8a. Diskrete Optimierung

Netzwerkflussprobleme

Optimierung SoSe 2020

Dr. Alexey Agaltsov

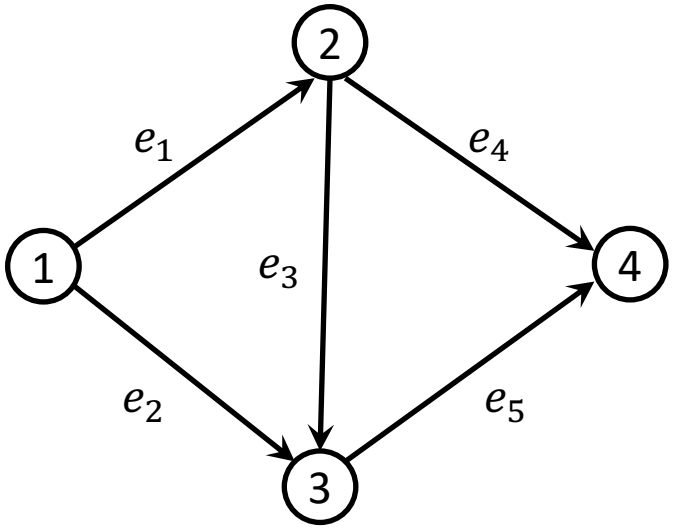
Plan

- Netzwerkflussprobleme: Theorie
- Netzwerkflussprobleme: Beispiele
- Total unimodulare Matrizen

Gerichteter Graph

Ein gerichteter Graph ist ein Tupel $G = (V, E)$ wobei:

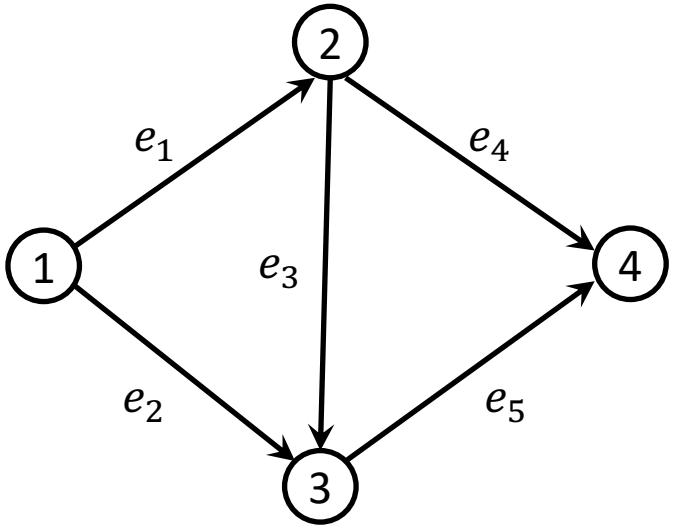
1. $V \neq \emptyset, |V| < \infty$ ist eine Knotenmenge
2. $E \subseteq V \times V$ ist eine gerichtete Kantenmenge



Knoten-Kanten-Inzidenzmatrix

$$A = (a_{ie}) \in \mathbb{R}^{|V| \times |E|}$$

$$a_{ie} = \begin{cases} 1, & i = e[0] \ (i \text{ ist der Startknoten von } e) \\ -1, & i = e[1] \ (i \text{ ist der Endknoten von } e) \\ 0, & \text{sonst} \end{cases}$$



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

Netzwerkflussproblem

Seien gegeben:

- Gerichteter Graph $G = (V, E)$
- Vorrat $b_i \in \mathbb{Z}$ im Knoten $i \in V$

$$b_i < 0 \sim \text{Bedarf}$$

$$\sum_{i \in V} b_i = 0$$

- Untere und obere Transportkapazitäten $l_e \leq u_e \; \forall e \in E$
- Transportkosten $c_e \; \forall e \in E$

Netzwerkflussproblem

Jeder Kante e ordnen wir einen Fluss $x_e \in \mathbb{Z}$ entlang E. Vektor $x = (x_e)_{e \in E}$ heißt zulässiger Netzwerkfluss, falls:

- Die Kapazitätsbeschränkung ist erfüllt:

$$l_e \leq x_e \leq u_e \quad \forall e \in E$$

- Die Bedarfe und die Vorräte werden verglichen:

$$b_i = \sum_{e[0]=i} x_e - \sum_{e[1]=i} x_e \quad \forall i \in V$$

Flusserhaltungsbedingung

$$Ax = b, \quad b = (b_i)_{i \in V}$$

Netzwerkflussproblem

Minimiere *Transportkosten* $= \sum_{e \in E} c_e x_e$ über $x = (x_e)_{e \in E} \in \mathbb{Z}^{|E|}$

u.d.N. $Ax = b$ *Flusserhaltungsbedingung*

$\ell_e \leq x_e \leq u_e \ \forall e \in E$ *Kapazitätsbeschränkung*

Plan

- Netzwerkflussprobleme: Theorie
- Netzwerkflussprobleme: Beispiele
- Total unimodulare Matrizen

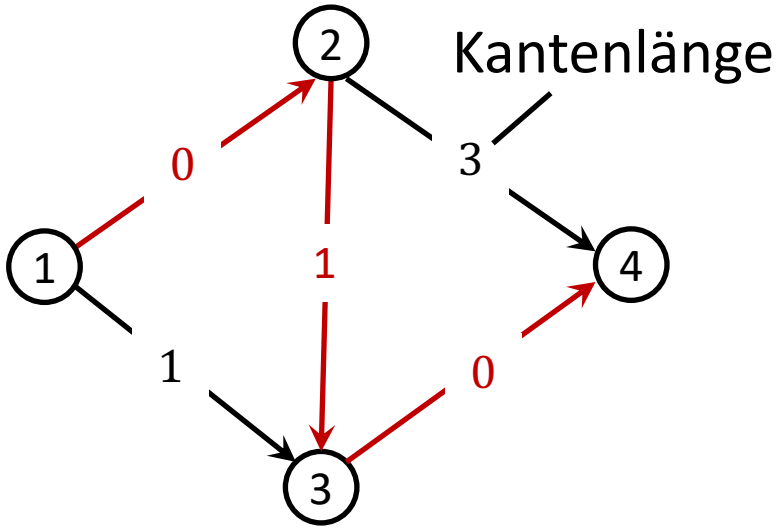
Wege

- Die Kantenfolge $(e_1, \dots, e_N)$ heißt Weg von $e_1[0]$ nach $e_N[1]$ falls:

$$e_i[1] = e_{i+1}[0] \quad i = 1, \dots, N-1$$

- Sei $c_e \geq 0$ die Länge der Kante $e \in E$
- Die Länge des Weges $P = (e_1, \dots, e_N)$:

$$\ell(P) := c_{e_1} + \cdots + c_{e_N}$$



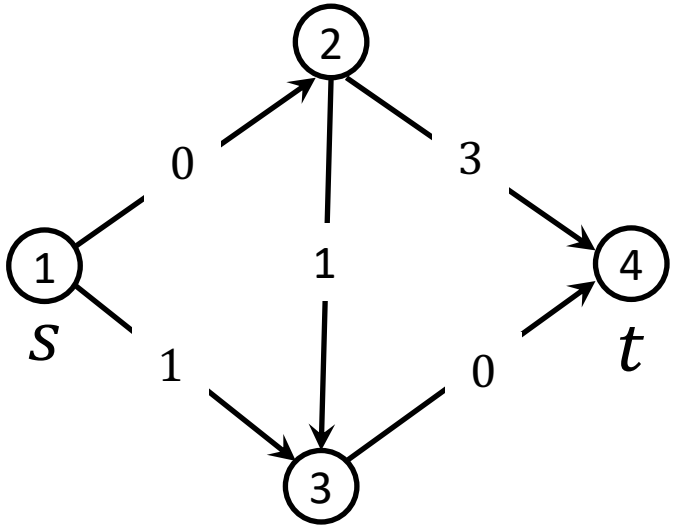
Weg von 1 nach 4

Problem des kürzesten Pfades

Seien gegeben:

- Gerichteter Graph $G = (V, E)$
- Startknoten $s \in V$ und Endknoten $t \in V$
- Die Länge $c_e \geq 0$ jeder Kante $e \in E$

Finde den möglichst kurzen Weg von s nach t



Entscheidungsvariablen

Minimiere *Pfadlänge* $= \sum_{e \in E} c_e x_e$ über $x = (x_e)_{e \in E} \in \mathbb{Z}^{|E|}$

u.d.N. $Ax = b$ *Flusserhaltungsbedingung*

Knoten-Kanten-Inzidenzmatrix $0 \leq x_e \leq 1 \ \forall e \in E$

- Entscheidungsvektor $x = (x_e)_{e \in E}$ beschreibt einen Pfad:

$$x_e = \begin{cases} 1, & \text{Die Kante } e \text{ gehört dem Pfad} \\ 0, & \text{sonst} \end{cases}$$

- Vektor $b = (b_i)_{i \in V}$ der Bedarfe und Vorräte:

$$b_i = \begin{cases} 1, & i = s \quad (i \text{ ist der Startknoten des Graphen}) \\ -1, & i = t \quad (i \text{ ist der Endknoten des Graphen}) \\ 0, & \text{sonst} \end{cases}$$

Beispiel: Zuordnungsproblem

Beim Lagenschwimmen werden konsekutiv vier Schwimmarten von vier Schwimmern verwendet

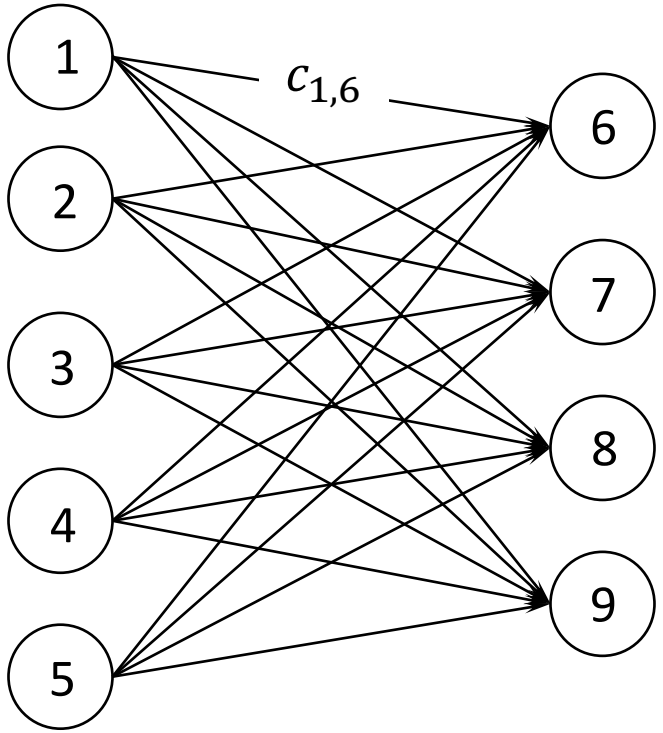
Die besten Schwimmer zeigen die folgenden Durchschnittszeiten:

Schwimmart \ Schwimmer	1	2	3	4	5
Rücken (6)	33.0	35.6	34.6	36.3	33.5
Brust (7)	40.5	35.7	38.5	35.9	38.2
Schmetterling (8)	32.1	29.5	28.8	33.2	30.2
Freistil (9)	29.7	28.9	28.6	30.5	31.0

Ordne die Schwimmer den Schwimmarten so zu, das die durchschnittliche gesamte Schwimmzeit minimal ist

Modellierung

- Knoten $i = 1, \dots, 5$ sind Schwimmer und $i = 6,7,8,9$ Schwimmarten
- Wir fügen eine Kante (i, j) für alle $i \in \{1, \dots, 5\}$ und $j \in \{6, \dots, 9\}$
- Die Länge c_{ij} ist die durchsch. Schwimmzeit für i beim Stil j



Modellierung

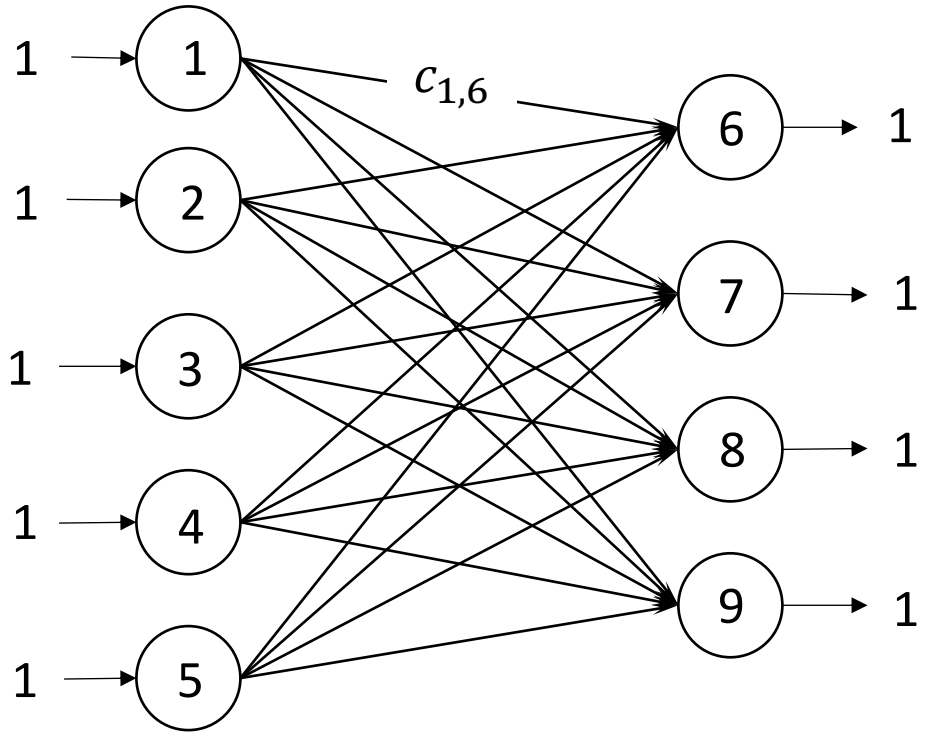
- $x_{ij} = \begin{cases} 1, & i \text{ ist ein Schwimmer, dem die Schwimmart } j \text{ zugeordnet wird} \\ 0, & \text{sonst} \end{cases}$

- Vorräte $b_i = 1$ für $i = 1, \dots, 5$

Jeder Schwimmer schwimmt nur einmal

- Bedarfe $b_i = -1$ für $i = 6, \dots, 9$

Höchstens ein Schwimmer per Stil

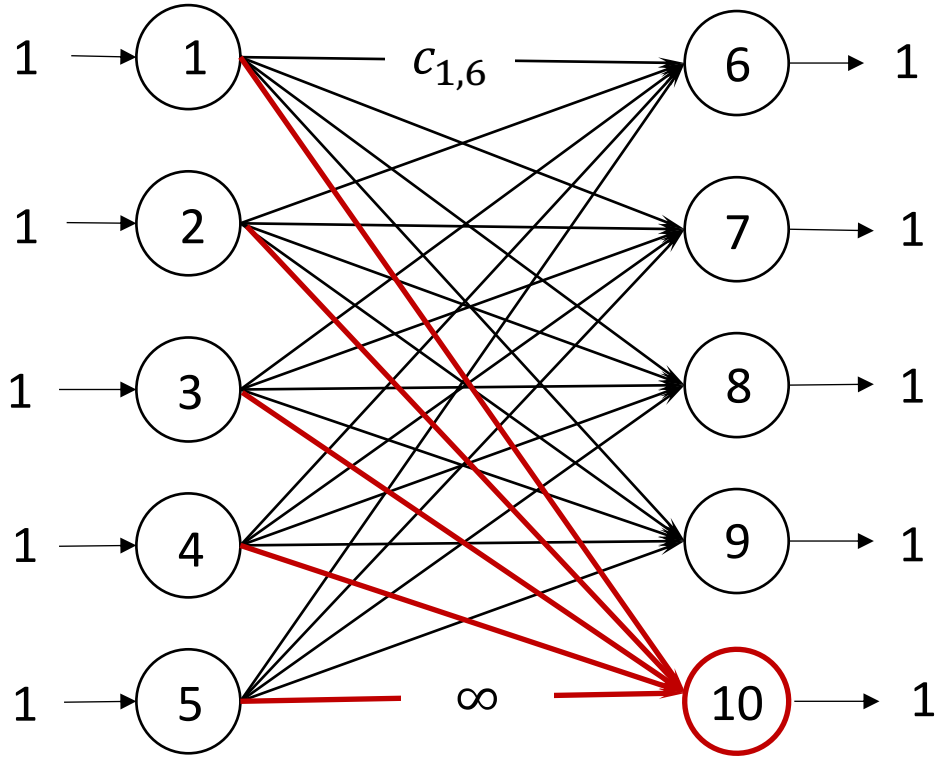


Gesamtvorrat ≠ Gesamtbedarf!

Modellierung

- Wir fügen einen Dummy-Knoten mit Bedarf von 1 hinzu
- Der entsprechende Schwimmer nimmt nicht teil
- Die zugehörigen Schwimmzeiten

$$c_{i,10} = \infty$$



Als Netzwerkflussproblem

Minimiere *Gesamtschwimmzeit* $= \sum_{i,j} c_{ij} x_{ij}$

u.d.N. $Ax = b$ *Flusserhaltungsbedingung*

$0 \leq x_{ij} \leq 1 \quad \forall i, j$

$x_{ij} \in \mathbb{Z}, i \in \{1, \dots, 5\}, j \in \{6, \dots, 10\}$

```python
import numpy as np
import cvxpy as cp

N = 5 # Knotenzahl
M = 1000 # große Zahl

b = np.hstack([np.ones(N),-np.ones(N)])
A = np.vstack([np.repeat(np.eye(N),N,axis=1),
               np.tile(-np.eye(N),N)])

c = np.array([33.0,40.5,32.1,29.7,M, # Schwimmer 1
              35.6,35.7,29.5,28.9,M, # Schwimmer 2
              34.6,38.5,28.8,28.6,M, # Schwimmer 3
              36.3,35.9,33.2,30.5,M, # Schwimmer 4
              33.5,38.2,30.2,31.9,M])# Schwimmer 5

x = cp.Variable(N*N)

# Nebenbedingungen
NB = [ A@x == b, x <= 1, x >= 0]

# Zielfunktion
f = cp.Minimize(cp.sum(cp.multiply(c,x)))

# Lösung
P = cp.Problem(f,NB).solve()

print(x.value)
```

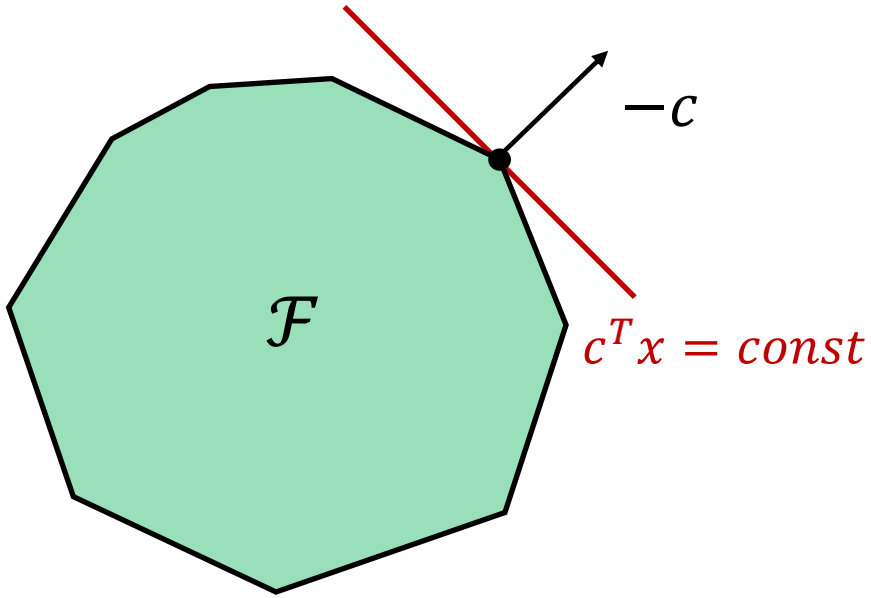
```
Schwimmer 1 [1 0 0 0 0
Schwimmer 2  0 0 0 1 0
Schwimmer 3  0 0 1 0 0
Schwimmer 4  0 1 0 0 0
Schwimmer 5  0 0 0 0 1]
```

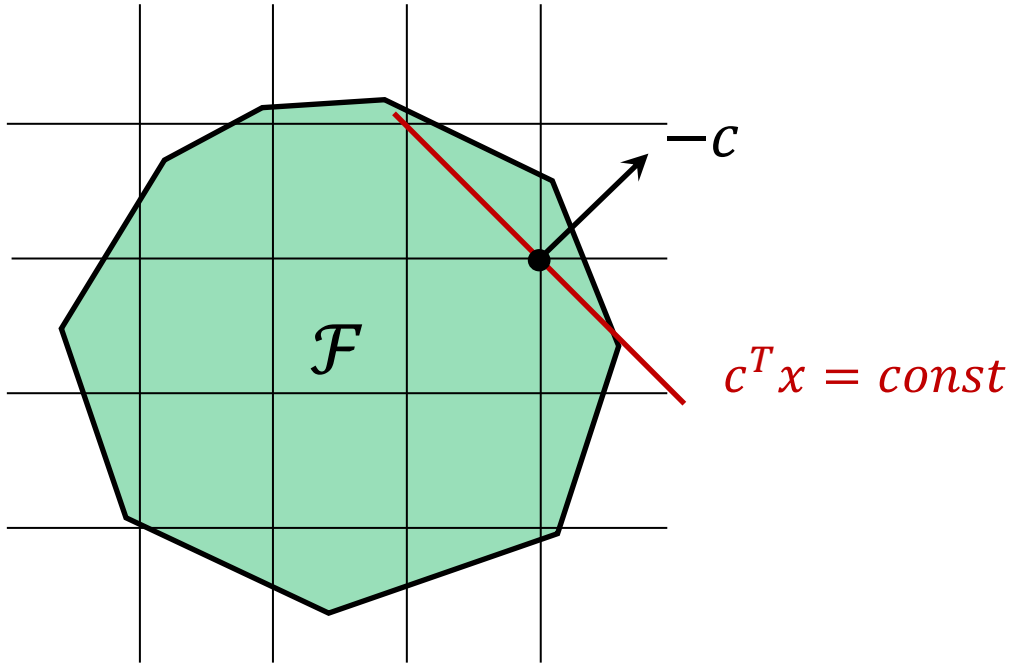
Rücken Brust Schmetterling Freistil –

Plan

- Netzwerkflussprobleme: Theorie
- Netzwerkflussprobleme: Beispiele
- Total unimodulare Matrizen

Ganzzahliges lineares Programm





Minimiere $c^T x$ über $x \in \mathbb{R}^n$

u.d.N. $Ax \leq b$

Minimiere $c^T x$ über $x \in \mathbb{Z}^n$

u.d.N. $Ax \leq b$

Ganzzahlige Lösungen

Minimiere $c^T x$ über $x \in \mathbb{R}^n$
u.d.N. $Ax \leq b$

Minimiere $c^T x$ über $x \in \mathbb{Z}^n$
u.d.N. $Ax \leq b$

- Seien alle Eckpunkte von $\mathcal{F} = \{x \in \mathbb{R}^n : Ax \leq b\}$ *ganzzahlig* (i.e. $\in \mathbb{Z}^n$)
- Sei $\bar{x}$ eine optimale Lösung des linearen Programms, die wir mit dem Simplex-Verfahren fanden
- Dann ist $\bar{x}$ eine optimale Lösung des ganzzahligen linearen Programms

Wie kann man garantieren, dass alle Eckpunkte von $\mathcal{F}$ ganzzahlig sind?

Total unimodulare Matrizen

- $A \in \mathbb{Z}^{m \times n}$ heißt total unimodular (kurz TU), falls für jede quadratische Teilmatrix $B \subseteq A$ gilt:

$$\det B \in \{0,1,-1\}$$

- Also kann A nur Elemente $0,1,-1$ enthalten

Satz 8.1. Ganzzahlige Eckpunkte

$$\mathcal{F} = \{x \in \mathbb{R}^n : Ax \leq b\} \qquad A \text{ hat vollen Spaltenrang}$$

Angenommen:

- $A \in \mathbb{Z}^{m \times n}$ ist total unimodular
- $b \in \mathbb{Z}^m$

Ist $\bar{x}$ ein Eckpunkt von $\mathcal{F}$, so $\bar{x} \in \mathbb{Z}^n$

Beweis

Sei $\bar{x} \in \mathcal{F}$ ein Eckpunkt

Satz 6.5

$\exists \mathcal{B} \subseteq \{1, \dots, m\}$ so, dass:

- $A[\mathcal{B}, :] \in \mathbb{R}^{n \times n}$ ist regulär
- $A[\mathcal{B}, :]\bar{x} = b[\mathcal{B}]$

$$\bar{x} = A[\mathcal{B}, :]^{-1} b[\mathcal{B}] \quad \in \mathbb{Z}^n$$
$$\in \mathbb{Z}^{n \times n} \quad \in \mathbb{Z}^n$$

□

TU-erhaltende Operationen

- A ist TU $\Leftrightarrow$ A^T ist TU

  B ist eine Teilmatrix von A $\Leftrightarrow$ B^T ist eine Teilmatrix von A^T

- A ist TU $\Leftrightarrow$ $[A \;\; I]$ ist TU, wobei I ist die Einheitsmatrix

TU-erhaltende Operationen

- A ist TU $\Leftrightarrow$ A^T ist TU

  B ist eine Teilmatrix von A $\Leftrightarrow$ B^T ist eine Teilmatrix von A^T

- A ist TU $\Leftrightarrow$ $[A \;\; I]$ ist TU, wobei I ist die Einheitsmatrix

  Beispiel:

$$[A, I] = \begin{bmatrix} a_{11} & a_{12} & 1 & 0 \\ a_{21} & a_{22} & 0 & 1 \end{bmatrix}$$

Teilmatrizen 1×1 *sind gleich* $0,1, a_{ij}$

Teilmatrizen 2×2 *sind* $A, I, \begin{bmatrix} a_{1j} & 1 \\ a_{2j} & 0 \end{bmatrix}, \begin{bmatrix} a_{1j} & 0 \\ a_{2j} & 1 \end{bmatrix}$

Ihre Determinanten sind gleich $0,1, \pm a_{ij}, \det A$

Satz 8.2. Hinreichende Bedingung der TU

Sei $A = (a_{ij}) \in \mathbb{Z}^{m \times n}$ mit $a_{ij} \in \{0,1,-1\}$ $\forall i,j$. Angenommen:

- Jede Spalte von A enthält höchstens 2 verschiedene von Null Elemente
- $\{1,\dots,m\} = M_1 \sqcup M_2$, sodass für jede Spalte j mit zwei verschiedenen von Null Elementen gilt:

$$\sum_{i \in M_1} a_{ij} = \sum_{i \in M_2} a_{ij}$$

Dann ist A total unimodular

Beweis

Angenommen, A ist nicht TU

Sei $B = A[I, K]$ die kleinste Teilmatrix mit $\det B \notin \{0,1,-1\}$

Jede Spalte von B enthält zwei verschiedene von Null Elemente

Annahme

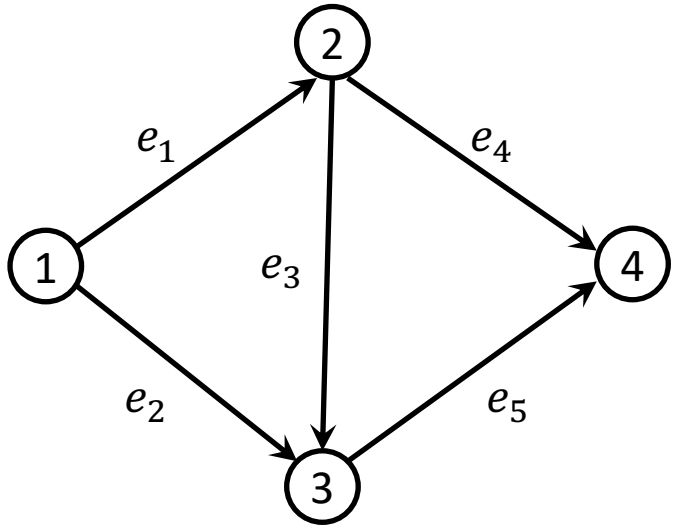
$$\sum_{i \in I \cap M_1} B[i,:] - \sum_{i \in I \cap M_2} B[i,:] = 0$$

Zeilen von B sind linear abhängig

Widerspruch zu $\det B \neq 0$

□

Beispiel 8.3: Knoten-Kanten-Inzidenzmatrix



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

- Jede Spalte der Inzidenzmatrix enthält genau zwei $\neq 0$ Elemente: 1 und -1

  Jede Kante hat nur einen Startknoten und nur einen Endknoten

- Die Knoten-Kanten-Inzidenzmatrix ist total unimodular nach Satz 8.2

  $M_1 = \{1,2,3,4\},\ M_2 = \emptyset$

- Mehrere Netzwerkflussprobleme lassen sich mit dem Simplex-Verfahren lösen

Zusammenfassung

- Netzwerkflussprobleme: Theorie
- Netzwerkflussprobleme: Beispiele
- Total unimodulare Matrizen

Nächstes Video

- 8b. Diskrete Optimierung: Branch-and-Bound