Das Minimum Cost Flow Problem

Andrea Jaax

May 20, 2014

Contents

1 Notationen und Annahmen

1.1 Problemdefinition

Gegeben:

- $G = (N, A)$ gerichteter Graph mit Knotenmenge N und Kantenmenge A
- $u : A \to \mathbb{N}_0^+$ Kapazitätsfunktion (beschränkt maximalen Fluss jeder Kante nach oben)
- $c : A \to \mathbb{N}_0^+$ lineare Kostenfunktion (gibt die Kosten für den Transport einer Flusseinheit auf jeder Kante an)

- $b : N \to \mathbb{N}$ Angebot / Nachfrage am Knoten mit

$$> 0 \qquad \text{Angebotsknoten} \tag{1}$$
$$b(i) = 0 \qquad \text{Transportknoten} \tag{2}$$
$$< 0 \qquad \text{Nachfrageknoten.} \tag{3}$$

Sei $x : A \to \mathbb{N}_0^+$ eine Flussfunktion.
Ein **zulässiger Fluss** erfüllt folgende Bedingungen:

1. Massenbalancebedingungen:

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \; \forall i \in N \tag{4}$$

2. Kapazitätsbedingungen:

$$0 \leq x_{ij} \leq u_{ij} \; \forall (i,j) \in A \tag{5}$$

Notwendige Bedingung für einen zulässigen Fluss:

$$\sum_{i \in N} b(i) = 0 \tag{6}$$

Das Min-Cost Flow Problem minimiert die Zielfunktion

$$z(x) = \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{7}$$

unter Einhaltung der Massenbalance- und Kapazitätsbedingungen.

1.2 Annahmen

Folgende Annahmen werden getroffen:

1. Alle Daten sind ganzzahlig.
   → Multiplikation

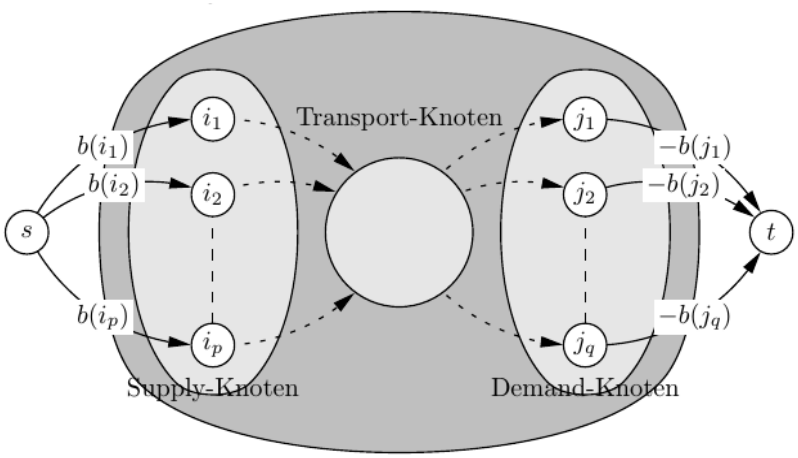
2. Der Graph G ist gerichtet.
   → Kpt 2.4 Undirected Arcs to Directed Arcs

3. Es gilt $\sum_{i \in N} b(i) = 0$ und das Min-Cost Flow Problem hat eine zulässige Lösung.
   → Kapitel 1.2.1

4. Es gibt einen direkten Weg zwischen jedem Paar von Knoten mit unbeschränkter Kapazität.
   $\rightarrow$ Füge künstliche Kanten $(1, j)$ und $(j, 1)$ für jedes $j \in N$ mit sehr grossen Kosten und unendlicher Kapazität ein.

5. Alle Kosten sind nicht negativ. $\rightarrow$ Kpt 2.4 Arc Reversal

1.2.1 Test: hat das Min-Cost Flow Problem eine Lösung?



Konstruktion:
Wir fügen zusätzlich einen Quellknoten s^* ein, welcher mit allen Knoten i mit $b(i) > 0$ verbunden ist. Die Kanten (s, i) besitzen Kapazität $b(i)$.
Ausserdem fügen wir einen zusätzlichen Zielknoten t^* ein, welcher mit allen Knoten j mit $b(j) < 0$ verbunden ist. Die Kanten (j, t^*) besitzen Kapazität $-b(j)$.
Algorithmus:
Löse das Maximum $s^* - t^*$ Flow Problem.
Interpretation:
Wenn der maximale Fluss alle zu s^* adjazenten Kanten saturiert, besitzt das Min-Cost Flow Problem eine zulässige Lösung.

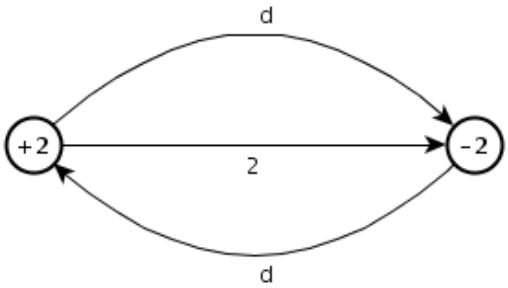
1.3 Restnetzwerk

Gegeben ist ein graph $G = (N, A)$ und eine Flussfunktion x. Die Residualkapazität r_{ij} einer Kante (i, j) ist der maximale zusätzliche Fluss der von i nach j geschickt werden kann. Jede Kante $(i, j) \in A$ wird durch zwei Kanten (i, j) und (j, i) ersetzt. Die Kante (i, j) hat die Kosten c_{ij} und die Restkapazität $r_{ij} = u_{ij} - x_{ij}$; die Kante (j, i) hat die Kosten $-c_{ij}$ und die Restkapazität $r_{ij} = x_{ij}$.
Das Restnetzwerk besteht nur aus Kanten mit positiver Restkapazität.

2 Optimalitätsbedingungen

2.1 Negative Cycle Optimality

Augmenting Cycle Theorem:
Seien x und x^o zwei zulässige Lösungen eines Netzwerkflussproblems.
Dann gilt: $x = x^0 + \sum_{k=1}^{j} f(W_k), k \leq m$, wobei W_k Kreise in $G(x^o)$ sind.
Ausserdem gilt: $c(x) = c(x^o) + \sum_{k=1}^{j} c(W_k) f(W_k)$.



Negative Cycle Optimality:
Ein zulässiger Fluss x^* ist eine optimale Lösung des Min-Cost Flow Problems genau dann, wenn es keine negativen Kreise im Restnetzwerk gibt.

Beweis:
$\boxed{\Rightarrow}$
Annahme: $\exists$ ein zulässiger Fluss x und $G(x)$ enthält einen negativen Kreis.
Durch eine Flusserhöhung entlang des negativen Kreises werden die Kosten gesenkt.
$\Rightarrow x$ kann nicht optimal sein.
$\boxed{\Leftarrow}$
Annahme: x^* ist ein zulässiger Fluss und $\nexists$ negativen Kreise in $G(x^*)$.
Sei x^o ein optimaler Fluss und $x^* \neq x^o$, dann ist $x^o - x^*$ eine Flusszirkulation. Die Kosten der Flüsse auf diesen Kreisen entsprichen der Differenz $c(x^o) - c(x^*)$. Da alle Kreise in $G(x^*)$ nicht negativ sind, gilt: $c(x^o) - c(x^*) \geq 0$, also $c(x^o) \geq c(c^*)$.
Aus der Optimalität von x^o folgt: $c(x^o) \leq c(x^*)$.
$\Rightarrow c(x^o) = c(x^*) \quad \Rightarrow x^*$ ist optimal.

2.2 Reduced Cost Optimality

Definition:
Sei $\pi : N \to \mathbb{R}$ eine Kantenbewertung. Die reduzierten Kosten $c^\pi : A \to \mathbb{N}$ sind definiert durch $c_{ij}^\pi = c_{ij} - \pi(i) + \pi(j)$.

Interpretation: c_{ij}^{π} sind optimale reduzierte Kosten für eine Kante (i, j) in der Hinsicht, dass es die Kosten dieser Kante relativ zu den kürzeste Wege Distanzen $\pi(i)$ und $\pi(j)$.

Eigenschaften der reduzierten Kosten

- Sei P ein Pfad von k nach l, dann gilt

$$\sum_{(i,j)\in P} c_{ij}^{\pi} = \sum_{(i,j)\in P} c_{ij} - \pi(k) + \pi(l) \tag{8}$$

- Sei W ein gerichteter Kreis, dann gilt

$$\sum_{(i,j)\in W} c_{ij}^{\pi} = \sum_{(i,j)\in W} c_{ij} \tag{9}$$

Reduced Cost Optimality:
Eine zulässige Lösung x^* ist eine optimale Lösung des Min-Cost Flow Problems genau dann, wenn eine Menge von Knotenpotentialen π die folgende reduzierte Kosten Optimalitätsbedingung erfüllt:

$$c_{ij}^{\pi} \geq 0 \;\; \forall (i, j) \in G(x^*) \tag{10}$$

Beweis:
Wir zeigen, dass die reduced cost optimality zur negativ cycle optimality äquivalent ist:
$\boxed{\Rightarrow}$
Annahme: $\exists$ ein Potential π mit $c_{ij}^{\pi} \geq 0 \forall (i, j) \in G(x^*)$.
$\Rightarrow \sum_{(i,j)\in W} c_{ij}^{\pi} \geq 0$ für jeden gerichteten Kreis in $G(x^*)$.
$\Rightarrow \sum_{(i,j)\in W} c_{ij}^{\pi} = \sum_{(i,j)\in W} c_{ij} \geq 0$
$\Rightarrow G(x^*)$ entält keine negativen Kreise.
$\boxed{\Leftarrow}$
Annahme: Für die Lösung x^* enthält $G(x^*)$ keine negativen Kreise.
Sei $d()$ die Distanz des kürzesten Weges von Knoten 1 zu allen anderen Knoten in $G(x^*)$. Wenn Netzwerke keine negativen Kreise besitzen, sind die Distanzlabels wohldefiniert und erfüllen die Dreiecksungleichung $d(j) \leq d(i) + c_{ij}$ für alle Kanten $(i, j) \in G(x^*)$.
$\Rightarrow c_{ij} - (-d(i)) + (-d(j)) \geq 0$. Definiere $\pi = -d$.
$\Rightarrow x^*$ erfüllt die reduced cost optimality Bedingung.

Interpretation: c_{ij} sind die Kosten um eine Flusseinheit von Knoten i nach Knoten j zu transportieren. Wir interpretieren $\mu(i) \equiv -\pi(i)$ als die Kosten, eine Flusseinheit an Knoten i zu erhalten. Dann sind $c_{ij} + \mu(i)$ die Kosten für j, eine Flusseinheit von Knoten i zu erhalten und zu Knoten j zu transportieren.

2.3 Complementary Slackness Optimality Bedingungen

Die Complementary Slackness Optimality Bedingung bezieht sich nicht auf das Restnetzwerk, sondern auf den Originalgraphen.

Complementary Slackness Optimality Bedingungen:
Eine zulässige Lösung x^* ist eine optimale Lösung des Min-Cost Flow Problems genau dann, wenn eine Menge von Knotenpotentialen π, die reduzierten Kosten und Flusswerte die flogenden complementary slackness optimality Bedingungen für jede Kante $(i,j) \in A$ erfüllt:

$$\text{Wenn } c_{ij}^{\pi} > 0, \text{ dann } x_{ij}^* = 0. \tag{11}$$

$$\text{Wenn } 0 < x_{ij}^* < u_{ij}, \text{ dann } c_{ij}^{\pi} = 0. \tag{12}$$

$$\text{Wenn } c_{ij}^{\pi} < 0, \text{ dann } x_{ij}^* = u_{ij}. \tag{13}$$

Beweis:

$\boxed{\Rightarrow}$

Annahme: x ist ein optimaler Fluss.
Aus der reduced cost optimality folgt: $\exists$ Potential π mit $c_{ij}^{\pi} \geq 0$ für alle Kanten $(i,j) \in G(x)$.
Dann folgt:

1. Falls $c_{ij}^{\pi} > 0 \Rightarrow (j,i) \in G(x)$, da $c_{ji}^{\pi} = -c_{ij}^{\pi} < 0$.
   $\Rightarrow x_{ij} = 0$.

2. Falls $0 < x_{ij}^* < u_{ij}$, dann sind (i,j) und (j,i) in $G(x)$.
   $\Rightarrow c_{ij}^{\pi} \geq 0$ und $c_{ji}^{\pi} \geq 0$. Ausserdem gilt $c_{ij}^{\pi} = -c_{ji}^{\pi}$.
   $\Rightarrow c_{ij}^{\pi} = c_{ji}^{\pi} = 0$.

3. Falls $c_{ij}^{\pi} < 0$, dann folgt, dass (i,j) nicht in $G(x)$ ist.
   $\Rightarrow x_{ij} = u_{ij}$.

$\boxed{\Leftarrow}$

Annahme: Sei π so gegeben, dass (11), (12) und (13) für alle $(i,j) \in A$ erfüllt sind.
Zu zeigen: $c_{ij}^{\pi} \geq 0 \; \forall (i,j) \in G(x)$.

1. Kante (i,j): $c_{ij}^{\pi} > 0$ ✓
   Kante (j,i): $x_{ij} = 0 \Rightarrow (j,i) \notin G(x)$ ✓

2. Kante (i,j): $c_{ij}^{\pi} = 0$ ✓
   Kante (j,i): $c_{ji}^{\pi} = 0$ ✓

3. Kante (i,j): $x_{ij}^* = u_{ij} \Rightarrow (i,j) \notin G(x)$ ✓
   Kante (j,i): $c_{ij}^{\pi} < 0 \Rightarrow c_{ji}^{\pi} = -c_{ij}^{\pi} > 0$ ✓

3 Minimum Cost Flow Dualität

Zu jedem primalen Problem gibt es ein duales Problem. Für das primale Min-Cost Flow Problem ist es das folgende Maximierungsproblem:

$$\text{Maximiere} \quad w(\pi, y) = \sum_{i \in N} b_i \pi_i - \sum_{(i,j) \in A} u_{ij} y_{ij} \tag{14}$$

$$\begin{aligned} s.t. \qquad c_{ij} - \pi_i + \pi_j + y_{ij} &\geq 0 \qquad \forall (i,j) \in A, && (15)\\ y_{ij} &\geq 0 \qquad \forall (i,j) \in A, && (16)\\ \pi_i \text{ vorzeichenunbeschränkt} &\qquad \forall i \in N. && (17) \end{aligned}$$

Allgemein gilt: primales und duales Problem besitzen den gleichen Optimalwert.

4 Erste Algorithmen

4.1 Der Cycle Canceling Algorithmus

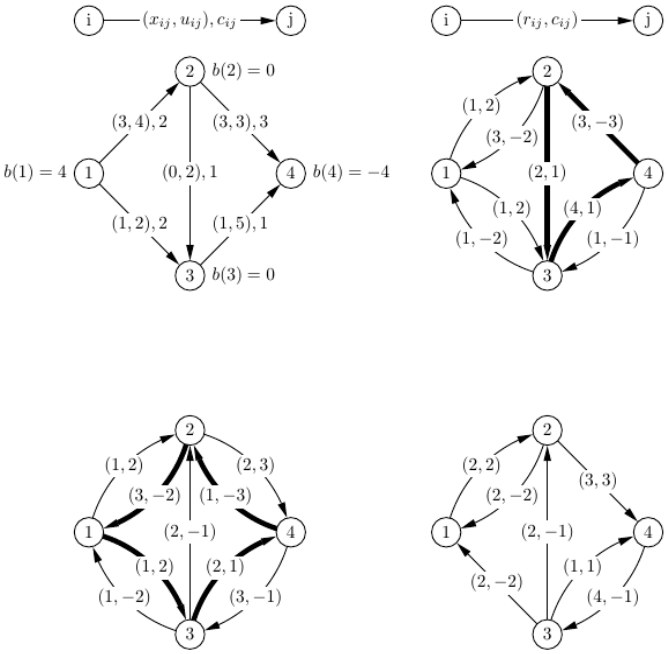
Der Cycle Canceling Algorithmus basiert auf der Negative Cycle Optimality Bedingung.

Idee:
Der Algorithmus berechnet als erstes einen zulässigen Fluss. Dieser kann wie in Abschnitt 1.2.1 bestimmt werden.
Anschliessend werden negative Kreise im Restnetzwerk $G(x)$ gesucht und durch Flussvergrösserungen eliminiert.
Der Algorithmus terminiert, wenn im Restnetzwerk keine negativen Kreise mehr vorhanden sind.

Implementierung:

```
BEGIN
  erzeuge einen feasible flow x in G;
  WHILE G(x) enthaelt einen negativen Kreis DO
  BEGIN
    finde einen negativen Kreis W in G(x);
    δ = min{r_ij : (i,j) ∈ W};
    erhoehe den Fluss entlang W um, δ Einheiten;
  END
END
```

Beispiel:



Laufzeit: $\mathcal{O}(n \cdot m^2 \cdot C \cdot U)$.

4.2 Der Successive Shortest Path Algorithmus

Der Cycle Canceling Algorithmus startet mit einem zulässigen Fluss x und versucht diesen zu optimieren.
Im Gegensatz dazu startet der Successive Shortest Path Algorithmus mit einem Pseudofluss, der zwar optimal ist und die Kapazitätsbedingungen erfüllt, aber die Massenbalancebedingungen der Knoten verletzt. Der Algorithmus versucht diesen Pseudofluss so zu verändern, dass die Optimalität erhalten bleibt und die Massenbalancebedingungen nicht mehr verletzt werden.

Idee:
In jedem Schritt wählt der Algorithmus einen Knoten s mit Excessüberschuss aus und einen Knoten t mit unerfüllter Nachfrage und sendet Fluss von s nach t entlang eines kürzesten Weges im Restnetzwerk. Der Algorithmus terminiert, wenn die aktuelle Lösung alle Massenbalancebedingungen erfüllt.

Pseudofluss:
Ein Pseudofluss ist eine Funktion $x : A \to \mathbb{N}^+$, welche nur die Kapazitätsbedingungen

erfüllt.
Für einen Pseudofluss s definieren wir die Imbalance eines Knotens i als

$$e(i) = b(i) + \sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} \; \forall i \in N. \tag{18}$$

1. $e(i) > 0$ i ist ein Excess-Knoten,
2. $e(i) = 0$ i ist balanciert,
3. $e(i) < 0$ i ist ein Demand-Knoten.

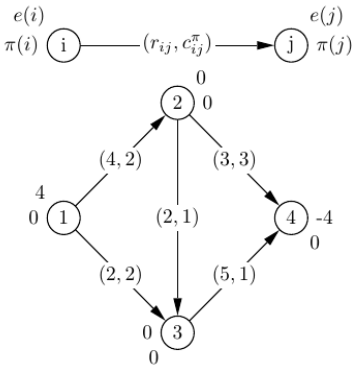
Beobachtungen:
Sei $E = \{i \in N | e(i) > 0\}$ und $D = \{i \in V | e(i) < 0\}$, dann

- $\sum_{i\in N} e(i) = \sum_{i\in N} b(i) = 0$
- $\sum_{i\in E} e(i) = -\sum_{i\in D} e(i)$

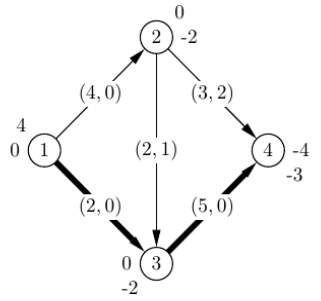
$\Rightarrow$ Hat das Netzwerk einen Excess-Knoten, so muss es auch einen Demand-Knoten haben.
Implementierung:

```
BEGIN
  x=0; π = 0;
  e(i) = b(i) ∀ i ∈ N;
  initialisiere E = {i ∈ N|e(i) > 0} und D = {i ∈ V|e(i) < 0};
  WHILE E ≠ ∅ DO
  BEGIN
    waehle k ∈ E und l ∈ D;
    berechne Shortest-Path-Distanzen d bzgl. der reduzierten Kosten c_ij^π;
    sei P ein Shortest Path von k nach l;
    π = π − d;
    aktualisiere die reduzierten Kosten c_ij^π;
    δ = min{e(k), −e(l), min{r_ij|(i, j) ∈ P}};
    erhoehe den Fluss entlang von P um δ Einheiten;
    aktualisiere x, G(x), e, E, D;
  END
END
```

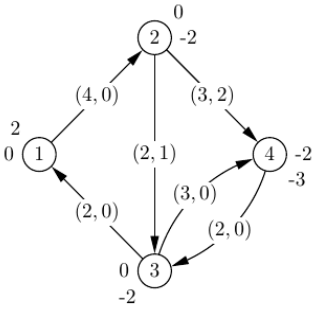
Beispiel:



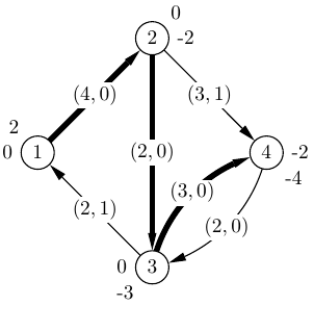
- $G(x)$ am Anfang mit $x = 0$ und $\pi = 0$
- $E = \{1\}$ und $D = \{4\}$
- $k = 1$ und $l = 4$
- $d = (0, 2, 2, 3)$



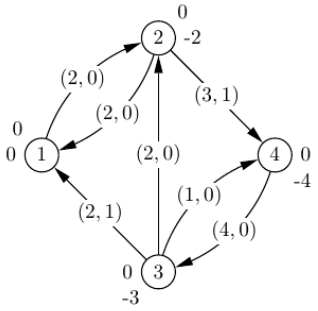
- $P = (1, 3, 4)$
- aktualisiere $\pi = \pi - d$
- aktualisiere $c^\pi_{ij} = c^\pi_{ij} - (d(j) - d(i))$
- $\delta = 2$
- erhöhe Fluß entlang P um δ Einheiten



- aktualisiere $G(x)$ und $e(i)$
- $E = \{1\}$ und $D = \{4\}$
- $k = 1$ und $l = 4$
- $d = (0, 0, 1, 1)$



- $P = (1, 2, 3, 4)$
- aktualisiere $\pi = \pi - d$
- aktualisiere $c^\pi_{ij} = c^\pi_{ij} - (d(j) - d(i))$
- $\delta = 2$
- erhöhe Fluß entlang P um δ Einheiten



- $E = \emptyset$ und $D = \emptyset \Rightarrow$ Algorithmus terminiert