- $\forall (i,j) \in E : c_{ij}^{\pi} = 0$

enthält E^* die Kante (i,j) mit $l_{ij}^* = 0$ und $u_{ij}^* = u_{ij}$.

Übung:
Führen Sie die Transformation auf das maximale Flussproblem zuende. Entwerfen Sie insbesondere einen Algorithmus, der, gegeben ein Flussnetzwerk und eine optimale duale Lösung, eine optimale primale Lösung ausgibt.

3.3 Algorithmen zur Lösung des minimalen Kosten Netzwerkflussproblems

3.3.1 Nichtpolynomielle Algorithmen

3.3.1.1 Negative Kreisalgorithmen

Idee:

Ausgehend von einer zulässigen primalen Lösung wird sukzessive, unter Beibehaltung der primalen Zulässigkeit, diese verbessert. Hierzu werden im Restnetzwerk G_x gerichtete Kreise negativer Kosten identifiziert und auf diesen Flüsse augmentiert. Wenn keine gerichteten Kreise negativer Kosten mehr im Restnetzwerk existieren, dann terminiert das Verfahren. Aus Satz 3.5 folgt dann, dass die gefundene Lösung optimal ist.

Algorithmus NEGATIVE KREISE

Eingabe: Flussnetzwerk $G = (V, E, u, c, b)$.

Ausgabe: zulässiger Fluss minimaler Kosten

Methode:

(1) Berechne einen zulässigen Fluss x in G.

(2) while G_x enthält Kreis negativer Kosten
do
Identifiziere solchen Kreis W;
$\delta := \min \{ r_{ij} \mid (i,j) \in W \}$;
Augmentiere δ Flusseinheiten auf W
und ändere x und G_x entsprechend
od;

(3) Gib x aus.

Laufzeitanalyse:

- Schritt (1)
(mit Hilfe des maximalen Flussproblems) $O(n^3)$
- Mittels des Bellman-Ford-Algorithmus kann ein Kreis negativer Kosten in $O(n \cdot m)$ Zeit identifiziert werden.

Falls die Kapazitäten und Kosten der Kanten alle ganzzahlig sind, dann reduziert jede Augmentierung die Flusskosten um mindestens eine Einheit. Da mCU eine obere Schranke für die anfängliche Flusskosten und 0 eine untere Schranke für die optimalen Flusskosten sind, ist mCU eine obere Schranke für die

Anzahl der Durchläufe der while-Schleife.

Also hat der Algorithmus eine Gesamtlaufzeit von $O(n \cdot m^2 C U)$.

3.3.1.2 Sukzessive-Kürzeste-Wege-Algorithmen

Idee:

Wir starten mit einer primalen Lösung x, die zwar die Kapazitätsbedingungen erfüllt, jedoch Flussbalancierungsbedingungen verletzt und mit Knotenpotentiale, die bzgl. x das reduzierte Kosten-Optimalitätskriterium erfüllen. Unter Beibehaltung des reduzierten Kosten-Optimalitätskriteriums wird sukzessive die primale Zulässigkeit herbeigeführt. Satz 3.6 impliziert, dass die resultierende Lösung optimal ist.

Bezeichnungen:

Sei $G = (V, E, u, c, b)$ ein Flussnetzwerk. $x: E \to \mathbb{R}^+$ heißt Pseudofluss, falls die Flussgrenzenbedingungen erfüllt sind. Bezüglich des Pseudoflusses x definieren wir die Unbalance $e(i)$ eines Knotens $i \in V$ durch:

$$e(i) := b(i) + \sum_{j:(j,i)\in E} x_{ji} - \sum_{j:(i,j)\in E} x_{ij}.$$

$e(i)$ heißt

$\begin{cases} \text{\underline{Überfluss} des Knotens } i & \text{falls } e(i) > 0 \\ \text{\underline{Defizit} des Knotens } i & \text{falls } e(i) < 0 \end{cases}$

Ein Knoten $i \in V$ mit $e(i) = 0$ heißt balanciert.
Seien

$$S := \{ i \in V \mid e(i) > 0 \} \text{ und } T := \{ i \in V \mid e(i) < 0 \}.$$

Das Restnetzwerk G_x bzgl. eines Pseudoflusses x ist genauso definiert, wie das Restnetzwerk bzgl. eines Flusses.

Beobachtung:
Seien π eine Menge von Knotenpotentiale und P ein beliebiger gerichteter Pfad von Knoten k zum Knoten l in G_x. Dann gilt

$$\sum_{(i,j) \in P} c_{ij}^{\pi} = \sum_{(i,j) \in P} c_{ij} + \pi(l) - \pi(k).$$

Also ändern die Knotenpotentiale alle Pfadlängen zwischen einem Paar l, k von Knoten nur um eine konstante Größe.

$\Rightarrow$

Die kürzesten Pfade bzgl. c und bzgl. c^{π} sind dieselben.

Lemma 3.3
Sei $G = (V, E, u, c, b)$ ein Flussnetzwerk, π eine Menge von Knotenpotentialen und x ein Pseudofluss, der bzgl. π das reduzierte Kosten-

Optimalitätskriterium erfüllt. Sei x' ein Pseudofluss, den man aus x durch Senden von Fluss entlang eines kürzesten Pfades von einem Knoten k zu einem Knoten l in G_x erhält. Dann gibt es eine Menge π' von Knotenpotentiale, bzgl. denen x' das reduzierte Kosten-Optimalitätskriterium erfüllt.

Beweis:

Voraussetzung $\Rightarrow$ $c_{ij}^{\pi} \geq 0 \quad \forall (i,j) \in E_x$.

$\forall j \in V$ sei $d(j)$ die kürzeste Distanz des Knotens j vom Knoten k in G_x bzgl. den Kantenlängen c^{π}.

Beh.:

x erfüllt das reduzierte Kosten-Optimalitätskriterium auch bzgl. den Knotenpotentialen $\pi' := \pi - d$.

Bew. d. Beh.:

$\forall (i,j) \in E_x$ gilt:

$$d(j) \leq d(i) + c_{ij}^{\pi}$$

$$\Rightarrow d(j) \leq d(i) + c_{ij} - \pi(i) + \pi(j)$$

Es gilt:

$$c_{ij}^{\pi'} = c_{ij} - \pi'(i) + \pi'(j)$$

$$= c_{ij} - (\pi(i) - d(i)) + (\pi(j) - d(j))$$

$$= \underbrace{c_{ij} - \pi(i) + \pi(j) + d(i)}_{\geq d(j)} - d(j)$$

$$\geq 0$$

Also erfüllt x auch das reduzierte Kosten-Optimalitätskriterium bzgl. π'!

$\square$

Sei P ein kürzester Pfad vom Knoten k zum Knoten l in G_x. Für alle Kanten $(i,j) \in P$ gilt:

$$d(j) = d(i) + c_{ij}^{\pi}$$

Also gilt $\forall (i,j) \in P$:

$$c_{ij}^{\pi'} = c_{ij} - \pi'(i) + \pi'(j)$$

$$= \underbrace{c_{ij} - \pi(i) + \pi(j) + d(i)}_{= d(j)} - d(j)$$

$$= 0.$$

Also erhält die Vergrößerung des Flusses über eine beliebige Kante (i,j) auf P das reduzierte Kosten-Optimalitätskriterium aufrecht.

Beachte, dass falls (j,i) dem Restnetzwerk $G_{x'}$ hinzugefügt wird, dann gilt wegen $c_{ij}^{\pi'} = 0$

$$c_{ji}^{\pi'} = -c_{ij}^{\pi'} = 0.$$

$\blacksquare$

Algorithmus SKW

Eingabe: Flussnetzwerk $G = (V, E, u, c, b)$

Ausgabe: zulässiger Fluss minimaler Kosten

Methode:

(1) $x := 0$; $\pi := 0$; (* erfüllt r.-k.-Ok., da $c_{ij} \geq 0 \ \forall (i,j) \in E$ *)

(2) $e(i) := b(i) \ \forall \ i \in V$;

$S := \{ i \in V \mid e(i) > 0 \}$;

$T := \{ i \in V \mid e(i) < 0 \}$;

(3) while $S \neq \emptyset$

do

(3a) Wähle $k \in S$ und $l \in T$;

(3b) $\forall j \in V$ berechne kürzeste Distanz $d(j)$ von k nach j bzgl. c^π in G_x;

(3c) Berechne kürzesten Pfad P von k nach l;

(3d) $\pi := \pi - d$;

(3e) $\delta := \min \{ e(k); -e(l), \min \{ r_{ij} \mid (i,j) \in P$

(3f) Augmentiere δ Einheiten Fluss entlang P;

(3g) Update x, S und T

od;

(4) Gib x aus.

Laufzeitanalyse:

(1) und (2) $O(n+m)$

Sukzessive Augmentierung kürzester Wege:

- Zusammenhangsannahme $\Rightarrow$
  G_x enthält immer einen gerichteten Pfad vom Knoten k zum Knoten l.

- Jede Iteration löst ein kürzesten-Wege-Problem mit nichtnegativen Kantenlängen und reduziert den Überfluss eines Knotens um mindestens eine Einheit.

$\Rightarrow$ Anzahl der Iterationen $\leq n \cdot U$.

Verwendung von Dijkstras Algorithmus
$\Rightarrow$
Gesamtzeit $O(n \cdot U \cdot S(n,m,C))$,
wobei $S(n,m,C)$ die von Dijkstras Algorithmus benötigte Zeit ist.

Bemerkung:
Es gilt:
$$S(n,m,C) = O(m + n\log n)$$
und auch
$$S(n,m,C) = O(\min\{m \log\log C,\ m + n\sqrt{\log C}\})$$
(siehe AMO, Network Flows)

Weitere Algorithmen:

Siehe AMO, Network Flows.

3.3.2 Polynomzeitalgorithmen

3.3.2.1 Der Right-hand-side Scaling Algorithmus (RHS-Scaling)

Idee:

Sorge dafür, dass beim sukzessiven-kürzesten Wege-Algorithmus in jedem Schritt ein genügend großer Fluss augmentiert wird, so dass sich die Anzahl der Iterationen erheblich reduziert.

Bemerkung:

Ohne Flussoberschranken an den Kanten wäre die Situation einfacher. Wir erreichen dies durch folgende einfache Transformation:

Annahme: $(i,j) \in E$ mit $u_{ij} < \infty$.

$\Rightarrow$

zulässiger Fluss x erfüllt stets $x_{ij} \leq u_{ij}$.

Die Einführung einer Schlupfvariablen $s_{ij} \geq 0$ führt zu

$$x_{ij} + s_{ij} = u_{ij}$$

$$\Leftrightarrow \; -x_{ij} - s_{ij} = -u_{ij} \qquad (*)$$

Nach Hinzufügen eines zusätzlichen Knotens k auf der Kante (i,j) mit Gleichung $(*)$ als Flussbalancierungsbedingung erhalten wir

Netzwerktransformation:

$$
\begin{tikzcd}
\overset{b(i)}{\textcircled{i}} \arrow[rr, "{(c_{ij},\, u_{ij})}", "x_{ij}"'] & & \overset{b(j)}{\textcircled{j}}
\end{tikzcd}
$$

$\Longleftrightarrow$

$$
\begin{tikzcd}
\overset{b(i)}{\textcircled{i}} \arrow[rr, "{(c_{ij},\, \infty)}", "x'_{ik} = x_{ij}"'] & & \overset{b(k) = -u_{ij}}{\textcircled{k}} & & \overset{b(j) + u_{ij}}{\textcircled{j}} \arrow[ll, "{(0,\, \infty)}"', "x'_{jk} = s_{ij}"]
\end{tikzcd}
$$

Übung:
Beweisen Sie die Korrektheit der obigen Netzwerktransformation.

Im Folgenden nehmen wir an, dass die Kanten des zugrundeliegenden Netzwerkes nicht mit Kapazitäten behaftet sind.

Die Basisidee des RHS-Skalierungsalgorithmus ist die folgende:

Seien x ein Pseudofluss und π ein Vektor von Knotenpotentiale. Für $\Delta \in \mathbb{Z}$ sei

$$S(\Delta) := \{\, i \in V \mid e(i) \geq \Delta \,\} \quad \text{und}$$

$$T(\Delta) := \{\, i \in V \mid e(i) \leq -\Delta \,\}.$$

Ein Paar (x, π) heißt $\underline{\Delta\text{-optimal}}$, falls (x, π) die komplementärer Schlupfbedingungen erfüllt und $S(\Delta) = \emptyset$ oder $T(\Delta) = \emptyset$.

Erinnerung:
Die komplementärer Schlupfbedingungen des minimalen Kosten Netzwerkflussproblems ohne Kapa=

zitäten sind:

$$\text{i)}\ c^{\pi}_{ij} > 0 \implies x_{ij} = 0 \qquad \forall (i,j) \in E.$$
$$\text{ii)}\ x_{ij} > 0 \implies c^{\pi}_{ij} = 0$$

Bemerkung:

- Falls $x = 0$ und $\pi = 0$, dann ist (x, π) Δ-optimal $\forall \Delta > U$. Falls x ganzzahlig und (x, π) Δ-optimal für ein $\Delta < 1$, dann ist x ein optimaler Fluss

Der RHS-Skalierungsalgorithmus startet mit einem Δ-optimalen Pseudofluss für $\Delta := 2^{\lceil \log U \rceil}$. In jeder Iteration wird Δ halbiert und dann für das neue Δ ein Δ-optimaler Pseudofluss berechnet. Das Verfahren terminiert mit einem optimalen Fluss x sobald $\Delta < 1$.

Gegeben einen 2Δ-optimalen Pseudofluss erhält der Skalierungsalgorithmus einen Δ-optimalen Pseudofluss durch Lösen von $\leq n$ kürzesten-Weg-Probleme nebst Augmentieren von Δ Einheiten Fluss entlang eines kürzesten Weges von einem Knoten in $S(\Delta)$ zu einem Knoten in $T(\Delta)$. Der Faktor Δ heißt <u>Skalierungsfaktor</u>. Der maximale Zeitraum, in dem der Skalierungsfaktor den Wert Δ hat, heißt <u>Δ-Skalierungsphase.</u> Offensichtlich gibt es $\lceil \log U \rceil + 1$ Skalierungsphasen.

Die formale Beschreibung des RHS-Skalierungsalgorithmus sieht folgendermaßen aus:

Algorithmus RHS - SCALING

Eingabe: Flussnetzwerk $G = (V, E, u, c, b)$

Ausgabe: zulässiger Fluss minimaler Kosten

Methode:

(1) $x := 0$; $\pi := 0$; $e := b$;

(2) $\Delta := 2^{\lceil \log U \rceil}$;

(3) **while** $\exists i \in V : e(i) \neq 0$

  do

   $S(\Delta) := \{ i \in V \mid e(i) \geq \Delta \}$;

   $T(\Delta) := \{ i \in V \mid e(i) \leq -\Delta \}$;

(4)   **while** $S(\Delta) \neq \emptyset$ and $T(\Delta) \neq \emptyset$

    do

(4a)   Wähle $k \in S(\Delta)$ und $\ell \in T(\Delta)$;

(4b)   Berechne $\forall j \in V$ kürzeste Distanz von k zu j in G_x bzgl c^π.

(4c)   Berechne kürzesten Pfad P von k nach ℓ

(4d) $\pi(i) := \pi(i) - d(i) \ \forall i \in V$;

(4e)   Augmentiere Δ Einheiten Fluss entlang P;

(4f)   Update x, $S(\Delta)$ und $T(\Delta)$

    od;

   $\Delta := \Delta / 2$

  od;

(5) Gib x aus.

Es ist nicht a priori klar, dass Δ Einheiten Fluss entlang P augmentiert werden können. Dies folgt direkt aus folgendem Lemma.

Lemma 3.4

Die Restkapazitäten der Kanten im Restnetzwerk $G_x = (V, E_x, r, c')$ sind stets ∞ oder ein ganzzahliges Vielfaches von Δ.

Beweis: (Induktion über # der Augmentierungen)

Induktionsanfang:

Zu Beginn sind alle Restkapazitäten ∞. ✓

Induktionsannahme:

Nach t Augmentierungen sind die Restkapazitäten der Kanten im Restnetzwerk ∞ oder Integervielfache von Δ.

$t \rightsquigarrow t+1$:

Eine Augmentierung ändert die Restkapazität einer Kante um 0 oder um Δ Einheiten. Also impliziert die Induktionsannahme, dass auch nach der Augmentierung die Restkapazitäten der Kanten im Restnetzwerk ∞ oder Integervielfache von Δ sind. Die Halbierung des Skalierungsfaktors verdoppelt den ganzzahligen Faktor vor diesem. ∎

Bemerkung:

Für das Lemma 3.4 benötigen wir, dass die Kanten des Flussnetzwerkes nicht mit Kapazitäten behaftet sind.

Satz 3.9

Der RHS-Skalierungsalgorithmus berechnet einen zulässigen Fluss minimaler Kosten in einem Flussnetzwerk $G = (V, E, c, b)$ ohne Kapazitäten in $O(n \cdot \log U \cdot S(n, m, C))$ Zeit.

Beweis

Korrektheit:

Der RHS-Skalierungsalgorithmus ist ein Spezialfall des sukzessiven kürzesten Weg-Algorithmus.

$\Rightarrow$

RHS-Skalierungsalgorithmus terminiert mit einem zulässigen Fluss minimaler Kosten.

Laufzeit:

Beh.:

Pro Skalierungsphase werden maximal n Augmentierungen durchgeführt.

Bew. d. Beh.:

Zu Beginn der Δ-Skalierungsphase gilt:

$$S(2\Delta) = \emptyset \quad \text{oder} \quad T(2\Delta) = \emptyset.$$

Wir diskutieren beide Fälle nacheinander.

a) $\underline{S(2\Delta) = \emptyset}$:

Zu Beginn der Δ-Skalierungsphase gilt: $|S(\Delta)| \leq n$.

$\forall\, i \in S(\Delta)$ gilt: $\Delta \leq e(i) < 2\Delta$.

Jede Augmentierung schickt Δ Flusseinheiten von einem Knoten $k \in S(\Delta)$ zu einem Knoten $l \in T(\Delta)$

$\Rightarrow$

Jede Augmentierung vermindert $|S(\Delta)|$ um eins.

$\Rightarrow$

Die Anzahl der Augmentierungen ist durch $|S(\Delta)| \leq n$ beschränkt.

b) $\underline{T(2\Delta) = \emptyset}$:

analog

$\square$

Es werden maximal $1 + \lceil \log U \rceil$ Skalierungsphasen benötigt.

$\Rightarrow$

Gesamtzeit ist $O(n \cdot \log U \cdot S(n,m,C))$.

$\blacksquare$

Bemerkung:

- Da der RHS-Skalierungsalgorithmus ein Flussnetzwerk ohne Kapazitäten benötigt, muss im allgemeinen Fall das gegebene Flussnetzwerk zunächst transformiert werden. Das transformierte

Netzwerk enthält $\leq n+m$ Knoten.

$\Rightarrow$

Pro Skalierungsphase werden maximal $n+m$ Augmentierungen durchgeführt.

$\Rightarrow$

Gesamtlaufzeit ist $O(m \cdot \log U\, S(n+m, m, C))$.

Eine genauere Betrachtung reduziert diese Laufzeit auf
$$O(m \log U\, S(n, m, C)).$$
(Siehe AMO, Network flows, S. 360 ff)

- Die RHS-Skalierungsmethode ist ein Kapazitäten-skalierungsalgorithmus. Man kann auch die Kosten skalieren oder beides miteinander kombinieren. (siehe AMO, Network flows, S. 362 ff.)

3.3.2.2 Ein stark polynomieller Algorithmus

Ziel:

Die RHS-Skalierungsmethode derart zu modifizieren, dass die Gesamtlaufzeit stark polynomiell wird.

Schlüsselidee (Eva Tardos 1984).

Δ-Phase:

Identifiziere alle Kanten, deren Fluss in der Δ-Phase derart groß ist, dass diese in allen nachfolgenden Δ-Phasen garantiert positiven Fluss haben.

Beobachtung:

Ein Fluss der Größe $2n\Delta$ reicht hierfür aus.

Bew. d. Beob.:

In einer Skalierungsphase werden $\leq n$ Augmentierungen durchgeführt.

$\Rightarrow$

Auf einer Kante kann der Fluss höchstens um $n \cdot \Delta$ Einheiten reduziert werden.

Die Summation der maximalen Flussreduktion in der Δ-Phase und allen nachfolgenden Phasen ergibt:

$$\sum_{i=0}^{\log \Delta} \frac{n \cdot \Delta}{2^i} = n\Delta \sum_{i=0}^{\log \Delta} 2^{-i} < 2 \cdot n\Delta.$$

$\square$

Eine Kante, deren Fluss in der Δ-Phase den Wert $2n\Delta$ überschreitet, heißt sicher.

Idee:

Kontraktion der sicheren Kanten $e = (i, j)$.

Durchführung:

- Ersetzung der Knoten i und j durch Superknoten p.
- Ersetzung jeder Kante
  - (k, i) oder (k, j) durch die Kante (k, p)
  - (i, k) oder (j, k) durch die Kante (p, k)

Mehrfachkanten werden als verschiedene Kanten interpretiert. Die Kosten einer neuen Kante sind dieselben wie die der ersetzten Kante. Setze

$$b(p) := b(i) + b(j).$$

Gemäß Definition gilt dann

$$e(p) = e(i) + e(j).$$

Folgendes Lemma gibt eine theoretische Rechtfertigung der Kontraktion von sicheren Kanten:

Lemma 3.5

Sei $x_{k\ell} > 0$, $(k,\ell) \in E$ in einer optimalen Lösung des minimale Kosten Netzwerkflussproblems. Dann sind bezüglich jeder Menge von optimalen Knotenpotentialen die reduzierten Kosten der Kante (k,ℓ) gleich null.

Beweis:

Übung

■

Bezeichne P das aktuelle minimale Kosten Netzwerkflussproblem mit Kantenkosten c_{ij}, $(i,j) \in E$.

Annahme:

Während der Lösung von P durch den RHS-Skalierungsalgorithmus wird für eine Kante (k,ℓ) festgestellt, dass diese sicher ist. D.h., $x_{k\ell} \geq 2n\Delta$.

Da der Fluss auf der Kante (k,l) positiv ist, sind die reduzierte Kosten der Kante (k,l) gleich null. D.h.,

$$(*) \qquad c_{kl} - \pi(k) + \pi(l) = 0.$$

Wir betrachten nun dasjenige minimale Kosten Netwerkflussproblem P', das wir aus P erhalten, indem wir jeder Kante $(i,j) \in E$ die Kosten

$$c'_{ij} := c_{ij} - \pi(i) + \pi(j)$$

zuordnen.

Dann folgt aus $(*)$

$$c'_{kl} = 0.$$

<u>Erinnerung</u>:

P und P' haben dieselben optimalen Lösungen.

Da die Kante (k,l) sicher ist, gibt es eine optimale Lösung x für P und somit auch für P', in der $x_{kl} > 0$.

Sei π' eine Menge von optimalen Knotenpotentialen für P'.

$\Rightarrow$

$$c'_{kl} - \pi'(k) + \pi'(l) = 0$$

Da $c'_{kl} = 0$ gilt somit

$$\pi'(k) = \pi'(l).$$

Beobachtung:

Wenn wir zum dualen LP des minimalen Kosten Netzwerkflussproblems die zusätzliche Bedingung, dass die Knotenpotentiale von k und l gleich sein müssen, dann hat dies keine Auswirkungen auf die optimalen Lösungen des linearen Programmes.

Erinnerung:

primales LP:

$$\min \sum_{(i,j)\in E} c_{ij} x_{ij}$$

wobei

$$(1) \quad \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = b(i) \quad \forall i \in V$$

$$(2) \quad x_{ij} \geq 0 \quad \forall (i,j) \in E.$$

duales LP:

$$\max \sum_{i\in V} b(i)\,\pi(i)$$

wobei

$$\pi(i) - \pi(j) \leq c_{ij} \quad \forall (i,j) \in E.$$

Beachte: Kanten sind nicht mit Kapazitäten behaftet.

Frage:

Wie können wir das duale LP unter der zusätzlichen Bedingung, dass zwei Knotenpotentiale $\pi(k)$ und $\pi(l)$ gleich sein müssen, lösen?

Idee:

Ersetze im dualen LP einfach $\pi(k)$ und $\pi(l)$ durch $\pi(p)$.

Auf diese Art und Weise erhalten wir ein lineares Programm mit einem Knotenpotential weniger.

Übung:

Zeigen Sie, dass das resultierende LP das duale Problem zu dem minimale Kosten Netzwerkflussproblem ist, das wir aus P erhalten, indem wir die Knoten k und l zu einem Knoten p schrumpfen.

Folgendes Lemma zeigt, dass es eine sichere Kante nach einer "genügend kleinen" Anzahl von Skalierungsphasen gibt:

Lemma 3.6

Sei $i \in V$ ein Knoten, für den nach Beendigung der Δ-Skalierungsphase $|b(i)| > 5n^2\Delta$ dann gibt es eine zum Knoten i inzidente Kante mit Fluss $\geq 3n\Delta$.

Beweis:

Zunächst beweisen wir, dass $|e(i)| < 2n\Delta$.

Der Pseudofluss, mit dem die Δ-Skalierungsphase startet, ist 2Δ-optimal. D.h.

$$S(2\Delta) = \emptyset \quad \text{oder} \quad T(2\Delta) = \emptyset.$$

$\Rightarrow$

Der Gesamtüberschuss oder das Gesamtdefizit ist $< 2n\Delta$.

$\Rightarrow$

$$|e(i)| < 2n\Delta.$$

Wir unterscheiden zwei Fälle:

<u>1. Fall:</u> $b(i) > 0$.

Der ausgehende Fluss des Knotens i ist

$$\geq b(i) - e(i).$$

Es gibt $\leq n$ ausgehende Kanten von i mit positiven Fluss. (überzeugen Sie sich.)

Mindestens eine dieser Kanten hat Fluss

$$\geq \frac{b(i) - e(i)}{n} \geq \frac{5n^2\Delta - 2n\Delta}{n} \geq 3n\Delta.$$

<u>2. Fall:</u> $b(i) < 0$

analoge Argumentation bezüglich eingehenden Kanten.

∎

<u>Ziel:</u>
Beweis, dass jeder Knoten vor seiner Kontraktion lediglich $O(\log n)$-mal an einer Augmentierung

teilnimmt.

<u>Bezeichnung</u>:
Ein Knoten $i \in V$ heißt <u>regeneriert in der Δ-Skalierungsphase</u>, falls $i \notin S(2\Delta) \cup T(2\Delta)$ am Ende der 2Δ-Skalierungsphase, aber zu Beginn der Δ-Skalierungsphase $i \in S(\Delta) \cup T(\Delta)$.

<u>Bemerkung</u>:
Für jeden in der Δ-Skalierungsphase regenerierten Knoten i gilt: $\Delta \leq |e(i)| < 2\Delta$.

Zu Beginn des Algorithmus gilt $e(i) = b(i)$ $\forall\, i \in V$. Innerhalb $\log(5n^2) = O(\log n)$ Skalierungsphasen hat sich Δ mindestens um den Faktor $\frac{1}{5n^2}$ verkleinert, so dass

$$|b(i)| > 5n^2\Delta.$$

Lemma 3.6 $\Rightarrow$

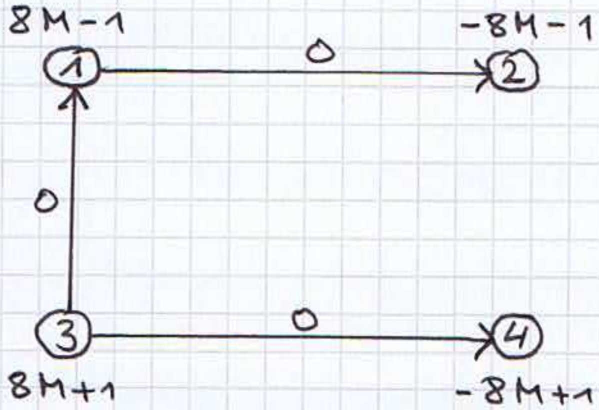
Der Knoten ist inzident zu einer sicheren Kante, die nun kontrahiert wird.

Obige Betrachtung legt nahe, dass der Algorithmus eine Laufzeit von $O(n \cdot \log n \cdot S(n,m,C))$ hat.

<u>Aber</u>:

Es ist möglich, dass ein geschrumpfter Knoten v mit $b(v) \approx 0$ aber $e(v)$ groß regeneriert wird.

<u>Beispiel</u>:



Alle Kantenkosten seien null. Am Knoten i steht e(i), an den Kanten der aktuelle Fluss. Zu Beginn gilt $e(i) = b(i)$.

Es existiert eine eindeutige zulässige Lösung. Diese muss der Algorithmus finden.

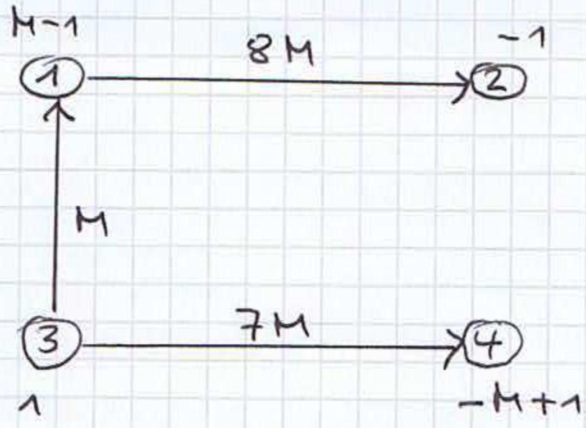
Nehmen wir an, dass der Algorithmus folgende Vorgehensweise hat:

<u>8M - Skalierungsphase:</u>

8M Einheiten Fluss werden vom Knoten 3 zum Knoten 2 geschickt.

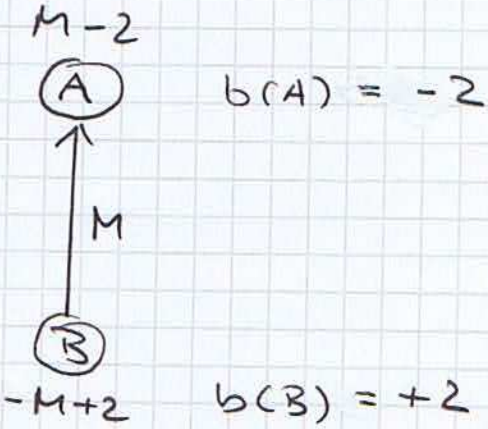
<u>4M-, 2M- und M - Skalierungsphase:</u>

Es wird Fluss vom Knoten 1 zum Knoten 4 geschickt



Die Kanten (1,2) und (3,4) können geschrumpft werden. Die Kante (3,1) kann nicht geschrumpft werden.

$\leadsto$



Obwohl $b(A) = -2$ und $b(B) = +2$, sind noch $\log M$ Skalierungsphasen notwendig, bis der Fluss auf der Kante (B,A) genügend klein ist, so dass auch diese Kante geschrumpft werden kann und der Algorithmus somit terminiert.

Also ist der Algorithmus <u>nicht</u> stark polynomiell!

<u>Frage</u>: Wie beheben wir dieses Problem?

Zur Beantwortung dieser Frage analysieren wir die oben beschriebene Situation genauer:

Es ist möglich einen geschrumpften Knoten zu erzeugen, dessen Überschuss/Defizit mit dem Skalierungsfaktor Δ vergleichbar ist, jedoch nur ein sehr kleines Angebot bzw. Nachfrage besitzt. Demzufolge kann solcher Knoten häufig regenerieren, bevor dieser geschrumpft wird.

<u>Idee</u>:

Wir modifizieren die Augmentierungsregeln etwas wie folgt:

- Sei $1/2 < \alpha < 1$.

  Der Startknoten eines augmentierenden Pfades benötigt Überschuss $\geq \alpha \cdot \Delta$ (anstatt Δ) und der Endknoten des augmentierenden Pfades benötigt ein Defizit $\geq \alpha\Delta$ (anstatt Δ).

Der stark polynomielle Algorithmus ist der RHS-Skalierungsalgorithmus mit den folgenden Modifikationen:

1) Sobald eine Kante $e=(i,j)$ sicher wird, führt der Algorithmus die Kontraktion dieser Kante durch. Da maximal $(n-1)$ Kontraktionen erfolgen und nach $O(\log n)$ Skalierungsphasen solche garantiert durchgeführt wird, ergibt sich unmittelbar die stark polynomielle Laufzeit.

2) Wir erlauben Augmentationen von einem Knoten i mit $e(i) \geq \alpha\Delta$ zu einem Knoten j mit $e(j) \leq -\alpha\Delta$. Falls $\frac{1}{2} < \alpha < 1$, dann ist der Algorithmus stark polynomiell. Falls $\alpha = 1$, dann ist der Algorithmus nicht stark polynomiell.

3) Im Gegensatz zum RHS-Skalierungsalgorithmus, der das primale minimale Kosten Netzwerkfluss=problem löst und mit einem optimalen Fluss terminiert, löst der stark polynomielle Al=gorithmus das duale minimale Kosten Netz=werkflussproblem und erhält einen optima=len Vektor von Knotenpotentiale. Unter Ver=wendung dieses Vektors wird dann ein optima=ler Fluss berechnet.

In der Beschreibung des stark polynomiellen Al=gorithmus bezeichnen V' und E' die Knoten- bzw. Kantenmenge des geschrumpften Netzwerkes.

<u>Algorithmus</u> spRHS-SKALIERUNG

<u>Eingabe</u>: Flussnetzwerk $G = (V, E, c, b)$
<u>Ausgabe</u>: zulässiger Fluss x minimaler Kosten
<u>Methode</u>:

(1) $x := 0$; $\pi := 0$; $e := b$; $V' := V$; $E' := E$;
(2) $\Delta := \max\{\, e(i) \mid i \in V' \}$;

(3) <u>while</u> $\exists\, i \in V'$ mit $e(i) \neq 0$

<u>do</u>

(3a) <u>if</u> $x_{ij} = 0 \;\forall\, (i,j) \in E'$ <u>and</u> $e(i) < \Delta \;\forall i \in V'$

<u>then</u>

$\Delta := \max\{\, e(i) \mid i \in V' \}$

<u>fi</u>;

(3b) <u>while</u> $\exists\, (k,l) \in E'$ mit $x_{kl} \geq 3n\Delta$

<u>do</u>

<u>for</u> alle $(i,j) \in E'$

<u>do</u>

$c_{ij} := c_{ij} - \pi(i) + \pi(j)$

<u>od</u>;

Kontraktion (k, l)

<u>od</u>;

$S(\Delta) := \{\, i \in V' \mid e(i) \geq \alpha\Delta \}$;

$T(\Delta) := \{\, i \in V' \mid e(i) \leq -\alpha\Delta \}$;

<u>while</u> $S(\Delta) \neq \emptyset$ and $T(\Delta) \neq \emptyset$

<u>do</u>

Wähle $k \in S(\Delta)$ und $l \in T(\Delta)$;

Berechne $\forall\, j \in V' \setminus \{k\}$ die kürzeste Distanz $d(j)$ von k nach j in G'_x, wobei die reduzierten Kosten die Kantenlängen sind;

$\pi(i) := \pi(i) - d(i) \quad \forall\, i \in V'$;

Augmentiere Δ Einheiten Fluss entlang eines kürzesten Pfades P von k nach l in G'_x;

update $x, r, e, S(\Delta)$ und $T(\Delta)$

<u>od</u>;

$$\Delta := \frac{\Delta}{2}$$

od;

(4) Expandiere das Netzwerk und berechne einen optimalen Fluss x.

Bemerkung:

- Im aufeinanderfolgenden Skalierungsphasen halbiert sich der Skalierungsfaktor oder erfährt eine noch größere Reduktion.

- Immer wenn der Algorithmus die Kontraktion zweier Knoten durchführt, werden die Kosten der Kanten auf ihre aktuellen reduzierten Kosten gesetzt. Wie wir uns bereits überlegt haben, hat dies keine Auswirkung auf die optimale Lösung des minimale Kosten Netzwerkflussproblems.

Als nächstes werden wir die Korrektheit des Al= gorithmus beweisen und seine Laufzeit analysieren.

Lemma 3.7

In jedem Schritt des Algorithmus spRHS-SKALIERUNG sind die Flüsse und Restkapazitäten der Kanten im Restnetzwerk ∞ oder ein ganzzahliges Vielfaches von Δ.

Beweis:

analog zum Beweis des Lemmas 3.4.
Beachte, dass stets, wenn der Skalierungsfaktor mehr als halbiert wird, alle Kanten Nullfluss haben. Übung

Lemma 3.7 impliziert, dass in jeder Augmentierung eines Pfades P auch Δ Flusseinheiten über P geschickt werden kann.

Der Algorithmus hält stets einen Pseudofluss, der die komplementäre Schlupfbedingungen erfüllt, aufrecht und terminiert, sobald der Pseudofluss ein Fluss ist. D.h., der Algorithmus terminiert mit einem optimalen Fluss im geschrumpften Netzwerk.

Bevor wir die Expansion des geschrumpften Netzwerkes und die Berechnung eines optimalen Flusses im expandierten Netzwerk beschreiben, analysieren wir die Laufzeit, die spRHS-SKALIERUNG zur Berechnung des optimalen Flusses im geschrumpften Netzwerk benötigt.

Zunächst passen wir die Definition der Regenerierung eines Knotens in der Δ-Skalierungsphase an. Seien Δ' und Δ die Skalierungsfaktoren zweier aufeinanderfolgenden Skalierungsphasen. D.h., $\Delta' \geq 2\Delta$. Wir sagen, Knoten k regeneriert in der Δ-Skalierungsphase, falls

$$\alpha\Delta \leq |e(k)| < \alpha\Delta'.$$

Beachte, dass ein Knoten, der durch Kontraktion in der Δ-Skalierungsphase entsteht, nicht in der Δ-Skalierungsphase regeneriert.

Wir werden zunächst zeigen, dass die Gesamtanzahl der Augmentierungen $\leq n$ plus die Anzahl der regenerierten Knoten ist. Dann werden wir zeigen, dass zum ersten Zeitpunkt, wenn ein Knoten k regeneriert $|e(k)| \leq \frac{2 \cdot \alpha \cdot |b(k)|}{2 - 2\alpha}$ erfüllt ist. Zusammen mit Lemma 3.6 impliziert dies für festes α, dass ein Knoten $O(\log n)$-mal regeneriert, bevor er geschrumpft wird. Hieraus folgt eine $O(n \log n)$ obere Schranke für die Gesamtanzahl der Augmentierungen. Schließlich werden wir zeigen, dass die Gesamtanzahl der Skalierungsphasen auch $O(n \cdot \log n)$ ist, was dann die Angabe einer oberen Schranke für die Laufzeit ermöglicht.

<u>Lemma 3.8</u>

Die Gesamtanzahl der Augmentierungen während der Δ-Skalierungsphase ist beschränkt durch die Anzahl der in der Δ-Skalierungsphase regenerierten Knoten plus der Anzahl der in der Δ-Skalierungsphase erfolgten Kontraktionen.

<u>Beweis:</u>

Sei Δ' der Skalierungsfaktor der vorangegangenen Skalierungsphase. Dann gilt $\Delta \leq \frac{\Delta'}{2}$.
Am Ende der Δ'-Skalierungsphase gilt

$$S(\Delta') = \emptyset \quad \text{oder} \quad T(\Delta') = \emptyset .$$

1. Fall: $S(\Delta') = \emptyset$.

Betrachten wir folgende Potentialfunktion:

$$F := \sum_{i \in S(\Delta)} \left\lfloor \frac{e(i)}{\alpha \Delta} \right\rfloor.$$

Jede Augmentierung sendet Δ Flusseinheiten von einem Knoten in $S(\Delta)$.
Das Senden von Δ Flusseinheiten von einem Knoten $k \in S(\Delta)$ zu einem Knoten $l \in T(\Delta)$ kann nicht den Knoten l nach $S(\Delta)$ bringen.
(Beachte: $e(l) \leq -\alpha\Delta$. Wegen $\alpha > \frac{1}{2}$ gilt nach der Augmentierung $e(l) < \alpha \cdot \Delta$).

$\Rightarrow$
F vermindert sich mindestens um eins.
$\Rightarrow$
Gesamtanzahl von Augmentierungen

$$\leq F_a - F_e + Z,$$

wobei F_a der anfängliche Wert von F, F_e der letzte Wert von F und Z der Gesamtzuwachs von F ist.

Da $S(\Delta') = \emptyset$ befinden sich zu Beginn der Δ-Skalierungsphase nur regenerierte Knoten in $S(\Delta)$.
Falls $\Delta = \frac{\Delta'}{2}$, dann gilt $e(i) < 2\alpha\Delta \;\; \forall i \in S(\Delta)$.
Wegen $\left\lfloor \frac{e(i)}{\alpha\Delta} \right\rfloor = 1 \;\; \forall i \in S(\Delta)$ ist dann

$$F_a = |S(\Delta)|.$$

Falls $\Delta < \frac{\Delta'}{2}$, dann gilt $e(i) \leq \Delta \; \forall \, i \in S(\Delta)$.
Wegen $\lfloor \frac{e(i)}{\alpha \Delta} \rfloor \leq \lfloor \frac{1}{\alpha} \rfloor = 1$ gilt dann

$$F_a \leq |S(\Delta)|.$$

Ferner gilt

$$F_e \geq 0.$$

$\forall$ reele Zahlen $e(i)$ und $e(j)$ gilt:

$$\left\lfloor \frac{e(i) + e(j)}{\alpha \Delta} \right\rfloor \leq \left\lfloor \frac{e(i)}{\alpha \Delta} \right\rfloor + \left\lfloor \frac{e(j)}{\alpha \Delta} \right\rfloor + 1$$

$\Rightarrow$

Eine Kontraktion erhöht den Wert von F höchstens um eins.

$\Rightarrow$

$Z \leq$ Anzahl der Kontraktionen.

<u>2 Fall:</u> $T(\Delta') = \emptyset$

analog $\blacksquare$

<u>Lemma 3.9</u>

Falls für $(i,j) \in E'$ in der Δ-Skalierungsphase $x_{ij} \geq 3n\Delta$, dann gilt in allen nachfolgenden Skalierungsphasen $x_{ij} > 0$.

<u>Beweis:</u>

Lemma 3.8 $\Rightarrow$

Die Gesamtanzahl der in einer $\bar{\Delta}$- Skalierungsphase erfolgten Augmentierungen

$\leq n +$ Anzahl der in $\bar{\Delta}$-Skalierungsphase erfolgten Kontraktionen

Die Flussänderung der Kante (i,j) ist pro in der $\bar{\Delta}$-Skalierungsphase erfolgten Augmentierung $\leq \bar{\Delta}$.

Insgesamt gibt es höchstens $n-1$ Kontraktionen

$\Rightarrow$

Flussänderung auf der Kante (i,j) in den nachfolgenden Skalierungsphasen ist

$$\leq (n-1)\Delta + n \cdot \sum_{\ell=0}^{\log \Delta} \frac{\Delta}{2^\ell} < 3n\Delta.$$

■

Lemma 3.10

$\forall k \in V'$ gilt stets $e(k) \equiv b(k) \bmod \Delta$.

Beweis:

Definition $\Rightarrow$

$$e(k) = b(k) + \sum_{j:(j,k)\in E'} x_{jk} - \sum_{j:(k,j)\in E'} x_{kj}.$$

Da alle Kantenflüsse ganzzahlige Vielfache von Δ sind, können wir schreiben

$$e(k) = b(k) + w\Delta \quad \text{für ein } w \in \mathbb{Z}$$

Also gilt: $e(k) \equiv b(k) \bmod \Delta$

■

Lemma 3.11

Wenn ein Knoten k zum ersten Mal regeneriert, dann gilt $|e(k)| \leq \frac{2\alpha\,|b(k)|}{2-2\alpha}$.

Beweis:

Annahme:
Knoten k regeneriert zu Beginn der Δ-Skalierungsphase zum ersten Mal. Sei Δ' der Skalierungsfaktor der vorangegangenen Skalierungsphase.

Falls $\Delta < \frac{\Delta'}{2}$, dann ist jeder Kantenfluss null und $e(k) = b(k)$. Wegen $\alpha > \frac{1}{2}$ gilt $\frac{2\alpha}{2-2\alpha} > 1$. Also gilt $|e(k)| \leq \frac{2\alpha\,|b(k)|}{2-2\alpha}$.

Falls $\Delta = \frac{\Delta'}{2}$, dann impliziert Lemma 3.7, dass alle Kantenflüsse ganzzahlige Vielfache von 2Δ sind. Wir unterscheiden zwei Fälle:

1. Fall: $e(k) > 0$

Da k regeneriert gilt dann

$$\alpha\Delta \leq e(k) < 2\alpha\Delta.$$

Lemma 3.10 $\Rightarrow$ $e(k) = b(k) + w \cdot 2\Delta$ für ein $w \in \mathbb{Z}$

Falls $w \leq 0$, dann gilt $e(k) \leq b(k)$ und somit die Behauptung.

Falls $w \geq 1$ dann gilt

$$e(t) \geq b(t) + 2\Delta$$
$$\Leftrightarrow \quad b(t) \leq e(t) - 2\Delta$$

Wegen $e(t) < 2\alpha\Delta$ folgt hieraus

$$(*) \qquad b(t) < (2\alpha - 2)\Delta.$$

Wegen $\alpha < 1$ folgt hieraus

$$b(t) < 0.$$

$\Rightarrow$ Multiplikation von $(*)$ mit -1 ergibt

$$|b(t)| > (2 - 2\alpha)\Delta.$$

$\Delta > \frac{e(t)}{2\alpha}$ ergibt

$$|b(t)| > \frac{(2-2\alpha)}{2\alpha} e(t)$$
$$\Leftrightarrow \quad e(t) < \frac{2\alpha\,|b(t)|}{(2-2\alpha)}$$

<u>2. Fall</u>: $e(t) < 0$

analog Übung

$\blacksquare$

<u>Lemma 3.12</u>

Jeder Knoten regeneriert $O(\log n)$-mal.

Beweis:

Annahme:
Knoten k regeneriert zum ersten Mal in der Δ^*-Skalierungsphase.

Lemma 3.11 $\Rightarrow$

$$\alpha \Delta^* \leq |e(k)| \leq \frac{2\alpha}{2-2\alpha} |b(k)|$$

Also gilt

$$\Delta^* \leq \frac{2}{2-2\alpha} |b(k)|$$

Da jede Skalierungsphase den aktuellen Skalierungsfaktor zumindest halbiert, gilt nach

$$\lceil \log(\tfrac{2}{2-2\alpha} \cdot 5n^2) \rceil = O(\log n)$$

Skalierungsphasen für den aktuellen Skalierungsfaktor Δ

$$\Delta \leq \frac{\Delta^*}{\frac{2}{2-2\alpha} \cdot 5n^2} \leq \frac{|b(k)|}{5n^2}.$$

Lemma 3.6 $\Rightarrow$

Falls der Knoten k noch existiert, dann gibt es eine zu k inzidente Kante e mit $x_e \geq 3n\Delta$.

$\Rightarrow$

x_e ist sicher und der Knoten k wird geschrumpft, kann also nicht mehr regenerieren.

$\Rightarrow$ k regeneriert $O(\log n)$-mal

Aus den obigen Lemmata folgt direkt folgender Satz:

Satz 3.10

Die Gesamtanzahl der Augmentierungen über alle Skalierungsphasen ist $O(n \cdot \log n)$.

Beweis:

Übung

■

Als nächstes beweisen wir eine obere Schranke für die Anzahl der durchgeführten Skalierungsphasen.

Satz 3.11

Die Anzahl der Skalierungsphasen ist $O(n \cdot \log n)$.

Beweis:

Satz 3.10 $\Rightarrow$

Die Gesamtanzahl der Skalierungsphasen, in denen mindestens eine Augmentierung erfolgt, ist $O(n \cdot \log n)$.

z.z.: Die Anzahl der Skalierungsphasen, in denen keine Augmentierung erfolgt, ist $O(n \log n)$.

Betrachte hierzu eine Δ-Skalierungsphase, in der keine Augmentierung erfolgt. Sei Δ' der Skalierungsfaktor der vorangegangenen Skalierungsphase. Wir unterscheiden zwei Fälle:

<u>1. Fall</u> Δ wird auf $\max\{e(i) \mid i \in V'\}$ gesetzt.

Dann gilt $\Delta < \frac{\Delta'}{2}$. Wegen $\frac{1}{2} < \alpha < 1$ gilt für den Knoten $k \in V'$ mit $e(k) = \Delta$

$$\alpha\Delta \leq |e(k)| < \alpha\Delta'.$$

$\Rightarrow$
Der Knoten k regeneriert in der Δ-Skalierungs=phase.

Lemma 3.12 $\Rightarrow$

Dieser Fall kann nur $O(n \cdot \log n)$-mal auftreten.

<u>2. Fall</u> $\neg$ 1. Fall

Dann gilt $\Delta = \frac{\Delta'}{2}$ und

$\exists\, (i,j) \in E'$ mit $x_{ij} > 0$ oder
$\exists\, k \in V'$ mit $e(k) \geq \Delta$.

Es genügt zu zeigen, dass innerhalb von $O(\log n)$ Skalierungsphasen eine Kontraktion erfolgen muss. Da maximal $n-1$ Kontraktionen möglich sind, folgt hieraus, dass der Fall nur $O(n \cdot \log n)$-mal eintreten kann.

Beweis von Lemma 3.12 $\Rightarrow$

Falls ein Knoten regeneriert, dann nimmt er inner=halb von $O(\log n)$ Skalierungsphasen an einer Kontrak=tion teil. Also genügt es zu zeigen, dass innerhalb von $O(\log n)$ Skalierungsphasen ein Knoten regeneriert.

Solange keine Augmentierung erfolgt, ändert sich auf keiner Kante der Fluss und an keinem Knoten die Unbalance. Bevor eine Augmentierung erfolgen kann, muss mindestens ein Knoten regenerieren

a) Annahme:

$\exists\ (i,j) \in E'$ mit $x_{ij} > 0$

Lemma 3.7 $\Rightarrow \quad x_{ij} \geq \Delta.$

Falls innerhalb den nächsten $\lceil \log 3n \rceil$ Skalierungs=phasen keine Regenerierung eines Knotens und so=mit kein Augmentierung erfolgt, dann gilt danach:

i) Der Skalierungsfaktor Δ^* ist $\leq \frac{\Delta}{3n}$

ii) $x_{ij} \geq \Delta$ ist unverändert. D.h.,

$$x_{ij} \geq 3n\Delta^* \text{ und somit sicher.}$$

$\Rightarrow$ Kontraktion (i,j) wird durchgeführt.

$\Rightarrow$ Unterfall kann nur $O(n \cdot \log n)$-mal eintreten

b) Annahme:

$\exists\ k \in V'$ mit $e(k) \geq \Delta.$

$\Rightarrow S(\Delta) \neq \emptyset$

Da keine Augmentierung durchgeführt wird, gilt $T(\Delta) = \emptyset$.

Betrachte Knoten $l \in V'$ mit $e(l) < 0$ und $|e(l)|$ maximal.

$$e(t) \geq \Delta \quad \Rightarrow \quad |e(e)| \geq \frac{\Delta}{n}.$$

Nach $\lceil \log n \rceil$ Skalierungsphasen gilt für den Skalierungsfaktor Δ^*

$$\Delta^* \leq \frac{\Delta}{n}.$$

$\Rightarrow$ Der Knoten e regeneriert innerhalb von $\log n$ Skalierungsphasen.

Also kann auch dieser Unterfall nur $O(n \cdot \log n)$-mal eintreten.

Insgesamt haben wir bewiesen, dass die Anzahl der Skalierungsphasen $O(n \cdot \log n)$ ist.

■

Satz 3.12

Der Algorithmus SPRHS-SKALIERUNG berechnet in $O((n \cdot \log n)\, S(n, m, C))$ Zeit einen zulässigen Fluss minimaler Kosten im geschrumpften Netzwerk.

Beweis:

Wir haben bereits diskutiert, dass das Lemma 3.7 die Korrektheit des Algorithmus impliziert. Somit verbleibt noch der Beweis der oberen Schranke für die Laufzeit des Algorithmus.

Initialisierung $\qquad O(n+m)$

pro Skalierungsphase:

- Reduktion des Skalierungsfaktors um mehr als $1/2$ $\quad O(n+m)$

- Identifikation der sicheren Kanten nebst Kontraktion dieser $\quad O(m)$

- Identifikation von $S(\Delta)$ und $T(\Delta)$ $\quad O(n)$

$\sum$ über alle Skalierungsphasen $\quad O((n \cdot \log n)(m+n))$

pro Augmentierung:

Lösung des kürzesten Wegeproblems $\quad S(n,m,C)$

Durchführung der Augmentierung $\quad O(n)$

$\sum$ über alle Augmentierungen $\quad O((n \cdot \log n) \cdot S(n,m,C))$

■

Ziel:

Expansion der geschrumpften Knoten nebst Berechnung eines optimalen Flusses im ursprünglichen Netzwerk.

Idee:

- Berechne zunächst eine optimale Menge von Knotenpotentialen (d.h., eine optimale Lösung des dualen Problems).

- Berechne dann mittels Lösen eines maximalen Flussproblems einen optimalen Fluss im ursprünglichen Netzwerk.

Wir erhalten eine optimale Menge von Knoten= potentialen mittels wiederholter Anwendung des folgenden Lemmas:

<u>Lemma 3.13</u>

Seien P ein Problem mit Kantenkosten c_{ij} und P' dasselbe Problem mit Kantenkosten $c_{ij} - \pi(i) + \pi(j) \quad \forall (i,j) \in E$. Falls π' eine optimale Menge von Knotenpotentialen für P' ist, dann ist $\pi + \pi'$ eine optimale Menge von Knotenpotentialen für P.

<u>Beweis</u>:

Zeige dass folgendes gilt:

x erfüllt komplementäre Schlupfbedingung bezüglich Kantenkosten $c_{ij} - \pi(i) + \pi(j)$, $(i,j) \in E'$ und Knotenpotentialen π'

$\Rightarrow$

x erfüllt komplementäre Schlupfbedingung bezüglich Kantenkosten c_{ij}, $(i,j) \in E'$ und Knotenpotentialen $\pi + \pi'$.

Folgere dann hieraus die Behauptung.

Übung

Durchführung:

Die Knoten im geschrumpften Netzwerk werden in der umgekehrten Reihenfolge, in der diese durch den Algorithmus SPRHS-SKALIERUNG kreiert worden sind, wieder expandiert.

Betrachten wir hierzu die Situation, in der die Kante $(k,l) \in E'$ durch den Algorithmus zu einem Superknoten p geschrumpft wird. Dieser verfährt wie folgt:

(1) **for** alle $(i,j) \in E'$
  do
    $c_{ij} := c_{ij} - \pi(i) + \pi(j)$
  od;
(2) Kontraktion(k,l).

Die Expansion des Superknotens p zur Kante (k,l) erfolgt durch Rückgängigmachen der obigen Transformationen in der umgekehrten Reihenfolge.

$\leadsto$

(1) Weise den Knoten k und l das Potential des Superknotens p zu;
(2) Addiere π auf den aktuellen Vektor π' von Knotenpotentialen.

Übung
Zeigen Sie, dass nach Expansion aller Knoten die resultierende Menge von Knotenpotentialen

eine optimale Lösung des dualen Problems ist.

Wie man aus einer optimalen dualen Lösung eine optimale primale Lösung erhält, haben wir uns bereits überlegt.

Übung
Arbeiten Sie den Algorithmus zur Berechnung eines optimalen zulässigen Flusses aus.

4. Fisher-Marktgleichgewichte

Ziel:
Anwendung von Netzwerkflüssen zur Lösung eines Problems aus den Wirtschaftswissenschaften.

Literatur:

N. Nisan, T. Roughgarden, E. Tardos, V. Vazirani (eds.), Algorithmic Game Theory, Cambridge University Press (2007), Ch. 5.

X. Deng, Ch. Papadimitriou, S. Safra, On the complexity of equilibria, 34th STOC (2002), 67-71.
(Ausgearbeitete Arbeit kann von X. Deng's Homepage bezogen werden.)

N. R. Devanur, Ch. Papadimitriou, A. Saberi, V. Vazirani, Market equilibrium via a primal-dual algorithm for a convex program, JACM (2008), 1-18.