Rebecca Nauli
naulireb@students.zhaw.ch

Optimierungsprobleme & Lineare Programme

Allgemeines Optimierungsproblem

- **Allgemeines Optimierungsproblem:**

$$\Pi : \min\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in S\}, \quad \text{wobei } S \subseteq \mathbb{R}^n$$

- Entscheidungsvariablen: $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$
- zulässige Menge: $S \subseteq \mathbb{R}^n$
- Zielfunktion: $f : S \to \mathbb{R}$
- **zulässige Lösung:** $x \in S$
- **optimale Lösung:** $x^* \in S$ so dass $f(x^*) \leq f(x)$ für alle $x \in S$
- **Optimum, optimaler Zielfunktionswert:** $f(x^*)$

Optimierungsproblem Rezept

1. Variablen definieren
2. Parameter definieren
3. Zielfunktion definieren
4. Nebenbedingungen definieren

Beispiel Transportproblem:

Firma XY produziert in Basel und Zürich ein bestimmtes Produkt. An einem Tag werden 14 Tonnen in Basel und 16 Tonnen in Zürich produziert. Die Firma hat drei Abnehmer, die sich in Luzern, Chur und Winterthur befinden. Der tägliche Bedarf der Abnehmer ist 9 Tonnen für Luzern, 10 Tonnen für Chur und 11 Tonnen für Winterthur.

Firma XY möchte ihre Transportkosten minimieren, deshalb hat sie die Transportpreise pro Tonne auf allen Verbindungen erhoben. Die Ergebnisse sind in folgender Tabelle zusammen getragen.

	Luzern	Chur	Winterthur
Basel	11	4	5
Zürich	5	4	7

Wie soll die Firma XY ihre täglichen Transporte planen, damit die täglichen Transportkosten so klein wie möglich sind?

Transportproblem:

Variablen:

$x_{BL}, x_{BC}, x_{BW}, x_{ZL}, x_{ZC}, x_{ZW}$

Zielfunktion:

$11 x_{BL} + 4 x_{BC} + 5 x_{BW} + 5 x_{ZL} + 4 x_{ZC} + 7 x_{ZW} \rightarrow \min$

Nebenbedingungen:

$x_{BL} \geq 0,\ x_{BC} \geq 0,\ x_{BW} \geq 0,\ x_{ZL} \geq 0,\ x_{ZC} \geq 0,\ x_{ZW} \geq 0$

$x_{BL} + x_{BC} + x_{BW} = 14$

$x_{ZL} + x_{ZC} + x_{ZW} = 16$

$x_{BL} + x_{ZL} = 9$

$x_{BC} + x_{ZC} = 10$

$x_{BW} + x_{ZW} = 11$

Rebecca Nauli
naulireb@students.zhaw.ch

Definition: Lineare Funktion & Lineare Programm (LP)

Definition: Lineare Funktion

Eine **Funktion** $f : \mathbb{R}^n \to \mathbb{R}$ heisst **linear**, falls $f(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{x}$, wobei $\boldsymbol{a} \in \mathbb{R}^n$.

Definition: Lineare Nebenbedingung

Eine **Nebenbedingung** heisst **linear**, falls sie eine der folgenden Formen hat

- $f(\boldsymbol{x}) \leq \beta$
- $f(\boldsymbol{x}) \geq \beta$
- $f(\boldsymbol{x}) = \beta$

wobei f eine lineare Funktion ist und $\beta \in \mathbb{R}$.

Definition: Lineares Programm (LP)

Das Problem der Minimierung/Maximierung einer linearen Funktion, so dass eine endliche Anzahl von linearen Nebenbedingungen erfüllt ist, heisst **lineares Programm**.

Beispiel Zuordnungsproblem:

Firma WaterTech hat 4 Jobs zu vergeben. Diese sollen von 4 Mitarbeitern ausgeführt werden. Aus Erfahrung weiss die Firma, wie lange ein bestimmter Mitarbeiter für einen bestimmten Job braucht. Diese Informationen sind in folgender Tabelle aufgeführt.

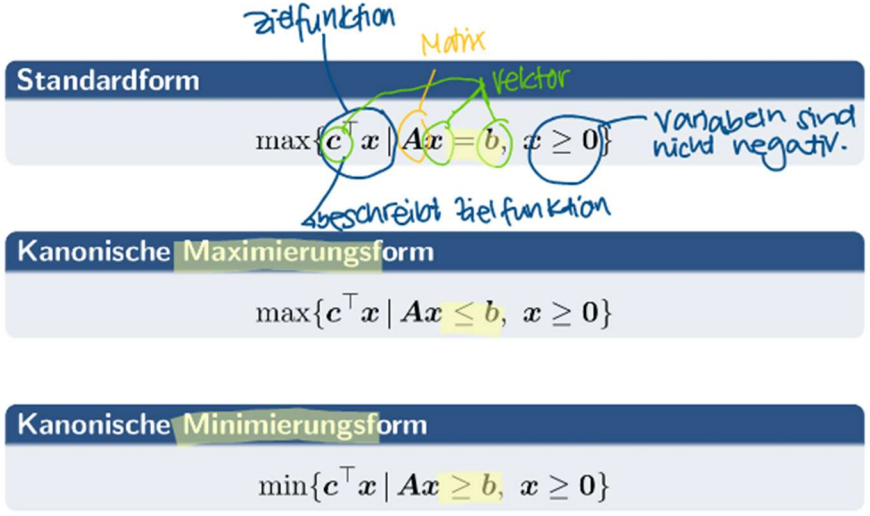
MA / Job	1	2	3	4
1	3	5	1	7
2	8	2	2	4
3	2	1	6	8
4	8	3	3	2

Zusätzlich soll gelten:

- Jedem MA soll genau ein Job zugewiesen werden
- Jeder Job soll genau einem MA zugewiesen werden

Wie sollen die Jobs den einzelnen MA zugeteilt werden, damit die Gesamtzeit, die für die Ausführung der Jobs notwendig ist, minimal wird?

Lineare Programme – verschiedene Formen



Rebecca Nauli
naulireb@students.zhaw.ch

Transformation zwischen Formen

- Ungleichung $\leq$ zu Ungleichung $\geq$:
  $\boldsymbol{a}^\top \boldsymbol{x} \leq b \quad \leftrightarrow \quad -\boldsymbol{a}^\top \boldsymbol{x} \geq -b$
- Gleichung zu Ungleichungen:
  $\boldsymbol{a}^\top \boldsymbol{x} = b \quad \rightarrow \quad \boldsymbol{a}^\top \boldsymbol{x} \leq b, \ \boldsymbol{a}^\top \boldsymbol{x} \geq b$
- Ungleichungen zu Gleichungen:
  $\boldsymbol{a}^\top \boldsymbol{x} \leq b \quad \rightarrow \quad \boldsymbol{a}^\top \boldsymbol{x} + s = b, \ s \geq 0$
  $\boldsymbol{a}^\top \boldsymbol{x} \geq b \quad \rightarrow \quad \boldsymbol{a}^\top \boldsymbol{x} - s = b, \ s \geq 0$
- Nichpositive Variable zur nichtnegativen Variable:
  $x_j \leq 0 \quad \rightarrow \quad \overline{x_j} := -x_j, \ \overline{x_j} \geq 0$
- Freie Variable zu nichtnegativen Variablen:
  x_j frei $\quad \rightarrow \quad x_j = x_j^+ - x_j^-, \ x_j^+, x_j^- \geq 0$

Ergänzungen zu den Transformationen:

- s: *slack variable*, dt: *Schluckvariabel*, s *ist unbekannt*
- *NB*: *Nicht positive Variabel in* $\rightarrow \cdot -1$ *Vorzeichenwechsel*
- *freie Variabel*: $x \in \mathbb{R}, x_1 \rightarrow x_1 - x_2; \ x_1, x_2 \geq 0$
- *ZF*: *Von* ***min zu max*** *(& vice versa)* $\rightarrow \cdot -1$ *(Vorzeichenwechsel)*
- !!! *Gleichung zu Ungleichung ergibt* **2** *Ungleichungen* !!!

Beispiel Umformung

$\max \{ c^\top x : Ax = b, x \geq 0 \}$ Standardform
$\max \{ c^\top x : Ax \leq b, x \geq 0 \}$ kan. Maximierungsform
$\min \{ c^\top x : Ax \geq b, x \geq 0 \}$ kan. Minimierungsform

$A = \begin{pmatrix} -1 & 2 \\ 2 & 1 \\ 2 & -1 \end{pmatrix}$ 3×2

$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ $x = (x_1, x_2)^\top$

$C = (1,1)^\top \rightarrow (x_1 + x_2 \rightarrow \max)$ kommt davon

$b = (8, 14, 10)^\top$

Standardform

$x_1 + x_2 \rightarrow \max$

$-x_1 + 2x_2 + x_3 = 8$

$2x_1 + x_2 + x_4 = 14$

$2x_1 - x_2 + x_5 = 10$

$x_1, x_2, x_3, x_4, x_5 \geq 0$

$A = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ -1 & 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 \\ 2 & -1 & 0 & 0 & 1 \end{pmatrix} \rightarrow 3 \times 5$

$x = (x_1, x_2, x_3, x_4, x_5)^\top$

$C = (1, 1, 0, 0, 0)$

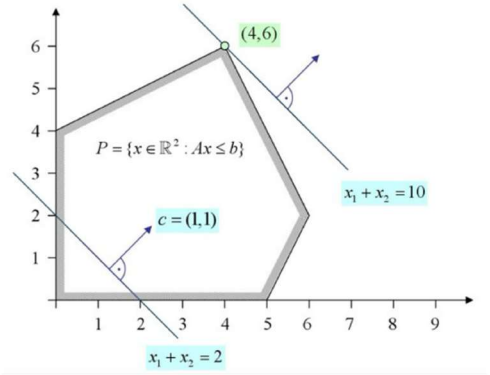
$b = (8, 14, 10)^\top$

Definition: Polyeder

Definition

Ein **Polyeder** $P \subseteq \mathbb{R}^n$ ist ein Durchschnitt von endlich vielen Halbräumen, d.h. $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{Ax} \leq \boldsymbol{b}\}$.

$$\begin{aligned} x_1 + x_2 &\rightarrow \max \\ -x_1 + 2x_2 &\leq 8 \\ 2x_1 + x_2 &\leq 14 \\ 2x_1 - x_2 &\leq 10 \\ x_1, x_2 &\geq 0 \end{aligned}$$



Rebecca Nauli
naulireb@students.zhaw.ch

Optimale Lösung

Definition

Ein LP heisst **zulässig,** falls $P \neq \emptyset$, d.h. es gibt eine zulässige Lösung).
Ein LP heisst **unzulässig,** falls $P = \emptyset$, d.h. es gibt keine zulässige Lösung.
Ein LP heisst **unbeschränkt,** falls es für jede Zahl $\alpha \in \mathbb{R}$ eine zulässige Lösung $\boldsymbol{x}^\alpha \in P$ gibt mit $\boldsymbol{c}^\top \boldsymbol{x}^\alpha \geq \alpha$ (im Fall einer Maximierung).

Zulässige Lösungen	Optimale Lösungen	Grafisch
Zulässig	• exakt 1 optimale Lösung • ∞ viele optimale Lösungen -> keine optimalen Lösungen	• 1 Ecke im Polyeder • Mind. 2 Ecken oder Seite im Polyeder
Unzulässig	• Keine optimalen Lösungen	• Leere Menge
Unbeschränkt	• Keine optimale Lösung	• /

Ergänzungen:

- Jede optimale Lösung ist eine zulässige Lösung.
- Nicht jede zulässige Lösung ist optimal.

Lösbarkeit von LP

Satz

Jedes LP erfüllt genau eine der folgenden Möglichkeiten:

- Das LP ist unzulässig
- Das LP ist unbeschränkt
- Das LP besitzt genau eine optimale Lösung
- Das LP besitzt unendlich viele optimale Lösungen

Satz

Es sei ein LP mit der zulässigen Menge P gegeben. Zusätzlich gelte Folgendes:

(i) Das Polyeder P besitzt mindestens eine Ecke

(ii) Das LP besitzt mindestens eine optimale Lösung

Dann ist mindestens eine Ecke des Polyeders P optimal.

Rebecca Nauli
naulireb@students.zhaw.ch

Simplex-Methode

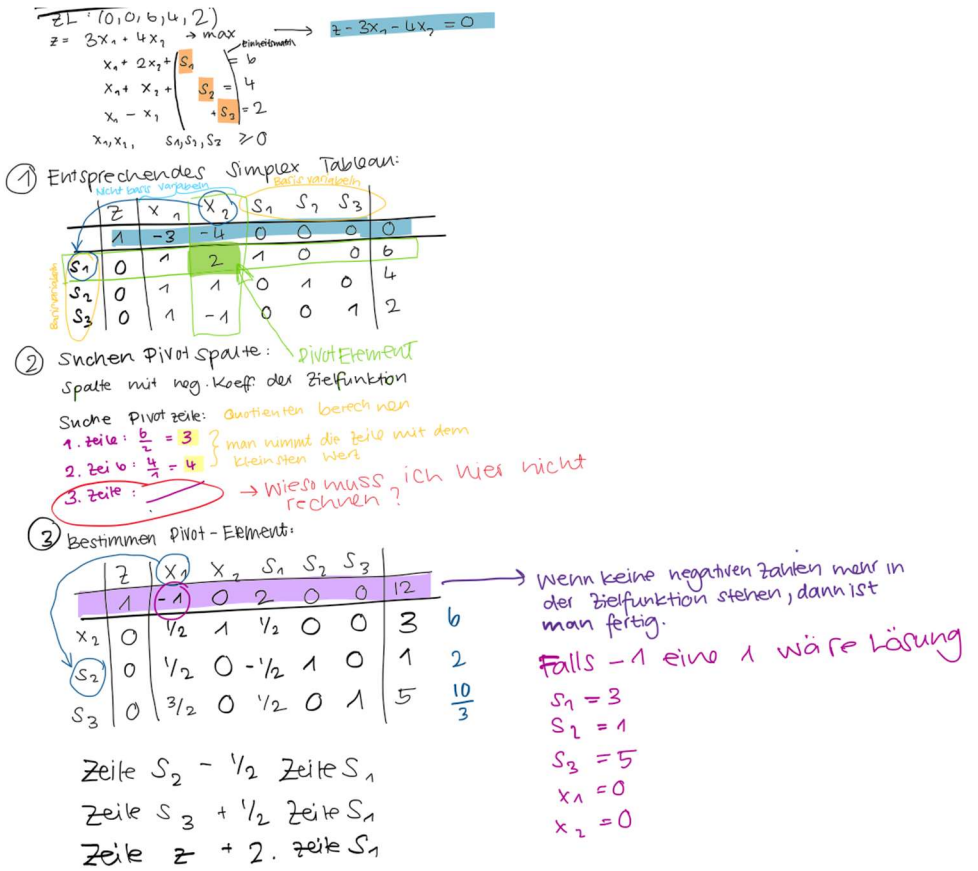
Voraussetzungen:

- Gültiges Simplex-Tableau
  - Rechte Seite muss $b \geq 0$
  - Alle Spalten der Einheitsmatrix vorhanden → Basisvariablen (B), Rest: Nichtbasisvariablen (NB)
    - $Basisvariabel = b\ (rechte\ Seite)$
    - $Nichtbasisvariabel\ = 0$
  - Pivotspalte: kleinster negativer Koeffizient in Zielfunktion
  - Pivotzeile: kleinster Quotient → $Quotient = \frac{rechte\ Seite\ (b)}{Pivotspaltenelement}$
  - Pivotelement: Schnittpunkt von Pivotzeile &-spalte
- Optimales Simplex Tableau
  - z-Zeile → Alle Einträge unter $NB\ \geq\ 0$ → 1 optimale Lösung
  - z-Zeile → Einträge unter min. $NB\ \leq\ 0$ → unendlich viele optimale Lösung
- **Unbeschränktes LP**
  - Negativer Koeffizient in z-Zeile ohne PE (da PE ≤ 0) → unbeschränkt

Aufstellung Simplex Tableau Procedure:

1. Von gegebener Form in Standardform umformen
2. Tabelle erstellen
3. Tabelle einfüllen
4. Pivot-Spalte (PS) identifizieren (kleinster Koeffizient)
5. Pivot-Zeile (PZ) identifizieren (kleinster Quotient)
   a. !!! ignoriere alle Quotienten = negativ oder 0
6. Pivot-Element (PE) bestimmen (Schnittpunkt PS & PZ)
7. PZ mit Konstante multiplizieren sodass PE = 1
8. Übrige Zeile multiplizieren mit Vielfachen der Pivotzeile damit Element in Pivot-Spalte = 0
9. Fertig wenn Koeffizienten in z-Zeile >0 (nicht-negativ)

Beispiel Simplex Tableau



Rebecca Nauli
naulireb@students.zhaw.ch

Zwei-Phasen-Methode

Startbedingungen:

- LP in Maximierungsform mit nicht negativen Variablen (x) & - rechten Seiten (b)
- Neue Variablen
  - Schlupfvariablen → Standardform
  - Jede Gleichung ohne Schlupfvariable → künstliche Variable

Originale NB	Neue Variablen
≤	+ s (schlupf)
=	+ y (künstliche)
≥	-s +y (- schlupf + künstliche)

- Zielfunktion (erste Phase)
  - Minimiere die Summe aller künstlichen Variablen
  - Negative künstliche Variablen maximieren
- Erste Phase:
  - Künstliche Zielfunktion (w) = 0 → zulässig
  - Künstliche Zielfunktion w ≠ 0 → unzulässig

Satz

Sei das LP

$$\max\{\boldsymbol{c}^\top \boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\} \qquad (1)$$

gegeben, wobei $\boldsymbol{b} \geq \boldsymbol{0}$. Weiter betrachten wir das LP

$$\min\{\boldsymbol{1}^\top \boldsymbol{y} \mid \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}, \boldsymbol{y} \geq \boldsymbol{0}\} \qquad (2)$$

Dann gilt:

- Das LP (2) besitzt eine optimale Lösung.
- Das LP(1) besitzt eine zulässige Lösung $\iff$ der optimale Zielfunktionswert von (2) ist Null.

Beispiel Zwei-Phasen-Methode:

Beispiel:

$x_1 + x_2 \to \max$

$x_1 + 2x \leq 6$

$2x_1 + x_2 = 6$

$x_1 + 3x_2 \geq 6$

$x_1, x_2 \geq 0$

$x_1 + x_2 \to \max$

$x_1 + 2x_2 + s_1 = 6$

$2x_1 + x_2 + y_1 = 6$

$x_1 + 3x_2 - s_2 + y_2 = 6$

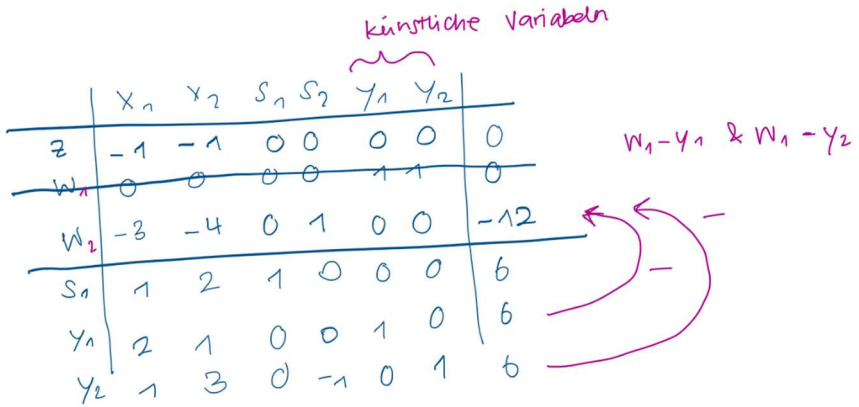
$x_1, x_2 \geq 0$

$s_1, s_2 \geq 0$

$y_1, y_2 \geq 0$

$y_1, y_2 \to \min$

$W = -y_1 - y_2 \to \max$



Rebecca Nauli
naulireb@students.zhaw.ch

Basislösung

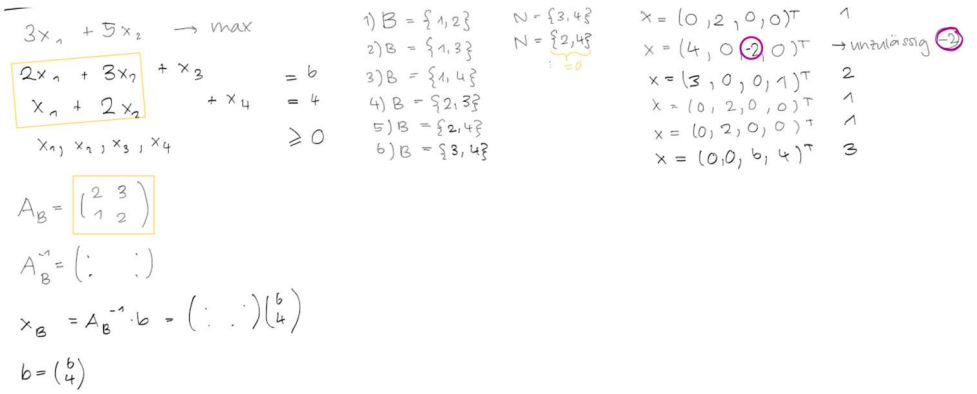
Satz

Wenn alle Koeffizienten in der $z-$Zeile eines gültigen Simplex-Tableaus nichtnegativ sind, d.h. wenn

$$c_B^\top A_B^{-1} A_N - c_N^\top \geq 0,$$

dann ist die zulässige Basislösung $\begin{pmatrix} A_B^{-1} b \\ 0 \end{pmatrix}$ optimal.

Beispiel Basislösung:



Dualität linearer Programme

(P) LP in kanonischer Form → (D) LP in kanonischer Form

Primal-duales Paar:

$$\begin{array}{llll} & c^T x & \to & \max \\ (\mathrm{P}) & Ax & \leq & b \\ & x & \geq & 0 \end{array} \qquad \begin{array}{llll} & b^T y & \to & \min \\ (\mathrm{D}) & A^T y & \geq & c \\ & y & \geq & 0 \end{array}$$

http://www.orklaert.de/dualitaet.php

Dual zu dual = primal

Gegeben sei das primal-duale Paar (P)-(D) in kanonischer Form. Das duale Problem zu (D) ist (P).

LP in Standardform → in duales Problem

Primal-duales Paar:

$$\begin{array}{llll} & c^T x & \to & \max \\ (\mathrm{P1}) & Ax & = & b \\ & x & \geq & 0 \end{array} \qquad \begin{array}{llll} & b^T y & \to & \min \\ (\mathrm{D1}) & A^T y & \geq & c \end{array}$$

Vom Primalproblem zum Dualproblem II:

max	$\leftrightarrow$	min
Zielfunktionsvektor	$\leftrightarrow$	Vektor der rechten Seite
Vektor der rechten Seite	$\leftrightarrow$	Zielfunktionsvektor
A	$\leftrightarrow$	A^T
Nebenbedingung "$\leq$"	$\leftrightarrow$	Variable ≥ 0
Nebenbedingung "$=$"	$\leftrightarrow$	freie Variable
Nebenbedingung "$\geq$"	$\leftrightarrow$	Variable ≤ 0
Variable ≥ 0	$\leftrightarrow$	Nebenbedingung "$\geq$"
freie Variable	$\leftrightarrow$	Nebenbedingung "$=$"
Variable ≤ 0	$\leftrightarrow$	Nebenbedingung "$\leq$"

Rebecca Nauli
naulireb@students.zhaw.ch

Beispiel Dualität

(P) kan. Form → (D) kan. Form:

Variablen

$x_i \quad i = 1, \dots, 3$ die zu produzierende Menge von Produkt i

Zielfunktion

$20x_1 + 36x_2 + 24x_3 \to \max$

NB:

$3x_1 + 7x_2 + 4x_3 \le 200$

$7x_1 + 10x_2 + 6x_3 \le 320$

$x_1, x_2, x_3 \ge 0$

Variablen: y_1, y_2 Preise für Material 1, 2.

Zielfunktion:

$200y_1 + 320y_2 \to \min$

NB:

$3y_1 + 7y_2 \ge 20$

$7y_1 + 10y_2 \ge 36$

$4y_1 + 6y_2 \ge 24$

$y_1, y_2 \ge 0$

Allgemein

(P)	(D)
$c^T x \to \max$	$b^T y \to \min$
$Ax \le b$	$A^T y \ge c$
$x \ge 0$	$y \ge 0$

Aufgabe 1

a) $2x_1 + x_2 - x_3 + 7x_4 \to \min$

y_1: $x_1 + 2x_2 + x_3 \ge -3$ → $y_1 \ge 0$

y_2: $3x_1 + x_2 - x_4 = 6$ → $y_2 \in \mathbb{R}$

y_3: $4x_1 + 3x_2 \le 3$ → $y_3 \le 0$

$x_1, x_2 \ge 0$ → NB $\le$

$x_3 \le 0$ → NB $\ge$

$-3y_1 + 6y_2 + 3y_3 \to \max$

$y_1 + 3y_2 + 4x_1 \le 2$

$2y_1 + y_2 + 3y_3 \le 1$

$y_1 \ge -1$

$-y_2 = 7$

$y_1 \ge 0$

$y_2 \in \mathbb{R}$

$y_3 \le 0$

Von Standardform (P) zu (D):

(c) Das LP in der Standardform lautet

$$
\begin{array}{lrrrrrrrl}
 & -2x_1 & -x_2 & +3x_3 & & & & \to & \max \\
y_1 = & x_1 & +2x_2 & +x_3 & & & & = & 8 \\
y_2 = & x_1 & +x_2 & +2x_3 & +x_4 & & & = & 12 \\
y_3 = & 2x_1 & +2x_2 & & & -x_5 & & = & 5 \\
y_4 = & & x_2 & & & & -x_6 & = & 2 \\
 & x_1, & x_2, & x_3, & x_4, & x_5, & x_6 & \ge & 0
\end{array}
$$

Das dazu duale LP ist gegeben durch

$$
\begin{array}{rrrrll}
8y_1 & +12y_2 & +5y_3 & +2y_4 & \to & \min \\
y_1 & +y_2 & +2y_3 & & \ge & -2 \\
2y_1 & +y_2 & +2y_3 & +y_4 & \ge & -1 \\
y_1 & +2y_2 & & & \ge & 3 \\
 & y_2 & & & \ge & 0 \\
 & & -y_3 & & \ge & 0 \\
 & & & -y_4 & \ge & 0
\end{array}
$$

Rebecca Nauli
naulireb@students.zhaw.ch

P \ D	Optimalität	Unbeschränktheit	Unzulässigkeit
Optimalität	ja	nein	nein
Unbeschränktheit	nein	nein	ja
Unzulässigkeit	nein	ja	ja

Starke Dualität (SD)

Schwache Dualität

Beispiel aus den Übungen ist möglich.

SD

Schwache Dualität

Schwache Dualität

Gegeben sei das primal-duale Paar (P)-(D) in kanonischer Form. Sei x^* zulässig in (P) und sei y^* zulässig in (D). Dann gilt:

$$c^T x^* \leq b^T y^*.$$

- x* zulässig in (P), y* zulässig in (D) & $c^T x^* = b^T y^*$ dann folgt dass
  - x* optimal in (P)
  - y* optimal in (D)

Starke Dualität

Starke Dualität

Wenn das primale Problem eine optimale Lösung besitzt, so hat auch das duale Problem eine optimale Lösung; die optimalen Zielfunktionswerte stimmen überein.

Komplementarität

Primal-duales Paar:

$$\begin{array}{rrcl} & c^T x & \to & \max \\ \text{(P2)} & Ax + s & = & b \\ & x, s & \geq & 0 \end{array} \qquad \begin{array}{rrcl} & b^T y & \to & \min \\ \text{(D2)} & A^T y - t & = & c \\ & y, t & \geq & 0 \end{array}$$

Satz über komplementären Schlupf

Sei (x^*, s^*) eine zulässige Lösung von (P2) und (y^*, t^*) eine zulässige Lösung von (D2). Dann sind diese zulässigen Lösungen optimal genau dann, wenn

$$\begin{array}{rcll} x_j^* t_j^* & = & 0 & \text{für alle } j \text{ und} \\ y_i^* s_i^* & = & 0 & \text{für alle } i. \end{array}$$

Beispiel Komplementarität

$24x_1 + 20x_2 + 9x_3 \to \max$

$8x_1 + 2x_2 - 3x_3 \leq 7$

$4x_1 + 5x_2 + 3x_3 \leq 17$

$x_1 + 3x_2 + 4x_3 \leq 16$

$x_1, x_2, x_3 \geq 0$

Wollen bestimmen, ob $x_1 = 2, x_2 = 0, x_3 = 3$ eine optimale Lösung ist.

Mit Schlupfvariablen haben wir:

$24x_1 + 20x_2 + 9x_3 \to \max$

$8x_1 + 2x_2 - 3x_3 + s_1 = 7$

$4x_1 + 5x_2 + 3x_3 + s_2 = 17$

$x_1 + 3x_2 + 4x_3 + s_3 = 16$

$x_1, x_2, x_3 \geq 0$

$s_1, s_2, s_3 \geq 0$

Rebecca Nauli
naulireb@students.zhaw.ch

Bestimmen s_1, s_2, s_3

$s_1 = 7 - 8x_1 - 2x_2 + 3x_3 = 7 - 16 + 9 = 0$

$s_2 = 17 - 4x_1 - 5x_2 - 3x_3 = 17 - 8 - 9 = 0$

$s_3 = 16 - x_1 - 3x_2 - 4x_3 = 16 - 2 - 12 = 2 > 0$

$\Rightarrow s_1, s_2, s_3 \geq 0 \Rightarrow$ es ist eine zulässige Lösung

Duales Problem lautet:

$$7y_1 + 17y_2 + 16y_3 \rightarrow \min$$
$$8y_1 + 4y_2 + y_3 - t_1 = 24$$
$$2y_1 + 5y_2 + 3y_3 - t_2 = 20$$
$$-3y_1 + 3y_2 + 4y_3 - t_3 = 9$$
$$y_1, y_2, y_3 \geq 0$$
$$t_1, t_2, t_3 \geq 0$$

Komplementaritätsbedingung:

$x_1 t_1 = 0$ $\quad y_1 s_1 = 0$

$x_2 t_2 = 0$ $\quad y_2 s_2 = 0$

$x_3 t_3 = 0$ $\quad y_3 s_3 = 0$

Wissen: $x_1 = 2, x_2 = 0, x_3 = 3$

$s_1 = 0, s_2 = 0, s_3 = 2$

$x_1 > 0 \Rightarrow t_1 = 0$

$x_2 = 0$ keine Info über t_2

$x_3 > 0 \Rightarrow t_3 = 0$

$s_3 > 0 \Rightarrow y_3 = 0$

$$7y_1 + 17y_2 + 16y_3 \rightarrow \min$$
$$8y_1 + 4y_2 + y_3 - t_1 = 24$$
$$2y_1 + 5y_2 + 3y_3 - t_2 = 20$$
$$-3y_1 + 3y_2 + 4y_3 - t_3 = 9$$
$$y_1, y_2, y_3 \geq 0$$
$$t_1, t_2, t_3 \geq 0$$

Ist es möglich folgendes zu erfüllen?

$$7y_1 + 17y_2 \rightarrow \min$$
$$8y_1 + 4y_2 = 24 \leftarrow$$
$$2y_1 + 5y_2 - t_2 = 20$$
$$-3y_1 + 3y_2 = 9 \leftarrow$$
$$y_1, y_2 \geq 0$$
$$t_2 \geq 0$$

(1) $\Rightarrow 2y_1 + y_2 = 6 \Rightarrow y_2 = 6 - 2y_1$

(3) $\Rightarrow -y_1 + y_2 = 3$ ← einsetzen

$-y_1 + (6 - 2y_1) = 3$

$-3y_1 = -3$

$y_1 = 1$

$y_2 = 6 - 2y_1 = 4$

(2) $\Rightarrow t_2 = 2y_1 + 5y_2 - 20 = 2 + 20 - 20 = 2$

$y_1 = 1, y_2 = 4, y_3 = 0$

$t_1 = 0, t_2 = 2, t_3 = 0$

Ja, $x_1 = 2, x_2 = 0, x_3 = 3$ ist eine optimale Lösung.

Tweedback Quiz

1. Sei ein zul. LP gegeben. Hat das duale Problem immer eine opt. Lösung?
→ Nein

2. Sei ein LP in Max. form gegeben mit Optimalwert = 100
wie gross ist der ZFW einer zulässigen Lösung im entsprechenden dual Problem.
(D) $b^T y \rightarrow \min$
(P) $c^T x \rightarrow \max$
→ ≥ 100 (untere Schranke) → $c^T x \leq b^T y \Rightarrow b^T y \geq 100$
→ schwache Dualität

3. Die LP-Relaxierung eines ganzzahligen Maximierungs-LP hat den Optimalwert = 80. Wie gross ist der Optimalwert des ganzzahligen LPs?
→ nicht genug Information → wir wissen nicht ob das ILP zulässig ist oder nicht.
≤ 80 wenn Information das LP zulässig
LP kann optimale Lösung haben aber ILP nicht.
→ sich Fragen ob man alle Information hat um Frage zu beantworten.

Rebecca Nauli
naulireb@students.zhaw.ch

Ganzzahlige lineare Programme

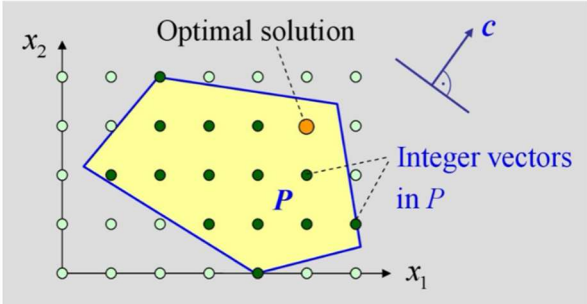
Lineares Programm

$$\max\{c^T x | x \in P\} \, mit \; P = \{x \in \mathbb{R}^n | Ax \leq b\}$$

Ganzzahliges lineares Programm = Integer Linear Program (ILP)

- Ganzzahlige Variablen modellieren typischerweise Anzahl von unteilbaren Objekten

$$\max\{c^T x | x \in P \cap \mathbb{Z}^n\} \, mit \; P = \{x \in \mathbb{R}^n | Ax \leq b\}$$



Binäres lineares Programm = Binary Linear Program

- Ja/Nein-Entscheidungen durch binäre Variablen modelliert

$$\max\{c^T x | x \in P \cap \{0,1\}^n\} \, mit \; P = \{x \in \mathbb{R}^n | Ax \leq b\}$$

Gemischt-ganzzahliges lineares Programm = Mixed Integer Linear Program

Mindestens eine Variable darf nur ganzzahlige Werte annehmen
Beispiel:

- x_1: $ganzzahlig$
- x_2: $kann\ reelle\ Werte\ annehmen$

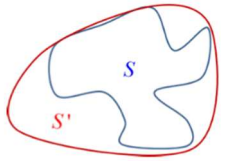
- Komplexität: ganzzahlige lineare Programme gehören zur am schwierigsten lösbaren Problemklasse
  - NP: vollständig: nur exponentielle Algorithmen bekannt
  - Im Gegensatz zu «klassischen» LPs, die polynomiell lösbar sind

Relaxierung von Optimierungsproblemen

Relaxierung des Optimierungsproblems

$$max\{f(x) | x \in S\}$$

durch Vergrösserung des Lösungsraum: ersetze S durch S', sodass $S \subseteq S'$ gilt.



Proposition

Sei $S \subseteq S'$. Dann gilt:
$\max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in S\} \leq \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in S'\}$

Definition: Relaxierung eines Optimierungsproblems

Es sei das Optimierungsproblem $\Pi : \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in S\}$ gegeben. Das Optimierungsproblem $\Pi' : \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in S'\}$ ist eine **Relaxierung von Π** falls gilt $S \subseteq S'$.

LP-Relaxierung

ILP: $\prod: \max\{c^T x | \; Ax \leq b, x \in \mathbb{Z}^n\} \rightarrow S$

LP-Relaxierung: $\prod^{LP}: \max\{c^T x | \; Ax \leq b\} \rightarrow S'$

Binäres LP: $\prod: \max\{c^T x | Ax \leq b, x \in \{0,1\}^n\} \rightarrow S$

LP-Relaxierung: $\prod^{LP}: \max\{c^T x | Ax \leq b, 0 \leq x \leq 1\} \rightarrow S'$

➔ Im Falle einer Maximierung liefert die LP-Relaxierung eine obere Schranke für den optimalen Zielfunktionswert des ILPs.

Rebecca Nauli
naulireb@students.zhaw.ch

Beispiel: Knapsack-Problem

Eine Firma möchte b Franken in eine Auswahl von n Projekten investieren. Jedes Projekt i $(i = 1, \ldots, n)$ benötigt eine Investition in der Höhe a_i Franken und verspricht einen Gewinn von c_i Franken. Wie soll investiert werden, damit der Gewinn maximal wird?

Projekt	Investition	Gewinn
1	a_1	c_1
2	a_2	c_2
3	a_3	c_3
4	a_4	c_4
⋮	⋮	⋮
n	a_n	c_n

Variabeln:

n: Anzahl Projekte

x_i $\begin{cases} 0 & \text{nicht investieren} \\ 1 & \text{investiere in Projekt } i \end{cases}$

$x \in \{0,1\}^n$

Zielfunktion

$$\sum_{i=1}^{n} c_i \cdot x_i \rightarrow \max$$

NB:

$$\sum_{i=1}^{n} a_i \cdot x_i \leq b$$

Für nicht ganzzahlige Lösung: $0 \leq x_i \leq 1$

Bsp. $b = 300$

	2	4	3	1
i	1	2	3	4
a_i	100	50	200	100
c_i	10	2	15	20
$\frac{c_i}{a_i}$	0.1	0.04	0.075	0.2
	②	④	③	①

Vorgang: zuerst 100 in ①
dann 100 in ②
dann 100 in ③ ist aber 200

$\Rightarrow$ $x_1 = 1$, $x_2 = 1$, $x_3 = 0.5$, $x_4 = 0$

Gewinn: $20 + 10 + 7.5 = 37.5$ (nur wenn ich Teilprojekte erlaube)

Beispiel: Plant Location Problem

Ein Zulieferer versorgt n Städte. In jeder Stadt kann ein Lager betrieben werden. Für jede Stadt i betragen die jährlichen Fixkosten für die Betreibung des Lagers f_i; wenn von diesem Lager ein Kunde in der Stadt j versorgt wird, kostet das c_{ij}. Wo sollen Lager betrieben werden, damit die Kosten minimal werden?

Stadt	Fixkosten	Lieferungskosten
i	f_i	c_{ij}

Variabeln:

n: Anzahl Städte

f_i: Fixkosten pro Lager in Stadt i;

c_{ij}: Kosten für Lieferung an Kunde j in Stadt i;

$y_i \in \{0,1\}^n$ $\quad y_i \begin{cases} 1 & \text{Lager in Stadt } i \\ 0 & \text{sonst} \end{cases}$

$x_{ij} \in \{0,1\}^n$ $\quad x_{ij} \begin{cases} 1 & \text{Zulieferer in Stadt } i \text{ beliefert Kunde in Stadt } j \\ 0 & \text{sonst} \end{cases}$

Zielfunktion:

$$\sum_{i=1}^{n} f_i \cdot y_i + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \rightarrow \min$$

NB:

- $\sum_{i=1}^{n} x_{ij} = 1$ (jeder Kunde in Stadt j wird aus genau 1 Lager beliefert) — für jedes j → mehrere NB → n Nebenbedingungen
- $x_{ij} \leq y_i$ Lager ist geöffnet wenn x_{ij} beliefert wird
  Wenn $x_{ij} = 1$, dann $y_i = 1$
  Wenn $y_i = 0$ (nicht offen), dann $x_{ij} = 0$ (niemand kann beliefert werden aus diesem Lager)
  Anzahl NB: $n \cdot n = n^2$

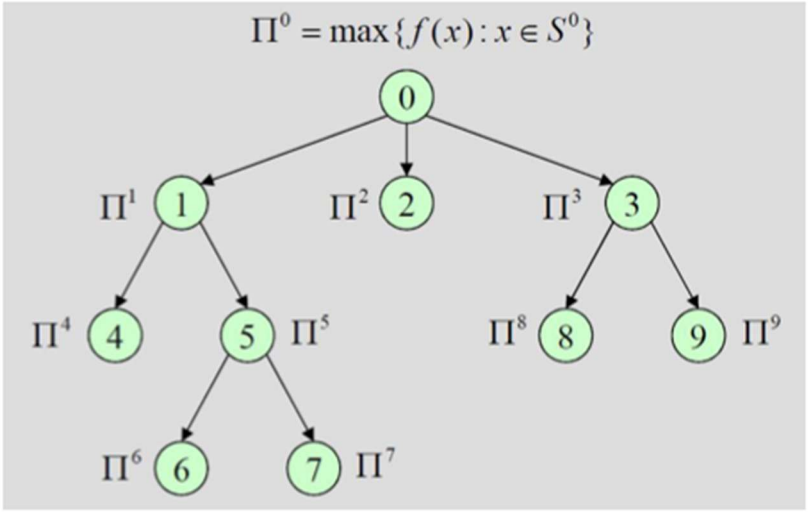
Rebecca Nauli
naulireb@students.zhaw.ch

Branch-and-Bound Methode

Branch-and-Bound (B&B) ist eine allgemeine Methode zum Lösen von Optimierungsproblemen (nicht auf ILPs beschränkt).

Grober Ablauf (bei Maximierungen)

1. Lösungsraum in kleinere Mengen iterativ aufteilen → Branch
2. In jedem Subproblem
   a. Berechne obere Schranke z.B. mit Relaxierung → Bound
   b. Bestimme zulässige Lösung = untere Schrank → Bound
3. Nutze diese Schrankeninfo zum «wegschneiden» von gewissen Subproblemen
   a. Z.B. falls die obere Schranke von Knoten 2 (x^U/z^U) < als untere Schranke von Knoten 1 (x^L/z^L) → Knoten 2 wegschneiden



Detaillierter Ablauf

In jedem Subproblem (Knoten) suche eine optimale Lösung der LP-Relaxierung.

1. Aufstellung des LPs mit ZFW & NB
2. LP-Relaxierung aufstellen
3. Erster Knoten ausrechnen
   a. x^U & z^U $aufstellen\ und\ ausrechnen$ → beginne mit ganzzahligen Zahlen von links nach rechts
   b. x^U & z^U aufstellen und ausrechnen → nicht ganzzahlige Zahl abrunden → erster Kandidat für optimale Lösung
4. Danach wird dasjenige x, welches gebrochen ist auf- bzw. abgerundet. Dies ergibt 2 neue Knoten. Begonnen wird mit dem Knoten der aufgerundet wird.
5. Die Knoten werden wieder nach dem vorherigen Prinzip (Schritt 3) ausgerechnet. → Zuerst fixierte Komponente, dann wieder links nach rechts.
   a. Nicht-Ganzzahlig: Falls $z_2^U \leq z_1^L$ → Knoten fertig, nicht weiter verfolgen
   b. Nicht-Ganzzahlig & Kandidat: $z^u < z_{Kandidat}^{U/L}$ → Knoten fertig
   c. Nicht-Ganzzahlig & keine der obenen Bedingungen → weiter branchen
   d. Ganzzahlig: Falls $z_2^L = z_2^U$ → Lösungskandidat für Optimum
6. So lange weiter branchen bis alle Knoten fertig sind.

Rebecca Nauli
naulireb@students.zhaw.ch

Beispiel: Knapsack

Knapsack Beispiel

	Vase	Uhr	Radio
Wert	70	50	60
Gewicht	1	2	3
Wert/Gewicht	70	25	20

⇒ 2^3 Möglichkeiten (Gegenstände)

Greedy – Ansatz → löst LP-Relaxierung

Rucksack – 5 kg

Sortierung ist ok.

Vase $x_1 \in \{0,1\}$
Uhr $x_2 \in \{0,1\}$
Radio $x_3 \in \{0,1\}$

- $70x_1 + 50x_2 + 60x_3 \rightarrow max$
- $x_1 + 2x_2 + 3x_3 \leq 5$
- $x_1, x_2, x_3 \in \{0,1\}$

→ Greedy-Problem funktioniert nur wenn Problem Ansatz mit einer NB.

LP-Rel: $0 \leq x_1, x_2, x_3 \leq 1$

beginne von links und gehe durch
NB, x^u = upperbound, geht nur Knapsack
x_L = lower bound

$x^u = (1, 1, \frac{2}{3})$
$z^u = 70 + 50 + \frac{2}{3} \cdot 60 = 160$
$x_L = (1,1,0)$
$z_L = 120$

$x_3 \leq \lfloor 2/3 \rfloor$ | $x_3 \geq \lceil 2/3 \rceil$

$x_3 = 0$

$x^u = (1,1,0)$
$z^u = 120$
(zu klein, erledigt)
da schon 120

$x_3 = 1$

$x^u = (1, \frac{1}{2}, 1)$
$z^u = 70 + 25 + 60 = 155$
~~$x^u = (1,0,1)$~~
~~$z^u = (130)$~~

zuerst fixierte Komponente dann wieder von links nach rechts

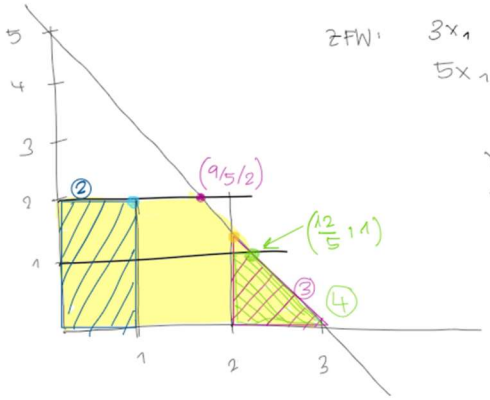
$x_2 = 0$ | $x_2 = 1$

$x^u = (1,0,1)$
$z^u = 130$
(erledigt)
da wir schon 130 erhalten haben.

$x^u = (0,1,1)$
$z^u = 110$
(erledigt)

Beispiel Branch & Bound (B&B)

Beispiel – B&B



ZFW: $3x_1 + 5x_2 \rightarrow max$
$5x_1 + 3x_2 \leq 15$
$x_2 \leq 2$
$x_1, x_2 \geq 0$
$x_1, x_2 \in \mathbb{Z}$

① $x^u = (\frac{9}{5}, 2)$
$z^u = 3 \cdot \frac{9}{5} + 10 = \frac{27}{5} + 10 = 15.4$

$x_1 \leq 1$ | $x_1 \geq 2$ → auf und abrunden von $\frac{9}{5}$

② $x^u = (1, 2)$
$z^u = 13$
fertig

③ $x^u = (2, \frac{5}{3})$
$z^u = 6 + 5 \cdot \frac{5}{3} = 14\frac{1}{3}$

$x_2 \leq 1$ | $x_2 \geq 2$

④ $x^u = (12/5, 1)$
$z^u = 3 \cdot \frac{12}{5} + 5 =$
12.2
fertig
↳ 12.2 < 13
Knoten uninteressant

⑤ unzulässig

Rebecca Nauli
naulireb@students.zhaw.ch

Optimierungen in Graphen

Optimale Wege in allgemeinen Graphen

Definitionen

Ungerichteter Graph

Graph: $G = (V, E)$

Knotenmenge: V

Kantenmenge: E

Kanten: $e \in E$

Gewicht: d_e

Grad:

Gerichteter Graph

Graph: $G = (V, E)$

Knotenmenge: V

Bogenmenge: E

Bogen: $e \in E$

Gewicht: d_e

Grad:

Übersicht über die wichtigsten Weg-Algorithmen

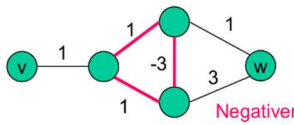
Ziel: kürzester oder längster Weg finden

➔ Länge eines Weges = Summe der Längen der Kanten

➔ Voraussetzung optimaler Weg:

- Kürzester Weg: keine negativen Zyklen

Beispiel:



Länge eines kürzesten Weges von v nach w ist nicht definiert (-∞)

Negativer Zyklus der Länge -1

- Längste Wege: keine positiven Zyklen

"One-To-All":	*Kürzeste Wege:*	*Längste Wege:*
a) Keine Zyklen Gewichte beliebig	**Algorithmus (Top. Sortierung)** Aufwand: $O(m)$	**Algorithmus (Top. Sortierung)** Aufwand: $O(m)$
b1) Zyklen erlaubt Gewichte ≥ 0	**Algorithmus von Dijkstra** Aufwand: $O(n^2)$	(Problem nicht definiert, falls positive Zyklen vorhanden)
b2) Zyklen erlaubt Gewichte ≤ 0 Bem: Selten in der Praxis	(Problem nicht definiert, falls negative Zyklen vorhanden)	**Algorithmus von Dijkstra** Aufwand: $O(n^2)$ Bem: Gewichte invertieren und kürzesten Weg suchen.
c) Zyklen erlaubt Gewichte beliebig	**Algorithmus von Bellman-Ford** Aufwand: $O(n \cdot m)$ Bem: Erkennt negative Zyklen.	**Algorithmus von Bellman-Ford** Aufwand: $O(n \cdot m)$ Bem: Erkennt positive Zyklen.
"All-To-All":	***Kürzeste Wege:***	***Längste Wege:***
d) Zyklen erlaubt Gewichte beliebig	**Algorithmus von Floyd-Warshall** Aufwand: $O(n^3)$ Bem: Erkennt negative Zyklen.	**Algorithmus von Floyd-Warshall** Aufwand: $O(n^3)$ Bem: Erkennt positive Zyklen.

Beispiel für Graphenalgorithmen:

- Problemgrösse: Anzahl Knoten $n=100$, Anzahl Kanten $m=1000$.

Polynomialer Aufwand (d.h. „effizient lösbar"):

- $O(m)$: Anzahl Rechenschritte in Grössenordung $m=1000$
- $O(n^2)$: Anzahl Rechenschritte in Grössenordnung $n^2=10'000$
- $O(n \cdot m)$: Anzahl Rechenschritte in Grössenordnung $n \cdot m = 100'000$
- $O(n^3)$: Anzahl Rechenschritte in Grössenordnung $n^3=1'000'000$

Exponentieller Aufwand (d.h. „nicht effizient lösbar"):

- $O(10^n)$: Anzahl Rechenschritte in Grössenordnung $10^n=10^{100}$!!!

Rebecca Nauli
naulireb@students.zhaw.ch

Algorithmus von Dijkstra

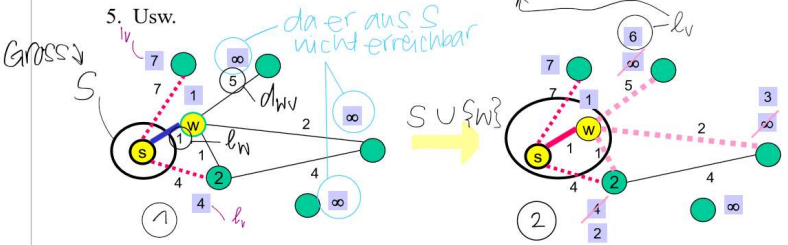
Problemvariante: „one-to-all“

Voraussetzung: Gewichte $d_e \geq 0$

Vorgehen

Vorgehen:

1. Beginne mit $S = \{s\}$. Berechne „die kürzesten Wege von s aus S hinaus“ (d.h. kürzeste Wege mit Zwischenknoten aus S, Längen l_v).
2. Wähle den „kürzesten Weg hinaus“ (d.h. den „nähesten“ Knoten w ausserhalb S)
3. Füge w zu S hinzu: $S := S \cup \{w\}$.
4. Updatiere von w aus die „kürzesten Wege von s aus S hinaus“:
   Für alle $v \in V - S$ mit $(w,v) \in E$: Wenn $l_v > l_w + d_{wv}$ dann $l_v := l_w + d_{wv}$
5. Usw.



Initialisierung:

$S := \{s\}, l_s := 0, (pred_s := s)$

FOR ALL $v \in V - \{s\}$ DO

  IF $(s,v) \in E$ THEN $l_v := d_{sv}$ $(pred_v := s)$

  ELSE $l_v := \infty$ $(pred_v := \emptyset)$

Iterationen:

WHILE $S \neq V$ DO

  Bestimme $w \in V - S$ so dass $l_w = \min\{l_v : v \in V - S\}$;

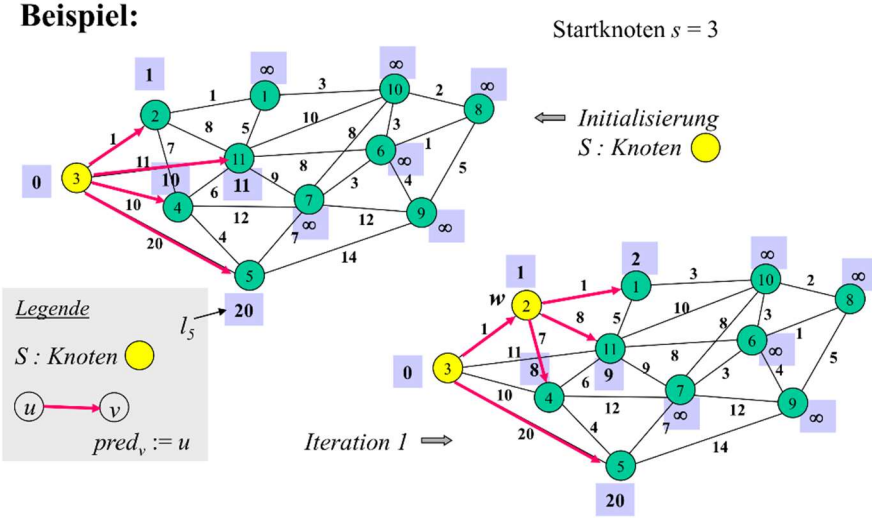
  $S := S \cup \{w\}$

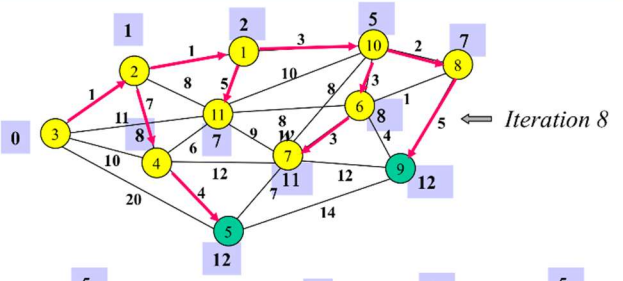
  FOR ALL $v \in V - S$ WITH $(w,v) \in E$ DO

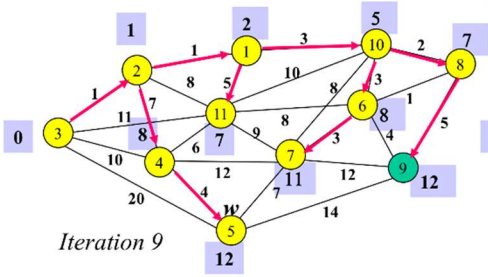
    IF $l_v > l_w + d_{wv}$ THEN $l_v := l_w + d_{wv}$ $(pred_v := w)$

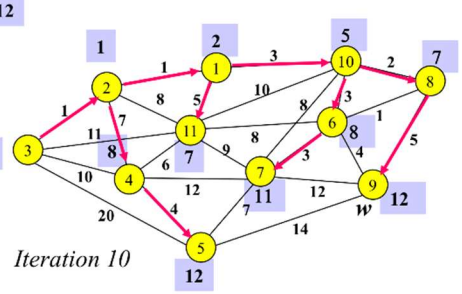
Beispiel Algorithmus von Dijkstra

Beispiel:









Rebecca Nauli
naulireb@students.zhaw.ch

Algorithmus von Floyd-Warshall

Problemvariante: „all-to-all“

Voraussetzung: keine

Algorithmus von Floyd-Warshall:

Initialisierung:

- Numeriere die Knoten von 1 bis n.
- Für jedes Paar *(i, j)*, $1 \le i, j \le n$, setze:

$$l_{ij}^0 := \begin{cases} d_{ij} & falls \ (i,j) \in E \\ \infty & falls \ (i,j) \notin E \ und \ i \neq j \\ 0 & falls \ i = j \end{cases} \qquad v_{ij}^0 := \begin{cases} i & falls \ (i,j) \in E \\ \varnothing & falls \ (i,j) \notin E \ und \ i \neq j \\ i & falls \ i = j \end{cases}$$

Verbindung nicht vorhanden

Iterationen:

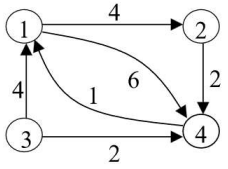
- Für alle Zwischenknoten $k = 1,2,...n$, führe aus:
  - Für jedes Paar (i, j), $1 \le i, j \le n$, mit $i \neq k, j \neq k$, setze:

$$l_{ij}^k := \begin{cases} l_{ij}^{k-1} & falls \ l_{ik}^{k-1} + l_{kj}^{k-1} \ge l_{ij}^{k-1} \\ l_{ik}^{k-1} + l_{kj}^{k-1} & sonst \end{cases} \qquad v_{ij}^k := \begin{cases} v_{ij}^{k-1} & falls \ l_{ik}^{k-1} + l_{kj}^{k-1} \ge l_{ij}^{k-1} \\ v_{kj}^{k-1} & sonst \end{cases}$$

Vergleich — Vergleich

zusammen gesetzter Weg ≥ bisheriger Weg

Beispiel Algorithmus von Floyd-Warshall



	1	2	3	4
1		4		6
2				2
3	4			2
4	1			

Distanzmatrix (d_{ij})

Initialisierung:

Weg 0 — Matrix (l_{ij}^0, v_{ij}^0)

Länge von 2-4 in der Iteration 0

(l_{24}^0, v_{24}^0)

0	1	2	3	4
1	0 / 1	4 / 1	∞	6 / 1
2	∞ / 1	0 / 2	∞	2 / 2
3	4 / 3	∞	0 / 3	2 / 3
4	1 / 4	∞	∞	0 / 4

Vorgänger von 4 auf dem KW von 2 – 4

l_{ij}^0 Länge eines KW von i nach j

v_{ij}^0 Vorgänger von j auf KW von i nach j

Erlaubte Zwischenknoten { }

Matrix (l_{ij}^0, v_{ij}^0)

0	1	2	3	4
1	0 / 1	4 / 1	∞	6 / 1
2	∞ / 1	0 / 2	∞	2 / 2
3	4 / 3	∞	0 / 3	2 / 3
4	1 / 4	∞	∞	0 / 4

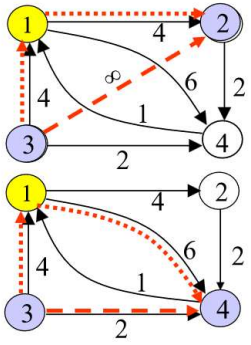
(l_{32}^0, v_{32}^0) (l_{34}^0, v_{34}^0)

$k = 1$: Matrix (l_{ij}^1, v_{ij}^1)

1	1	2	3	4
1	0 / 1	4 / 1	∞	6 / 1
2	∞ / 2	0 / 2	∞	2 / 2
3	4 / 3	8 / 1	0 / 3	2 / 3
4	1 / 4	5 / 1	∞	0 / 4

(l_{32}^1, v_{32}^1) (l_{34}^1, v_{34}^1)

2 neue Verbindungen



$l_{32}^1 := l_{31}^0 + l_{12}^0 = 4 + 4 = 8 \ \ da \ \ l_{31}^0 + l_{12}^0 = 8 < l_{32}^0 = \infty$

$v_{32}^1 := v_{12}^0 = 1 \ \ da \ \ l_{31}^0 + l_{12}^0 < l_{32}^0$

$l_{34}^1 := l_{34}^0 = 2 \ \ da \ \ l_{31}^0 + l_{14}^0 = 4 + 6 \ge l_{34}^0 = 2$

$v_{34}^1 := v_{34}^0 = 3 \ \ da \ \ l_{31}^0 + l_{14}^0 \ge l_{34}^0$

k=1: Erlaubte Zwischenknoten {1}

l_{ij}^1 Länge eines KW von i nach j

v_{ij}^1 Vorgänger von j auf KW

Identifikation eines kürzesten Weges in der Resultat-Matrix (l_{ij}^4, v_{ij}^4)

Kürzester Weg von 2 nach 1:

KW hat die Länge 3.

- Der Vorgänger von **1** (Endknoten) (KW von 2 nach 1) ist **4**;
- Der Vorgänger von **4** (KW von 2 nach 4) ist **2**.
- **2** ist der Startknoten: Stop.

⇒ *Kürzester Weg: 2 → 4 → 1*

4	1	2	3	4
1	0 / 1	4 / 1	∞	6 / 1
2	3 / 4	0 / 2	∞	2 / 2
3	3 / 4	7 / 1	0 / 3	2 / 3
4	1 / 4	5 / 1	∞	0 / 4

Rebecca Nauli
naulireb@students.zhaw.ch

Optimale Wege in azyklischen Graphen

Topologische Sortierung

Problemvariante: «one-to-all»

Voraussetzung: keine Zyklen

Algorithmus: Längste Wege in azyklischen Graphen

Initialisierung (Annahme: s hat keine Vorgänger):
Startknoten
$S := \{s\}$, $l_s := 0$, $(pred_s := \emptyset)$
FOR ALL $v \in V - s$ DO $l_v := -\infty$ $(pred_v := \emptyset)$
↓nicht abgearbeitete Knoten
Iterationen:
WHILE $S \neq V$ DO
Bestimme $v \in V - S$ so dass v keine Vorgänger hat in $G[V-S]$.
$S := S \cup \{v\}$
Setze $l_v = max\{l_u + d_{uv}: u \text{ ist Vorgänger von } v\}$ $(pred_v := u^*)$

Bemerkungen:

- $G[V-S]$ ist der resultierende Graph, wenn die Knoten von S entfernt werden.
- u^* ist derjenige Vorgänger, welcher das Maximum ergibt.
- Beachte: Wegen Topol. Reihenfolge sind alle l_u schon berechnet für $u \in S$.
- Zur Annahme: Falls s Vorgänger hat in G, entferne zuerst alle „Vorfahren" von s.

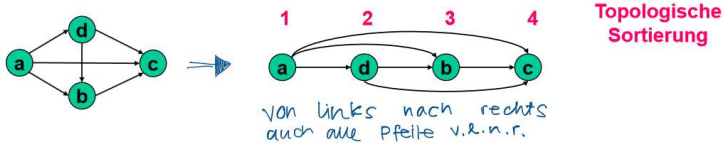
Vorgehen

1. Topologische Sortierung – Knoten 1
2. Alle Vorgänger vergleichen
3. Wege zusammenzählen
    a. Längster/Kürzester Weg bleibt stehen
4. Topologische Sortierung – Knoten 2

Topologische Sortierung:

- Jeder azyklische Graph hat eine ***Topologische Sortierung***:
  Die Knoten können so von *1* bis *n* numeriert werden, dass gilt:

Bogen
Wenn $(i,j) \in E$, dann ist $i < j$.

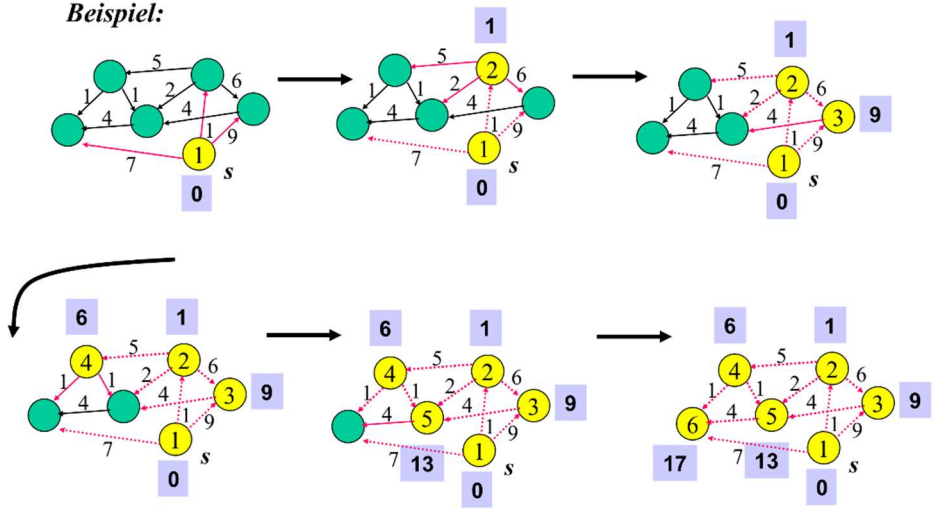


Prinzipielles Vorgehen zur Bestimmung einer Topologischen Sortierung:

1. Wähle einen Knoten v, der keine Vorgänger hat und gebe ihm die nächste Nummer. (Beachte: v existiert immer, wenn keine Zyklen vorhanden!)
2. Entferne Knoten v (mit seinen Bogen) und gehe zu Schritt 1.

Beispiel Algorithmus von Topologischer Sortierung

Beispiel:



Rebecca Nauli
naulireb@students.zhaw.ch

LP Formulierung für das Wegproblem

- *Problem:* Kürzester Weg von Quelle s nach Senke t in gerichtetem s-t-Graphen $G = (V, E)$ mit positiven Bogengewichten $c_{vw}>0$, $(v,w)\in E$

$$\min \sum_{(v,w)\in E} c_{vw}x_{vw} \qquad (0)$$

$$\sum_{w\in V:(v,w)\in E} x_{vw} - \sum_{u\in V:(u,v)\in E} x_{uv} = \begin{cases} 1 & v = s \\ 0 & v \in V - \{s,t\} \\ -1 & v = t \end{cases} \quad \text{"Flow Balance"} \qquad (1)$$

$$0 \le x_{vw} \le 1 \qquad (v,w) \in E \quad \text{"Binary Conditions"} \qquad (2)$$

$$x_{vw} \text{ integer} \qquad (v,w) \in E \quad \text{"Integer Conditions"} \qquad (3)$$

- Es gilt:
  - Jede Lösung $x\in R^{|E|}$ von (1)...(3) ist Inzidenzvektor einer Bogenmenge $S^x\in E$, welche in einen s-t-Pfad und eine Menge von Zyklen zerlegt werden kann.
  - Die "Umkehrung" gilt ebenfalls.
- Da alle Gewichte positiv sind, entspricht eine optimale Lösung des ILP (0)...(3) somit einem (minimalen) s-t-Pfad. (Warum?)

- Es kann gezeigt werden:
  - Das durch (1)...(2) definierte Polyeder P ist "ganzzahlig", d.h. alle seine Eckpunkte sind ganzzahlig:

$$P = \{x \in R^{|E|} : x \text{ erfüllt (1)...(2)}\}$$

  - Somit kann die Ganzzahligkeitsforderung (3) weggelassen werden und das Wegproblem kann als **LP (0)...(2)** gelöst werden.
  - Das Polyeder P ist also die **konvexe Hülle** aller Inzidenzvektoren $x\in\{0,1\}^{|E|}$, die einer Kantenmenge $S^x\in E$ entsprechen, welche in einen s-t-Pfad und eine Menge von Zyklen zerlegt werden kann.
- Das LP (0)...(2) gehört zur Klasse der sogenannten Flussprobleme.
  - Flussprobleme bilden einen wichtigen Bereich der Kombinatorischen Optimierung und haben viele praktische Anwendungen.

Optimale Zyklen – Traveling Salesman Problem (TSP)

Heuristiken

- Das TSP (d.h. die Bestimmung eines Hamiltonschen Zyklus minimaler Länge) ist - ausser in Spezialfällen - ein „schwieriges“ („NP-vollständiges“) Problem, d.h. es sind keine Algorithmen mit polynomialem Lösungsaufwand bekannt.
  - Sogar das Problem, festzustellen, ob ein Graph einen Hamiltonschen Zyklus besitzt, ist schon NP-vollständig!
- Aus diesem Grunde wurden schon früh zahlreiche Heuristiken für das TSP entwickelt.
- Einige klassische Heuristiken (mit gewissen allgemeinen Eigenschaften) werden im folgenden besprochen.

Konstruktive Heuristiken

Nearest Neighbor

Initialisierung:
Wähle einen Startknoten v.
$i(1) := v$; $S := \{v\}$; $k := 1$; *(* S sind die schon besuchten Knoten *)*

Iterationen:
WHILE $S \neq V$ *DO (* d.h. solange nicht alle Knoten besucht sind *)*
Ausgehend vom letzten besuchten Knoten $i(k)$, bestimme einen unbesuchten Knoten $j \in V\text{-}S$, welcher *am nähesten* von $i(k)$ ist, d.h.

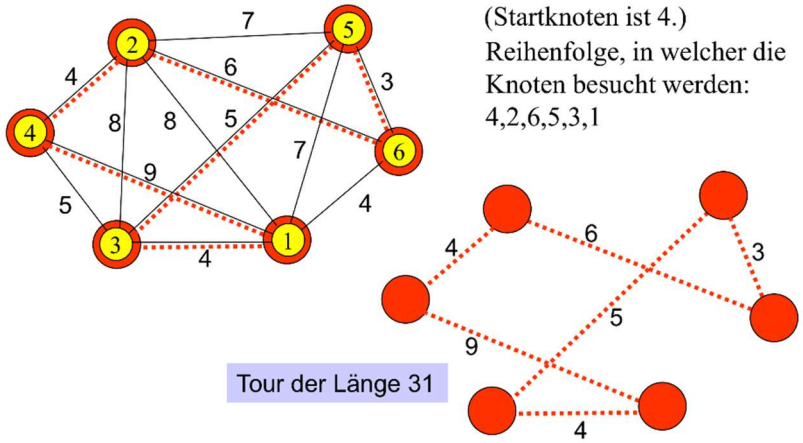
$$d_{i(k),j} = \min \{d_{i(k),v} : (i(k),v) \in E\}$$

$i(k+1) := j$; $S := S \cup \{j\}$; $k := k + 1$;

Finalisierung:
Schliesse den Weg $i(1), \ldots, i(n)$ mit der Kante $(i(n), i(1))$ zu einer Tour.

Rebecca Nauli
naulireb@students.zhaw.ch

Beispiel Nearest Neighbor



Insertion Heuristiken (konstruktive Heuristiken)

Initialisierung:
Wähle zwei Knoten v und w und starte mit der „Teiltour“ v, w, v.
(Wähle v und w beispielsweise so, dass d_{vw} minimal ist. *)*
$i(1) := v$; $i(2) := w$; $S := \{v, w\}$; $k := 2$;

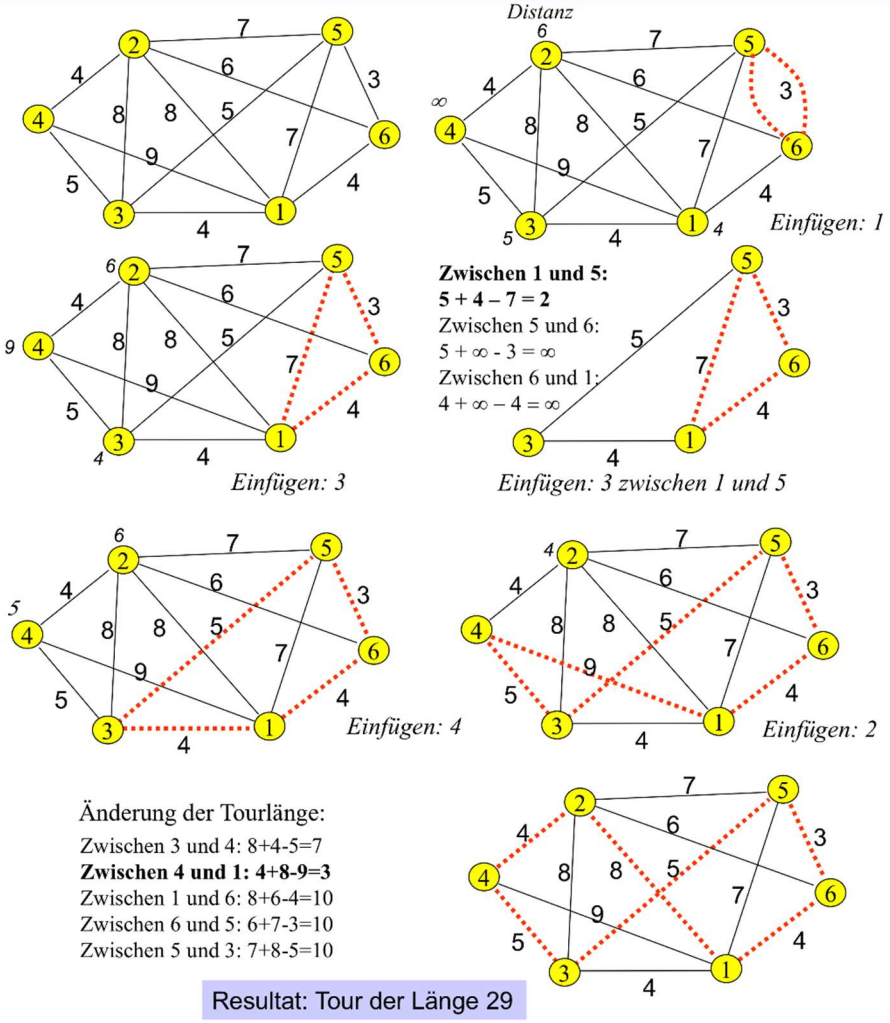
Iterationen:
WHILE $S \neq V$ DO
(a) Bestimme einen *geeigneten Knoten* $j \in V\text{-}S$ ausserhalb der Teiltour $i(1), ..., i(k), i(1)$.
(b) *Füge* den Knoten j an *geeigneter Stelle* in die Teiltour *ein*, woraus eine neue Teiltour $i(1), ..., i(k+1), i(1)$ resultiert.
$S := S \cup \{j\}$; $k := k + 1$;

- Je nach Wahl des einzufügenden Knotens j in (a) und der Einfügestelle in (b) existieren verschiedene Versionen der „Insertion“ Heuristik.
- Definiere die *Distanz* eines Knotens $j \in V\text{-}S$ (ausserhalb der Teiltour) zur Teiltour $i(1), ..., i(k), i(1)$ als $\text{dist}(j) := \min \{d_{i(r),j} : r = 1,...,k\}$.

Nearest Insertion (B1)

(a) Wahl des einzufügenden Knotens j: *minimale Distanz* zur Teiltour, d.h. $\text{dist}(j) = \min \{\text{dist}(v): v \in V\text{-}S\}$
(b) Wahl der Einfügestelle: siehe unten.

Beispiel Nearest Insertion



Rebecca Nauli
naulireb@students.zhaw.ch

Farthest Insertion (B2)

(a) Wahl des einzufügenden Knotens j: *maximale Distanz* zur Teiltour, d.h.
$\text{dist}(j) = \max \{\text{dist}(v): v \in V\text{-}S\}$

(b) Wahl der Einfügestelle: siehe unten.

Wahl der Einfügestelle für die Versionen B1) und B2):

- Seien p und q zwei *aufeinanderfolgende* Knoten der Teiltour, d.h. $p = i(r)$ und $q = i(r+1)$ für ein gewisses $1 \le r \le k$.
- Wenn der Knoten j zwischen p und q in die Teiltour eingefügt wird, verändert sich die Länge der Teiltour um $\Delta(j,p,q) = d_{pj} + d_{jq} - d_{pq}$
- Der Knoten j wird an der *besten Stelle* p^*, q^* in die Teiltour eingefügt, d.h. so dass die Veränderung $\Delta(j,p,q)$ *minimal* ist:
$\Delta(j,p^*,q^*) = \min \{\Delta(j,p,q): p = i(r), q = i(r+1) \text{ für ein } 1 \le r \le k\}$

Cheapest Insertion (B3)

- Hier wird simultan darüber entschieden, (a) welcher Knoten j ausserhalb der Tour gewählt wird, und (b) an welcher Stelle dieser Knoten in die Tour eingefügt wird:
- Für jeden Knoten $v \in V\text{-}S$ ausserhalb der Tour wird die beste Einfügestelle p^*, q^* bestimmt (wie vorher):
$\Delta(v,p^*,q^*) = \min \{\Delta(v,p,q): p = i(r), q = i(r+1) \text{ für ein } 1 \le r \le k\}$
- Beachte: Die Länge der Tour ändert sich um
$d_v := \Delta(v,p^*,q^*) = d_{p^*v} + d_{vq^*} - d_{p^*q^*}$
wenn v an der besten Stelle p^*, q^* eingefügt wird.
- Es wird derjenige Knoten $j \in V\text{-}S$ zum Einfügen gewählt (und an optimaler Stelle eingefügt), welcher die Länge der Tour am wenigsten erhöht, d.h.
$d_j = \min \{d_v: v \in V\text{-}S\}$

Verbesserungs Heuristiken (Local Search)

- Es gibt viele verschiedene Varianten.
- Grundprinzip des „Local Search“ („*Meta-Heuristik*“):
  - Eine gegebene Lösung wird sukzessive verbessert, indem kleine („lokale“) Veränderungen (innerhalb einer bestimmten „Nachbarschaft“) vorgenommen werden.
  - Wenn keine Verbesserung mehr möglich ist, stoppt das Verfahren und die gegebene Lösung ist „lokal optimal“.
- Die bekanntesten „Local Search“ Verfahren für das TSP sind die „k-opt“ Heuristiken (i.a. $k = 2, 3$).
- Sie basieren auf dem Austausch von k Kanten in einer Tour durch k Kanten ausserhalb der Tour.

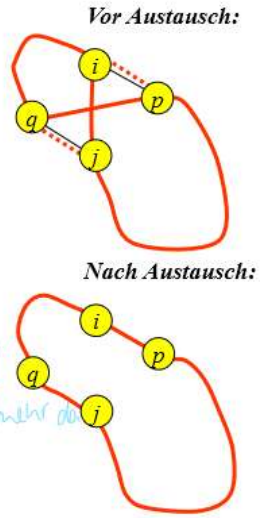
2-opt Heuristik

„2-opt“ Heuristik

Initialisierung: Beginne mit einer Tour T.

Iterationen: Solange nicht alle Austausche versucht wurden:

- Bestimme eine Kantenpaar (i,j) und (p,q) in der Tour (i, j, p, q verschieden), für welches der Austausch noch nicht versucht wurde.
- Vertausche versuchsweise die Kanten (i,j) und (p,q) durch die Kanten (i,p) und (j,q) ausserhalb der Tour und berechne die Änderung der Tourlänge: $\Delta = d_{ip} + d_{jq} - d_{ij} - d_{pq}$
- Wenn $\Delta < 0$, dann akzeptiere den Austausch (die Tour wurde „verbessert“). Benenne die neue Tour wiederum mit T und fahre fort.

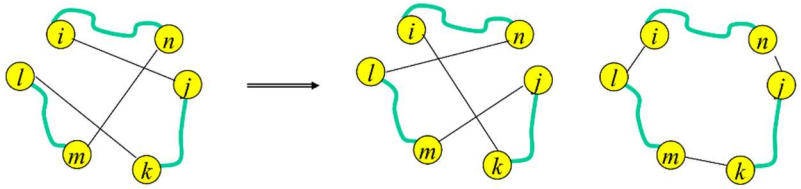


Rebecca Nauli
naulireb@students.zhaw.ch

3-opt Heuristik

- Lokale Veränderung: Anstatt $k = 2$ werden nun $k = 3$ Kanten in der Tour vertauscht mit 3 Kanten ausserhalb der Tour.
- Wenn drei Kanten (i,j), (k,l), (m,n) aus der Tour entfernt werden, entstehen 3 disjunkte Wege.
- Es gibt verschiedene Möglichkeiten, diese 3 Wege durch drei andere Kanten zu einer neuen Tour zusammenzufügen (für $k = 2$ nur eine!)

Beispiel (*2 Möglichkeiten sind aufgeführt*):



ILP Formulierung für das symmetrische TSP

- Von Dantzig, Fulkerson und Johnson (1954).
- *Problem:* Bestimme minimale Tour in ungerichtetem Graphen $G=(V, E)$

$$\min \sum_{(v,w)\in E} c_{vw} x_{vw}$$

$\sum_{w\in V:(v,w)\in E} x_{vw} = 2$	$v \in V$	"Degree Constraints"	(1)
$\sum_{v,w\in S:(v,w)\in E} x_{vw} \le \lvert S \rvert - 1$	$\varnothing \subset S \subset V$	"Subtour Constraints"	(2)
$0 \le x_{vw} \le 1$	$(v,w) \in E$	"Binary Conditions"	(3)
x_{vw} integer	$(v,w) \in E$	"Integer Conditions"	(4)

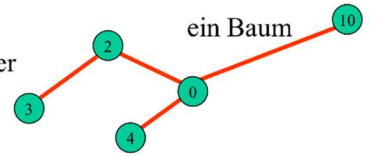
- Es gilt:
  - Jede Lösung $x \in R^{\lvert E \rvert}$ von (1)...(4) ist Inzidenzvektor einer Tour $S^x \in E$.
  - Jeder Inzidenzvektor $x^S \in R^{\lvert E \rvert}$ einer Tour $S \in E$ ist eine Lösung von (1)...(4).
- Es sind weitere "Cuts" (gültige Ungleichungen) für das TSP bekannt.

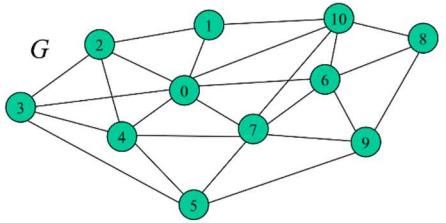
Optimale Bäume

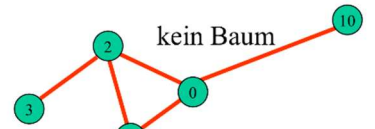
Bäume in Graphen

7.1.1 Definitionen

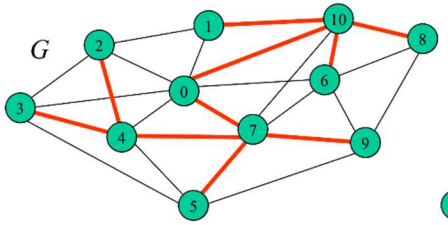
- Sei $G = (V, E)$ ein ungerichteter Graph.
  Ein *Baum* in G ist ein Subgraph von G, der
  (i) zusammenhängend ist und
  (ii) keine Zyklen enthält.

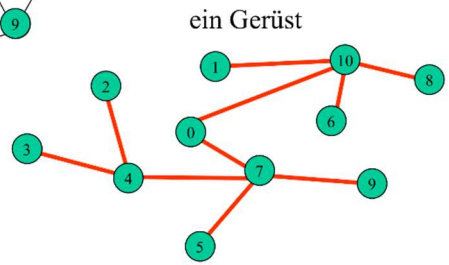






- Ein *Gerüst („spanning tree“)* in G ist ein Baum in G, welcher alle Knoten von G umfasst.





Rebecca Nauli
naulireb@students.zhaw.ch

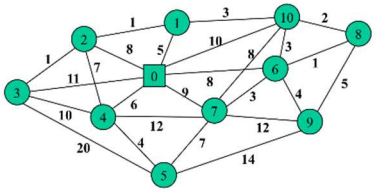
- Sei $G = (V, E)$ ein gewichteter Graph mit Kantengewichten (-längen) d_e, $e \in E$. Das *Gewicht (die Länge)* eines Baumes in G ist definiert als die Summe der Gewichte seiner Kanten.

Problem des minimalen Gerüsts:
Bestimme ein Gerüst minimalen Gewichts in G, d.h. bestimme ein Gerüst von G, dessen Länge minimal ist unter allen möglichen Gerüsten von G.

Beispiel

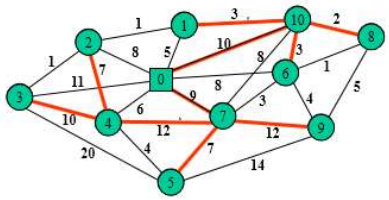
Beispiel: Planung eines Kabelnetzes

- Zehn Einfamilienhäuser einer Neubau-Siedlung sollen an einen Verzweigungsknoten des Kabelfernseh-Netzes angeschlossen werden.
- Im untenstehenden Graphen sind die Häuser als Knoten ○ und der Verzweigungsknoten als □ eingezeichnet.
- Die möglichen Punkt-zu-Punkt-Verbindungen sowie die zugehörigen Konstruktionskosten (in Fr. 1000.-) sind als gewichtete Kanten des Graphen dargestellt.



Wie soll das Kabelnetz konstruiert werden, damit die Gesamtkosten minimal sind?

- Damit jedes Haus (direkt oder indirekt) an die Verzweigungsdose angeschlossen ist, genügt es, das Kabelnetz so zu konstruieren, dass es ein Gerüst im Graphen darstellt:
  - Das Netz muss offensichtlich zusammenhängend sein.
  - Ein Netz minimaler Kosten enthält sicher keinen Zyklus. Andernfalls könnte eine beliebige Kante des Zyklus entfernt werden, und es wären weiterhin alle Häuser an die Verzweigungsdose angeschlossen.
- Ein Netz minimaler Kosten kann somit gefunden werden, indem ein Gerüst minimalen Gewichts im Graphen bestimmt wird.



11 Knoten n
10 Kanten da n−1

Ein Gerüst mit Gewicht
10+7+12+9+7+3+
10+12+3+2 = 75
(nicht minimal!)

Minimales Gerüst und Kruskal Algorithmus

- Das Problem des minimalen Gerüsts ist eines der einfachsten Probleme der Kombinatorischen Optimierung! Es kann mit dem Kruskal-Algorithmus (Typ „Greedy“) auf einfache Weise exakt gelöst werden.

Algorithmus von Kruskal :

- Sei n die Anzahl Knoten und m die Anzahl Kanten des Graphen. Ordne die Kanten nach aufsteigenden Gewichten: $e_1, e_2, e_3, \ldots e_m$ d.h. $d_{e_1} \le d_{e_2} \le d_{e_3} \le \ldots \le d_{e_m}$

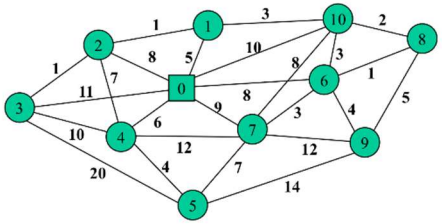
Initialisierung: Menge der gewählten Kanten $J := \emptyset$; $i := 0$;

Iterationen:

WHILE $|J| < n-1$ DO
  $i := i+1$
  Falls $\{e_i\} \cup J$ keinen Zyklus enthält, wähle e_i, d.h. $J := \{e_i\} \cup J$.

Übung:

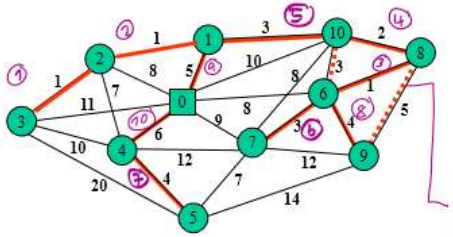
- Bestimme ein Gerüst minimalen Gewichts mit dem Kruskal-Algorithmus für das vorangehende Beispiel des Kabelnetzes.



Rebecca Nauli
naulireb@students.zhaw.ch

Lösung:

- Ein minimales Gerüst ist:



Minimale Kosten:
1+1+3+2+1+3+4+5+
6+4 = 30

falls diese Kante genommen würde Zyklus entstehen

7.1.3 Formulierungen als ILP und LP

- Die Probleme der Kombinatorischen Optimierung *mit linearer Zielfunktion* können als ganzahlig-lineare Programme (ILP: Integer Linear Program) formuliert werden.
- Bei gewissen Problemformulierungen kann die Ganzzahligkeits-Forderung weggelassen werden, und das Problem kann als Lineares Programm (LP) gelöst werden.
  (Eigenschaft der „natürlichen Ganzzahligkeit“: Alle Eckpunkte des den Lösungsraum beschreibenden Polyeders sind ganzzahlig.)
- Auch wenn ein kombinatorisches Problem leicht lösbar ist, kann es sein, dass eine Formulierung als ILP nicht evident ist, und dass es „bessere“ und „schlechtere“ ILP-Formulierungen gibt.
- Zur Illustration (ohne Beweise) im folgenden zwei verschiedene ILP-Formulierungen für das Problem des minimalen Gerüsts (wobei die zweite die Eigenschaft der natürlichen Ganzzahligkeit hat).

A) Eine korrekte (aber „schlechte“) ILP-Formulierung

- Führe für jede Kante (i,j) des Graphen eine binäre (0/1) Variable x_{ij} ein, mit der Bedeutung:

$$x_{ij} = \begin{cases} 1 & \text{wenn die Kante } (i,j) \text{ gewählt wird} \\ 0 & \text{sonst} \end{cases}$$

Bewirkt das keine Zyklen entstehen

$$\text{Minimiere} \sum_{\text{alle Kanten } (i,j)} d_{ij} x_{ij} \quad \text{unter den Bedingungen}$$

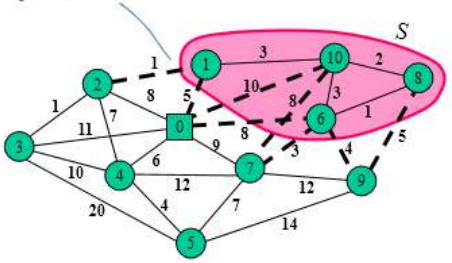
$$\sum_{\substack{\text{alleKanten } (i,j) \\ \text{mit } i \in S,\ j \notin S}} x_{ij} \geq 1 \quad \text{für jede nicht-leere Knotenmenge } S, \text{ welche den Knoten 0 (d.h. den Verzweigungsknoten) nicht enthält.}$$

$$x_{ij} = 0 \ \textit{oder}\ 1 \quad \text{für jede Kante } (i,j) \in E$$

- *Bedeutung der Restriktionen:* Garantieren, dass die Auswahl der Kanten einen zusammenhängenden Subgraphen bildet:

$$\sum_{\substack{\text{alleKanten } (i,j) \\ \text{mit } i \in S,\ j \notin S}} x_{ij} \geq 1$$

alle Kanten die über Grenze hinaus



Es muss ein Weg existieren von jedem Knoten von S zum Knoten 0 ⇒
Mindestens eine ausgewählte Kante (i,j) (mit $x_{ij} = 1$) muss aus der Menge S hinausführen.

Wie viele solche Restriktionen?
Die Anzahl Knotenteilmengen S wächst exponentiell mit der Anzahl Knoten.
Folglich gibt es eine exponentielle Anzahl Restriktionen!

Rebecca Nauli
naulireb@students.zhaw.ch

B) Eine „gute“ ILP- (bzw. LP-) Formulierung

Minimiere $\sum_{\text{alle Kanten } (i,j)} d_{ij} x_{ij}$ unter den Bedingungen:

$\sum_{\substack{\text{alle Kanten } (i,j) \\ \text{mit } i \in S_p,\ j \in S_q, q \neq p}} x_{ij} \geq k - 1$ Für jede Partition* von V in Mengen $S_1,\ldots, S_k$, $2 \leq k \leq n$

$(x_{ij} = 0 \text{ oder } 1$ für alle Kanten $(i,j) \in E)$ *(überflüssig!)*

$x_{ij} \geq 0$ für alle Kanten $(i,j) \in E$ *(genügt!)*

def:

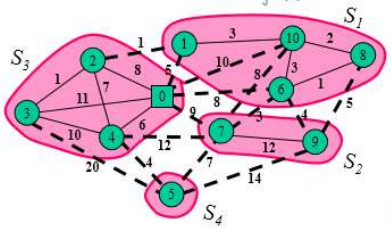
*) Eine Partition von V in Teilmengen $S_1,\ldots, S_k \subset V$ ist gegeben, falls $V = S_1 \cup \ldots \cup S_k$ ist und die Mengen $S_1,\ldots, S_k$ paarweise disjunkt sind, d.h. $S_p \cap S_q = \emptyset$, $1 \leq p < q \leq k$.

- *Bedeutung der Restriktionen:* Garantieren den Zusammenhang zwischen den Teilmengen jeder Partition:

$\sum_{\substack{\text{alle Kanten } (i,j) \\ \text{mit } i \in S_p,\ j \in S_q, q \neq p}} x_{ij} \geq k - 1$

Mindestens k-1 Kanten müssen ausgewählt werden (mit $x_{ij} = 1$) „zwischen“ den Mengen $S_1,\ldots, S_k$

i in einer Menge
j in anderer Menge



Beispiel:
$k = 4$: eine Partition $S_1,\ldots,S_4$:
Mindestens k-1= 3 Kanten müssen ausgewählt werden (mit $x_{ij} = 1$) unter den gestrichelten Kanten ----

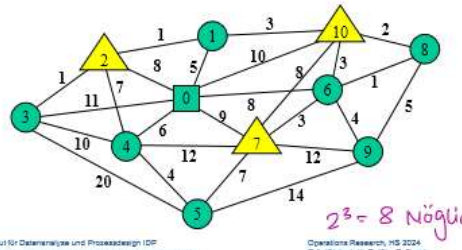
max. Partition = 11
da 11 Knoten
jeder Knoten kann eine Menge sein

Steiner-Bäume

7.2.1 Definitionen

Beispiel: Planung eines Kabelnetzes (modifiziert).

- Sieben Einfamilienhäuser müssen an eine Verzweigungsdose angeschlossen werden, wobei für die Verbindungen Zwischenknoten verwendet werden können (im Bsp: 0, 1, 2 oder 3 Zwischenknoten).
- Im untenstehen Graphen sind folgende Knoten eingezeichnet:
  - *Häuser* ○ *Verzweigungsdose* □ *mögliche Zwischenknoten* △



Wie soll das Kabelnetz eingerichtet werden, damit die Anschlusskosten minimal sind?

$2^3 = 8$ Möglichkeiten
Ja oder Nein
3 Optionen

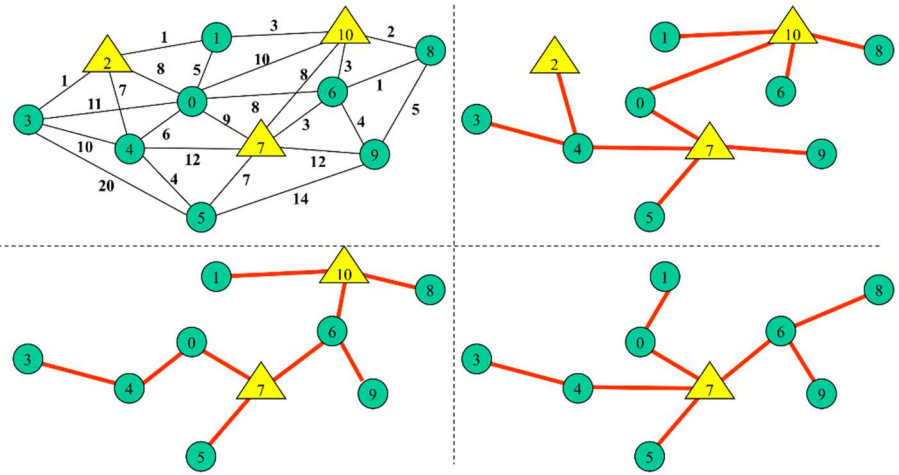
- Es geht darum, einen Baum zu bestimmen, welcher alle Häuser-Knoten sowie den Verzweigungsknoten enthält, d.h. die Knoten $N = \{0, 1, 3, 4, 5, 6, 8, 9\}$.
  Zusätzlich kann (muss aber nicht) der Baum auch Zwischenknoten aus der Menge V-$N = \{2,7,10\}$ benützen.
- Ein solcher Baum bildet einen *Steiner-Baum* im Graphen $G = (V, E)$ mit „obligatorischen“ Knoten N und „fakultativen“ Knoten V-N („Steiner-Knoten“ genannt).

Häuser = Zwischenknoten

- Das *Problem des minimalen Steiner-Baums* besteht darin, unter allen möglichen Steiner-Bäumen von G einen mit minimalem Gewicht zu finden.

Rebecca Nauli
naulireb@students.zhaw.ch

Einige Steiner-Bäume im Beispiel :



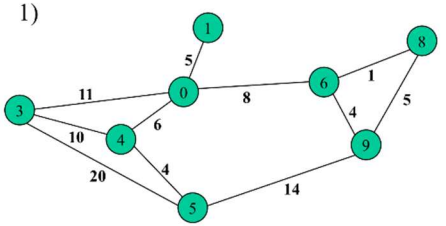
7.2.2 Ein einfacher Lösungsansatz (exakt, „naiv“)

- Alle möglichen Kombinationen (Auswahlen) von Steiner-Knoten prüfen.
- Für jede Auswahl von Steiner-Knoten wird das entsprechende Problem des minimalen Gerüsts gelöst (die Knotenmenge ist festgelegt).
- Unter diesen minimalen Gerüsten, wähle dasjenige mit kleinstem Gewicht.
- Anzahl Kombinationen (Auswahlen) von Steiner-Knoten wächst **exponentiell** mit der Anzahl Steiner-Knoten $|V\text{-}N|$: **$2^{|V\text{-}N|}$ Auswahlen**
- Folglich ist Ansatz nur möglich, wenn Anzahl Steiner-Knoten klein ist!
- ***Illustration im Beispiel mit Steiner-Knoten V-N = {2, 7, 10}:***
  Mögliche Kombinationen von Steiner-Knoten:

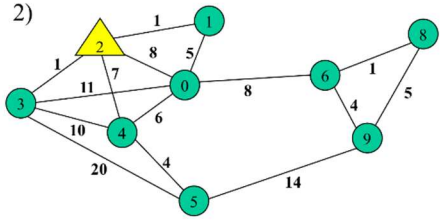
1)	∅	5)	{2,7}
2)	{2}	6)	{2,10}
3)	{7}	7)	{7,10}
4)	{10}	8)	{2,7,10}

- Die acht entsprechenden Probleme des minimalen Gerüsts:

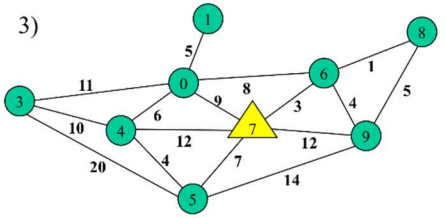
1)



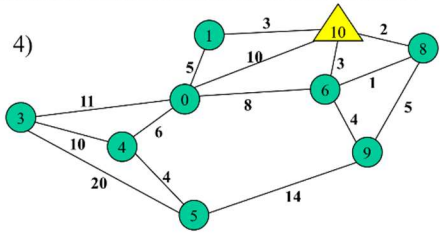
2)



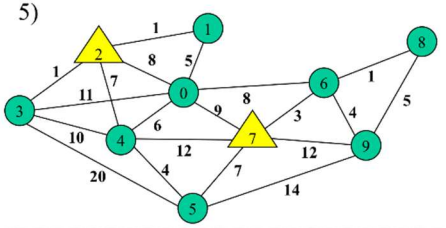
3)



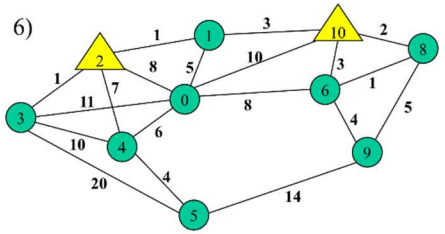
4)



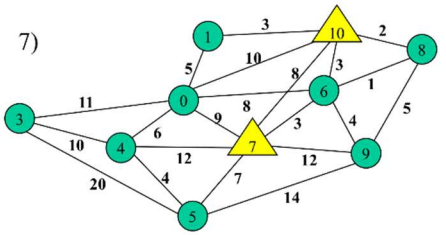
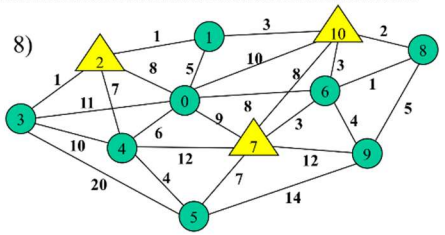
5)



6)



7)



8)