Universität zu Köln

Polynomielle Kombinatorische Optimierungsalgorithmen

Mitschrift der Vorlesung von Prof. M. Jünger
WS 2002/2003

Hartmut Wahl
hwahl (at) hwahl (dot) de

Diese Mitschrift wurde von angefertigt, um mir das Lernen des Stoffes zu erleichtern und ist keine offizielle Mitschrift des Lehrstuhls. Sie ist nicht fehlerkorrigiert und sollte mit einer gehörigen Portion Misstrauen genossen werden. Korrekturen und Anregungen bitte an mich.

Ich habe beim Lernen noch einige Fehler entdeckt und korrigiert aber übernehme keine Garantie für Richtigkeit: Wer eine Garantie will kaufe sich einen Toaster!

Wer kleine Korrekurvorschläge hat, kann sie mir gerne schicken. Wenn jemand noch größere Änderungen hat (z.B. ein Stichwortverzeichnis wäre fein) kann ich auch gerne den LaTeX–Code und die Grafiken zur Verfügung stellen. Allerding ist dies mein erstes Dokument mit solchen Umfang in LaTeX, der Code mag einem Profi etwas unbeholfen und umständlich vorkommen.

Literatur:
Prof. Jünger wies auf dem Semesterapparat hin, der sich in der Bibliothek der Informatik im Polighaus befindet in dem sich ein Buch befindet:
Willam J. Cook, William H. Cunningham, William R. Pulleyblank, Alexander Schrijver „Combinatorial Optimization“, New York: Wiley 1998.
Dieses Buch umfasst einen Großteil des Stoffes der Vorlesung.
Anders als im Buch angegeben befindet sich das Erratum unter:
http://www.math.uwaterloo.ca/~whcunnin/bookpage.html

Die Homepage der Vorlesung ist:
http://www.informatik.uni-koeln.de/ls_juenger/people/liers/Poly/index.html

Inhaltsverzeichnis

Kapitel 0

Einführung

Vor der Einführung machte Prof. Jünger eine ganz kurze informelle Vorstellung der kombinatorischen Optimierung, bei der er uns bat nicht mit zuschreiben. Dieser Teil ist deshalb hier nicht wieder gegeben.

0.1 Matrizen Exkurs

Auf Bitten von Kommilitonen machte Prof. Jünger einen kleinen Exkurs in die Matrizen–Schreibweise, die er verwendet.

$Ax = b \;\; A \in \mathbb{R}^{m \times n} \;\; b \in \mathbb{R}^m$

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

Matrizen-Multiplikation:
$A \in \mathbb{R}^{m \times k} \;\; B \in \mathbb{R}^{k \times n}; C \in \mathbb{R}^{m \times n}$

$A \cdot B = C$ läuft dann folgendermaßen:

$$c_{ij} = \sum_{p=1}^{k} a_{ip} * b_{pi} \;\; \forall\, i \quad \text{für alle i Zeilen}$$

Gleichungssysteme:

$Ax = b$ dabei ist $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$ anders geschrieben:

$$\begin{pmatrix} a_{11} \\ \vdots \\ a_{m1} \end{pmatrix} x_1 + \begin{pmatrix} a_{12} \\ \vdots \\ a_{m2} \end{pmatrix} x_2 + \ldots + \begin{pmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{pmatrix} x_n = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

Form mit transponierten Vektoren:
$c \in \mathbb{R}^n,\ y \in \mathbb{R}^m$

$$\begin{aligned} y^T A &= c^T \\ y_1(a_{11},\ldots,a_{1n}) + y_2(a_{21},\ldots,a_{2n}), + \ldots + y_m(a_{m1},\ldots,a_{mn}) &= (c_1,\ldots c_n) \end{aligned}$$

Multiplikation von Vektoren:
$a,\ b \in \mathbb{R}^n$

$$ab = \sum_{i=1}^{n} a_i b_i = (a_1,\ldots,a_n) \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

Eigentlich müsste man $a^T b$ schreiben, da a ja ein transponierter Vektor ist. Mathematiker vergessen so etwas aber schon mal gerne, das es ja ohnehin klar ist...

0.2 Allgemeine Formulierungen von Optimierungsproblemen

0.2.1 Generisches Optimierungsproblem

Gegeben:	$A \in \mathbb{R}^{p\times r}$, $B \in \mathbb{R}^{p\times r}$, $C \in \mathbb{R}^{q\times r}$, $D \in \mathbb{R}^{q\times s}$
	$a \in \mathbb{R}^r$, $b \in \mathbb{R}^s$, $c \in \mathbb{R}^p$, $d \in \mathbb{R}^q$
(LP)	$\max a^T x + b^T y$
	$Ax + By = c$
	$Cx + Dy \leq d$
	$x \geq 0$

Dies ist die allgemeine Formulierung eines (linearen) generischen Optimierungsproblems. Für ein ganzzahliges Optimierungsproblem (IP) gilt zusätzlich zum LP noch:

$$x \in \mathbb{Z}^r,\ y \in \mathbb{Z}^s$$

Ein gemischt ganzzahliges Optimierungsproblem liegt vor wenn nur Teile der Vektoren x und y integral (ganzzahlig) sind.

0.2.2 Kombinatorisches Optimierungsproblem

Gegeben:	E eine endliche Grundmenge
	$y \subseteq 2^E$, y ist eine zulässige Teilmenge von E
	c: $E \to \mathbb{R}$ (z.b. Zuweisung von Entfernungen zu den Kanten)
	Für $F \subseteq E$: $c(F) = \sum_{e \in F} c(e)$
Gesucht:	$I^\star \in y$ mit $c(I^\star) \geqq c(I)$, für alle $I \in y$
oder:	$I^\star \in y$ mit $c(I^\star) \leqq c(I)$, für alle $I \in y$

Kleine Aufwärmaufgabe

Für $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ gilt:
$(\exists x \in \mathbb{R}^n)\ Ax = b$ (Gleichungssystem ist lösbar)
genau dann wenn:
$(\nexists y \in \mathbb{R}^m)\ y^T A = 0,\ y^T b = -1$

Hiernach erklärte Prof. Jünger noch kurz und informell die Triangulation für Input-/Outputmodelle. In der Nächsten Stunde wurden zunächst Reihenfolgeprobleme mit einigen Beispielen im Graphenzeichnen und in der Flugplanung veranschaulicht.

0.3 Einführung in die lineare Optimierung

Es folgt ein Beispiel aus der Ölraffinierung: Beim Prozess des so genannten „Crackens“ wird Rohöl in drei Produkte aufgespalten: Schweres Öl (S), mittelschweres Öl (M) und leichtes Öl (L). Nun gibt es zwei verschiedene Crack-Prozesse die zu einer unterschiedlichen Aufteilung der Endprodukte führen.

$$\text{Prozess 1: 10 ME (= Mengeneinheiten) Rohöl ergibt} \longrightarrow \left.\begin{array}{l} \text{2 ME S} \\ \text{2 ME M} \\ \text{1 ME L} \end{array}\right\} \text{Kosten 3 GE}$$

$$\text{Prozess 1: 10 ME (= Mengeneinheiten) Rohöl ergibt} \longrightarrow \left.\begin{array}{l} \text{1 ME S} \\ \text{2 ME M} \\ \text{4 ME L} \end{array}\right\} \text{Kosten 5 GE}$$

Die Firma hat Lieferverpflichtungen, deswegen muss sie mindestens 3 ME S, 5 ME M und 4 ME L produzieren.

Variablen: x_1, Produktionsniveau von Crackprozess 1
x_2, Produktionsniveau von Crackprozess 2
$x_1 \geqq 0, x_2 \geqq 0$
z.B. $x_1 = 1,5$Ł $\Leftrightarrow$ 15 ME Rohöl für Prozess 1

Damit erhalten wir das folgende lineare Optimierungsproblem:

$$\begin{array}{lll} \min & 3x_1 + 5x_2 & \text{(Zielfunktion)} \\ \text{s.t.} & (1) \quad 2x_1 + x_2 \geqq 3 & \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right. \\ & (2) \quad 2x_1 + 2x_2 \geqq 5 & \\ & (3) \quad x_1 + 4x_2 \geqq 4 & \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} \text{Restriktionen} \\ & (4) \quad x_1 \geqq 0 & \\ & (5) \quad x_2 \geqq 0 & \end{array}$$

Jeder Vektor $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$, der (1), (2)...., (5) erfüllt heißt zulässige Lösung.

Graphische Darstellung (gerade keine Lust auf xfig, das bekommt ihr ja wohl selber hin :)

Aus der n.v. Grafik kann man ganz einfach die optimale Lösung erkennen: $x^* = \begin{pmatrix} 2 \\ 0,5 \end{pmatrix}, Wert = 8,5$.

Nähme man nun aber eine beliebige Zielfunktion $ax_1 + bx_2$ sowie $a < 0$ und definiere eine Folge von Punkten $x^{(n)} = \begin{pmatrix} n \\ 0 \end{pmatrix}$ mit $n \geq 4$. Weiterhin gilt: $ax_1^{(n)} + bx_2^{(n)} = na$. Da a negativ ist ist der Wert der Zielfunktion offensichtlich nicht nach unten beschränkt.

0.3.1 Lösungen für Lineare Probleme

Simpler Lösungsalgorithmus für ein LP

Zunächst stellen wir eine Vermutung an. Wenn eine Optimallösung existiert, so existiert eine die im Durchschnitt von n Hyperebenen liegt (wird später bewiesen). Eine Hyperebene ist ein (n-1)-dimensionaler Unterraum also im R^2 eine Gerade im R^3 eine Ebene, u.s.w.

Davon ausgehend konstruieren wie eine einfache Lösungsmethode. Man schreibe (1), (2), ..., (5) als Gleichungen und löse alle $\binom{5}{2} = 10$ Gleichungssysteme mit 2 Gleichungen. Von allen Lösungen, die zulässig sind, ist die mit dem kleinsten Zielfunktionswert optimal.

Aufwand: Benötigen $\binom{m}{n}$ Aufrufe des Gleichungssytemlösers mit n Variablen und n Gleichungen. → Viel zu langsam in der Praxis.

Idee des Simplexalgorithmus

Der Simplexalgorithmus geht folgendermaßen vor:
(a) Finde eine erste Lösung wie im simplen Algorithmus.
(b) Gehe durch Austausch je einer Hyperebene zu einer benachbarten Lösung deren Zielfunktionswertes nicht schlechter ist. (c) Wiederhole (b) bis Optimalität festgestellt ist. (→ via Dualitätstheorie, die später noch behandelt wird).

Bestimmung von Schranken

Wir zeigen nun, dass $x^* = \begin{pmatrix} 2 \\ 0,5 \end{pmatrix}$ optimal ist:
Beobachtung 1: Die Lösungsmenge einer Ungleichung ist invariant unter Skalierung mit positiven Skalaren, z.B. (1):

$$\{x|2x_1 + x_2 \geqq 3\} = \{x|4x_1 + 2x_2 \geqq 6\}$$

Negative Faktoren kehren die Ungleichung um.

Beobachtung 2: Erfüllt ein Vektor zwei gleich gerichtete (beide „$\leqq$“ oder beide „$\geqq$“) so auch die Summe der beiden Ungleichungen. z.B. (1) + (3):

$$\begin{array}{lrcl} (1) & 2x_1 + x_2 & \geqq & 3 \\ (3) & x_1 + 4x_2 & \geqq & 4 \\ \hline & \underbrace{3x_1 + 5x_2}_{ZF} & \geqq & 7 \end{array}$$

Nun können wir behaupten, dass wir mindestens 7 GE ausgeben müssen. → Jeder Vektor der (1)...(5) erfüllt hat einen ZF-Wert von wenigstens 7. Also ist 7 eine untere Schranke für das Minimum. Nun versuchen wir eine bessere Schranke zu finden:

$$\begin{array}{lrcl} (1) & 2x_1 + x_2 & \geqq & 3 \\ (2) & 2x_1 + 2x_2 & \geqq & 5 \\ \hline & 4x_1 + 3x_2 & \geqq & 8 \end{array}$$

In dieser Ungleichung ist der Koeffizient für $x_1 = 4$ und ist damit größer als der Koeffizient in der Zielfunktion. Damit ist dies Ungleichung nicht zum Abschätzen einer unteren Schranke geeignet. Also ein erneuter Versuch indem wir (2) mit 1,5 multiplizieren. Damit erhalten wir:

$$3x_1 + 3x_2 \geqq 7,5$$

Dualität

Zwar haben wir nun eine bessere untere Schranke aber dieser Probier-Ansatz ist etwas unbefriedigend. Darum nun ein systematischer Ansatz zum Finden der Faktoren für die Gleichungen um eine Schranke zu finden. Da die Nichtnegativ-Bedingungen in diesem Fall irrelevant sind suchen wir für unser Problem drei Multiplikatoren (y_1, y_2, y_3) für die Gleichungen (1) (2) (3).

$$\begin{array}{lrcl} \max & 3y_1 + 5y_2 + 4y_3 & \rightarrow & \text{Die Rechte Seite (Schranke) maximieren)} \\ \text{s.t.} & 2y_1 + 2y_2 + y_3 & \leqq & 3 \\ & y_1 + 2y_2 + 4y_3 & \leqq & 5 \end{array} \Bigg\} \text{Damit die ZF-Koeffizienten nicht überschritten werden}$$

Damit haben wir nun das entsprechende duale Optimierungsproblem. Nach dessen Konstruktion liefert der Zielfunktionswert jeder zulässigen Lösung des dualen Problems eine untere Schranke für den Zielfunktionswert des primalen Problems. Wir betrachten:

$$y^* = \begin{pmatrix} y_1^* \\ y_2^* \\ y_3^* \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{7}{6} \\ \frac{2}{3} \end{pmatrix}$$

dies entspricht:

$$\begin{array}{lrcl} \frac{7}{6} \cdot (2): & \frac{14}{6}x_1 + \frac{14}{6}x_2 & \geqq & \frac{35}{6} \\ \frac{2}{3} \cdot (3): & \frac{4}{6}x_1 + \frac{16}{6}x_2 & \geqq & \frac{16}{6} \\ \hline & 3x_1 + 5x_2 & \geqq & 8,5 \end{array}$$

Schwacher Dualitätssatz Es seien $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ für die Optimierungsaufgaben:

(P) $\min\{c^T x | Ax \geqq b,\ x \geqq 0\}$
d.h. bestimme $x^* \in \mathbb{R}$ mit $Ax^* \geqq b$, $x^* \geqq 0$ und $c^T x^*$ so klein wie möglich.

(D) $\max\{y^T b | y^T A \leqq c^T,\ y \geqq 0\}$
d.h. bestimme $y^* \in \mathbb{R}^m$ mit $g^{*T} A \leqq c^T,\ y^* \geqq 0$ und $y^{*T} b$ so groß wie möglich.

so gilt folgendes: Seien $x_0 \in \mathbb{R}^n$, $y_0 \in \mathbb{R}^m$ Punkte mit $Ax_0 \geqq b$, $x_0 \geqq 0$ und $y_0^T A \leqq c^T$, $y_0 \geqq 0$ dann gilt $y_0^T b \leqq c^T x_0$
Beweis:

$$y_0^T b \leqq y_0^T (Ax_0) = (y_0^T A) x_0 \leqq c^T x_0 \quad \text{q.e.d.}$$

Der starke Dualitätssatz (später) besagt $y^{*T} b = c^T x^*$ für Optimallösung x^* von (P) und y^* von (D).

Ökonomische Interpretation des Dualen Programms

Ein Manager der Ölraffinerie würde sich nun die Frage stellen, ob es sinnvoll L, M uns S selbst herzustellen, oder ob man es auf dem Markt kaufen sollte b.z.w. bei welchen Marktpreis eine Produktion nicht mehr lohnt. In diesem Fall drückt y_1 den Preis in GE für S aus, y_2 für M, y_3 für L. Der Marktpreis für die benötigten Mengen beträgt somit:

$$3y_1 + 5y_5 + 4y_3 \ \text{GE}$$

der Ausstoß von Crackprozess 1 hat den Marktpreis:

$$2y_1 + 2y_2 + 3y_3 \ \text{GE} > 3 \quad \rightarrow \text{Produktion Lohnt}$$

der Ausstoß von Crackprozess 2 hat den Marktpreis:

$$y_1 + 2y_2 + 4y_3 \ \text{GE} > 5 \quad \rightarrow \text{Produktion Lohnt}$$

y_i werden auch als Schattenpreise bezeichnet. In unserem Beispiel betragen sie:

$$(S):\ y_1^* = 0 \quad (M):\ y_2^* = \tfrac{7}{6} \quad (L):\ y_3^* = \tfrac{2}{3}$$

Hier ist zu erkennen dass S praktisch „wertlos“ für die Firma ist. In der tat ist es in der Optimallösung im Überfluss vorhanden (also mehr als die Lieferverpflichtungen der Firma verlangen. D.h. man würde das Öl zunächst zu jedem Preis verkaufen (solange man die insgesamte Produktionsmenge dadurch nicht erhöhen muss).

0.3.2 Formen von Linearen Problemen

Kanonische Form eines LPs

Die Kanonische Form eines LPs lautet:

$$\begin{array}{lll} \max\ c^T x & & \\ Ax & \leqq & b \\ x & \geqq & 0 \end{array}$$

Transformationen von anderen Standardformen in die kanonische Form:

1) $$\begin{array}{rcl} \max a^T y & & \\ By & = & d \\ y & \geqq & 0 \end{array} \quad \xrightarrow{x=y} \quad \begin{array}{rcl} \max a^T x & & \\ \begin{pmatrix} B \\ -B \end{pmatrix} x & \leqq & \begin{pmatrix} d \\ -d \end{pmatrix} \\ x & \geqq & 0 \end{array}$$

2) $$\begin{array}{rcl} \max a^T y & & \\ By & \leqq & d \end{array} \quad \xrightarrow{x=\begin{pmatrix} y+ \\ y- \end{pmatrix}} \quad \begin{array}{rcl} \max \begin{pmatrix} a \\ -a \end{pmatrix}^T x & & \\ (B,-B)x & \leqq & d \\ x & \geqq & 0 \end{array}$$

3) $$\begin{array}{rcl} \max a^T y & & \\ By & = & d \end{array} \quad \xrightarrow{x=\begin{pmatrix} y+ \\ y- \end{pmatrix}} \quad \begin{array}{rcl} \max \begin{pmatrix} a \\ -a \end{pmatrix}^T x & & \\ \begin{pmatrix} B & -B \\ -B & B \end{pmatrix} x & \leqq & \begin{pmatrix} d \\ -d \end{pmatrix} \\ x & \geqq & 0 \end{array}$$

Als Hausaufgabe dürfen wir die Transformationen in die andere Richtung machen :), (bei $By = d$ geht es nicht).

Jedes LP kann also auch in die Form $\max\{c^T x | Ax \leqq b\}$ transformiert werden.

Farkas Lemma Wir suchen ein Analogon zur „Aufwärmaufgabe“:
$A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m$
$(\exists x)\ \ Ax = b$ genau dann wenn $(\nexists y)\ \ y^T A = 0,\ \ y^T b = -1$ für Ungleichungsprobleme der Form $Ax \leqq b$

Fourier-Motzkin Elimination Durch Multiplikation der Ungleichungen mit positiven Skalaren und Umordnung erhalten wir:

$$(*)\begin{cases} x_1+(a_i')^Tx' & \leqq & b_i^* \ \forall\, i \in \{1,2,3,...m'\} & \text{alle mit pos. Faktor vor } x_1 \\ -x_1+(a_i')^Tx' & \leqq & b_i^* \ \forall\, i \in \{m'+1,...m''\} & \text{alle mit neg. Faktor vor } x_1 \\ (a_i')^Tx' & \leqq & b_i^* \ \forall\, i \in \{m''+1,...m\} & \text{hier war gar kein } x_1 \text{ drin} \end{cases}$$

wobei $x' = (x_2, x_3, ...x_n)$
$(a_i')^T$ ist die i-te Zeile von A ohne den ersten Eintrag

Die ersten beiden Zeilen von (*) sind äquivalent zu

$$\max_{m'+1\leqq j\leqq m''} \left((a_j')^Tx' - b_j\right) \leqq x_i \leqq \min_{1\leqq i\leqq m'} \left(b_i - (a_i')^Tx'\right)$$

d.h. das Orginalsystem $A \leqq b$ hat eine Lösung genau dann wenn (in Zukunft g.d.w.) das System:

$$(**)\begin{cases} (a_j')^Tx' - b_j & \leqq & b_i - (a_i')^Tx' & \forall\, i \in \{1,2,3,...m'\} \\ & & & \forall\, i \in \{m'+1,...m''\} \\ (a_i')^Tx' & \leqq & b_i' & \forall\, i \in \{m''+1,...m\} \end{cases}$$

eine Lösung hat (x_i kann geeignet gewählt werden).
Umformulierung von (**)

$$(***)\begin{cases} (a_i'+a_j')^Tx' & \leqq & b_i+b_j & \forall\, i \in \{1,2,3,...m'\} \\ & & & \forall\, i \in \{m'+1,...m''\} \\ (a_i')^Tx' & \leqq & b_i' & \forall\, i \in \{m''+1,...m\} \end{cases}$$

Schreibe (***) als $A'x' \leqq b'$
Also $(\exists x)\ Ax \leqq b$ g.d.w. $(\exists x')A'x' \leqq b'$

Fazit: Wir haben die Variable x eliminiert und haben jetzt in (***) $n-1$ Variablen und $m'(m''-m')+m-m''$ Ungleichungen $\rightarrow$ weniger Variablen aber mehr Ungleichungen.

Diese Iteration liefert ein nicht polynomielles Verfahren zur Lösung von Ungleichungssystemen.

Lemma 0.1 *von Farkas für Ungleichungen*
Seien $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$
$(\exists x \in \mathbb{R}^n)\ Ax \leqq b$ *g.d.w.* $(\nexists y \in \mathbb{R}^m)\ y \geqq 0,\ y^TA = 0,\ y^Tb < 0$

Beweis

„⇒“ $Ax \leqq b$ habe eine Lösung $\bar{x}$
Annahme: $(\exists y \geqq 0)\ y^T A = 0,\ y^T b < 0$

$$\begin{aligned} \Rightarrow 0 &> y^T b \\ &\geqq y^T A\bar{x} \\ &= 0 \ \lightning \end{aligned}$$

Mit $0 > 0$ ergibt sich also ein Widerspruch. Also gilt die Hin-Richtung.

„⇐“ $Ax \leqq b$ habe keine Lösung.
1) Induktionsverankerung: $A \in \mathbb{R}^{m\times 1}$ Nach Skalierung mit positiven Skalaren $\bar{y}$ haben alle Ungleichungen die Form:

$$(1)\ \ x_n \leqq \alpha; \quad \text{oder } (2)\ \ -x_n \leqq \beta$$

Nicht Lösbarkeit bedeutet also dass sich zwei Ungleichungen widersprechen. O.b.d.A (=Ohne Beschränkung der Allgemeinheit) widersprechen sich die i-te Form (1) und die (j)-te der Form (2). D.h. $-\beta \leqq x_n \leqq \alpha$ mit $\alpha < -\beta$ d.h. $\alpha+\beta = \gamma < 0$
Man wähle nun $y_i = -\frac{\bar{y}_i}{\gamma} > 0$ da $\gamma < 0$

$$\begin{aligned} y_j &= -\tfrac{\bar{y}_j}{\gamma} > 0 \\ y_k &= 0 \quad \text{für } k \notin \{i,j\} \end{aligned}$$

$$\Rightarrow \quad y \geqq 0$$

$$\begin{aligned} y^T A &= \left(\ldots,0,-\tfrac{\bar{y}_i}{\gamma},0,\ldots,0,-\tfrac{\bar{y}_j}{\gamma},0,\ldots\right) \begin{pmatrix} \vdots \\ * \\ \frac{1}{y_i} \\ * \\ \vdots \\ * \\ -\frac{1}{y_j} \\ * \\ \vdots \end{pmatrix} \\ &= -\tfrac{1}{\gamma} + \tfrac{1}{\gamma} = 0 \\ y^T b &= -\tfrac{\alpha}{\gamma} - \tfrac{\beta}{\gamma} = -\tfrac{\alpha+\beta}{\gamma} = -1 < 0 \end{aligned}$$

2) Induktion: $A \in \mathbb{R}^{m\times n}\ \ n > 1$
$Ax \leqq b$ hat keine Lösung $\Leftrightarrow A'x \leqq b'$ hat keine Lösung

3) Induktionsannahme $(\exists y' \geqq 0)\ y'^T A' = 0$
$y'^T b' = -1$
d.h. $(\exists y' \geqq 0) y'^T [A'b'] = (\underbrace{0,\ldots,0}_{n-1},-1)$
Jede Zeile der Matrix $[0,\ A',\ b']$ ist Summe von nicht-negativ skalierten Zeilen von $[A,\ b]$
$\Rightarrow$ $(\underbrace{0,\ldots,0}_{n},-1)$ ist nicht-negative Linearkombination der Zeilen von $[A, b]$
d.h. $(\exists y \geqq 0)\ y^T A = 0 \quad y^T b = -1$

Korollar 0.2 *Lemma von Farkas, 1894*

Seien $A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m$

$$\begin{array}{rcl} (\exists x \in \mathbb{R}^n)\ x & \geqq & 0 \\ Ax & = & b \end{array} \quad g.d.w. \quad \begin{array}{rcl} (\nexists y \in \mathbb{R}^m)\ y^T A & \geqq & 0 \\ y^T b & < & 0 \end{array}$$

Beweis: Für $\bar{A} = \begin{pmatrix} A \\ -A \\ -I \end{pmatrix}$ (mit I = Einheitsmatrix) und $\bar{b} = \begin{pmatrix} b \\ -b \\ 0 \end{pmatrix}$ gilt:

$$(\exists x \geqq 0)\, Ax = b \overset{\text{Konstruktion}}{\Leftrightarrow} \begin{array}{rcl} (\exists \bar{x})\ \bar{A}\bar{x} & \leqq & \bar{b} \end{array}$$

$$\overset{\text{Satz 0.1 (Lemma für Ungl.)}}{\Leftrightarrow} \begin{array}{rcl} (\nexists \bar{y} \geqq 0)\ \bar{y}^T \bar{A} & = & 0 \\ \bar{y}^T b & \leqq & 0 \end{array}$$

$$\left[\bar{y} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}\right] \Leftrightarrow \nexists \begin{pmatrix} u \\ v \\ w \end{pmatrix} \geqq 0 \quad \begin{array}{rcl} u^T A - v^T A - w^T & = & 0 \\ u^T b - v^T b & < & 0 \end{array}$$

$$[y = u - v] \qquad \Leftrightarrow \qquad \begin{array}{rcl} (\nexists y)\ y^T A & \geqq & 0 \\ y^T b & < & 0 \end{array} \quad \text{q.e.d}$$

Farkas Lemma kann auch anders betrachtet werden:

$$\begin{array}{lrcl} \text{Entweder} & (\exists x) Ax & = & b, \quad x \geqq 0 \\ \text{oder} & (\exists y) y^T A & \geqq & 0, \quad y^T b < 0 \end{array}$$

aber nicht beides. Dies kann man im R^2 auch graphisch darstellen in dem man die Matrix $A = (A_1, A_2$ als Zusammensetzung aus zwei Vektoren A_1 und A_2 betrachten, die jeweils eine „halbe“ Ebene beschreiben. $y^T A \geqq$ beschreibt dann den Schnitt aus diesen beiden „halben“ Ebenen. $y^T b < 0$ beschreibt wiederum eine „halbe“ Ebene. Wenn jetzt nun $Ax = b$

gelten soll so muss b sich als eine Linearkombination der Vektoren A_1 und A_2 darstellen lassen. Anschaulich bedeutet dass das b in der Zeichnung „zwischen“ A_1 und A_2 liegen muss dann wiederum wird sich die Flächen die durch $y^T A \geqq 0$ und durch $y^T b < 0$ beschrieben werden niemals schneiden.

Leider habe ich an dieser Stelle nicht die Zeit die Zeichnungen zu malen. Freiwillige vor.

Korollar 0.3 *Das System $Ax \leqq b$ habe wenigstens eine Lösung. Dann erfüllt jede Lösung x von $Ax \leqq b$ die Ungleichung $c^T x \leqq \delta$ genau dann, wenn ein $y \geqq 0$ existiert, so dass $y^T A = c^T$ und $y^T b \leqq \delta$.*

Beweis:

„$\Leftarrow$“ $(\exists y \geqq 0) y^T A = c^T,\ y^T b \leqq \delta$
für alle $x \in \mathbb{R}^n$ gilt:

$$\begin{aligned} Ax \leqq b \Rightarrow y^T Ax &\leqq y^T b \\ \Rightarrow\ c^T x &\leqq y^T b \\ \Rightarrow\ c^T x &\leqq \delta \end{aligned}$$

„$\Rightarrow$“ Annahme

$$\begin{aligned} (\nexists y \geqq 0) y^T A &= c^T \\ y^T b &\leqq \delta \end{aligned}$$

$$\Rightarrow (\nexists y \geqq 0,\ \lambda \geqq 0)\ \ (y^T, \lambda) \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} = (c^T, \delta)$$

$$\overset{\text{(Farkas)}}{\Rightarrow} \left(\exists \begin{pmatrix} z \\ \mu \end{pmatrix}\right) \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} z \\ \mu \end{pmatrix} \geqq 0$$

$$(c^T, \delta) \begin{pmatrix} z \\ \mu \end{pmatrix} < 0$$

Fall 1 $\mu = 0$:

$$\Rightarrow \begin{cases} Az \geqq 0 \\ c^T z < 0 \end{cases}$$

Sei x^0 Lösung von $Ax \leqq b$. Für r hinreichend groß gilt dann:

$$\begin{aligned} A(x^0 - rz) &\leqq b \\ c^T(x^0 - rz) &> \delta \end{aligned}$$

$\bar{X} = x^0 - rz$ ist Lösung die $c^T\bar{x} \leqq \delta$ verletzt.

Fall 2 $\mu > 0$:
Es gilt $Az + b\mu \geqq 0$

$$\begin{aligned} \Rightarrow \quad Az &\geqq -b\mu \\ \Rightarrow \quad A\left(-\tfrac{z}{\mu}\right) &> \delta \end{aligned}$$

aber $c^T z + \delta\mu < 0$

$$\begin{aligned} \Rightarrow \quad c^T z &< -\delta\mu \\ \Rightarrow \quad c^T\left(-\tfrac{z}{\mu}\right) &> \delta \end{aligned}$$

Also liefert die Lösung $\bar{x} = -\frac{z}{\mu}$ den Widerspruch.

0.4 Dualitätstheorem

Für das lineare Optimierungsproblem:

$$(P) \quad \{\max c^T x | Ax \leqq b\}$$

bezeichnen wir:

- Jede Lösung $\bar{x}$ von $Ax \leqq b$ als zulässige Lösung
- Jede zulässige Lösung $\bar{x}$ die $c^T x$ maximiert als Optimallösung.

Wir definieren das duale lineare Optimierungsproblem zu (P) als (D)

$$\min\{y^T b | y \geqq 0,\ y^T A = c^T\}$$

Satz 0.4 *Schwaches Dualitätstheorem*
Seien $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ und $\bar{x}$ zulässig für (P) und y zulässig für (D). Dann gilt $c^T\bar{x} \leqq \bar{y}^T b$

Beweis: $c^T\bar{x} = (y^T A)\bar{x} = y^T(Ax) \leqq y^T b$ q.e.d.

Satz 0.5 *Dualitätstheorem (von Neumann, 1947)*
Seien $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$. Dann gilt:

$$\max\{c^T x | Ax \leqq b\} = \min\{y^T b | y \geqq 0,\ y^T A = c^T\}$$

Beweis:

1. Schwache Dualität (Satz 0.4) impliziert:

$$\begin{array}{lll} (*) \qquad \delta := & \sup\{c^T x | Ax \leqq b\} \leqq \inf\{y^T b | y \geqq 0,\ y^T A = c^T\} \\ \Rightarrow & (\forall x)(Ax \leqq b \Rightarrow c^T x \leqq \delta) \\ \overset{\text{Korollar 0.3}}{\Rightarrow} & (\exists y)\ y \geqq 0,\ y^T A = c^T, y^T b \leqq \delta \end{array}$$

$\Rightarrow$ Das Infimum wird angenommen.
$\Rightarrow$ Das Minimum existiert und ist $= \delta$

2. Annahme:
$$(\nexists x) A \leqq b,\ c^T x \geqq \delta$$
$\Rightarrow$
$$(\nexists x) \begin{pmatrix} A \\ -c^T \end{pmatrix} x \leqq \begin{pmatrix} b \\ -\delta \end{pmatrix}$$

Lemma von Farkas für Ungleichungen $\Longrightarrow$

$$\left(\exists \begin{pmatrix} z \\ \lambda \end{pmatrix} \in \mathbb{R}^{m+1}\right), \begin{pmatrix} z \\ \lambda \end{pmatrix} \geqq 0 \text{ bzw. } z \geqq 0,\ \lambda \geqq 0$$
$$(z^T, ;\ \lambda) \begin{pmatrix} A \\ -c^T \end{pmatrix} = 0 \text{ bzw. } z^T A - \lambda c^T = 0$$
$$(z^T,\ \lambda) \begin{pmatrix} b \\ -\delta \end{pmatrix} \leqq 0 \text{ bzw. } z^T b - x\lambda \leqq 0$$

$Ax \leqq b$ habe eine Lösung x^0
Annahme $x = 0 \Rightarrow 0 = z^T A x^0 \leqq z^T b < 0$ ↯
Also gilt: $\lambda > 0$

Für $y = \frac{z}{\lambda}$ gilt $\quad y \geqq 0$ da $\begin{array}{l} z \geqq 0 \\ \lambda \geqq 0 \end{array}$

$$\begin{aligned} y^T A &= \tfrac{1}{\lambda} z^T A = \tfrac{1}{\lambda} \lambda c^T = c^T \\ y^T b &= \tfrac{1}{\lambda} z^T b < \tfrac{1}{\lambda} \lambda \delta = \delta \end{aligned}$$

$\Rightarrow$ Infimum ist $< \delta$ ↯ zu $(*)$

0.5 Dualisieren von LPs

Im folgenden Abschnitt beschrieb Prof. Jünger ein „Kochrezept“ zum Dualisieren von LPs. Gegeben sei das folgende primale Problem:

$$\begin{array}{lrcll} \text{(P)} & \max a^T x + b^T y & & & \\ & Ax + By & = & c & [u] \\ & Cx + Dy & \leqq & d & [v] \\ & x & \geqq & 0 & \end{array}$$

Wir erhalten folgendes duale Problem:

$$\begin{array}{lllll} \text{(D)} & \min u^T c + v^T d & & & \\ & u^T A + v^T C & \text{(Un)gleichungsbeziehung} & a^T & [x] \\ & u^T B + v^T D & \text{(Un)gleichungsbeziehung} & b^T & [y] \end{array}$$

Nun gilt: Multiplikatoren für Gleichungen ergeben nicht vorzeichenbeschränkte Variablen und Multiplikatoren für Ungleichungen ergeben Vorzeichenbeschränkte Variablen, in unserem Fall also:

$$\begin{array}{lrcll} \text{(D)} & \min u^T c + v^T d & & & \\ & u^T A + v^T C & \geqq & a^T & [x] \\ & u^T B + v^T D & = & b^T & [y] \\ & v & \geqq & 0 & \end{array}$$

Daraus kann man leicht die Spezialfälle ableiten:

$$\begin{array}{llllll} (P) & \max a^T x & & & (D) & \min v^T d \\ & Cx & \leqq & d \;\rightarrow & & v^T C \geqq a^T \\ & x & \geqq & 0 & & v \geqq 0 \\ (P) & \max a^T x & & & (D) & \min v^T c \\ & Ax & = & c \;\rightarrow & & u^T A \geqq a^T \\ & x & \geqq & 0 & & \\ (P) & \max b^T y & & & (D) & \min v^T d \\ & Dy & \leqq & d \;\rightarrow & & v^T D = b^T \\ & & & & & v \geqq 0 \end{array}$$

Satz 0.6 *Satz vom komplementären Schlupf. Seien* $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$
Sei x^* *zulässig für* $\max\{c^T x | Ax \leqq b\}$ *und*
sei y^* *zulässig für* $\min\{y^T b | y \geqq 0,\ y^T A = c^T\}$

Dann sind x^* und y^* Optimallösungen wenn gilt:
$y_i^* = 0$ *oder* $a_i x^* = b_i \ \forall i$
wobei a_i die i-te Zeile von A bezeichnet.

Beweis:

„⇒“ x^* und y^* seien Optimallösungen
sei $i \in \{1,2,\ldots,m\}$ mit $y_i^* > 0$ und $a_i x^* < b$

$$\begin{aligned} \Rightarrow \quad 0 &= y^{*T} b - c^T x^* \\ &= y^{*T} b - y^{*T} A x^* \\ &= y^{*T}(b - A x^*) \\ &\geqq y_i^*(b_i - a_i x^*) \\ &> 0 \quad \lightning \end{aligned}$$

„⇐“ Für alle $i \in \{1,2,\ldots,m\}$ gilt $y_i^* = 0$ oder $a_i x^* = b_i$

$$\begin{aligned} \Rightarrow \quad y^{*T} b - c^T x^* &= y^{*T}(b - A x^*) \\ &= \sum_{i=1}^{m} y_i^*(b_i - a_i x^*) \\ &= 0 \qquad \text{q.e.d.} \end{aligned}$$

Dann folgt wegen starker Dualität x^*und y^*sind Optimallösungen.

0.6 Der Simplexalgorithmus

Es folgt eine Skizze des Verfahrens für LPs in der Form:

$$\begin{aligned} (P) \qquad \max c^T x & \\ A x &= b \\ x &\geqq 0 \end{aligned}$$

mit $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $\operatorname{rang}(A) = m$.
Sei T die Menge der Spaltenindizes von A. Für $B \subseteq T$ sei A_B die Submatrix von A mit den Spalten aus B und c_B der Subvektor von c mit den entsprechenden Komponenten.

Falls $|B| = m$ und die Spalten von A_B linear unabhängig sind so ist B eine Basis von A,

b.z.w. A_B eine Basismatrix. Für eine Basis B ist die Lösung x von $Ax = b$ mit $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$:

$$\begin{aligned} N &= T \setminus B, \qquad (N = \text{die die nicht in der Basis sind}) \\ x_B &= A_B^{-1} b, \quad x_N = 0 \end{aligned}$$

die zu B gehörige Basislösung.
x_B ist die eindeutige Lösung des Gleichungssystems $A_B x_B = b$.
Weiterhin ist $y_T = c_B^T A_B^{-1}$ (die eindeutige Lösung von $y^T A_B = c_B^T$)
eine Lösung des dualen LPs:

$$\begin{aligned} (D) \qquad & \min y^T b \\ & y^T A \geqq c^T \end{aligned}$$

Basislösungen x und y haben die Eigenschaft $x_i > 0 \Rightarrow y^T a_i = c_i$, wobei a_i die i-te Spalte von A ist.

D.h. x und y sind zulässig für (P) und (D).
$\overset{\text{Satz vom kompl. Schlupf 0.6}}{\Rightarrow}$ x und y sind optimal.

Eine Basis B ist zulässig falls $x_B \geqq 0$.
Der Simplexalgorithmus generiert eine Folge von zulässigen Basen. Jedes mal wird y auf duale Zulässigkeit überprüft.
y dual zulässig $\rightarrow$ STOP (Optimalität)

Sei x zulässige aber nicht optimale Basislösung.
Wähle $i \in T$ mit $y^T a_i < c_i$
Plan: In B wird ein Index entfernt und i hinzugefügt.

Wir lösen $A_B z = a_i$ (d.h. z ist die i-te Spalte nach Transformation durch die Basis).
Für $\varepsilon > 0$ ersetzen wir x_B durch $x_B - \varepsilon z$ und setzen $x_i = \varepsilon$ und erhalten eine neue Lösung von $Ax = b$ mit Zielfunktionswert:

$$\begin{aligned} c_B^T(x_B - \varepsilon z) + c_i \varepsilon &= c_B^T x_B + \varepsilon(c_i - c_B^T z) \\ &= c_B^T x_B + \varepsilon(c_i - y^T A_B z) \\ &= c_B^T x_B + \varepsilon(c_i - y^T a_i) \end{aligned}$$

Es gilt $c_i - y^T a_i > 0$, d.h. jedes $\varepsilon > 0$ verbessert die Lösung.
Die neue Lösung soll zulässig bleiben, d.h. wir wählen:

$$\max\{\varepsilon | x_B - \varepsilon z \geqq 0\}$$

Falls das Maximum nicht existiert so ist (P) unbeschränkt $\rightarrow$ STOP (Unbeschränktheit)

sonst existiert ein Index J mit $z > 0$ und $(x_B - \varepsilon_z)_j = 0$. Wir wählen j als die Basis verlassenden Index. Dieser Test heißt „ratio test“ (Verhältnistest).
$j \in B$ mit $z_j > 0$ und Verhältnis $\frac{x_j}{z_j}$ minimal unter $\frac{x_k}{z_k}$ mit $k \in B$
Die neue Basis ist also $(B \setminus \{j\}) \cup \{i\}$.

0.6.1 Simplex Algorithmus-Zusammenfassung

An dieser stelle eine Formulierung in Pseudo-Code

loop
  Finde die eindeutige Lösung von $y^T A_B = c_B^T$
  if für alle $i \in T \setminus B$ gilt: $y^T a_i \geqq c_i$ **then**
    STOP: die Lösung ist optimal.
  else
    wähle i mit $y^T a_i < c_i$
    Finde die eindeutige Lösung von $A_B z = a_i$
    Finde größtes $\varepsilon \geqq 0$ mit $x_b - \varepsilon z \geqq 0$
    if $\nexists \varepsilon$ **then**
      STOP: Problem ist unbeschränkt
    else
      wähle $j \in B$ mit $z_j > 0$ und $(x_B - \varepsilon_z)_j = 0$
      Ersetze B durch $(B \cup \{i\}) \setminus \{j\}$
      Ersetze x_B durch $x_B - \varepsilon z$
      Setze $x_i = \varepsilon$
    end if
  end if
end loop

0.6.2 Beispielrechnung mit dem Simplexalgorithmus

Gegeben ist das folgende lineare Problem:

$$\begin{array}{rrcrcrcl} \max & 3x_1 & + & 5x_2 & + & 4x_3 & & \\ \text{s.t.} & 2x_1 & + & 2x_2 & + & x_3 & \leqq & 3 \\ & x_1 & + & 2x_2 & + & 4x_3 & \leqq & 5 \\ & & & & & x_1, x_2, x_3 & \geqq & 0 \end{array}$$

Einführung von Schlupfvariablen um Gleichungen zu erhalten:

$$\begin{array}{rrcrcrcrcrcl} \max & 3x_1 & + & 5x_2 & + & 4x_3 & & & & & & \\ \text{s.t.} & 2x_1 & + & 2x_2 & + & x_3 & + & x_4 & & & = & 3 \\ & x_1 & + & 2x_2 & + & 4x_3 & & & + & x_5 & = & 5 \\ & & & & & & & x_1, x_2, x_3, x_4, x_5 & & & \geqq & 0 \end{array}$$

ergibt die Form:

$$\begin{aligned} \max c^T x & \\ Ax &= b \\ x &\geqq 0 \end{aligned}$$

$$\text{mit} \quad A = \begin{pmatrix} 2 & 2 & 1 & 1 & 0 \\ 1 & 2 & 4 & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

$$c^T = \begin{pmatrix} 3 & 5 & 4 & 0 & 0 \end{pmatrix}$$

Die Anfangsbasis sei nun $B = \{4, 5\}$ d.h. $A_B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

$$x_B = A_B^{-1} b = \begin{pmatrix} 3 \\ 5 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

1. Iteration:

$$y^T A_B = c_b^T \Rightarrow (y_1, y_2) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = (0, 0) \rightarrow y = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Für jedes i aus $\{1, 2, 3\}$ gilt:

$$y^T a_i = 0 < c_i$$

Wir nehmen $i = 2$

$$A_B z = a_i : \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} z_4 \\ z_5 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \rightarrow z = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

größtes ε mit:

$$x_b - \varepsilon z \geqq 0 : \quad \begin{pmatrix} 3 \\ 5 \end{pmatrix} - \varepsilon \begin{pmatrix} 2 \\ 2 \end{pmatrix} \geqq 0$$

$$\rightarrow \varepsilon = \frac{3}{2}$$

Für $j = 4$ ist $z_j = 2 > 0$ und
$$\begin{aligned}(x_B - \varepsilon z)\\ &= x_4 - \varepsilon z_4\\ &= 3 - \tfrac{3}{2}\cdot 2 = 0\end{aligned}$$

Neue Basis: $B = \{2,\ 5\}$

$$A_B = \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix} \quad x_B = \begin{pmatrix} x_2 \\ x_5 \end{pmatrix} = \begin{pmatrix} \frac{3}{2} \\ 2 \end{pmatrix}$$

2. Iteration

$$y^T \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix} = (5,\ 0) \to y = \begin{pmatrix} \frac{5}{2} \\ 0 \end{pmatrix}$$

Überprüfe $y^T a_i < c_i$

$$\text{für } x_1 \quad : \quad \begin{pmatrix} \frac{5}{2} & 0 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = 5 \geqq 3$$

$$\text{für } x_2 \quad : \quad \begin{pmatrix} \frac{5}{2} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{5}{2} \geqq 0$$

$$\text{für } x_3 \quad : \quad \begin{pmatrix} \frac{5}{2} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 4 \end{pmatrix} = \frac{5}{2} < 4 \to i = 3$$

$$A_B z = a_i: \quad \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} z_2 \\ z_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \end{pmatrix} \to z = \begin{pmatrix} \frac{1}{2} \\ 3 \end{pmatrix}$$

Größtes ε mit:

$$x_B - \varepsilon z \geqq 0: \quad \begin{pmatrix} \frac{3}{2} \\ 2 \end{pmatrix} - \varepsilon \begin{pmatrix} \frac{1}{2} \\ 3 \end{pmatrix} \geqq 0$$

$$\to \varepsilon = \frac{2}{3} \to j = 5$$

neue Basis $B = \{2,\ 3\}$

$$A_B = \begin{pmatrix} 2 & 1 \\ 2 & 4 \end{pmatrix} \quad x_B = \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \frac{7}{6} \\ \frac{2}{3} \end{pmatrix}$$

3. Iteration:

$$y^T \begin{pmatrix} 2 & 1 \\ 2 & 4 \end{pmatrix} = (5,\ 4) \to y = \begin{pmatrix} 2 \\ \frac{1}{2} \end{pmatrix}$$

überprüfe $y^T a_i < c_i$

$$\text{für } x_1\text{:} \quad \begin{pmatrix} 2 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \frac{9}{2} \geqq 3$$

$$\text{für } x_4\text{:} \quad \begin{pmatrix} 2 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 2 \geqq 0$$

$$\text{für } x_5\text{:} \quad \begin{pmatrix} 2 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{2} \geqq 0$$

$\rightarrow$ Lösung ist optimal.
Frage: Terminiert das Verfahren?
Wenn immer $\varepsilon > 0$: Ja denn es gibt nur endlich viele Basen.
Problem: Im Fall $\varepsilon = 0$ haben wir keine ZF-Verbesserung. Eine Folge solcher degenerierter Iterationen kann zurück zur selben Basis führen. $\rightarrow$ keine Terminierung.

Aber es gibt Regeln zur Wahl von i und j die solche „Zyklen“ verhindern. Mit einer solchen sog. Pivotregel terminiert das Verfahren immer. Blands Regel: Nimm den kleinsten Index.

Wie erhält man nun aber die erste zulässige Basis:

- Man multipliziere die Gleichungen mit negativer rechter Seite mit -1, so dass das neue System die rechte Seite $b \geqq 0$ hat.
- Man löse das folgende LP:

$$\begin{aligned} \text{(HLP)} \qquad \max & \sum_{i=1}^{n} -x_i' \\ (A,\, I) & \begin{pmatrix} x \\ x' \end{pmatrix} = b \\ & \begin{pmatrix} x \\ x' \end{pmatrix} \geqq 0 \end{aligned}$$

  mit I als erste Basismatrix (I natürlich immer linear unabhängig).

  Anmerkung: Es gilt (P) hat eine zulässige Lösung g.d.w. (HLP) eine Optimallösung mit Wert 0 hat.

- Ist der Optimalwert von (HLP) kleiner als 0:

  STOP: (P) hat keine Zulässige Lösung

  sonst sei B die optimale Basis.

- Löse:

$$
\begin{aligned}
(\mathrm{P}') \qquad & \max(c^T,\, 0^T) \begin{pmatrix} x \\ x' \end{pmatrix} \\
s.t. \qquad & (A, I) \begin{pmatrix} x \\ x' \end{pmatrix} = b \\
& \begin{pmatrix} x \\ x' \end{pmatrix} \geqq 0
\end{aligned}
$$

  mit Startbasis B.

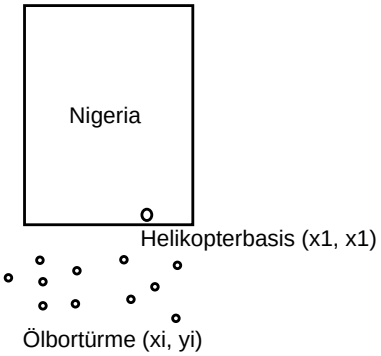
  Anmerkung: Es gibt Techniken, um die künstliche Variablen im Laufe des Verfahrens zu minimieren und zwar die künstlichen Nichtbasisvariablen sofort und die künstlichen Basisvariablen durch geeignete Pivots.

Kapitel 1

Probleme, Schranken, Zertifikate

1.1 Verschiedene Probleme

Ölbohrtürme



Der Helikopter fliegt eine vorgegebene Menge von Ölbohrtürmen ab um die Pumpmenge zu regulieren und fliegt danach zurück zur Basis. Die Zeit, die er dafür benötigt ist proportional zur Flugstrecke (euklidische Distanzen):

$$t(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = t(j,i)$$

Im Gegensatz dazu hat das Leiterplatten-Problem die L_∞-Metrik die auch Max Metrik genannt wird weil immer die maximale Strecke der beiden strecken in x oder y Richtung genommen wird. Der Bohrer wird von zwei Motoren über die Platine gefahren, die unabhängig voneinander ein und ausgeschaltet werden können.

Euklidisches Rundreise-Problem

Gegeben: Menge V von Punkten in der euklidischen Ebene.

Gesucht: Eine Tour minimaler Länge

$$n = |V| \qquad \frac{(n-1)!}{2} \text{ mögliche Touren}$$

Für den Fall dass $n = 23$ bräuchte ein Computer für die Auswertung einer Tour 1ηs ($= 10^{-9}$s) $\rightarrow$ Computer braucht zirka 178 Jahrhunderte zur Bewertung aller Touren.

Plotten zusammenhängender Zeichnungen

Das plotten eines Straßennetzes auf einer 60x60 cm Platte
Länge des Straßennetzes: 44,47m
Die Standard-Plottersoftware legte 67 km zurück und brauchte 9 Stunden und 11 Minuten.

Alternativ wurden die zu plottenden Strecken von einer „intelligenten“ Person durchgeführt.
Leerstrecke: 57,8 m
Zeit: 51 min

Nach Anwendung einer Matching Heuristik: Leerstrecke: 13,62 m
Zeit: 36 min 45 sek.

Kleine Übungsaufgabe In jedem Graphen ist die Anzahl der Punkte mit ungerader Anzahl ein- und ausgehender Kanten gerade (Ein Satz von Euler).

Nach Euler gilt auch, dass man einen Graphen genau dann ohne abzusetzen und ohne eine Strecke zwei mal zu gehen zeichnen kann wenn die Anzahl der Kanten an jedem Knoten gerade ist (einmal muss man rein, einmal raus, falls man noch einmal rein geht, muss man auch wieder raus).

Das hilft uns nun beim Plot-Problem wir suchen nun die minimale Paarung „ungerader“ Knoten (die Gesamtlänge der verbindenden Kanten soll minimal sein).

Andere Rundreise-Probleme

Chinesisches Postboten-Problem: Dasselbe Problem wie beim Plotten aber auf dünnen Graphen (Graphen mit relativ wenig Kanten).

Landpostboten-Problem: Der Landpostboten muss sowohl in den Dörfern möglichst optimal laufen als auch die Dörfer möglichst Optimal abfahren. Das Euklidische Rundreise Problem ist hiervon ein Spezialfall (wenn pro Dorf nur ein Haus beliefert werden muss).

Euklidisches perfektes Matching-Problem

Gegeben: Menge V von Punkten in der Euklidischen Ebene $|V|$ gerade.
Gesucht: Eine Menge von Geraden Linien minimaler Gesamtlänge so dass jeder Punkt Endpunkt genau einer Linie ist.

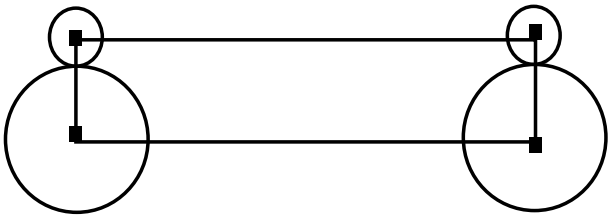
1.2 Dualität und Schranken

TSP-Versuch (Travelling Salesman oder eukl. Rundreise-Problem).

[Zeichnung]

Die Zeichnung stellt eine Menge von Punkten die durch Kanten verbunden sind so dass eine Rundreise entsteht um alle Punkte herum sind nun Kreise, die sich nicht überschneiden sondern sich falls ihre Mittelpunkte Nachbarn auf der Rundreise sind lediglich berühren, bis auf eine Stelle, wo eine Lücke entsteht. Jeder Kreis wird nun genau 2 mal durchlaufen + das eine Stück wo eine Lücke ist. Damit ist die Summe aller Kreisdurchmesser + das extra Stück eine untere Schranke. Falls man es schafft die Kreise so anzuordnen, dass sie sich nicht überschneiden aber Nachbarn sich immer berühren so hat man die Optimallösung.

Oft ist aber auch keine Kombination der Kreise möglich.



In diesem Fall kann man dann Punkte zusammenfassen und mit „Gräben“

[3 Zeichnungen] habe ich leider nur ansatzweise und nicht die Zeit sie am Computer nach zu zeichnen.

An das Euklidische Matching-Problem kann man ebenfalls mit dieser Methode heran gehen.

Euklidisches Rundreise untere Schranken Problem

Gegeben: Menge V von Punkten in der euklidischen Ebene.

Gesucht: nicht überlappendes System von Scheiben und Gräben das:

$$B = 2\sum_{v\in V} r_v + 2\sum_{m\in M} w_m$$

maximiert.

Euklidisches perf. Matching untere Schranken Problem

Gegeben: Menge V von Punkten in der euklidischen Ebene. $|V|$ gerade.
Gesucht: nicht überlappendes System von Scheiben und Gräben das:

$$\bar{B} = \sum_{v\in V} r_v + \sum_{m\in M} w_m$$

maximiert.

In der L_∞-Matrix nimmt man Quadrate.

1.2.1 Lineare Optimierung und untere Schranke Probleme

Euklidisches Rundreise-Problem nur mit Scheiben

Als LP formuliert sieht es so aus:

$$\begin{aligned}(P_1) \qquad \max \sum_{u\in V} 2r_u & \\ r_u + r_v &\leqq t(u,\, v), \text{ für alle Paare } \{u,\, v\} \\ r_u &\geqq 0 \quad \forall n \in V\end{aligned}$$

Das Duale LP lautet folgendermaßen:

$$\begin{aligned}(D_1) \qquad \min \sum_{u,v} t(u,v)x_{uv} & \\ \sum_{v\neq u} x_{uv} &\geqq 2 \quad \forall u \in V \\ x_{uv} &\geqq 0, \text{ für alle Paare } \{u,\, v\}\end{aligned}$$

Beispiel mit vier Städten:

$$\max \begin{pmatrix} 2 & 2 & 2 & 2 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}$$

$$s.t. \quad \begin{pmatrix} 1 & 1 & & \\ 1 & & 1 & \\ 1 & & & 1 \\ & 1 & 1 & \\ & 1 & & 1 \\ & & 1 & 1 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} \leqq \begin{pmatrix} t(1,2) \\ t(1,3) \\ t(1,4) \\ t(2,3) \\ t(2,4) \\ t(3,4) \end{pmatrix}$$

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} \geqq 0$$

Dual

$$\min t(1,2) + \ldots + t(3,4) \begin{pmatrix} x_{12} \\ \vdots \\ x_{34} \end{pmatrix}$$

$$\begin{pmatrix} x_{12} & \ldots & x_{34} \end{pmatrix} \begin{pmatrix} 1 & 1 & & & \\ 1 & & 1 & & \\ 1 & & & 1 & \\ & 1 & 1 & & \\ & 1 & & 1 & \\ & & 1 & 1 & \end{pmatrix} \geqq \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

$$x \geqq 0$$

Hinzufügen von Gräben

$$(P_2) \qquad \max \sum_{u \in V} 2r_u + \sum_{s \in M} 2w_s$$

$$r_u + r_v + \sum_{s \in M,\, |\{u,v\} \cap s| = 1} w_s \leqq t(u,\, v), \text{ für alle Paare } \{u,\, v\}$$

$$r_u \geqq 0 \quad \forall u \in V$$

$$w_s \geqq 0 \quad \forall s \in M$$

Das Duale LP lautet folgendermaßen:

$$
\begin{array}{lll}
(D_2) \quad \min \sum_{u,v} t(u,v) x_{uv} & & \\
\sum_{v \neq u} x_{uv} & \geqq & 2 \ \forall u \in V \\
\sum_{|\{u,v\} \cap s| = 1} x_{uv} & \geqq & 2 \ \forall s \in M \\
x_{uv} & \geqq & 0, \text{ für alle Paare } \{u, v\}
\end{array}
$$

Für jede Tour T erfüllt der Inzidenzvektor $\bar{X}$ die Restriktionen von (D_2).
$\overset{\text{Schwache Dualität}}{\Rightarrow}$ Die Optimallösung für (D_2) ist eine untere Schranke für die Länge einer Tour. Wenn man mehr Ungleichungen zu (D_2) hinzufügt erhält man eine bessere untere Schranke ($\rightarrow$ nächstes Semester).

Analoge Konstruktion für perfektes euklidisches Matching

O ist die Familie der Teilmengen von V ungerader Kardinalität.

$$
\begin{array}{lll}
(P_2) \quad \max \sum_{u \in V} r_u + \sum_{s \in O} w_s & & \\
r_u + r_v + \sum_{s \in O,\, |\{u,v\} \cap s| = 1} w_s & \leqq & t(u, v), \text{ für alle Paare } \{u, v\} \\
r_u & \geqq & 0 \ \forall u \in V \\
w_s & \geqq & 0 \ \forall s \in O
\end{array}
$$

Das Duale LP lautet folgendermaßen:

$$
\begin{array}{lll}
(D_2) \quad \min \sum_{u,v} t(u,v) x_{uv} & & \\
\sum_{v \neq u} x_{uv} & \geqq & 1 \ \forall u \in V \\
\sum_{|\{u,v\} \cap s| = 1} x_{uv} & \geqq & 1 \ \forall s \in O \\
x_{uv} & \geqq & 0, \text{ für alle Paare } \{u, v\}
\end{array}
$$

Für alle $x_{uv} > 0$ sind die entsprechenden dualen Ungleichungen mit Gleichheit erfüllt. Optimalität durch schwache Dualität und komplementären Schlupf.

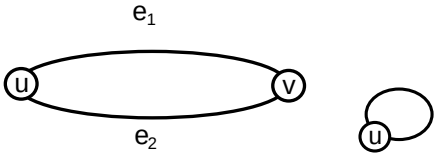
Kapitel 2

Optimale Bäume und Wege

Definitionen für ungerichtete Graphen

Ungerichteter Graph $G = (V, E)$
Knotenmenge $V = V(G)$
Kantenmenge $E = E(G)$
Relation $E \to V \times V$
$e = \{u, v\}$
„e ist inzident mit u und v “
„u und v sind adjazent“

parallele Kanten Schleife



Einfacher Graph:

- keine parallelen Kanten
- keine Schleifen

„$e = uv$“ „$e = vu$“

Subgraph H von G

$$V(H) \subseteq V(G)$$

$$E(H) \subseteq E(G)$$

$A \subseteq E \quad G \setminus A$: Subgraph von G nach Entfernen der Kanten in A

$B \subseteq V$ $G \setminus B$: Subgraph von G nach entfernen der Knoten in B und aller inzidenter Kanten. Kann auch als $G(V \setminus B)$ oder als durch $V \setminus B$ indizierter Subgraph bezeichnet werden.

„$G \setminus x$“ statt $G \setminus \{x\}$ für $x \in V$ oder $x \in E$

$n = |V|$ $m = |E|$ Konvention

Ein Subgraph H von G ist aufspannend
$\Leftrightarrow V(H) = V$

Weg P in G: Folge $v_0, e_1, v_1, \ldots, e_k, v_k$
$v_0, v_1, \ldots, v_k \in V, \quad e_1, e_2, \ldots, e_k \in E$
$e_i = v_{i-1}v_i$ für $1 \leqq i \leqq k$

„Weg von v_0 nach v_k“ oder „(v_0, v_k)-Weg“
Der Weg ist geschlossen, falls $v_0 = v_k$
Der Weg ist einfach, falls $v_0, v_1, \ldots, v_k$ distinkt
Ein Weg ist ein Kreis falls:

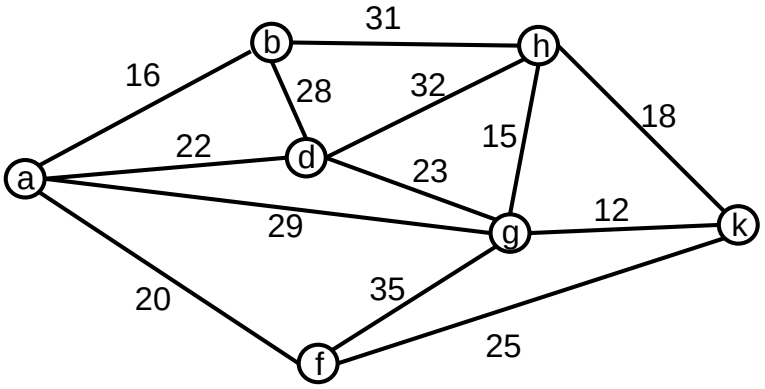
1. Geschlossenheit
2. $v_0, v_1, \ldots, v_k$ distinkt sind
3. $k \geqq 1$

G hat (u,v)-Weg $\Rightarrow$ G hat einen einfachen (u,v)-Weg
Länge von $P = v_0, e_1, v_1, \ldots, e_k, v_k$: k

G ist zusammenhängend :$\Leftrightarrow$ zu jedem Knotenpaar (u,v) aus V gibt es einen (u,v)-Weg

2.1 Minimal aufspanndende Bäume

Beispiel:



Dieses Netzwerk sei ein Kommunikationsnetzwerk und bilde Terminals ab und die den Kanten zugeordneten Zahlen entsprächen den Kosten für die direkte Verbindung zwischen den Terminals damit ergibt sich das

Verbindungs-Problem

Gegeben: Ein zusammenhängender Graph $G = (V,E)$ und positive Kosten für alle $e \in E$
Gesucht: Ein aufspannender, zusammenhängender Subgraph mit minimalen Kosten.

Lemma 2.1 *Eine Kante $e = uv \in E$ ist Kante eines Kreises in G genau dann, wenn es einen (u,v)-Weg in $G \setminus e$ gibt.*

$\rightarrow$ die Optimallösung des Verbindungs-Problems hat keine Kreise

Einen Graphen ohne Kreis nennt man auch Wald, einen zusammenhängenden Wald nennt man auch Baum.

Minimal aufspannendes Baum-Problem

Gegeben: Ein zusammenhängender Graph $G(V,E)$ $c_e \in R$ für alle $e \in E$
Gesucht: Der Aufspannende Baum in G mit minimalen Kosten (MSB) im Englischen (MST = minimal spanning tree). Das Verbindungs-Problem und das MSB-Problem sind äquivalent für $c_e > 0$.

Nun ein bisschen Wiederholung aus Informatik I:
Notation: $G = (V,E)$
$A \subseteq V$: $\delta(A) := \{uv \in E| \quad \underbrace{u \in A,\ v \in V \setminus A}_{\text{Kanten die auf den Schnitten liegen}} \quad \}$
$\delta(A)$ wird auch als „Schnitt in G“ bezeichnet (indiziert von $A \subseteq V$)
$\gamma(A) = \{u,v \in A,\ uv \in E\}$ ergibt alle Kanten in A

Prims Algorithmus für MSB

Initialisiere $H = (V(H), T)$ als $\{\{r\}, \varnothing\}$
while (H ist nicht aufspannender Baum) **do**
  Füge eine Kante minimaler Kosten aus $\delta(V(H))$ zu H
end while

In unserem Kommunikationsproblem erhält man nun, wenn man bei Knoten a anfängt:

Hinzugefügte Kante	Kosten
(a, b)	16
(a, f)	20
(a, d)	22
(d, g)	23
(g, k)	12
(g, h)	15

Kruskals Algorithmus für MSB

Sortiere E als $< e_1, e_2, \ldots, e_m >$, so dass $c_{e_1} \leqq c_2 \leqq \ldots \leqq c_{e_m}$
Initialisiere $H = (V, T)$ als $(V, \varnothing)$
for $i = 1$ to m **do**
  if Endknoten von e_i in verschiedenen Komponenten von H **Then** Füge e_i zu T hinzu.
end for

mit unserem Beispiel erhält man:

Hinzugefügte Kante	Kosten
(g, k)	12
(g, h)	15
(a, b)	16
(a, f)	20
(a, d)	22
(d, f)	23

Beide Algorithmen können mit einer Laufzeit $O(m \log n)$ implementiert werden.

Hier legte Prof. Jünger eine Folie auf mit Punkten um die herum ein Analogrechner Kreise „auf bläst“ bis zwei Kreise aneinander stoßen. Wenn zwei Kreise aneinander stoßen und zwischen den Komponenten noch keine Verbindung besteht so wird eine Kante dort eingefügt. Mag es jemand malen?

MSB und lineare Optimierung

Notation: Menge A, $p \in \mathbb{R}^A$, $B \subseteq A$

$$p(B) = \sum_{j \in B} p_j$$

LPMSB $\min c^T x$

$$\begin{aligned} x(\gamma(S)) &\leqq |S| - 1 \quad \forall S, \varnothing \neq S \subset V \\ x(E) &= |V| - 1 \\ x_e &\geqq 0 \ \forall e \in E \end{aligned}$$

In diesem Fall wird explizit nicht $x_e \in \{0,1\}$ gefordert, da die Ganzzahligkeits-Bedingung das Problem sehr schwer werden lässt.

Die Inzidenzvektoren x^0 von aufspannenden Bäumen T sind zulässige Lösungen und $c^T x^0 = c(T)$, d.h. gleicher Zielfunktionswert.
$\Rightarrow$ der Wert von (LPMSB) ist eine untere Schranke für die Kosten eines MSB.

Satz 2.2 *Sei x^0 der Inzidenzvektor eines MSB bezüglich der Kosten c_e. Dann ist x^0 eine Optimallösung von LPMSB.*
Beweis: Für eine Kanten-Teilmenge $A \subseteq E$ sei $\kappa(A)$ die Anzahl der Zusammenhangs-Komponenten des Subgraphen (V,A) von $G = (V,E)$:

(LPMSB') $\min c^T x$

$$\begin{aligned} x(A) &\leqq |V| - \kappa(A) \ \forall A \subset E \\ x(E) &= |V| - 1 \\ x_e &\geqq 0 \ \forall e \in E \end{aligned}$$

Beweis: Das (LPMSB') hat die gleiche zulässigen Lösungen wie das (LPMSB) d.h. auch die gleiche Optimallösung. In $x(A) \leqq |V| - \kappa(A)$ setzen wir speziell $A = \gamma(S)$ und erhalten:

$$\begin{aligned} x(\gamma(S)) &\leqq |V| - \underbrace{\kappa(\gamma(S))}_{\geqq |V \setminus S| + 1} \\ &\leqq |V| - |V \setminus S| - 1 \\ &= |S| - 1 \end{aligned}$$

Umgekehrt folgt $x(A) \leqq |V| - \kappa(A)$, denn:
Sei $A \subseteq E$ und $s_1, s_2, \ldots, s_k$ $(k = \kappa(A))$ die Knotenmengen der Zusammenhangs-Komponenten von (V,A).

$$\begin{array}{lrcl} \text{Dann gilt} & x(A) & \leqq & \sum_{i=1}^{k} x(\gamma(s_i)) \\ & & \leqq & \sum_{i=1}^{k} (|s_i|-1) \\ & & = & |V|-k \end{array}$$

Es reicht also zu zeigen x^0 ist optimal für (LPMSB'). Wir zeigen dies nun für ein x^0 das Ergebnis aus Kruskals Algorithmus ist. Im (LPMSB') können wir für die Zielfunktion auch schreiben $max - c^T x$. Damit erhalten wir als duales LP:

$$\begin{array}{lll} \text{(DLPMSB')} \quad \min \sum_{A \subset E \text{ so dass } e \in A} (|V| - \kappa(A)) y_A & & \\ \sum_{A \subset E \text{ so dass } e \in A} y_A & \geqq & -c_e \ \forall e \in E \\ y_A & \geqq & 0 \ \forall A \subset E \end{array}$$

y_E ist in diesem Fall nicht vorzeichen-beschränkt. Mittels Kruskals Algorithmus konstruieren wir nun eine Optimallösung von (DLPMSB').

Sei $e_1, e_2, \ldots, e_m$ die sortierte Folge der Kanten in Kruskals Algorithmus.

$$R_i := \{e_1, e_2, \ldots, e_i\} \quad (1 \leqq i \leqq m) \quad R_0 := \varnothing$$

Weiterhin definieren wir y^0:

$$\begin{array}{rcl} y^0_{R_i} & = & c_{e_{i+1}} - c_{e_i} \\ y^0_{R_m} & = & -c_{e_m} \\ y^0_A = 0 \text{ sonst} \quad (A \neq R_i) & & \end{array}$$

Es gilt nun (wg. Sortierung) $y^0_A \geqq 0 \quad \forall A \neq E$ und mit $e = e_i$:

$$\begin{array}{rcl} \sum_{A \subset E \text{ so dass } e \in A} y^0_A & = & \sum_{j=1}^{m} y^0_{R_j} = \sum_{j=i}^{m-1} (c_{e_{j+1}} - c_{e_j}) - c_{e_m} \\ & = & c_{e_{i+1}} - c_{e_i} + c_{e_{i+2}} - c_{e_{i+1}} + \ldots + c_{e_m} - c_{e_{m-1}} - c_{e_m} \\ & = & -c_{e_i} = -c_e \end{array}$$

$\Rightarrow$ alle „$\geqq$“ sind mit „=“ erfüllt.

$$\Rightarrow \begin{cases} y^0 \text{ ist zulässig für (DLPMSB')} \\ \text{kompl. Schlupf } x^0_e > 0 \Rightarrow \sum_{A \subset E \text{ so dass } e \in A} y_A = -c_e \ \forall e \in E \end{cases}$$

Noch zu zeigen:

$$\text{Kompl. Schlupf: } y^0_A > 0 \Rightarrow x^0(A) = |V| - \kappa(A)$$

Sei $y_A > 0 \Rightarrow A = R_i$ für $1 \leqq i \leqq m$
Annahme: $x^0(R_i) < |V| - \kappa(R_i)$
$x^0(R_0) = |V| - \kappa(R_0) \Rightarrow \exists e \in R_i$: Hinzufügen von e zu $T \cap R_i$ erniedrigt die Zahl der Zusammenhangs-Komponenten von $(V, R_i \cap T)$
$\Rightarrow e \notin T$ $\lightning$ denn Kruskal hätte e gewählt, d.h. $e \in T$

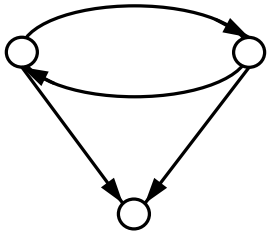
Damit ergibt sich nun insgesamt: x^0, y^0 sind zulässig und erfüllen die komplementären Schlupfbedingungen.
$\Rightarrow x^0$ ist optimal für (LPMSB')
$\Rightarrow x^0$ ist optimal für (LPMSB)

Bemerkung: Dies beweist die Korrektheit von Kruskals Algorithmus.

2.2 Kürzeste Wege

Anwendung: Kürzeste Fahrstrecke von A nach B in einem Straßen-Netzwerk.
Es gibt auch Einbahnstraßen $\rightarrow$ gerichtete Graphen
Gerichteter Graph $G = (V, E)$ auch genannt „Digraph“ mit
$V = V(G)$ Knoten
$E(G)$ gerichtete Kanten oder auch Bögen (arc)
Für $e \in E$: $t(e)$: Ende von e (tail)
$h(e)$: Spitze von e (head)

Die anderen Begriffe gelten analog zu nicht gerichteten Graphen.
Den zu einem gerichteten Graphen zugeordneten ungerichteten Graphen erhält man durch das weglassen der Richtung.
Parallele Kanten sind Kanten die dieselbe Spitze und und die gleichen Enden haben. Antiparrallele Kanten sind zwei Kanten bei denen die eine dort die Spitze hat wo die andere das Ende und umgekehrt.



Als Digraph ist dieser Graph ein einfacher Graph als ungerichteter Graph nicht.

$e = uv \rightarrow t(e) = u, h(e) = v$

Kanten e_i eines Weges $P = v_0, e_1, v_1, e_2, \ldots e_k, v_k$. Der Weg heißt vorwärts gerichtet falls $t(e_i) = v_{i-1}$ und $h(e_i) = v_i$. Analog ist ein gerichteter Kreis ein Kreis bei dem alle Kanten vorwärts gerichtet sind.

Kürzeste Wege Problem

Gegeben: Digraph G, Knoten v aus V, Gewichte $c_e \in \mathbb{R}\ \forall e \in E$

Gesucht: Für alle $v \in V$ einen gerichteten Weg von r nach v mit minimalen Kosten, falls ein solcher existiert.

Konvention: Zur Vermeidung der Nicht-Existenz eines (r,v)-Weges fügt man Kanten $rv \in E$ mit sehr hohem Gewicht hinzu.

Ohne Beschränkung der Allgemeinheit kann von einem einfachen Graphen ausgegangen werden, da man alle parallelen Kanten auf die günstigste Kante reduziert.

Die Grundidee der folgenden Algorithmen sieht dabei folgendermaßen aus:

- $\forall v \in V$ gerichteter Weg von r nach v mit Kosten y_v
- Finde eine Kante $vw \in E$ mit $y_v + c_{vw} < y_w$: dann existiert ein kürzester Weg (in Bezug auf alle Wege die wir bisher kennen) von v nach w mit Kosten $y_v + c_{vw}$.
- Sind die y_v $(v \in V)$ Kosten für kürzeste (u,v)-Wege so gilt:
  $(*)\ y_v + c_{vw} \geqq y_w$
  $y \in R^v$ heißt zul. Potential falls $(*)$ gilt und $y_r = 0$ Falls $y_r \neq 0$, einfach y_v durch $y_v - y_r$ ersetzen.

Lemma 2.3 *Sei y ein zulässiges Potential und P ein (r,v)-Weg (gerichtet), dann gilt: $c(P) \geqq y_v$*

Beweis: Sei $P = v_0, e_1, v_1, e_2, \ldots e_k, v_k$, mit $v_0 = r$, $v_k = v$ so gilt:

$$c(P) = \sum_{i=1}^{k} c_{e_i} \geqq \sum_{i=1}^{k} (y_{v_i} - y_{v_{i-1}}) = y_{v_k} - \underbrace{y_{v_0}}_{=0} = y_{v_k} \quad \text{q.e.d}$$

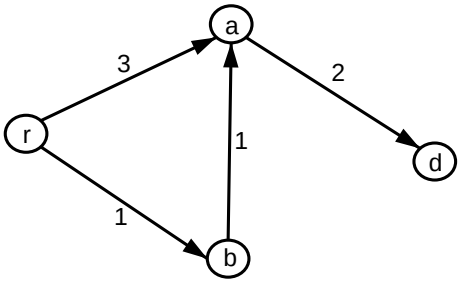
Lemma 2.4 *Es existiert eine Lösung des Kürzeste Wege Problems, die nur Kanten eines aufspannenden Baumes enthält*

Beweis: Subwege von kürzesten Wegen sind kürzeste Wege
$\Rightarrow \forall v \in V,\ v \neq r$ genügt die letzte Kante eines gerichteten (r,v)-Weges

Fords Algorithmus

Initialisiere:
$y_r := 0,\ \ y_v := \infty\ \ \forall v \neq r$
$p_r := 0,\ \ p_v := -1\ \ \forall v \neq r\ \ \ (0,-1) \notin V$
while (y kein zul. Potential) **do**
  Finde eine Kante vw mit $y_v + c_{vw} < y_w$
  setze $y_w := y_v + c_{vw}$ und $p_w := v$
end while

Beispiel 1



	Start		$vw = ra$		$vw = rb$		$vw = ad$		$vw = ab$	
	y	p	y	p	y	p	y	p	y	p
r	0	0								
a	∞	-1	3	*r*					2	*b*
b	∞	-1			1	*r*				
d	∞	-1					5	*a*	4	*a*

Eigenschaften

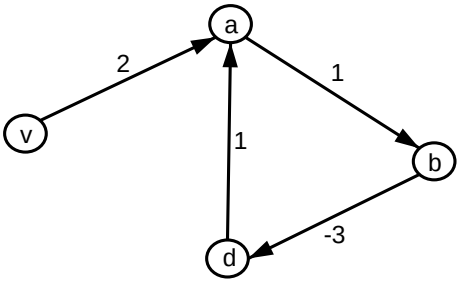
1. Nach jeder Iteration gilt:

$$y_v \geqq y_{p(v)} + c_{p(v)v}$$

   Dies gilt mit „=“ wenn y_v und $p(v)$ gesetzt werden, nachher kann $y_{p(v)}$ nur kleiner werden.

2. Nach Terminierung gilt:

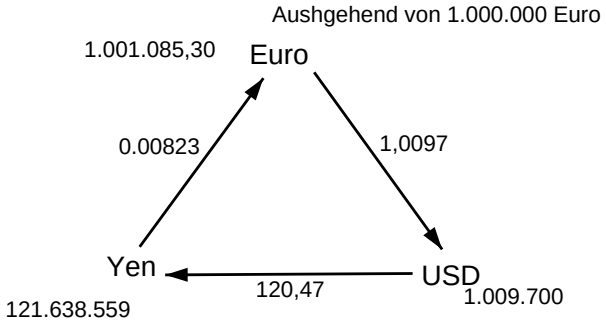
$$y_v \geqq y_{p(v)} + c_{p(v)v}$$



	Start		$vw = ra$		$vw = ab$		$vw = bd$		$vw = da$		$vw = ab$		
	y	p	y	p	y	p	y	p	y	p	y	p	
r	0	0											
a	∞	-1	2	r					1	d			...
b	∞	-1			3	a					2	a	
d	∞	-1					0	b					

Hier terminiert das Verfahren nicht, da $y_a, y_b, y_c, \to -\infty$.
Grund: negativer Kreis abd.

Der Algorithmus soll dies erkennen. Es gibt aber auch Anwendungen für negative Kreise, z.B. im Währungstausch.



Arbitrage-Möglichkeiten mit den Kursen vom 18.11.2002

Gesucht sind also Folgen $v_0, v_1, \ldots, v_k = v_0$ mit:

$$\prod_{i=1}^{k} r_{v_{i-1}v_i} > 1$$

Kosten: $c_{uv} = -\log r_{uv}$

Gewinn falls $\sum_{i=1}^{k} \log r_{v-1v_i} = -\log \prod_{i=1}^{k} r_{v_{i-1}v_i} < 0$

Lemma 2.5 *Hat (G,c) keinen negativen Kreis, so gilt nach jeder Iteration von Fords Algorithmus:*

1. *Ist $y_v \neq \infty$ so ist y_v Kosten eines einfachen (r,v)-Weges*
2. *Ist $p(v) \neq -1$ so definiert p einen einfachen (r,v)-Weg mit Kosten höchstens y_v*

Beweis:

1. Sei y_v^j der Wert y_v nach j Iterationen.
   $y_v^j \neq \infty \rightarrow y_v^j$ Kosten eines (r,v)-Weges P.
   Annahme: P ist nicht einfach $\Rightarrow \exists$ Folge $v_0, v_1, \ldots, v_k$, $v_k = v_0$
   Iterationszahlen $q_0 < q_1 < \ldots < q_k$ mit $y_{v-1}^{q_{i-1}} + c_{v_{i-1}v_i} = y_{v_i}^{q_i}$
   und Kosten des geschlossenen Weges:

$$\begin{aligned} \sum_{i=1}^{k} c_{v_{i-1}v_i} &= \sum_{i=1}^{k} (y_{v_i}^{q_i} - y_{v_{i-1}}^{q_{i-1}}) \\ &= y_{v_k}^{q_k} - y_{v_0}^{q_0} < 0 \end{aligned}$$

   y_{v_k} wurde in Iteration q_k erniedrigt. Also haben wir einen negativen Kreis und damit Widerspruch.

2. Annahme P ist nicht einfach
   $\Rightarrow \exists v_0, v_1, \ldots, v_k$, $v_k = v_0$
   $p(v_i) = v_{i-1} \quad (1 \leqq i \leqq k)$
   Die Kosten des geschlossenen Weges sind $\leqq 0$, da $c_{p(v)v} \leqq y_v - y_{p(v)}$
   Die letzte Änderung ergibt: $p(v)$ mit Verkleinerung von $y_p(v_i) \Rightarrow$ „<“
   $\Rightarrow$ negativer Kreis, Widerspruch

Die Kosten sind höchstens y_v: Es gebe einen einfachen (r,v)-Weg und sei $v_0, e_1, v_1, \ldots, e_k, v_k$ mit $v_0 = r$, $v_k = v$, $p(v_i) = v_{i-1}, \quad (1 \leqq i \leqq k)$
Kosten $\sum_{i=1}^{k} c_{v_{i-1}v_i} \leqq \sum_{i=1}^{k} (y_{v_i} - y_{v_{i-1}}) = y_v - y_r = y_v$ q.e.d.

Satz 2.6 *Hat (G,c) keinen negativen Kreis, so terminiert Fords Algorithmus nach endlich vielen Schritten mit einem korrekten Ergebnis*

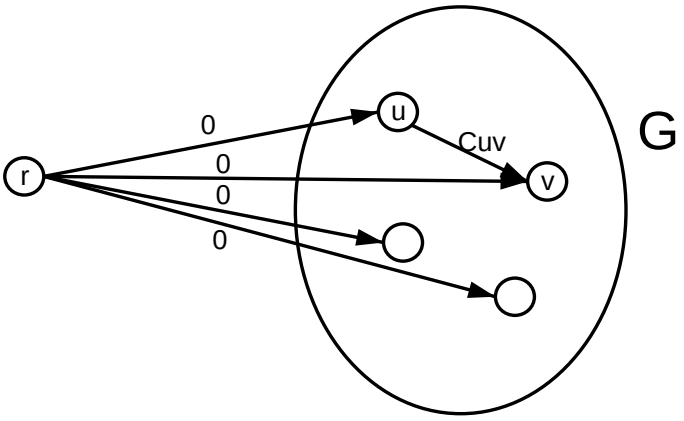
Beweis:
$\exists$ nur endlich viele einfache gerichtete Wege in G.
Lemma 2.5 $\Rightarrow$ Es gibt nur endlich viele mögliche Werte für die y_v. In jedem Schritt wird ein y_v erniedrigt und keines erhöht $\rightarrow$ endlich viele Schritte.

Bei Terminierung definiert p einfache (r,v)-gerichtete Wege für alle $v \in V$ mit Kosten höchstens y_v.
Lemma 2.3 $\Rightarrow$ Es gibt keine kürzere (r,v)-gerichtete Wege.

Satz 2.7 *(G,c) hat ein zulässiges Potential genau dann, wenn es keinen negativen Kreis gibt.*

Beweis:
„⇒“ negativer Kreis ⇒ kein zulässiges Potential

„⇐“ (G,c) habe keinen negativen Kreis:
Man konstruiere folgenden Graphen (G',c'):



r ist jeweils mit allen Knoten von G durch gerichtete Kanten des Gewichtes 0 verbunden. Damit hat (G',c') keinen negativen Kreis. Anwendung von Fords Algorithmus auf (G',c') gibt zulässiges Potential für (G',c'). Diese ist auch zulässiges Potential für (G,c). q.e.d.

Satz 2.7 Gilt auch ohne Annahme der Existenz von (r,v)-Wegen.
Problem: nicht einfache beliebig kurze (r,v)-gerichtete Wege.
Die Suche nach einfachen (r,v)-Wegen (endlich viele!) ist jedoch NP-schwierig.

Satz 2.8 *Falls $c_e \in \mathbb{Z}$ für alle $e \in E$:*

$$c = 2\max_{e\in E}|c_e| + 1$$

und (G,c) hat keinen negativen Kreis so terminiert Fords Algorithmus nach höchstens cn^2 Iterationen

Beweis: Übungsaufgabe

2.2.1 Zulässige Potentiale und lineare Optimierung

Satz 2.9 *Sei G ein gerichteter Graph, $r,s \in V$, $c \in \mathbb{R}^E$*
Falls für alle $v \in V$ ein kürzester (r,v)-Weg existiert, so gilt:

$$\begin{aligned} & \min\{c(P) | P \text{ ist } (r,s) \text{ gerichteter Weg}\} \\ = \; & \max\{y_s | y \text{ ist ein zulässiges Potential}\} \end{aligned}$$

Beweis: Analyse von Fords Algorithmus

Als duales LP formuliert erhält man (ohne Bedingung $y_r = 0$):

$$\begin{aligned} &\text{(DLPKW)} \qquad \max y_s - y_r \\ &y_w - y_v \leqq c_{vw} \quad \forall v,w \in E \end{aligned}$$

Für das LP definieren wir:

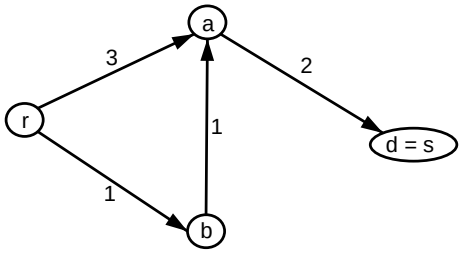
Sei $b_v = \begin{cases} 1 \text{ falls } v = s \\ -1 \text{ falls } r = v \\ 0 \text{ sonst} \end{cases}$

$$\begin{aligned} &\text{(LPKW)} \qquad \min \sum_{e \in E} c_e x_e \\ \sum_{w \in V,\ vw \in E} x_{wv} - \sum_{w \in V,\ wv \in E} x_{vw} &= b_v \quad \forall v \in V \\ x_{vw} &\geqq 0 \quad \forall vw \in E \end{aligned}$$

Jedem (r,s)-gerichteten Weg P entspricht eine zulässige Lösung für (LPKW) mit Wert $c(P)$. $x_e^P =$ Anzahl des Vorkommens von e in P. ($\in \{0,1\}$ falls P einfach ist).

Verhalten des Simplexalgorithmus beim LPKW

Gegeben sei folgender Graph:



Dann ergibt sich folgender Aufbau:

		r a	r b	b a	ad		b
	r	-1	-1				-1
A	a	1		1	-1	=	
	b		1	-1			
	d				1		1

c	3	1	1	2

$$\begin{aligned} &\text{(LKWP)} \quad \min c^T x \\ Ax &= b \\ x &\geqq 0 \end{aligned}$$

Der Simplexalgorithmus betrachtet nun eine Folge von Basislösungen wobei eine Basis einer „max. linear unabh. Kantenteilmenge“ entspricht. Es ist also eine „Spaltenbasis“ von A.

In jeder Iteration hat man eine Basis B und x, y mit $x_e = 0 \ \forall e \notin B$.

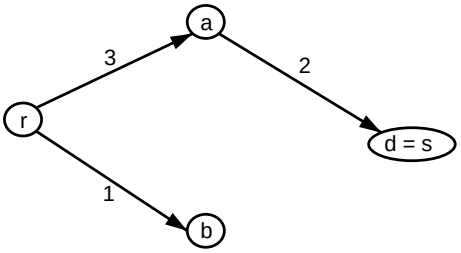
$$y_w - y_v = c_{vw} \forall vw \in B$$

Satz 2.10 *Sei G ein zusammenhängender gerichteter Graph, A die Knoten-Kanten Inzidenzmatrix.*

$A_{\cdot J}$, $J \subseteq E$ ist eine Spaltenbasis von A genau dann wenn J die Kantenmenge eines Aufspannenden Baumes in G ist.

Beweis: Übungsaufgabe

Baum T_1



Ein Basiswechsel Beispiel.

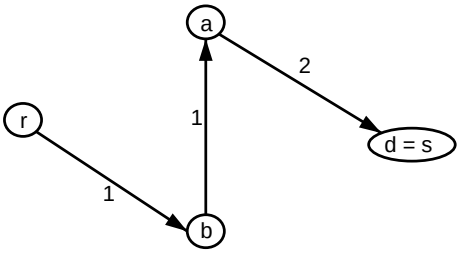
		r a	r b	ad		y
	r	-1	-1			0
B	a	1		-1	=	3
	b		1			1
	d			1		5

c	3	1	2

reduzierte Kosten: $c_{ba} - y^T A_{\cdot ba} = 1 - 2 = -1 < 0$

$\rightarrow$ *ba* geht in die Basis, *ra* raus (hat die Spitze am selben Punkt).

Baum T_2



		r a	b a	ad		y
	r	-1				0
B	a		1	-1	=	2
	b	1	-1			1
	d			1		4

c | 3 | 1 | 2 |

reduzierte Kosten: $3-2=1>0$

Die Gleichungen $y_w - y_v = c_{vw} \forall vw \in B$ gelten in Fords Algorithmus nicht immer, d.h. jede Iteration dieses speziellen Simplex Verfahrens entspricht einer Folge von Iterationen in Fords Algorithmus (Änderung des Baumes gefolgt von y-Änderungen bei gleichem Baum.

2.2.2 Varianten des Ford Algorithmus

Korrekturschritte in konstanter Zeit. Laufzeit abhängig von der Reihenfolge $S =< f_1, f_2, \ldots, f_e >$ der gewählten Kanten. Bei schlechter Wahl erhält man eine Laufzeit von $\Theta(2^{|E|})$ (Übungen)

Ein (r,v)-Weg $P = \quad r = v_0, e_1, v_1, \ldots, c_k, v_k = v$ ist in S eingebettet falls $e_1, e_2, \ldots_k$ eine geordnete Subsequenz von S ist.

Lemma 2.11 *Wenn Fords Algorithmus die Sequenz S benutzt, so gilt für jedes $v \in V$ und jeden (r,v)-Weg der in S eingebettet ist $y_v \leqq c(P)$*

Beweis: Sei $P = \underbrace{v_0}_{=r}, e_1, v_1, \ldots, c_k, \underbrace{v_k}_{=v}$
Nach erster Iteration mit $e_1 : y_{v_1} \leqq y_r + c_{e_1}$
Nach zweiter Iteration mit $e_2 : y_{v_2} \leqq y_{v_1} + c_{e_1} + c_{e_2}$
Nach dritter Iteration mit $e_k : y_v = y_{v_k} \quad \leqq \quad y_{v_{k-1}} + c_{e_k}$

$$\leqq \quad \sum_{i=1}^{k} c_{e_i} = c(P). \quad \text{q.e.d}$$

Also suchen wir S mit kürzesten eingebetteten (r,v)-Wegen für alle $v \in V$ S „kurz“ da die Laufzeit proportional zur Länge.

Ford-Bellmann-Algorithmus

Jeder einfache gerichtete Weg in G ist eingebettet in $< s_1, s_2, s_3, \ldots, s_{n-1} >$ mit $s_i =$ irgendeine Permutation von $E = \{e_1, e_2, \ldots, e_m\}$

Initialisiere y, p;
$i = 0$;
while ($i < n$ und y kein zulässiges Potential) **do**
  i := i+1;
  for $e = e_1$ to e_m **do**
    $v = t(e)$; $w = h(e)$;

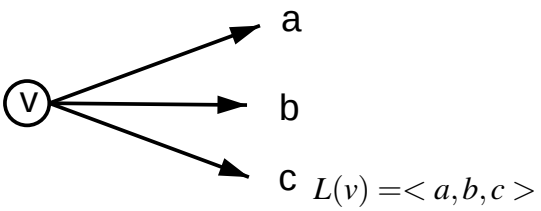
if $(y_v + c_{vw} < y_w)$ **then**
$y_w := y_v + c_{vw}$; $p_w := v$
end if
end for
end while
if $(i < n)$ **then**
„Kürzester Weg gefunden“
else
STOP „es existiert ein negativer Kreis“
end if

Satz 2.12 *Der Ford-Bellmann Algorithmus ist korrekt und hat Laufzeit* $O(nm)$.

Beweis: Korrektheit wg. Lemma 2.11, Laufzeit offensichtlich.

Verbesserungen der Laufzeit sind nur bei speziellen Annahmen über G und c möglich. Vorgestellt werden nun Verbesserungen in der Praxis:

Vorwärts-Stern-Datenstruktur



$L(v) =< a, b, c >$

Weiterhin benötigen wir:
$Q =< f_1, f_2, \ldots, f_e >$ Schlange (Queue) von Kunden.
$v \Leftarrow Q$ ergibt $v = q_1,\ Q =< f_2, f_3, \ldots, f_e >$
$Q \Leftarrow v$ ergibt $Q =< f_1, f_2, \ldots, f_e, v >$
Zusätzlich Inzidenzvektor: isinQ Q[v]
Anzahl der Knoten in Q: numQ

dies erlaubt delete, append $Q \stackrel{?}{=} \varnothing,\ \ v \stackrel{?}{\in} Q$ in konst. Zeit (keine Negativen Kreise).

Initialisiere $y,\ p$;
$Q :=< v >$;
while $(Q \neq \varnothing)$ **do**
$v \Leftarrow Q$;
for all $(w \in L(v))$ **do**
if $(y_v + c_{vw} < y_w)$ **then**
$y_w := y_v + c_{vw}$; $p_w := v$ }
if $(w \notin Q)$ **then**

```
            Q ⇐ w;
          end if
        end if
      end for
    end while
```

Azyklische Graphen

Keine gerichteten Kreise, insbesondere keine negativen Kreise. Kompatible Knoten Sequenz $< v_1, v_2, \ldots, v_n >$ für alle Knoten $v_i, v_j \in E$ gilt: $i < j$ kann in Zeit $O(m)$ gefunden werden (Topologische Sortierung, siehe Übungsaufgabe 20).

OBdA: sei $r = v_1$

$$r = v_i \Rightarrow v_j \quad (j < i) \text{ nicht erreichbar}$$

wie oben, aber:

```
  ⋮
  for i = 1 to n do
    for all (w ∈ L(v_i)) do
      (jeder Pfad ist eingebettet)
      if (y_v + c_vw < y_w) then
        y_w := y_v + c_vw; p_w := v
      end if
    end for
  end for
```

Satz 2.13 *In azyklischen Graphen kann das kürzeste Wege Problem in Zeit $O(m)$ gelöst werden*

Nicht negative Kosten:

$$c_e \geqq 0 \ \forall e \in E$$

Wie vorhin, aber Knotensequenz dynamisch bestimmt nach $v_1, v_2, \ldots, v_i$.
Wähle $v_{i+1} = v \in \{v_1, v_2, \ldots, v_i\}$ mit minimalem y_v.

Lemma 2.14 *Für alle $w \in V$ sei y'_w der Wert von y_w wenn w gewählt wird. Wird u vor v gewählt. So gilt $y'_u \leqq y'_v$.*

Beweis: Annahme $y'_v < y'_u$ und v ist der erste Knoten bei dem das passiert.
Als u gewählt wurde galt: $y'_u = y_u \leqq y_v$. Nach Wahl von u und vor Wahl von v wurde

$y_v < y'_u$, sagen wir das letzte Mal bei Wahl von w

$$v_1, v_2, \ldots, u, \ldots, w, \ldots, v, \ldots$$

Also fand bei w die letzte Erniedrigung von y_v statt. Zu diesem Zeitpunkt wurde gesetzt:

$$y_v := y'_w + c_{wv}$$

Aber $\left.\begin{array}{rcl} y'_w & > & y'_u \quad \text{(erster Fehler bei } v\text{)} \\ c_{vw} & \geqq & 0 \end{array}\right\} y'_v \geqq y'_u \quad \lightning$

Dijkstras Algorithmus

Initialisiere y, p;
$S := V$
while $(S \neq \varnothing)$ **do**
  wähle $v \in S$ mit y_v minimal
  entferne v aus S;
  Bearbeite Vorwärtsstern von v (aber nur Kanten mit vw mit $w \in S$!)
end while

Satz 2.15 *Falls $c_e \geqq 0 \ \forall e \in E$, so ist Dijkstras Algorithmus korrekt und läuft in $O(n^2)$*

Beweis:
Am Ende gilt $(*)$ $y_v + c_{vw} \geqq y_w \ \forall\, vw \in E$

Annahme: $(*)$ gilt nicht
$(*)$ galt nach Bearbeitung von Knoten v $\Rightarrow y_v$ wurde später erniedrigt, etwa als q bearbeitet wurde.
$\Rightarrow y_v = \underbrace{y'_q}_{\geqq y'_v} + \underbrace{c_{qv}}_{\geqq 0} \geqq y'_v$ also nicht erniedrigt $\lightning$
denn wenn q nach v bearbeitet wurde $\overset{\text{Lemma 2.14}}{\Rightarrow} y'_q \geqq y'_v$
$w \notin S$: $y_v \geqq y_w \overset{\text{Lemma 2.14}}{\Rightarrow} y_v + \underbrace{c_{vw}}_{\geqq 0} \geqq y_{wv}$ also hier kein Test nötig.

Laufzeit $O(m)$ für Vorwärtsstern plus die Zeit zum Finden von v mit minimalem y_v. Naiv implementiert erhält man damit eine Laufzeit von $O(n^2)$

Aus der Informatik I. Für Dünne Graphen mit Heaps $O(m \log n)$. Noch besser geht es mit Fibonacci Heaps (Wer hat die Fibonacci Beweise nicht geliebt?): $O(n \log n + m)$.

Wie dem auch sei, sei $S(n,m)$ die Zeit, die benötigt wird.

Beobachtung:
Kennt man ein zulässiges Potential y so kann man den Kostenvektor c nach $c' \geqq 0$ transformieren: $c'_{vw} = c_{vw} + y_v - y_w (\geqq 0)$ denn $\forall (r,s)$-Wege P: sind die Kosten:

$$c'(P) = c(P) + y_r - y_s$$

Anwendung: Kürzeste (r,v)-Wege für alle Paare (r,v). Allgemein beste Methode: n mal wie gehabt.
Zeit $O(n \cdot S(n,m))$, wenn $c \geqq 0$, also $O(n^2 m)$ im allgemeinen Fall.

Verbesserung von $O(n^2 m)$: Wenn Ford Bellman einmal ausgeführt ist liefert er uns ein zulässiges Potential für jeden Punkt.

$$\left.\begin{array}{l}\text{Transformation s.o.} \\ (n-1) \text{ mal Dijkstra}\end{array}\right\} \text{zusammen } O(n \cdot S(n,m))$$

Einheitskosten und Breitensuche

Finde (r,v)-Wege mit minimaler Kantenzahl.

Lemma 2.16 *Falls $c_e = 1$ für alle $e \in E$, so ist der erste endliche Wert für y_v auch der endgültige. Wird y_v vor y_w zugewiesen, so gilt $y_v \leqq y_w$.*

Beweis: klar für $v = r$. Sei $v \neq r$ und v später gewählt als Knoten $u \Rightarrow y_v \geqq y_u$ (Lemma 2.14).
zu endgültig: Erster endliche Wert für y_v wird gesetzt zu y_{w+1} [Knoten w wird vor v gewählt]

Behauptung y_v wird nicht mehr geändert
Annahme: bei Wahl eines anderen Knotens z.B. q wird y_v erniedrigt zu y'_v.
Situation: $\ldots w \ldots v \ldots q$
q wird nach w gewählt $\Rightarrow y_q \geqq y_w$
$y'_v = y_q + 1 \geqq y_w + 1 = y_v$ $\lightning$ zu erniedrigt.

Satz 2.17 *Bei Einheitskosten kann Dijkstras Algorithmus mit Laufzeit $O(m)$ implementiert werden.*

Beweis: Wegen Lemma 2.16 kann man den nächsten Knoten aus einer Schlange wählen (y_w sind überflüssig).

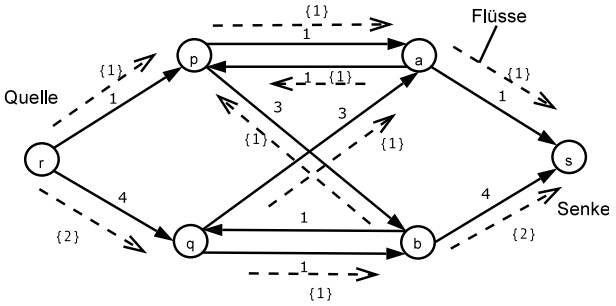
Dieser Algorithmus heißt auch Breitensuche.

```
Initialisiere p;
Q =< r >;
while (Q ≠ ∅) do
   v ⇐ Q;
   for all (w ∈ L(v)) do
      if (p(w) = −1) then
         Q ⇐ w;
         p(w) = v;
      end if
   end for
end while
```

Kapitel 3

Maximale Flussprobleme

3.1 Flüsse und Schnitte



Die Flüsse können nun als Kollektion von einfachen (r,s)-Wegen (die nicht notwendigerweise verschieden sind) betrachtet werden. In der Grafik sind allerdings nur die kumulierten Flüsse dargestellt. $\{p_1, p_2, \ldots p_k\}$ mit:

1. jedes $e \in E$ ist höchstens u_e

2. die Anzahl der Wege werden maximiert

$$x_e = |\{i | P_i \text{ benutzt } e\}|$$

Bedingungen für x:

$$(*) \quad \sum_{w \in V,\, wv \in E} x_{wv} - \sum_{w \in V,\, vw \in E} x_{vw} = 0$$

$$(**) \quad 0 \leqq x_{vw} \leqq u_{vw} \forall\, vw \in E$$

x_{vw} ist ganzzahlig $\forall vw \in E$. Die Zielfunktion lautet:

$$max \sum_{w \in V,\, ws \in E} x_{ws} - \sum_{w \in V,\, sw \in E} x_{sw} = k \ (= \text{Flusswert})$$

Damit haben wir das Maximum ganzzahlige Flussproblem formuliert.

$x \in R^E$ heißt Fluss, falls x $(*)$ erfüllt.
$x \in R^E$ heißt zulässiger Fluss, falls x $(**)$ erfüllt.

$f_x(v) := \sum_{w \in V,\, wv \in E} x_{wv} - \sum_{w \in V,\, vw \in E} x_{vw}$ Dies nennt man auch „Nettofluss“ oder „Überschuss“ bei v

$(*) = f_x(v) = 0$ ist die „Flusserhaltungsbedingung“ und f_s ist der Flusswert.

Lemma 3.1 *Eine Familie $\{P_1, P_2, \ldots, P_k\}$ von (r,s)-Wegen mit $|\{i | P_i \text{ benutzt } e\}| \leqq u_e$ existiert genau dann wenn ein ganzzahliger zulässiger (r,s)-Fluss mit Flusswert k existiert.*

Beweis: $\exists$ Familie von Wegen $\Rightarrow \exists$ ein Fluss, klar.
„$\Leftarrow$“ Gegeben sein ein Fluss $x \in R^e$ mit Flusswert k. Entferne Kreise:

$$x_e := x_e - 1 \ \ \forall\, e \in \text{ Kreis } C$$

$\Rightarrow$ Flusswert k bleibt.

Finde $P_1 : v \ldots s$ (mit alle n Kanten > 0) $x_e := x_e - 1 \forall\, e \in P_1$. $\rightarrow$ Flusswert ist $k-1$.

Finde P_k

Maximales Flussproblem

$$\begin{aligned} \max f_x(s) & \\ f_x(v) &= 0 \ \ \forall v \in V \setminus \{r,s\} \\ 0 \leqq x_e &\leqq u_e \ \ \forall\, e \in E \end{aligned}$$

Für einen Kreisfluss der Menge α gilt: $0 < \alpha \leqq \min\{u_e, e \in C\}$

Lemma 3.2 *Jeder zulässige (r,s)-Fluss ist die Summe von höchstens m Kreis und Weg-Flüssen.*

Beweis: Analog zu 3.1 mit Kreis und Wegeflüssen.

Maximale Flüsse und minimale Schnitte

Definition des gerichteten Schnittes R:

$$R \subseteq V,\ \delta(R) = \{vw | vw \in E,\ v \in R,\ w \notin R\}$$

Ein (r,s)-gerichteter Schnitt: $r \in R,\ s \notin R$
$A \subseteq V$, $\bar{A} := V \setminus A$ $\delta(v)$ für $\delta(\{v\})$, $\delta(\bar{v})$ für $\delta(\{\bar{v}\})$.

Lemma 3.3 *Für jeden (r,s)-Schnitt $\delta(R)$ und jeden (r,s)-Fluss x gilt $x(\delta(R))$ - $x(\delta(\bar{R})) = f_x(s)$*

Im Beispiel: $R = \{r,q,p\} \Rightarrow 4 - 1 = 3$

Beweis:

$$\begin{aligned} \sum_{v \in \bar{R} \setminus \{s\}} f_x(v) &= 0 \\ f_x(s) &= f_x(s) \\ \text{addiert:} \quad \sum_{v \in \bar{R}} f_x(v) &= f_x(s) \end{aligned}$$

Wenn Kante e beide Endpunkte in R hat:

$$\begin{aligned} e = vw,\ v,w \in R &: \text{tritt nicht auf, 0-Koeffizient in Summe} \\ v,w \in \bar{R} &: \left.\begin{array}{l} v\text{-Gleichung, Koeffizient -1} \\ w\text{-Gleichung, Koeffizient 1} \end{array}\right\} \text{Koeffizient 0} \\ v \in R,\ w \notin R &: \text{Koeffizient 1} \\ v \notin R,\ w \in R &: \text{Koeffizient -1} \end{aligned}$$

$f_x(v) = \sum_{wv \in E} x_{wv} - \sum_{vw \in E} x_{vw}$
$\Rightarrow \sum_{v \in R} f_x(v) = x(\delta(R)) - x(\delta(\bar{R}))$ q.e.d.

Korollar 3.4 *Für jeden zulässigen (r,s)-Fluss x und jeden (r,s)-Schnitt $\delta(R)$ gilt:*

$$f_x(s) \leqq u(\delta(R)) \qquad u = \textit{Kapazität}$$

Im Beispiel: $R = \{r,p,q\} \rightarrow f_x(s) \leqq 8$

Beweis:

$$\begin{aligned} x(\delta(R)) &\leqq u(\delta(R)) \\ -x(\bar{R})) &\leqq 0 \\ \overset{3.3}{\Rightarrow} f_x(s) &\leqq u(\delta(R)) \quad \text{q.e.d.} \end{aligned}$$

Dies ist ähnlich zur schwachen Dualität.

Satz 3.5 *Max Flow Min Cut Theorem*
Gibt es einen maximalen (r,s)-Fluss, so gilt

$$\max\{f_x(s)|x \text{ ist zul. } (r,s)\text{-Fluss}\} = \min\{u(\delta(R))|\delta(R) \text{ ist ein } (r,s)\text{-Schnitt}\}$$

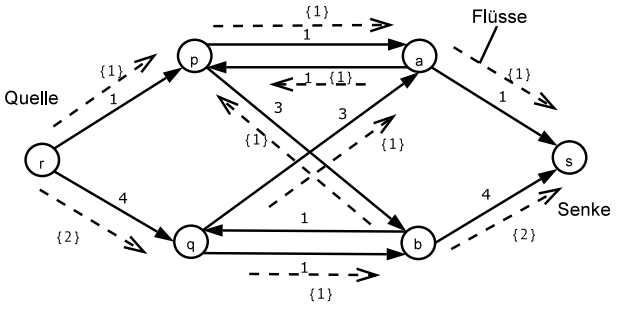
(entspricht der starken Dualität)

Beweisidee:

1. Finde (r,s)-Weg P mit $x_e < u_e \forall\, e \in P$
2. Erhöhe Fluss (alle $x_e \in P$) um $\min\{u_e - x_e | e \in P\}$

Im Beispiel: kein maximaler Fluss, trotzdem funktioniert diese Idee nicht.

Generalisierung: $\forall$ Vorwärtskanten $x_e < u_e$ und $\forall$ Rückwärtskanten $x_e > 0$



Man betrachte den Weg r,q,a,p,b,s und setze Kante p,a zu 0.
$\rightarrow$ p,b kann auf 2 gesetzt werden, b,s auf 3, r,q auf 3 und q,a auf 2.
Nun $R = \{r,q,a\} = 4$, also ist 4 der optimale Fluss. (Der Schnitt R muss immer die Quelle enthalten).

Beweis: Mit Korollar 3.4 ist zu zeigen:

$$\exists \text{ zul. Fluss x}, \exists \text{ Schnitt } \delta(R): \quad f_x = u(\delta(R))$$

Sei x maximal zulässiger Fluss
Def: $R := \{v \in V | \exists\, x \text{ erhöhender } (r,v)\text{-Weg}\}$ „$(u\{r\})$“

Behauptung: $\delta(R)$ ist zugehöriger optimaler Schnitt.
Es gilt: $r \in R$, $s \notin R$ (x optimal)
$\forall\, vw \in \delta(R)$ $x_{vw} = u_{vw}$ (sonst $r \rightarrow \ldots \rightarrow v \rightarrow w$, x erhöhend, aber $w \notin R$)
$\forall\, vw \in \delta(\bar{R})$ $x_{vw} = 0$ (analog). $\overset{3.3}{\Rightarrow} f_x(s) = \underbrace{x(\delta(R))}_{=u(\delta(R))} - \underbrace{x(\delta(\bar{R}))}_{=0} = u(\delta(R))$

Satz 3.6 *Ein zulässiger Fluss ist maximal genau dann, wenn kein x-erhöhender Weg existiert.*

Beweis:
„⇒“ klar
„⇐“ Beweis von Satz 3.5 liefert Schnitt $\delta(R)$ mit $f_x(s) = u(\delta(R)) \overset{3.4}{\Rightarrow} x$ ist maximaler zulässiger Fluss. q.e.d.

Satz 3.7 *Ist u ganzzahlig und existiert ein maximal zulässiger Fluss, so existiert ein ganzzahliger maximal zulässiger Fluss.*

Beweis: wähle ganzzahligen maximalen Fluss. x und u ganzzahlig. $\exists$ x-erhöhender Weg $\Rightarrow \exists$ höherer ganzzahliger Fluss.
Einfacher: Existiert ein maximaler zulässiger Fluss so existiert auch der minimal zulässige Schnitt. Der wiederum setzt sich als Summe der ganzzahligen Kapazitäten zusammen $\Rightarrow$. Diese Summe ist dann auch ganzzahlig.

Korollar 3.8 *Ist x ein zulässiger (r,s)-Fluss und $\delta(R)$ ein (r,s)-Schnitt, so ist $f_x(s)$ Maximum und $u(\delta(R))$ Minimum genau dann wenn: $x_e = u_e \ \forall e \in \delta(R)$*
$x_e = 0 \ \forall e \in \delta(\bar{R})$ q.e.d.

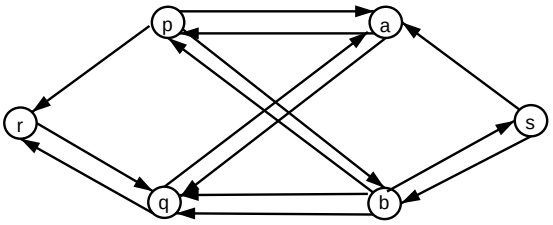
3.2 Erhöhender Weg (Augmenting Path) Algorithmus

Starte mit zulässigem Fluss x (z.B. $x = 0$)
while ($\exists x$-erhöhender Weg P) **do**
  $E_1 := \min\{u_e - x_e | e$ ist Vorwärtskante in $P\}$
  $E_2 := \min\{x_e | e$ ist Rückwärtskante in $P\}$
  $E := \min\{E_1, E_2\}$ „x-Breite von P“
  if $(E = \infty)$ **then**
    STOP „kein maximaler Fluss“
  else
    erhöhe Fluss auf P um E
  end if
end while
STOP x maximaler Fluss $R := \{v \in V | \exists x$-erhöhender (r,v)-Weg$\}$ min Schnitt $\delta(R)$.

Suche nach einem x-erhöhendem Weg

Hilfsdigraph $G(x)$: $V(G(x)) := V$
$vw \in E(G(x))$ genau dann wenn $vw \in E$ und $x_{vw} < u_{vw}$ oder $wv \in E$ und $x_{wv} > 0$.
(r,s)-Wege in $G(x) \Leftrightarrow x$-erhöhende Wege in G

Genannt Breitensuche: Zeit $O(m)$ pro Iteration. Im Beispiel (Hilfsdigraph zum ersten Fluss mit Flusswert 3):

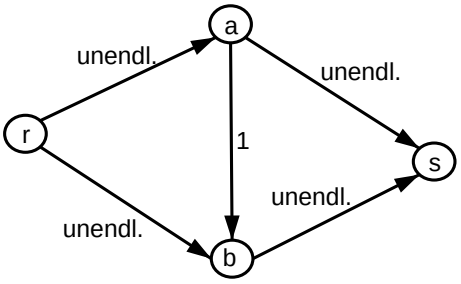


Satz 3.9 *Ist u ganzzahlig und der maximale Flusswert $k < \infty$, so terminiert der erhöhender Wege Algorithmus nach höchstens k Erhöhungen:*

Beweis: u ganzzahlig $\Rightarrow$ alle x ganzzahlig und jede Erhöhung erhöht den Fluss mindestens um 1. q.e.d.

D.h. für rationale u terminiert der E-W Algorithmus (Skalierung auf ganze Kapazitäten) - dies gilt also nicht für irrationale Kapazitäten.

Problem:



Der Algorithmus wählt immer abwechselnd *rabs* und *rbas* $\rightarrow$ keine Terminierung wenn P immer über *ab* geht. Wenn nicht ∞ sondern große Zahl M, so macht der Algorithmus M Iterationen und hat einen Fluss von $2M$.
$\Rightarrow$ Schranke in 3.9 ist scharf

Inputlänge: $n + m + logM$

Ausweg: Kürzeste x-erhöhende, d.h. x-erhöhende Wege mit minimaler Kantenzahl.

Satz 3.10 *Dinitz [1970], Edmonds und Kasp [1972]*
Wenn jede Erhöhung auf einem kürzesten, x-erhöhenden Weg statt findet, so erfolgen höchstens $n \cdot m$ Erhöhungen.

Beweis: weiter unten nach anderen Sätzen.

- Keine Annahmen über:
  - Existenz eines maximalen Flusses

  - Rationalität von u

- Breitensuche findet einen kürzesten x-erhöhenden Weg in Zeit $O(m)$

  Edmonds und Kasp:

  „... so simple that it is likely to be incorporated innocently into a computer implementation."

Korollar 3.11 *Der E-W-Algorithmus mit Breitensuche löst das Max Flow Problem in Zeit* $O(nm^2)$

Zum Beweis von Satz 3.10 zwei Lemmata:

Fluss $x \to x$-erhöhender Weg P $\underbrace{v_0}_{=r}, v_1, \ldots, \underbrace{v_k}_{=s}$ Fluss x'

$d_x(v,w)$ kürzeste Länge eines $v-w$-Weges in $G(x)$ ($=\infty$ falls keiner existiert).
d.h. $d_x(r,v_i) = i \quad d_x(v_i,s) = k-i \ \ \forall\, 0 \leqq i \leqq k$
$wv \in E(G(x'))$, $wv \notin E(G(x)) \Rightarrow (\exists i)\ w = v_i,\ v = v_{i-1}$

Kante ist jetzt in $G(x')$ da Fluss dort aufgebaut wurde.

Lemma 3.12 *Für jedes $v \in V$ gilt:*

$$d_{x'}(r,v) \geqq d_x(r,v) \text{ und } d_{x'}(v,s) \geqq d_x(v,s)$$

Beweis: Annahme: $(\exists v \in V)\ \ d_{x'}(r,v) < d_x(r,v)$
Wähle v mit $d_{x'}(r,v)$ Minimum, es gilt $d_{x'}(r,v) > 0 \ \ (v \neq 0)$

Zu $G(x')$: P': r $\underbrace{\to \to \ldots \to}_{\substack{\text{Vorwärts- oder}\\ \text{Rückwärtskanten}}}$ $w \to v$, w ist also der vorletzte Knoten.
Länge des Weges $= d_{x'}(r,v)$

Annahme:

$$\begin{aligned} d_x(r,v) &> d_{x'}(r,v) \\ &= d_{x'}(r,w)+1 \quad (*) \\ &\geqq d_x(r,w)+1 \end{aligned}$$

$\Rightarrow wv \in E(G(x')) \quad wv \notin E(G(x))$
$\Rightarrow (\exists i)\ w = v_i,\ v = v_{i-1}$
$\overset{(*)}{\Leftrightarrow} \quad \underbrace{i-1}_{\text{= Distanz von } v = v_{i-1}} > \underbrace{i}_{\text{=Distanz von } w = v_i} +1 \quad \lightning$
$d_{x'}(r,s) \geqq d_x(v,s)$ analog q.e.d.

D.h. $\leqq n-1$ Phasen, in jeder Phase: Augmentierungen auf Wegen einer festen Länge k.
Frage: Wie oft?

$$E(x) := \{e \in E | e \text{ ist Kante eines kürzesten } x\text{-erhöhenden Weges}\}$$

Lemma 3.13 *gilt $d_{x'}(r,s) = d_x(r,s)$, so gilt $E(x') \subset E(x)$*

Beweis: Sei $k = d_x(r,s) = d_{x'}(r,s)$, $e \in E(x')$
$\Rightarrow e = v_{i-1}v_i$ oder $e = v_i v_{i-1}$ in x'-erhöhendem Weg:

$$P' : \overbrace{v_0}^{=v}, \underbrace{v_i, \ldots v_{i-1}}_{d_{x'}(r,v)=i-1}, \underbrace{v_i, \ldots \overbrace{v_k}^{s}}_{d_{x'}(w,s)=k-i}$$

$\overset{3.12}{\Rightarrow} d_x(r,v) + d_x(w,s) = k-1$

Annahme: $e \notin E(x) \Rightarrow x_e \neq x'_e \Rightarrow e \in P \Rightarrow e \in E(x)$ $\lightning$
$\Rightarrow E(x') \subseteq E(x)$

Noch zu Zeigen: $E(x') \neq E(x)$: $\exists e \in P$: e vorwärts und $x'_e = u_e$ oder e rückwärts und $x'_e = 0$

Annahme: $e \in E(x')$
$\Rightarrow$ Jeder x'-erhöhende Weg benutzt e anders herum als in P
$(\exists i)$ Länge von $P' > -\underbrace{i}_{r \ldots v_i} + \underbrace{k-i+1}_{v_{i-1} \ldots s} + \underbrace{1}_{v_i \ldots v_{i-1}} = k+2$ $\lightning$
$\Rightarrow e \notin E(x')$

Anschaulich: e ist der Flaschenhals von P.

Beweis von Satz 3.10:
Lemma 3.12 $\Rightarrow$ höchstens $n-1$ Phasen
Lemma 3.13 $\Rightarrow$ höchstens m Erhöhungen pro Phase
$\Rightarrow mn$ Erhöhungen. q.e.d.

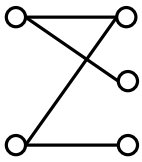
D.h. Gesamtlaufzeit für kürzeste erhöhende Wege Algorithmus $O(nm^2)$. Später noch ein Algorithmus mit Laufzeit $O(n^2 m)$.

3.3 Anwendungen von Max Flow Min Cut

3.3.1 Bipartite Matchings und Knotenüberdeckungen

Bipartite Graphen $G(V,E)$ ungerichtet
$V = P \cup Q \quad E = \{pq | p \in P, q \in Q\}$
$P \cap Q = \varnothing \quad \{P,Q\}$ Bipartition der Graphen.

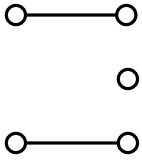
Bsp:



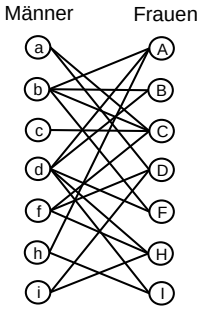
Bipartites Matching M

$M \subseteq E$ mit keine zwei kanten in M haben einen gemeinsamen Endknoten, b.z.w. $(\forall\, v \in V)|\{e \in M | v \in e\}| \leqq 1$

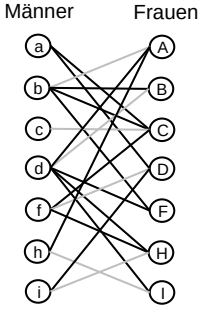
Im Beispiel z.B:



Heiratsproblem



Hierbei verdeutlichen die Kanten ein „sich mögen“ und es sollen möglichst viele Paare verheiratete werden aber nur solche die sich mögen. Eine Lösung ist:



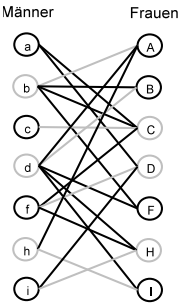
Hierbei können also sechs Paare verheiratet werden, wären auch mehr möglich?

$C \subseteq V$ heißt Knotenüberdeckung falls:

$$(\forall vw \in E)\ v \in C \text{ oder } w \in C$$

Offensichtlich handelt es sich hier um Schwache Dualität.
M Matching im bipartiten Graphen $G = (V,E)$
C Knotenüberdeckung im bipartiten Graphen $G = (V,E)$
$\Rightarrow |M| \leqq |C|$

nun die Knotenüberdeckung (grau gefärbte Knoten:)



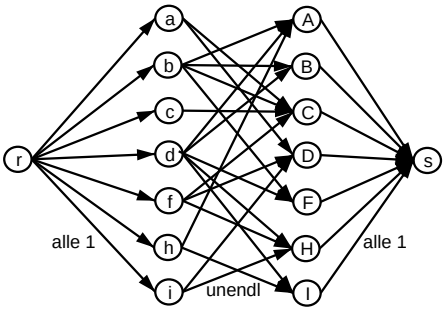
Diese hat nun auch 6 Knoten also ist die Lösung auch optimal:
$6 = \max |M| = \min |C|$

Satz 3.14 *Satz von König [1931]*
Für bipartite Graphen $G = (V,E)$ gilt:

$$\max\{|M|| \textit{ ist Matching } \} = \min\{|C|| \textit{ ist Knotenüberdeckung } \}$$

Beweis: Aus ungerichtetem Graphen $G = (V,E)$ konstruieren wir einen gerichteten Graphen $G'(V',E')$ wie folgt:
$V' := V \cup \{r,s\}$ $\quad E' : \forall pq \in E,\ p \in P, q \in Q$ gerichtete Kanten $pq \in E'$ mit der Kapazität $u_{pq} = \infty$, zusätzlich $\forall p \in P$ gerichtete Kante rp mit Kapazität $u_{rp} = 1$ und $\forall q \in Q$ gerichtete Kante qs mit Kapazität $u_{qs} = 1$

Im Beispiel:



Sei x ein ganzzahliger (d.h. 0/1) Fluss in G' mit Flusswert k.

Wir definieren $M \subseteq E$ durch:

$$\text{Für } pq \in E \text{ gilt } \begin{cases} pq \in M \text{ falls } x_{pq} = 1 \\ pq \notin M \text{ falls } x_{pq} = 0 \end{cases}$$

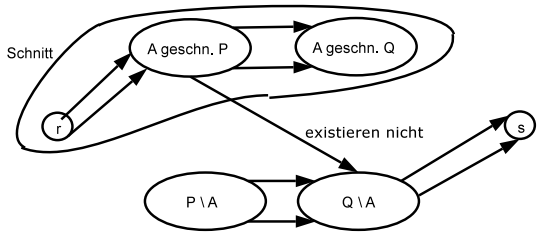
Dann ist M ein Matching in G mit Kardinalität k.

Umgekehrt: Sei M ein Matching in G mit Kardinalität k.
Wir definieren x_{vw} für $vw \in E'$ durch:

$$\begin{aligned} v \in P,\ w \in Q \quad &: x_{vw} = \begin{cases} 1 \text{ falls } vw \in M \\ 0 \text{ falls } vw \notin M \end{cases} \\ v = r,\ w \in P \quad &: x_{vw} = \begin{cases} 1\ (\exists e \in M)\ w \in e \\ 0 \text{ sonst} \end{cases} \\ v \in P,\ w = s \quad &: x_{vw} = \begin{cases} 1\ (\exists e \in M)\ v \in e \\ 0 \text{ sonst} \end{cases} \end{aligned}$$

Dann ist x ein ganzzahliger zulässiger Fluss in G' mit Flusswert $|M|$.

D.h. wir können das maximale Kardinalität Matching mit Max-Flow Berechnung bestimmen mit höchstens $\min\{|P|, |Q|\} \leqq n$ Flusserhöhungen. Satz 3.9 liefert uns die Laufzeit $O(mn)$ Sei $\delta'(\{r\} \cup A)$ mit $A \subseteq V$ ein minimaler (r,s)-Schnitt in G'



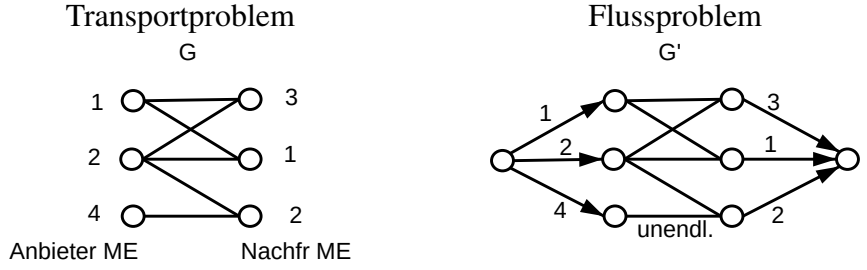
Endliche Kapazität des Schnitts $\Rightarrow (\nexists vw \in G)\ v \in A \cap P,\ w \in Q \setminus A$.
$\Rightarrow$ Jede Kante $e \in G$ ist inzident zu einem Knoten in $C := (P \setminus A) \cup (Q \cap A)$
$\Rightarrow C$ ist Knotenüberdeckung mit Kardinalität $|C| = |P \setminus A| + |Q \cap A| =$ Kapazität des Schnitts $= \max\{|M| | M \text{ Matching}\}$
D.h. der Algorithmus kann auch eine minimale Knotenüberdeckung berechnen.

3.3.2 Transportproblem

Gegeben: bipartiter Graph $G = (V,E)$, $V = P \cup Q$, $a \in \mathbb{Z}_+^P$, $b \in \mathbb{Z}_+^Q$
Gesucht: $x \in \mathbb{R}^e$ mit:

$$\begin{aligned} \sum_{q\in Q,\ pq\in E} x_{pq} &\leqq a_p \ \forall p \in P \\ \sum_{p\in P,\ pq\in E} x_{pq} &= b_q \\ x_{pq} &\geqq 0 \ \forall pq \in E \\ x_{pq} &\quad \text{ganzzahlig } \forall\ pq \in E \end{aligned}$$

Nun ist gefragt ob alle Nachfragen b mit den gegebenen Kapazitäten a überhaupt befriedigt werden können. Mathematisch ist also gefragt, ob eine zulässige Lösung für das Problem existiert. Auch dieses Problem ähnlich wie das Matching-Problem wird auf ein Max-Flow Problem übertragen: Beispiel:

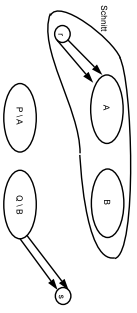


$\exists$ Lösung $x \in \mathbb{Z}^E$ $\quad\Leftrightarrow \exists$ ganzzahliger zul. Fluss mit Wert $\sum_{q\in Q} b_q$

Nach der Umformung kann man einfach den Max-Fluss Algorithmus anwenden:
Max-Flow-Min-Cut Theorem:
(TB) hat Lösung genau dann wenn jeder (r,s)-Schnitt in G' als maximale Kapazität wenigstens $\sum_{q\in Q} b_q$ hat.

Für $A \subseteq P$, $B \subseteq Q$ ist die Kapazität des Schnitts:

$$\delta'(A \cup B \cup \{r\}) = \sum_{i\in P\setminus A} a_i + \sum_{j\in B} b_j$$



$$\sum_{i\in P\setminus A} a_i + \sum_{j\in B} b_j \geqq \sum_{q\in Q} b_q \text{ g.d.w. } \sum_{i\in P\setminus A} a_i \geqq \sum_{j=Q\setminus B} b_j$$

Wir können die Überprüfung beschränken auf solche $A \subseteq P$ für die gilt: Jeder Knoten aus $P \setminus A$ ist mit einem Knoten aus $Q \setminus B$ adjazent.

Für $C \subseteq V$:

$$N(C) := \{w \in V | vw \in E \text{ für ein v } \in C\} = (\text{ungerichtete Nachbarmenge})$$

also können wir annehmen:

$$N(Q \setminus B) = P \setminus A$$

Fazit:

Satz 3.15 $\exists$ *Lösung für (TB) genau dann wenn* $(\forall\, C \subseteq Q)\quad a(N(C)) \geqq b(C)$

Interpretation: Für jede Kundenteilmenge muss der Bedarf durch die möglichen Produzenten gedeckt sein.

3.4 Minimale Schnitte und lineare Optimierung

Max-Fluss Problem als LP

$$\begin{array}{lll}
(LPMF) & \max \displaystyle\sum_{w\in V,\, ws\in E} x_{ws} - \sum_{w\in V,\, sw\in E} x_{sw} & \\
(y_v) & \displaystyle\sum_{w\in V,\, wv\in E} x_{wv} - \sum_{w\in V,\, vw\in E} x_{vw} & = \quad 0 \ \forall\, v \in V \setminus \{r,s\} \\
(z_{vw}) & x_{vw} & \leqq \quad u_{vw} \ \forall\, vw \in E \\
 & x_{vw} & \geqq \quad 0 \ \forall\, vw \in E
\end{array}$$

Duales LP

$$\begin{array}{lll}
(DLPMF) & \min \displaystyle\sum_{vw\in E} u_{vw} z_{vw} & \\
 & -y_v + y_w + z_{vw} & \geqq \quad 0 \ \forall\, vw \in E,\ v,w \in V \setminus \{r,s\} \\
 & y_w + z_{rw} & \geqq \quad 0 \ \forall\, rw \in E \\
 & -y_v + z_{vr} & \geqq \quad 0 \ \forall\, vr \in E \\
 & -y_v + z_{vs} & \geqq \quad 1 \ \forall\, vs \in E \\
 & y_w + z_{sw} & \geqq \quad -1 \ \forall\, sw \in E \\
 & z_{vw} & \geqq \quad 0 \ \forall\, vw \in E
\end{array}$$

Vereinfachung durch setzen der neuen Dualvariablen:

$$y_r = 0 \qquad y_s = -1$$

erhalten wir die gemeinsame Form:

$$-y_v + y_w + z_{vw} \geqq 0 \ \forall \ vw \in E$$

Die Addition von 1 zu allen y_v $(v \in V)$ ändert die Restriktion nicht, also ist (DLPMF) äquivalent zu:

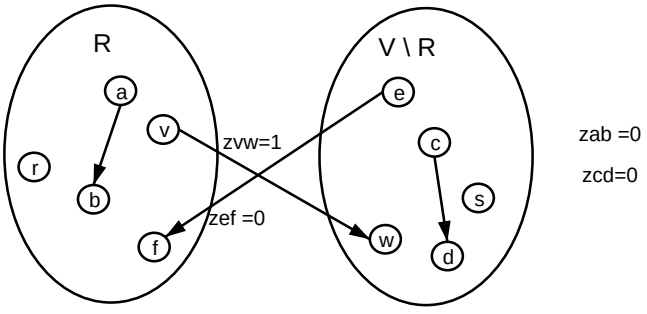
$$\begin{aligned} (DLPMF') \quad \min \sum_{vw \in E} u_{vw} z_{vw} & \\ y_r &= 1 \\ y_s &= 0 \\ -y_v + y_w + z_{vw} &\geqq 0 \ \forall \ vw \in E \\ z_{vw} &\geqq 0 \ \forall \ vw \in E \end{aligned}$$

Satz 3.16 *Falls (DLPMF') eine Optimallösung hat, so hat (DLPMF') eine Optimallösung der folgenden Form:*
Für einen (r,s)-Schnitt $\delta(R)$ ist y der charakteristische Vektor von R und z der charakteristische Vektor von $\delta(R)$

Beweis:
Wähle $\delta(R)$ MIT $r \in R$ als minimalen Schnitt und y, z als die entsprechenden charakteristischen Vektoren. Dann gilt:

$$y_r = 1, \ y_s = 0, \ z_{vw} \geqq 0 \ \forall vw \in E$$

und:



$$\begin{array}{lll} -y_a - y_b + z_{ab} & & -y_c + y_d - z_{cd} \\ -1+1+0 & = 0 & = 0+0+0 \quad = 0 \\ -y_v + y_w + z_w & & -y_e + y_f + z_{ef} \\ = -1+0+1 & = 0 & = 0+1+0 \quad > 0 \end{array}$$

Somit ist y_z zulässig für (DLPMF') und der ZF-Wert ist:

$$\begin{array}{lll} \sum_{vw \in E} u_{vw} z_{vw} & = & u(\delta(R)) \\ & \overset{\text{Max-Flow-Min-Cut}}{=} & \text{Wert des max. Flusses} \\ & = & \text{Optimaler Wert (LPMF)} \\ & \overset{\text{Dualitätsth.}}{=} & \text{Optimaler Wert von (DLPMF')} \end{array}$$

Formulierung des minimalen Schnitt Problems ohne y-Variable

$$\begin{array}{lll} (LPMC) & \min \sum_{e \in E} u_e z_e & \\ \text{s.t.} & \sum_{e \in P} z_e & \geqq 1 \ \forall \text{ einfache } (r,s)\text{-Wege } P \\ & z_e & \geqq 0 \ \forall\, e \in E \end{array}$$

Offensichtlich: Jeder charakteristische Vektor eines (r,s)-Schnittes $\delta(R)$ $z_e = \begin{cases} 1 \text{ falls } e \in \delta(R) \\ 0 \text{ sonst} \end{cases}$ ist zulässig für (LPMC).

Satz 3.17 *Falls (LPMC) eine Optimallösung hat, so hat (LPMC) eine Optimallösung, die charakteristischer Vektor eines (r,s)-Schnittes ist.*

Beweis: Wäre $u_e < 0$ für ein $e \in E$, so wäre (LPMC) unbeschränkt, also gilt: $u_e \geqq 0 \, \forall\, e \in E$
Sei z' charakteristischer Vektor eines minimalen (r,s)-Schnittes $\delta(R)$. Das duale LP lautet:

$$\begin{array}{lll} (DLPMC) & \max \sum_{P \text{ einf. } (r,s)\text{-Weg}} w_P & \\ \text{s.t.} & \sum_P w_P & \leqq u_e \ \forall\, e \in E \\ & w_P & \geqq 0 \ \forall \text{ einf. } (r,s)\text{-Wege } P \end{array}$$

Sei x ein maximaler Fluss mit Wert F.
Wiederhole:
Finde einfachen (r,s)-Weg P der Form:

$$r \overset{x_{e_1}>0}{\longrightarrow} \bullet \overset{x_{e_2}>0}{\longrightarrow} \bullet \longrightarrow \ldots \overset{x_{e_{k-1}}>0}{\longrightarrow} \bullet \overset{x_{e_k}>0}{\longrightarrow} s$$

Setze $w_p := \min\{x_{e_i} | 1 \leqq i \leqq k\}$
Setze für $1 \leqq i \leqq k$: $x_{e_i} := x_{e_i} - w_p$ bis $f_x = 0$. Nun gilt $\sum_P w_p = F$
Die Prozedur terminiert, da immer wenigstens ein x_e Null wird.
w ist zulässig für (DLPMC) und

$$\sum_P w_p = \sum u_e z'_e$$

$\Rightarrow z'$ ist optimal für (LPMC)

3.5 Der Algorithmus von Goldberg und Tarjan [1988]

Vereinbarung: Falls $vw \in E$ und $wv \notin E$ setzen wir $u_{wv} = x_{wv} = 0$ (nur für Beschreibung des Algorithmus, die Implementierung fügt Kante nicht hinzu).

Für $x \in R^e$ (nicht notwendigerweise Fluss) mit $0 \leqq x_e \leqq u_e \; \forall \; e \in E$ definieren wir den Hilfsdigraphen $G(x)$.
$vw \in E(G(x)) \Leftrightarrow x_{vw} < u_{vw}$ oder $x_{wv} > 0$. Parallele Kanten in $G(x)$ sind nicht erlaubt.

Restkapazität von vw: $\bar{u}_{vw} := u_{vw} - x_{vw} + x_{wv}$
($\bar{u}_{vw}$ kann auf vw zusätzlich fließen ohne $0 \leqq x_{vw} \leqq u_{vw}$ zu verletzen. Eine solche Lösung verletzt im Allgemeinen aber die Flusserhaltungsbedingung)

x heißt Präfluss (Preflow) falls:

$$f_x(v) \geqq 0 \;\; \forall \, v \in V \setminus \{r,s\}$$

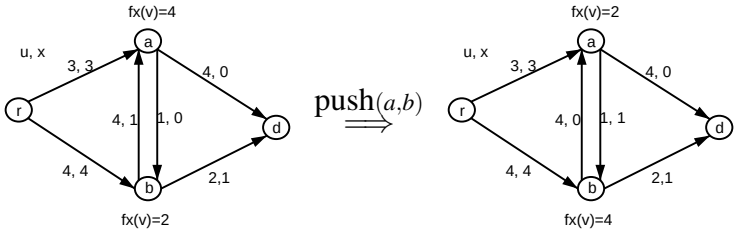
Push Operation für $vw \in E$ mit $\bar{u}_{vw} > 0$ und $f_x(v) > 0$

$$x_{vw} \leftarrow x_{vw} + \underbrace{\min(\bar{u}_{vw},\; f_x(v))}_{:=\varepsilon}$$

Erhält die „zulässiger Präfluss“ Eigenschaft. (Falls auch $wv \in E$ und $\varepsilon < \bar{u}_{vw}$)

$$\begin{aligned} x_{wv} &\leftarrow x_{wv} - \underbrace{\min(\varepsilon,\; x_{wv})}_{:=\varepsilon'} \\ x_{vw} &\leftarrow x_{vw} + \varepsilon - \varepsilon' \end{aligned}$$

Beispiel:



Zu erkennen ist dass hier auch die „zulässiger Präfluss“ Eigenschaft erhalten bleibt.
Knoten $v \in V \setminus \{r,s\}$ ist *aktiv* falls $f_x(v) > 0$
$\Rightarrow$ Ein Präfluss ist ein Fluss, falls es keine aktiven Knoten gibt.

Grundoperationen:

1. Wähle einen aktiven Knoten v
2. Wähle $vw \in G(x)$
3. push vw

Wesentlich ist die richtige Reihenfolge, sonst könnte der Algorithmus unendlich operieren: push(vw), push(wv), push(vw), push$(wv), \ldots$

$d \in (\mathbb{Z}^+ \cup \{\infty\})^V$ ist eine gültige Nummerierung (valid labeling) bezüglich eines Präflusses x falls:

$$\begin{aligned} d(r) &= n \\ d(s) &= 0 \\ d(v) &\leqq d(w)+1 \ \ \forall\, vw \in G(x) \end{aligned}$$

Invariante des GT-Algorithmus: Zulässiger Präfluss mit gültiger Nummerierung.

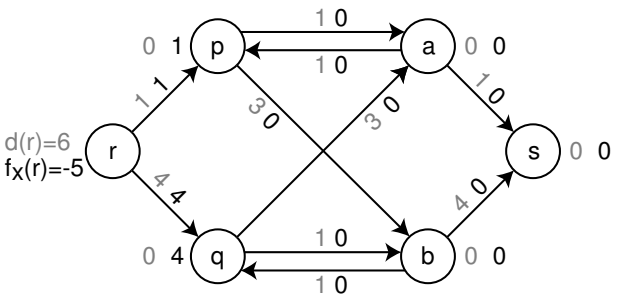
Initialisierung:
Initialisiere (x,d);

$$\forall\, ij \in E \text{ setze } x_{ij} = \begin{cases} u_{ij}, \text{ falls } i = r \\ 0 \text{ sonst} \end{cases}$$

$$\forall\, i \in V \text{ setze } d_i = \begin{cases} n, \text{ falls } i = r \\ 0 \text{ sonst} \end{cases}$$

x, d erfüllen gültige Nummerierung (wir müssen annehmen, dass alle $u_{rv} < \infty$ falls nicht, so kann man diese Bedingung herstellen. In der Übungsaufgabe...)

Beispiel:



Lemma 3.18 *Ist x ein zulässiger Präfluss und d eine gültige Nummerierung bezüglich x, so existiert ein (r,s)-Schnitt $\delta(R)$, so dass*

$$\begin{aligned} x_{vw} &= u_{vw} \ \forall\, vw \in \delta(R) \\ x_{vw} &= 0 \ \forall\, vw \in \delta(\bar{R}) \end{aligned}$$

Beweis: In einer gültigen Nummerierung $\exists k,\ 0 < k < n,\ d(v) \neq k \forall\ v \in V$ ($n-1$ Zahlen auf $n-2$ Knoten, einer bleibt über)
Setze $R := \{v \in V | d(v) > k\}$
dann ist $r \in R,\ s \notin R$ und wegen gültiger Nummerierung kann keine Kante in $G(x)$ R verlassen $\Rightarrow$ Behauptung.

Korollar 3.19 *Hat ein zulässiger Fluss eine gültige Nummerierung, so ist x_e ein maximum Fluss.*

Korollar 3.19 liefert Terminierungsbedingung. Invariante:

1. zulässiger Präfluss
2. gültige Nummerierung (d.h. saturierter Schnitt)

Terminierung sobald Fluss.

vgl. Erhöhender Wege-Algorithmus: Invariante:

1. zulässiger Fluss
2. Terminierung, sobald ein Schnitt saturiert ist.

Die gültige Nummerierung approximiert Distanzen in $G(x)$

Lemma 3.20 *Für jeden zulässigen Präfluss x und jede gültige Nummerierung D bezüglich x gilt:*

$$d_x(v,w) \geqq d(v) - d(w) \ \ \forall v,w \in V$$

Beweis: für $d_x(v,w) = \infty$ trivial.
für $d_x(v,w) < \infty$, P ein kürzester gerichteter (v,w)-Weg in $G(x)$.
Wir addieren die Ungleichungen $d(p) - d(q) \leqq 1$ für alle $pq \in P$.
Summe: $d(v) - d(w)$ q.e.d.

Insbesondere gilt:

$$\begin{aligned} d(v) &\leqq d_x(v,s) \\ d(v) - n &\leqq d_x(v,r) \end{aligned}$$

Also $d(v) \geqq n \Rightarrow d_x(v,s) = \infty$
D.h. in diesem Fall sollte der Fluss Richtung Quelle „gepusht“ werden.

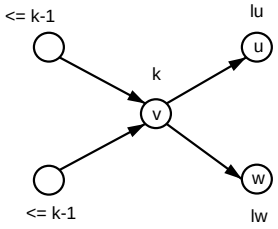
Strategie : $push(vw)$ mit $d(w) < d(v)$

$$\left.\begin{array}{l} \text{gültige Nummerierung} \\ vw \in G(x) \\ d(w) < d(v) \end{array}\right\} \rightarrow d(w) = d(v) - 1$$

vw eine *zugelassene Kante* (admissible arc) falls $vw \in E(G(x)), d(v) = d(w) + 1$
Ein $push(vw)$ auf vw erhält die gültige Nummerierung. wv könnte in $E(G(x'))$ sein, aber $d(w) \leqq d(v) + 1$ gilt bereits vorher.

Was ist aber nun wenn v aktiv ist aber $(\nexists vw \in G(x))\, d(v) = d(w) + 1$? Antwort: $relabel(v)$: $d(v) \leftarrow min\{d(w) + 1 | vw \in E(G(x))\}$.

Auch danach ist die Nummerierung in $G(x)$ noch gültig:



$l_u \geqq k - 1$
hier: $l_u \geqq k$
$l_w \geqq k - 1$
hier: $l_w \geqq k$

Wenn l_w Minimum ist, dann erhält v also die Nummer $l_w + 1$

$process(v)$

while $\exists$ zugelassene Kante vw **do**
  $push(vw)$;
  if v aktiv **then**
    $relabel(v)$

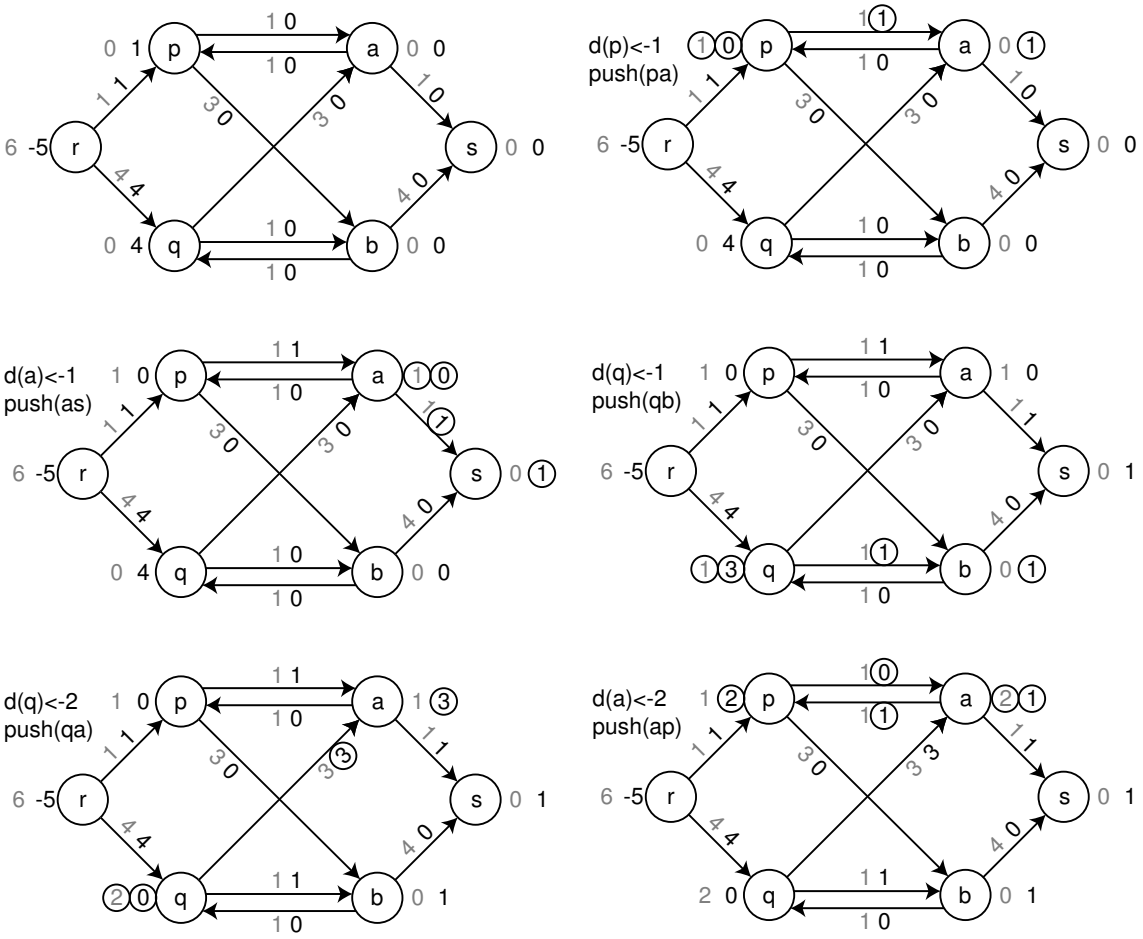
end if
end while

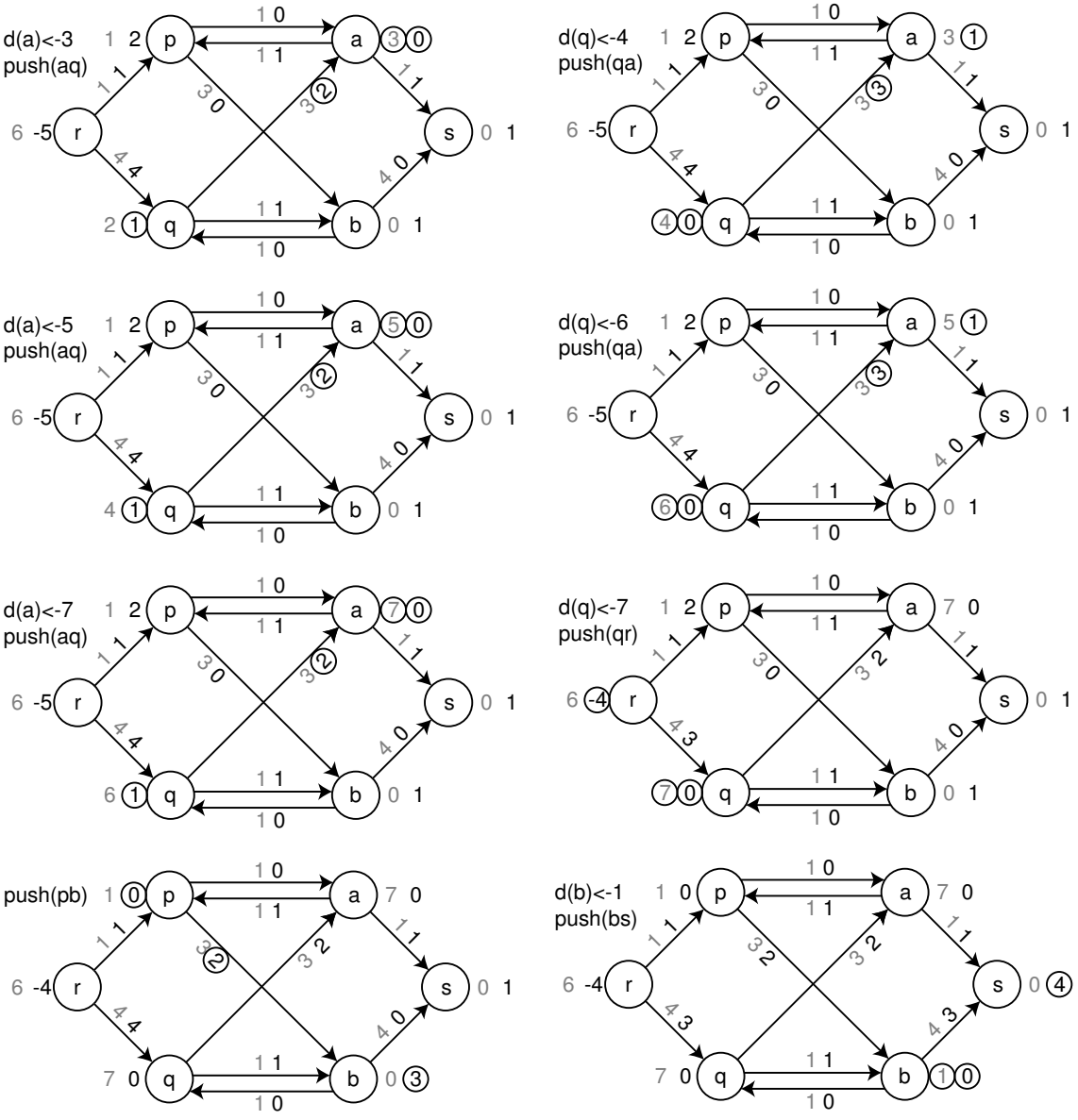
Der Goldberg-Tarjan Algorithmus

initialize (x,a);
while x is not a flow **do**
choose active node v;
$process(v)$
end while

Beispiel: Wir wählen immer v mit größter Distanz $d(v)$ „maximum distance“-Version des GT-Algorithmus. Vielen Dank an Daniel Teske für die Zeichnungen...

An den Knoten sind die grauen Zahlen die Distanzlabel, die dunklen Nettofluss/Überschuss, an den Kanten sind die Kapazitäten hell und der Fluss dunkel.





Satz 3.21 *Der (max-distance) GT-Algorithmus ist korrekt. Der GT-Algorithmus führt $O(n^2)$ relabel Operationen durch, sowie $O(n^2m)$ push-Operationen in der Urversion und $O(n^2\sqrt{m})$ in der max-dist-Version.*

Die Urversion kann so implementiert werden, dass sie $O(n^2m)$ Zeit benötigt, die max-dist-Version so, dass sie $O(n^2\sqrt{m})$ Zeit benötigt.

Beweis: Korrektheit folgt aus Korollar 3.19: Falls GT-Algorithmus terminiert, so mit einem maximalen Fluss.

Die Laufzeit ist sehr aufwendig zu beweisen, deswegen verzichten wir an dieser Stelle darauf.

Aus der Praxis kann man sagen das der GT-Algorithmus dort gute Ergebnisse erzielt.

3.6 Minimale Schnitte in ungerichteten Graphen („Min-Cut“)

Gegeben: ungerichteter Graph $G = (V, E)$ mit Kantenkapazitäten $z_e \geqq 0 \ \forall\, e \in E$.

Gesucht: $\varnothing \neq S \subset V$ mit $z(\delta(S))$ Minimum.
Dieses Problem tritt z.B. als Teilproblem des TSP auf.

Spezialfall: $z_e = 1 \forall\, e \in E$: Bestimmung des Knotenzusammenhangs. (minimale Anzahl von Kanten die entfernt werden müssen um G unzusammenhängend zu machen).

Sei G zusammenhängend (sonst ist es trivial), z.B. Breitensuche findet in Linearzeit ein S mit $z(\delta(S)) = 0$. Das Problem kann offensichtlich via $\binom{n}{2}$ (r,s)-Max Flow-Min-Cut Berechnungen gelöst werden. $\rightarrow$ Laufzeit $O(n^5)$

Schrumpfen eines Knotenpaares $i, j \in V$

1. $V \leftarrow V \setminus \{i, j\} \cup \{p\}$ $\quad p$ ist also neuer Knoten.
2. Ersetze in p jedes Paar von Kanten

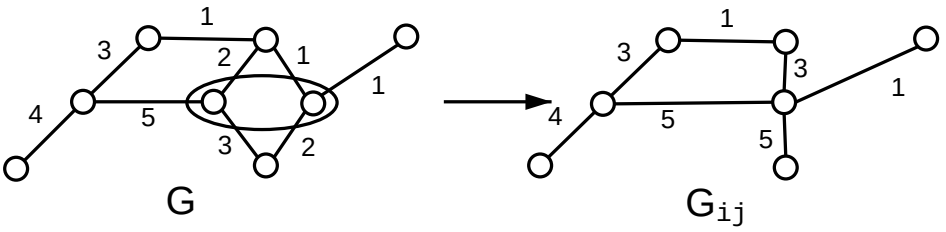
$$e' = ik \quad e'' = jk$$

   durch:

$$e^* = pk \text{ mit } z^*_e = z_{e'} + z_{e''}$$

3. Ersetze in E alle Kanten ie durch pe mit $z_{pe} = z_{ie}$
4. Ersetze in E alle Kanten je durch pe mit $z_{pe} = z_{je}$

Beispiel:



$\lambda(G)$: Kapazität eines minimum Schnittes in G
$\lambda(G; i, j)$: Kapazität eines minimum i-j Schnittes in G

Lemma 3.22 $\lambda(G) = \min(\lambda(G_{ij}),\ \lambda(G; i, j))$

Beweis: trivial

Das Verfahren

$z^* := \infty$;
wiederhole $n-1$ mal:
wähle $r, s \in V$;
Bestimme (r,s)-min-cut $\delta(S)$;
Falls $z(\delta(s)) < z^*$
$S^* := S$;
$z^* := z(\delta(S))$;
Schrumpfe r und s

liefert (mit G-T Algorithmus) eine Lösung in $O(n^4)$ Zeit.

Ein einfacher Min-Cut Algorithmus

Grundidee: Nagamochi und Iberaki [1992]
Effiziente Implementierung von Nagamochi, Ono und Iberaki [1993]
einfache Korrektheitsbeweise: Frank [1994], Stoer u. Wagner [1994]
Beschreibung von Stoer und Wagner:
Notation $z(A:B)$: $\sum_{\substack{a \in A \\ b \in B}} z_{ab}$ für $A, B \subseteq V$
$z(A:b)$ statt $z(A,\{b\})$

Erinnerung: Prims Algorithmus für einen minimum spannenden Baum: Min Spanning Tree (G,z,a) [$a \in V$ beliebig].

$W := \{a\}$;
$T := \varnothing$;
while $W \neq V$ **do**
wähle $v \notin W$ mit $z_{wv} = \min\{z_{wv} | w \in W, wv \in E\}$
$W := W \cup \{v\}$; (Füge einen am schwächsten mit W verbundenen Knoten hinzu)
$T := T \cup \{uv\}$;
end while

ähnlich:
Min_Cut_Phase
$W := \{a\}$;
while $W \neq V$ **do**
wähle $v \notin W$ mit $z(W:v) = \max\{z(W:v) | v \notin W\}$;

$W := W \cup \{v\}$; (Füge einen am stärksten mit W verb. Knoten hinzu)
end while
Sei s der vorletzte, t der vorletzte zu W hinzugefügte Knoten.
$z^* := z(\delta(t))$;
$\delta^* :=$ Menge der von t repräsentierten Orginalknoten (Phasenschnitt);
Schrumpfe s und t

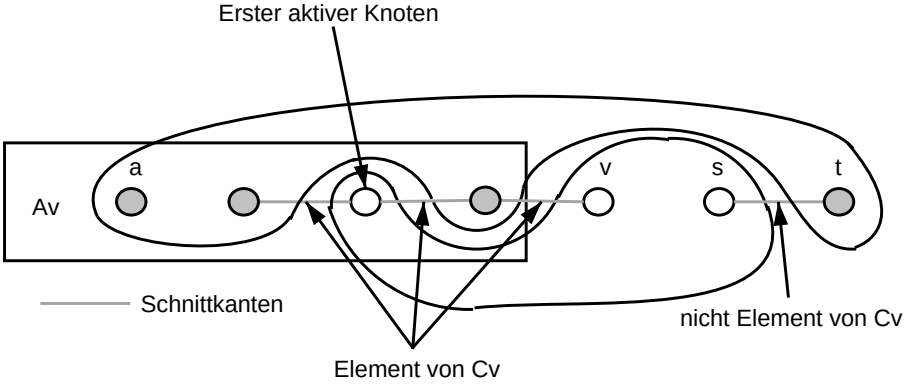
Berechnung des Min-Cut(g,z,a)**;**

$z_{\min} := \infty$;
while $|V| > 1$ **do**
Min_Cut_Phase(G,z,a);
if $z^* < z_{\min}$ **then**
$z_{\min} := z^*$;
$\delta_{\min} := \delta^*$;
end if
end while

Lemma 3.23 *Jeder Phasenschnitt ist ein minimum (s,t)-Schnitt im jeweiligen „Phasengraphen“, wobei s vorletzter und t letzter zu W hinzugefügter Knoten ist.*

Beweis:
Situation: $\overset{\text{Hinzufügungsreihenfolge}}{\Longrightarrow}$
$a \to \bullet \to \ldots \to s \to t$
Sei $C \subseteq E$ ein beliebiger(s,t)-Schnitt in G
Wir zeigen $z(C) \geqq z^* = z(\delta(t))$
$v \neq a$ heißt *aktiv* bezüglich C, falls $wv \in C$ mit $w =$ direkter Vorgänger von v.
$A_v \subseteq V$: Menge aller Vorgänger von v (ohne v)
$C_v := \{xy \in C | x,y \in A_v \cup \{v\}\} \subseteq C$

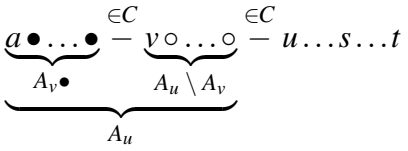
Beispiel:



Behauptung: Für jeden aktiven Knoten gilt $z(A_v : v) \leqq z(C_v)$. Beweis: Induktion über aktive Knoten:

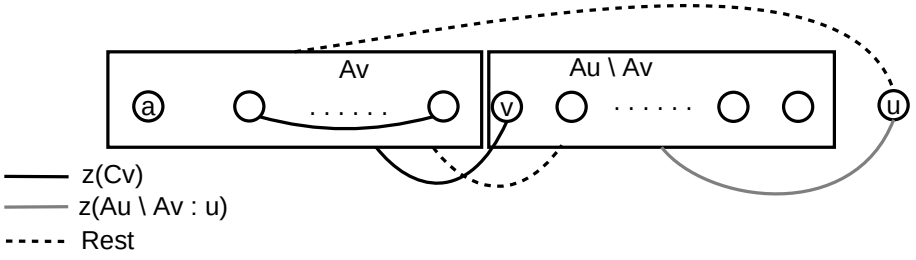
Für den ersten aktiven Knoten v_1 gilt $z(A_{v_1} : v_1) = z(C_{v_1})$. Die Behauptung gelte für alle aktiven Knoten bis zu v. Sei n der nächste aktive Knoten.

Situation:



Es gilt:

$$\begin{aligned} z(A_u : u) &= z(A_v : u) + z(A_u \setminus A_v : u) \\ &\leqq z(A_v : v) \quad \text{(Da } v \text{ stärkster mit } A_v \text{ verb. Knoten)} \\ &\leqq z(C_v) \\ &\leqq z(C_v) + z(A_u \setminus A_v : u) \end{aligned}$$



alles zusammen: $z(C_u) \leqq z(C_u)$ q.e.d (Beh.)
$st \in C \Rightarrow t$ aktiv $\Rightarrow z^* = z(\delta(t)) = z(A_t : t) \leqq z(C_t) = z(C)$ q.e.d. (Lemma)

Laufzeit

MINIMUM_CUT_PHASE
Prioritätsschlange für Knoten in $V \setminus \{a\}$
Schlüssel für $v \in V \setminus \{a\}$
key$(v) = z(W : v)$: Gesamtkapazität der Kanten zwischen v und derzeitigem W. Neu hinzuzufügender Knoten v via EXTRACT_MAX
Update: key$(u) :=$ *key*$(u) + z_{vu}$ für $u \notin W$ falls $vu \in E \rightarrow$ INCREASE_KEY_OPERATION für jede Kante $e \in E$ genau einmal

insgesamt:

$$\left.\begin{array}{rcll} n & = & |V| & \text{EXTRACT\_MAX} \\ m & = & |E| & \text{INCREASE\_KEY} \end{array}\right\} \text{Operationen}$$

mit Binär-Heaps (Informatik I) je: $\left.\begin{array}{l} O(\log n) \\ O(\log n) \end{array}\right\}$ Zeit

mit Fibonacci-Heaps je $\left.\begin{array}{l} O(\log n) \\ O(1) \end{array}\right\}$ Zeit

d.h. $O(m + n\log n)$ Zeit für MIN_CUT_PHASE
$O(n)$-Phasen $\rightarrow O(mn + n^2 \log n)$ Gesamtzeit.

Satz 3.24 *Der Phasenschnitt-Algorithmus findet einen minimum Schnitt in einem ungerichteten Graphen in $O(nm + n^2 \log n)$ Zeit.*

In dieser Vorlesung teilte Prof. Jünger noch ein Blatt mit Beispielen zum Phasenschnittalgorithmus sowie einen Artikel (Jünger M.; Rinaldi G.; Thienel S. (2000). *Practical performance of efficient minimum cut algorithms*. Algorithmica v26n1 S. 172-195).

Kapitel 4

Minimale Kostenflüsse

4.1 Optimalitätsbedingungen, Reduktionen

$$\text{(LPMKFP)} \qquad \min \sum_{e \in E} c_e x_e$$
$$\begin{aligned} f_x(v) &= b_v \quad \forall\, v \in V \\ 0 \leqq x_e &\leqq u_e \quad \forall e \in E \end{aligned}$$

Man kann es auch ganzzahlig betrachten, dann gilt $x_e \in \mathbb{Z}$.

$u \in (\mathbb{R} \cup \{\infty\})^E \quad b(V) = 0 \quad b_v > 0$ Nachfrage an Knoten v
$b_v < 0$ Vorrat

Spezialfälle:

- $c = 0$: Zulässigkeitsproblem (Lösbar als Max-Fluss Problem)
- $r, s \in V,\ b_r = -1,\ b_s = 1,\ b_v = 0\ \forall\, v \in V \setminus \{r,s\},\ u_e = \infty\ \forall\, e \in E \rightarrow$ dann ist es das LPKW
- Transportproblem:
  - G bipartit $V = P \cup Q$ (Bipartition)
  - a_p $(p \in P)$ Vorrat
  - b_q $(q \in Q)$ Nachfrage
  - c_{pq} $(pq \in E)$ Transportkosten

$$\min \sum_{pq \in E} c_{pq} x_{pq}$$
$$\begin{aligned} \sum_{pq \in E} x_{pq} &= b_q \quad \forall\, q \in Q \\ \sum_{pq \in E} x_{pq} &= a_p \quad \forall p \in P \ (\cdot -1 \text{ macht daraus ein LPMKFP}) \\ x_{pq} &\geqq 0 \quad \forall pq \in E \end{aligned}$$

wenn zusätzlich $x_{pq} \leqq u_{pq}$ gilt handelt es sich um ein kapazitiertes Transportproblem.

- $u_e = \infty \; \forall e \in E$: Transshipment Problem

Dualisieren:

$$\begin{aligned} \text{(LPMKFP)} \qquad \min \sum_{e \in E} c_e x_e & \\ f_x(v) &= b_v \;\; \forall\, v \in V \qquad (y) \\ -x_{vw} &\geqq -u_{vw} \;\; \forall\, vw \in E \qquad (z) \\ x_{vw} &\geqq 0 \;\; \forall\, vw \in E \end{aligned}$$

$$\begin{aligned} \text{(DLPMKFP)} \qquad \max \sum_{v \in V} b_v y_v - \sum_{vw \in E} u_{vw} z_{vw} & \\ -y_v + y_w - z_{vw} &\leqq c_{vw} \;\; \forall\, vw \in E \\ z_{vw} &\geqq 0 \;\; \forall\, vw \in E \end{aligned}$$

falls $u_{vw} = \infty \rightarrow$ keine Variable z_{vw}

$\bar{c}_{vw} := c_{vw} + y_v - y_w$ „reduzierte Kosten von vw“
$u_e = \infty \Rightarrow \bar{c}_e \geqq 0$
$u_e \neq \infty \Rightarrow z_e \geqq -\bar{c}_e,\, z_e \geqq 0$
Setze $z_e = \max\{0, -\bar{c}_e\}$ d.h. z_e „unnötig“

Beschränkung auf y:
Komplementäre Schlupfbedingungen

$$\begin{aligned} x_e > 0 \quad &\Rightarrow \quad -\bar{c}_e = z_e = \max\{0, -\bar{c}_e\} \\ &\Leftrightarrow \quad \bar{c}_e \leqq 0 \end{aligned}$$

$z_e > 0 \Rightarrow -\bar{c}_e > 0 \Rightarrow \bar{c} < 0 \Rightarrow x_e = u_e$
$\Leftrightarrow -\bar{c}_e > 0 \Leftrightarrow c_e < 0$

Satz 4.1 *Eine zulässige Lösung x von (LPMKFP) ist optimal genau dann wenn ein $y \in \mathbb{R}^V$ existiert, so dass für alle $e \in E$ gilt:*

$$\begin{aligned} \bar{c}_e < 0 \quad &\Rightarrow \quad x_e = u_e \quad (\neq \infty) \\ \bar{c}_e > 0 \quad &\Rightarrow \quad x_e = 0 \end{aligned}$$

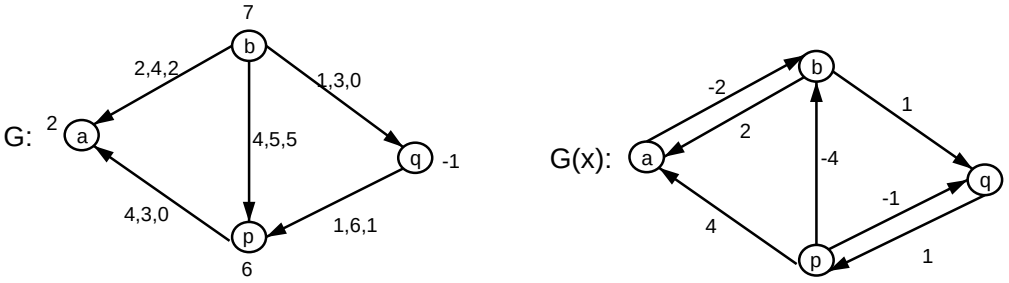
Kosten eines Weges $P \bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \ldots \rightarrow \bullet \rightarrow \bullet$:

$$C_p = \sum_{e} \in P,\, e \text{ vorwärts } c_e - \sum_{e} \in P,\, e \text{ rückwärts } c_e$$

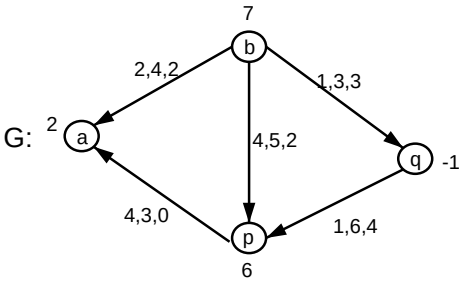
Ein x-erhöhender Weg P mit Erhöhung um ε, entspricht einer Kostenänderung

$$c^T x \rightarrow c^T x + \varepsilon C_P$$

Verbesserung eines zulässigen x-erhöhenden Kreises mit negativen Kosten mit einem Hilfsdigraph G(x):
$\forall vw \in E$ mit $x_{vw} < u_{vw}$ Kosten $c'_{vw} = c_{vw}$
$\forall vw \in E$ mit $x_{vw} > 0$ Kosten $c'_{wv} = -c_{vw}$
Beispiel (c,u,x):

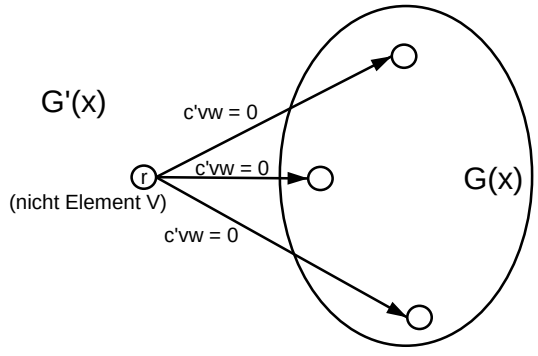


z.B. negativen Kreis pbq um 3 erhöhen (Kostenminderung um 6):



Satz 4.2 *Eine Zulässige Lösung x von (LPMKFP) ist optimal g.d.w. kein x-erhöhender Kreis mit neg. kosten existiert.*

Beweis:



Löse das kürzeste Wege Problem in G'.
Resultat: Entweder negativer Kreis oder zulässiges Potential:

1. Möglichkeit: Negativer Kreis: enthält r nicht, liefert also x-erhöhenden Kreis mit negativen Kosten.

2. Möglichkeit: zul. Potential:

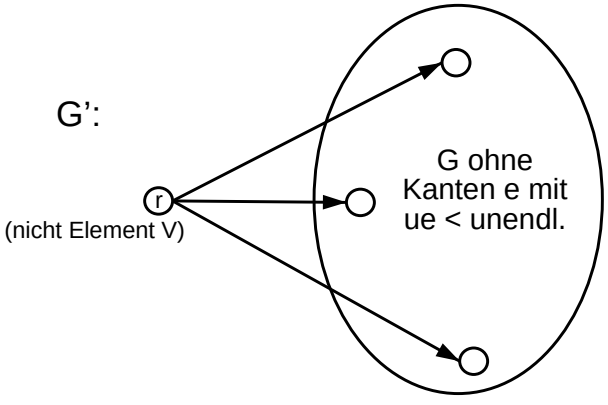
$$
\begin{aligned}
& y_v + c'_{vw} \geqq y_w \;\forall\, vw \in E(G(x)) \\
\Leftrightarrow\ & \left\{ \begin{array}{l} y_v + c_{vw} \geqq y_w \text{ falls } x_{vw} < u_{vw} \\ y_v - c_{wv} \geqq y_w \text{ falls } x_{wv} > 0 \end{array} \right\} \\
\Leftrightarrow\ & \left\{ \begin{array}{l} x_e < u_e \Rightarrow \bar{c}_e \geqq 0 \\ x_e > 0 \Rightarrow \bar{c}_e \leqq 0 \end{array} \right\} \text{„Konstruktion eines optimalen } y \text{ aus einem optimalen } x\text{“} \\
\Leftrightarrow\ & \left\{ \begin{array}{l} \bar{c}_e < 0 \Rightarrow x_e = u_e \\ \bar{c}_e > 0 \Rightarrow x_e = 0 \end{array} \right\} \text{Optimalitätsbed. aus Satz 4.1 q.e.d.}
\end{aligned}
$$

Satz 4.3 *Hat (LPMKFP) eine Optimallösung und ist ganzzahlig, so kann y in Satz 4.1 ganzzahlig gewählt werden, d.h. (DLPMKFP) hat eine ganzzahlige Optimallösung.*

Beweis: Kantengewichte ganzzahlig $\Rightarrow$ Kürzeste Wege Kosten ganzzahlig q.e.d.

Satz 4.4 *hat (LPMKFP) eine zulässige Lösung so existiert eine Optimallösung genau dann, wenn es keinen negativ gerichteten Kreis gibt, dessen Kapazitäten alle unbeschränkt sind.*

Beweis:
„$\Rightarrow$“ $\exists$ negativer Kreis… $\Rightarrow$ (LPMKFP) unbeschränkt. trivial.
„$\Leftarrow$'"$\nexists$ negativer gerichteter Kreis ...
Es reicht zu zeigen (DLPMKFP) hat zulässige Lösung (y,z)
Für $u_e \neq \infty$ wähle $z_e = \max\{0, -c_e\}$
Also brauchen wir y mit $\bar{c}_e \geqq 0 \;\; \forall e$ mit $u_e = \infty$



wähle y als zulässiges Potential in G'. q.e.d.

Konstruktion eines optimalen x aus optimalem y

Bedingungen für x:

$$(*)\begin{cases} f_x(v) = b_v \ \forall\, v \in V \\ 0 \leqq x \leqq u \\ \bar{c}_e > 0 \Rightarrow x_e = 0 \\ \bar{c}_e < 0 \Rightarrow x_e = u_e \end{cases}$$

Definiere:

$$u'_e = \begin{cases} 0 \text{ falls } \bar{c}_e > 0 \\ u_e \text{ sonst} \end{cases}$$

$$l'_e = \begin{cases} u_e \text{ falls } \bar{c}_e < 0 \\ 0 \text{ sonst} \end{cases}$$

$$(*) \Leftrightarrow (**) \quad \begin{cases} f_x(v) = b_v \\ l' \leqq x \leqq u' \end{cases}$$

Löse $(**)$ mittels Maximum-Fluss-Algorithmus.

Satz 4.5 *Sind b und u ganzzahlig und hat (LPMKFP) eine Optimallösung, so existiert eine ganzzahlige Optimallösung. q.e.d.*

Reduktionen:

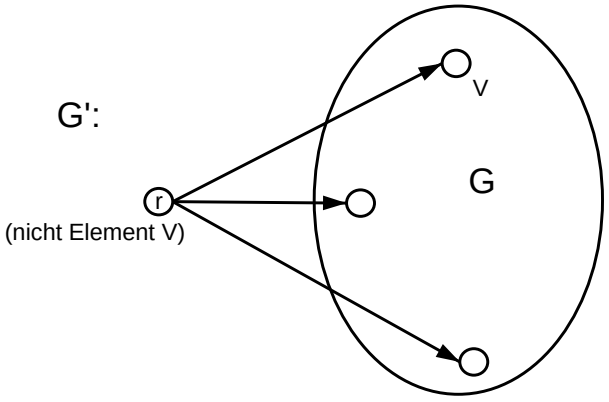
A

$$\begin{array}{rcll} & \min c^T x & & \\ a_v \leqq f_v(x) & \leqq & b_v & \forall\, v \in V \\ l_e \leqq x_e & \leqq & u_e & \forall e \in E \end{array}$$

auf (MKFP)

Wir dürfen annehmen, dass

A1 $(\forall\, v \in V)\ a_v = b_v$



$$\begin{array}{rcl} c_{rv} & = & 0 \\ l_{rv} & = & 0 \\ u_{rv} & = & b_v - a_v \\ b_r & = & -\sum_{v \in V} b_v \end{array}$$

Ersetze alle a_v durch b_v

im neuen Problem gilt:

$$x_{rv} + f_x(v) = b_r \ \forall r \in V$$
$$\Rightarrow x_{rv} = b_v - f_x(v)$$

D.h.:

$$\begin{aligned} & a_v \leqq f_x(v) \leqq b_v \Leftrightarrow 0 \leqq x_{rv} \leqq b_v - a_v \\ & \Leftrightarrow a_v \leqq b_v - \underbrace{x_{rv}}_{\geqq 0} \quad \underbrace{\leqq}_{\text{redundant}} \quad l_v \\ & \Leftrightarrow x_{rv} \leqq b_v - a_v \end{aligned}$$

Zulässige Lösungen des neuen Problems eingeschränkt auf die alten Kanten sind genau die zulässigen Lösungen des alten Problems:

- Gleicher ZF-Wert
- $O(m)$ Kanten $O(n)$ Knoten, d.h. keinen Effizienzverlust

A2 $(\forall\, e \in E)\; l_e \neq -\infty$ oder $u_e \neq \infty$

Dies wird eine Übungsaufgabe sein!

A3 $(\forall\, e \in E)\; l_e \neq \infty$

Nach A2 gilt: $l_e = -\infty \Rightarrow u_e \neq \infty$

Ersetze $v \xrightarrow{e} w$ durch $v \xleftarrow{e'} w$

$$\begin{aligned} c_{e'} &= -c_e \\ l_{e'} &= -u_e \\ u_{e'} &= \infty \end{aligned}$$

A4 $(\forall\, e \in E)\; l_e = 0$

$u_e := u_e - l_e$

$$\begin{aligned} (e = vw)\; b_v &:= b_v + l_e \\ b_w &= b_w - l_e \end{aligned}$$

$l_e := 0$

vorher: $\overbrace{v}^{b_v} \xrightarrow{l_e \leqq x_e \leqq u_e} \overbrace{b_w}^{b_w}$ Fluss mit Kosten k

nachher: $\overbrace{v}^{b_v + l_e}\, v \xrightarrow{0 \leqq x_e - l_e \leqq u_e - l_e} \overbrace{w}^{b_w - l_e}$ Fluss mit Kosten $k - c_e l_e$

B (MKFP) auf das Transshipment

$v \xrightarrow{c_e,\, u_e \neq \infty} w \quad \rightarrow \quad v \xrightarrow{c_e, \infty} p \xleftarrow{0, \infty} q \xrightarrow{0, \infty} w$ mit $b_p = u_e$ und $b_q = -u_e$

Vor der Umwandlung $O(m+n)$ Knoten und $O(m)$ Kanten. Nach der Umwandlung $O(n+2m)$ Knoten und $O(3m)$ Kanten. Das ist in der Praxis ineffizient, aber genug um zu zeigen, dass gewisse Algorithmen polynomiell sind.

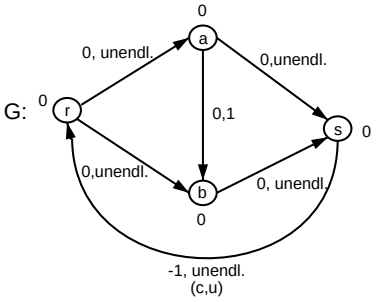
4.2 Primale Minimum Kosten Fluss Algorithmen

4.2.1 Basis-Algorithmus

Dieser Algorithmus stammt von Kanorovich [1942]. Er läuft so:

Finde eine zulässige Lösung x;
while ($\exists$ x-erhöhender Kreis mit negativen Kosten) **do**
  Finde einen x-erhöhenden Kreis C mit negativen Kosten;
  if (C hat keine Rückwärtskanten und keine Vorwärtskanten mit endl. Kapazität) **then**
    STOP „unbeschränkt“;
  end if
  Augmentiere x auf C;
end while

Dabei stellt sich die Frage, wie viele Flusserhöhungen durchgeführt werden.
Max-Fluss-Problem als spezielles MKFP



x-erhöhende Wege $\hat{=}$ negative Kreise.
$\Rightarrow$ evtl. keine Terminierung.

Abhilfe Max-Fluss: Kürzeste augmentierende Wege
hier: „negativste“ Kreise?
nein: die Kreise im Beispiel sind „negativst“, stattdessen:
minimale Durchschnittskosten für Kreis C

$$:= \frac{\text{Kosten } C}{\text{Länge } C}$$

$\hat{=}$ kürzeste Wege im Max-Fluss.
Zeit pro Iteration $O(mn)$ mit Bellmann Ford $\leftarrow$ zu teuer
aber: ergibt polynomiellen Algorithmus (nach Goldberg u. Tarjan [1989])

4.2.2 Die Netzwerk-Simplex Methode

Erinnerung: Satz 2.10: Spaltenbasen entsprechen eins zu eins den aufspannenden Bäumen.

Hier anderer Zugang: Spezialisierung des Basis-Algorithmus. Annahme: G ist zusammenhängend (OBdA: sonst Zusammenhangs-Komponenten getrennt behandeln)

$\rightarrow G$ hat aufspannenden Baum (ab jetzt genannt Baum)
Zunächst Spezialisierung auf das Transshipment-Problem. ($u_e = \infty \; \forall \, e \in E$)

Baum-Lösung $x \in \mathbb{R}^E$ mit:

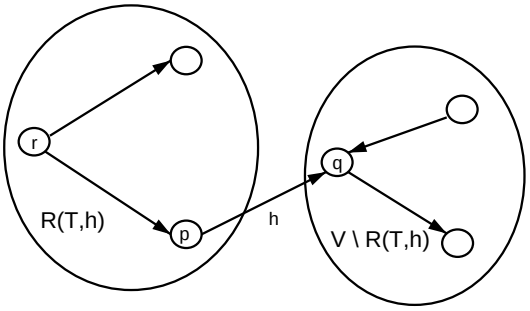
$$\begin{aligned} f_x(v) &= b_v \quad \forall v \in V \\ x_e &= 0 \quad \forall \, e \notin T \end{aligned}$$

für einen Baum T

Lemma 4.6 *Zu je zwei Knoten u und v eines Baumes T existiert ein eineindeutiger (u,v)-Weg in T. Beweis klar. q.e.d*

Lemma 4.7 *Ein Baum T bestimmt eindeutig seine Baumlösung.*

Beweis:
$r \in V(T)$ beliebig fest „Wurzel“
$h = pq \in E(T)$
$R(T,h) = \{v \in V(T) | \text{eindeutiger } (r,v)\text{-Weg in T benutzt } h \text{ nicht}\}$



$$\Rightarrow x_h = \begin{cases} -b(R(T,h)) \text{ falls } p \in R(T,h) \\ b(R(T,h)) \text{ falls } q \in R(T,h) \end{cases} \quad \text{q.e.d.}$$

Die Umkehrung gilt nicht, z.B. alle $b_v = 0 \Rightarrow$ alle Baumlösungen $\hat{=} \, x = 0$

Satz 4.8 *Hat (G,b) eine zulässige Lösung, so auch eine zulässige Baumlösung, hat (G,b) eine optimale Lösung, so auch eine optimale Baumlösung.*

Beweis: Sei x eine zulässige Lösung, aber keine Baumlösung
$\Rightarrow \exists$ Kreis C mit positivem Fluss auf jeder Kante (OBdA habe C eine Rückwärtskante, sonst wähle C anders herum).

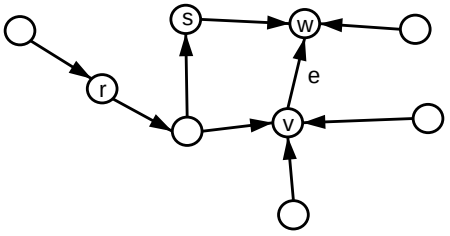
$$\varepsilon := \min\{x_e | e \text{ ist Rückwärtskante}\} \; x_e \rightarrow \begin{cases} x_e + \varepsilon \text{ falls } e \text{ Vorwärtskante in } C \\ x_e - \varepsilon \text{ falls } e \text{ Rückwärtskante in } C \end{cases}$$

Resultat: Ein neues zulässiges x mit weniger Kanten mit positivem Fluss
Schließlich: Baumlösung

Jetzt x optimale Lösung, Kreis C mit positivem Fluss.
Aus Satz 4.2: (zulässiges x ist optimal g.d.w. kein neg. Kreis) $\Rightarrow$ C hat Kosten 0
Weiter wie oben

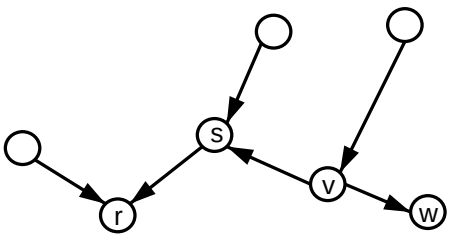
Grundidee der Netzwerk-Simplex-Methode

- Erzeuge Eine Folge von Baumlösungen
- Suche nach speziellen negativen Kreisen zu jeder Kante $e = vw \notin T$ existiert ein eindeutiger Kreis C(T,e) mit:
  - $f \in C(T,e) \rightarrow f \in T \cup \{e\}$
  - e ist Vorwärtskante in $C(T,e)$
  - Der „Anfangsknoten“ s von $C(T,e)$ ist der erste gemeinsame Knoten der einfachen (v,r)- und (w,r)-Wege in T



Satz 4.9 *Definiert der Baum T die zulässige Baumlösung und hat $C(T,e)$ nicht-negative Kosten für alle $e \notin T$, so ist x optimal.*

Beweis: Sei $y \in \mathbb{R}^V$ mit $y_v =$ Kosten des einfachen (r,v)-Wege in T. Für alle $v,w \in V$ gilt: Kosten des einfachen (v,w)-Wege $= y_w - y_v$.

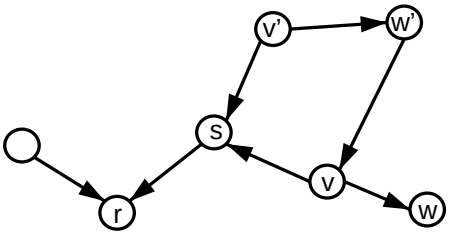


Also gilt für die reduzierten Kosten: $\bar{c}_{vw} = c_{vw} + y_v - y_w$

falls $vw \in T$: $\bar{c}_{vw} = c_{vw} + y_v - y_w = y_w - y_v + y_v - y_w = 0$

falls $vw \in E \setminus T$: $\bar{c}_{vw} = \underbrace{c_{vw}}_{\text{Kosten } vw} + \underbrace{y_v - y_w}_{\text{Kosten im Baum von } w \text{ nach } v}$

$= \text{Kosten von } C(T, vw) \geqq 0$



D.h. die Bedingung von Satz 4.1 (Komplementärer Schlupf $\bar{c}_e > 0 \Rightarrow x_e = 0$) sind erfüllt.
$\overset{\text{Satz 4.1}}{\Rightarrow}$ Optimalität.

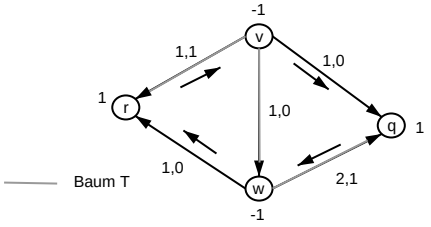
Test ob T diese Optimalitätsbedingungen erfüllt:

- Berechne y: Zeit $O(n)$
- Berechne $\bar{c}$: Zeit $O(m)$

} Besser als $O(mn)$ für Bellmann-Ford aber:

Problem: Verbessert $C(T,e)$ die Lösung?

$C(T,e)$ kann eine Rückwärtskante mit 0-Fluss haben.



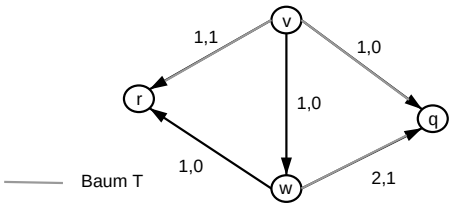
$C(T,wr)$ hat positive Kosten 1.

$C(T,vq)$ hat negative Kosten -2, aber Rückwärtskante vw mit 0-Fluss.

Aber x ist nicht optimal.

Eine Einheit Fluss auf dem durch die kurzen Pfeile markierten Kreis liefert eine bessere Lösung (um -1).

Trotzdem:

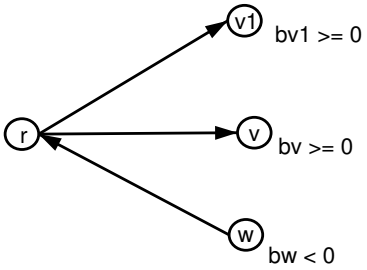


Netzwerk-Simplex-Algorithmus für das Transshipment-Problem

Finde Baum T mit zulässiger Baumlösung x;

Berechne für alle $v \in V$ $y_v =$ Kosten des (r,v)-Weges in T;
while $(\exists e = vw\ (e \notin T)$ mit $\bar{c}_e = c_e + y_v - y_w < 0)$ **do**
  Finde solches e;
  if ($C(T,e)$ hat keine Rückwärtskante) **then**
    STOP unbeschränkt;
  end if
  Berechne $\Theta = \min\{x_j | j$ rückwärts in $C(T,e)\}$
  Bestimme Rückwärtskante h von $C(T,e)$ mit $x_h = \Theta$;
  Augmentiere x um Θ auf $C(T,e)$;
  Setze $T := (T \cup \{e\}) \setminus \{h\}$;
  Berechne neue y;
end while

Erzeugung eines ersten Baumes (Initialisierung):

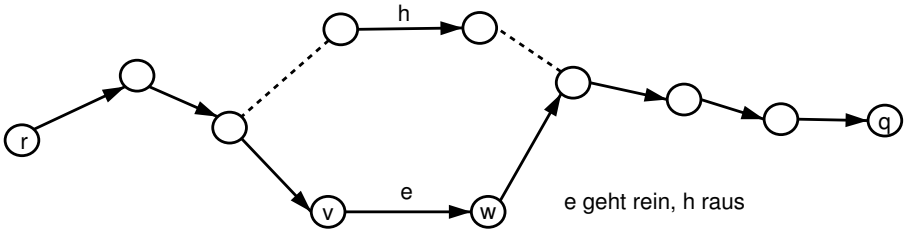


Künstliche Kanten, Hohe kosten M falls $\notin G$
$M := n \cdot \max\{|c_e| :\ e \in E\} + 1$

Lemma 4.10 *Sei T der alte $\hat{T}$ der neue Baum, so gilt für $e = vw$:*

$$\hat{y}_q = \begin{cases} y_q & \forall\, q \in R(T,h) \\ y_q + \bar{c}_e & \forall\, q \notin R(T,h), \textit{falls } v \in R(T,h) \\ y_q - \bar{c}_e & \forall q \notin R(T,h), \textit{falls } w \in R(T,h) \end{cases}$$

Beweis:



für $q \in R(T,h)$: gleicher Weg $\rightarrow$ gleiche Kosten
für $q \notin R(T,h)$ und $v \in R(T,h)$

$$\begin{aligned} \hat{y}_q &= y_v + c_e + (y_q - y_w) \\ &= y_q + \bar{c}_e \end{aligned}$$

für $q \notin R(T,h)$, $w \in R(T,h)$ analog. q.e.d.

Endlichkeit des Netzwerk Simplex-Algorithmus

Sei T Baum mit 0-Flusskante ist *degeneriert* (hat 0-Kante), dann ist Zykeln möglich (Übungsaufgabe), aber vermeidbar. Dafür zunächst eine Definition:
$h = pq \in T$ ist *von r* gerichtet, falls $p \in R(T,h)$
$h = pq \in T$ ist *auf r* gerichtet, falls $q \in R(T,h)$

T ist *stark zulässig* falls für den zugehörigen Fluss gilt:
$\forall\, h \in T$ mit $x_h = 0$ ist h von r gerichtet.

Jeder nicht degenerierte Baum und der Anfangsbaum sind stark zulässig.

C-Regel: Wähle h als *erste* Rückwärtskante von $C(T,e)$ mit $x_h = \Theta$. Diese Regel wurde 1976 von Cunningham aufgestellt.

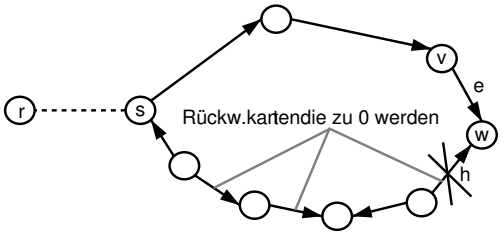
Lemma 4.11 *Ist T stark zulässig und entsteht $\hat{T}$ aus T mit der C-Regel, so ist auch $\hat{T}$ stark zulässig.*

Beweis: Situation Ee rein, h raus, x alter, $\hat{x}$ neuer Fluss.

1. $g \notin C(T,e)$
   g von r in $\hat{T} \Leftrightarrow g$ ist von r in T:
   $\hat{x}_g = x_g$

2. $g \in C(T,e)$

   (a) $\Theta > 0$:
   $\hat{x}_g = 0 \Leftrightarrow$ rückwärts in $C(T,e)$ und $x_q = \Theta$
   h ist erste von diesen



   Es wird h genommen $\Rightarrow$ alle anderen sind von r.

(b) $\Theta = 0$

$\hat{x}_g = 0$ g.d.w. $x_g = 0$ (incl. $\hat{x}_e = 0$)

T stark zulässig $\Rightarrow$ alle diese außer e vorwärts in (s,v) und (s,w)-Wegen in T.

Letzter von diesen wird gewählt.

$\Rightarrow$ alle anderen sind von r in $\hat{T}$

e ist vorwärts in $C(T,e) \Rightarrow e$ von r in $\hat{T}$

$\Rightarrow \hat{T}$ ist stark zulässig q.e.d.

Satz 4.12 *Startet der Netzwerk-Simplex-Algorithmus mit einem stark zulässigen Baum und benutzt er die C-Regel, so terminiert er.*

Beweis: Wir zeigen: Zykeln unmöglich.

Sei $\underset{(y)}{T} \overset{h \text{ raus, } e \text{ rein}}{\Longrightarrow} \underset{(\hat{y})}{\hat{T}}$ degenerierte Iteration.

h sei Vorwärtskante auf (s,w)-Weg.

$$R(T,h) \xrightarrow{h} \ldots \rightarrow w$$

$\Rightarrow w \notin R(T,h) \Rightarrow v \in R(T,e)$

$\overset{4.10}{\Rightarrow} \hat{y}_w = y_w + \underbrace{\bar{c}_e}_{<0} < y_w$

$\Rightarrow \hat{y}_a \leqq y_a \ \forall\, a \in V$ und $\hat{y}_w < y_w$

$\Rightarrow \sum_{a \in V} \hat{y}_a < \sum_{a \in V} y_a$

$\Rightarrow$ keine Wiederholung derselben Situation. q.e.d.

Generalisierung von Transshipment auf Minimum Fluss

(Skizze) analog Simplex-Algorithmus mit unteren und oberen Schranken.
Baum-Lösung: $x \in \mathbb{R}^E$ $\exists$ Baum T und eine Partition (L,U) von $E \setminus T$ mit:

$$\begin{aligned} f_x(v) &= b_v \\ x_e &= 0 \ \forall\, e \in L \\ x_e &= u_e \forall\, e \in U; \end{aligned}$$

Satz 4.13 *Hat (G,b,u) eine zulässige (b.z.w optimale) Lösung so auch eine zul. (b.z.w. optimale) Baumlösung*

Beweis analog. q.e.d.

Iteration wie vorhin, aber wenn $x_e = u_e$ dann Kreis andersherum (e rückwärts).

Gegeben: (T,L,U), $e \in L \cup U$
Kreis $C(T,L,U,e)$:

- Kanten alle aus $T \cup \{e\}$
- e vorwärts, falls $e \in L$, sonst rückwärts
- Anfangsknoten s wie gehabt.

y_v $(v \in V)$ wie gehabt, Kosten von $C(T,L,U,e) = \begin{cases} \bar{c}_e & \text{falls } e \in L \\ -\bar{c}_e & \text{falls } e \in U \end{cases}$

Analogon zu Satz 4.9:

Satz 4.14 *Definiert der Baum (T,L,U) die zul. Baumlösung x und hat $C(T,L,U,e)$ nicht-negative Kosten für alle $e \notin T$, so ist x optimal.*

Netzwerk-Simplex-Methode für Minimum-Kosten-Flussprobleme

Finde Baum (T,L,U) mit zul. Baumlösung;
Berechne für alle $v \in V$ $y_v =$ Kosten eines (r,v)-Weges in T;
while ($\exists e = vw \in L$ mit $\bar{c}_e < 0$ oder $\exists e \in U$ mit $\bar{c}_e > 0$) **do**
  Finde ein solches e;
  if ($C(T,L,U,e)$ hat keine Rückwärtskante und keine Vorwärtskante endl. Kapazität)
  then
    STOP „unbeschränkt“;
  end if
  Berechne $\Theta_1 := \min\{x_j | j$ rückwärts in $C(T,L,U,e)\}$
  $\Theta_2 := \min\{u_j - x_j | j$ vorwärts in $C(T,L,U,e)\}$
  $\Theta := \min\{\Theta_1, \Theta_2\}$;
  Finde $h \in C(T,L,U,e)$ mit h rückwärts und $x_h = \Theta$ oder h vorwärts und $u_e - x_h = \Theta$;
  Augmentiere x um Θ auf $C(T,L,U,e)$;
  $T := (T \cup \{e\}) \setminus \{h\}$;
  Berechne neues L und neues U;
  Berechne neues y;
end while

Stark zulässige Baumlösung T,x:
$\forall\, e \in T$ mit $x_e = 0$ ist e von r gerichtet
$\forall\, e \in T$ mit $x_e = u_e$ ist e auf r gerichtet

C-Regel: Wähle erste mögliche Kante

4.3 Primal-Duale min Kosten Flussalgorithmen

Idee: In jeder Iteration haben wir ein Paar (x,y) mit $0 \leqq x \leqq u$ und Optimalitätsbedingungen aus Satz 4.1:

$$x_e = u_e \ \forall\, e \in E \text{ mit } \bar{c}_e < 0$$
$$x_e = 0 \ \forall\, e \in E \text{ mit } \bar{c}_e > 0$$

Zusammen mit $0 \leqq x \leqq u$ sind dies die Primal-Dual-Bedingungen.

Ziel: Erfüllung der Flussbedingungen:

$$f_x(v) = b_v \ \forall\, v \in V$$

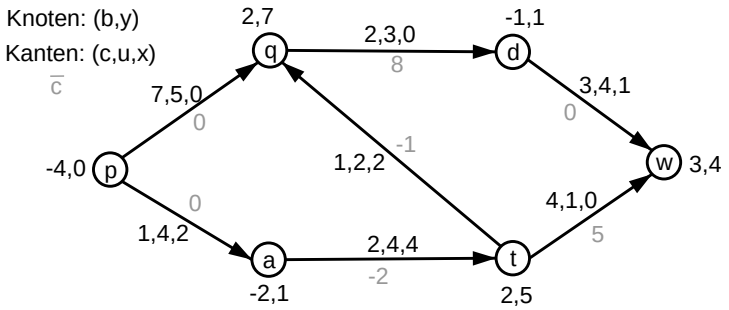
Erste Lösung:
$c \geqq 0 : \ x = 0,\ y = 0$
sonst Lösen des kürzesten Wege-Problems wie in Satz 4.4.
Resultat: Entweder $\nexists$ Optimallösung oder y mit $\bar{c}_e \geqq 0$ wenn $u_e = \infty$

Wähle $x_e = u_e$ falls $\bar{c}_e < 0$
$x_e = 0$ sonst

Also: Anfangslösung kann durch höchstens eine kürzeste Wege-Berechnung ermittelt werden.
Und Existenz einer Lösung: Max-Fluss-Problem
Ab jetzt (OBdA): Existenz einer Optimallösung gesichert.

$v \in V$ heißt x-Quelle, falls $b_v < f_x(v)$ „zu viel Fluss“
x-Senke, falls $b_v > f_x(v)$ „zu wenig Fluss“

Beispiel:



p ist einzige x-Quelle und w ist einzige x-Senke.
Es sind alle Bedingungen erfüllt → Zulässiges Paar x, y

Suche x-erhöhenden Weg von x-Quelle zu einer x-Senke, Hoffnung: mehr erfüllte Flussbedingungen.

Begrenzung der Erhöhung ε:

$$\begin{aligned} \varepsilon &\leqq b_s - f_x(s) \\ \varepsilon &\leqq f_x(r) - b_r \\ \varepsilon &\leqq \text{ Kapazität von P} \end{aligned}$$

1. Versuch: $P = p,q,d,w$
Erhöhung von x scheitert an $\bar{c}_{qd} = 8 > 0$
Analog bei $\bar{c}_{ij} < 0$ kann x_{ij} nicht erniedrigt werden.
Also: P darf nur „Gleichheitskanten“ e haben, d.h. $\bar{c}_e = 0$

Annahme $\nexists$ solches P
$\Rightarrow (\exists R \subseteq V)$ R enthält alle x-Quellen und keine x-Senken (R enthält alle von einer x-Quelle erreichbaren Knoten auf x-erhöhenden Wegen aus Gleichheitskanten) und:

$$\begin{aligned} e \in \delta(R) &\Rightarrow \bar{c}_e \neq 0 \text{ oder } x_e = u_e \\ e \in \delta(\bar{R}) &\Rightarrow \bar{c}_e \neq 0 \text{ oder } x_e = 0 \end{aligned}$$

Im Beispiel $R = \{p,q,a\}$

Definiere Partition:

$$\delta(R) \cup \delta(\bar{R}) = T_1 \cup T_2 \cup T_3 \cup T_4$$

mit $T_1 = \{e \in \delta(R) | \bar{c}_e \leqq 0,\ x_e = u_e\}$
$T_2 = \{e \in \delta(R) | \bar{c}_e > 0,\ x_e = 0\}$
$T_3 = \{e \in \delta(\bar{R}) | \bar{c}_e \geqq 0,\ x_e = 0\}$
$T_4 = \{e \in \delta(\bar{R}) | \bar{c}_e < 0, x_e = u_e\}$

Im Beispiel:

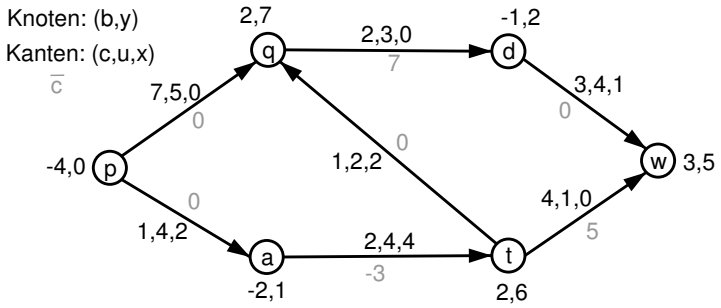
$$\begin{aligned} T_1 &= \{at\} \\ T_2 &= \{qd\} \\ T_3 &= \varnothing \\ T_4 &= \{tq\} \end{aligned}$$

Duale Änderung: Erhöhe y_v ($v \in \bar{R}$) um $\sigma > 0$
$e \in \gamma(R) \cup \gamma(\bar{R})$: $\bar{c}_e$ unverändert, mit $\gamma(R) =$ „innerhalb R“
$e \in \delta(R)$: $\bar{c}_e \to \bar{c}_e - \sigma$
$e \in \delta(\bar{R})$: $\bar{c}_e \to \bar{c}_e + \sigma$
$\Rightarrow$ Bedingungen sind noch erfüllt, falls $\sigma \leqq \bar{c}_e \ \forall\, e \in T_2$, $\sigma \leqq -\bar{c}_e \ \forall\, e \in T_4$

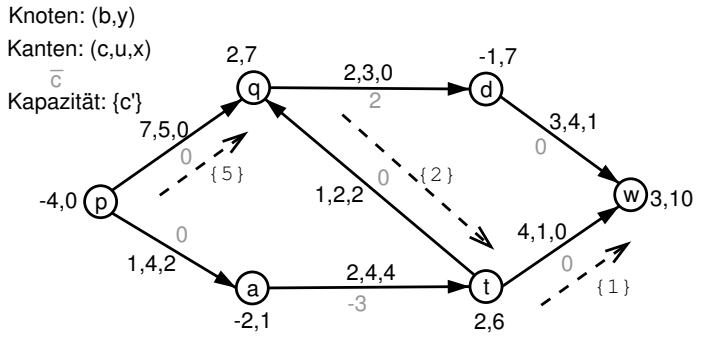
σ maximal: Für eine Kante $e \in T_2 \cup T_4$ ist $\bar{c}_e^{(\text{neu})} = 0$. D.h. R blockt nicht mehr die Suche nach x-erhöhendem Weg mit Gleichheitskanten.

Im Beispiel: $\begin{aligned} \sigma &:= \min\{\bar{c}_{qd},\, -\bar{c}_{tq}\} \\ &= \min\{8, 1\} \\ &= 1 \end{aligned}$

$\Rightarrow tq$ wird Gleichheitskante.



zweite Duale Änderung:



Also erfolgt eine Flusserhöhung um 1 auf dem Gestrichelten Weg.

Was aber tun, wenn $T_2 = T_4 = \varnothing$

Annahme: $T_2 = T_4 = \varnothing$
$\bar{R}$ enthält wenigstens eine x-Senke und keine x-Quelle.

$$\begin{aligned} \Rightarrow b(\bar{R}) &> \sum_{v \in \bar{R}} f_x(v) \\ &= \underbrace{x(\delta(R))}_{T_1} - \underbrace{x(\delta(\bar{R}))}_{T_3} \\ &= u(\delta(R)) - 0 \\ &= u(\delta(R)) \quad \text{Widerspruch} \end{aligned}$$

Denn Bedarf $\bar{R}$ > Kapazität R.
Also ist $T_2 = T_4 = \varnothing$ unmöglich. Eine duale Änderung erzeugt immer wenigstens eine neue Gleichheitskante.

Primal Dualer Algorithmus für das Minimum Kosten Flussproblem von Ford-Fulkerson, (1956):

Finde x, y mit $x_e = u_e \quad \forall\, e \in E$ mit $\bar{c}_e < 0$
$x_e = 0 \quad \forall\, e \in E$ mit $\bar{c}_e > 0$
$0 \leqq x_e \leqq u_e \quad \forall\, e \in E$
while (x ist nicht zulässig) **do**
  if ($\exists$ erhöhender Weg P mit Gleichheitskanten) **then**
    Finde solches P und erhöhe x auf P;
  else
    Finde $\bar{R} \subseteq V$ das alle solche Wege blockt und ändere y für $\bar{R}$;
  end if
end while

- Augmentierungen nach höchstens $n-1$ sukzessiven Änderungen von y
- # Augmentierungen $\leqq \sum_{\substack{v \in V, \\ v \text{ ist } x\text{-Senke}}} b_v - f_x(v)$ falls u, b ganzzahlig.

Falls Anfangsfluss x' nicht ganzzahlig dann nur:

Satz 4.15 *Falls der P-D-MKFP Algorithmus x-erhöhende Wege mit minimaler Kantenzahl benutzt, so terminiert er nach endlich vielen Schritten.*

Beweis: Übung q.e.d. (Hach, ich liebe diese kurzen Beweise :)

Billigste Augmentierende Wege

Wenn x und y die Bedingungen des P-D–Algorithmus erfüllen, $r, s \in V$ und P ein x-erhöhender (r,s)-Weg, so hat:

$$\begin{aligned} c(P) &= \sum_{\substack{vw \\ \text{vorwärts in } P}} c_{vw} - \sum_{\substack{vw \\ \text{rückwärts in } P}} c_{vw} \\ &\geqq \sum_{\substack{vw \\ \text{vorwärts in } P}} y_w - y_v - \sum_{\substack{vw \\ \text{rückwärts in } P}} y_w - y_v \\ &= y_s - y_r \end{aligned}$$

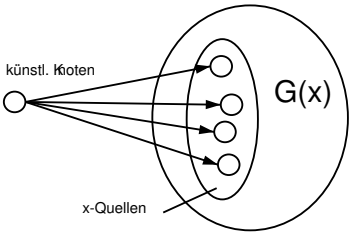
Vorwärtskanten
$x_{vw} < u_{vw} \Rightarrow \bar{c}_{vw} \geqq 0$
$\Leftrightarrow c_{vw} \geqq y_w - y_v$
Rückwärtskanten
$x_{vw} > 0 \Rightarrow \bar{c}_{vw} \leqq 0$
$\Leftrightarrow c_{wv} \leqq y_w - y_v$

nur Gleichheitskanten in P:
$c(P) = y_s - y_r$
d.h. P ist billigster augmentierender Weg.

Idee: Finde Augmentierung durch Lösen kürzester Wege Probleme. Kürzeste Wege Berechnung in G(x), d.h. i.A. negative Kosten: Bellmann-Ford $O(mn)$.
Aber Trick aus Kapitel 2:

Statt c_{vw} Kosten $\bar{c}_{vw} + y_v - y_w$
$\forall$ (r,s)-Wege $\bar{c}(P) = c(P) + y_r - y_s$
P-D-Bedingungen erfüllt $\Rightarrow$ $G(x)$ hat nicht-negative Kosten, d.h. wir können Dijkstra anwenden.

Wir müssen die y berechnen.
$\sigma_v :=$ Kosten eines billigsten (bzgl. $\bar{c}$) x-erhöhenden Wegs von einer x-Quelle nach v ($= \infty$ falls keiner existiert).
Berechnung:



Augmentierung auf (r,s)-Weg P (mit Kosten $\bar{c}$ berechnet).
Es gilt für alle Vorwärtskanten vw in P:

$$\begin{aligned} & \sigma_v + \bar{c}_{vw} = \sigma_w \\ \Rightarrow \quad & c_{vw} + y_v - y_w + \sigma_v - \sigma_w = 0 \\ \Rightarrow \quad & (c_{vw} + y_v + \sigma_v) - (y_w + \sigma_w) = 0 \end{aligned}$$

Gleiches gilt für die Rückwärtskanten analog.
Also für $y'_v := y_v + \sigma_v$ ist P ein Weg mit Gleichheitskanten.
$\Rightarrow x'$ neues x und y' erfüllen die P-D-Bedingungen für alle $e \in P$.
(andere Kanten ?)

4.3.1 Primal-Dualer Algorithmus mit billigsten augmentierenden Wegen

Finde x, y die P-D-Bedingungen erfüllen;
while (x ist nicht zulässig) **do**
  Finde für alle $v \in V$ einen bezüglich $\bar{c}$ billigsten x-erhöhenden Weg P_v von einer x-Quelle zu v;
  $\sigma_v :=$ Kosten von P_v (∞, falls $\nexists P_v$);
  Wähle x-Senke mit minimalem σ_s;
  Augmentiere x auf P_s;

Für alle $v \in V$ $y_v \leftarrow y_v + \min\{\sigma_v, \sigma_s\}$;
end while

Lemma 4.16 *x und y im P-D-Algorithmus mit billigsten augmentierenden Wegen erfüllt die P-D-Bedingungen.*

Beweis:
$(x,y) \to (x',y')$, (x,y) erfüllt die P-D-Bedingungen.

Zu Zeigen: (x',y') erfüllt P-D-Bedingungen.
$0 \leqq x_e \leqq u_e$ bleiben bei jeder Augmentation erhalten.
Komplementäre Schlupfbedingungen gelten für alle $e \in P_s$

Sei $vw \notin P \Rightarrow x'_{vw} = x_{vw}$
Z.Z.:

(a) $x_{vw} \leqq u_{vw} \Rightarrow c_{vw} + y'_v - y'_w \geqq 0$

(b) $x_{vw} > 0 \Rightarrow c_{vw} + y'_v - y'_w \leqq 0$

(a) 1. Fall: $\sigma_v \leqq \sigma_s$

$$\begin{aligned} c_{vw} + y'_v - y'_w &= \bar{c}_{vw} - y_v + y_w + y'_v - y'_w \\ &= \bar{c}_{vw} - y_v + y_w + y_v + \min\{\sigma_v, \sigma_s\} - y_w - \min\{\sigma_w, \sigma_s\} \\ &= \bar{c}_{vw} + \min\{\sigma_v, \sigma_s\} - \min\{\sigma_w, \sigma_s\} \\ &\overset{\sigma_v \leq \sigma_s}{=} \bar{c}_{vw} + \sigma_v - \min\{\sigma_w, \sigma_s\} \\ &\geqq \bar{c}_{vw} + \sigma_v - \sigma_w \\ &\geqq 0 \end{aligned}$$

2. Fall: $\sigma_v \geqq \sigma_s$

$$\begin{aligned} c_{vw} + y'_v - y'_w &= \bar{c}_{vw} + \min\{\sigma_v, \sigma_s\} - \min\{\sigma_w, \sigma_s\} \\ &\overset{\sigma_v \geqq \sigma_s}{=} \bar{c}_w + \underbrace{\sigma_s - \min\{\sigma_w, \sigma_s\}}_{\geqq 0} \\ &\geqq \bar{c}_{vw} \\ &\geqq 0 \end{aligned}$$

x, y erhalten P-D-Bedingungen

(b) analog q.e.d.

Satz 4.17 *Sind b und u ganzzahlig, x der ganzzahlige Ausgangsfluss, und*

$$B_x = \sum_{\substack{v \in V \\ v \text{ x-Senke}}} (b_v - f_x(v))$$

so löst der primal-duale Algorithmus mit billigsten augmentierenden Wegen das Minimum-Kosten-Flussproblem in Zeit $O(S(n,m)B_x)$.

Spezialfall Transshipment: Der Anfangsfluss kann als 0-Fluss gewählt werden, d.h:

$$B_x = B = \sum_{\substack{v \in V \\ b_v > 0}} b_v$$

Lemma 4.18 *Ist b ganzzahlig, so löst der primal-duale Algorithmus mit billigsten augmentierenden Wegen das Transshipment-Problem in Zeit $O(S(n,m)B)$. q.e.d.*

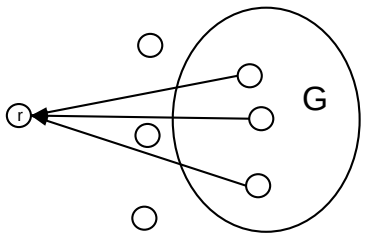
Anm.: also kein polynomieller Algorithmus

4.4 Skalierungsalgorithmen für das MKFP

Annahmen:

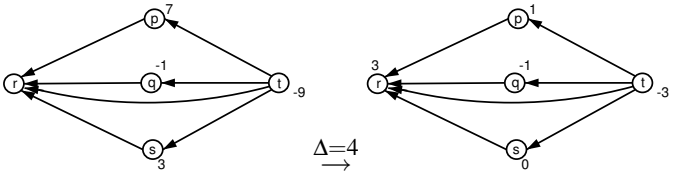
- Transshipment ($u_e = \infty \; \forall \; e \in E$) (O.B.d.A.)
- $b_v \in \mathbb{Z} \; \forall v \in V$
- Existenz einer Optimallösung gesichert (O.B.d.A.)
- $(\exists r \in V) \; b_r = 0$

$$(\forall \; v \in V \setminus \{r\}) \; rv \in E, \; vr \in E, c_{vr} = 0$$



In jeder zulässigen Lösung $x_{vr} = 0$ d.h. O.B.d.A.

Skalierung von (G,c,b) **mit** $\Delta > 0$:
$b_v \to b'_v = \left\lfloor \frac{b_v}{\Delta} \right\rfloor \quad \forall\, v \in V \setminus \{r\}$
$b_r \to b'_r = -b'(V \setminus \{r\})$
$\Rightarrow b'(v) = 0$
Beispiel:



Lemma 4.19 *Hat* (G,c,b) *eine Optimallösung, so auch jede Skalierung von* (G,c,b)

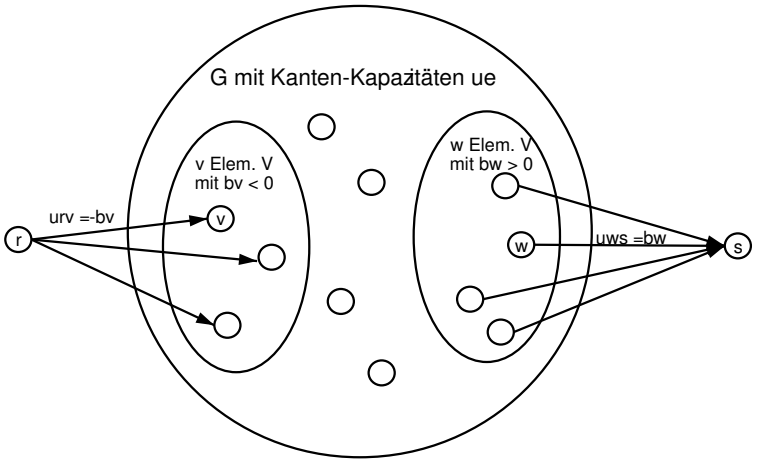
Beweis:

$$\underset{\substack{\text{kein negativer}\\ \text{gerichteter Kreis}}}{(G,c,b)} \xrightarrow{\text{Skaliert mit } \Delta} \underset{\substack{\text{kein negativer}\\ \text{gerichteter Kreis}}}{(G,c,b')}$$

$\Rightarrow (G,c,b')$ ist nicht unbeschränkt, d.h. zu zeigen (G,c,b') hat eine zulässige Lösung.

Der Übergang zur nächsten Aussage wurde mit einem langen Handout von Herrn Professor Jünger gezeigt, dieses sei an dieser Stelle nun zitiert:

> In Übungsaufgabe 36 wurde gezeigt, dass die Zulässigkeit von
>
> $$\begin{aligned} f_x(v) &= b_v \quad \forall\, v \in V \\ 0 \le x_e &\le u_e \quad \forall\, e \in E \end{aligned}$$
>
> für $G = (V,E)$ und $b(V) = 0$ mit Hilfe einer Maximum-Fluss-Berechnung entschieden werden kann. Eine Lösung besteht in der Definition eines Hilfsdigraphen G':



Dann gilt offensichtlich:

$$\begin{aligned}(\exists x)\ f_x(v) &= b_v \quad \forall\, v \in V\\ 0 \le x_e &\le u_e \quad \forall\, e \in E\end{aligned}$$

genau dann, wenn $b(v) = 0$ und $\exists$ (r,s)-Fluss in G' mit Flusswert $\sum_{v\in V,\, b_v>0} b_v$.

Wir können daraus eine andere notwendige und hinreichende Bedingung herleiten:

$\exists$ (r,s)-Fluss in G' mit Flusswert $\sum_{v\in V,\, b_v>0} b_v$ genau dann, wenn $b(v) = 0$ und

$$(\nexists A \subseteq V)\ u(\delta'(A \cup \{r\})) < \sum_{v\in V,\, b_v>0} b_v (\text{Max-Flow-Min-Cut})$$

genau dann, wenn $b(v) = 0$ und:

$$(\nexists A \subseteq V)\ u(\delta(A)) + \sum_{v\notin A,\, b_v<0} (-b_v) + \sum_{v\in A,\, b_v>0} b_v < \sum_{v\in V,\, b_v>0} b_v$$

genau dann, wennn $b(v) = 0$ und

$$\begin{aligned}(\nexists A \subseteq V)\ u(\delta(A)) &< \sum_{v\in V,\, b_v>0} b_v - \sum_{v\in A,\, b_v>0} b_v + \sum_{v\notin A,\, b_v<0} b_v\\ &= \sum_{v\notin A,\, b_v>0} b_v + \sum_{v\notin A,\, b_v<0} b_v\\ &= \sum_{v\notin A} b_v\end{aligned}$$

genau dann, wenn $b(v) = 0$ und

$$(\forall\, A \subseteq V)\ b(A) \le u(\delta(\bar{A}))$$

Es gilt also der

Satz

$$\begin{aligned}(\exists x)\ f_x(v) &= b_v \quad \forall\, v \in V\\ 0 \le x_e &\le u_e \quad \forall\, e \in E\end{aligned}$$

genau dann, wenn $b(v) = 0$ und $(\forall\, A \subseteq V)\ b(A) \le u(\delta(\bar{A}))$.

Für den Spezialfall des Transshipment-Zulässigkeitsproblems ($u_e = \infty\ \forall\, e \in E$) folgt sofort das

Korollar

$$\begin{aligned}(\exists x)\ f_x(v) &= b_v \quad \forall\, v \in V\\ x_e &\ge 0 \quad \forall V \in E\end{aligned}$$

genau dann, wenn $b(v) = 0$ und

$$(\forall A \subseteq V)\ \ \delta(\bar{A}) = \varnothing \Rightarrow b(A) \leq 0$$

(Falls b (und u) ganzzahlig sind, gelten Satz und korollar auch mit ganzzahligem x)

Soviel zum Zitat, weiter in der Vorlesung

$(\forall A \subseteq V)\ \delta(\bar{A}) = \varnothing \Rightarrow b'(A) \leqq 0$

$$\begin{aligned} \delta(\bar{A}) = \varnothing \quad &\Rightarrow \quad r \in \bar{A} \\ &\Rightarrow \quad b'(A) \quad = \quad \sum_{v \in A} \left\lfloor \frac{b_v}{\Delta} \right\rfloor \\ &\qquad\qquad \leqq \quad \sum_{v \in A} \frac{b_v}{\Delta} \\ &\qquad\qquad = \quad \tfrac{b(A)}{\Delta} \end{aligned}$$

Daraus und (G,c,b) hat eine zulässige Lösung $\Rightarrow b(A) \leqq 0$ folgt: Auch $b'(A) \leqq 0$. q.e.d.

Schrittweise Skalieren nach Edmonds und Karp [1972]

Genannt: „successive scaling“

Für $k > 0$ sei (G,c,b^k) die Skalierung von (G,c,b) mit $\Delta = 2^k$. (Entfernen der k letzten Bits in der Dualdarstellung.)
$b^0 = b$
(x', y') sei ein Paar von optimalen primalen/dualen Lösungen für (G,c,b^{k+1}).
$\Rightarrow x = 2x'$, $y = y'$ erfüllen die P-D-Bedingungen für (G,c,b^k).
$x_e \geqq 0$ und $\underbrace{x_e = 0}_{\text{für } (G,c,b^k)} \quad \forall\, e \in E$ mit $\bar{c}_e > 0$

Für $v \neq r$ mit $V = V_1 \cup V_2 \cup \{r\}$ gilt:

$$\begin{aligned} b_v^k \quad &= \quad b_v^{k+1} \qquad v \in V_1 \\ \text{oder: } b_v^k \quad &= \quad b_v^{k+1} + 1 \quad v \in V_2 \end{aligned}$$

$$\begin{aligned} \Rightarrow b_r^k \quad &= \quad -\sum_{v \in V_1} 2b_v^{k+1} - \sum_{v \in V_2} (2b_v^{k+1} + 1) \\ &= \quad -\sum_{v \in V} 2b_v^{k+1} - |V_2| \\ &= \quad 2b_r^{k+1} - |V_2| \leqq 2b_r^{k+1} \end{aligned}$$

$$\rightarrow \sum_{\substack{b_v^k > f_x(v) \\ v \text{ ist } x\text{-Senke}}} b_v^k - f_x(v) \leqq n-1$$

D.h. der P-D-Algorithmus mit Startlösung (x,y) löst (G,c,b) durch höchstens $n-1$ nicht-negative-Kosten-kürzeste-Wege-Berechnungen.

Schrittweiser Skalierungsalgorithmus für das Transshipment Problem

$K := \min\{k \mid |b_v| \leqq 2^k \ \forall\, v \in V\}$; Anzahl der Bits im längsten b_v
$x := 0$;
for $(k := K, K-1, K-2, \ldots, 1)$ **do**
  löse (G,c,b^k) mittels P-D-Algorithmus mit Startlösung $(2x,y)$;
  Sei (x,y) primal/duale Optimallösung;
end for

Satz 4.20 *Sei (G,c,b) eine Instanz des Transshipment-Problems und $b \in \mathbb{Z}^V$ mit Optimallösung. Dann löst der schrittweise Skalierungsalgorithmus (G,c,b) in Zeit $O(nS(n,m)(1+\log B))$ mit:*

$$B = \sum_{\substack{v \in V \\ b_v > 0}} b_v$$

Beweis:
$k < K$: P-D-Algorithmus mit $O(n)$ kürzeste-Wege-Berechnungen (vorhin).

$$\begin{aligned}(G,c,b^k): \sum_{b_v^k > 0} b_v^k &= -\sum_{b_v^k < 0} b_v^k \\ &= |\{v \mid b_v < 0\}| = O(n)\end{aligned}$$

D.h. zusammen $\leqq (K+1)n$ kürzeste-Wege-Berechnungen. $K+1 \leqq 1 + \lceil \log(\max |b_v|) \rceil \leqq 2 + \log B$ q.e.d.

Polynomieller Algorithmus für Transshipment-Problem führt zu polynomiellen Algorithmus für MKFP. Leider gehen die Zahlen b_v $(v \in V)$ in die Laufzeitschranke ein. Edmonds und Karp [1972] fragten, ob es einen *stark polynomiellen Algorithmus* für das MKFP gibt (polynomielle Laufzeitfunktion in $n = |V|$ und $m = |E|$).

Diese Frage wurde 1985 von Eva Tardos mit „ja“ beantwortet. Die Details der resultierenden „Skalier- und Kontraktionsalgorithmen“ sind kompliziert, hier aus Zeitgründen, nur das beste Resultat.

Satz 4.21 *Aufgestellt von Orlin [1988]: Ein gewisser „Skalier und Kontraktionsalgorithmus“ löst das Minimum-Kosten-Flussproblem in Zeit $O(m \log n S(n,m))$.*

Kapitel 5

Optimale Matchings

5.1 Matchings und alternierende Wege

Ungerichteter Graph $G = (V, E)$
Definitionen:

Matching $M \subseteq E$ Jeder Knoten in V ist inzident mit höchstens einer Kante in M.

M **überdeckt** $v \in V$ $(\exists e \in M)$ v inzident e
sonst ist v *M-exponiert*. Anzahl überdeckter Knoten: $2|M|$
Anzahl der exponierten Knoten: $|V| - 2|M|$
$\nu(G)$: Kardinalität eines maximum Matchings
$def(G)$: Minimum Anzahl exponierter Knoten

$$def(G) = |V| - 2\nu(G) \quad \text{„Defizit“}$$

Perfektes Matching Matching $M \subseteq E$ das alle Knoten in V überdeckt.

Probleme:

- Hat G ein perfektes Matching?
- Finde ein maximum Matching in G
- Finde ein perf. Matching mit Minimum Gewicht in G mit Kantengewichten.

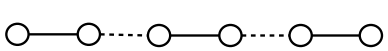
Wenn G bipartit: Max. Matching $\rightarrow$ ganzzahliges Max-Fluss-Problem

$$\left.\begin{array}{l}\text{min. Gewicht}\\ \text{perf. Matching}\end{array}\right\} \rightarrow \text{Min-Kosten-Flussproblem (Übungsaufgabe 35)}$$

Weitere Definitionen:

M-alternierender Weg abwechselnd Kanten $e \in M$ und $e \notin M$

M-augmentierender (-erhöhender) Weg



Konvention: ..... $e \in M$, ___ $e \notin M$

ist ein M-alternierender Weg dessen Endknoten M-exponiert sind.

Für G bipartit:
M-augmentierender Weg $\leftrightarrow$ x augmentierender Weg in der Max-Fluss-Formulierung.
Kap. 3 $\Rightarrow$ M Maximum $\leftrightarrow \nexists$ augmentierender Weg

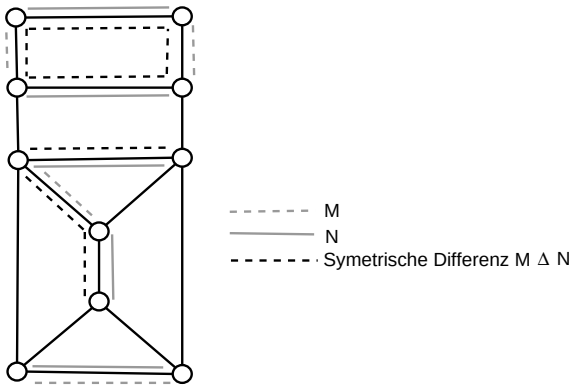
Satz 5.1 *(nach Berge [1957]) Ein Matching in G ist maximum genau dann, wenn kein M-augmentierender Weg existiert.*

Beweis:

„$\Rightarrow$“ $\exists\, M$-augmentierender Weg P
$\Rightarrow M \,\Delta\, E(P)$ überdeckt 2 Knoten mehr (Δ steht für symetrische Differenz, „Drehe Kanten um“)
$\Rightarrow M$ ist nicht maximum

„$\Leftarrow$“ M ist nicht maximum
$\Rightarrow \exists$ Matching N mit $|N| > |M|$
$J := N \Delta M$

Beispiel:



J : Knotendisjunkte alternierende Wege und alternierende Kreise gerader Kardinalität.
$|N| > |M| \Rightarrow$ Ein Weg (von allen) in J hat mehr N-Kanten als M-Kanten. Dieser ist ein M-augmentierender Weg. q.e.d.

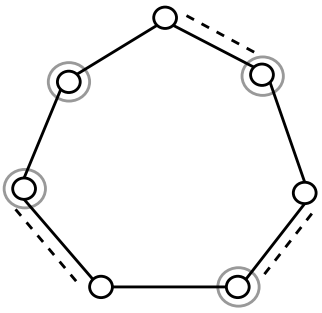
ungerade Kreise:
Knotenüberdeckung: $A \subseteq V$ mit: Jede Kante hat wenigstens einen Endknoten in A, also gilt:

$$\text{Kardinalität einer Knotenüberdeckung} \geqq \text{Kardinalität eines Matchings}$$

Satz von König (Aufgabe 24)

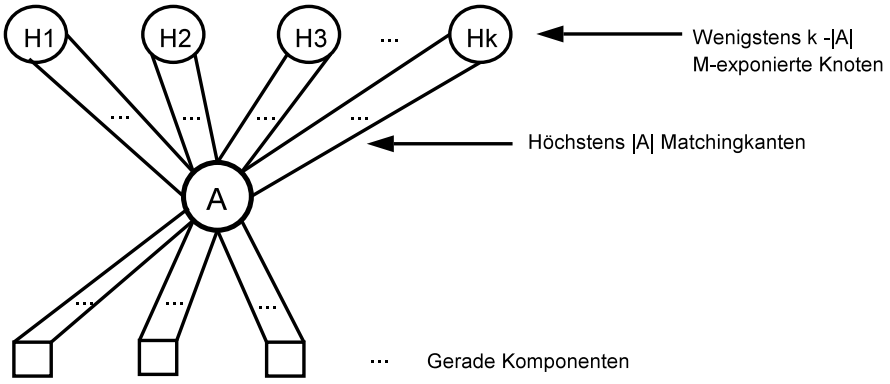
$$\begin{aligned} \text{Wenn } G \text{ bipartit } \quad \Rightarrow \quad & (\exists \text{ Knotenüberdeckung } A) \\ & (\exists \text{ Matching } M) \\ & |A| = |M| \end{aligned}$$

Das gilt *nicht* allgemein, so z.B. nicht in ungeraden kreisen wie dem folgenden mit der Länge $k+1$ (Eine minimale Knotenüberdeckung A ist durch die grauen Kreise, ein maximales Matching M durch die gestrichelten Linien angedeutet):

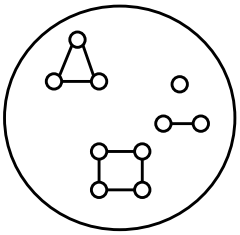


Hier gilt offensichtlich $|M| \leqq k$ und $|A| \geqq k+1$. Wir suchen nun eine bessere Schranke:

Sei $A \subseteq V$ mit $G \setminus A$ (Knoten A gelöscht hat k Komponenten $h_1, h_2, \ldots, h_k$ mit $|V(H_i)|$ ungerade.



Nun definieren wir $OC(G) :=$ Anzahl ungerader Komponenten von G, z.B.:



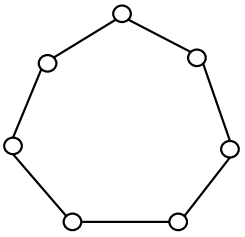
$OC(G) = 2$

Also $(\forall A \subseteq V)$:

$$\nu(G) = \frac{1}{2}(|V| - OC(G \setminus A) + |A|)$$

Falls A Knotenüberdeckung ist:
$|V| - |A|$ ungerade Komponenten in $G \setminus A$ (einzelne Knoten)
$\Rightarrow \nu(G) \leqq \frac{1}{2}(|V| - (|V| - |A|) + |A|) = |A|$

Spezialfall, d.h. die untere Schranke ist wenigstens so gut wie die Überdeckungsschranke, sogar besser:



Ungerader Kreis der Länge $2k+1$
$A = \varnothing$: $\nu(G) \leqq \frac{1}{2}(|V| - 1) = k$
(bisher nur $k+1$, also besser!)

Für jedes $A \subseteq V$ heißt die vorhin bewiesene Schranke für Kardinalität eines Matchings:

$$\nu(G) \leqq \frac{1}{2}(|V| - OC(G \setminus A) + |A|)$$

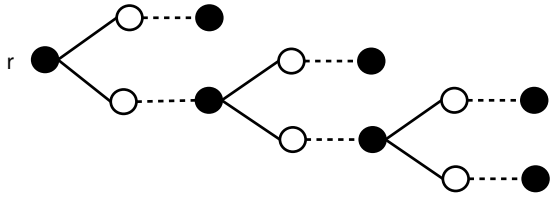
Dies ist die „Tutte-Berge-Schranke“. Es folgt sofort die sog. „Tutte Bedingung“:
Hat $G = (V,E)$ ein perfektes Matching, so gilt für jede Teilmenge $A \subseteq V$: $OC(G \setminus A) \leqq |A|$

Wir werden später sehen, dass die Tutte Berge Schranke angenommen wird und die Tutte-Bedingung auch hinreichend ist.

5.2 Maximum Matching

Zunächst suche nach perfektem Matching (danach max. Matching)

Alternierende Bäume
$G = (V,E)$, $M \subseteq E$ Matching, r ein M-exponierter Knoten, M-alternierender Baum T:

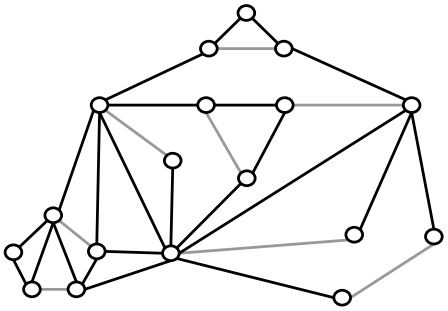


$A(T) = \{v \in V(T) | v \text{ hat ungerade Distanz von } r\}$ „ungerade Knoten“ ∘
$B(T) = \{v \in V(T) | v \text{ hat gerade Distanz von } r\}$ „gerade Knoten “ •

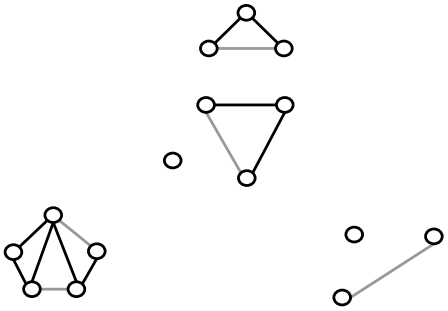
Eigenschaften:

- Jedes $v \in V(T) \setminus \{r\}$ wird von einer Kante $e \in M(T)$ überdeckt.
- Für jedes $v \in V(T)$ ist der (r,v)-Weg in T M-alternierend.
- $|B(T)| = |A(T)| + 1$

Zum Anfang der Vorlesung legte Prof. Jünger einen Beispielgraph auf:



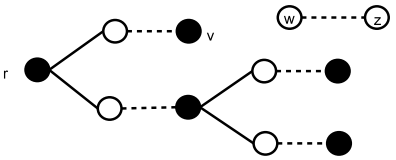
$G = (V,E)$
$|V| = 18$
$|M| = 8$, d.h. zwei exponierte Knoten



$OC(G \setminus A) = 5$
$\overset{\text{TB-Formel}}{\Rightarrow} \nu(G) \leqq \frac{1}{2}(18-5+3) = 8$
$\Rightarrow M$ Matching max. Kardinalität.

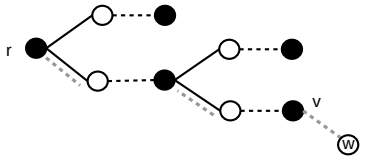
„Unterprogramme"

Benutze *vw* **zur Baumerweiterung**
Eingabe: Matching M' von G', M' alternierender Baum T, $vw \in E(G')$ mit $v \in B(T)$, $w \notin V(T)$, w M'-überdeckt.



Aktion: Sei wz die überdeckende Kante in M'
$T \leftarrow (V(T) \cup \{w,z\},\ E(T) \cup \{vw, wz\})$

Benutze *vw* **zur** M'**-Augmentierung**
Eingabe: Matching M' von G', M' alternierender Baum T mit Wurzel r, $vw \in E(G')$ mit $v \in B(T)$, $w \notin V(T)$, w M'-exponiert.



Aktion: Sei P der (r,v)-Weg in $T \cup \{vw\}$
$M' \leftarrow M' \,\Delta\, E(P)$

Ein M-alternierender Baum T in G ist *frustriert* falls jede Kante in $E(G)$ mit einem Ende in $B(T)$ das andere Ende in $A(T)$ hat.

Lemma 5.2 *Hat G ein Matching M und einen frustrierten M-alternierenden Baum T, so hat G kein perfektes Matching.*

Beweis: $\forall v \in B(T)$ ist $\{v\}$ ungerade Komponente von $G \setminus A(T)$, d.h. für $A = A(T) \subseteq V$ gilt:
$OC(G \setminus A) \geqq |B(T)| > |A(T)| = |A|$
$\overset{\text{Tutte-Bed.}}{\Rightarrow}$ G hat kein perf. Matching.

Der Bipartite Fall

Perf. Matching Algorithmus für bipartite Graphen

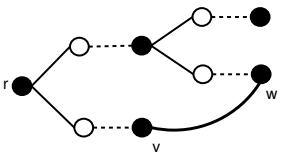
$M \leftarrow \varnothing$;
$T \leftarrow (\{r\}, \varnothing)$; ($r$ beliebig)
while ($\exists vw \in E$ mit $v \in B(T)$, $w \notin V(T)$) **do**

if (w ist M-exponiert) **then**
Benutze vw zur M-Augmentierung;
if ($\nexists$ M-exponierter Knoten in G) **then**
STOP „M ist perfektes Matching“;
else
$T := (\{r\}, \varnothing)$ wobei r M-exponierter Knoten;
end if
else
Benutze vw zur Baumerweiterung;
end if
end while
STOP „G hat kein perf. Matching“;

Lemma 5.3 *Ist in obigem Algorithmus die while-Bedingung nicht erfüllt, so hat G kein perf. Matching.*

Beweis: Wir zeigen T ist frustriert ($\overset{5.2}{\Rightarrow} \nexists$ perf. Matching)
$\nexists vw \in E$ mit $v \in B(T)$, $w \notin V(T)$
$\Rightarrow (\forall vw \in E$ mit $v \in B(T))$ $w \in A(T)$ oder $w \in B(T)$

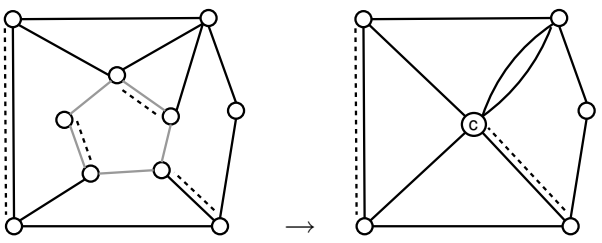
Annahme $w \in B(T)$



$\Rightarrow G$ hat ungeraden Kreis
$= G$ ist nicht bipartit $\lightning \Rightarrow T$ ist frustriert $\overset{5.2}{\Rightarrow} G$ hat kein perf. Matching. q.e.d.

Schrumpfen ungerader Kreise
C ungerader Kreis in G.
$G' := G \times C$: Subgraph von G der durch das Schrumpfen des Kreises C entsteht.

$V(G') = (V \setminus V(C)) \cup c$ (c ist neuer Knoten statt Kreis)
$E(G') = E \setminus \gamma(V(C))$ mit Endknoten in $V(G)$ ersetzt durch c.
Bsp:



Die Matchingkanten sind durch die gestrichelten Linien angedeutet.

Lemma 5.4 *Sei C ein ungerader Kreis in G, $G' = G \times C$ und M' ein Matching in G'. Dann existiert ein Matching M in G mit $M \subseteq M' \cup E(C)$ und die Anzahl M-exponierter Knoten in G ist dieselbe wie die der M-exponierten Knoten in G'.*

D.h. $\nu(G) \geqq \nu(G \times C) + \frac{|V(C)|-1}{2}$
b.z.w.: $def(G) \leqq def(G \times C)$

Der Blütenschrumpfalgorithmus

Im Original: "blossom shrinking“ für perf. Matching von Jack Edmonds, Name des Papers: „Paths, Trees and Flowers“.

Aus G entsteht durch wiederholtes Schrumpfen ungerader Kreise G'. G' hat also zum einen Orginalknoten von G als auch Pseudoknoten, die durch Schrumpfen entstanden sind.

$v \in V(G')$: $S(v)$ Menge der zu v gehörigen Originalknoten (rekursiv ermittelt).
$|S(v)|$ ist immer ungerade ($S(v) = v$ für Originalknoten)
$\{S(v) | v \in G'\}$ Partition von $V(G)$

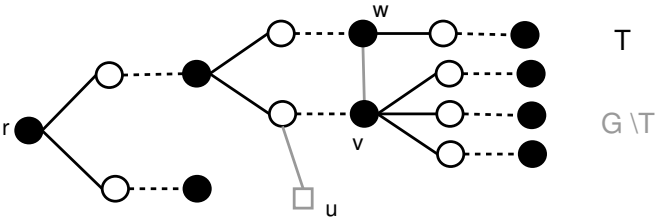
Analogon zu Lemma 5.2:

Lemma 5.5 *Sei G' von G abgeleitet, M' Matching in G', T ein M'-alternierender Baum in G' und kein Pseudoknoten in $A(T)$. Ist T frustriert, so hat G kein perf. Matching*

Beweis: Entfernung von $A(T)$ aus $V(G)$ ergibt für alle $v \in B(T)$ ungerade Komponenten mit Knotenmenge $S(v)$.
$\Rightarrow OC(G \setminus A(T)) \geqq |B(T)| > |A(T)|$
$\overset{\text{Tutte Bed.}}{\Longrightarrow}$ G hat kein perf. Matching. q.e.d.

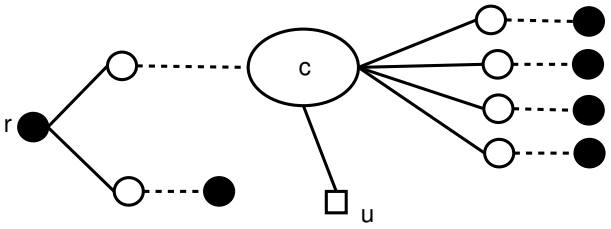
- Produziere abgeleitete Graphen G'
- Perf. Matching in $G' \rightarrow$ perf. Matching in G
- Gewisse frustrierte Bäume in $G' \rightarrow$ kein perf Matching

Welche Kreise schrumpfen wir aber nun?

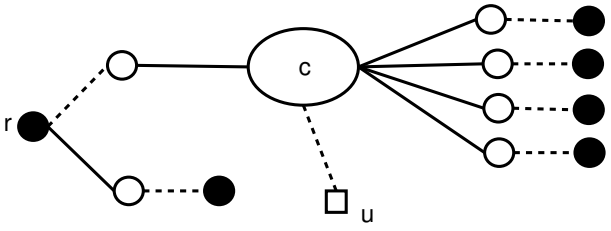


- M ist nicht optimal ($\exists$ perf. Matching!)
- keine Erweiterung/Augmentierung möglich
- T nicht frustriert.

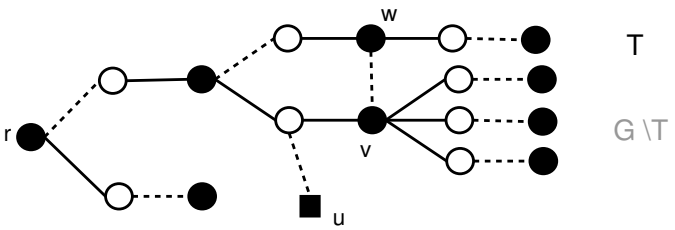
vw erzeugt ungeraden Kreis „Blüte“ (Edmonds)
Schrumpfen der Blüte.



Behalte T und M, Augmentiere:



Entschrumpfen:



Benutze vw zum Schrumpfen und adaptiere M' und T

Eingaben:

- Matching M' von G'
- M'-alternierenden Baum T
- $vw \in E(G')$ mit $v,w \in B(T)$

Sei C der Kreis gebildet aus vw und dem (v,w)-Weg in T.

$G' \leftarrow G' \times C$;
$M' \leftarrow M' \setminus E(C)$;
$E(T) \leftarrow E(T) \setminus E(C)$;

Lemma 5.6 *Nach Anwendung des obigen Unterprogramms „Schrumpf“ ist M' ein Matching in G', T ein M'-alternierender Baum in G' und $c \in B(T)$.*

Beweis: Übungen :)

Blüten-Schrumpf-Algorithmus für perfektes Matching
Eingabe: Graph G und Matching M in G

if (M ist perfekt) **then**
  STOP „M ist ein perfektes Matching“;
end if
$M' \leftarrow M$;
$G' \leftarrow G$;
Wähle M'-exponierten Knoten r in G';
$T \leftarrow (\{r\}, \varnothing)$;
while ($\exists vw \in E'$ mit $v \in B(T)$, $w \notin A(T)$) **do**
  case ($w \notin V(T)$ und w ist M'-exponiert)
    Benutze vw zur M'-Augmentierung;
    Erweitere M' zu einem Matching M von G;
    $M' \leftarrow M$;
    $G' \leftarrow G$;
    if ($\nexists M'$-exponierter Knoten in G') **then**
      STOP „M' ist ein perfektes Matching“;
    else
      $T \leftarrow (\{r\}, \varnothing)$ wobei r M'-exponiert ist;
    end if
  case ($w \notin V(T)$ und w ist M'-überdeckt)
    Benutze vw zur Baumerweiterung;
  case ($w \in B(T)$)
    Benutze vw zum Schrumpfen und adaptiere M' und T;
end while
STOP „G hat kein perfektes Matching“;

Satz 5.7 *Der Blütenschrumpf-Algorithmus terminiert nach: $O(n)$ Augmentierungen*
$O(n^2)$ Schrumpf-Schritten
$O(n^2)$ Baumerweiterungs-Schritten
Er entscheidet korrekt, ob G ein perf. Matching hat.

Beweis: M' ist in jeder Iteration ein Matching, es gibt jedes mal weniger exponierte Knoten $\Rightarrow O(n)$ Augmentierungen.

Jeder Schrumpf-Schritt:

- erniedrigt die Anzahl der Knoten in G'
- erhält die Anzahl, der Nicht-Baumknoten

Jeder Baumerweiterungsschritt:

- erniedrigt die Anzahl der Nicht-Baumknoten
- erhält die Anzahl der Knoten in G'

Zwischen den Augmentierungen höchstens $O(n)$ Schrumpf/Baumerweiterungsschritte
$\Rightarrow$ Insgesamt $O(n^2)$ Schrumpf/Baumerweiterungsschritte

Jedes G' ist aus G abgeleitet
Hält der Algorithmus mit „$\nexists$ perf. Matching“
$\Rightarrow \exists$ frustrierter Baum T mit Voraussetzungen für Lemma 5.5
$\Rightarrow G$ hat kein perf. Matching q.e.d.

Der Blüten-Schrumpf Algorithmus für maximum Matching Der Blüten-Schrumpf Algorithmus für perf. Matching terminierte mit: STOP „G hat kein perfektes Matching“. Wir übernehmen G', M' und T und arbeiten wie folgt weiter:

Entferne $V(T)$ aus G';
Falls ein exponierter Knoten übrig ist: Anwendung des Blütenschrumpf Algorithmus auf neues G' ohne entfernte Kanten. (Neues G' hat keine Pseudoknoten.);
Wiederhole bis kein exponierte Knoten übrig bleibt;
Restauriere Original G' mit allen produzierten Matchingkanten;
Bestimme korrespondierendes Matching in G

Seien $T_1, T_2, T_3 \dots T_k$ die generierten frustrierten Bäume
$\Rightarrow k$-exponierte Knoten (die Wurzeln) in G' UND G
$\Rightarrow |M| = \frac{1}{2}(|V| - k)$
Sei $A := \bigcup_{i=1}^{k} A(T_i)$
Die Entfernung von A aus G ergibt eine ungerade Komponente für jeden Knoten aus $B(T_i)$ für alle $i \in \{1, 2, \dots, k\}$

$$
\begin{aligned}
& (\forall\, T_i) B(T_i) = A(T_i) + 1 \\
\Rightarrow \quad & OC(G \setminus A) \quad \geqq \quad \sum_{i=1}^{k} |B(T_i)| \\
& = \sum_{i=1}^{k} |A(T_i) + 1| \\
& = |A| + k
\end{aligned}
$$

$$\begin{aligned} \overset{\text{Tutte-Berge}}{\Rightarrow} \quad & \frac{1}{2}(|V| - OC(G \setminus A) + |A|) \\ = \quad & \frac{1}{2}(|V| - k) \\ = \quad & |M| \end{aligned}$$

(Obere Schranke: $|M| \leqq \frac{1}{2}(|V| - OC(G \setminus A) + |A|)$
$\Rightarrow |M = \frac{1}{2}(|V| - OC(G \setminus A) + |A|)$
$\Rightarrow$ ist maximum Matching in G

Somit haben wir algorithmisch den folgenden zentralen Satz von Berge [1958] bewiesen.

Satz 5.8 *Tutte Berge Formel*
Für $G(V,E)$ gilt:

$$\nu(G) = \min_{A \subseteq V}\{\frac{1}{2}(|V| - OC(G \setminus A) + |A|)\}$$

Daraus folgt ein älteres Resultat von Tutte [1947]

Korollar 5.9 *(Tuttes Matching Theorem)*
$G = (V,E)$ hat ein perf. Matching genau dann wenn für jede Teilmenge $A \subseteq V$ gilt:

$$OC(G \setminus A) \leqq |A|$$

Ohne Beweis:

Satz 5.10 *(Micali und Vazirani [1980])*
Es gibt eine Implementation des Blüten-Schrumpf Algorithmus mit $O(\sqrt{n}m)$. q.e.d.

Im folgenden „spielte“ Prof. Jünger mit zwei Studenten jeweils auf dem Overheadprojektor ein kleines Spiel auf einen Graphen auf dem ein perf. Matching existiert. Es ging darum dass abwechselnd ein einfacher Pfad (ein Knoten darf nur einmal vorkommen) erweitert. Wer die letzte Erweiterung macht gewinnt.

Satz 5.11 *Hat G ein perfektes Matching, so kann der Spieler der Anfängt seinen Gewinn erzwingen.*

Beweis: Übungen

5.3 Perfektes Matching mit minimum Gewicht

Das Lineare Problem für perfektes Matching (mit) minimum Gewicht.

$$\left.\begin{array}{l} \min \sum_{e\in E} c_e x_e \\ x(\delta(v)) = 1 \quad \forall\, v \in V \\ x_e \geqq 0 \quad \forall e \in E \\ (x_e \in \{0,1\} \ \forall\, e \in E) \end{array}\right\} \begin{array}{l}\text{Relaxierung} \\ \text{(LPPMMG')}\end{array} \Bigg\} \text{Korrekte Formulierung}$$

bipartiter Fall:

Satz 5.12 *(Birkoff [1946])*
Sei G ein bipartiter Graph und $c \in R^E$ dann hat G ein perf. Matching genau dann wenn LPPMMG' eine zulässige Lösung hat. Hat G ein perf. Matching, so ist das minimum Gewicht eines perf. Matchings in G gleich dem opt. ZF-Wert der Relaxierung (LPPMMG')

Beweis erfolgt durch die Korrektheit des folgenden Algorithmus

$$\begin{array}{ll} \text{(DLPPMMG')} & \max \sum_{v\in V} y_v \\ & y_u + y_v \leqq c_e \quad \forall\, e := uv \in E \end{array}$$

Für $y \in \mathbb{R}^V$ und $e = uv$ sei:

$$\bar{c}_e = \bar{c}_e(y) = c_e - (y_u + y_v)$$

y zulässig für (DLPPMMG') $\Leftrightarrow \bar{c}_e \geqq 0 \ \forall e \in E$
Für zulässiges y:
$E_= = E_=(y) = \{e \in E | \bar{c}_e = 0\}$ „Gleichheitskanten bzgl. y"

Komplementärer Schlupf:

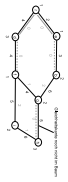
$$\text{(KS)} \quad x_e > 0 \Rightarrow \bar{c}_e = 0 \ \ (\forall\, e \in E)$$

x Inzidenzvektor eines perf. Matchings.

$$\text{(KS)} \Leftrightarrow M \subseteq E_=$$

Für ein zulässiges y bestimme $E_=$ und benutze auf diesen den Blüten-Schrumpf Algorithmus. Resultat: perfektes Matching: Optimallösung oder:
Matching M in $G_=$ und M-alternierender Baum T mit Gleichheitskanten zwischen $B(T)$ und $A(T)$.

Beispiel:



Die grauen Kanten sind die Kanten des Baumes und die grauen Zahlen sind die jeweiligen reduzierten Kosten.

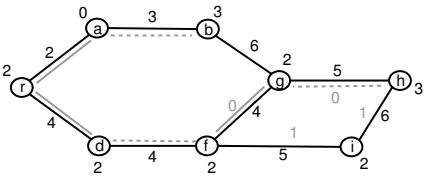
$\exists$ perf. Matching, aber kein perf. Matching in $G_=$

Verändere y
(M und T sollen in $E_=$ bleiben, $\bar{c}_e$ soll kleiner werden für Kanten mit einem Knoten $\in B(T)$ und dem anderen $\notin B(T)$)

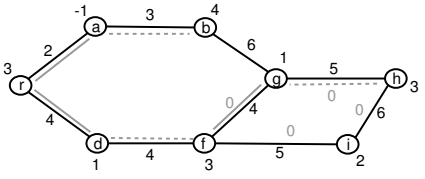
Erhöhe $y_v \leftarrow y_v + \varepsilon \ \forall v \in B(T)$
Erniedrige $y_v \leftarrow y_v - \varepsilon \ \forall v \in A(T)$
Wähle $\varepsilon > 0$ maximal unter:

$$y_u + y_v \leqq c_{uv} \ \ (\bar{c}_{uv} \geqq 0)$$

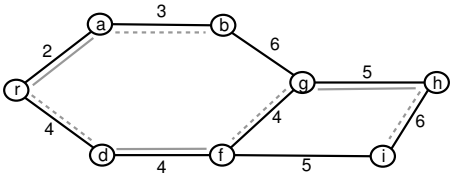
Ergebnis: neue Kante, mit einem Knoten $\in B(T)$, der andere $\notin A(T)$ in $E_=$.
G ist bipartit $\Rightarrow v \notin V(T) \rightarrow$ Augmentierung/Baumerweiterung möglich. Im Beispiel $\varepsilon = 1$ aufgrund der Kante fg.

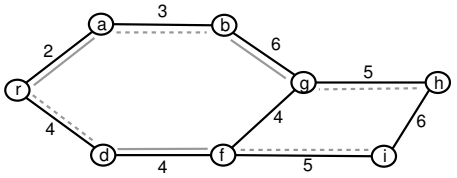


Daraufhin ergibt sich wieder $\varepsilon = 1$:



Diesmal haben wir zwei Möglichkeiten zu Augmentieren, entweder hc oder fc, je nachdem welche wir wählen ergeben sich zwei verschiedene Matchings minimalen Gewichts.





Minimum Kosten perf. Matching Algorithmus für bipartite Graphen

(auch bekannt unter dem Namen „Ungarische Methode“ [Kuhn 1955])
Eingabe: Graph G

Sei y zulässig für ($DLPPMMG'$), M ein Matching in $G_=$;
if M ist perfekt **then**
  STOP „Matching M perfekt mit minimum Kosten“;
end if
$T \leftarrow (\{r\}, \varnothing)$ wobei r M-exponiert ist;
loop
  while ($\exists vw \in E_=$ mit $v \in B(T)$, $w \notin V(T)$) **do**
    if (w ist M-exponiert) **then**
      Benutze vw zur M-Augmentierung;
      if ($\nexists$ M-exponierter Knoten in G) **then**
        STOP „Matching M perfekt mit minimum Kosten“;
      else
        $T \leftarrow (\{r\}, \varnothing)$ wobei r M-exponiert ist;
      end if
    else
      Benutze vw zur Baumerweiterung;
    end if
  end while
  if ($\forall vw \in E$ mit $v \in B(T)$ gilt $w \in A(T)$) **then**
    STOP „G hat kein perfektes Matching“;
  else
    $\varepsilon \leftarrow \min\{\bar{c}_{vw} | v \in B(T),\ w \notin V(T)\}$;
    forall $(v \in B(T))$ $y_v \leftarrow y_v + \varepsilon$;
    forall $(v \in A(T))$ $y_v \leftarrow y_v - \varepsilon$;
  end if
end loop

Laufzeit:
Schlimmstenfalls eine Dualänderung pro Baumerweiterung, d.h. $O(n^2)$ Dualänderungen mit jeweils $O(m)$ Arbeit, insgesamt $O(n^2 m)$ Zeit. Kann verbessert werden zu $O(nS(n,m))$ Zeit.

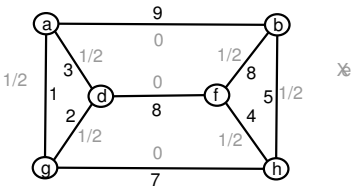
Erstes zulässiges y kann in Zeit $O(n^2)$ gefunden werden: Sei (V_1, V_2) Bipartition von G.

forall $(v \in V_1)$ $y_v \leftarrow \min\{c_{vw} | w \in V_2\}$
forall $(w \in V_2)$ $y_w \leftarrow \min\{c_{vw} - y_v | v \in V_1\}$

Dann gilt $\bar{c}_{vw} = c_{vw} - y_v - y_w \geqq\!\!\rightarrow 0$ nach Konstruktion.

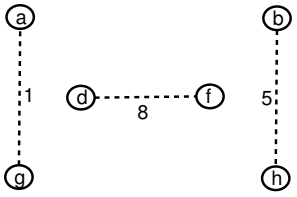
Allgemeiner Fall (nicht bipartit)

Im allgemeinen Fall gilt Satz 5.12 nicht, z.B.:

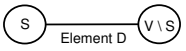


Wert: $\sum_{e \in E} c_e x_e = \frac{1}{2}(1+2+3+4+5+6) = 10,5$

Aber: $\exists$ Minimum perf. Matching mit Kosten 14:



Entwicklung einer besseren Schranke (Edmonds[1965])
Schnitt $D = \delta(S)$ ungerade, falls |S| ungerade.



In jedem perf. Matching M existiert eine Kante aus $\delta(S)$
x Inzidenzvektor eines perf. Matchings in G.
$\Rightarrow$ ($\forall$ ungerade Schnitte D in G) $x(D) \geqq 1$

Hinzufügen von $x(D) \geqq 1$ ergibt bessere Schranke. Im Beispiel wählen wir $D = \{ab, df, gh\}$ ergibt Optimalwert 14.

$\mathscr{C}$ = Menge aller ungeraden Schnitte, die nicht der Form $\delta(v)$ für $v \in V$ sind.

$$\begin{array}{lrcll} \text{(LPPMMG)} & \min \sum_{e \in E} c_e x_e & & & \\ & x(\delta(v)) & = & 1 & \forall\, v \in V \\ & x(D) & \geqq & 1 & \forall\, D \in \mathscr{C} \\ & x_e & \geqq & 0 & \forall\, e \in E \end{array}$$

Damit erhalten wir nicht nur eine bessere untere Schranke, sondern:

Satz 5.13 *Matching Polytop Theorem von Edmonds [1965]*

Sei $G=(V,E)$ ein Graph und $c \in R^E$. Dann hat G ein perf. Matching genau dann, wenn (LPPMMG) eine zulässige Lösung hat. Hat G ein perf. Matching, so ist das minimum Gewicht eines perf. Matchings in G gleich dem optimalen ZF-Wert von (LPPMMG) (vgl. Birkoff für bipartiten Fall)

Beweis: Korrektheit durch folgenden Algorithmus. q.e.d.

$$\begin{array}{lll} \text{(DLPPMMG)} & \max \sum_{v\in V} y_v + \sum_{D\in\mathscr{C}} Y_D & \\ & y_v + y_w + \sum_{e\in D\in\mathscr{C}} Y_D \leqq c_e & \forall\, e=vw\in E \\ & Y_D \geqq 0 & \forall\, D\in\mathscr{C} \end{array}$$

Für (y,Y) und $e \in E$:

$$\bar{c}_e = \bar{c}_e(y,Y) = c_e - y_v - y_w - \sum_{e\in D\in\mathscr{C}} Y_D$$

(y,Y) zulässig für (DLPPMMG) $\Leftrightarrow$ $\{$ $\begin{array}{l} Y_D \geqq 0 \ \forall\, D\in\mathscr{C} \\ \bar{c}_e \geqq 0 \ \forall\, e\in E \end{array}$

Komplementärer Schlupf:

$$\begin{array}{lllll} \text{(KS)} & x_e > 0 & \Rightarrow & \bar{c}_e = 0 & (\forall\, e\in E) \\ & Y_D > 0 & \Rightarrow x(D) = 1 & (\forall D\in\mathscr{C}) & \end{array}$$

x Inzidenzvektor eines perf. Matchings:

$$\text{(KS)} \Rightarrow \begin{array}{lllll} & e\in M & \Rightarrow & \bar{c}_e = 0 & (\forall\, e\in E) \\ & Y_D > 0 & \Rightarrow |M\cap D| = 1 & (\forall\, D\in\mathscr{C}) & \end{array}$$

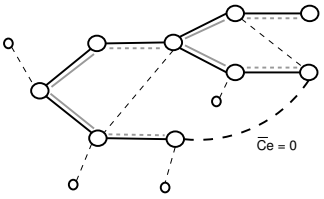
G' abgeleitet von G

$\Rightarrow$ Jeder ungerade Schnitt in G' ist eine ungerader Schnitt in G. D.h. jeder Schnitt der Form $\delta_{G'}(v)$ korrespondiert zu einem ungeraden Schnitt in G. Nur solche erhalten positive Y_D. D.h. Variablen y_v reichen, hier $y_v \geqq 0$

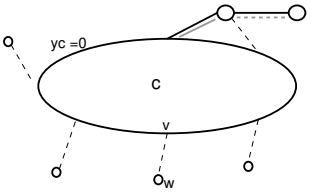
Vorgehen analog zum bipartiten Fall.
y Änderung: ε wie gehabt plus:

$$\varepsilon \leqq \frac{1}{2}\bar{c}_e \text{ für } e_{vw} \text{ mit } v,w\in B(T)$$

Bei einer Gleichheitskante sind beide Endknoten $\in B(T)$
Also Schrumpfe Blüte:



Komponente wird zu c zusammengefasst:



Für alle vw mit $v \in V(C)$ und $w \notin V(C)$: $c'_{vw} \leftarrow c_{vw} - y_v$
Dies erhält die reduzierten Kosten auf allen übrig gebliebenen Kanten. $(G,c) \rightarrow (G',c')$ ist „abgeleitetes Paar“.

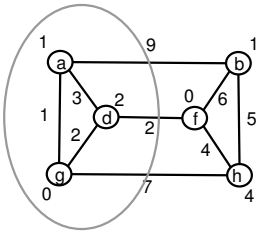
Benutze vw zum Schrumpfen und adaptiere M' und T (Neu)
Eingaben:

- Matching M' von G'
- M'-alternierenden Baum T
- $vw \in E(G')$ mit $v,w \in B(T)$

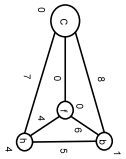
Sei C der Kreis gebildet aus vw und dem (v,w)-Weg in T

$G' \leftarrow G' \times C$;
$M' \leftarrow M' \setminus E(C)$;
forall $(st \in E(G')$ mit $s \in V(C), t \notin V(C))$ $c'_{st} \leftarrow c'_{st} - y_s$;
$y_C \leftarrow 0$;

Beispiel:



ergibt:



wobei ein perfektes Matching in der unteren Zeichnung einem perfekten Matching in der oberen Zeichnung entspricht.

Lemma 5.14 $(G,c) \to (G',c')$ *durch Schrumpfen eines ungeraden Kreises C aus Gleichheitskanten bzgl. dual zulässiger Lösung.*

M'-perfektes Matching in G':
(y',Y') zulässig für (DLPPMMG) für (G',c')
$(M',(y',Y'))$ erfülle (KS und $y'_c \geqq 0$)

M ist perf. Matching in G durch Erweiterung mit Kanten aus $E(C)$
Definiere (y,Y):

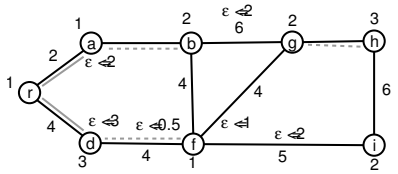
- Schon definiert für $v \notin V(C)$
- $y_v = y'_v$ für $v \in V \setminus V(C)$
- $Y_D = \left\{ \begin{array}{ll} Y'_D & \text{falls } Y'_D > 0 \\ y'_C & \text{falls } D = \delta_{G'}(C) \\ 0 & \text{sonst} \end{array} \right\}$ für $D \in \mathscr{C}$

dann ist (y,Y) zulässig für (DLPPMMG) für (G,c) und $(M,(y,Y))$ erfüllen die KS-Bedingungen.
q.e.d.

Dualvariablenänderung

Wie im bipartiten Fall:
plus $\varepsilon \leqq \frac{1}{2}c_e$ für Kanten mit beiden Endknoten in $B(T)$
plus $\varepsilon \leqq y_v$ für Pseudoknoten in $A(T)$

Beispiel:



(bei Dieser Zeichnung bin ich mir nicht sicher ob ich alles abgemalt habe...)

bf wird Gleichheitskante, danach Blütenschrumpfen.

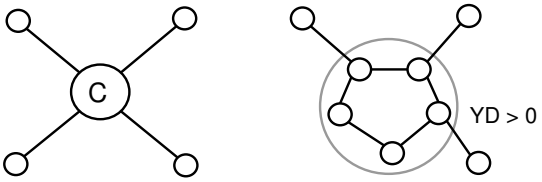
Ändere y
Eingaben:

- abgeleitetes Paar (G',c')

- zulässige Lösung y für (DLPPMMK) für (G', c')
- Matching M' von G' aus Gleichheitskanten

$\varepsilon_1 \leftarrow \min\{\bar{c}_e | e = vw \in E(G'),\ v \in B(T),\ w \notin V(T)\}$;
$\varepsilon_2 \leftarrow \min\{\frac{\bar{c}_e}{2} | e = vw \in E(G'),\ v \in B(T),\ w \in B(T)\}$;
$\varepsilon_3 \leftarrow \min\{y_v | v \in A(T),\ v$ Pseudoknoten in $G'\}$;
$\varepsilon \leftarrow \min\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$;
forall $(v \in B(T))\ y_v \leftarrow y_v + \varepsilon$;
forall $(v \in A(T))\ y_v \leftarrow y_v - \varepsilon$;

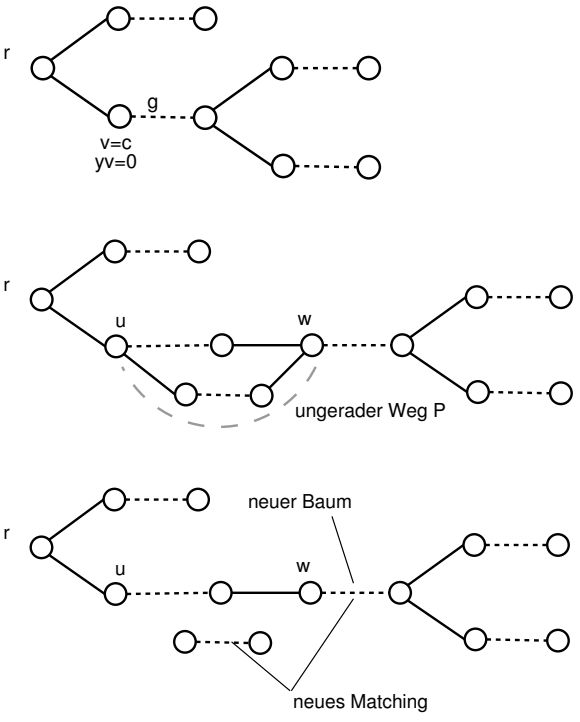
Expandiere ungerade Pseudoknoten



Dies ist nicht von der Form $D = \delta_{G'}(v)$, also nur „erlaubt wenn $y_c = 0$. Wir müssen expandieren wenn ε_3 „greift'". Dann entsteht keine neue Gleichheitskante zum Schrumpfen, Augmentieren, oder Erweitern.

Wir sind dabei, einen Baum T aufzubauen, deshalb außer adaptieren von M' und c' auch Adaptieren von T.

Beispiel:



Expandiere Pseudoknoten v und adaptiere M', T, c'
Eingaben:

- Matching M' aus Gleichheitskanten in abgeleitetem Graphen G'
- M'-alternierender Baum T aus Gleichheitskanten
- Pseudoknoten v in G' mit $y_v = 0$

Ersetze G' und T durch Expandierung von C;
(Neuer Baum T hat Kantenmenge $E(T) \setminus E(P)$.)
Ersetze M' durch expandiertes Matching;
forall ($st \in E(G')$ mit $s \in V(C)$, $t \notin V(C)$) $c'_{st} \leftarrow c'_{st} + y_s$;

Lemma 5.15 *Nach Anwendung des Expansions Unterprogramms ist M' ein Matching in $E_=$ und T ein M'-alternierender Baum aus Gleichheitskanten. q.e.d.*

Blüten-Schrumpf Algorithmus für perfektes Matching mit minimum Kosten
Eingabe: Graph G:

Sei y zulässig für (DLPPMMG), M' ein Matching in $G_=$, $G' = G$;
if (M' ist perfekt) **then**
  STOP „Matching M' perfekt mit minimum Kosten“;
end if
$T \leftarrow (\{r\}, \varnothing)$ wobei r M'-exponiert in G' ist;
loop
  case ($\exists\, vw \in E_=(G')$, $v \in B(T)$, $w \notin V(T)$ M'-exponiert)
    Benutze vw zur M'-Augmentierung;
    if ($\nexists$ M'-exponierter Knoten in G') **then**
      Erweitere M' zu perfektem Matching M in G;
      STOP „Matching M perfekt mit minimum Kosten“;
    else
      $T \leftarrow (\{r\}, \varnothing)$ wobei r M'-exponiert ist;
    end if
  case ($\exists vw \in E_=(G')$ mit $v \in B(T)$, $w \notin V(T)$ M' überdeckt)
    Benutze vw zur Baumerweiterung;
  case ($\exists vw \in E_=(G')$ mit $v \in B(T)$, $w \in B(T)$)
    Benutze vw zum Schrumpfen und adaptiere M', T, c';
  case ($\exists$ Pseudoknoten $v \in A(T)$ mit $y_v = 0$)
    Expandiere v und adaptiere M', T,c';
  case (nichts von Obigem)
    if ($\forall vw \in E$ mit $v \in B(T)$ gilt $w \in A(T)$) **then**

STOP „G hat kein perfektes Matching“;
else
Ändere y;
end if
end loop

Satz 5.16 *Der obige Algorithmus terminiert nach $O(n)$ Augmentierungen und $O(n^2)$ Baumerweiterungsschritten, $O(n^2)$ Expandierungsschritten und $O(n^2)$ Dualen Änderungsschritten.*

Er berechnet ein perf. Matching mit minimum Gewicht oder stellt korrekt fest, dass G kein perfektes Matching hat.

Beweis:
$O(n)$ Augmentierungsschritte. klar.

Wir zeigen $O(n)$ andere Schritte beim Aufbauen eines einzelnen Baums T:
Nach jeder dualen Änderung folgt einer der anderen Schritte. Diese Zählen wir:
Schrumpfen erzeugt einen neuen geraden „B“-Knoten in T. Expandiert werden ungerade Pseudoknoten („A-knoten“).

Wir betrachten:

$$\begin{aligned} S_A &= \sum_{v \in A(T),\ v\text{Pseudokn.}} |S(v)| \\ S_B &= \sum_{v \in B(T),\ v\text{Pseudokn.}} |S(v)| \\ S_0 &= \sum_{v \notin V(T),\ v\text{Pseudokn.}} |S(v)| \\ S = S_a - S_0 & \end{aligned}$$

Expansion vermindert S
Schrumpfen und Baumerw. erhöhen S nicht
$\Rightarrow O(n)$ Expandierungen

Schrumpfen erhöht S_B. Die anderen Schritte erniedrigen S_B nicht.
$\Rightarrow O(n)$ Schrumpfungen

Baumerweiterung erhöht $|E(T)|$ um 2. Erniedrigung von $|E(T)|$ durch Schrumpfen ist insgesamt höchstens n. Expandierung erniedrigt $|E(T)|$ nicht.
$\Rightarrow O(n)$ Baumerweiterungen.

Terminierung „G hat kein perfektes Matching“: Korrektheit, Beweis wie im ungerichteten Fall.

Terminierung mit perfektem Matching. Korrekt nach Lemma 5.14 q.e.d.

Im Beweis von Satz 5.13 (Edmonds) fehlt jetzt nur noch:
$\nexists$ perf. Matching $\Leftrightarrow$ (LPPMMG) hat keine zul. Lösung.
„$\Leftarrow$“ klar
„$\Rightarrow$“ man zeigt: $\nexists$ perfektes Matching $\Rightarrow$ DLPPMMG ist unbeschränkt. q.e.d.

Satz 5.17 *(Gabow [1990])*
Es gibt eine Implementierung des Blütenschrumpf Algorithmus mit Laufzeit $O(nm+n^2\log n)$.

Optimale Duallösungen von Matchingproblemen

Satz 5.18 *Hat (DLPPMMG) eine Optimallösung, so auch eine verschachtelte, d.h. für* $s_1, s_2 \in \mathscr{S} = \{s \subseteq V | Y_{\delta(S)} > 0\}$ *gilt entweder* $S_1 \cap S_2 = \varnothing$ *oder* $S_1 \subseteq S_2$ *oder* $S_2 \subseteq S_1$.

Beweis: $y_v > 0$ im Blütenschrumpf Algorithmus nur für Pseudoknoten in Abgeleiteten Graphen.

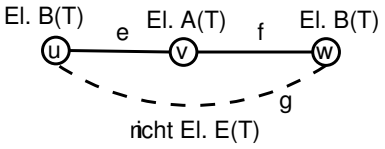
Dreiecksungleichung für die Gewichte

$$c_{uv} + c_{vw} \geqq c_{uw} \quad \forall\, u,v,w \in V$$

Satz 5.19 *Sei* $K_u = (V,E)$ *ein vollständiger Graph auf n Knoten, wobei n gerade. Sei* $c \in \mathbb{R}^E$, $c \geqq 0$ *und c erfülle die Dreiecksungleichung.*

Startet der Blütenschrumpfalgorithmus mit $(y,Y) = 0$, *so gilt für die duale Optimallösung bei Terminierung* $y \geqq 0$.

Beweis: Wir zeigen, dass y während der Ausführung des Blütenschrumpf Algorithmus nicht-negativ bleibt.
$y_v \leftarrow y_v - \varepsilon \Rightarrow v$ ist ungerader „A“-Knoten in T.



Erweiterte Duallösung zu einer zulässigen Duallösung (y,Y) im Originalgraphen. Sei:

$$\begin{aligned} \mathscr{C}_e &= \{D \in \mathscr{C} | Y_D > 0 \text{ und } e \in D\} \\ \mathscr{C}_f &= \{D \in \mathscr{C} | Y_D > 0 \text{ und } f \in D\} \\ \mathscr{C}_g &= \{D \in \mathscr{C} | Y_D > 0 \text{ und } g \in D\} \end{aligned}$$

$\Rightarrow \mathscr{C}_g = \mathscr{C}_e \dot{\cup} \mathscr{C}_f$ wobei $\dot{\cup}$ die disjunkte Vereinigung ist.

ε_0 gilt $\varepsilon \leqq \frac{1}{2}\bar{c}_g$, also:

$$\begin{aligned}
2\varepsilon &\leqq && \bar{c}_g \\
&= && c_{uw} - y_u - y_w - Y(\mathscr{C}_g) \\
&\leqq && c_{uv} + c_{vw} - y_u - y_w - Y(\mathscr{C}_e) - Y(\mathscr{C}_f) \\
&= && (c_{uv} - y_u - Y(\mathscr{C}_e)) + (c_{vw} - y_w - Y(\mathscr{C}_f)) \\
&= && \bar{c}_e + y_v + \bar{c}_f + y_v \\
&\overset{\bar{c}_e=\bar{c}_f=0}{=} && 2y_v \Rightarrow \varepsilon \leqq y_v \quad y_v \text{ bleibt nicht-negativ}
\end{aligned}$$

Prima, jetzt können wir uns Zertifikate für die minimalen Matchings (Zeichnungen mit den Kreisscheiben und Gräben, Kapitel 1) ausstellen, indem wir die dualen Bedingungen dafür verwenden.

Mit dem Algorithmus rechnen wir nun die Radien der Kreisscheiben aus. Ungerade Schnitte entsprechen ungeraden Knotenmengen. Gräben entstehen um ungerade Knotenmengen. y und Y im Blütenschrumpfalgorithmus entsprechen also genau den Kreisscheiben (y) bzw. Gräben (Y). Dies gilt aber nur für euklidische Distanzen, bei der Max-Metrik wären es z.B. Quadrate.