{l+1} ≠ ∅
06    l ← l + 1
07    A wählt Eingabedatum γ_l ∈ Γ_l und Entscheidung σ_l.
08    I_{l+1} ← I_l ∪ {γ_l}
09    S_{l+1} ← S ∪ {(γ_l, σ_l)}
10    ADV wählt Γ_{l+1} ⊆ Γ_l \ {γ_l}.
11 FÜR j = l BIS 1
12    FALLS S\{x_j} zulässige Lösung ist:
13        S ← S\{γ_j}
14 ADV wählt eine Lösung S_ADV bzgl. der Instanz I
15 FALLS S_ADV keine zulässige Lösung oder I keine zulässige Instanz ist, gewinnt A
16 FALLS S keine zulässige Lösung ist, gewinnt ADV
17 FALLS α ≥ c(S)/c(S_ADV) ist, gewinnt A sonst gewinnt ADV.
```

Abbildung 3.6: Das adaptive Stapel-Spiel

Strategie mit der folgenden Invariante. Für jede Iteration l und für alle Instanzen I mit $I_l \subset I$ und $I \backslash I_l \subseteq \Gamma_l$ entsprechen die $l-1$-ten von der Strategie A gewählten Eingabedaten mit Entscheidungen $(\gamma_i, \sigma_i) \in S_l$ den ersten $l-1$-ten vom Algorithmus gewählten Eingabedaten mit Entscheidungen bzgl. der Instanz I.
In der Iteration 1 gilt die Invariante, da noch keine Eingabedaten gewählt wurden. Wir zeigen den Induktionsschritt von l auf $l+1$. Es sei in der Iteration l im Spiel

$$I_l = \{\gamma_1, ..., \gamma_{l-1}\}$$

und

$$S_l = \{(\gamma_1, \sigma_1), ..., (\gamma_{l-1}, \sigma_{l-1}\}$$

gegeben und Γ_l die vom Gegner gestellte Instanz. Der Algorithmus definiert bzgl. der Menge S_l eine Ordnung auf Γ_l. Es sei γ_l das kleinste Element von Γ_l bzgl. dieser Ordnung. Für γ_l trifft der Algorithmus eine Entscheidung σ_l. Die Strategie A wählt auch (γ_l, σ_l). Es ist

$$I_{l+1} = \{\gamma_1, \ldots, \gamma_l\}$$

und

$$S_{l+1} = \{(\gamma_1, \sigma_1), ..., (\gamma_l, \sigma_l)\}$$

und der Gegner wählt ein $\Gamma_{l+1} \subseteq \Gamma_l \backslash \{\gamma_l\}$.
Für alle Instanzen I mit $I_{l+1} \subseteq I$ und $I \backslash I_{l+1} \subseteq \Gamma_{l+1}$ gilt trivialerweise $I_l \subseteq I$. Weiterhin ist

$$I \backslash I_l \subseteq \Gamma_{l+1} \cup \{\gamma_l\} \subseteq \Gamma_l \cup \{\gamma_l\} = \Gamma_l.$$

Nach Induktionsvoraussetzung sind die $l-1$-ten von der Strategie gewählten Eingabedaten mit Entscheidungen $(\gamma_i, \sigma_i) \in S_l$ gleich den vom Algorithmus gewählten Eingabedaten mit Entscheidungen bzgl. der Instanz I.
Nach Definition wurde γ_l von ALG bzgl. Γ_l und S_l im Schritt l gewählt. Da $\gamma_l \in I \backslash I_l \subseteq \Gamma_l$ ist, wird γ_l von ALG auch bzgl. der Instanz I gewählt. Es folgt die Invariante.
Falls ADV keine gültige Instanz I konstruiert, hat die Strategie A gewonnen. Im anderen Fall hat die Strategie A durch die Invariante die Kosten des Algorithmuses $c(ALG(I))$. Da ALG ein α-Approximationsalgorithmus ist, folgt

$$c(\mathrm{ALG}(I)) \leq \alpha c(\mathrm{OPT}(I)) \leq \alpha c(\mathrm{ADV}(I))$$

und die Strategie A hat das Spiel gewonnen. □

Wir werden die Kontraposition benötigen.

Korollar 3.4. *Es sei ein Gegner im obigen Spiel mit α gegeben, der gegen jede Strategie gewinnt. Dann gibt es keinen adaptiven Stapel-Algorithmus mit Approximationsfaktor kleiner oder gleich α.*

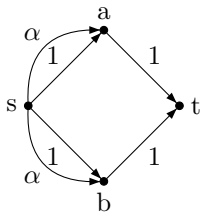
3.2 Kürzeste Wege

Wir betrachten nun verschiedene kürzeste Weg-Probleme. Wir betrachten zuerst das folgende schon in der Literatur behandelte Problem.

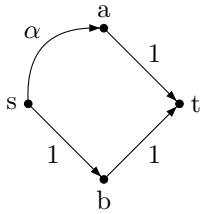
Problem 3.5 (Kürzester Weg in einem gerichteten Netzwerk)**.**
***Instanz**: Ein gerichtetes Netzwerk $G = (V, E, c)$ und zwei Knoten s, t.*
***Eingabe**: Kanten mit Gewicht (e, c_e).*
***Lösung**: Ein gerichteter Baum mit Wurzel s, der den Knoten t enthält.*
***Maß**: Summe der Kantengewichte auf dem Weg von s nach t im Baum.*
***Ziel**:* min.

Satz 3.6. *Kein fester Prioritäts-Algorithmus hat einen konstanten Approximationsfaktor für das kürzeste Weg-Problem 3.5 (siehe [DI05a]).*

Beweis: Der Gegner wählt die folgende Instanz mit $\alpha > 1$:



Die Instanz ist unzulässig, da z.B. zwischen den beiden Knoten s und a zwei Kanten mit den Gewichten 1 und α existieren. Durch Löschen geeigneter Kanten wird die Instanz später zulässig. Der Algorithmus wählt nun eine feste Ordnung $<$ auf den Eingabedaten. Aus Symmetriegründen können wir $(a, t) < (b, t)$ annehmen. Der Gegner wählt nun die zulässige Instanz Γ_1:



Es wird nun das feste Prioritäts-Spiel gespielt, bis der Algorithmus das Eingabedatum (a, t) wählt und dieses entscheiden muss. Wir betrachten die beiden möglichen Fälle.

- Der Algorithmus lehnt (a, t) ab. Der Gegner löscht in diesem Fall die Kante (b, t). Der Algorithmus kann keine zulässige Lösung finden, während der Gegner die Lösung $(s, a), (a, t)$ wählen kann.
- Der Algorithmus akzeptiert (a, t). Der Gegner löscht in diesem Fall keine Kanten und wählt die Lösung $(s, b), (b, t)$ mit Kosten 2. Der Algorithmus kann nur noch zwischen den beiden Bäumen mit den Kanten $(s, a), (a, t)$ und $(s, a), (a, t), (s, b)$ mit Kosten $\alpha + 1$ wählen. Der Approximationsfaktor ist daher mindestens $\frac{\alpha}{2} \to \infty$.

□

Im Satz 3.12 zeigen wir für ungerichtete Graphen, dass auch kein fester Stapel-Algorithmus einen konstanten Approximationsfaktor hat. Dieser Beweis lässt sich für gerichtete Graphen übertragen.

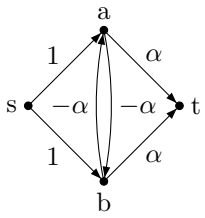
Satz 3.7. *Der Algorithmus von Dijkstra (bzw. Nicholson) ist ein adaptiver Prioritäts-Algorithmus mit der Approximationsgüte* 1 *für das kürzeste Weg-Problem 3.5 (siehe [Dij59],[Nic66] und [DI05a]).*

Beweis: Die beiden genannten Algorithmen sind ohne den Löschschritt Primal-Dual-Algorithmen ohne Löschschritt. Mit dem Satz 3.2 folgt die Behauptung. □

Analog ist der Algorithmus von Dijkstra (bzw. Nicholson) (mit Löschschritt) ein adaptiver Stapel-Algorithmus.

Satz 3.8. *Falls Netzwerke mit negativen Kantengewichten ohne negativen Zyklen im kürzester Wege Problem 3.5 zugelassen sind, hat kein adaptiver Prioritäts-Algorithmus einen konstanten Approximationsfaktor [DI05a].*

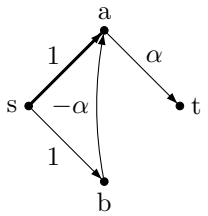
Beweis: Der Gegner wählt die folgende unzulässige Instanz Γ_1 mit $\alpha > 1$.



Wir unterscheiden die folgenden Fälle

1. Der Algorithmus wählt als erstes die Kante(s, a).

   (a) Der Algorithmus akzeptiert (s, a). Der Gegner präsentiert die folgende Instanz Γ_2:

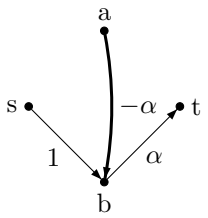
   

   Der Gegner kann die Lösung $\{(s, b), (b, a), (a, t)\}$ mit Kosten 1 wählen, während der Algorithmus nur noch die Lösungen $\{(s, a), (a, t)\}$, $\{(s, a), (a, t), (s, b)\}$ mit Kosten $\alpha + 1$ wählen kann. Der Approximationsfaktor beträgt $\frac{\alpha+1}{1} \to \infty$.

   (b) Der Algorithmus lehnt (s, a) ab. Der Gegner wählt die Instanz und Lösung $\{(s, a), (a, t)\}$. Der Algorithmus kann keine Lösung konstruieren.

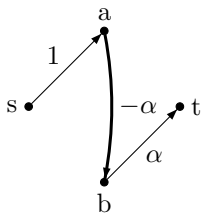
2. Der Algorithmus wählt als erstes die Kante (a, b).

   (a) Der Algorithmus akzeptiert (a, b). Der Gegner präsentiert die folgende Instanz

   und die Lösung $\{(s, b), (b, t)\}$. Der Algorithmus kann keine gültige Lösung finden.

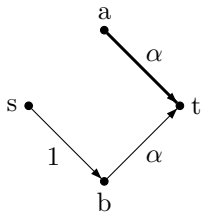
   (b) Der Algorithmus lehnt (a, b) ab. Der Gegner präsentiert die Instanz



und die Lösung $\{(s,a),(a,b),(b,t)\}$. Der Algorithmus kann keine gültige Lösung bilden.

3. Fall Der Algorithmus wählt zuerst die Kante (a,t).

   (a) Der Algorithmus akzeptiert die Kante (a,t). Der Gegner wählt die Instanz



   und die Lösung $\{(s,b),(b,t)\}$. Der Algorithmus kann keine Lösung finden.

   (b) Der Algorithmus lehnt die Kante (a,t) ab. In diesem Fall wählt der Gegner die Instanz und Lösung $\{(s,a),(a,t)\}$. Der Algorithmus kann keine Lösung finden.

Der Beweis für die Kanten (s,b), (b,a) und (b,t) ist analog. □

Dieses kürzeste Weg-Problem hat gezeigt, dass die Klasse der festen Prioritäts-Algorithmen eine echte Unterklasse der adaptiven Prioritäts-Algorithmen ist. Der Algorithmus von Bellman und Ford zeigt, dass die Klasse der adaptiven Prioritäts-Algorithmen eine echte Unterklasse der dynamischen Programmierungs-Algorithmen ist. Wir betrachten nun das natürlichste kürzester Wege-Problem in ungerichteten Graphen, wie es auch in [Dij59] oder [Nic66] definiert wurde:

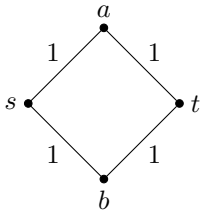
Problem 3.9 (Kürzester Weg)**.**
Instanz*: Ein Netzwerk $G=(V,E,c)$ und zwei Knoten $s,t \in V$.*
Eingabe*: Kanten mit Gewicht (e,c_e).*
Lösung*: Ein Weg von s nach t.*
Maß*: Summe der Kantengewichte (auf dem Weg von s nach t).*
Ziel*:* min.

Satz 3.10. *Kein fester Prioritäts-Algorithmus hat einen konstanten Approximationsfaktor für das kürzeste Wege Problem 3.9.*

Beweis: Der Beweis wird im nächsten Satz allgemeiner geführt. □

Satz 3.11. *Kein adaptiver Prioritäts-Algorithmus hat einen konstanten Approximationsfaktor für das kürzeste Weg-Problem 3.9.*

Beweis: Wir betrachten den folgenden Graph:



Wir unterscheiden die folgenden Fälle

1. Die Kante $\{s, a\}$ wird vom Algorithmus zuerst gewählt.

   (a) Falls der Algorithmus die Kante $\{s, a\}$ akzeptiert, löscht der Gegner die Kante $\{a, t\}$. Der Algorithmus kann keine Lösung finden und der Gegner hat den Lösungsweg (s, b, t).

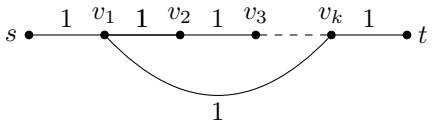
   (b) Falls der Algorithmus die Kante $\{s, a\}$ nicht akzeptiert, löscht der Gegner die Kanten des Weges (s, b, t). Der Algorithmus kann keine Lösung finden, während der Gegner den Lösungsweg mit den Knoten (s, a, t) wählt.

2. Die anderen Fälle sind analog.

□

Satz 3.12. *Kein fester Stapel-Algorithmus hat einen konstanten Approximationsfaktor für das kürzeste Weg-Problem 3.9.*

Beweis: Der Gegner wählt den vollständigen Graph K_n. Dann existiert eine Kante e mit $s, t \notin e$, die zuletzt unter diesen gewählt wird. Wir nennen diese Kante $\{v_1, v_k\}$ mit $k = n - 2$. Nach Löschen einiger Kanten, erhalten wir die folgende Instanz:



Die Kante $\{v_1, v_k\}$ wird nach allen Kanten $\{v_i, v_{i+1}\}$ gewählt.

1. Falls der Algorithmus eine Kante $\{v_i, v_{i+1}\}$ nicht akzeptiert, dann löscht der Gegner die Kante $\{v_1, v_k\}$ und präsentiert den Lösungsweg $(s, v_1, \ldots, v_k, t)$. Der Algorithmus kann keine Lösung finden.

2. Falls der Algorithmus alle $\{v_i, v_{i+1}\}$ akzeptiert hat, wird spätestens im inversen Löschschritt die Kante $\{v_1, v_k\}$ von der Lösung entfernt. Somit bleibt für den Algorithmus nur die Lösung $(s, v_1, \ldots, v_k, t)$ mit Kosten $k + 1$, während der Gegner den Weg (s, v_1, v_k, t) mit Kosten 3 wählt. Der Approximationsfaktor ist $\alpha \geq \frac{k+1}{3} = \frac{n-1}{3} \to \infty$.

□

Satz 3.13. *Die Algorithmen von Dijkstra und Nicholson sind adaptive Stapel-Algorithmen für das kürzeste Weg-Problem 3.9.*

Beweis: Die beiden genannten Algorithmen mit Löschschritt sind Primal-Dual-Algorithmen mit Löschschritt. Mit dem Satz 3.2 folgt die Behauptung. □

Wir betrachten nun das kürzester Weg Problemstellung wie im Artikel von [DI05a] allerdings in ungerichteten Graphen. Die Beweise von [DI05a] lassen sich leider nicht übernehmen. Aber wir können die obigen Beweise verwenden.

Problem 3.14. *[Kürzester Weg]*
***Instanz**: Ein Netzwerk $G = V, E, c)$ und zwei Knoten $s, t \in V$.*
***Eingabe**: Kanten mit Gewicht (e, c_e).*
***Lösung**: Ein Baum, der s und t verbindet.*
***Maß**: Summe der Kantenkosten des Weges im Baum, der s und t verbindet.*
***Ziel**:* min.

Wir haben die folgenden Sätze

Satz 3.15. *Kein fester Prioritäts-Algorithmus hat eine konstante Approximationsgüte für das kürzeste Weg-Problem 3.14.*

Beweis: Analog zum Beweis von 3.12 □

Satz 3.16. *Kein fester Stapel-Algorithmus hat eine konstante Approximationsgüte für das kürzeste Weg-Problem 3.14.*

Beweis: Analog zum Beweis von 3.12 □

Der Algorithmus von Dijkstra ist in diesem Fall ein adaptiver Prioritäts-Algorithmus und auch ein adaptiver Stapel-Algorithmus der Güte 1.
Falls ein Algorithmus beim kürzesten Wege-Problems einen Baum berechnet, ist es auch natürlich dem Algorithmus die gesamten Kosten dieses Baumes zu berechnen. Dies betrachten wir in der nächsten Problemstellung:

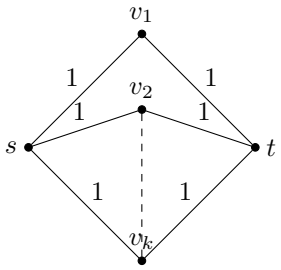
Problem 3.17 (Kürzester Weg Problem)**.**
***Instanz**: Ein Netzwerk $G = V, E, c)$ und zwei Knoten $s, t \in V$.*
***Eingabe**: Kanten mit Gewicht (e, c_e).*
***Lösung**: Ein Teilgraph, der s und t verbindet.*
***Maß**: Summe der Kantengewichte des Teilgraphens.*
***Ziel**:* min.

Satz 3.18. *Kein fester Prioritäts-Algorithmus hat einen konstanten Approximationsfaktor für das kürzeste Wege Problem 3.17.*

Beweis: Analog zum Beweis von 3.12 □

Satz 3.19. *Kein adaptiver Prioritäts-Algorithmus hat einen konstanten Approximationsfaktor für das kürzeste Weg-Problem 3.17.*

Beweis: Der Gegner wählt die folgende Instanz:



Falls der Algorithmus A eine Kante e wählt und nicht akzeptiert, löscht der Gegner alle Kanten bis auf die zwei Kanten, die den s, t-Weg über e definieren. Der Algorithmus kann keine Lösung finden und der Gegner hat den Weg über e als Lösung.
Falls der Algorithmus eine Kante akzeptiert, löscht der Gegner die zugehörige Kante, die den s, t-Weg vervollkommen würde. Dies kann $k - 1$-Mal durchgeführt werden. Der verbleibende s, t-Weg wird nicht gelöscht. Die Kosten des Algorithmuses sind $a = k + 1$ und der Gegner hat Kosten adv $= 2$. Der Approximationsfaktor beträgt $\alpha \geq \frac{k+1}{2} = \frac{n-1}{2} \to \infty$. □

Satz 3.20. *Kein fester Stapel-Algorithmus hat einen konstanten Approximationsfaktor für das kürzeste Weg-Problem 3.17.*

Beweis: Analog zum Beweis von 3.12 □

Satz 3.21. *Der Algorithmus von Dijkstra bzw. Nicholson ist ein adaptiver Stapel-Algorithmus.*

Beweis: Folgt aus dem Satz 3.2. □

Das nächste Problem zeigt, dass die Stapel-Algorithmen im Allgemeinen nicht besser als die Prioritäts-Algorithmen sind:

Weg(Weg)		
	Prioritäts	Stapel
fester	∞	∞
adaptiver	∞	1 Dijkstra

Baum(Weg)		
	Prioritäts	Stapel
fester	∞	∞
adaptiver	1 Dijkstra	1 Dijkstra

Baum(Alle)		
	Prioritäts	Stapel
fester	∞	∞
adaptiver	∞	1 Dijkstra

Zyklenfrei(Weg)		
	Prioritäts	Stapel
fester	∞	∞
adaptiver	1 Dijkstra	1 Dijkstra

Zyklenfrei(Alle)		
	Prioritäts	Stapel
fester	∞	∞
adaptiver	∞	1 Dijkstra

Teilgraph (Weg)		
	Prioritäts	Stapel
fester	1	∞
adaptiver	1(Dijkstra)	1 Dijkstra

Teilgraph(Alle)		
	Prioritäts	Stapel
fester	∞	∞
adaptiver	∞	1 Dijkstra

Abbildung 3.7: feste/adaptive Prioritäts/Stack-Algorithmen für das kürzeste Weg-Problem
Bei der Bezeichnung x(y) ist x ein typisches Element der Lösung und (y) gibt die Kostenberechnung an. Bei (Weg) werden die Kantenkosten des kürzesten Weges addiert, während bei (Alle) die Kosten aller Kanten in der Lösung addiert werden Die Einträge in den Tabellen geben den besten Approximationsfaktor gefolgt vom Algorithmus an.

Problem 3.22 (Kürzester Weg Problem)**.**
Instanz*: Ein Netzwerk $G = V, E, c)$ und zwei Knoten $s, t \in V$.*
Eingabe*: Kanten mit Gewicht (e, c_e).*
Lösung*: Ein Teilgraph, der s und t verbindet.*
Maß*: Summe der Kantengewichte des minimalen Weges im Teilgraph, der s und t verbindet.*
Ziel*:* min*.*

Satz 3.23. *Jeder Algorithmus mit einer beliebigen Ordnung, der alle Kanten akzeptiert, ist ein fester bzw. adaptiver Prioritäts-Algorithmus für das kürzeste Weg-Problem 3.22.*

Beweis: Offensichtlich. □

Satz 3.24. *Kein fester Stapel Algorithmus hat einen konstanten Approximationsfaktor für das kürzeste Weg-Problem 3.22.*

Beweis: Siehe Satz 3.12. □

Satz 3.25. *Der Algorithmus von Dijkstra ist ein adaptiver Stapel-Algorithmus mit Güte* 1 *für das kürzeste Weg-Problem 3.22.*

Die anderen Probleme bei denen die Lösung ein Baum, zyklenfreier Graph oder Teilgraph ist, der s und t verbindet können analog bewiesen werden. Wir geben eine Übersicht in Tabelle 3.7 an. Das erste Wort bezeichnet die Menge der Lösungen und das folgende Wort gibt die Art der Kostenberechnung an. Bei (Weg) werden die Kantenkosten des minimalen Weges der Lösung und bei (Alle) werden alle Kanten der Lösung addiert.

3.3 Minimale Spannbäume

In diesem Abschnitt betrachten wir das Problem des minimalen spannenden Baumes:

Problem 3.26 (Minimal spannender Baum)**.**
Instanz*: Ein Netzwerk $G = (V, E, c)$*
Eingabe*: Kanten mit Gewicht (e, c_e).*
Lösung*: Ein zusammenhängender Teilgraph.*
Maß*: Summe der Kantengewichte im Teilgraph.*
Ziel*:* min*.*

Satz 3.27. *Der Algorithmus von Kruskal [Kru56] ist ein fester Prioritäts-Algorithmus.*

Beweis: Der Beweis von Satz 2.43 und 3.2 zeigt, dass der Algorithmus ein Prioritäts-Algorithmus ist. Da für jede Kante e die dualen Variablen y_S von beiden inzidenten Knoten wachsen, ist der Zeitpunkt zu dem eine Kante gewählt wird genau $\frac{1}{2}c_e$. Dies ist unabhängig von den vorher gewählten Kanten und somit ist der Algorithmus ein fester Prioritäts-Algorithmus. □

Satz 3.28. *Der Algorithmus von Prim [Pri57] ist ein adaptiver Prioritäts-Algorithmus.*

Beweis: Da der Algorithmus von Prim zu Beginn einen beliebigen Knoten wählt und diesen iterativ durch Kanten erweitert, ist dieser Satz bei der oben genannten Problemstellung nicht ganz richtig. Denn der Algorithmus hat zu Beginn keine Informationen, welche Knoten existieren. Wir beheben dies, indem wir den Algorithmus von Prim abwandeln. In der ersten Iteration wählt der Algorithmus Prim die kostengünstigste Kante $e = \{u, v\}$ und erweitert danach wie im alten Algorithmus die Zusammenhangskomponente u, v. Der Algorithmus von Prim ist ein Dual-Fitting-Algorithmus und ein zu 3.2 analoger Beweis für Dual-Fitting-Algorithmen würde zeigen, dass der Algorithmus ein adaptiver Prioritäts-Algorithmus ist. □

Da die beiden Algorithmen minimale Lösungen ausgeben, sind sie auch zugleich Stapel-Algorithmen.

3.4 Minimale Steiner-Bäume

Wir betrachten nun das Problem des minimalen Steiner-Baumes:

Problem 3.29 (Steiner-Baum)**.**
Instanz*: Ein Netzwerk $G = (V, E, c)$ Eine Menge T von Terminals und eine Menge S von Steinerknoten mit $V = S \dot\cup T$*
Eingabe*: Kanten mit Gewicht (e, c_e).*
Informationen *Der Algorithmus kennt T, S und somit V.*
Lösung*: Ein Teilgraph, der alle Terminals verbindet.*
Maß*: Summe der Kantengewichte des Teilgraphens.*
Ziel*:* min*.*

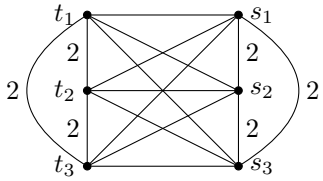
Im Falle eines metrischen Netzwerkes sind die beiden folgenden Sätze bekannt:

Satz 3.30. *Der Algorithmus von [KMB81] und [Ple81], der einen minimalen Spannbaum über alle Terminals berechnet, ist ein fester Prioritäts-Algorithmus mit der Approximationsgüte* 2*.*

Beweis: Siehe Satz 3.27 (bzw. Satz 3.28 für einen adaptiven Prioritätsalgorithmus). Die Approximationsgüte wird in [KMB81] und [Ple81] gezeigt. □

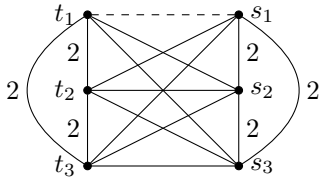
Satz 3.31. *Für metrische Netzwerke hat kein adaptiver Prioritäts-Algorithmus einen besseren Approximationsfaktor als $\frac{6}{5} = 1.2$ [DI05a].*

Beweis: Der Gegner ADV wählt die folgende unzulässige Instanz mit den Terminals $T = \{t_1, t_2, t_3\}$ und Steiner-Knoten $S = \{s_1, s_2, s_3\}$:

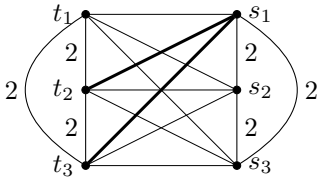


Zwischen einem Terminal t_i und Steiner-Knoten s_j ist eine Kante e mit Kosten $c_e = 1$ und eine weitere mit Kosten $c_e = \frac{4}{3}$. Der Algorithmus ALG wählt nun eine Kante. Wir betrachten nun alle Fälle.

1. Der Algorithmus wählt eine Kante $e = \{s_i, t_j\}$ mit $c_e = 1$. Ohne Einschränkung ist es die Kante $\{s_1, t_1\}$.

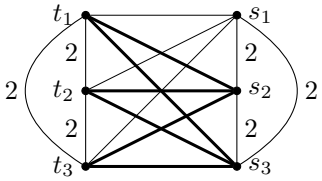


   (a) Falls der Algorithmus die Kante $\{s_1, t_1\}$ akzeptiert, dann löscht der Gegner die Kanten $\{s_1, t_i\}$, $i = 1, 2$ mit Kosten 1 und lässt nur die Kanten mit Gewicht $\frac{4}{3}$. Weiterhin löscht der Gegner die Kanten $\{s_3, t_i\}$, $i = 1, 2, 3$ mit Kosten $\frac{4}{3}$. Die Kanten mit Kosten $\frac{4}{3}$ sind durch dicke Linien und die Kanten mit Kosten 1 sind durch dünne Linien dargestellt.



   Der Gegner wählt die drei Kanten verbleibenden Kanten $\{s_3, t_i\}$, $i = 1, 2, 3$ mit Kantenkosten 1 als Lösung und hat Kosten adv = 3. Der Algorithmus kann die Lösung $\{s_1, t_i\}$, $i = 1, 2, 3$ mit Kosten alg $= 1 + 2 \cdot \frac{4}{3} = 3\frac{2}{3}$, die Lösung $\{s_1, t_1\}$, $\{s_3, t_i\}$, $i = 1, 2, 3$ mit Kosten alg = 4 oder eine noch schlechtere Lösung wählen. Der Approximationsfaktor beträgt mindestens $\alpha \geq \frac{\min\{3\frac{2}{3}, 4\}}{3} = \frac{11}{9} \geq \frac{6}{5}$.

   (b) Falls der Algorithmus die Kante $\{s_1, t_1\}$ ablehnt, dann löscht der Gegner die Kanten $\{s_1, t_i\}$, $i = 1, 2, 3$ mit Kosten $\frac{4}{3}$ und alle restlichen Kanten $\{s_i, t_j\}$ mit Kosten 1.



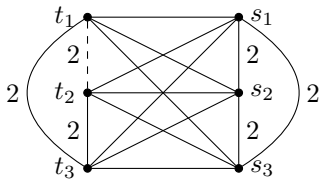
   Der Gegner wählt die Lösung $\{s_1, t_i\}$, $i = 1, 2, 3$ mit den Kantenkosten 1 und den Gesamtkosten opt = 3. Der Algorithmus kann die Lösung $\{t_1, t_2\}, \{t_2, s_1\}, \{t_3, s_1\}$ mit Kosten alg = 4, die Lösung $\{t_1, t_2\}, \{t_2, t_3\}$ mit Kosten alg = 4, die Lösung $\{s_2, t_i\}$, $i = 1, 2, 3$ mit Kosten alg = 4 oder ähnliche Lösungen mit Kosten 4 oder mehr wählen. Der Approximationsfaktor ist $\alpha \geq \frac{4}{3} \geq \frac{6}{5}$.

2. Der Algorithmus wählt eine Kante $e = \{s_i, t_j\}$ mit $c_e = \frac{4}{3}$. Ohne Einschränkung ist es die Kante $\{s_1, t_1\}$.

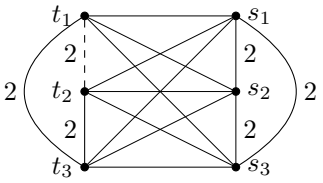
   (a) Falls der Algorithmus die Kante akzeptiert, verhält sich der Gegner wie im obigen Akzeptanz-Fall. Es folgt $\alpha \geq \frac{\min\{4, 4\frac{1}{3}\}}{3} = \frac{4}{3} > \frac{6}{5}$.

   (b) Falls der Algorithmus die Kante $\{s_1, t_1\}$ ablehnt, dann löscht der Gegner die Kanten $\{s_1, t_i\}$, $i = 2, 3$ mit Kosten $\frac{4}{3}$ und alle restlichen Kanten $\{s_i, t_j\}$ mit Kosten 1. Der Gegner wählt die Lösung $\{s_1, t_i\}$, $i = 1, 2, 3$ mit den Gesamtkosten opt $= 3\frac{1}{3}$. Der Algorithmus kann die Lösung $\{t_1, t_2\}, \{t_2, s_1\}, \{t_3, s_1\}$ mit Kosten alg $= 4$, die Lösung $\{t_1, t_2\}, \{t_2, t_3\}$ mit Kosten $alg = 4$, die Lösung $\{s_2, t_i\}$, $i = 1, 2, 3$ mit Kosten alg $= 4$ oder ähnliche Lösungen mit Kosten 4 oder mehr wählen. Der Approximationsfaktor ist $\alpha \geq \frac{4}{3\frac{1}{3}} = \frac{6}{5}$.

3. Der Algorithmus wählt eine Kante $\{t_i, t_j\}$ mit Kosten 2. Ohne Einschränkung sei es die Kante $\{t_1, t_2\}$.



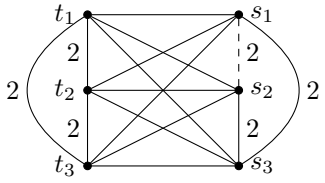
   (a) Falls der Algorithmus die Kante $\{t_1, t_2\}$ akzeptiert, löscht der Gegner alle Kanten $\{t_i, s_j\}$ mit Kosten $\frac{4}{3}$.



   Der Gegner kann die Lösung $\{s_1, t_i\}$, $i = 1, 2, 3$ mit Kosten opt $= 3$ wählen. Der Algorithmus hat mindestens die Kosten 4. Der Approximationsfaktor ist $\alpha \geq \frac{4}{3} \geq \frac{6}{5}$.

   (b) Falls der Algorithmus die Kante $\{t_1, t_2\}$ ablehnt, lässt der Gegner den Algorithmus fortfahren, bis einer der anderen Fälle eintritt.

   Der Algorithmus wählt eine Kante $\{s_i, s_j\}$ mit Kosten 2. Ohne Einschränkung sei es die Kante $\{s_1, s_2\}$.



   (a) Falls der Algorithmus die Kante $\{s_1, s_2\}$ akzeptiert, dann löscht der Gegner alle Kanten $\{t_i, s_j\}$ mit Kosten $\frac{4}{3}$. Der Gegner kann die Lösung $\{s_1, t_i\}$, $i = 1, 2, 3$ mit Kosten opt $= 3$ wählen. Der Algorithmus hat mindestens die Kosten 4. Der Approximationsfaktor ist $\alpha \geq \frac{4}{3} \geq \frac{6}{5}$.

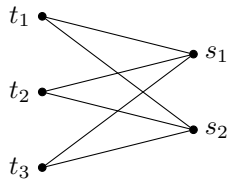
   (b) Falls der Algorithmus die Kante $\{s_1, s_2\}$ ablehnt, lässt der Gegner den Algorithmus fortfahren, bis einer der anderen Fälle eintritt.

□

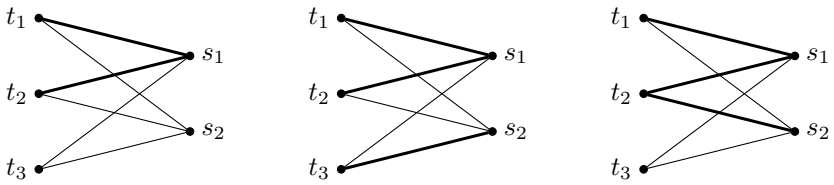
Im Artikel von [DI05a] ist ein adaptiver Prioritäts-Algorithmus mit Approximationsfaktor 1.75 für das Problem des metrischen Steiner-Baumes mit Gewichten in $[1, 2]$ angegeben. Wir betrachten nun wieder den allgemeinen Fall. [BCM05] haben den folgenden Satz bewiesen:

Satz 3.32. *Kein adaptiver Stapel-Algorithmus hat einen besserern Approximationsfaktor als* $\frac{4}{3}$.

Beweis: Der Gegner wählt die folgende Instanz mit Kantenkosten 1:



Die Menge der Terminals ist $\{t_1, t_2, t_3\}$ und die Menge der Steiner-Knoten ist $\{s_1, s_2\}$. Der Gegner löscht keine Kante bis zu dem Zeitpunkt, zu dem der Algorithmus zwei Kanten zu einem Steiner-Knoten s_i akzeptiert hat. Ohne Einschränkung der Allgemeinheit ist $s_i = s_1$ und der Algorithmus hat bis zu diesem Zeitpunkt alle gewählten Kanten akzeptiert. Denn falls der Algorithmus vor dem Zeitpunkt eine Kante $e = \{t_i, s_j\}$ nicht akzeptiert, dann löscht der Gegner eine Kante zu dem anderen Steiner-Knoten s_{j+1}. Diese existiert, da der obige Zeitpunkt noch nicht eingetreten ist. Der Algorithmus muss für seine Lösung beide Steiner-Knoten benutzen und hat somit mindestens die Kosten alg ≥ 4. Der Gegner wählt die minimale Lösung über den Steiner-Knoten s_j. Zu dem genannten Zeitpunkt sind wir nach Umnummerierung der t_i in einem der folgenden Fälle.

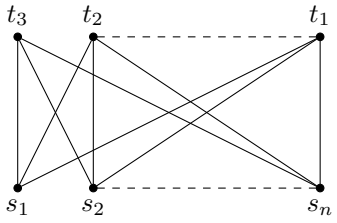


Die dicken Kanten kennzeichnen akzeptierte Kanten und die dünnen Kanten sind noch nicht betrachtete Kanten. Der Gegner löscht nun die Kante $\{t_3, s_1\}$. Der Algorithmus muss eine Lösung mit Mindestkosten 4 wählen. Falls der Algorithmus alle zu s_2 inzidenten Kanten akzeptiert, wird mindestens im inversen Löschschritt eine dieser inzidenten Kanten gelöscht. Der Gegner kann die optimale Lösung mit dem Steiner-Knoten s_2 wählen. Der Approximationsfaktor ist $\alpha \geq \frac{4}{3}$. □

Wir können die Lücke zum Algorithmus PDg schließen.

Satz 3.33. *Kein adaptiver Stapel-Algorithmus hat einen besseren Approximationsfaktor als* 2.

Beweis: Der Gegner wählt die folgende Instanz mit den Kantenkosten 1, den Terminals $T = \{t_1, \ldots t_n\}$ und Steiner-Knoten $S = \{s_1, \ldots, s_n\}$.



Die Anfangsinstanz besteht aus dem vollständigen bipartitem Graphen des Knotenpaares (T, S). Falls der Algorithmus eine Kante $e = \{t_i, s_j\}$ wählt und ablehnt, dann löscht der Gegner alle Schnittkanten der aktuellen Zusammenhangskomponente von t_i bis auf die Kante e. Der Algorithmus kann keine Lösung finden, während der Gegner die Zusammenhangskomponente, die Kante e und alle Kanten $\{s_j, t_k\}$ mit t_k nicht in der Zusammenhangskomponente bilden kann.
Wir nehmen deshalb an, dass der Algorithmus alle gewählten Kanten akzeptiert, bis er eine Lösung gefunden hat. Nach finden dieser Lösung kann der Algorithmus zum Löschschritt übergehen, da alle später zugefügten Kanten sowieso gelöscht würden.
Der Gegner wartet solange, bis der Algorithmus zwei Zusammenhangskomponenten, die beide jeweils Terminals enthalten, durch Hinzufügen einer Kante $e = \{t_i, s_j\}$ zu einer Zusammenhangskomponente zusammenschließt. In diesem Fall löscht der Gegner alle noch nicht betrachteten zu s_j inzidenten Kanten. Für die Lösung A und für jeden Steiner-Knoten s_j gilt nach dem Löschschritt $\delta_A(s_j) = 2$ oder $\delta_A(s_j) = 0$. Denn falls $\delta_A(s_j) > 2$ ist, dann existieren ohne Einschränkung der Allgemeinheit die drei Kanten $e_1 = \{t_1, s_j\}$, $e_2 = \{t_2, s_j\}$ und $e_3 = \{t_3, s_j\}$, die zu den Zeitpunkten $i_1 < i_2 < i_3$ hinzugefügt wurden und später nicht gelöscht werden. Zum Löschzeitpunkt von e_2 wir diese Kante nicht gelöscht und ist daher eine Kante, die die Zusammenhangskomponente von t_2 mit der Zusammenhangskomponente von t_1 verbindet. Da die Kante e_1 vor der Kante e_2 akzeptiert wurde, fügt die Kante e_2 beim Zeitpunkt der Hinzunahme auch die Zusammenhangskomponente von t_1 und die Zusammenhangskomponente von t_2 zusammen. Der Gegner löscht nun alle noch nicht betrachten Schnittkanten von s_j. Dies ist im Widerspruch zur späteren Hinzunahme der Kante e_3. Falls $\delta_A(s_j) = 1$ ist, verbindet die betreffende Kante keine Terminals und wird gelöscht.
Da die Lösung ein Baum ist, benutzt der Algorithmus genau $n - 1$ Steiner-Knoten und hat Kosten $2(n-1)$. Der Gegner kann über den unbenutzten Steiner-Knoten eine Lösung mit Kosten n konstruieren. Der Approximationsfaktor ist somit $\alpha \geq \frac{2n-2}{n} \to 2$ □

Der Algorithmus PDg ist ein $2 - \frac{2}{n}$-approximativer adaptiver Stapel-Algorithmus, wobei n die Anzahl der Terminals bezeichnet. Das heißt die obige untere Schranke ist dicht.

3.5 Minimale Steiner-Wälder

Wir betrachten nun das folgende Problem:

Problem 3.34 (minimaler Steiner-Wald)**.**
***Instanz**: Ein Netzwerk $G = (V, E, c)$, und eine Familie von Terminalmengen $T_i \subseteq V$.*
***Eingabe**: Kanten mit zugehörigen Kosten (e, c_e).*
***Informationen**: Der Algorithmus kennt die Familie der Terminalmengen und V.*
***Lösung**: Ein Teilgraph, der alle Terminals in jedem T_i verbindet.*
***Maß**: Summe der Kantengewichte des Teilgraphens.*
***Ziel**:* min.

Da das Problem des minimalen Steiner-Waldes die Probleme des kürzesten Weges und minimalen Steiner-Baumes enthält, haben wir die folgenden Sätze

Satz 3.35. *Kein Prioritäts-Algorithmus hat einen konstanten Approximationsfaktor.*

Beweis: Siehe Satz 3.18 und Satz 3.19. □

Satz 3.36. *Kein adaptiver Stapel-Algorithmus hat einen konstanten Approximationsfaktor.*

Beweis: Siehe Satz 3.20. □

Satz 3.37. *Kein adaptiver Stapel-Algorithmus hat einen Approximationsfaktor besser als* 2*.*

Beweis: Siehe Satz 3.33. □

Der Algorithmus PDg ist ein adaptiver Stapel-Algorithmus der Güte 2.

Kapitel 4

Das Netzwerk-Erstellungs-Spiel

4.1 Einführung

In diesem Kapitel betrachten wir das Netzwerk-Erstellungs-Spiel. Gegeben ist ein vollständiger ungerichteter Graph

$$K_n = (V, \{e = \{u, v\} \mid u, v \in V\})$$

und eine positive Zahl α. Wir nehmen $n := |V| \geq 3$ an. Jeder Knoten $v \in V$ repräsentiert einen Spieler und jeder Spieler wählt eine **Spieler-Strategie**, das heißt eine Kantenmenge S_v, die dieser Spieler **kauft**. Es bezeichne $S = (S_v)_{v \in V}$ die **Gesamt-Strategie** und $G(S) = (V, \bigcup_{v \in V} S_v)$ der Graph der gekauften Kanten. Gekaufte Kanten können von jedem Spieler benutzt werden und die Kosten der Gesamt-Strategie S für den Spielers v betragen

$$c_v(S) := \alpha |S_v| + \sum_{w \in V} d_{G(S)}(v, w),$$

wobei $d_{G(S)}$ die kanonische Distanzfunktion im Graphen $G(S)$ ist.

Definition 4.1 (Nash-Gleichgewicht)**.** *Die Gesamt-Strategie S ist ein **Nash-Gleichgewicht**, falls kein Spieler v seine Kosten senken kann, während die anderen Spieler-Strategien beibehalten werden. Das heißt für alle Spieler v und alle zugehörigen Spieler-Strategien S'_v von v und Gesamt-Strategien $S' = S'_v \times (S_w)_{w \in V, w \neq v}$ ist*

$$c_v(S') \geq c_v(S).$$

*Ein Graph G ist ein **Nash-Gleichgewicht**, fall ein Nash-Gleichgewicht S mit $G = G(S)$ existiert.*

Für eine gegebene Gesamtstrategie S werden wir die Kosten $c_v(S')$ einer alternativen Gesamtstrategie $S' = S'_v \times (S_w)_{w \in V, w \neq c}$ des Spielers v oft mit $c_v(S'_v)$ oder $c(S_v)$ abkürzen. Weiterhin werden wir den Graphen $G(S')$ der Strategieänderung des Spielers v mit $G(S'_v)$ abkürzen. Da die Kantenkosten nicht geteilt werden können, wird in einem Nash-Gleichgewicht jede Kante maximal von einem Spieler gekauft. Da Schlingen die Distanz nicht verringern, werden auch diese nicht gekauft.Jedes Nash-Gleichgewicht ist zusammenhängend, da $\alpha < \infty$ ist. Wir haben das folgende Lemma.

Lemma 4.2. *In einem Nash-Gleichgewicht S kauft jeder Spieler v nur inzidente Kanten.*

Beweis: Es sei die Gesamt-Strategie S ein Nash-Gleichgewicht und $G(S)$ der Graph der gekauften Kanten. Weiterhin sei $e = \{v, w\}$ eine Kante, die von einem Spieler u mit $u \notin e$ gekauft wird. Es bezeichne $G(S) \backslash e$ den Graphen $G(S)$ ohne die Kante e. Wir definieren die Menge der Knoten, die durch Löschen der Kante e eine höhere Distanz zu u haben:

$$S := \{x \mid d_{G(S) \backslash e}(u, x) > d_{G(S)}(u, x)\}.$$

Es ist $|S| \geq 1$, da S ein Nash-Gleichgewicht ist. Im Graphen $G(S) \backslash e$ existiert ein Weg von u nach v oder ein Weg von u nach w, da sonst die Kosten des Spielers selbst durch den Bau von e unendlich wäre. Im Fall $d_{G(S) \backslash e}(u, v) = d_{G(S) \backslash e}(u, w)$ könnte jeder kürzester Weg von u nach y, der in Richtung v nach w über die Kante e verläuft, durch den kürzeren, direkten Weg von u nach w verkürzt werden.

Es bleibt der Fall $d_{G(S)\setminus e}(u,v) \leq d_{G(S)\setminus e}(u,w)$. In diesem Fall könnte Spieler u seine Kosten senken, indem er statt der Kante e die Kante $\{u,w\}$ wählt. Für alle $x \in S$ verkürzt sich dadurch die Distanz zu u um $d(u,v) > 0$. Für alle anderen Knoten bleibt die Distanz gleich oder verringert sich. □

Wir können uns bei Betrachtung der Nash-Gleichgewichte also auf den Fall beschränken, dass nur inzidente Kanten gekauft werden. Dieses Spiel wurde von den Autoren Fabrikant,Luthra, Maneva, Papdimitriou und Shenker ([FLM+03]) betrachtet. Weiterhin wurde der Artikel [AEED+06] für dieses Kapitel verwendet. Falls ein Spieler v die Kante $\{v,w\}$ kauft, werden wir es in den Bildern oft durch eine gerichtete Kante (v,w) andeuten und bei der Spieler-Strategie häufig nur die Endpunkte angeben. Gerichtete Kanten definieren nur den Käufer. Sie können auch weiterhin von jedem Spieler in jeder Richtung benutzt werden.

Definition 4.3 (Globale Kosten)**.** *Es sei $G = (V,E)$ ein Teilgraph von K_n. Die **globalen Kosten** des Graphen G betragen*

$$c(G) = \alpha|E| + \sum_{v,w} d_G(v,w).$$

Es bezeichne OPT *den Graphen mit den geringsten globalen Kosten* opt*.*

Wir werden die Kosten der Nash-Gleichgewichte mit den optimalen Kosten vergleichen.

Definition 4.4 (Preis der Stabilität)**.** *Der **Preis der Stabilität** ist*

$$\text{PoS} := \min_{G \text{ ist ein Nash-Gleichgewicht}} \frac{c(G)}{\text{opt}}.$$

Definition 4.5 (Preis der Anarchy)**.** *Der **Preis der Anarchy** ist*

$$\text{PoA} := \max_{G \text{ ist ein Nash-Gleichgewicht}} \frac{c(G)}{\text{opt}}.$$

Für die globalen Kosten haben wir das folgende Lemma.

Lemma 4.6. *Es sei $G = (V,E)$ ein Graph. Es ist*

$$c(G) \geq \alpha|E| + 2|E| + 4\left(\binom{n}{2} - |E|\right) = 2n(n-1) + (\alpha-2)|E|.$$

Es gilt genau dann Gleichheit, wenn der Durchmesser von G maximal 2 ist.

Beweis: Die Ungleichung folgt aus der Tatsache, dass zwei nicht adjazente Knoten mindestens die Distanz 2 haben. □

Lemma 4.7.

1. *Im Netzwerk-Erstellungs-Spiel ist* OPT *der vollständige Graph K_n für $\alpha < 2$.*
2. *Jeder Graph mit Durchmesser kleiner oder gleich 2 ist optimal für $\alpha = 2$.*
3. *Für $\alpha > 2$ ist* OPT *der **Stern**, das heißt es existiert ein Knoten u, der zu allen anderen Knoten adjazent ist und die anderen Knoten sind nur zu u adjazent.*

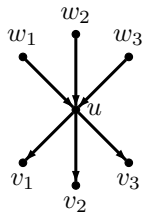
Beweis: Im Fall $\alpha \leq 2$ wird die untere Schranke von Lemma 4.6 für viele Kanten minimal. $|E|$ ist für K_n maximal und K_n nimmt auch die Schranke von Lemma 4.6 an.
Im Fall $\alpha \geq 2$ wird die untere Schranke von Lemma 4.6 für wenige Kanten minimal. Für zusammenhängende Graphen ist $|E|$ für Bäume minimal. Der Stern hat einen Durchmesser von 2 und nimmt somit die Schranke von Lemma 4.6 an. □

Der Stern und der vollständige Graph sind auch Nash-Gleichgewichte.

Beispiel 4.8. *Der Stern ist für $\alpha \geq 1$ ein Nash-Gleichgewicht mit den Kosten*

$$c(\textit{Stern}) = (n-1)(\alpha + 2n - 2).$$

Beweis: Es sei S eine Gesamt-Strategie, die den Stern erzeugt. Weiterhin sei u der Knoten, der zu allen anderen Knoten im Stern adjazent ist.



Falls der Spieler u eine Kante zu einem Knoten v löschen würde, würde die Distanz auf ∞ steigen. Falls ein Spieler w zu einer Spieler-Strategie $S'_w \neq \{\}$ wechselt, die nicht den Knoten u enthält, könnte der Spieler, bei gleichen oder geringeren Kosten, ein Knoten $x \in S'_w$ durch den Knoten u ersetzen. Falls ein Spieler w zu einer Spieler-Strategie S'_w wechselt, die unter anderem den Knoten u und x enthält, bringt der Bau zum Knoten x nur eine Distanzverbesserung von 1. Dies wäre nur im Fall $\alpha = 1$ sinnvoll. Die Kosten des Sterns betragen

$$c(\text{Stern}) = \alpha(n-1) + 1(n-1) + (n-1)(1+2(n-2)) = (n-1)(\alpha + 2n - 2).$$

□

Beispiel 4.9. *Der vollständige Graph K_n ist für $\alpha \leq 1$ ein Nash-Gleichgewicht mit den Kosten*

$$c(K_n) = \frac{\alpha+2}{2} n(n-1).$$

Beweis: Es sei $(S_v)_{v \in V}$ eine Gesamtstrategie, die den Graphen K_n erzeugt und u ein Spieler. Jede andere (sinnvolle) Spieler-Strategie S'_u von u hat weniger Kanten als S_u. Beim Löschen einer Kante würde nur die Distanz des inzidenten Knotens um 1 wachsen. Es ist $-\alpha + 1 \geq 0$. (Falls der Spieler alle Kanten löscht könnte die Distanz sogar auf ∞ steigen.) Die Kosten von K_n betragen

$$c(K_n) = \alpha \binom{n}{2} + n(n-1) = \frac{\alpha+2}{2} n(n-1).$$

□

Lemma 4.10. *Es sei G ein Nash-Gleichgewicht und v und w zwei Knoten, die den Abstand d haben. Weiterhin sei $k \in \mathbb{N}$.*

1. *Im Fall $d \geq 2k$ ist $\alpha \geq k^2$ und*
2. *im Fall $d \geq 2k-1$ ist $\alpha \geq k^2 - k$.*

Beweis:

1. Im Fall $d \geq 2k$ könnte Spieler v eine Kante zu einem Knoten x mit Distanz $2k$ bauen. Die Distanzkosten würden mindestens um

   $$2k - 1 + (2k-1) - 2 + (2k-2) - 3 + \cdots + (k+1) - k = k^2$$

   sinken. Da G ein Nash-Gleichgewicht ist, folgt $\alpha - k^2 \geq 0$.

2. Im Fall $d \geq 2k-1$ könnte Spieler v eine Kante zu einem Knoten x mit Distanz $2k-1$ bauen. Die Distanzkosten würden mindestens um

   $$(2k-1) - 1 + (2k-2) - 2 + (2k-3) - 3 + \cdots + (k+1) - (k-1) = k^2 - k$$

   sinken. Da G ein Nash-Gleichgewicht ist, folgt $\alpha - k^2 + k \geq 0$.

□

Korollar 4.11. *Es sei G ein Nash-Gleichgewicht und D der Durchmesser von H. Es ist*

$$D \leq \sqrt{4\alpha - 1}$$

Beweis: Wir betrachten zwei Fälle.

1. Im Fall $D = 2k$ für ein $k \in \mathbb{N}$ ist nach Lemma 4.10
$$D = 2k \leq 2\sqrt{\alpha} \leq \sqrt{4\alpha + 1}.$$
2. Im Fall $D = 2k - 1$ für ein $k \in \mathbb{N}$ ist $k \leq \frac{\sqrt{4\alpha+1}-1}{2}$ nach Lemma 4.10 und somit folgt
$$D = 2k - 1 \leq \sqrt{4\alpha + 1}.$$

□

4.2 Der Fall $\alpha < 2$

Satz 4.12. *Für* $0 \leq \alpha < 1$ *ist*
$$\text{PoS} = \text{PoA} = 1.$$

Beweis: Jedes Nash-Gleichgewicht hat nach Lemma 4.10 den Durchmesser 1 . Weiterhin ist der vollständige Graph K_n auch ein Nash-Gleichgewicht für $\alpha \leq 1$. Nach Lemma 4.7 ist OPT auch der vollständige Graph. Es folgt $\text{PoS} = \text{PoA} = 1$. □

Satz 4.13. *Für* $\alpha = 1$ *ist*
$$\text{PoS} = 1 \text{ und } \text{PoA} < \frac{4}{3}$$

Beweis: Der vollständige Graph K_n ist wieder ein Nash-Gleichgewicht und auch global optimal. Somit ist $\text{PoS} = 1$. Jedes Nash-Gleichgewicht hat nach Lemma 4.10 maximal einen Durchmesser von 2 und nimmt somit die Schranke von Lemma 4.6 an. Diese Schranke wird für den Stern maximal. Der Stern ist nach Beispiel 4.8 ein Nash-Gleichgewicht. Es folgt
$$\text{PoA} = \frac{c(\text{Stern})}{c(K_n)} = \frac{(n-1)(2n-1)}{\frac{3}{2}n(n-1)} = \frac{4}{3} - \frac{2}{3n} < \frac{4}{3}.$$

□

Satz 4.14. *Für* $1 < \alpha < 2$ *ist*
$$\text{PoA} < \frac{4}{\alpha + 2} < \frac{4}{3}$$

Beweis: OPT ist wieder der vollständige Graph K_n. Jedes Nash-Gleichgewicht hat nach Lemma 4.10 maximal den Durchmesser 2. Es folgt wie im obigen Satz
$$\text{PoA} = \frac{c(\text{Stern})}{\text{opt}} = \frac{\alpha + 2n - 2}{\frac{\alpha+2}{2}n} = \frac{2(\alpha-2)+4n}{(\alpha+2)n} < \frac{4}{\alpha+2}.$$

□

Da der vollständige Graph für $\alpha > 1$ kein Nash-Gleichgewicht ist, ist der Preis der Stabilität ist in diesem Fall etwas schwieriger. Wir werden PoS für kleine n angeben.

Lemma 4.15. *Im Fall* $1 < \alpha < 2$ *und* $n = 3$ *ist*
$$\text{PoS} = \text{PoA} = \frac{2\alpha + 8}{3\alpha + 6} < \frac{10}{9}$$

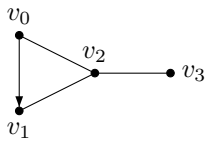
Beweis: Für $n = 3$ gibt es nur zwei zusammenhängende Graphen. Den Stern und den vollständigen Graphen. Nur der Stern ist ein Nash-Gleichgewicht. Es ist
$$\text{PoS} = \frac{c(\text{Stern})}{c(K_3)} = \frac{2(\alpha+4)}{\frac{\alpha+2}{2}6} = \frac{2\alpha+8}{3\alpha+6} < \frac{10}{9}.$$

□

Lemma 4.16. *Im Fall* $1 < \alpha < 2$ *und* $n = 4$ *ist*

$$\text{PoS} = \frac{2\alpha + 8}{3\alpha + 6} < \frac{10}{9} \text{ und } \text{PoA} = \frac{\alpha + 6}{2\alpha + 4} < \frac{7}{6}$$

Beweis: Es sei G ein Nash-Gleichgewicht. Wir zeigen, dass G dreiecksfrei ist. Es sei v_0, v_1, v_2 ein Dreieck und der vierte Punkt v_3 sei ohne Einschränkung der Allgemeinheit mit v_2 verbunden.



Es können noch weitere Kanten existieren. Ohne Einschränkung der Allgemeinheit baut v_0 die Kante $\{v_0, v_1\}$. Diese Kante kann gelöscht werden und nur die Distanz zu v_1 würde um 1 wachsen. Somit ist G kein Nash-Gleichgewicht. Das einzige mögliche Nash-Gleichgewicht, welches kein Baum ist, ist das Viereck. Im nächsten Beispiel wird gezeigt, dass es ein Nash-Gleichgewicht ist. Es folgt

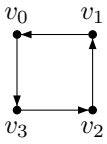
$$\text{PoS} = \frac{4\alpha + 16}{6\alpha + 12} = \frac{2\alpha + 8}{3\alpha + 6} < \frac{10}{9}.$$

Der Stern ist das teuerste Nash-Gleichgewicht und für den Preis der Anarchy folgt

$$\text{PoA} = \frac{c(\text{Stern})}{c(K_4)} = \frac{3\alpha + 18}{6\alpha + 12} = \frac{\alpha + 6}{2\alpha + 4} < \frac{7}{6}.$$

□

Beispiel 4.17. *Das Viereck* C_4 *mit der folgenden Orientierung*

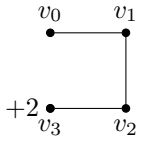


ist ein Nash-Gleichgewicht für $1 \leq \alpha \leq 2$ *mit Kosten*

$$c(C_4) = 4\alpha + 16.$$

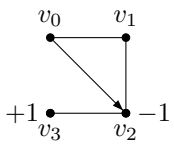
Beweis: Da das Viereck mit obiger Orientierung knotenautomorph ist, genügt es einen Spieler zu betrachten. Wir schauen uns alle Spieler-Strategien S des Spielers v_0 und geben die Kosten-Differenz $\delta := c(S) - c(S_{v_0})$ der Strategie S mit der Vierecks-Strategie $S_{v_0} = \{v_3\}$ an.

1. Der Spieler v_0 baut keine Kante.



   Die Distanz zum Knoten v_3 wächst um 2. Es ist $\delta = -\alpha + 2 \geq 0$.

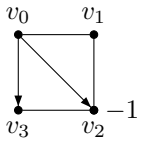
2. Der Spieler baut eine Kante zu v_2.



Die Distanz zum Knoten v_3 wächst um 1 und die Distanz zum Knoten v_2 sinkt um 1. Es ist $\delta = 0$.

3. Der Spieler v_0 baut eine Kante zu v_3. Dies ist die Vierecks-Strategie und somit ist $\delta = 0$.

4. Der Spieler baut zwei Kanten zu den beiden Knoten v_2 und v_3.



Die Distanz zum Knoten v_2 sinkt dadurch um 1 und es ist $\delta = \alpha - 1 \geq 0$.

Es folgt die Behauptung. □

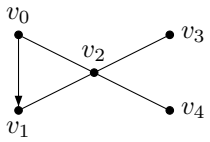
In Orientierungen, bei denen ein Spieler zwei Kanten baut, ist das Viereck nur für $\alpha = 1$ ein Nash-Gleichgewicht.

Lemma 4.18. *Im Fall* $1 < \alpha < 2$ *und* $n = 5$ *ist*

$$\text{PoS} = \frac{\alpha + 6}{2\alpha + 4} < \frac{7}{6} \text{ und } \frac{2\alpha + 16}{5\alpha + 10} < \frac{6}{5}$$

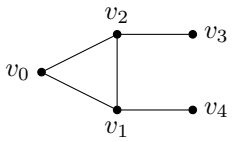
Beweis: Wir zeigen, dass ein Nash-Gleichgewicht, welches kein Baum ist, dreiecks- und viereckfrei ist. Zuerst zeigen wir die Dreiecksfreiheit. Es sei G ein Nash-Gleichgewicht und (v_0, v_1, v_2) ein Dreieck. Es existieren noch zwei weitere Punkte v_3, v_4.

1. Wir betrachten den Fall, dass die beiden Punkte v_3 und v_4 mit einer Ecke des Dreiecks verbunden sind. Ohne Einschränkung der Allgemeinheit ist es v_2.



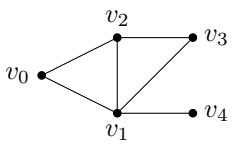
Es können noch weitere Kanten existieren. Wir nehmen an, dass die Kante $\{v_0, v_1\}$ von v_0 gebaut wird. Dies ist kein Nash-Gleichgewicht. Der Spieler v_0 könnte die Kante $\{v_0, v_1\}$ löschen und nur die Distanz zu v_1 würde sich um 1 erhöhen.

2. Im zweiten Fall sind v_3 und v_4 mit zwei verschiedenen Knoten verbunden. Ohne Einschränkung seiden dies die Knoten v_1 und v_2.



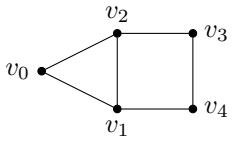
Die Knoten v_3 und v_4 haben im obigen Schaubild die Distanz 3. Daher sind die beiden Knoten über einen Knoten des Dreiecks verbunden oder es existiert eine Kante, die den Weg (v_3, v_2, v_1, v_4) abkürzt. Die erste Möglichkeit haben wir im ersten Fall betrachtet und betrachten nun zwei Fälle, bei denen der Weg abgekürzt wird.

(a) Im ersten Unterfall ist v_1 mit v_3 verbunden.



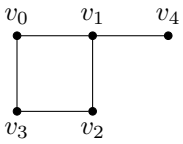
Es können noch weitere Kanten existieren. Nun könnte der Käufer der Kante (v_0, v_2) diese Löschen und würde dadurch nur einen Distanzverlust von eins haben. Also ist G kein Nash-Gleichgewicht.

(b) Im zweiten Unterfall ist v_3 direkt mit v_4 verbunden.



Es können noch weitere Kanten existieren. Der Käufer der Kante $\{v_1, v_2\}$ könnt diese löschen und hätte nur einen Distanzverlust von eins und wir haben wieder kein Nash-Gleichgewicht.

Damit haben wir die Dreiecksfreiheit bewiesen. Wir zeigen nun die Viereckfreiheit. Es sei nun G ein Nash-Gleichgewicht mit dem Viereck (v_0, v_1, v_2, v_3) und der fünfte Knoten v_4 sei ohne Einschränkung der Allgemeinheit mit v_1 verbunden.



□

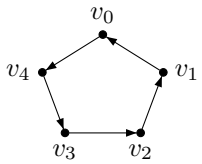
Der Abstand von v_3 und v_4 beträgt 3. Also müssen Knoten existieren, die diesen Weg abkürzen. Jede Abkürzung würde ein Dreieck erzeugen und wir wären in einem der obigen Fälle. Das Fünfeck ist somit das einzige mögliche Nash-Gleichgewicht, welches kein Baum ist. Im folgenden Beispiel wird gezeigt, dass es ein Nash-Gleichgewicht ist. Wir haben

$$\text{PoS} = \frac{5\alpha + 30}{10\alpha + 20} = \frac{\alpha + 6}{2\alpha + 4} < \frac{7}{6}$$

Der Stern ist wieder das teuerste Nash-Gleichgewicht und es folgt

$$\text{PoA} = \frac{4\alpha + 32}{10\alpha + 20} = \frac{2\alpha + 16}{5\alpha + 10} < \frac{18}{15} = \frac{6}{5}.$$

Beispiel 4.19. *Das Fünfeck C_5 mit der folgenden Orientierung*

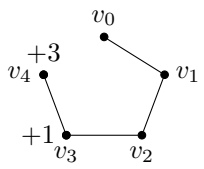


ist ein Nash-Gleichgewicht für $1 \leq \alpha \leq 4$ mit den Kosten

$$c(C_5) = 5\alpha + 30$$

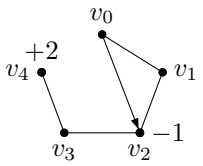
Beweis: Da das Fünfeck mit obiger Orientierung knotenautomorph ist, genügt es einen Spieler zu betrachten. Wir schauen uns alle Spieler-Strategien S des Spielers v_0 und geben die Kosten-Differenz $\delta := c(S) - c(S_{v_0})$ der Strategie S mit der Fünfecks-Strategie $S_{v_0} = \{v_4\}$ an.

1. Der Spieler v_0 baut keine Kante.



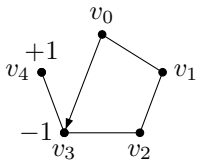
Die Distanz zum Knoten v_4 erhöht sich um 3 und die Distanz zum Knotens v_3 erhöht sich um 1. Es ist $\delta = -\alpha + 4 \geq 0$.

2. Der Spieler baut eine Kante zu dem Knoten v_2.



In diesem Fall steigt die Distanz zu v_4 um 2 und die Distanz zu v_2 sinkt um 1. Es ist $\delta = 2 - 1 = 1 > 0$.

3. Der Spieler baut eine Kante zu dem Knoten v_3.



Die Distanz zu v_4 erhöht sich um 1 und die Distanz zu v_3 verringert sich um 1. Es ist $\delta = 0$.

4. Der Spieler baut eine Kante zu dem Knoten v_4. Dies ist die Fünfecks-Strategie und somit ist $\delta = 0$.

5. Der Spieler baut zwei oder mehr Kanten. Es sei $l \geq 2$ die Anzahl der Kanten, die der Spieler baut. Die bestmögliche Strategie hat zu l vielen Knoten die Distanz 1 und zu den anderen Knoten die Distanz 2. Da es schon Strategien mit nur eine Kante gibt, bei denen alle Distanzen kleiner oder gleich zwei sind, existieren auch solche Strategien. Es ist

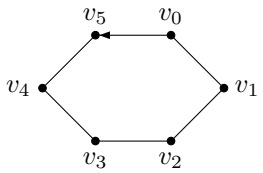
$$\delta = (l-1)\alpha - (l-1) = (l-1)(\alpha - 1) \geq 0.$$

□

In Orientierungen, bei denen ein Spieler zwei Kanten baut, ist das Fünfeck nur für $1 \leq \alpha \leq 2$ ein Nash-Gleichgewicht.

Beispiel 4.20. *Das Sechseck C_6 ist kein Nash-Gleichgewicht.*

Beweis: Es sei v_0 ein Spieler, der mindestens eine Kante baut. Es sei (v_0, v_5) diese Kante.



Der Spieler v_0 könnte zum Knoten v_4 wechseln. Dadurch würde sich die Distanz zu den Knoten v_4 und v_3 um ins verringern, während sich die Distanz zu dem Knoten v_5 um eins erhöht. □

Wir werden später noch weitere Nash-Gleichgewichte angeben.

4.3 Der Fall $\alpha \geq 2$

In diesem Abschnitt betrachten wir den schwierigeren Fall $\alpha \geq 2$. OPT ist ein Stern und dies ist auch ein Nash-Gleichgewicht. Somit ist PoS = 1. In [FLM+03] wurde gezeigt, dass für $\alpha \geq n^2$ jedes Nash-Gleichgewicht ein Baum ist. Weiterhin wurde der folgende Satz gezeigt.

Satz 4.21. *Der Preis der Anarchy für Bäume ist kleiner gleich fünf. Das heißt: Es ist*

$$\frac{c(G)}{\text{opt}} \leq 5$$

für alle Nash-Gleichgewichte G, die Bäume sind.

Im gleichen Artikel ist der folgende Satz.

Satz 4.22. *Für $2 \leq \alpha \leq n^2$ ist der Preis der Anarchy*

$$\text{PoA} = O(\sqrt{\alpha}).$$

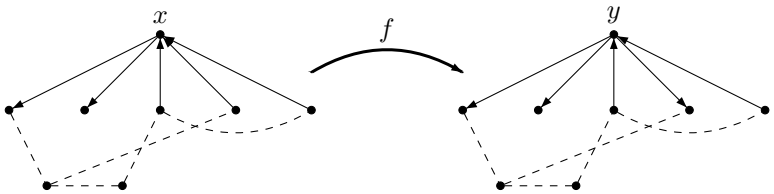
Diese Schranke wurde in [AEED+06] verbessert zu:

Satz 4.23. *Es sei $\alpha \geq 2$. Der Preis der Anarchy ist*

$$\text{PoA} = O\left(1 + \left(\min\{\frac{\alpha^2}{n}, \frac{n^2}{\alpha}\right)^{\frac{1}{3}}\right)$$

[FLM+03] haben nur Nash-Gleichgewichte gefunden, die Bäume sind, mit Ausnahme des Petersen-Graphen. Der Beweis, dass der Peterson-Graph eine Nash-Gleichgewicht ist, wurde in [FLM+03] nicht geführt. Dies holen wir nach. Dafür brauchen wir das folgende nützliche Lemma.

Lemma 4.24. *Es seien $(S_u)_{u \in V}$ und $(T_v)_{v \in V}$ zwei Gesamt-Strategien mit den zugehörigen ungerichteten Kauf-Graphen $G = G(S) = (V, E_G)$ und $H = G(T) = (V, E_H)$. Weiterhin sei $f : (V, E_G) \to (V, E_H)$ ein Isomorphismus. Der Spieler $y = f(x) \in V$ habe eine optimale Strategie. Für jede Kante e, die der Spieler x baut, baue der Spieler y die zugehörige Kante $f(e)$. (Der Spieler y könnte noch weitere Kanten bauen.) Dann hat auch der Spieler x eine optimale Strategie.*



Beweis: Es bezeichne F die Kanten e, die der Spieler y in der Strategie T_y baut und deren zugehörigen Kanten $f^{-1}(e)$ in der Strategie S_x nicht gebaut werden. Es ist

$$T_y = F \dot{\cup} f(S_x).$$

Es sei nun S eine weitere Strategie von Spieler x. Dann ist $T := F \dot{\cup} f(S)$ eine Strategie von Spieler y und aus der Optimalität von T_y folgt

$$c(T_y) = \alpha|T_y| + d_H(y) \leq c(T) = \alpha|T| + d_{G(T)}(y).$$

Dies ist genau dann der Fall, falls

$$\alpha(|T_y| - |T|) \leq d_{G(T)}(y) - d_H(y)$$

ist. Es ist $|T_y| - |T| = (|F| + |S_y|) - (|F| + |S|) = |S_y| - |S|$. Aus der Definition von F und der Isomorphie von f, folgt

$$d_G(x) = d_H(y) \text{ und } d_{G(S)}(x) = d_{G(T)}(y).$$

Somit ist

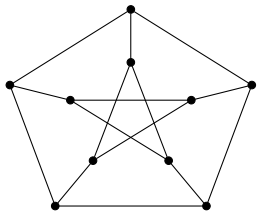
$$\alpha(|S_x| - |S|) \leq d_{G(S)}(x) - d_G(x).$$

Es folgt

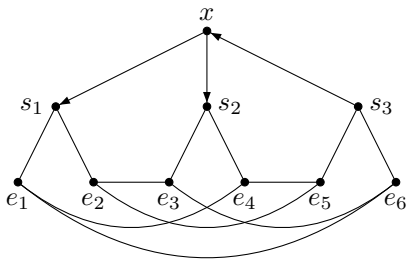
$$c(S_x) = \alpha|S_x| + d_G(x) \leq \alpha|S| + d_{G(S)}(x) = c(S)$$

und die Behauptung. □

Beispiel 4.25. *Der Peterson-Graph ist für* $1 \leq \alpha \leq 4$ *ein Nash-Gleichgewicht.*



Beweis: Man sieht leicht, dass Orientierungen existieren, bei denen jeder Spieler maximal zwei Kanten kauft. Da der Petersen-Graph knoten-automorph ist, können wir Lemma 4.24 anwenden und ohne Einschränkung einen Spieler x betrachten, der genau zwei Kanten baut. Die Strategie S_x des Spielers x sieht folgendermaßen aus:

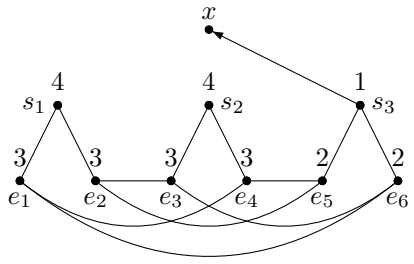


Die Knoten s_i bezeichnen die Söhne von x und die Knoten e_i die Enkel der Strategie S_x. Die Kosten betragen

$$c(S_x) = 2\alpha + 15.$$

Wir betrachten jetzt alle möglichen Spieler-Strategien S von Spieler x und tragen die Distanzen im Schaubild ein.

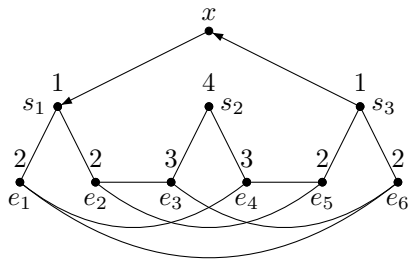
1. Der Spieler x baut keine Kante:



   Die Kosten betragen

$$c(S) = 25 \geq 2\alpha + 15 \Leftrightarrow \alpha \leq 5.$$

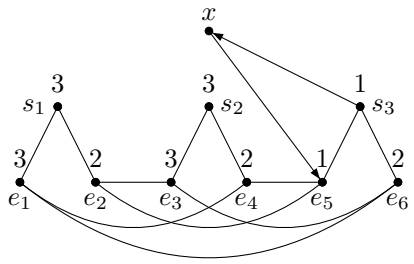
2. Der Spieler baut eine Kante zu s_1 oder s_2. Ohne Einschränkung der Allgemeinheit zu $s1$.



Die Kosten betragen

$$c(S) = \alpha + 20 \geq 2\alpha + 15 \Leftrightarrow \alpha \leq 5.$$

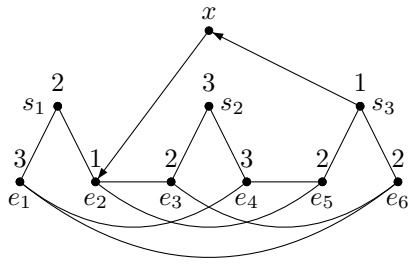
3. Der Spieler baut eine Kante zu einem Enkel, dessen Vater s_3 ist. Ohne Einschränkung der Allgemeinheit sei es e_5.



Die Kosten betragen

$$c(S) = \alpha + 20 \geq 2\alpha + 15 \Leftrightarrow \alpha \leq 5.$$

4. Der Spieler baut eine Kante zu einem Enkel, dessen Vater nicht s_3 ist. Ohne Einschränkung der Allgemeinheit sei es e_2.



Die Kosten betragen

$$c(S) = \alpha + 19 \geq 2\alpha + 15 \Leftrightarrow \alpha \leq 4.$$

5. Der Spieler baut zwei Kanten. Die beste Strategie mit zwei Kanten hat (mit der schon vorhandenen Kante) drei Knoten mit Distanz 1 und die restlichen Knoten haben Distanz 2. Die Strategie S_x ist solch eine Strategie.

6. Jede weitere Kante über zwei hinaus könnte nur für einen Knoten die Distanz von 2 auf 1 verringern. Die Kosten würden $\alpha - 1 \geq 0$ pro Kante betragen.

□

Es folgt die Behauptung.

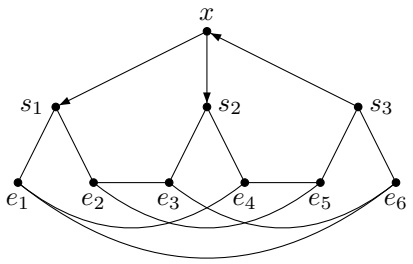
Definition 4.26 (schwaches (starkes) Nash-Gleichgewicht)**.** *Ein Nash-Gleichgewicht S heißt **schwach**, falls mindestens ein Spieler ohne erhöhte Kosten seine Strategie wechseln kann, während die anderen Spieler ihre Strategie beibehalten. Ein **starkes** Nash-Gleichgewicht ist ein nicht schwaches Nash-Gleichgewicht.*

Bis jetzt waren alle betrachteten Nash-Gleichgewichte, die keine Bäume waren, schwach. Es gilt sogar noch mehr.

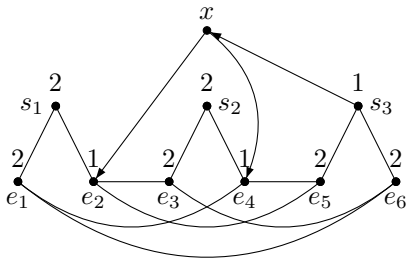
Definition 4.27 (transient)**.** *Ein (schwaches) Nash-Gleichgewicht heißt* ***transient****, falls es eine Sequenz von Spieler-Strategie-Wechsel existiert, die jeweils die Spielerkosten nicht verändern und zu einem Nicht-Nash-Gleichgewicht führen.*

Beispiel 4.28. *Der Petersen Graph ist transient.*

Beweis: Es gibt einen Spieler x, der zwei Kanten baut. Wie in Beispiel 4.25 sieht die Strategie des Spielers folgendermaßen aus



Der Spieler könnte stattdessen zwei Kanten zu den Enkeln e_2 und e_4 bauen.



Die Kosten würden auch

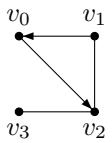
$$c(S) = 2\alpha + 15 = c(S_v)$$

betragen. Allerdings hat in dieser Situation der Spieler s_3 eine Kante zu dem Knoten x, welcher mit e_2 und e_4 adjazent ist. Der Spieler s_3 hat auch über den Knoten e_5 die Distanz 2 zu e_2 und e_4. Die Kante (s_3, x) bringt dem Spieler x also nur einen Distanz-Gewinn von 1 und könnte besser in einen anderen Knoten, wie zum Beispiel den Knoten s_1 mit Distanz 3, investiert werden. □

Wir kennen noch weitere transiente Beispiele.

Beispiel 4.29. *Das Viereck ist für* $1 < \alpha \leq 2$ *transient.*

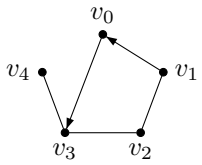
Beweis: Im Beweis von Beispiel 4.17 haben wir gesehen, dass der Spieler v_0 zu gleichen Kosten eine Kante zu v_2 bauen konnte.



Nun kann aber der Spieler v_1 seine Kante zu v_0 löschen und würde $1 - \alpha > 0$ sparen. □

Beispiel 4.30. *Das Fünfeck ist für* $1 \leq \alpha \leq 4$ *transient.*

Beweis: Im Beweis von Beispiel 4.19 haben wir gesehen, dass der Spieler v_0 zu gleichen Kosten eine Kante zu v_3 bauen konnte.



Nun kann der Spieler v_1 zum Knoten v_3 wechseln und die Distanz von v_4 und v_3 würde sich jeweils um eins verringern, während sich die Distanz zu v_0 um eins erhöht. □

[FLM+03] haben nur den Petersen-Graph gefunden, der ein Nash-Gleichgewicht und kein Baum ist. Wir haben bisher nur für kleine $n \leq 10$ weitere Nash-Gleichgewichte angegeben, die keine Bäume sind. In den nächsten Beispielen werden wir dies ändern. Wir benötigen zunächst noch eine Definition und ein Lemma.

Definition 4.31. *Es sei $G = (V, E)$ ein gerichteter Graph und $m \in \mathbb{N}$ gegeben. Es sei $V(m)$ die Knotenmenge V, wobei zu jedem Knoten $v \in V$ noch m weitere neue Knoten $v^1, \ldots, v^m$ existieren. Weiterhin sei $E(m)$ die Kantenmenge E erweitert mit den Kanten (v, v^i) für alle $i = 1, \ldots, m$ und Knoten $v \in V$. Wir bezeichnen den erweiterten Graph mit $G(m) := (V(m), E(m))$.*

Lemma 4.32. *Es sei G für alle $\alpha \in [a, b]$ ein Nash-Gleichgewicht. Weiterhin sei $m \in \mathbb{N}$ und $a(m+1) > 1$. Alle Knoten $v \in V$ haben eine optimale Strategie im Graphen $G(m)$ für $\alpha' \in [a(m+1), b(m+1)]$. Falls G transient ist (und $G(m)$ ein Nash-Gleichgewicht ist), dann ist auch $G(m)$ transient.*

Beweis: Es sei $u \in V$ ein Spieler. Wir bezeichnen die Spieler-Strategie von u in G mit S und in $G(m)$ mit S_m. Es ist $S_m = S \cup \{u^1, \ldots, u^m\}$. Der Spieler u muss die m Kanten zu den Knoten u^i bauen, da sonst die Distanz ∞ wäre.
Zunächst zeigen wir, dass nur Strategien zu Knoten $v \in V$ (mit Ausnahme der u^i) sinnvoll sind. Es sei S'_m eine Strategie von u, die einen Knoten $v^1 \notin V$ enthält. Der Spieler u könnte den Knoten v^1 durch den Vaterknoten v ersetzen. Die Distanz zu v^1 würde um 1 steigen, während die Distanz zu v und möglicherweise zu den anderen v^i um 1 fällt. Diese Ersetzung wäre nicht möglich, wenn der Knoten v schon selber in S' ist. In diesem Fall ist aber $d(u, v) = 1$ und das Löschen der Kante zu v^1 würde dem Spieler u nur einen Distanzverlust von 1 bringen. Da aber $a(m + 1) > 1$ ist, würde der Spieler diese Kante nicht bauen.
Es sei nun S'_m eine Strategie von Spieler u, die nur Kanten zu Knoten $v \in V$ (und den Knoten u^i) baut. Weiterhin sei S' die Strategie S'_m ohne die Knoten u^i. Die Kosten der Spieler-Strategie betragen

$$\begin{aligned} c_u(S'_m) &= \alpha'|S'_m| + \sum_{v \in V(m)} d_{G(S'_m)}(u, v) = \alpha'|S'| + \alpha' m + \sum_{v \in V} \left(d_{G(S')}(u, v) + m(d_{G(S')}(u, v) + 1) \right) \\ &= \alpha'|S'| + \sum_{v \in V} \left((m+1) d_{G(S')}(v, u) \right) + m(\alpha' + |V|) \end{aligned}$$

Da G ein Nash-Gleichgewicht ist, folgt mit $\alpha := \frac{\alpha'}{m+1} \in [a, b]$

$$c_u(S') = \alpha|S'| + \sum_{v \in V} d_{G(S')}(u, v) \geq c_u(S) = \alpha|S| + \sum_{v \in V} d_G(u, v).$$

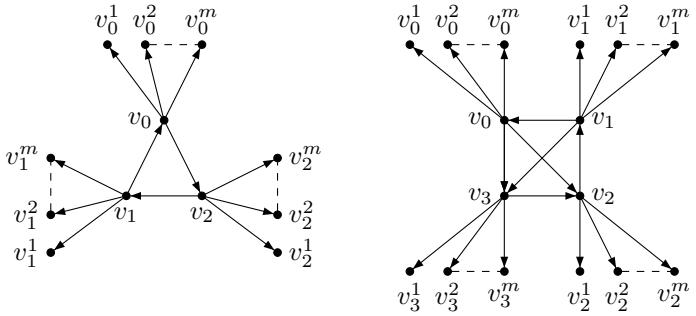
Somit ist

$$\begin{aligned} c_u(S'_m) &= (m+1) \left(\alpha|S'| + \sum_{v \in V} d_{G(S')}(u, v) \right) + m(\alpha' + |V|) \\ &\geq (m+1) \left(\alpha|S| + \sum_{v \in V} d_G(u, v) \right) + m(\alpha' + |V|) \\ &= \alpha'(|S| + m) + \sum_{v \in V} (d_G(u, v) + m(d_G(u, v) + 1)) = \alpha'|S_m| + \sum_{v \in V(m)} d_{G(m)}(u, v) \\ &= c_u(S_m). \end{aligned}$$

Die Strategie S_m des Spielers u ist also optimal.
Falls G transient ist, können die gleichen Strategiewechsel in $G(m)$ gemacht werden und $G(m)$ ist somit auch transient. □

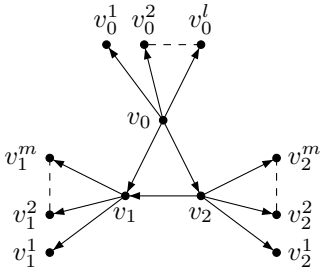
Um zu zeigen, dass $G(m)$ ein Nash-Gleichgewicht ist, müssen wir nur die Strategien der Punkte $v^i \notin V$ prüfen. Wie im Beweis des Lemmas genügt es dabei Strategien zu betrachten, die nur Knoten des Originalgraphen G enthalten. Wir werden einige Beispiele geben.

Beispiel 4.33. *Es sei $m \in \mathbb{N}$. Der Graph $K_n(m)$ ist für $\alpha = m+1$ ein transientes Nash-Gleichgewicht. $K_3(m)$ und $K_4(m)$ haben die folgende Gestalt:*



Beweis: Nach Lemma 4.32 und Beispiel 4.9 haben alle Spieler $v \in K_n$ eine optimale Strategie für $\alpha \leq m+1$. Wir betrachten einen Spieler $v^1 \notin K_n$. Dieser Spieler hat eine eingehende Kante von v. Es genügt Strategien $S \subseteq K_n$ zu betrachten. Der Spieler könnte zu $w \in K_n$ eine Kante bauen. Die Distanz zu w und allen w^i, $i = 1, \ldots, m$ sinkt um 1. Es ist $\alpha - (m+1) \geq 0$. Für l Kanten folgt analog $l\alpha - l(m+1) = l(\alpha - (m+1)) \geq 0$.
Es bleibt die Transitivität zu zeigen. Es sei $x \in K_n$ ein Knoten, der eine Kante zu $y \in K_n$ baut. Der Knoten $y^1 \notin K_n$ könnte mit gleichen Kosten eine Kante zu x bauen. Nun hätte der Spieler x bei der Kantenlöschung die Kosten $-\alpha + (m) = -1 < 0$. □

Beispiel 4.34. *Es seien $l, m \in \mathbb{N}_0$. Der folgende Graph ist für $l \leq m$ und $\alpha = m+1$ ein Nash-Gleichgewicht. Für $m \in \mathbb{N}$ ist er transient.*



Beweis: Wir betrachten zuerst den Spieler v_0. Der Spieler v_0 muss mindestens eine Kante bauen, da sonst die Distanzkosten ∞ wäre. Wie im Beweis von Lemma 4.32 sind die Knoten v_i sinnvoller als die Knoten v_i^j.

1. Der Spieler baut eine Kante. Ohne Einschränkung der Allgemeinheit baut er zu v_1. Die Distanz zu v_2 und allen v_2^i wächst um 1 Die Kostendifferenz beträgt

$$-\alpha + (m+1) = 0.$$

2. Falls der Spieler v_0 zwei Kanten baut, ist $\{v_0, v_1\}$ die beste Strategie. Dies ist die Nash-Strategie.

3. Da in der Nash-Strategie alle Knoten maximal die Distanz 2 haben, würde jede Strategie mit mehr Kanten höchstens die Distanz eines Knotens von 2 auf 1 senken. Die Kosten jeder zusätzlichen Kante wären $\alpha - 1 \geq 0$.

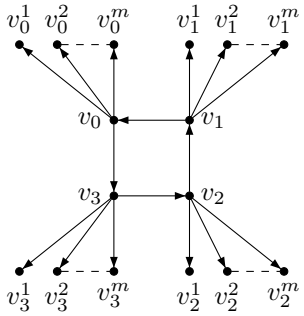
Analog haben die Spieler v_1 und v_2 auch optimale Strategien.
Wir betrachten nun einen Nichtdreieckknoten. Ohne Einschränkung der Allgemeinheit betrachten wir v_0^1 bzw. v_1^1.

1. Der Spieler baut eine oder zwei Kanten. Wieder sind Knoten im Dreieck sinnvoller. Die Distanz zu dem Dreiecksknoten und den zugehörigen Knoten fällt um 1. Es ist $\alpha - (m+1) \geq 0$ bzw. $\alpha - (l+1) \geq 0$ für jede Kante.

2. Der Spieler könnte drei oder mehr Kanten bauen. Dies würde wieder ab der dritten Kanten maximal nur für einen Knoten einen Distanzgewinn von 1 bringen.

Wir zeigen für $m \geq 1$, dass der Graph transient sind. Der Spieler v_1^1 baut mit gleichen Kosten eine Kante zu v_2. Der Spieler v_2 kann nun die Kante (v_2, v_1) löschen und hätte einen Gewinn von $\alpha - m = 1$. □

Beispiel 4.35. *Es sei $m \in \mathbb{N}$. Der folgende Graph ist ein transientes Nash-Gleichgewicht für $\alpha = 2(m+1)$.*



Beweis: Die Spieler $v_i \in C_4$ haben nach Lemma 4.32 und Beispiel 4.17 eine optimale Strategie für $\alpha \in [(m+1), 2(m+1)]$. Wir betrachten nun einen Spieler der nicht im Viereck ist. Ohne Einschränkung der Allgemeinheit sei es v_0^1. Falls dieser Kanten baut, dann sind wieder die Knoten des Vierecks v_1, v_2 und v_3 sinnvoller.

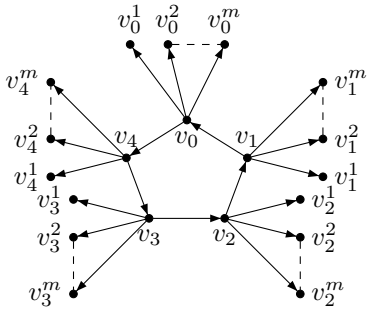
1. Der Spieler v_0^1 baut eine Kante zu v_1 (oder v_3). Die Distanz zu v_1 und v_2 und den zugehörigen Knoten v_1^i und v_2^i verringert sich jeweils um 1. Es ist $\alpha - 2(m+1) = 0$.

2. Der Spieler v_0^1 baut eine Kante zu v_2. Die Distanz zu v_2 und den zugehörigen Knoten v_2^i verringert sich jeweils um 2. Es ist $\alpha - 2(m+1) = 0$.

3. Wir schätzen Strategien mit zwei (oder mehr Kanten) durch die besten Strategien mit einer Kante ab. Die beste Strategie mit einer Kante brachte einen Distanzgewinn von $2(m+1)$. Somit kann die beste Strategie mit zwei Kanten maximal einen Distanzgewinn von $4(m+1)$ haben. Es ist $2\alpha - 4(m+1) = 0$. (Tatsächlich kann durch zwei Kanten nur ein Distanzgewinn von $3(m+1)$ erfolgen.) Analoges gilt für drei und mehr Kanten.

Es bleibt zu zeigen, dass der Graph transient ist. Der Spieler v_0^1 baut mit gleichen Kosten eine Kante zu v_1. Nun würde dem Spieler v_1 das Löschen der Kante $\{v_0, v_1\}$ einen Gewinn von $\alpha - m = 1$ bringen. □

Beispiel 4.36. *Es sei $m \in \mathbb{N}$. Der folgende Graph ist ein transientes Nash-Gleichgewicht für $3(m+1) \leq \alpha \leq 4(m+1)$.*



Beweis: Die Spieler v_i haben nach Lemma 4.32 und Beispiel 4.19 eine optimale Strategie für $\alpha \in [(m+1), 4(m+1)]$. Wir betrachten einen Spieler, der nicht im Fünfeck ist. Ohne Einschränkung der Allgemeinheit sei es v_0^1. Wie in den vorherigen Beispielen brauchen wir nur Strategien zu den Knoten des Fünfecks zu betrachten.

1. Der Spieler v_0^1 baut eine Kante zu v_1 (oder v_4). Die Distanz zu v_1, v_1^i, v_2 und v_2^i verringert sich jeweils um 1. Es ist $\alpha - 2(m+1) \geq 0$.

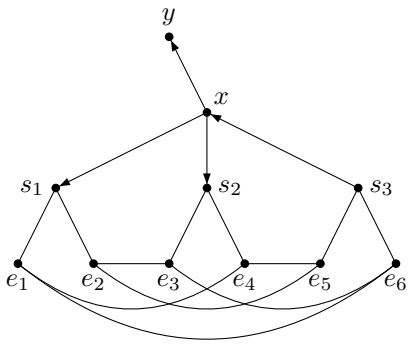
2. Der Spieler v_0^1 baut eine Kante zu v_2 (oder v_3). Die Distanz zu v_3 und v_3^i verringert sich um 2 und die Distanz zu v_3 und v_3^i verringert sich um 1. Die Kostendifferenz ist $\alpha - 3(m+1) \geq 0$.

3. Der Spieler v_0^1 baut zwei Kanten. Die beste Strategie brachte einen Distanzgewinn von $3(m+1)$. Die beste mögliche Strategie mit zwei Kanten, kann höchstens einen Distanzgewinn von $6(m+1)$ bringen. Es ist $2\alpha + 6(m+1) \geq 0$. Analoges gilt für drei oder mehr Kanten.

Da das Fünfeck nach Beispiel 4.30 transient ist, ist auch dieser Graph transient. □

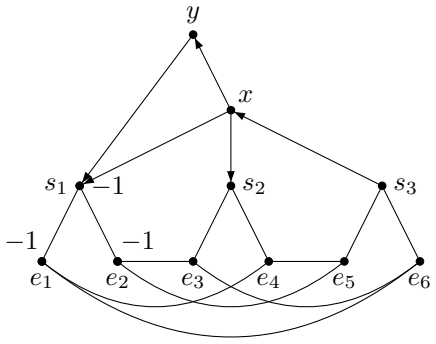
Beispiel 4.37. *Es sei $P = (V_P, E_P)$ der Peterson-Graph und $m \in \mathbb{N}$. Der Graph $P(m)$ ist für $\alpha = 4(m+1)$ ein Nash-Gleichgewicht.*

Beweis: Die Spieler $x \in V_P$ haben nach Lemma 4.32 und Beispiel 4.25 eine optimale Strategie für $\alpha \in [(m+1), 4(m+1)]$. Wir betrachten nun einen Spieler y, der nicht im Peterson-Graph ist und dessen Vater x ist Wie in den vorherigen Beispielen brauchen wir nur Strategien zu den Knoten des Peterson-Graphen zu betrachten. Da der Peterson-Graph knotenautomorph ist, sieht die Situation für den Spieler y ohne Einschränkung der Allgemeinheit folgendermaßen aus



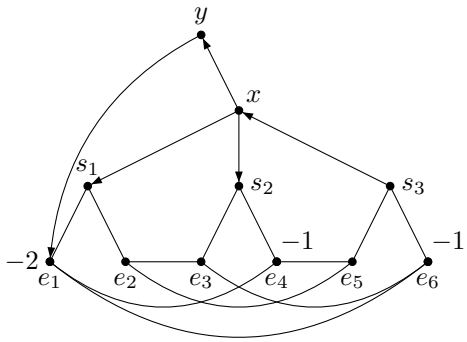
Alle Spieler außer y, die nicht im Peterson-Graph sind, wurden weggelassen.

1. Der Spieler baut eine Kante zu einem Sohn von x. Ohne Einschränkung sei es der Sohn s_1.



   Die Distanz zu s_1, e_1 und e_2 und den dazugehörigen Knoten s_1^i, e_1^i und e_2^i sinkt um 1. Die Kostenänderung beträgt $\alpha + 3(m+1) \geq 0$.

2. Der Spieler baut eine Kante zu einem Enkel von x. Ohne Einschränkung sei es der Enkel e_1.



Die Distanz zu e_1 und den dazugehörigen Knoten e_1^i fällt um 2 und die Distanz zu e_4 und e_6 und den dazugehörigen Knoten e_4^i und e_6^i fällt um 1. Die Kostendifferenz beträgt $\alpha + 4(m+1) \geq 0$.

3. Die beste Strategie mit einer Kante brachte einen Distanzgewinn von $4(m+1)$. Somit kann die beste Strategie mit l Kanten maximal einen Distanzgewinn von $4l(m+1)$ erzielen. Es ist

$$l\alpha - 4l(m+1) = l(\alpha - 4(m+1) \geq 0.$$

Nach Beispiel 4.28 ist der Peterson-Graph transient und $P(m)$ ist somit auch transient. □

4.4 Die Baumvermutung

In diesem Abschnitt behandeln wir die Baumvwermutung von [FLM+03]. Die Autoren Fabrikant und andere haben nur ein Nash-Gleichgewicht gefunden, welches kein Baum ist. Dieses Nash-Gleichgewicht ist der Peterson-Graph, welcher transient ist. Daher haben die Autoren die folgende Vermutung aufgestellt.

Vermutung 4.38. *Es existiert ein $A > 0$, so dass für alle $\alpha \geq A$ jedes nichttransiente Nash-Gleichgewicht mit den Baukosten α ein Baum ist.*

Wir werden in diesem Abschnitt die Vermutung widerlegen und zeigen, dass für alle $\alpha > 2$ sogar ein starkes Nash-Gleichgewicht existiert, welches nicht ein Baum ist. Der Author hat dieses Ergebnis auf den ersten Seiten in [AEED+06] veröffentlicht.

Lemma 4.39. *Es sei $q \in \mathbb{N}$. Der vollständige Graph K_q besitzt eine Orientierung mit*

$$\delta^+(v) - \delta^-(v) = 0 \text{ für ungerade } q$$
$$|\delta^+(v) - \delta^-(v)| = 1 \text{ für gerade } q$$

für alle Knoten v.

Beweis: Wir zeigen die Behauptung mit Induktion über q. Der Fall $q = 1$ ist trivial. Wir zeigen die Behauptung von q auf $q+1$.

1. Falls q ungerade ist, erfüllt der Graph K_q nach Voraussetzung $\delta^+_{K_q}(v) - \delta^-_{K_q}(v) = 0$. Wir fügen einen Knoten x hinzu und verbinden diesen Knoten mit jedem Knoten in K_q. Dabei wählen wir eine Orientierung von x so, dass $\delta^+(x) = \frac{q+1}{2}$ und $\delta^-(x) = \frac{q-1}{2}$ ist. Es ist $\delta^+(x) - \delta^-(x) = 1$. Jeder Knoten v von K_q hat nun entweder eine eingehende oder ausgehende Kante mehr und es ist $|\delta^+_{K_{q+1}}(v) - \delta^-_{K_{q+1}}(v)| = 1$.

2. Falls q gerade ist, erfüllt der Graph K_q nach Voraussetzung $|\delta^+_{K_q}(v) - \delta^-_{K_q}(v) = 0|$ für alle Knoten. Wir fügen wieder einen Punkt x hinzu und verbinden diesen mit K_q. Im Fall $\delta^+_{K_q}(v) - \delta^-_{K_q}(v) = 1$

fügen wir die Kante (x, v) hinzu und im Fall $\delta^+_{K_q}(v) - \delta^-_{K_q}(v) = -1$ fügen wir die Kante (v, x) hinzu. Für alle Knoten v in K_q gilt somit $\delta^+_{K_{q+1}}(v) - \delta^-_{K_{q+1}}(v) = 0$. Für den Knoten x gilt

$$\delta^+(x) = |\{v \mid \delta^+_{K_q}(v) - \delta^-_{K_q}(v) = 1\}| = \sum_{v:\delta^+_{K_q}(v)-\delta^-_{K_q}(v)=1} \delta^+_{K_q}(v) - \delta^-_{K_q}(v).$$

$$\delta^-(x) = |\{v \mid \delta^+_{K_q}(v) - \delta^-_{K_q}(v) = -1\}| = -\sum_{v:\delta^+_{K_q}(v)-\delta^-_{K_q}(v)=-1} \delta^+_{K_q}(v) - \delta^-_{K_q}(v).$$

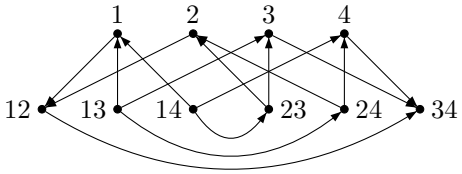
Es folgt

$$\delta^+(x) - \delta^-(x) = \sum_v \delta^+_{K_q}(v) - \delta^-_{K_q}(v) = 0.$$

□

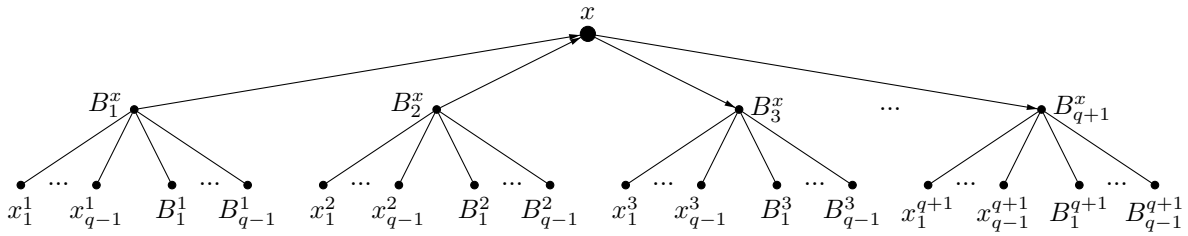
Wir werden zeigen, dass der π-Graph einer affinen Ebene eine starkes Nash-Gleichgewicht ist. Dieser Graph wurde in [Ste74], [Sca86], [BB88] und [BCN89] als Beispiel für einen geodätischen Graph mit Durchmesser 2 betrachtet.
Wir müssen den Graphen eine Orientierung, das heißt eine Gesamtstrategie geben. Es sei $(X, \mathcal{B})$ eine affine Ebene der Ordnung q. Die Anzahl der Blöcke, die zu einem gegeben Block B parallel sind, beträgt $\lambda_{q,0} = q - 1$. Jeder Äquivalenzklasse eines Blockes B ist ein vollständiger Graph K_q im π-Graphen. Wir orientieren K_q wie im obigen Lemma. Für $r := r(B) := \delta^-_{K_q}(B)$ und $s := s(B) := \delta^+_{K_q}(B)$ ist $r + s = q - 1$ und $|r - s| \leq 1$.
Es existieren $\frac{\lambda_0}{\lambda_{q,0}} = q + 1$ viele Äquivalenzklassen. Es seien $B^0, \ldots, B^q$ Repräsentanten dieser Äquivalenzklassen. Die Blöcke der Äquivalenzklasse $[B^q] = \{B^q_0, \ldots B^q_{q-1}\}$ bauen keine Kanten zu den Punkten aus X. Ein Block $B \in [B^i]$ für $0 \leq i \leq q - 1$ baut genau zwei Kanten zu den Punkten $B \cap B^q_i$ und $B \cap B^q_{i+1 \mod q}$. Die restlichen Kanten werden von den Punkten $x \in X$ gebaut.
Wir betrachten da kleinste Beispiel.

Beispiel 4.40. *Es sei $X = \{1, 2, 3, 4\}$ und $\mathcal{B}$ enthalte alle zwei elementigen Teilmengen von X. Das Tupel $(X, \mathcal{B})$ ist ein affine Ebene der Ordnung 2. Der zugehörige π-Graph sieht folgendermaßen aus.*



Dies ist der Peterson-Graph. Da der Block $\{1, 3\}$, im Schaubild mit der Zahl 13 gekennzeichnet, drei Kanten baut, ist dies nur ein Nash-Gleichgewicht für $1 \leq \alpha \leq 3$, statt wie in dem Beispiel 4.25 für $1 \leq \alpha \leq 4$.

Im nächsten Schaubild ist die Situation eines Punktes $x \in X$ für eine affine Ebene der Ordnung q dargestellt.



Es seien B^x_i, $i = 1, \ldots q + 1$ die λ_1 vielen Blöcke, die den Punkt x enthalten. In jeder Äquivalenzklasse $i = 0, \ldots, q + 1$ existiert genau solch ein Block. Die Punkte des Blockes B^x_i seien $x^i_1, \ldots, x^i_{q-1}$ (und

x). Die Blöcke der Äquivalenzklasse von B_i^x seien $B_1^i, \ldots, B_{q-1}^i$. Für $i \neq j$ sind die Punkte x_1^i und x_1^j verschieden, da sonst

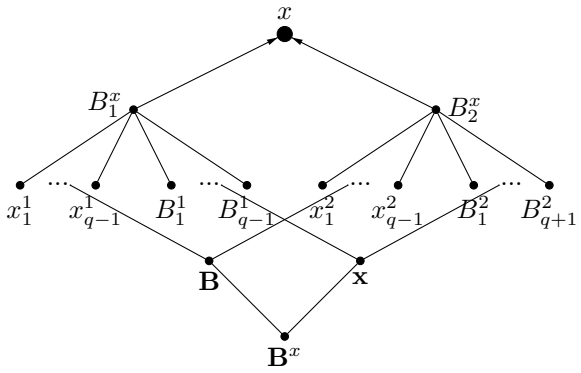
$$B_i^x = (x, x_1^i) = (x, x_1^j) = B_j^x$$

wäre. Da die Blöcke B_1^i und B_1^j für $i \neq j$ in verschiedenen Äquivalenzklassen liegen, haben diese Blöcke genau einen Punkt gemeinsam. Der Punkt x ist in $(x \parallel B^q) = B_j^q$ für ein $0 \leq j \leq q$ und hat genau zwei eingehende Kanten von den Blöcken $(x \parallel B^j)$ und $(x \parallel B^{j-1 \mod q})$. Ohne Einschränkung der Allgemeinheit sei $B_1^x = (x \parallel B^j)$ und $B_2^x = (x \parallel B^{j-1 \mod q})$. Die Kosten des Spielers x betragen

$$c(S_x) = (q-1)\alpha + (q+1) + 2(q+1)(2(q-1)) = (q-1)\alpha + 4q^2 + q - 3.$$

Lemma 4.41. *Es sei $q \geq 3$. Jeder Punkt $x \in X$ hat für $1 \leq \alpha \leq 2(q-1)$ eine optimale Strategie und für $1 < \alpha < 2(q-1)$ eine starke optimale Strategie.*

Beweis: Wir betrachten zunächst die Strategie S_0, bei der der Spieler x keine Kante baut. Sie sieht folgendermaßen aus.

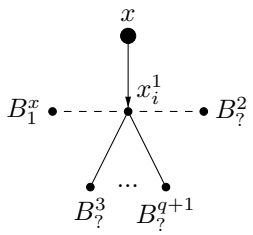


Da die Blöcke B_j^i für $3 \leq i, j \leq q+1$ nicht in der Äquivalenzklasse von B_1^x und B_2^x liegen, existieren Punkte $x_1^?$ und $x_2^?$ mit denen die Blöcke verbunden sind. (Das Fragezeichen ist ein Index, der hier nicht weiter angegeben wird.) Aus dem gleichen Grund sind die Blöcke nicht direkt mit B_1^x oder B_2^x verbunden. Wir haben solche Blöcke mit **B** im Schaubild bezeichnet.
Ein Punkt x_i^j mit $i \geq 3$ und $1 \leq j \leq q-1$ hat die Distanz 3 zum Spieler x, da der Punkte nicht im Block B^1 enthalten ist und es genau einen Block in der Äquivalenzklasse $[B^1]$ gibt, der x_i^j enthält. Analoges gilt für B_2 und die Äquivalenzklasse $[B_2]$. Wir haben solche Punkte mit **x** angedeutet.
Die Blöcke B_i^x mit $3 \leq i \leq q+1$ sind im Schaubild mit $\mathbf{B^x}$ angedeutet. Sie haben die Distanz 4, da sie nicht in der Äquivalenzklasse der $[B_1^x]$ und $[B_2^x]$ sind und keinen der Punkte x_j^1 und x_j^2 für $j = 1, \ldots, q-1$ enthalten. Falls sie zum Beispiel x_j^1 enthalten würden, würde $B_i^x = (x, x_j^1) = B_1^x$ folgen. Die Kostendifferenz zur Nash-Strategie beträgt

$$\delta(S_0) = c(S_0) - c(S_x) = -(q-1)\alpha + (q-1)2(q-1) + 3(q-1) = (q-1)\left(2q + 1 - \alpha\right) > 0.$$

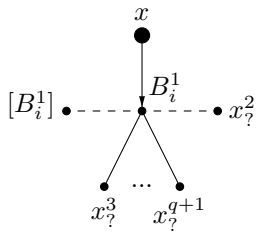
Ausgehend von der Strategie S_0 betrachten nun alle mögliche Strategie S_1 mit genau einer Kante. Wir werden uns für die Anzahl der Knoten interessieren, die dadurch auf Distanz 2 kommen. Wir müssen dafür fünf Fälle unterscheiden.

1. Der Spieler x baut eine Kante zu einem Knoten x_i^1 oder x_i^2 für $1 \leq i \leq q-1$. Ohne Einschränkung der Allgemeinheit sei es x_i^1.



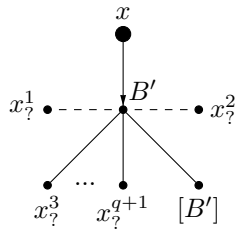
Der Punkt x_i^1 liegt in genau $q+1$ vielen Blöcken. Davon ist ein Block B_1^x und ein Block $B_?^2$ aus der Äquivalenzklasse von $[B_2^x]$. Diese beiden Blöcke waren vorher schon auf Distanz 1 oder 2. Die Anzahl der neuen Knoten mit Distanz 2 beträgt somit $q-1$.

2. Der Spieler baut eine Kante zu einem Block B_i^1 oder B_i^2 für $1 \leq i \leq q-1$. Ohne Einschränkung der Allgemeinheit betrachten wir B_i^1.



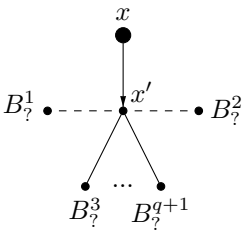
Jeder Block in der gleichen Äquivalenzklasse von B_i^1 hat schon die Distanz 1 oder 2 zum Spieler x. Weiterhin enthält B_i^1 einen Punkt $x_?^2$ aus B_2^x. Dieser hatte vorher auch schon die Distanz 2. Somit haben $q-1$ viele Punkt die Distanz 2 bekommen.

3. Der Spieler baut ein Kante zu einem Block B' mit $B' \notin [B_1^x] \cup [B_2^x]$, welcher nicht x enthält.



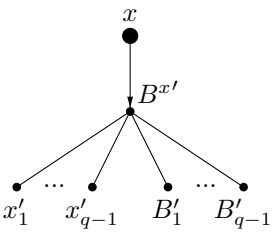
Der Block B' enthält einen Punkt $x_?^1$ aus B_1^x und einen Punkt $x_?^2$ aus B_2^x. Deren Distanz sinkt nicht. Neben den verbleibenden Punkten $q-2$ vielen Punkten haben auch die Blöcke der gleichen Äquivalenzklasse die Distanz 2. Es haben somit $2q-3$ viele Knoten die Distanz 2 bekommen. (Im Schaubild sind die Punkte mit Distanz 2 durch $x_?^3, \ldots, x^{q+1}$ dargestellt. Einen zur Äquivalenzklasse von B' zugehörigen Punkt gibt es aber nicht.)

4. Der Spieler baut zu einem Punkt $x' \notin \{x_1^1, \ldots, x_{q-1}^1, x_1^2, \ldots, x_{q-1}^2\}$.



Der Punkt x' ist in $q+1$ vielen Blöcken, jeder Block befindet sich in einer anderen Äquivalenzklasse. Zwei dieser Blöcke sind in der Äquivalenzklasse von B_1^x und B_2^x und haben somit keine Distanzverbesserung. Die Anzahl der neuen Knoten mit Distanz 2 ist $q-1$.

5. Der Spieler baut eine Kante zu einem Block $B^x \neq B_1^x, B_2^x$, welcher x enthält.



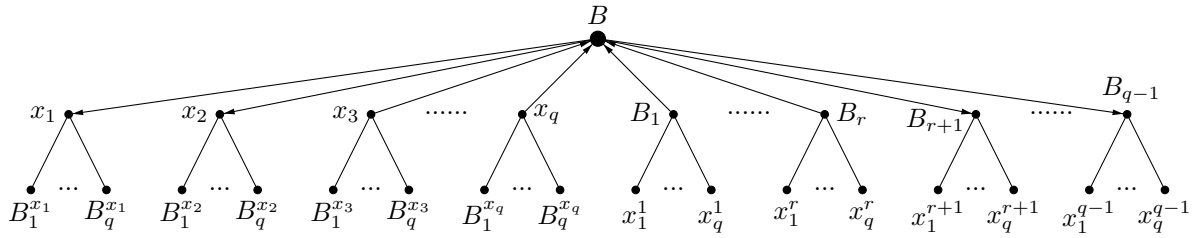
In diesem Fall ist die Anzahl der neuen Knoten mit Distanz 2 maximal und beträgt $2(q-1)$. Die Nash-Strategie besteht nur aus diesen Kanten. Auch wenn mehrere solcher Kanten gebaut werden, bleibt für jede Kante die Anzahl der neuen Knoten mit Distanz 2 immer $2(q-1)$.

Wenn der Spieler x zu einer Strategie S_l mit $l < q-1$ vielen Links wechselt, dann haben mindestens $(q-1-l)2(q-1)$ viele Knoten die Distanz 3 oder 4. Die Kostenänderung zur Nash-Strategie beträgt

$$\delta(S_l) \geq -(q-1-l)\alpha + (q-1-l)2(q-1) = (q-1-l)(2(q-1)-\alpha) \geq 0.$$

Jede andere Strategie S_{q-1} mit $q-1$ vielen Kanten außer der Nash-Strategie hat mindestens einen Knoten mit der Distanz 3 und somit ist die Kostenänderung $c(S_{q-1}) \geq 1 > 0$.
Jede Strategie mit mehr als $q+1$ Kanten, könnte maximal nur für einen Knoten die Distanz von 2 auf 1 verbessern und ist wegen $\alpha \geq 1$ schlecht. Für das starke Nash-Gleichgewicht können wir den Beweis analog führen. □

Das nächste Schaubild zeigt die Situation eines Blockes $B \notin [B^q]$.



Es seien $x_1, \ldots, x_q$ die Punkte, die der Block B enthält und $B_1, \ldots, B_{q-1}$ die zum Block B parallelen Blöcke. Der Block B hat eine eingehende Kante von den ersten r vielen Blöcken $B_1, \ldots, B_r$ und baut eine Kante zu den s vielen Blöcken $B_{r+1}, \ldots, B_{q-1}$, wobei r und s wie in Lemma 4.39 sind. Da wir einen Block $B \notin [B_q]$ betrachten baut der Block B genau zwei Kanten zu Punkten x_i. Ohne Einschränkung seien dies die Punkte x_1 und x_2.
Jeder Punkt x_i für $i = 1, \ldots, q$ ist in weiteren $\lambda_{2,1} = q$ vielen Blöcken, die wir mit $B_1^{x_i}, \ldots, B_q^{x_i}$ bezeichnen. Jeder Block B_i besitzt die q vielen Punkte $x_1^i, \ldots, x_q^i$. Da B und B_i in der selben Äquivalenzklasse und somit paarweise disjunkt sind, sind die Punkte x_i, x_j^k paarweise verschieden. Die Blöcke $B_1^{x_i}$ und $B_1^{x_j}$ sind für $i \neq j$ verschieden, da B der einzige Block mit den beiden Punkten x_i, x_j ist. Weiterhin ist ein Block $B_j^{x_i}$ verschieden von B_k, da der Block $B_j^{x_i}$ den Punkt x_i und der Block B_k den Punkt x_i nicht enthält. Die Kosten betragen

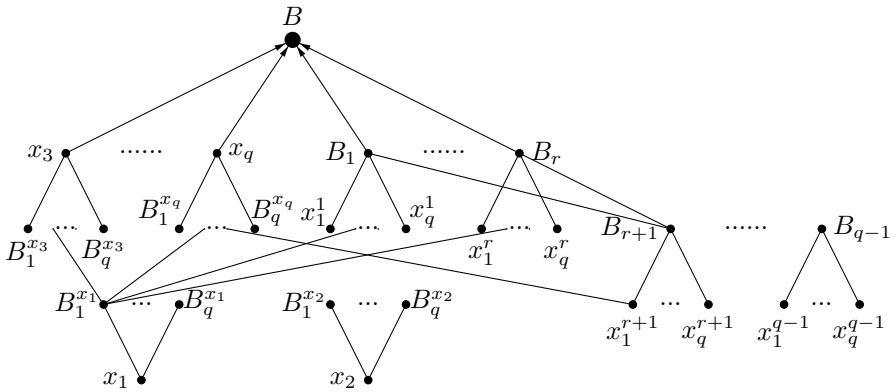
$$c(S_B) = (2+s)\alpha + (2q-1) + 2(2q-1)q = (s+2)\alpha + 4q^2 - 1.$$

Analog betragen die Kosten für einen Block $B \in [B^q]$.

$$c(S_B) = s\alpha + 4q^2 - 1.$$

Lemma 4.42. *Es sei $q \geq 11$. Jeder Block hat eine optimale Strategie für $1 \leq \alpha \leq q+1$ Die Strategie ist für $1 < \alpha < q+1$ stark.*

Beweis: Es sei $B \notin [B^q]$ ein Block. Wir betrachten zunächst die Strategie S_0, bei der B keine Kante baut. Das folgende Schaubild zeigt die Strategie S_0.

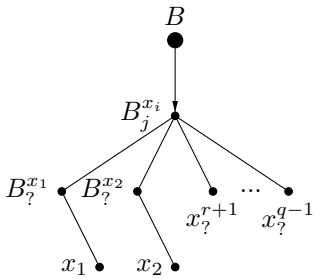


Die Blöcke $B_{r+1}, \ldots, B_{q-1}$ haben zu B die Distanz 2, da sie mit den Blöcken $B_1, \ldots B_r$ in der gleichen Äquivalenzklassen sind und die zugehörigen Punkte $x_1^{r+1}, \ldots, x_q^{r+1}, x_1^{r+2}, \ldots, x_q^{q-1}$ haben nun Distanz 3, da sie nicht in den Blöcken $B_1, \ldots, B_r$ enthalten sind. Jeder dieser Punkte ist in genau einem der Blöcke $B_{r+1}, \ldots B_{q-1}$ und genau in einem Block $B_?^{x_i}$ für $3 \leq i \leq q$. Die Blöcke $B_1^{x_1}, \ldots, B_q^{x_1}, B_1^{x_2}, \ldots, B_q^{x_2}$, die neben B den Punkte x_1 oder x_2 enthalten, haben die Distanz 3, da sie keinen der Punkte $x_3, \ldots, x_q$ enthalten, noch in der Äquivalenzklasse von B sind. Jeder der Blöcke $B_1^{x_1}, \ldots, B_q^{x_1}, B_1^{x_2}, \ldots, B_q^{x_2}$ besitzt für jedes x_i, $i = 1, \ldots, q$ genau einen adjazenten Block $B_?^{x_i}$. Weiterhin ist jeder dieser Blöcke genau über einen Punkt $x_?^i$ mit B_i für $1 \leq i \leq r$ verbunden. Die Punkte x_1 und x_2 sind in den Blöcken $B_1^{x_1}, \ldots, B_q^{x_1}, B_1^{x_2}, \ldots, B_q^{x_2}$, welche alle Distanz 3 haben. x_1 und x_2 haben somit die Distanz 4. Die Kostendifferenz der Strategie S_0 und der Nash-Strategie beträgt

$$\delta(S_0) = c(S_0) - c(S_B) = -(s+2)\alpha + s(q+1) + 2q + 6 = (q+1-\alpha)(s+2) + 4 > 0.$$

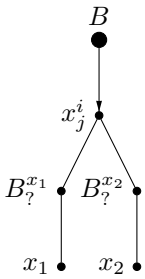
Wir betrachten nun alle Strategien mit einem Link. Wir müssen sechs Fälle unterscheiden.

1. Der Spieler B baut eine Kante zu einem Block $B_j^{x_i}$ mit $3 \leq i \leq q$, $1 \leq j \leq q$.



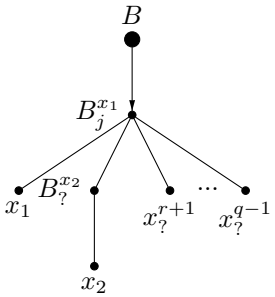
   Der Block $B_j^{x_i}$ ist mit genau einem Block $B_?^{x-1}$ der Blöcke $B_1^{x_1}, \ldots, B_q^{x_1}$ verbunden, welcher wiederum mit x_1 verbunden ist. Analog ist $B_j^{x_i}$ mit einem $B_?^{x_2}$ verbunden, welcher wiederum mit x_2 verbunden ist. Weiterhin ist B mit dem gemeinsamen Punkt $x_?^k$ von $B_j^{x_i}$ und B_k für $k = r+1, \ldots, q-1$ verbunden. Alle anderen Knoten haben keinen Distanzgewinn erfahren. Der Distanzgewinn einer solchen Strategie beträgt $s + 5$.

2. Der Spieler B baut eine Kante zu einem Punkt x_j^i mit $1 \leq i \leq r$ und $1 \leq j \leq q$.



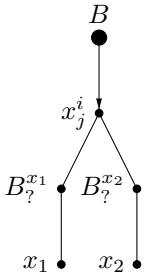
Der Punkt x_j^i ist mit genau einem Block $B_?^{x_1}$ der Blöcke $B_1^{x_1}, \ldots, B_q^{x_1}$, welcher wiederum mit x_1 verbunden ist. Analoges gilt für $B_?^{x_2}$ und x_2. Der Distanzgewinn beträgt 5.

3. Der Spieler B baut eine Kante zu einem Block $B_j^{x_1}$ oder $B_j^{x_2}$ mit $1 \leq j \leq q$. Ohne Einschränkung betrachten wir den Block $B_j^{x_1}$.



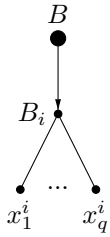
Der Block $B_j^{x_1}$ enthält den Punkt x_1. Weiterhin enthält $B_j^{x_1}$ für alle $k = r+1, \ldots, q-1$ genau ein Punkt von $x_1^k, \ldots, x_q^k$, welchen wir mit $x_?^k$ dargestellt haben. Der Block ist zu einem der Blöcke $B_1^{x_2}, \ldots B_q^{x_2}$ parallel. Dieser parallele Block enthält x_2. Der Distanzgewinn beträgt $s+6$.

4. Der Spieler B baut zu einem Punkt x_j^i mit $r+1 \leq i \leq q$ und $1 \leq j \leq q$.



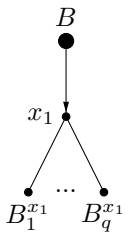
Der Punkt x_j^i ist mit einem Block $B_?^{x_1}$ von den Blöcken $B_1^{x_1}, \ldots B_q^{x_1}$ verbunden. Der Block $B_?^{x_1}$ ist mit x_1 verbunden. Analoges gilt für $B_?^{x_2}$ und x_2. Der Distanzgewinn beträgt 5.

5. Der Spieler baut eine Kante zu einem Block B_i mit $r \leq i \leq q-1$.



Der Block B_i enthält die Punkte $x_1^i, \dots x_q^i$. Der Distanzgewinn beträgt $q+1$.

6. Der Spieler B baut eine Kante zu dem Punkt x_1 oder x_2. Ohne Einschränkung der Allgemeinheit betrachten wir x_1.



Der Punkt x_1 ist in den Blöcken $B_1^{x_1}, \dots, B_q^{x_1}$. Der Distanzgewinn beträgt $q+3$

Die letzten beiden Strategien 5 und 6 lassen sich mischen, ohne dass der Distanzgewinn sich verringert. Die Nash-Strategie besteht aus solchen Kanten. Für $q \geq 11$ ist

$$q+1 > \frac{q}{2} + 6 \geq s+6.$$

Somit sind die letzten beiden Strategien besser als die anderen. Es sei S_l eine Strategie von B mit $l < s+2$. Wir ersetzen alle Strategien der Form 1 bis 4 durch die besseren Strategien der Form 5 und 6. Die Kosten können dadurch nur sinken. Da $l < s+2$ ist, können durch den Bau eines weiteren Links die Kosten mindestens um $q+1-\alpha \geq 0$ gesenkt werden.
Eine Strategie mit $q+1$ links ist genau dann optimal, wenn alle Strategien des Typs 5 und 6 genommen werden. In der Nash-Strategie haben alle Knoten die Distanz 2 oder 1 Eine Strategie mit $l > q+1$ vielen Kanten könnte maximal die Distanz von $l-(q+1)$ vielen Knoten von 2 auf 1 senken. Es ist

$$(l-(q+1))\alpha - (l-(q+1)) = (l-(q+1))(\alpha-1) \geq 0.$$

Mit Lemma 4.24 folgt die Behauptung auch für $B \in [B^q]$. Das starke Nash-Gleichgewicht sieht man analog □

Wir haben den folgenden Satz bewiesen.

Satz 4.43. *Der π-Graph einer affinen Ebene der Ordnung $q \geq 11$ ist für $1 \leq \alpha \leq q+1$ ein Nash-Gleichgewicht und für $1 < \alpha < q+1$ ein starkes Nash-Gleichgewicht.*

Korollar 4.44. *Die Baum-Vermutung ist falsch.*

Kapitel 5

Approximative Nash-Gleichgewichte in Netzwerk-Spielen

5.1 Einführung

In diesem Kapitel betrachten wir das Netzwerk-Spiel. Das Kapitel beruht auf [ADTW03] und [Eil06]. Gegeben ist ein ungerichtetes Netzwerk $G = (V, E, c)$ und eine Menge von Spielern I. Jeder Spieler i besitzt eine Funktion

$$f_i : \mathcal{P}(V) \to \mathbb{N}_0.$$

Eine **Spieler-Strategie** $(x_e^i)_{e \in E}$ des Spielers i ist eine Kantenfunktion x^i mit Werten in $\mathbb{R}_0^+$. Der Vektor $(x_e^i)_{e \in E, i \in I}$ ist die **Gesamt-Strategie**. Eine Kante e ist in der Gesamtstrategie **gekauft**, falls $\sum x_e^i \geq c_e$ ist. Gekaufte Kanten können von jedem Spieler benutzt werden. Es sei

$$x_e := \begin{cases} 1 & \text{falls } e \text{ gekauft ist} \\ 0 & \text{sonst} \end{cases}$$

der charakteristische Vektor der gekauften Kanten. Für jeden Spieler i muss das Netzwerk der gekauften Kanten **zulässig** sein, dass heißt es müssen die Bedingungen

$$\sum_{e \in \delta(S)} x_e \geq f_i(S), \forall S \subseteq V$$

erfüllt sein. Falls mindestens eine Bedingung nicht erfüllt ist, nennen wir die Spieler-Strategie und die Gesamtstratgie **unzulässig**. Die Kosten des Spielers i betragen

$$c_i := c_i((x_e^i)_{e \in E, i \in I}) := \sum_{e \in E} x_e^i.$$

Da die Spieler ihre Kosten minimieren wollen, werden wir für alle Kanten e ohne Einschränkung der Allgemeinheit $\sum x_e^i \leq c_e$ annehmen. Eine Kante ist somit gekauft, falls $\sum x_e^i = c_e$ ist.
Mit der Funktion $f(S) := \max_{i \in I} f_i(S)$ erfüllt das Netzwerk der gekauften Kanten einer zulässigen Gesamtstrategie das folgende globale Problem

$$\begin{aligned} \sum_e c_e x_e &\to \min \\ \sum_{e \in \delta(S)} x_e &\geq f(S) \forall S \subseteq V \\ x_e &\in \{0, 1\}. \end{aligned}$$

Dies ist das globale Netzwerk-Problem CP(f). Die Autoren Anshelevich, Dasgupta, Tardos und Wexler ([ADTW03]) haben das folgende Teilspiel betrachtet.

Beispiel 5.1 (Steiner-Wald-Spiel)**.** *Gegeben sei ein ungerichtetes Netzwerk $G = (V, E, c)$. Jeder Spieler $i \in I$ besitzt eine Terminalmenge $T_i \subseteq V$, die der Spieler im Netzwerk der gekauften Kanten verbinden möchte. Der Satz von Menger zeigt, dass dies ein Netzwerk-Spiel mit der Funktion*

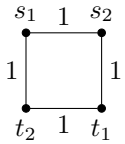
$$f_i(S) = \begin{cases} 1 & \text{falls } \emptyset \neq S \cap T_i \neq T_i \\ 0 & \text{sonst} \end{cases}$$

ist.

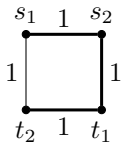
Das zugehörige globale Problem ist das Steiner-Wald-Problem. Dieses enthält das Steiner-Baum-Problem, welches $\mathcal{APX}$-hart ist [BP89]. Somit ist die Berechnung einer optimalen Spieler-Strategie für einen Spieler auch $\mathcal{APX}$-hart. [ADTW03] haben mit dem folgenden Beispiel gezeigt, dass im Allgemeinen kein Nash-Gleichgewicht existiert.

Beispiel 5.2. *Im Steiner-Wald-Spiel existiert im Allgemeinen kein Nash-Gleichgewicht.*

Beweis: Wir betrachten das folgende Netzwerk.



Der Spieler 1 besitzt die Terminalmenge $T_1 = \{s_1, t_1\}$ und der Spieler 2 besitzt die Terminalmenge $T_2 = \{s_2, t_2\}$.
Jede globale Lösung besitzt mindesten drei Kanten. Ohne Einschränkung der Allgemeinheit betrachten wir die folgende durch dicke Kanten gekennzeichnete optimale Lösung.



Die Kante $\{s1, s2\}$ wird nur von Spieler 1 benutzt. Daher muss Spieler 1 in einem Nash-Gleichgewicht die komplette Kante mit Kosten 1 bezahlen. Analog bezahlt nur der Spieler 2 die Kante $\{t_1, t_2\}$. Die Kante $e = \{s_2, t_1\}$ wir von beiden Spielern benutzt. Wir nehmen ohne Einschränkung der Allgemeinheit an, dass Spieler 1 einen positiven Anteil $0 < x_e^1 \leq 1$ in die Kante e investiert. Spieler 1 hat die Kosten $1 + x_e^1$. Der Spieler könnte die Investitionen in die Kanten $\{s_1, s_2\}$ und e zurücknehmen und 1 in die Kante $\{s_1, t_2\}$ investieren. Die Terminals von Spieler 1 sind verbunden, da Spieler 1 die von Spieler 2 gekaufte Kante $\{t_1, t_2\}$ kostenlos mitbenutzen kann. Es folgt, dass die globale Lösung kein Nash-Gleichgewicht ist. Die anderen Fälle sind analog. □

Wir werden dies Beispiel später noch genauer betrachten. Weiterhin haben [ADTW03] gezeigt, dass der Preis der Stabilität und der Preis der Anarchy $|I|$, die Anzahl der Spieler, sein kann, falls Nash-Gleichgewichte existieren. Wir werden daher approximative Nash-Gleichgewichte betrachten.

Definition 5.3 (approximative Nash-Gleichgewicht)**.** *Es sei $\alpha \geq 1$. Die Gesamt-Strategie $(x_e^i)_{e \in E, i \in I}$ ist ein α-**approximatives Nash-Gleichgewicht**, falls für alle Spieler $i \in I$ und alle zulässigen Spieler-Strategien $(\overline{x}_e^i)_{e \in E}$*

$$\alpha c_i((x_e^j)_{e \in E, j \in I, j \neq i} \times (\overline{x}_e^i)_{e \in E}) \geq c_i((x_e^j)_{e \in E, j \in I})$$

ist. Ein Graph G ist ein α-approximatives Nash-Gleichgewicht, wenn ein α-approximatives Nash-Gleichgewicht existiert, dessen Kaufgraph G ist

In einem α-approximative Nash-Gleichgewicht kann kein Spieler seine Kosten um den Faktor α senken, unter Beibehaltung der anderen Spieler-Strategien. 1-approximative Nash-Gleichgewichte sind Nash-Gleichgewichte. Im Steiner-Wald-Spiel haben [ADTW03] gezeigt, dass OPT ein 3-approximatives Nash-Gleichgewicht ist. Die Berechnung von OPT ist das Steiner-Wald-Problem, welches $\mathcal{APX}$-hart ist. Daher

```
01 X ← ∅
02 (x_e^i)_{e,i} ← 0
03 y ← 0
04 l ← 0
05 SOLANGE X unzulässig ist:
06    l ← l + 1
07    V(X) = {S | S ist minimal verletzt bzgl. X}
08    Erhöhe gleichmäßig y_S um ε_l für alle S ∈ V bis ein dichtes e_l existiert.
09    X ← X ∪ {e_l}
10 FÜR k ← l BIS 1
11    FALLS X\{e_k} zulässige Lösung ist DANN X ← X\{e_k}
12    INITIALISIERE Stapel
13    LEGE alle e ∈ δ(S) für S ∈ V({e_1,...,e_{k-1}}) mit |X ∩ δ(S)| ≥ 2α auf den Stapel
14    Für alle S ∈ V({e_1,...,e_{k-1}}) mit |X ∩ δ(S)| ≤ 2α − 1 wähle einen Spieler i mit
          f_i(S) = 1
15       FÜR alle e ∈ δ(S)
16          x_e^i ← x_e^i + ε_k
17       FÜR j ← 1 BIS 2α − 1 − |X ∩ δ(S)|
18          NIMM Kante e des Stapels
19          x_e^i ← x_e^i + ε_k
20 AUSGABE (x_e^i)_{e∈ E,i ∈ I}
```

Abbildung 5.1: Der Algorithmus NG

werden wir uns gute Approximationen für OPT anschauen. [ADTW03] haben gezeigt, dass jede globale 2-Approximation ein $4.65 + \epsilon$-approximatives Nash-Gleichgewicht ist.
Wir werden zeigen, dass der Primal-Dual-Algorithmus PDg, welcher eine globale 2-Approximation berechnet, auch ein 3-approximatives Nash-Gleichgewicht ist. Unser Ergebnis ist allgemeiner und nicht nur auf die Steiner-Wald-Spiele begrenzt. Wir werden zeigen, dass alle globalen Approximationen des Satzes 2.39 auch approximative Nash-Gleichgewichte sind. Wir modifizieren den Algorithmus PDg in Abbildung 2.4 zu dem Algorithmus NG in Abbildung 5.1, welcher ein approximatives Nash-Gleichgewicht berechnet.

Satz 5.4. *Es sei $\alpha \in \mathbb{N}$ und* PDg *in Polynomialzeit für das Netzwerk-Problem* CP(f). *Weiterhin gelte für alle unzulässigen Lösungen X und minimalen Erweiterungen Y von X*

$$\sum_{S \in V(X)} |Y \cap \delta(S)| \leq \alpha |V(X)|.$$

Dann berechnet der Algorithmus ***NG*** *in Polynomialzeit ein $2\alpha - 1$-approximatives Nash-Gleichgewicht.*

Beweis: Wir nehmen an, dass wir uns in der k-ten Iteration der Schleife des inversen Löschschrittes befinden und die Kante e_k eventuell nicht mehr in der Menge X ist. Eine Menge $S \in V(\{e_1, \ldots, e_{k-1}\})$ heißt **gut** , falls $|X \cap \delta(S)| < \alpha$ ist. Falls $\alpha \leq |X \cap \delta(S)| \leq 2\alpha - 1$ ist, heißt S **neutral**, sonst **schlecht**. Jede dieser Mengen ist verletzt bzgl. $\{e_1, \ldots, e_{k-1}\}$. Für gute und neutrale Mengen S wird genau ein Spieler i mit $f_i(S) = 1$ gewählt, der in alle Kanten $e \in X \cap \delta(S)$ (zusätzlich) ϵ_k investiert.
Wir behaupten, dass die untere Schranke des Spielers i sich um ϵ_k erhöht. Wir betrachten zunächst die Strategie des Spielers i, in der der Spiele in keine Kante investiert. In dieser Strategie hat der Spieler sein Ziel nicht erreicht. Da $f_i(S) = 1$ ist, muss der Spieler eine Schnittkante in S haben. Die Kanten $e \in \delta(S)$, $e \in X$ sind nicht gekauft, da mindesten der ϵ_k-Teil des Spielers i fehlt. Die Kanten $e \in \delta(S), e \notin X$ sind nicht in der globalen Approximationslösung und werden daher von den Spielern nicht gekauft. Der Spieler muss also in mindestens eine Kante $e \in \delta(S)$ investieren.

1. Falls er in eine Kante $e \in \delta(S) \cap X$ benutzt, muss er die fehlenden ϵ_k bezahlen.

2. Wenn er eine Kante $e \in \delta(S)$, $e \notin X$ benutzt muss er auch mindestens ϵ_k bezahlen, da die duale Variable y_S in Iteration $l = k$ um ϵ_k auf e gewachsen ist und somit $c_e \geq y_S \geq \epsilon_k$ ist.

Es folgt, dass der Spieler eine untere Schrank von ϵ_k hat. Es ist leicht zu sehen, dass sich die unteren Schranken eines Spielers addieren, da sich auf den Kanten auch die dualen Variablen addieren. Der Spieler einer guten oder neutralen Menge S kann ϵ_k in alle Schnittkanten der guten oder neutralen Menge und in $(2\alpha - 1 - |X \cap \delta(S)|)$ vielen Schnittkanten der bösen Mengen investieren.
Wir werden zeigen, dass in jeder Iteration k auch die bösen Kanten gekauft werden. Durch den inversen Löschschritt ist die Menge X in der Iteration k eine minimale Erweiterung von $\{e_1, \ldots, e_{k-1}\}$. Nach Voraussetzung ist somit

$$\sum_{S \in V(\{e_1,\ldots,e_{k-1}\})} |X \cap \delta(S)| \leq \alpha |V(\{e_1, \ldots, e_{k-1}\})|.$$

Es bezeichne G bzw. N bzw. B die Familie der guten bzw. neutralen bzw. bösen Mengen. Es ist

$$\sum_{S \in G} |X \cap \delta(S)| + \sum_{S \in N} |X \cap \delta(S)| + \sum_{S \in B} |X \cap \delta(S)| \leq \alpha(|G| + |N| + |B|).$$

Beim Entfernen der neutralen Mengen bleibt die Ungleichung erhalten.

$$\sum_{S \in G} |X \cap \delta(S)| + \sum_{S \in B} |X \cap \delta(S)| \leq \alpha(|G| + |B|) \tag{5.1}$$

Jede gute Menge hat mindestens eine Schnittkante in $X \cap \delta(S)$ und jede böse Menge hat nach Definiton mindestens 2α viele Kanten. Es folgt

$$|G| + 2\alpha|B| \leq \sum_{S \in G} |Y \cap \delta(S)| + \sum_{S \in N} |Y \cap \delta(S)|$$

und

$$\alpha|B| \leq (\alpha - 1)|G|. \tag{5.2}$$

Wenn wir Gleichung (5.2) in (5.1) einsetzen, erhalten wir

$$\sum_{S \in G} |Y \cap \delta(S)| + \sum_{S \in B} | \cap \delta(S)| \leq (2\alpha - 1)|G|.$$

Somit werden in jeder Iteration die Schnittkanten der bösen Mengen durch Spieler der guten Mengen gekauft. □

Korollar 5.5. *Für binäre, propere Funktionen ist* PDg *nach Satz 2.51 ein 2-Approximationsalgorithmus. Wir können aus der Lösung* PDg *ein 3-approximatives Nash-Gleichgewicht konstruieren.*

Beispiel 5.6. *Im Steiner-Wald-Spiel existiert ein 3-approximatives Nash-Gleichgewicht, welches eine globale 2-Approximation ist.*

Korollar 5.7. *Falls für alle unzulässigen Lösungen X maximal eine verletzte Menge gewählt wird, so gibt es keine bösen und neutralen Mengen und wir erhalten im Satz ein α-approximatives Nash-Gleichgewicht.*

Wir geben das Hitting Set-Spiel als Beispiel an.

Beispiel 5.8 (Hitting Set-Spiel)**.** *Jeder Spieler i besitzt eine Familie von Mengen $\mathcal{S}_i \subseteq \mathcal{P}(U)$ über dem Universum U. Jedes Element $e \in U$ hat die Kosten c_e. Jeder Spieler i kann $x_e^i \geq 0$ in das Element e investieren. Ein Element e ist gekauft, falls*

$$\sum_{i \in I} x_e^i \geq c_e$$

ist. Jeder Spieler muss für jede Menge $S \in \mathcal{S}_i$ ein gekauftes Element e mit $e \in S$ besitzen. Die Kosten des Spielers betragen

$$\sum_{e \in E} x_e^i.$$

Der Algorithmus PDHS *in Abbildung 2.2 berechnet eine globale $s_{\max}$-Approximation mit*

$$s_{\max} = \max_{S \in \bigcup\limits_{i \in I} \mathcal{S}_i} |S|.$$

Nach dem obigen Korollar ist dies ein $s_{\max}$-approximatives Nash-Gleichgewicht.

5.2 Punkt zu Punkt-Spiele

In diesem Abschnitt werden wir eine untere Schranke zum Satz 5.4 konstruieren. Wir betrachten hier das Netzwerk-Spiel mit binären Funktionen f_i. Für eine möglicherweise unzulässige Gesamt-Strategie $(x_e^i)_{e\in E, i\in I}$, die die globale Lösung $(x_e)_{e\in E}$ erzeugt, sind die optimalen Kosten des Spielers i durch das folgende IP(i) gegeben.

$$\begin{aligned}\sum_{e\in E}(c_e - x_e^{\neq i})\alpha_e^i &\to \min\\ \sum_{e\in\delta(S)} \alpha_e^i &\geq f_i(S), \forall S \subseteq V\\ \alpha_e^i &\in \{0,1\}, \forall e \in E,\end{aligned}$$

wobei $x_e^{\neq i} := \sum_{j\neq i, j\in I} x_e^j$ ist. $(\alpha_e^i)_{e\in E}$ sind die wählbaren Variablen.
Für die Kanten e mit $x_e = 0$ ist ohne Einschränkung der Allgemeinheit $x_e^{\neq i} = 0$. Für die Kanten e mit $x_e = 1$ ist $x_e^i \leq c_e - x_e^{\neq i}$. Wir ersetzen $c_e - x_e^{\neq i}$ im obigen IP durch x_e^i und erhalten das stärkere ganzzahlige Programm IP(i).

$$\begin{aligned}\sum_{e:x_e=1} x_e^i\alpha_e^i + \sum_{e:x_e=0} c_e\alpha_e^i &\to \min\\ \sum_{e\in\delta(S)} \alpha_e^i &\geq f_i(S), \forall S\\ \alpha_e^i &\in \{0,1\}, \forall e \in E\end{aligned}$$

Falls die Gesamt-Strategie $(x_e^i)_{e\in E, i\in I}$ minimal und zulässig ist, ist entweder $\sum_{i\in I} x_e^i = c_e$ oder $\sum_{i\in I} x_e^i = 0$. In diesem Fall haben die beiden ganzzahligen Programme den gleichen Wert. Da in diesem Abschnitt f_i binär ist, relaxieren wir $\alpha_e^i \in \{0,1\}$ zu $\alpha_e^i \geq 0$ und erhalten das folgende LP(i).

$$\begin{aligned}\sum_{e:x_e=1} x_e^i\alpha_e^i + \sum_{e:x_e=0} c_e\alpha_e^i &\to \min\\ \sum_{e\in\delta(S)} \alpha_e^i &\geq f_i(S), \forall S\\ \alpha_e^i &\geq 0, \forall e \in E\end{aligned}$$

Das duale Programm DP(i) lautet

$$\sum_{S\subseteq V} f_i(S)y_S^i \to \text{Max} \tag{5.3}$$

$$\sum_{S:e\in\delta(S)} y_S^i \leq c_e x_e^i, \forall e : x_e = 1 \tag{5.4}$$

$$\sum_{S:e\in\delta(S)} y_S^i \leq c_e, \forall e : x_e = 0 \tag{5.5}$$

$$y_S^i \geq 0, \forall S. \tag{5.6}$$

Der Spieler i kann nach Satz 2.34 oder Satz 2.40 das Programm IP(i) und DP(i) mit zugehörigen Schlupfbedingungen in Polynomialzeit optimal lösen, falls der Spieler i nur zwei Terminals verbinden muss. Dies ist im Punkt zu Punkt-Spiel der Fall.

Beispiel 5.9 (Punkt zu Punkt-Spiel)**.** *Gegeben sei ein ungerichtetes Netzwerk $G = (V, E, c)$. Jeder Spieler $i \in I$ besitzt genau zwei Terminals s_i, t_i, die der Spieler im Netzwerk der gekauften Kanten verbinden möchte.*

Li McCormick und Simch-Levi [LMSL92] haben gezeigt, das zugehörige globale Punkt zu Punkt-Problem $\mathcal{NP}$-hart ist. Somit ist die Berechnung einer optimalen Spieler-Strategie auch $\mathcal{NP}$-hart. Wir werden

daher wie bei den Primal-Dual-Algorithmen eine untere Schranke für die Spieler mit Hilfe des folgenden Programms DP(x) konstruieren.

$$\sum_{i,S} f_i(S) y_S^i \to \max \tag{5.7}$$

$$\sum_i x_e^i \leq c_e, \forall e : x_e = 1 \tag{5.8}$$

$$\sum_e x_e^i \leq \alpha \sum_S f_i(S) y_S^i, \forall i \tag{5.9}$$

$$\sum_{S: e \in \delta(S)} y_S^i \leq x_e^i, \forall i, e : x_e = 1 \tag{5.10}$$

$$\sum_{S: e \in \delta(S)} y_S^i \leq c_e, \forall i, e : x_e = 0 \tag{5.11}$$

$$x_e^i, y_S^i \geq 0 \tag{5.12}$$

x_e^i und y_S^i sind die wählbaren Variablen. An die Variablen x_e^i für Kanten e mit $x_e = 0$ ist keine Bedingung gestellt. Wir werden daher ohne Einschränkung der Allgemeinheit $x_e^i = 0$ für $x_e = 0$ annehmen.

Lemma 5.10. *Es sei $(x_e)_{e \in E}$ eine globale Lösung und $\alpha \geq 1$. Für jede Lösung $(x_e^i, y_S^i)_{e \in E, i \in I, S \subseteq V}$ von* DP(x) *ist*

$$\sum_{S \subseteq V} f_i(S) y_S^i$$

eine untere Schranke für die optimalen Kosten des Spielers i in der (unzulässigen) Gesamt-Strategie $(x_e^i)_{e \in E, i \in I}$.

Beweis: Es sei $(x_e^i, y_S^i)_{e \in E, i \in I, S \subseteq V}$ eine Lösung von DP(x). Die Bedingungen (5.4) und (5.5) entsprechen genau den Bedingungen (5.10) und (5.11). Somit ist

$$\sum_{S \subseteq V} f_i(S) y_S^i$$

eine untere Schranke für die optimalen Kosten von Spieler i unter der Gesamt-Strategie $(x_e^i)_{e \in E, i \in I}$. □

Satz 5.11. *Es sei $(x_e)_{e \in E}$ eine globale Lösung und $\alpha \geq 1$.*

1. *Wenn eine Lösung $(x_e^i, y_S^i)_{e \in E, i \in I, S \subseteq V}$ von* DP(x) *die Bedingung*

$$\sum_{e \in E} c_e x_e \leq \alpha \sum_{i \in I, S \subseteq V} f_i(S) y_S^i$$

erfüllt, dann ist $(x_e)_{e \in E}$ ein α-approximatives Nash-Gleichgewicht.

2. *Im Punkt zu Punkt-Spiel ist $x = (x_e)_{e \in E}$ genau dann ein α-approximatives Nash-Gleichgewicht, wenn die optimale Lösung $(x_e^i, y_S^i)_{e \in E, i \in I, S \subseteq V}$ von* DP(x) *die Bedingung*

$$\sum_{e \in E} c_e x_e \leq \alpha \sum_{i \in I, S \subseteq V} f_i(S) y_S^i$$

erfüllt.

Beweis: 1. Es sei $(x_e^i, y_S^i)_{e,S,i}$ eine Lösung von DP(x). Es bezeichne $(\overline{x}_e^i)_{e \in E}$ die optimale Strategie des Spielers i bei den gegebenen Spieler-Strategien $(x_e^j)_{j \neq i}$. Nach Lemma 5.10 gilt

$$\sum_{e \in E} \overline{x}_e^i \geq \sum_{S \subseteq V} f_i(S) y_S^i.$$

Mit Bedingung (5.9) folgt

$$\sum_{e \in E} x_e^i \leq \alpha \sum_{S \subseteq V} f_i(S) y_S^i \leq \alpha \sum_{e \in E} \overline{x}_e^i.$$

$(x_e^i)_{e\in E}$ ist also ein α-approximatives Nash-Gleichgewicht für den Spieler i. Das Problem ist, dass Kanten e mit $x_e = 1$ existieren, die von $(x_e^i)_{e\in E,i\in I}$ nicht vollständig gekauft werden. Wir werden $(x_e^i)_{e\in E,i\in I}$ zu einem α-approximativen Nash-Gleichgewicht erweitern. Jeder Spieler i kann noch

$$\alpha \sum_{e\in E} \overline{x}_e^i - \sum_{e\in E} x_e^i \geq \alpha \sum_{S\subseteq V} f_i(S) y_S^i - \sum_{e\in E} x_e^i$$

in die nicht vollständigen Kanten investieren. Wenn jeder Spieler $\alpha \sum_{S\subseteq V} f_i(S) y_S^i - \sum_{e\in E} x_e^i$ investiert, bleibt die Bedingung (5.9) (und auch die anderen Bedingungen) erhalten. Die Summe über alle Spieler i ist nach Voraussetzung

$$\sum_{i\in I}(\sum_{S\subseteq V} \alpha f_i(S) y_S^i - \sum_{e\in E} x_e^i) \geq \alpha \sum_{i\in I, S\subseteq V} f_i(S) y_S^i - \sum_{e\in E, i\in I} x_e^i \geq \sum_{e\in E} c_e x_e - \sum_{e\in E, i\in I} x_e^i.$$

Somit können alle Kanten e mit $x_e = 1$ bezahlt werden.

2. Wir zeigen, dass für jedes α-approximative Nash-Gleichgewicht $(x_e^i)_{e\in E,i\in I}$ eine Lösung von DP(x) mit

$$\sum_{e\in E} c_e x_e \leq \alpha \sum_{i\in I, S\subseteq V} f_i(S) y_S^i$$

existiert. Ohne Einschränkung der Allgemeinheit ist $(x_e^i)_{e\in E,i\in I}$ minimal. Die Bedingung (5.8) ist von $(x_e^i)_{e\in E,i\in I}$ erfüllt. Es bezeichne $(\overline{x}_e^i)_{e\in E}$ die optimale Strategie des Spielers i bei den gegebenen Spieler-Strategien $(x_e^j)_{j\neq i}$. Da $(x_e^i)_{e\in E,i\in I}$ minimal ist, ist

$$\sum_{e\in E} \overline{x}_e^i = \sum_{e:x_e=1} x_e^i \alpha_e^i + \sum_{e:x_e=0} c_e \alpha_e^i$$

für die optimale Lösung $(\alpha_e^i)_{e\in E}$ von IP(i). Im Punkt zu Punkt-Spiel ist opt(IP(i)) = opt(DP(i)). Es existiert also eine optimale Lösung $(y_S^i)_{S\subseteq V}$ von DP(i) mit

$$\sum_{e\in E} \overline{x}_e^i = \sum_{e:x_e=1} x_e^i \alpha_e^i + \sum_{e:x_e=0} c_e \alpha_e^i = \sum_{S\subseteq V} f_i(S) y_S^i.$$

Die Bedingungen (5.10) und (5.11) sind von $(x_e^i, y_S^i)_{e\in E,i\in I,S\subseteq V}$ erfüllt. Da $(x_e^i)_{e\in E,i\in I}$ ein α-approximatives Nash-Gleichgewicht ist,ist für alle i

$$\sum_{e\in E} x_e^i \leq \alpha \sum_{e\in E} \overline{x}_e^i = \alpha \sum_{S\subseteq V} f_i(S) y_S^i,$$

und die Bedingung (5.9) ist auch erfüllt. Es ist

$$\sum_{e\in E} c_e x_e = \sum_{e\in E, i\in I} x_e^i \leq \sum_{i\in I} \alpha \sum_{S\subseteq V} y_S^i = \alpha \sum_{i\in I, S\subseteq V} f_i(S) y_S^i.$$

Die andere Richtung haben wir im ersten Teil gezeigt.

□

Das duale Programm zu DP(x) lautet

$$\sum_{e:x_e=1} c_e \alpha_e + \sum_{i,e:x_e=0} c_e \delta_e^i \to \min \tag{5.13}$$

$$\sum_{e\in\delta(S):x_e=1} \gamma_e^i + \sum_{e\in\delta(S):x_e=0} \delta_e^i \geq f_i(S) + \alpha f_i(S)\beta^i, \forall i, S \tag{5.14}$$

$$\alpha_e + \beta^i \geq \gamma_e^i, \forall i, e : x_e = 1 \tag{5.15}$$

$$\alpha_e, \beta^i, \gamma_e^i, \delta_e^i \geq 0 \tag{5.16}$$

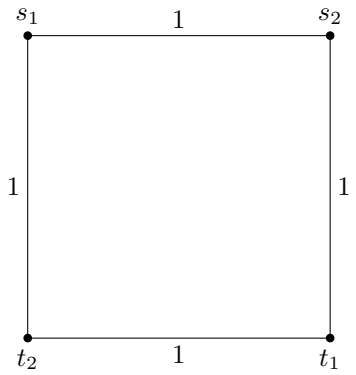
Wir nennen es LP(x). Die Variablen α_e, β^i, γ_e^i, δ_e^i sind freiwählbar.

Lemma 5.12. *Jede zulässige Lösung x von* CP(f) *ist eine zulässige Lösung von* LP(x) *mit dem gleichen Wert.*

Beweis: Es sei $x = (x_e)_{e \in E}$ eine Lösung von CP(f). Wir setzen $\alpha_e := \gamma_e^i := 1$ für $i \in I$ und $e \in E$ mit $x_e = 1$. Weiterhin sei $\beta^i := 0$ für alle $i \in I$ und $\delta_e^i := 0$ für alle $i \in I$ und $e \in E$ mit $x_e = 0$. □

Das Beispiel 5.2 war ein Beispiel für das Punkt zu Punkt-Spiel. Wir werden es noch einmal genauer betrachten.

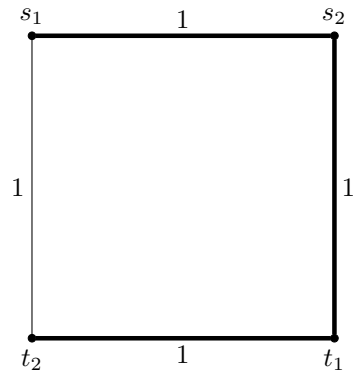
Beispiel 5.13. *Wir betrachten das folgende Punkt zu Punkt-Spiel*



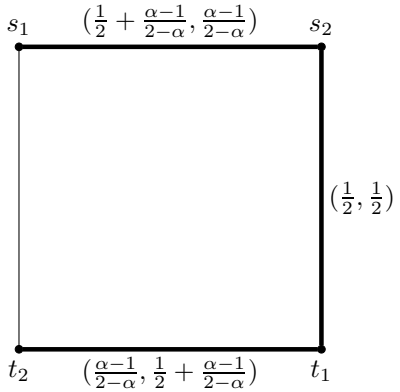
Der Spieler 1 *besitzt die Terminalmenge* $T_1 = \{s_1, t_1\}$ *und der Spieler* 2 *besitzt die Terminalmenge* $T_2 = \{s_2, t_2\}$.
Dieses Punkt zu Punkt-Spiel ist kein α*-approximatives Nash-Gleichgewicht für* $\alpha < \frac{6}{5} = 1.2$. *Für* $\alpha \geq \frac{6}{5}$ *ist es ein* α*-approximatives Nash-Gleichgewicht.*

Beweis:

1. Der Algorithmus PDg in Abbildung 2.4 setzt alle minimalen verletzten dualen Variablen auf $y_{s_1} = y_{s_2} = y_{t_1} = y_{t_2} = \frac{1}{2}$. Nun sind alle Kanten dicht und der Algorithmus 2.4 wählt eine Lösung $x = (x_e)_{e \in E}$ mit drei Kanten. Dies ist die optimale Lösung. Die Kosten betragen opt = 3 (und die untere Schrank ist $\sum_{S \subseteq V} f(S) y_S = 2$). Ohne Einschränkung der Allgemeinheit sei es die folgende Lösung OPT.



   Es sei $\alpha \leq \frac{6}{5}$. Wir wollen Satz 5.11 anwenden und werden die optimale Lösung von DP(OPT) und LP(OPT) angeben. (x_e^1, x_e^2) sind im folgenden Schaubild eingetragen.



Es ist $\frac{1}{2} + \frac{\alpha-1}{2-\alpha} + \frac{\alpha-1}{2-\alpha} \leq 1$ und die Bedingung (5.8) ist erfüllt. Die Kosten der Spieler i, $i = 1, 2$ betragen

$$\sum_{e \in E} x_e^i = 1 + 2\frac{\alpha - 1}{2 - \alpha} = \frac{\alpha}{2 - \alpha}.$$

Der Spieler 1 hat die untere Schranke

$$y^1_{\{s_1\}} = \frac{1}{2} + \frac{\alpha - 1}{2 - \alpha}, y^1_{\{s_1,s_2\}} = \frac{1}{2} - \frac{\alpha - 1}{2 - \alpha} > 0, y^1_{\{t_1\}} = \frac{\alpha - 1}{2 - \alpha}$$

für seine optimalen Kosten mit dem Wert

$$\sum_{S \subseteq V} f_1(S) y^1_S = 1 + \frac{\alpha - 1}{2 - \alpha} = \frac{1}{2 - \alpha}.$$

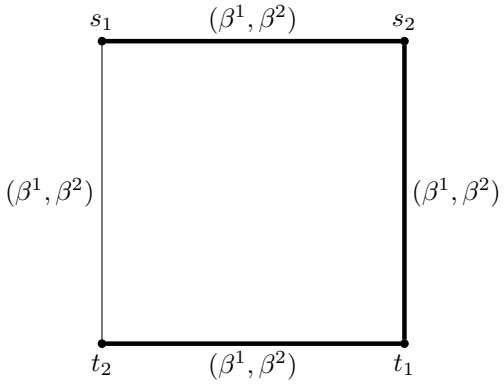
Der Spieler 2 hat eine analoge untere Schranke mit dem gleichen Wert. $(y^i_S)_{i \in I, S \subseteq V}$ erfüllen die Bedingung (5.10) und (5.11). Es ist

$$\sum_{e \in E} x_e^i = \frac{\alpha}{2 - \alpha} = \alpha f_i(S) y^1_S$$

und die Bedingung (5.9) ist somit erfüllt. Der Wert von DP(OPT) beträgt

$$\sum_{i \in I, S \subseteq V} f_i(S) y^i_S = \frac{2}{2 - \alpha}.$$

Wir geben nun die optimale Lösung von LP(OPT) an. Es ist $\alpha_e = 0$ für alle $e \in E$ und $\beta^i =$ für $i = 1, 2$. Das folgende Schaubild zeigt (γ^1_e, γ^2_e) für $\mathrm{OPT}_e = 1$ bzw. (δ^1_e, δ^2_e) für $\mathrm{OPT}_e = 0$.



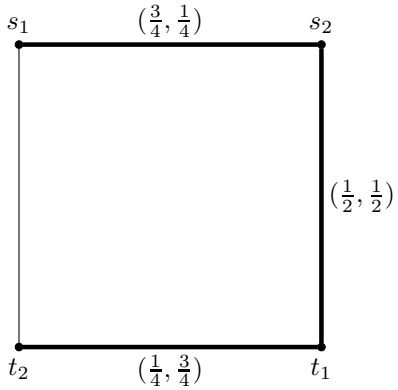
Es ist $2\beta^i = 1 + \alpha\beta^i$ und die Bedingung (5.15) ist erfüllt. Da $\alpha \geq 1$ ist, ist auch die Bedingung (5.14) erfüllt. Der Wert von LP(OPT) ist

$$2\beta^i = \frac{2}{2 - \alpha}.$$

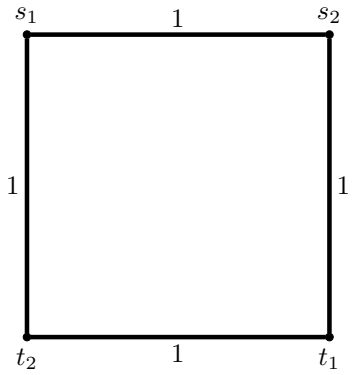
Die beiden angegebenen Lösungen haben den gleichen Wert und sind somit optimal. OPT ist nach Satz 5.11 genau dann ein α-approximatives Nash-Gleichgewicht, wenn

$$\alpha \sum_{i,S} f_i(S) y_S^i \geq \text{opt} \Leftrightarrow \frac{2\alpha}{2-\alpha} \geq 3 \Leftrightarrow \alpha \geq \frac{6}{5}$$

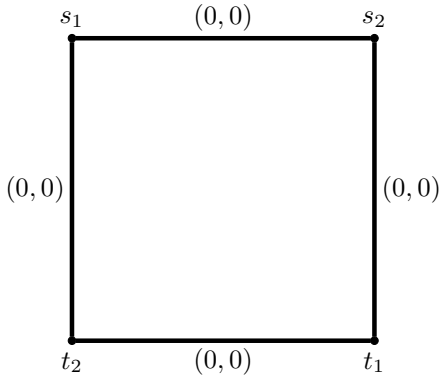
ist. Der folgende Graph gibt das $\frac{6}{5}$-approximative Nash-Gleichgewicht an.



2. Die andere zulässige globale Lösung x besteht aus allen vier Kanten. Der Algorithmus von Jain in Abbildung 2.6 berechnet zum Beispiel diese Lösung.



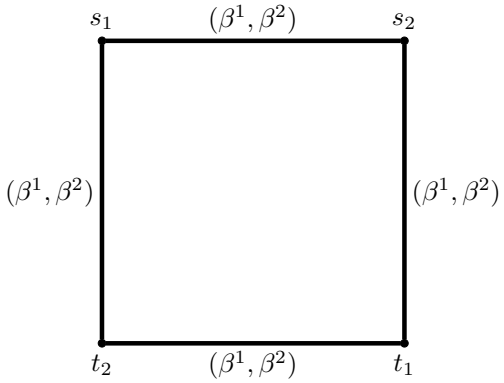
Es sei $\alpha < 2$. Das folgende Schaubild zeigt die optimale Lösung (x_e^1, x_e^2) von DP(x).



Es sind alle $y_S^i = 0$ und der Wert von DP(x) ist

$$\sum_{i \in I, S \subseteq V} y_S^i = 0$$

Die optimale Lösung von LP(x) ist $\beta^i = \frac{1}{2-\alpha}$ und die folgenden (γ_e^1, γ_e^2) im nächsten Schaubild



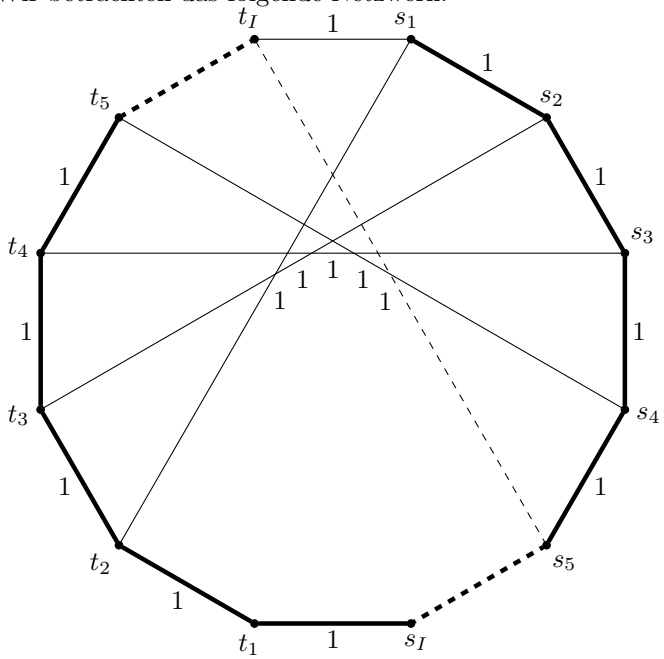
Der Wert von LP(x) beträgt auch 0 und die beiden Lösungen sind somit optimal. Es ist leicht zu sehen, dass dieses Spiel ein 2-approximatives Nash-Gleichgewicht ist.

□

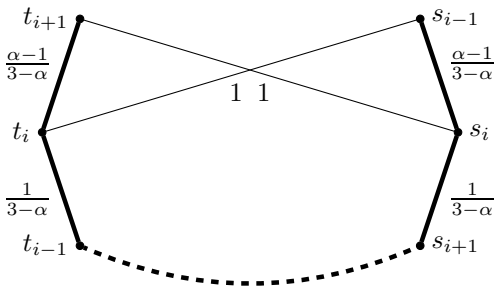
Wir haben im obigen Beispiel gesehen, dass wenn OPT ein α-approximatives Nash-Gleichgewicht ist, dann folgt $\alpha \geq 1.2$. Wir werden diese Schranke im folgenden Lemma verbessern.

Lemma 5.14. *Im Allgemeinen ist die globale optimale Lösung OPT im Punkt zu Punkt-Spiel kein α-approximatives Nash-Gleichgewicht für $\alpha < 1.5$.*

Beweis: Es sei $\alpha \leq \frac{3}{2}$. Wir betrachten das folgende Netzwerk.



Der Graph besteht aus einem Zyklus $(s_1, \ldots, s_I, t_1, \ldots, t_I, s_1)$ und den Kanten $\{s_i, t_{i+1}\}$ für $i = 1, \ldots, I-1$. Jede Kante hat Kosten von 1. Jeder Spieler $i \in \{1, \ldots, I\}$ möchte seine beiden Terminals s_i und t_i verbinden. Die dicken Kanten markieren eine optimale Lösung OPT mit opt $= 2I - 1$. Für einen Spieler $1 < i < L$ sind die optimalen Variablen x_e^i von DP(OPT) im nächsten Schaubild angegeben.



Es ist $\frac{1}{3-\alpha} + \frac{\alpha-1}{3-\alpha} = \frac{\alpha}{3-\alpha} \leq 1$ und die Bedingung (5.8) ist erfüllt. Die optimalen Variablen y_S^i sind

$$y^i_{\{s_i\}} = y^i_{\{t_i\}} = \frac{\alpha-1}{3-\alpha}, y^i_{\{s_i,\ldots,s_1\}} = y^i_{\{t_i,\ldots,t_L\}} = \frac{1}{3-\alpha} - \frac{\alpha-1}{3-\alpha} = \frac{2-\alpha}{3-\alpha}.$$

Die Bedingungen (5.10) und (5.11) sind erfüllt Der Wert für Spieler i beträgt

$$\sum_{S\subseteq V} f_i(S) y_S^i = 1 + \frac{\alpha-1}{3-\alpha} = \frac{2}{3-\alpha}.$$

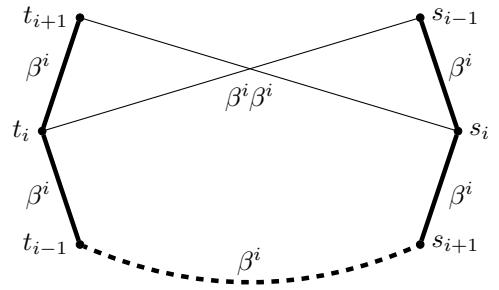
Es ist

$$\sum_{e\in E} x_e^i = \frac{2\alpha}{3-\alpha} = \alpha \sum_{S\subseteq V} y_S^i$$

und die Bedingung (5.9) ist erfüllt. Der Wert von DP(OPT) beträgt

$$\sum_{i,S} f_i(S) y_S^i = \frac{2(I-2)}{3-\alpha} + \sum_S f_1(S) y_S^1 + \sum_S f_I(S) y_S^L.$$

Das folgende Schaubild zeigt die optimalen (γ_e^1, γ_e^2) für $\mathrm{OPT}_e = 1$ bzw. (δ_e^1, δ_e^2) für $\mathrm{OPT}_e = 0$ von LP(OPT) für den Spieler i. Es ist $\alpha_e = 0$ für $e \in E$ und $\beta^i = \frac{1}{3-\alpha}$ für $i = 2, \ldots, I-1$.



Die Bedingung (5.15) ist erfüllt. Der Spieler i benutzt drei kanten-disjunkte Weg jeweils zu β^i. Es ist $3\beta^i = 1 + \frac{2}{3-\alpha}$ und die Bedingung (5.14) ist erfüllt. Spieler i steuert $2\beta^i = \frac{2}{3-\alpha}$ bei. Der Wert von LP(OPT) beträgt somit

$$\sum_{e:x_e=1} c_e\alpha_e + \sum_{i,e:x_e=0} c_e\delta_e^i = \frac{2(I-2)}{3-\alpha} + \text{Kosten der Spieler 1 und } I$$

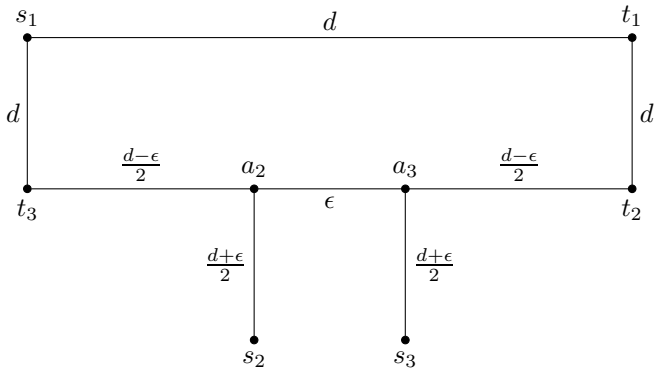
Für große I sind die Kosten des Spielers 1 und I vernachlässigbar und wir haben eine optimale Lösung von DP(OPT) gefunden. Das Spiel ist nach Satz 5.11 genau dann ein α-approximatives Nash-Gleichgewicht, wenn

$$\alpha \geq \frac{\text{opt}}{\text{opt(DP(OPT))}} = \frac{2I-1}{\frac{2I-2}{3-\alpha} + f_1(S)y_S^1 + f_I(S)y_S^I} \to 3-\alpha \text{ für } I \to \infty$$

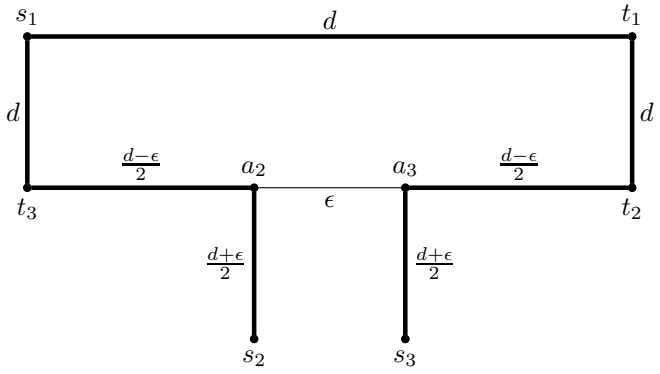
ist. Dies ist genau dann der Fall, wenn $\alpha \geq \frac{3}{2} = 1.5$ ist. □

[ADTW03] haben mit einem Graphen, der mehr Kanten hat, das gleiche Resultat gezeigt. Wir betrachten nochmal das Beispiel 2.54.

Beispiel 5.15. *Es sei der folgende Graph gegeben.*



Jeder Spieler i, $i = 1, 2, 3$ möchte seine Terminals $\{s_i, t_i\}$ verbinden. a_1 und a_2 sind Steiner-Knoten. Der Algorithmus PDg *berechnet in Beispiel 2.54 eine* 2*-Approximation, die im folgenden Schaubild durch die dicken Kanten dargestellt ist.*

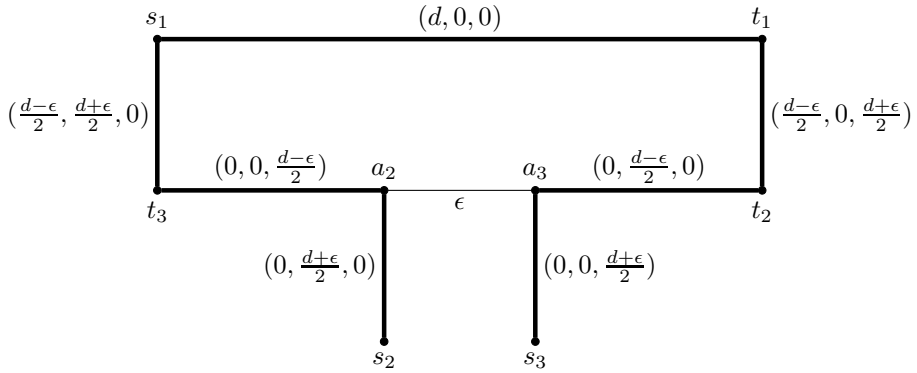


Es ist

$$\text{pdg} = 5d \text{ und } \text{opt} = 3d + \epsilon.$$

Diese Approximation ist ein 2*-approximatives Nash-Gleichgewicht.*

Beweis: Es sei $\alpha \geq 2$. Die optimalen Variablen (x_e^1, x_e^2, x_e^3) von DP(PDg) sind im folgenden Schaublid dargestellt.



Die Bedingung (5.8) ist erfüllt. Es ist

$$\sum_{e \in E} x_e^1 = 2d - \epsilon \text{ und } \sum_{e \in E} x_e^2 = \sum_{e \in E} x_e^3 = \frac{3}{2}d + \frac{\epsilon}{2}.$$

Die optimalen Variablen y^1_S des Spielers 1 sind

$$y^1_{\{s_1\}} = y^1_{\{t_1\}} = \frac{d-\epsilon}{2},\ y^1_{\{s_1,t_3,a_2,s_2\}} = \epsilon.$$

Die optimalen Variablen y^i_S der Spieler $i = 2, 3$ betragen

$$y^2_{\{s_2\}} = \frac{d+\epsilon}{2},\ y^2_{\{s_2,a_2,t_3\}} = \epsilon,\ y^2_{\{s_2,a_2,t_3,a_3,s_3\}} = \frac{d-\epsilon}{2} \text{ und}$$
$$y^3_{\{s_3\}} = \frac{d+\epsilon}{2},\ y^3_{\{s_3,a_3,t_2\}} = \epsilon,\ y^3_{\{s_3,a_3,t_2,a_2,s_2\}} = \frac{d-\epsilon}{2}.$$

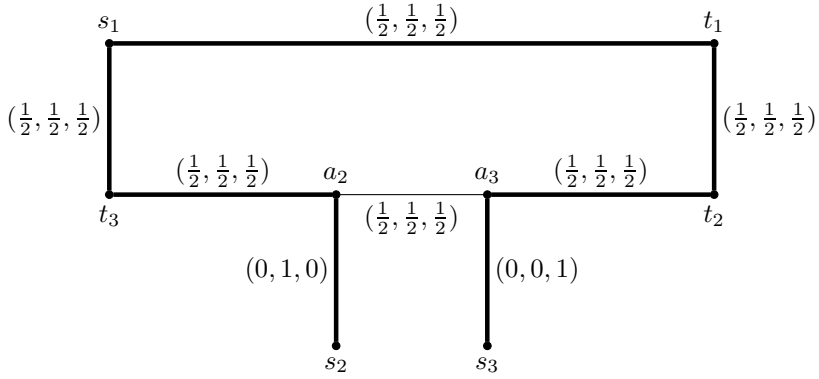
Die Variablen erfüllen die Bedingungen (5.10) und (5.11). Deren Wert beträgt

$$\sum_{S\subseteq V} f_1(S)y^1_S = d,\ \sum_{S\subseteq V} f_2(S)y^2_S = \sum_{S\subseteq V} f_3(S)y^3_S = d+\epsilon$$

und die Bedingung (5.9) ist erfüllt. Der Wert der Lösung von DP(PDg) beträgt

$$\sum_{i\in I, S\subseteq V} f_i(S)y^i_S = 3d + 2\epsilon.$$

Die optimale Variablen $(\gamma^1_e, \gamma^2_e, \gamma^3_e)$ bzw. $(\delta^1_e, \delta^2_e, \delta^3_e)$ von LP(PDg) sind im folgenden Schaubild angegeben.



Es ist $\beta^i = 0$, $i = 1, 2, 3$ und $\alpha_e := \max_{i=1,2,3} \gamma^i_e$. Die Bedingungen (5.14) und (5.15) sind erfüllt. Der Wert der Lösung von LP(PDg) ist

$$\sum_{e:x_e=0} c_e\alpha_e + \sum_{i,e:x_e=0} c_e\delta^i_e = 3d + 2\epsilon.$$

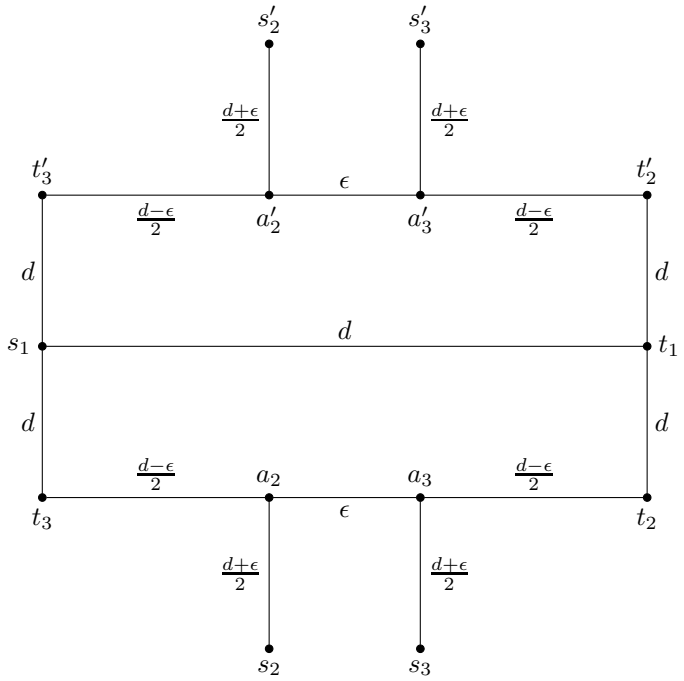
Da beide Werte der Lösungen LP(PDg) und DP(PDg) gleich sind, haben wir optimale Lösungen gefunden. Es ist

$$2\,\mathrm{opt}(\mathrm{DP}(\mathrm{PDg})) = 2(3d + 2\epsilon) \geq 5d = \mathrm{opt}$$

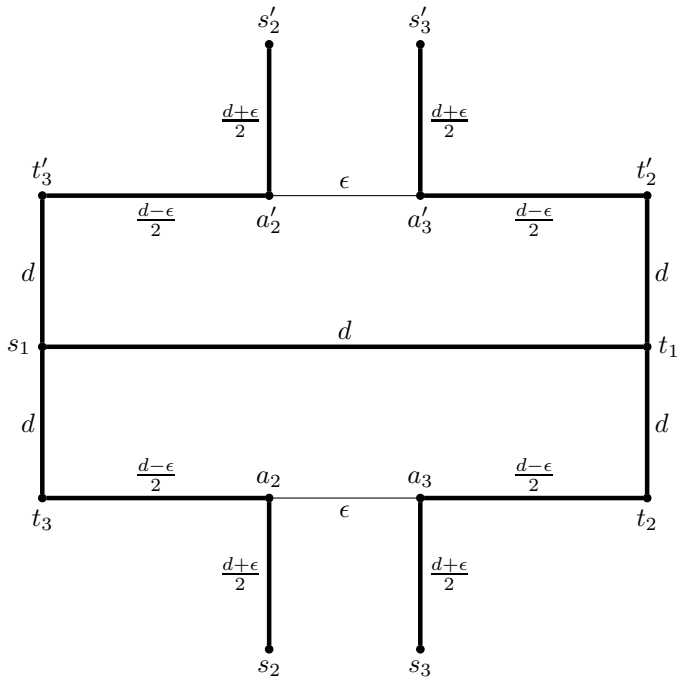
und wir haben nach Satz 5.11 ein 2-approximatives Nash-Gleichgewicht gefunden. □

In diesem Beispiel hatte der Spieler 1 zwei kantendisjunkte Wege, die die beiden Knoten s_1 und t_1 verbindet. Im nächsten Beispiel werden wir die Anzahl der Wege des Spielers 1 erhöhen, damit der Spieler nicht mehr in alle Kanten dieser Wege investieren kann. Dafür kopieren wir die Spieler 2 und 3.

Beispiel 5.16. *Wir betrachten das folgende Netzwerk.*



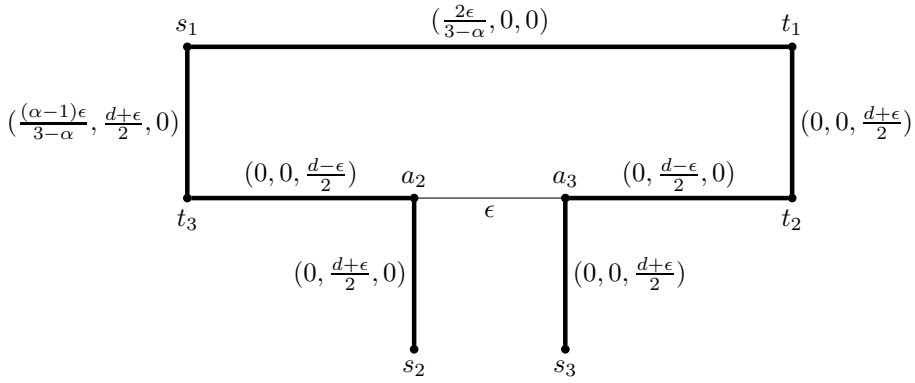
Die Spieler $i = 1, 2, 3$ möchten $\{s_i, t_i\}$ verbinden und die Spieler $2'$ und $3'$ möchten $\{s'_2, t'_2\}$ bzw. $\{s'_3, t'_3\}$ verbinden. Der Algorithmus PDg *berechnet wie in Beispiel 2.54 ohne Einschränkung der Allgemeinheit die folgende durch dicke Kanten dargestellte globale 2-Approximation.*



Es ist

$$\text{pdg} = 9d \text{ und } \text{opt} = 5d + 2\epsilon.$$

Es sei $\alpha \geq 2$. (Für kleinere α ist der Wert von DP(PDg) *natürlich auch kleiner.) Das folgende Schaubild zeigt die optimalen Variablen (x_e^1, x_e^2, x_e^3) (und analog die optimalen Variablen $(x_e^1, x_e^{2'}, x_e^{3'})$) von* DP(PDg).



In diesem Beispiel lassen wir den Spieler 1 *nicht in die Kante* (t_1, t_2) *investieren. Die Bedingung (5.8) ist erfüllt. Es ist*

$$\sum_{e\in E} x_e^1 = \frac{2\epsilon}{3-\alpha} + 2\frac{(\alpha-1)\epsilon}{3-\alpha} = \frac{2\alpha\epsilon}{3-\alpha} \text{ und } \sum_{e\in E} x_e^2 = \sum_{e\in E} x_e^3 = \frac{3}{2}d + \frac{\epsilon}{2}.$$

Die optimalen Variablen y_S^1 *des Spielers* 1 *sind*

$$y_{\{s_1\}}^1 = \frac{(\alpha-1)\epsilon}{3-\alpha} + \epsilon = \frac{2\epsilon}{3-\alpha}$$

Die optimalen Variablen y_S^i *der Spieler* $i = 2, 3$ *bzw.* $2', 3'$ *betragen*

$$y_{\{s_2\}}^2 = \frac{d+\epsilon}{2},\ y_{\{s_2,a_2,t_3\}}^2 = \epsilon,\ y_{\{s_2,a_2,t_3,a_3,s_3\}}^2 = \frac{d-\epsilon}{2} \text{ und}$$
$$y_{\{s_3\}}^3 = \frac{d+\epsilon}{2},\ y_{\{s_3,a_3,t_2\}}^3 = \epsilon,\ y_{\{s_3,a_3,t_2,a_2,s_2\}}^3 = \frac{d-\epsilon}{2}.$$

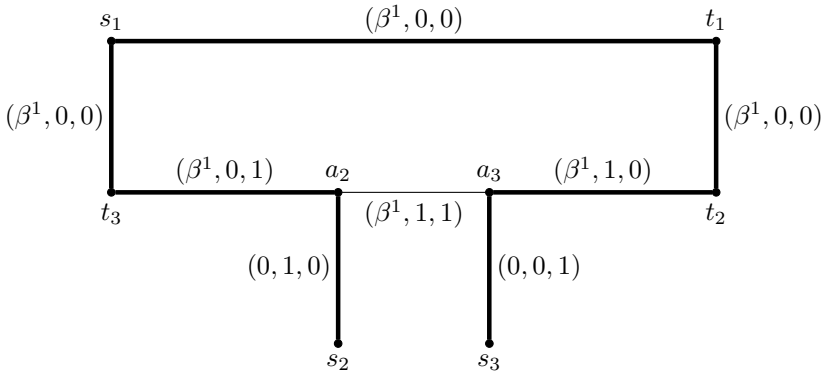
Die Bedingungen (5.10) und (5.11) sind erfüllt. Es ist

$$\sum_{S\subseteq V} f_1(S)y_S^1 = \frac{2\epsilon}{3-\alpha},\ \sum_{S\subseteq V} f_2(S)y_S^2 = \sum_{S\subseteq V} f_3(S)y_S^3 = d + \epsilon.$$

Die Bedingung (5.9) ist erfüllt. Der Wert dieser Lösung von DP(PDg) *beträgt*

$$\sum_{i\in I, S\subseteq V} f_i(S)y_S^i = \frac{2\epsilon}{3-\alpha} + 4(d+\epsilon) = 4d + 4\epsilon + \frac{2\epsilon}{3-\alpha}.$$

Die optimale Lösung von LP(PDg) *ist* $\beta^1 = \frac{1}{3-\alpha}$ *und* $(\gamma_e^1, \gamma_e^2, \gamma_e^3)$ *bzw.* $(\delta_e^1, \delta_e^2, \delta_e^3)$*, welche im folgenden Schaubild für die Spieler* $i = 1, 2, 3$ *(und analog* $1, 2', 3'$*) angegeben sind.*



Es ist $\alpha_e = 1$ *für die Kanten* (t_3, a_2), (a_3, t_2), (a_2, s_2) *und* (a_3, s_3). *Die restlichen Variablen sind* 0. *Die Bedingung (5.15) ist erfüllt. Da* $3\beta^1 = 1 + \alpha\beta^1$ *ist, ist auch die Bedingung (5.14) erfüllt. Der Wert der Lösung von* LP(PDg) *beträgt*

$$\sum_e c_e \alpha_e + \sum_{i,S} c_e \delta_e^i = 4(d + \epsilon) + 2\beta^1 \epsilon.$$

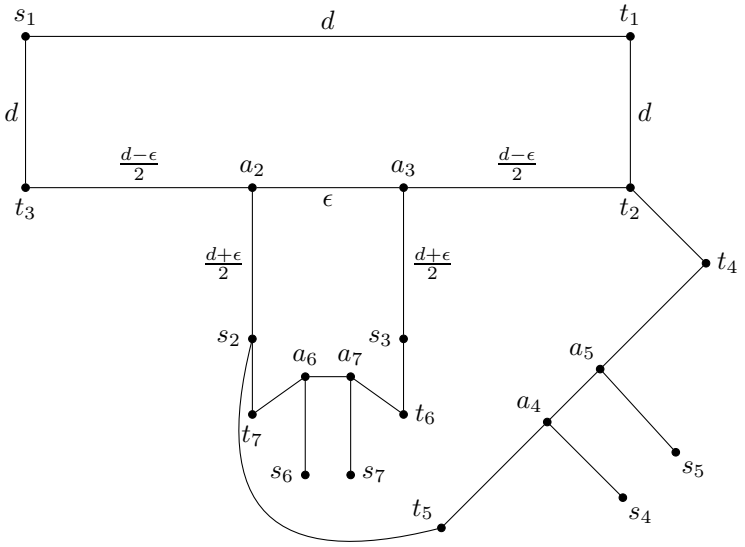
Da die beiden Werte gleich sind, haben wir optimale Lösungen von DP(PDg) *und* LP(PDg) *gefunden. Dieses Spiel ist nach Satz 5.11 genau dann ein* α*-approximatives Nash-Gleichgewicht, wenn*

$$\text{pdg} = \sum_{e \in E} c_e x_e \leq \alpha \sum_{i \in I, S \subseteq} f_i(S) y_S^i \Leftrightarrow \alpha \geq \frac{9d}{4d + 4\epsilon + \frac{2\epsilon}{3-\alpha}} \to \frac{9}{4} = 2.25 \text{ für } \epsilon \to 0$$

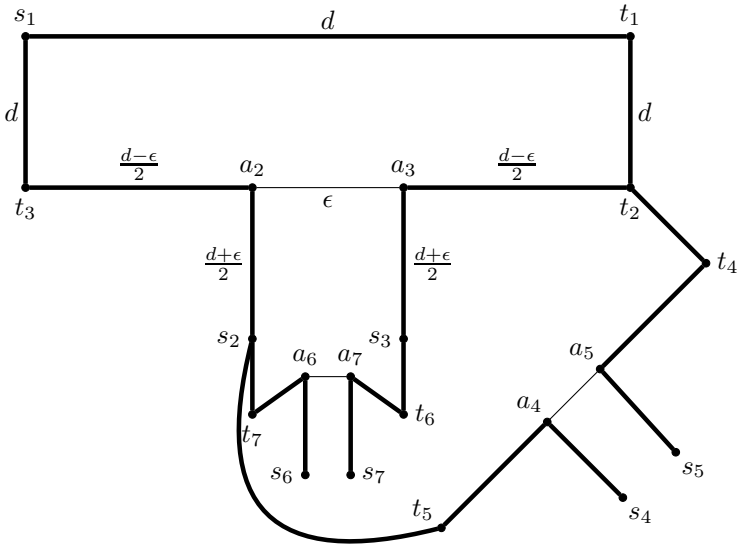
ist.

In dem Beispiel haben wir gesehen, dass der Algorithmus PDg im Allgemeinen kein α-approximatives Nash-Gleichgewicht für $\alpha < 2.25$ berechnet. Die Spieler 2 und 3 (und $2'$ und $3'$) haben in diesem Spiel einen kantendisjunkten Weg benutzt, welcher deren Terminals verbindet. Wir werden im nächsten Beispiel diese Anzahl durch Hinzufügen von sechs Spielern auf drei erhöhen.

Beispiel 5.17. *Wir betrachten den folgenden Graphen.*



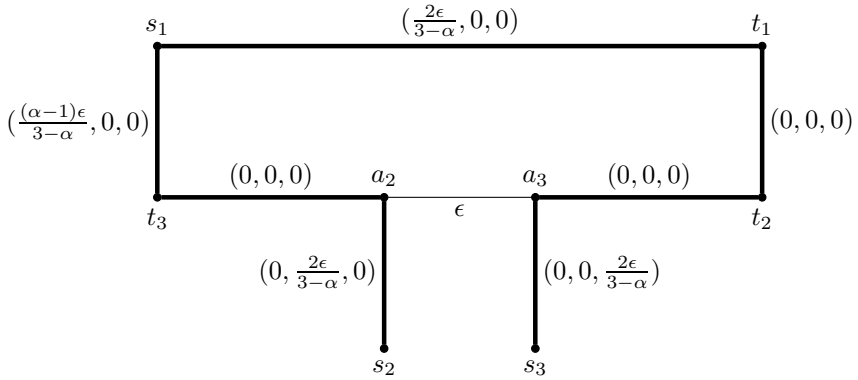
Wir haben die Spieler 8 *und* 9, *die die Knoten* t_3 *und* s_3 *verbinden, im Schaubild weggelassen. Sie sind analog zu den Spielern* 4 *und* 5. *Für die Spieler* $2'$ *und* $3'$ *werden analog neue Spieler hinzugefügt. Jeder Spieler* $i = 1, \ldots, 7$ *im obigen Netzwerk möchte seine Terminals* $\{s_i, t_i\}$ *verbinden. Der Algorithmus* PDg *berechnet nach Beispiel 2.54 ohne Einschränkung der Allgemeinheit die folgende durch dicke Kanten gekennzeichnete* 2*-Approximation.*



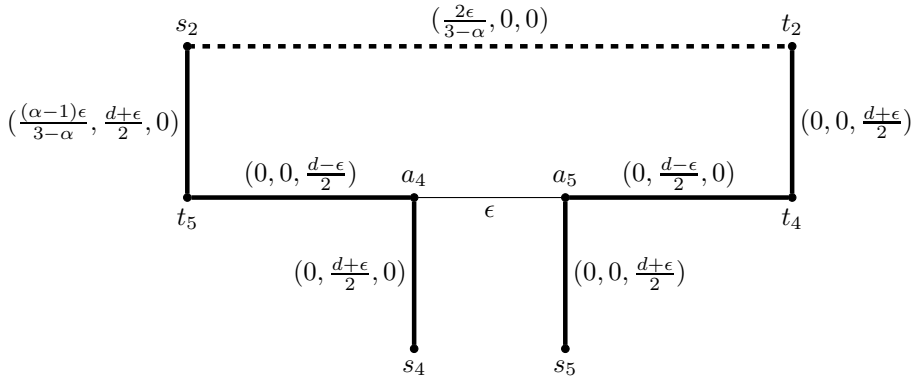
Es ist

$$\text{pdg} = 9d + 6 \cdot 4d = 33d \text{ und } \text{opt} = 17d + 8\epsilon.$$

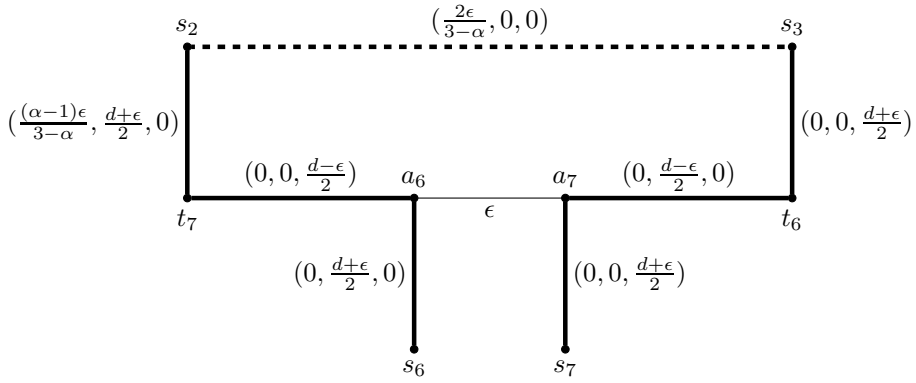
In den nächsten Schaubildern werden wir die optimalen Variablen $(x_e^i)_{e\in E, i\in I}$ *von* DP(PDg) *angeben. Dabei werden wir immer nur Teile des Netzwerkes darstellen. Weiterhin werden wir nicht den vollständigen Vektor* $(x_e^i)_{e\in E, i\in I}$*, sondern nur die* x_e^i *der interessanten Spieler angeben. Die optimalen Variablen* $(x_e^i)_{e\in E}$ *der Spieler* $(1,2,3)$ *sind im folgenden Schaubild angezeigt.*



Die optimalen Variablen $(x_e^i)_{e\in E}$ *der Spieler* $(2,4,5)$ *befinden sich im folgenden Schaubild.*



Die Linie (s_2, t_2) *bezeichnet den Weg* $(s_2, \ldots, t_2)$ *in* PDg*. Für die Spieler* $(2,4,5)$ *sind die optimalen Variablen* $(x_e^i)_{e\in E}$ *im nächsten Schaubild.*



Die Linie (s_2, s_3) *bezeichnet den Weg* $(s_2, \ldots, s_3)$ *in* PDg. *Es ist*

$$\sum_{e\in E} x_e^1 = \sum_{e\in E} x_e^2 = \sum_{e\in E} x_e^3 = \frac{2\alpha\epsilon}{3-\alpha} \text{ und } \sum_{e\in E} x_e^4 = \frac{3}{2}d + \frac{\epsilon}{2}.$$

Die optimalen $(y_S^i)_{i\in I, S\subseteq V}$ *sind*

$$y^1_{\{s_1\}} = y^2_{\{s_2\}} = y^3_{\{s_3\}} = \frac{(\alpha-1)\epsilon}{3-\alpha} + \epsilon = \frac{2\epsilon}{3-\alpha}$$

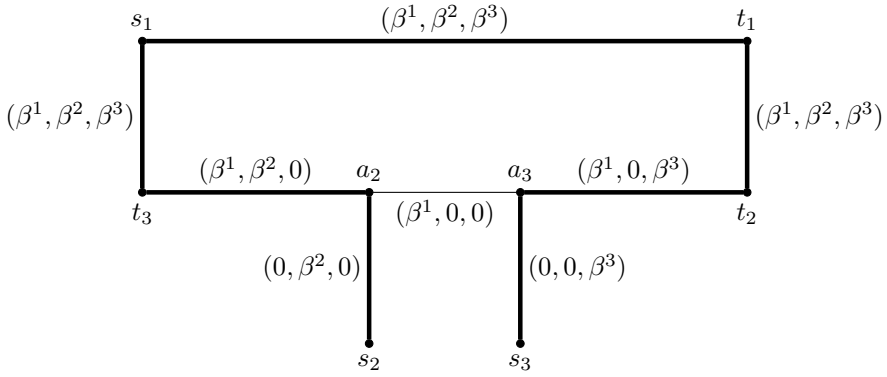
und

$$y^4_{\{s_4\}} = \frac{d+\epsilon}{2}\ y^4_{\{s_4,a_4,t_5\}} = \epsilon\ y^4_{\{s_4,a_4,t_5,a_5,s_5\}} = \frac{d-\epsilon}{2}.$$

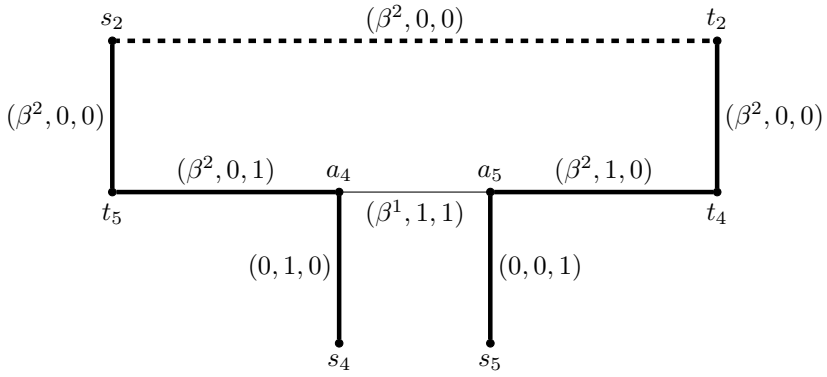
Die Spieler $5, 6, 7$ *haben zum Spieler* 4 *analoge Variablen. Die Bedingungen (5.8), (5.9), (5.10) und (5.11) sind wie im vorherigen Beispiel erfüllt. Es ist* $\sum\limits_{S\subseteq V} y_S^4 = d+\epsilon$ *und der Wert dieser Lösung von* DP(PDg) *ist*

$$\sum_{i\in I, S\subseteq V} f_i(S) y_S^i = 12(d+\epsilon) + 5\frac{2\alpha}{3-\alpha} \to 12d \text{ für } \epsilon \to 0.$$

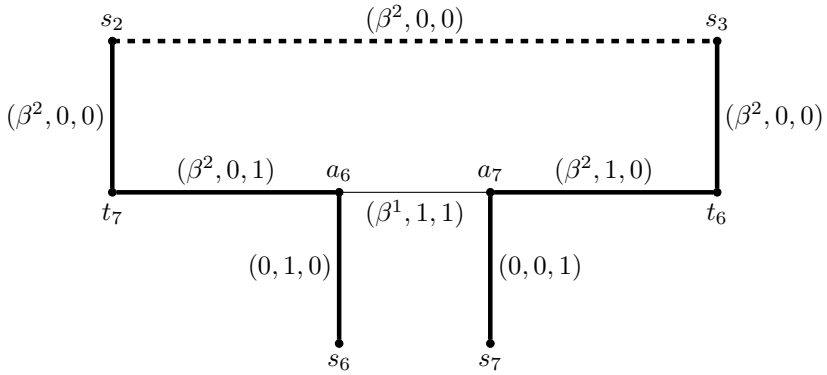
Die optimale Lösung von LP(PDg) *ist* $\beta^1 = \beta^2 = \beta^3 = \frac{1}{3-\alpha}$ *und die folgenden* $(\gamma_e^i)_i$ *bzw.* $(\delta_e^i)_i$ *für die Spieler* $(1, 2, 3)$.



Im folgenden Schaubild sind $(\gamma_e^i)_i$ *bzw.* $(\delta_e^i)_i$ *für die Spieler* $(2, 4, 5)$.



Schließlich sind im folgenden Schaubild $(\gamma_e^i)_i$ *bzw.* $(\delta_e^i)_i$ *für die Spieler* $(2, 6, 7)$.



Es ist $\alpha_e = 1$ *für die Kanten, in denen mindestens ein Spieler eine* 1 *hat. Die Bedingung (5.15) ist erfüllt. Da* $3\beta^i = 1 + \alpha\beta^i$ *ist, ist auch die Bedingung (5.14) erfüllt. Der Wert dieser Lösung von* LP(PDg) *ist*

$$12(d + \epsilon) + 5 \cdot 2\beta^i.$$

Da beide Werte gleich sind, haben wir optimale Lösungen gefunden. Dieses Spiel ist nach Satz 5.11 genau dann ein α*-approximatives Nash-Gleichgewicht, wenn*

$$\text{pdg} = 33d \leq \alpha 12d$$

ist. Dies ist genau dann der Fall, wenn $\alpha \geq \frac{33}{12} = \frac{9}{4} = 2.75$ *ist.*

Wir werden obiges Beispiel iterieren.

Satz 5.18. *Der Algorithmus* PDg *berechnet im Punkt zu Punkt-Spiel kein* α*-approximatives Nash-Gleichgewicht für* $\alpha < 3$.

Beweis: Wir iterieren das obige Beispiel. Für jedes Spielerpaar werden drei weitere Spielerpaare hinzugefügt, so dass der Beitrag zu DP(PDg) des Spielerpaares fast 0 wird. Es bezeichne i_k die Anzahl der neuen Spieler in der Iteration k. Weiterhin sei PDg(k) die Lösung des Algorithmus PDg in der Iteration k und pdg(k) die Kosten von PDg(k). Den Grenzwert für $\epsilon \to 0$ der Kosten der optimalen Lösung von DP(PDg(k)) in der Iteration k bezeichnen wir mit opt(DP(PDg))(k).

1. Es ist
$$\text{pdg}(k + 1) = \text{pdg}(k) + 2di_k,$$
da jedes neue Spielerpaar $4d$ zur globalen Lösung beiträgt.

2. Weiterhin ist
$$\text{opt}(\text{DP}(\text{PDg}))(k + 1) = 3\,\text{pdg}(k) = di_{k+1},$$
da je drei Spielerpaare die Kosten eines Spielerpaares annähernd zu 0 werden lassen und nur die neu Spieler effektiv etwas zum linearen Programm DP(PDg$(k+1)$ beitragen. Der effektive Beitrag beträgt d für jeden neuen Spieler.

Nach Satz 5.11 ist $\mathrm{PDg}(k+1)$ ein α-approximatives Nash-Gleichgewicht, wenn

$$\alpha \geq \frac{\mathrm{pdg}(k+1)}{\mathrm{opt}(\mathrm{DP}(\mathrm{PDg}))(k+1)} = 2 + \frac{\mathrm{pdg}(k)}{di_{k+1}} = 2 + \frac{1}{3}\frac{\mathrm{pdg}(k)}{\mathrm{pdg}(k)}$$

ist. Die Folge $x_{k+1} := \frac{\mathrm{pdg}(k+1)}{\mathrm{opt}(\mathrm{DP}(\mathrm{PDg}))(k+1)} = 2 + \frac{x_k}{3}$ hat den Grenzwert 3. Es folgt die Behauptung □

Wir haben somit eine passende untere Schranke zum Korollar 5.5.

Korollar 5.19. *Für binäre propere Funktionen f berechnet der Algorithmus* PDg *im Allgemeinen kein α-approximatives Nash-Gleichgewicht für $\alpha < 3$.*

Für unkreuzbare Funktion folgt das gleiche Korollar. Die Autoren [ADTW03] haben sich speziell für das Steiner-Wald-Spiel interessiert.

Beispiel 5.20. *Im Steiner-Wald-Spiel ist im Allgemeinen die von* PDg *berechnete Lösung kein α-approximatives Nash-Gleichgewicht für $\alpha < 3$.*

5.3 Offene Probleme

In diesem Abschnitt zählen wir ein paar offene Probleme auf.

- Wir haben für propere (und einige andere) binäre Funktionen f gezeigt, dass jeder globale α-Approximationsalgorithmus des Satzes 2.39 ein Algorithmus zur Berechnung eines $2\alpha - 1$-approximatives Nash-Gleichgewicht impliziert. Für allgemeine propere Funktionen existiert ein kombinatorischer $2\mathcal{H}(f_{\max})$-Approximationsalgorithmus und der nichtkombinatorische 2-Approximationsalgorithmus von Jain. Kann man mit Hilfe dieser globalen Approximationen auch approximative Nash-Gleichgewichte konstruieren?

- Wir haben für binäre propere Funktionen gezeigt, dass der Algorithmus PDg im Allgemeinen kein α-approximatives Nash-Gleichgewicht für $\alpha < 3$ berechnet. Wie sehen untere Schranken für andere Funktionen aus? Insbesondere für nichtbinäre propere und für schwach supermodulare Funktionen?

- In Kapitel drei haben wir einen Gegner für eine große Familie von Approximationsalgorithmen betrachten, die unter anderen die Greedy-Algorithmen enthält. Gibt es einen Gegner für approximative Nash-Gleichgewichte oder zumindest für eine wichtige Familie von approximativen Nash-Gleichgewichten?

Literaturverzeichnis

[AB04] S. Angelopoulos and A. Borodin. The power of priority algorithms for facility location and set cover. *Algorithmica*, 40:271–291, 2004.

[ACG+99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Algorithms*. Springer, 1999.

[ADTW03] E. Anshelevich, A. Dasgupta, E. Tardos, and T. Wexler. Near-optimal network design with selfish agents. *Proc. 35th ACM Symposium on Theory of Computing*, 2003.

[AEED+06] S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty. On nash equilibria for a network creation game. *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 89–98, 2006.

[AKR91] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem in networks. *Proc. 23rd Annual ACM Symposium on Theory of Computing*, pages 134–144, 1991.

[BB88] A. Blokhuis and A.E. Brouwer. Geodetic graphs of diameter two. *Geometriae Dedicata*, 25:527–533, 1988.

[BCM05] A. Borodin, D. Cashman, and A. Magen. How well can primal-dual and local-ratio algorithms perform? *Lecture Notes in Computer Science*, 3580:943–955, 2005.

[BCN89] A.E. Brouwer, A.M. Cohen, and A. Neumaier. *Distance-Regular Graphs*. Springer Verlag, 1989.

[BE81] R. BarYehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.

[Beu82] A. Beutelspacher. *Einführung in die endliche Geometrie I Blockpläne*. B.I. Wissenschaftsverlag, 1982.

[BEY98] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BNR02] A. Borodin, M.N. Nielsen, and C. Rackoff. (incremental) priority algorithms. *13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[BP89] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[Bro95] A.E. Brouwer. Block designs. In R.L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, volume I, chapter 14, pages 693–745. Elsevier, 1995.

[BYBFR04] R. Bar-Yehuda, K. Bendel, A. Freund, and D. Rawitz. Local ratio: A unified framework for approximation algorithms: In memoriam: Shimon Even 1935-2004. *ACM Computing Surveys*, 36(4):422–463, 2004.

[BYR06] R. Bar-Yehuda and D. Rawitz. A tale of two methods. *Lecture Notes in Computer Science*, 3895:196–217, 2006.

[Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 3(4):232–235, 1979.

[DI05a] S. Davis and R. Impagliazzo. Models of Greedy algorithms for graph problems. *Electronic Colloquium on Compunational Complexity*, 120:1–36, 2005.

[DI05b] S. Davis and R. Implagliazzo. Models of Greedy algorithms for graph problems. *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 381–390, 2005.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[Edm67] J. Edmonmds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 69B:125–130, 1967.

[Eil06] S. Eilts. On approximative nash equilibria in survivable network design problems. *Work report*, 2006.

[Fei98] U. Feige. A threshold of ln n for approximating set cover. *Journal of the Acm*, 45(4):634–652, 1998.

[FLM+03] A. Fabrikant, A. Luthra, E. Maneva, C.H. Papadimitriou, and S. Shenker. On a network creation game. *Proc. 22nd Annual ACM Symposium on Principles of Distributed Computing*, pages 347–351, 2003.

[FR04] A. Freund and D. Rawitz. Combinatorial interpretations of dual fitting and primal fitting. *Lecture Notes in Computer Science*, 2909:137–150, 2004.

[GGP+94] M.X. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D.P. Williamson. Improved approximation algorithms for network design problems. *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994.

[GGW98] H.N. Gabow, M.X. Goemans, and D.P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming*, 82:13–40, 1998.

[GW95] M.X. Goemans and D.P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.

[GW97] M.X. Goemans and D.P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D.S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 4, pages 144–191. Pws Publishing, 1997.

[Hoc82] D.S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *Siam Journal on Computing*, 11(3):555–556, 1982.

[Hro03] J. Hromkovic. *Algorithmics for Hard Problems*. Springer, 2003.

[Jai01] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 1:39–60, 2001.

[JMM+03] K. Jain, M Mahdian, E. Markakis, A. Saberi, and V.V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM*, 50(6):795–824, 2003.

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and Sciences*, 9:256–278, 1974.

[Jun94] D. Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI Wissenschaftsverlag, 1994.

[KMB81] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15:141–145, 1981.

[Kru56] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. of the American Mathemtical Society*, 7:48–50, 1956.

[LMSL92] C.L. Li, S.T. McCormick, and D. Simchi-Levi. The point-to-point delivery and connection problems: Complexity and algorithms. *Discrete Applied Mathematics*, 36:267–292, 1992.

[Lov75] L. Lovasz. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[MS78] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Publishing Company, 2. edition, 1978.

[Nic66] T.A.J. Nicholson. Finding the shortest route between two points in a network. *Computer Journal*, 9:275–280, 1966.

[OM87] P. Orponen and H. Mannila. On approximation preserving reductions: Complete problems and robust measures. *Technical Report C*, 28, 1987. Department of Computer Science, University of Helsinki.

[Ore62] O. Ore. *Theory of Graphs*. American Mathematical Society, 1962.

[Ple81] J. Plesnik. A bound for Steiner tree problem in graphs. *Mathematica Slovaca*, 31:155–163, 1981.

[Pri57] R.C. Prim. Shortest connection networks and some generalization. *Bell System Technical Jorunal*, 36:1389–1401, 1957.

[RS84] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. *Proc. 29th Annual ACM Symposium on the Theory of Computing*, 475-484.

[RZ00] G. Robins and A Zelikovsky. Improved Steiner tree approximation in graphs. *Proc. 11th Annual ACM-SIAM Symposium on Algorithms*, pages 770–779, 2000.

[Sca86] R. Scapellato. Geodetic graphs of diameter two and some related structures. *Journal of Combinatorial Theory, Series B*, 41(2):218–229, 1986.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience Publication, 1986.

[Sla97] P. Slavik. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25:237–254, 1997.

[Ste74] J.G. Stemple. Geodetic graphs of diameter 2. *Journal of Combinatorial Theory B*, 17:266–280, 1974.

[Thi01] M. Thimm. On the approximability of the Steiner tree problem. *Lecture Notes in Computer Science*, 2136:678–689, 2001.

[WGMV95] D.P. Williamson, M.X. Goemans, M. Mihail, and V.V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15:435–454, 1995.

[Wil02] D.P. Williamson. The primal-dual method for approximation algorithms. *Mathematical Programming, Series B*, 91(3):447–478, 2002.