Lineare Optimierung

Allgemeines lineares Optimierungsproblem (LP)

$$A, \dots, F = Matrizen, \qquad a, \dots, e = Vektoren$$

$$\begin{array}{ll} max & c^T x + d^T y \\ u.d.n & Ax + By \geq a \\ & Cx + Dy \leq b \\ & Ex + Fy = e \\ & x > 0 \end{array}$$

Verschiedene Formen eines Linearen Optimierungsproblem

- Kanonische Form (Maximierung) $\max\ (c^T x) \quad u.d.N. \quad Ax \leq b,\ x \geq 0$
- Kanonische Form (Minimierung) $\min\ (c^T x) \quad u.d.N. \quad Ax \geq b,\ x \geq 0$
- Standard-Form (Maximierung) $\max\ (c^T x) \quad u.d.N. \quad Ax = b,\ x \geq 0$
- Standard-Form (Minimierung) $\min\ (c^T x) \quad u.d.N. \quad Ax = b,\ x \geq 0$

Transformationen

Umwandlung von min- zu max-Problemen

$$\min c^T x \Leftrightarrow \max -c^T x$$

Umwandlung von $\leq$ zu $\geq$ Ungleichungen

$$Ax \geq b \Leftrightarrow -Ax \leq -b$$

Ersatz von Gleichungen durch Ungleichungen

$$Ax = b \Leftrightarrow \begin{pmatrix} Ax \leq b \\ Ax \geq b \end{pmatrix}$$

Umwandlung von Ungleichungen zu Gleichungen

$$Ax \leq b \Leftrightarrow \begin{pmatrix} Ax + s = b \\ s \geq 0 \end{pmatrix}$$

Umwandlung von Vorzeichenunbeschränkten Variablen

$$x = x^+ - x^-, \qquad x^+ \geq 0, \qquad x^- \geq 0$$

Beispiel Umformulierung zur Kanonischen Form

$$\min 3x_1 + 2x_2$$

$$\begin{array}{llll} u.d.N. & 1x_1 & +1x_2 & +1x_3 & = 7 \\ & & +1x_2 & +1x_3 & \leq 5 \end{array}$$

$$x_1 \leq 0, \qquad x_2 \geq 0, \qquad x_3 \in \mathbb{R}$$

- Schritt 1: $x_1 \leq 0 \rightarrow x_1 \geq 0$
- Schritt 2: $x_3 \in \mathbb{R} \rightarrow x_3 \in \mathbb{R}^+$
- Schritt 3: $Ax = b \rightarrow Ax \leq b \ \& \ Ax \geq b$
- Schritt 4: $min \rightarrow max$

$$\max 3x_1 - 2x_2$$

$$\begin{array}{llllll} u.d.N. & -1x_1 & +1x_2 & +1x_3^+ - x_3^- & \leq +7 \\ & +1x_1 & -1x_2 & -1x_3^+ + x_3^- & \leq -7 \\ & & +1x_2 & +1x_3^+ - x_3^- & \leq +5 \end{array}$$

$$x_1 \geq 0, \qquad x_2 \geq 0, \qquad x_3 = x_3^+ - x_3^-, \qquad x_3^+ \geq 0, \qquad x_3^- \geq 0$$

Lineare Optimierung (Graphisch)

Graphische Lösung von LP's

Ausgangsproblem $(x_1 \geq 0, x_2 \geq 0)$

$$\min z = -10x_1 - 40x_2$$

$$40x_1 + 24x_2 \leq 480$$

$$24x_1 + 48x_2 \leq 480$$

$$60x_2 \leq 480$$

Niveaulinie

$$c = -10x_1 - 40x_2, \qquad c\ fix$$

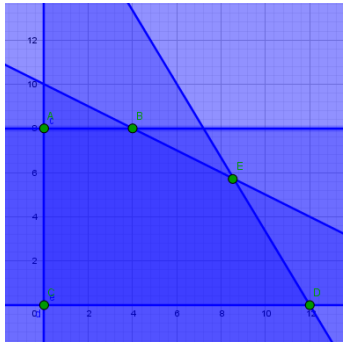
$$40x_2 = -10x_1 - c$$

$$x_2 = -0.25x_1 - c/40$$

$$c = -80, \qquad x_2 = -0.25x_1 + 2$$

$$c = -160, \qquad x_2 = -0.25x_1 + 4$$

Optimallösungen sind unter den **Eckpunkten** des zulässigen Bereiches zu finden.



Charakterisierung der Eckpunkte

Für die Chrakterisierung der Eckpunkte wechselt man in die *Standardform*. Dabei werden sogenannte *Schlupfvariablen* eingeführt.

$$\min z = -10x_1 - 40x_2$$

$$40x_1 + 24x_2 + x_3 = 480$$

$$24x_1 + 48x_2 + x_4 = 480$$

$$60x_2 + x_5 = 480$$

Darstellung in Matrix schreibweise

$$\min z = \underbrace{(-10, -40,0,0,0)}_{c} \cdot (x_1 \quad \dots \quad x_5)^T$$

$$\underbrace{\begin{pmatrix} 40 & 24 & 1 & 0 & 0 \\ 24 & 48 & 0 & 1 & 0 \\ 0 & 60 & 0 & 0 & 1 \end{pmatrix}}_{A} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_5 \end{pmatrix} = \underbrace{\begin{pmatrix} 480 \\ 480 \\ 480 \end{pmatrix}}_{b}$$

Beobachtungen

- Jede Auswahl von Spalten, die zu einer Ecke gehören sind *linear unabhabhängig*
- Jede Auswahl von linear *unabhängigen Spalten* definiert eine Ecke.

Ecken aus Grafik ausgelesen

1. $(0,0)$ $(0,0,480,480,480)$
2. $(0,8)$ $(0,8,288,96,0)$
3. $(4,8)$ $(4,8,128,0,0)$
4. $(\frac{60}{7}, \frac{40}{7})$ $(\frac{60}{7}, \frac{40}{7}, 0,0, \frac{960}{7})$
5. $(12,0)$ $(12,0,196,480)$

Für die Berechnung der Eckpunkte sind nur 3 Spalten von A wichtig.

Spalten aus A zu den Variablen $x_i > 0$

1. $\left\{\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}\right\} = \{A_3, A_4, A_5\}$
2. $\left\{\begin{pmatrix}24\\48\\60\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}\right\} = \{A_2, A_3, A_4\}$
3. $\left\{\begin{pmatrix}40\\24\\0\end{pmatrix}, \begin{pmatrix}24\\48\\60\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}\right\} = \{A_1, A_2, A_3\}$

Simplexverfahren Theorie

Formale Lösung

1. Basis B wählen
2. Erste Umformulierung

$$\begin{array}{c} \min z = c^T x \\ Ax = b \\ x \geq 0 \end{array} \quad \Leftrightarrow \quad \begin{array}{c} \min z = \tilde{c}^T \tilde{x} + d^T y \\ B\tilde{x} + Dy = b \\ \tilde{x} \geq 0, y \geq 0 \end{array}$$

3. Gleichungsrestriktionen

$$B\tilde{x} + Dy = b \Leftrightarrow \tilde{x} = B^{-1}b - B^{-1}Dy$$

4. Basislösung auf Zulässigkeit überprüfen
5. Substitution

$$\tilde{c}^T\tilde{x} + d^T y = \tilde{c}^T(B^{-1}b - B^{-1}dy) + d^T y$$

$$\tilde{c}^T\tilde{x} + d^T y = \tilde{c}^T B^{-1}b - (\tilde{c}^T B^{-1}D - d^T)y$$

6. Zweite Umformulierung

$$\begin{array}{c} \min z = \tilde{c}^T B^{-1}b - (\tilde{c}^T B^{-1}D - d^T)y \\ \tilde{x} + B^{-1}Dy = B^{-1}b \\ \tilde{x} \geq 0, y \geq 0 \end{array}$$

Definition: *zulässige Basis*

B heisst zulässige Basis, wenn $B^{-1}b \geq 0$ gilt. Wenn B eine zulässige Basis ist $\rightarrow \tilde{x} = B^{-1}b,\ y = 0$ ergibt eine zulässige Basislösung.

Simplexkriterium

Sei B eine zulässige Basis.

- Gilt in (4) $\tilde{c}^T B^{-1}D - d^T \leq 0$, dann ist $(\tilde{x} = B^{-1}b, y = 0)$ optimal
- Gilt in (4) $B^{-1}b > 0$ und ist $\tilde{x} = B^{-1}b, y = 0)$ optimal, dann gilt $\tilde{c}^T B^{-1}D - d^T \leq 0$

Pivotspalte

- $\max |z_i|$ Betragsmässig grösstes Element in der z Zeile

Pivotzeile

- $\min Q$ Kleinstes Element in der Spalte $Q = b_i/x_i$

Simplexverfahren

1. Zulässige Basislösung bestimmen und Simplextableau abfüllen
2. Alle nicht Basis-Variablen (NBV) der Zielfunktionszeile (max: $z \leq 0$, min: $z \geq 0$) gefunden?
   a. Erfüllt: *Optimale Lösung* gefunden!
   b. Sonst: Wähle NBV mit positivem Koeffizienten aus (Pivotspalte)
3. Betrachte alle Basis-Variablen (BV) in Pivotspalte
   a. Falls alle Koeffizienten $\leq 0 \rightarrow$ *Problem unbeschränkt*
   b. Sonst:
      i. Bilde Quotienten Q aus rechter Seite und positiven (> 0) Koeffizienten in Pivotspalte.
      ii. Wähle BV mit minimalem Quotienten aus (*Pivotzeile*) und tausche die gewählte NBV mit der BV aus.
      iii. Transformiere Simplextablau durch Gauss-Schritte, so dass in Pivotspalte ein Einheitsvektor entsteht mit 1 beim Pivotelement.

Simplexverfahren - Beispiel

Beispiel: Übung 1, Aufgabe 3

$$\min 300x_1 + 400x_2$$

$$\begin{array}{lrrrrrl} u.d.N. & +2x_1 & +3x_2 & x_3 & 0 & 0 & = 33 \\ & +1x_1 & +1x_2 & 0 & x_4 & 0 & = 15 \\ & +1x_1 & +3x_2 & 0 & 0 & x_5 & = 27 \end{array}$$

Daten

$$C = (300,400,0,0,0)^T, \qquad b = (33,15,27)^T$$

$$A = \begin{pmatrix} 2 & 3 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 3 & 0 & 0 & 1 \end{pmatrix} = (A_1, A_2, A_3, A_4, A_5)$$

$$B = \begin{pmatrix} 3 & 1 & 0 \\ 1 & 0 & 1 \\ 3 & 0 & 0 \end{pmatrix} = A_2, A_3, A_4, \qquad D = (A_1, A_5) = \begin{pmatrix} 2 & 0 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

$$\tilde{x}^T = (x_2, x_3, x_4), \qquad y^T = (x_1, x_5)$$

$$\tilde{c}^T = (c_2, c_3, c_4) = (400,0,0), \qquad d^T = (c_1, c_5) = (300,0)$$

1. Erste Umformulierung

$$\begin{array}{c} \min z = c^T x \\ Ax = b \\ x \geq 0 \end{array} \quad \Leftrightarrow \quad \begin{array}{c} \min z = \tilde{c}^T \tilde{x} + d^T y \\ B\tilde{x} + Dy = b \\ \tilde{x} \geq 0, y \geq 0 \end{array}$$

$$\min(400,0,0)\tilde{x} + (300,0)y$$

$$\begin{pmatrix} 3 & 1 & 0 \\ 1 & 0 & 1 \\ 3 & 0 & 0 \end{pmatrix}\tilde{x} + \begin{pmatrix} 2 & 0 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} y, \qquad \tilde{x} \geq 0, y \geq 0$$

3. Gleichungsrestriktionen

$$B\tilde{x} + Dy = b \Leftrightarrow \tilde{x} = B^{-1}b - B^{-1}Dy$$

$$\tilde{x} = B^{-1}b - B^{-1}Dy = \begin{pmatrix} 9 \\ 6 \\ 6 \end{pmatrix} - \begin{pmatrix} 1/3 & 1/3 \\ 1 & -1 \\ 2/3 & -1/3 \end{pmatrix} y$$

4. Basislösung auf Zulässigkeit überprüfen

$$\tilde{x} = \begin{pmatrix} 9 \\ 6 \\ 6 \end{pmatrix} \geq 0 \rightarrow (\tilde{x}^T, 0,0) = (9,6,6,0,0), \qquad y = (0,0)$$

$(\tilde{x}^T, 0,0)$ ist eine zulässige Basislösung. → Ecke $(x_1, x_2) = (0,9)$ im original modell.

5. Substitution

$$\tilde{c}^T\tilde{x} + d^T y = \tilde{c}^T(B^{-1}b - B^{-1}dy) + d^T y$$

$$\tilde{c}^T\tilde{x} + d^T y = \tilde{c}^T B^{-1}b - (\tilde{c}^T B^{-1}D - d^T)y$$

$$\tilde{c}^T\tilde{x} + d^T y = (400,0,0) \cdot \begin{pmatrix} 9 \\ 6 \\ 6 \end{pmatrix} - \left((400,0,0) \cdot \begin{pmatrix} 1/3 & 1/3 \\ 1 & -1 \\ 2/3 & -1/3 \end{pmatrix} - (300,0) \right) y$$

$$= 3600 - (-166.67,133.33) \cdot \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

6. Zweite Umformulierung

$$\begin{array}{c} \min z = \tilde{c}^T B^{-1}b - (\tilde{c}^T B^{-1}D - d^T)y \\ \tilde{x} + B^{-1}Dy = B^{-1}b \\ \tilde{x} \geq 0, y \geq 0 \end{array}$$

$$\min z = 3600 - (-166.67,133.33) \cdot \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

$$\tilde{x} + \begin{pmatrix} 1/3 & 1/3 \\ 1 & -1 \\ 2/3 & -1/3 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 9 \\ 6 \\ 6 \end{pmatrix}, \qquad \tilde{x} \geq 0, y \geq 0$$

Simplexverfahren - Tableaux

Ausgangslage

$$\max 2x_1 + 1x_2 + 1x_3$$

$$3x_1 + 1x_2 + 2x_3 \leq 2$$

$$1x_1 + 1x_2 + 3x_3 \leq 5$$

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

Normalform

$$\max 2x_1 + x_2 + x_3$$

$$u.d.N. \quad \begin{matrix} 3x_1 & 1x_2 & 2x_3 & 1x_4 & 0 & = 2 \\ 1x_1 & 1x_2 & 3x_3 & 0 & 1x_5 & = 5 \end{matrix}$$

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

Lösungen

- Unbeschränkt — Pivotspalte ≤ 0
- Optimal — max: $z_{NBV} \leq 0$, min: $z_{NBV} \geq 0$

Ausgangslage

- Normalform in Tablaux umwandeln

	x_1	x_2	x_3	x_4	x_5	b_i
z	-2	-1	-1	0	0	0
	3	1	2	1	0	2
	1	1	3	0	1	5

Schritt 1

- Pivotspalte $min(z)$ — $min(z) = -2 \rightarrow x_1$
- Hilfsspalte $Q = b_i/x_i$ — $Q = (0.66, 5)$
- Pivotzeile $min(Q)$ — $min(Q) = 0.66 \rightarrow$ Zweite Zeile

	x_1	x_2	x_3	x_4	x_5	b_i	Q
z	**-2**	-1	-1	0	0	0	
	3	**1**	**2**	**1**	**0**	**2**	**0.66**
	1	1	3	0	1	5	5

Schritt 2

- Pivotspalte $min(z)$ — $min(z) = -0.33 \rightarrow x_2$
- Hilfsspalte $Q = b_i/x_i$ — $Q = (2, 6.5)$
- Pivotzeile $min(Q)$ — $min(Q) = 2 \rightarrow$ Zweite Zeile

	x_1	x_2	x_3	x_4	x_5	b_i	Q
z	0	**-0.33**	0.33	0.66	0	1.33	
	1	**0.33**	**0.66**	**0.33**	**0**	**0.66**	**2**
	0	**0.66**	2.33	-0.33	1	4.33	6.5

Finales Tableaux

Umgekehrtes Simplexkriterium überprüfen

$$z \geq 0, \qquad (1, 0, 1, 1, 0) \geq 0$$

	x_1	x_2	x_3	x_4	x_5	b_i
z	1	**0**	1	1	0	2
	3	**1**	**2**	**1**	**0**	**2**
	-2	**0**	1	-1	1	3

$Max.Zielwert: b_0, \qquad BV: x_n = b_i (i > 0), \qquad NBV: x_n = 0$

$Max.Zielwert: z = 2, \qquad BV: (x_2 = 2, x_5 = 3), \qquad NBV: (x_1 = x_3 = x_4 = 0)$

Simplexverfahren – 2-Phasenmethode

Ausgansglage	**Normalform**	**Rechte Seite $< 0 \rightarrow \cdot -1$**	**Hilfsproblem**
$\min -x_1 - x_2 = z$	$\min -x_1 - x_2 = z$	$\min -x_1 - x_2 = z$	$\min a_1 + a_2 = z_H$
$-x_1 + x_2 \leq 1$	$-x_1 + x_2 + x_3 = 1$	$-x_1 + x_2 + x_3 = 1$	$-x_1 + x_2 + x_3 + a_1 = 1$
$2x_1 - x_2 \leq -0.5$	$2x_1 - x_2 + x_4 = -0.5$	$-2x_1 + x_2 - x_4 = 0.5$	$-2x_1 + x_2 - x_4 + a_2 = 0.5$
$x_1, x_2 \geq 0$	$x_1, x_2, x_3, x_4 \geq 0$	$x_1, x_2, x_3, x_4 \geq 0$	$x_1, x_2, x_3, x_4, a_1, a_2 \geq 0$

Vorgehen

1. Umwandeln in Normalform
2. Rechte Seite negativ $\rightarrow \cdot -1$
3. Tableau erstellen
4. Hilfsproblem erzeugen
5. Tableau durch Hilfsproblem erweitern
6. Phase 1: Hilfsproblem lösen $\rightarrow z_H = 0$
7. Phase 2: Originalproblem lösen

Lösungen

- Unbeschränkt — Pivotspalte ≤ 0
- Unzulässig — Hilfsproblem nicht lösbar
- Zulässig — Hilfsproblem lösbar

$$z_H = a_1 + a_2 = (1 + x_1 - x_2 - x_3) + (0.5 + 2x_1 - x_2 + x_4)$$

$$z_H = a_1 + a_2 = 1.5 + 3x_1 - 2x_2 - x_3 + x_4$$

	x_1	x_2	x_3	x_4	a_1	a_2	b_i	Q
z_H	-3	**2**	1	-1	0	0	1.5	
z	1	**1**	0	0	0	0	0	
	-1	**1**	**1**	**1**	**1**	**0**	**1**	**1**
	-2	**1**	0	-1	0	1	0.5	2

	x_1	x_2	x_3	x_4	a_1	a_2	b_i	Q
z_H	1	0	**1**	1	0	-2	0.5	
z	3	0	**0**	1	0	-1	-0.5	
	1	**0**	**1**	**1**	**1**	**-1**	**0.5**	**0.5**
	-2	1	**0**	-1	0	1	0.5	–

	x_1	x_2	x_3	x_4	a_1	a_2	b_i	Q
z_H	0	0	**0**	0	-1	-1	0	
z	3	0	**0**	1	0	-1	-0.5	
	1	**0**	**1**	**1**	**1**	**-1**	**0.5**	**0.5**
	-2	1	**0**	-1	0	1	0.5	–

Simplexverfahren – Primales und Duales Verfahren

Vom **Primalen** (PP) zum **Dualen Problem** (DP)

1. Variablen ersetzen $c^T \to b^T,\ b \to c,\ x \to y$
2. Matrix transponieren $A \to A^T$
3. Operator ändern $\leq \to \geq$
4. Typ ändern $max \to min$

Das Lineare Problem sei in Kanonischer Form gegeben.

$$\begin{matrix} max & c^T x \\ bzgl & Ax \leq b \\ & x \geq 0 \end{matrix} \quad \Leftrightarrow \quad \begin{matrix} min & b^T y \\ bzgl & A^T y \geq c \\ & y \geq 0 \end{matrix}$$

Primales Problem

$$\begin{matrix} max & +x_1 & +3x_2 & & & \max(1,3) \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ bzgl. & +x_1 & +2x_2 & \leq & 110 & \begin{pmatrix} 1 & 2 \\ 1 & 4 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 110 \\ 160 \\ 100 \end{pmatrix} \\ & +x_1 & +4x_2 & \leq & 160 & \\ & +x_1 & +x_2 & \leq & 100 & \\ & +x_1 & +x_2 & \geq & 0 & \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq 0 \end{matrix}$$

Duales Problem

$$\min(110,160,100) \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \leq \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

$$(y_1, y_2, y_3)^T \geq 0$$

Schwache Dualität

Sei x_0 ein zulässiger Punkt von (*PP*), d.h. $Ax_0 \leq b, x_0 \geq 0$

und y_0 ist ein zulässiger Punkt von (*DP*), d.h. $A^T y_0 \geq c, y \geq 0$

so gilt $\underline{z} = c^T x_0 \leq b^T y_0 = \overline{z}$

Ausserdem gilt, falls

$$c^T x_0 = b^T y_0$$

Dann ist x_0 optimal für (*PP*) und y_0 optimal für (*DP*).

Starke Dualität

Für das primale und das duale Problem gilt genau eine der folgenden Möglichkeiten

1. PP und DP haben keine zulässige Lösung
2. PP ist unbeschränkt, DP ist unzulässig (oder vice versa)
3. Beide Probleme sind zulässig. Dann besitzen PP und DP auch Optimallösungen x^* und y^* mit $c^T x^* = b^T y^*$
4. Zulässige Punkte x^* von PP und y^* von DP sind nur optimal, wenn

$$\begin{cases} x_j^* (A^T y^* - c)_j = 0, & j = 1, \dots, m \\ y_j^* (Ax^* - b)_i = 0, & i = 1, \dots, n \end{cases}$$

$$\begin{aligned} x_j^* > 0 \quad & \to (A^T y^* - c)_j = 0 \\ (A^T y^* - c)_j > 0 \quad & \to x_j^* = 0 \\ y_i^* > 0 \quad & \to (Ax^* - b)_i = 0 \\ (Ax^* - b)_i < 0 \quad & \to y_i^* = 0 \end{aligned}$$

Ganzzahlige Optimierung

Ein ganzzahliges lineares Problem (GLP) ist ein Optimierungsproblem der Form

$$\max c^T x$$
$$Ax \leq b, \qquad (L)$$
$$x \geq 0$$
$$x \in \mathbb{Z}^n$$

d.h. die Variablen $x_i \in \mathbb{Z}^n$ für alle $i = 1, \dots, n$.

Lässt man in (L) die Bedingung $x \in \mathbb{Z}^n$ weg, so enthält man die LP-Relaxierung L' von L, d.h.

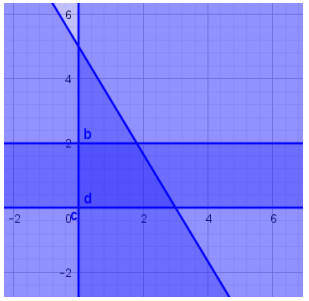
$$\max c^T x$$
$$Ax \leq b, \qquad (L')$$
$$x \geq 0$$

Bemerkung:

- Der Optimalwert von L' ist immer grösser oder gleich dem Optimalwert von L.
- Gilt in L auch $x \in \{0,1\}^n$, dann nennt man L auch ein binäres Problem.

Wie löst man ein GLP?

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



Man braucht ein neues Verfahren zum Lösen von GLP.

Wir betrachten ein Branch and Bound Verfahren für GLP, dies ist ein exaktes Verfahren auf dem praktisch alle aktuellen Optimierungsprobleme aufbauen.

Branch and Bound (Ganzzahlige Optimierung)

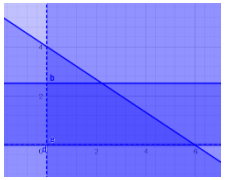
$$\max x_1 + 2x_2$$

$$2x_2 \leq 5$$

$$2x_1 + 3x_2 \leq 12$$

$$x_1, x_2 \geq 0$$

$$x_1, x_2 \in \mathbb{Z}$$



Relaxierung für Ausgangspunkt

$$\max x_1 + 2x_2$$

$$2x_2 \leq 5$$

$$2x_1 + 3x_2 \leq 12$$

$$x_1, x_2 \geq 0$$

Lösung des Simplexalgorithmus:

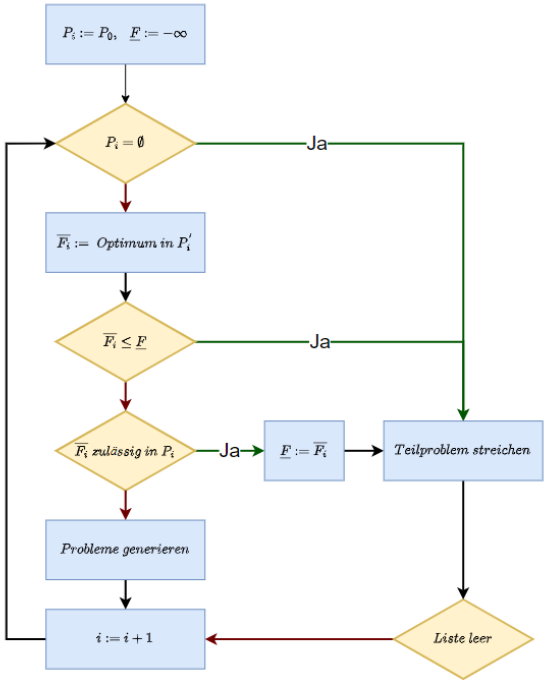
$$x_1 = 2.25, \qquad x_2 = 2.5, \qquad z = 7.25$$

$$P_0 = (2.25, 2.5), \qquad \underline{F} := -\infty$$

- Lösungen von P_0: $\overline{F} = 7.25 > \underline{F} = -\infty$
- 7.25 nicht zulässig in P_0
- Teile P_0 in P_1, P_2, P_3, P_4 auf
- Liste: $\{P_1, P_2, P_3, P_4\}$

$$P_1 = (2,2), \qquad P_2 = (2,3),$$

$$P_3 = (3,2), \qquad P_4 = (3,3)$$



$$P_1 = (2,2), \qquad \underline{F} := -\infty$$

- Lösungen von P_1: $\overline{F} = 5 > \underline{F} = -\infty$
- 5 zulässig in P_1
- $\underline{F} = 5$
- P_1 streichen → Liste: $\{P_2, P_3, P_4\}$

$$P_2 = (2,3), \qquad \underline{F} := 5$$

- Lösungen von P_2: $\overline{F} = 8 > \underline{F} = 5$
- 8 nicht zulässig in P_2
- $\underline{F} = 5$
- P_2 streichen → Liste: $\{P_3, P_4\}$

$$P_3 = (3,2), \qquad \underline{F} := 5$$

- Lösungen von P_3: $\overline{F} = 7 > \underline{F} = 5$
- 7 zulässig in P_3
- $\underline{F} = 7$
- P_3 streichen → Liste: $\{P_4\}$

$$P_4 = (3,3), \qquad \underline{F} := 7$$

- Lösungen von P_0: $\overline{F} = 9 > \underline{F} = 7$
- 9 nicht zulässig in P_4
- $\underline{F} = 7$
- P_4 streichen → Liste: $\{\}$

$\underline{F} = 7$ ist der Optimalwert an der Stelle $P_3 = (3,2)$.

$$x_1 = 3, \qquad x_2 = 2$$

Optimierung auf Graphen

Graphen

Ein Graph $G = (V, E)$ besteht aus

- einer endlichen Menge von Knoten V
- einer Menge von Kanten $E \subseteq V \times V$

Eigenschaften

Ein Pfad ist eine Sequenz benachbarter Knoten

- *einfacher Pfad* Kein Knoten kommt doppelt vor
- *zyklischer Pfad* Anfangs- und Endknoten sind identisch

Die *Pfadlänge* ist die Anzahl der Kanten des Pfades.

- *Vollständigen Graph* Jeder Knoten ist mit jedem anderen Knoten **direkt** verbunden
- *Verbundener Graph* Jeder Knoten ist mit jedem anderen Knoten **indirekt** verbunden

Die *Dichte eines Graphen* ist das Verhältnis der Anzahl Kanten zu Anzahl möglicher Kanten (0 – 1).

- *Dichter Graph* Nur wenige Kanten im Graph fehlen
- *Dünner Graph* Nur wenige Kanten im Graph sind vorhanden

Schwierigkeit des Problems

- Struktur des Graphen (zyklisch)
- Art der Gewichte (positiv, negativ, gemischt)
- Problemvariante (one-to-one, one-to-all, all-to-all)

Eine LP-Formulierung für das Wegproblem

Gesucht ist der kürzeste Weg von einer Quelle s zu einer Senke t in einem gerichteten s-t-Graphen $G = (V, E)$ mit

- positiven Gewichten $d_{vw} > 0 \quad (v, w) \in E$

Zielfunktion

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

Nebenbedingungen

Jeder Knoten (ausser Start- und Endknoten) hat gleich viele eingehende, wie ausgehende Kanten des Weges $(x_{uv} = 0)$.

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

Eine Kante $(v, w) \in E$ kann für den Weg verwendet werden.

- Verwendete Kanten $x_{vw} = 1$
- Nicht verwendete Kanten $x_{vw} = 0$

$$0 \le x_{vw} \le 1, \qquad (v, w) \in E$$

$$x_{vw}\ integer, \qquad (v, w) \in E$$

Optimierung auf Graphen

Dijkstra Algorithmus (*one-to-all*)

Berechne kürzeste Wege von einem Knoten s zu allen anderen Knoten im ungerichteten Graphen $G = (V, E)$.

Voraussetzungen: Gewichte $d_e \geq 0$

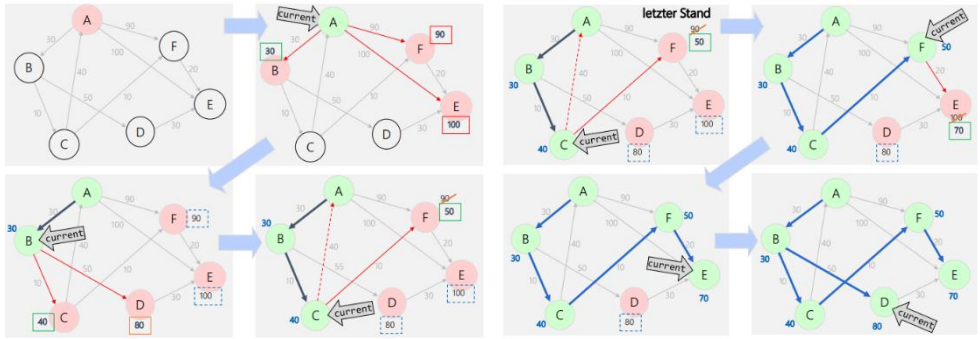
Teile die Knoten in 3 Gruppen auf

- Besuchte Knoten (kleinste Distanz bekannt)
- Benachbart zu allen bereits besuchten Knoten
- Ungesehene Knoten (der Rest)

Solange nicht alle Knoten besucht wurden (grün sind):

1. Berechne für alle benachbarten Knoten des aktuell besuchten Knotens die neuen Gewichte
2. Suche unter **allen** benachbarten Knoten (nicht nur jene des aktuellen Knotens) denjenigen, dessen Pfad zum Startknoten das kleinste Gewicht (kürzeste Distanz) hat (grüner Rahmen um Zahl).
3. Besuche diesen (neuen Knoten).

Soll nur der minimale Pfad für einen bestimmten Knoten gesucht werden, kann die Suche abgebrochen werden, sobald dieser «current» wird.



Floyd-Warshall (*all-to-all*)

Berechne kürzeste Wege von jedem Knoten i zu jedem Knoten j im ungerichteten Graphen $G = (V, E)$.

Voraussetzungen: Keine

Sei n die Anzahl Knoten, im Beispiel hier $n = 11$ und die Knoten seien nummeriert von 1 bis n.

In n sukzessiven Iterationen wird für jedes Knotenpaar (i, j) die Länge eines «bedingten» kürzesten Weges berechnet, wobei die Bedingungen lauten:

- *Iteration k:* Knoten $1, \dots, k$ sind als Zwischenknoten erlaubt

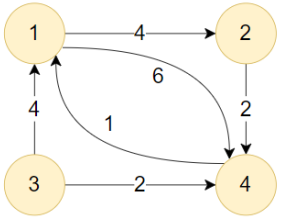
Example

- *Iteration 0:* $\{v_1 \to v_2 = 4, v_1 \to v_4 = 5, \dots\}$
- *Iteration 1:* $\{v_3 \to v_2 = 8\}$
- ...
- *Iteration 4:* $\{v_2 \to v_1 = 3, v_3 \to v_1 = 3, v_3 \to v_2 = 7\}$

k	1	2	3	4
1	Distance			
	Previous			
2	Distance			
	Previous			
3	Distance			
	Previous			
4	Distance			
	Previous			

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

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



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

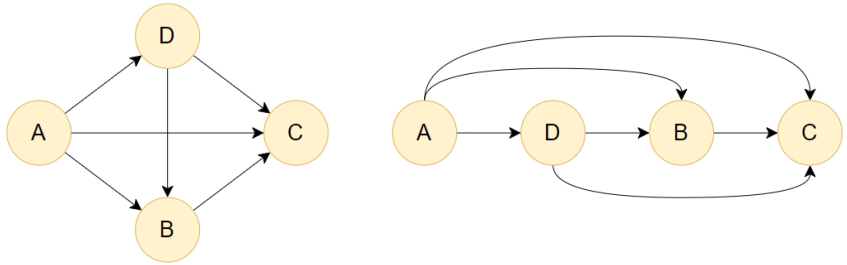
3	1	2	3	4
1	0	4	∞	6
	1	1		1
2	∞	0	∞	2
		2		2
3	4	8	0	2
	3	1	3	3
4	1	5	∞	0
	4	1		4

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

Optimierung auf Graphen

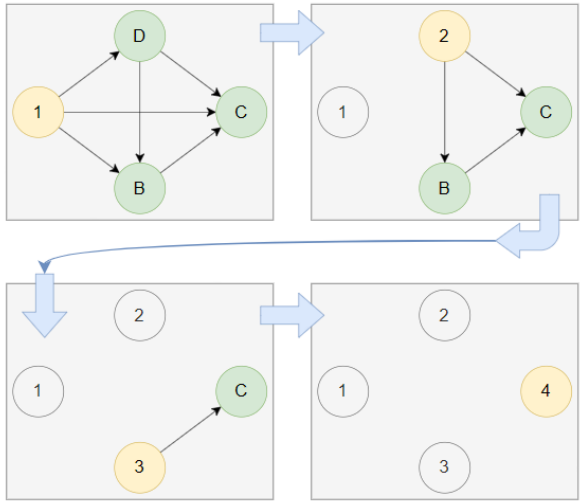
Topologische Sortierung

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



Vorgehen zur Bestimmung einer Topologischen Sortierung

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



Algorithmus: Längste Wege in azyklischen Graphen (*one-to-all*)

Berechne *längste Wege* von einer Quelle s zu allen anderen Knoten im gerichteten Graphen $G = (V, E)$.

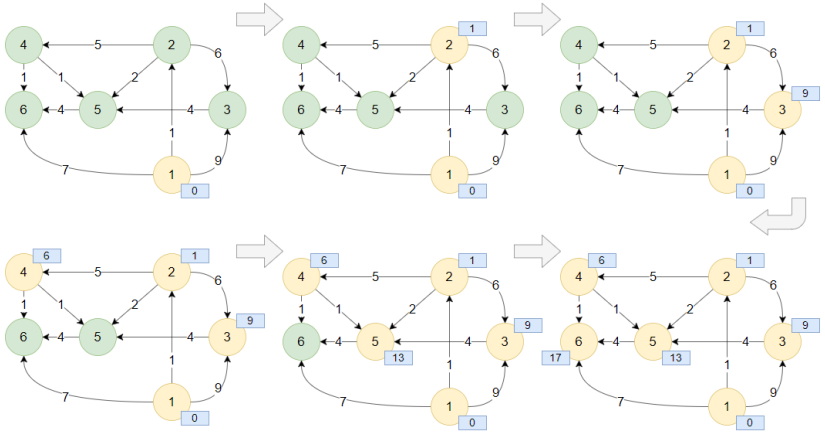
Voraussetzung: Keine Zyklen

1. Bestimme eine *Topologische Sortierung*
2. Iteriere über die Knoten v_i mit $i = (1, \dots, n)$
   - Bestimme die maximale Distanz d_i zum Knoten v_i ausgehend von jedem Vorgänger $pred$. $\max(d_{pred} + e_{pred,i})$

Beispiel

$n = Iteration, \quad l_v = Length, \quad pred_v = Previous$

1. $l_1 = 0 \quad pred_1 = 1 \quad = \max\{0\}$
2. $l_2 = 1 \quad pred_2 = 1 \quad = \max\{0 + 1\}$
3. $l_3 = 9 \quad pred_3 = 2 \quad = \max\{6 + 1, 9\}$
4. $l_4 = 6 \quad pred_4 = 2 \quad = \max\{1 + 5\}$
5. $l_5 = 13 \quad pred_5 = 3 \quad = \max\{6 + 1, 1 + 2, 9 + 4\}$
6. $l_6 = 17 \quad pred_6 = 5 \quad = \max\{6 + 1, 13 + 4, 7\}$



Optimale Zyklen (Traveling-Salesman Problem)

Gegeben sei ein ungerichteter Graph $G = (V, E)$. Seien $v, w \in V$ zwei unterschiedliche Knoten. Ein *Hamiltonscher Weg* von v nach w ist ein Weg von v nach w, welcher **alle Knoten von G genau einmal** besucht.

Ein *Hamiltonscher Zyklus* ist ein Zyklus, welcher **alle Knoten von G, genau einmal besucht**. Seien $d_e, e \in E$, die Kantengewichte in G.

Fundamentale Probleme: Bestimmung des *Hamiltonschen Weges minimaler Länge* von v nach w und des *Hamiltonschen Zyklus minimaler Länge.*

Traveling Salesman Problem (TSP)

Gegeben sind die Städte $1,2,3, \dots, n$. Die Matrix D_{ij} der Distanzen zwischen allen Städten (i, j).

Das *Traveling-Salesman* Problem lautet: Bestimme eine Reihenfolge $i(1), i(2), \dots, i(n)$ der Städte, so dass die zurückgelegte Distanz minimal ist:

$$d_{i(i),i(2)} + d_{i(2),i(3)} + \cdots + d_{i(n),i(1)} \rightarrow min!$$

Das TSP entspricht dem Problem des minimalen *Hamiltonschen Zyklus*.

Symmetrisches TSP im ungerichteten Graphen

- Distanzmatrix $D = (d_{ij})$, *symmetrisch*
- Graph $G = (V, E)$, *ungerichtet, vollständig* mit
  - V: Menge der Städte $V = \{1,2, \dots, n\}$
  - E: Menge der Kanten $E = \{(i, j): i, j \in V, i \neq j\}$
  - Kantenlängen d_{ij} für $(i, j) \in E$

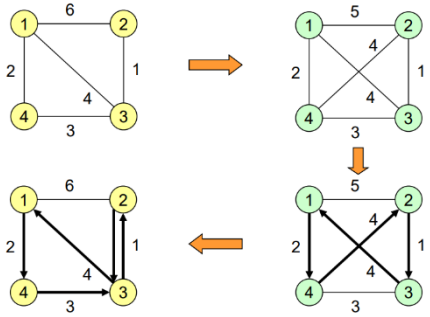
Das TSP in einem *unvollständigen* Graphen lässt sich immer in einem *vollständigen* Graphen formulieren, indem die fehlenden Kanten mit einem genügend grossen Gewicht M ergänzt werden (z.B. $M > \Sigma \{d_e: e \in E\}$).

Bestimme einen *Hamiltonschen Zyklus* minimaler Länge in G.

Graphisches TSP: Jede Stadt muss *mindestens* einmal besucht werden. Für die Tourenplanung in der Praxis meist die relevante Version.

Das Graphische TSP kann wie folgt als «normales» TSP formuliert werden:

Konstruiere einen *vollständigen* Graphen G', dessen Kantenlängen d'_{ij} der *Länge eines kürzesten Weges von i nach j* entsprechen und bestimme eine Tour in G'.



Einige Heuristiken für das TSP

Das TSP ist *NP-vollständig,* daher können Heuristiken sinnvoll sein.

Konstruktive Heuristiken (→ generiert eine Tour)

- Nearest Neighbor
- Nearest / Cheapest / Farthest Insertion

Verbesserungs-Heuristiken (Local Search)

- 2 bzw. 3-opt TODO

Optimale Zyklen (Traveling-Salesman Problem)

Nearest Neighbor

Initialisierung $(i = 0)$*:*

- Wähle einen Knoten $v_i \in V$ als Startknoten.
- Der Pfad wird initialisiert als $P := \{v_i\}$

Iterationen $i = (1, \dots, n)$*:*

1. Suche die *kürzeste Kante* e_i ausgehend vom derzeitigen Knoten v_{i-1}, wobei der Zielknoten v_i nicht bereits im Pfad P enthalten sein darf.
2. Falls *keine kürzeste Kante* gefunden wurde → *Iteration beenden*
3. Verbinde die Knoten v_{i-1} und v_i über die kürzeste Kante e_i
4. Füge den neuen Knoten v_i zum Pfad P hinzu

Um den Zyklus zu vollenden wird die letzte Knoten v_n noch mit dem ersten Knoten v_0 verbunden.

Nearest / Farthest Insertion

Initialisierung $(i = 0)$*:*

- Wähle einen Knoten $v_i \in V$ als Startknoten.
- Der Zyklus wird initialisiert als $Z := \{v_i\}$

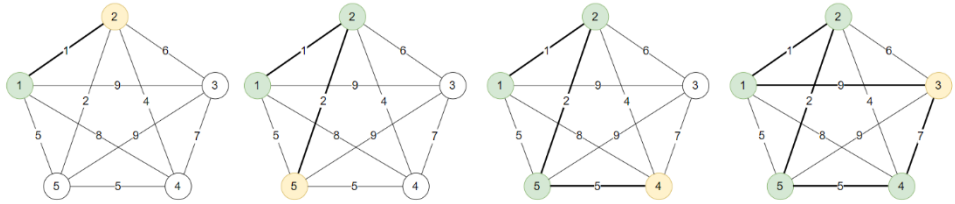
Iterationen $i = (1, \dots, n)$*:*

1. Suche einen Knoten k, der sich ausserhalb des Zyklus Z befindet und am nächsten zu einem Knoten in Z ist.
2. Finde eine Kante $\{x, y\}$ innerhalb von Z, sodass
   - *Nearest Insertion*: $d_{xk} + d_{ky} - d_{xy}$ minimal ist.
   - *Farthest Insertion*: $d_{xk} + d_{ky} - d_{xy}$ maximal ist.
3. Bilde einen neuen Zyklus Z, indem $\{x, y\}$ durch $\{x, k\}$ und $\{k, y\}$ ersetzt werden.
4. Falls der Zyklus Z alle Knoten enthält → *Iteration beenden*

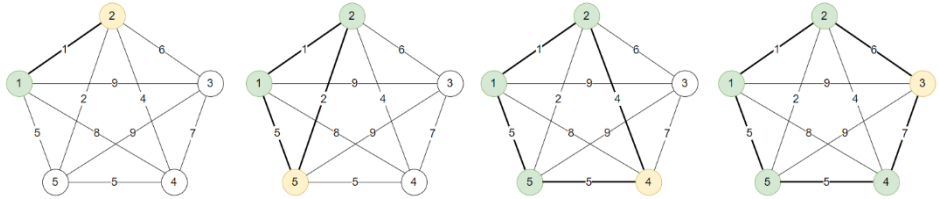
Ausgangslage

d_{ij}	1	2	3	4	5
1	–	1	9	8	5
2		–	6	4	2
3			–	7	9
4				–	5
5					–

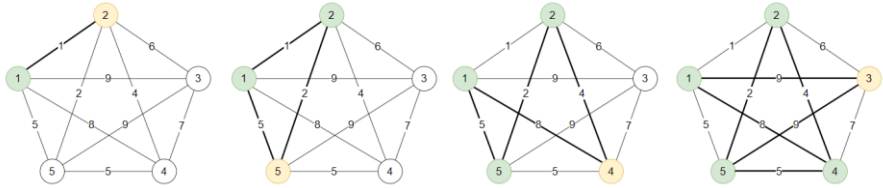
Nearest Neighbor



Nearest Insertion



Farthest Insertion

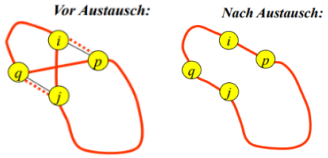


Optimale Zyklen (Traveling-Salesman Problem)

Verbesserungs-Heuristiken

Eine gegebene Lösung wird sukzessive verbessert, indem kleine Veränderungen vorgenommen werden. Grundprinziep des «Local Search».

2-opt Heuristik

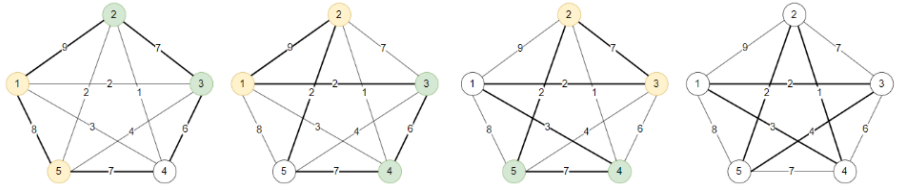


Initialisierung:

- Bestimme einen Zyklus / Tour T

Iterationen: Solange nicht alle Austausche versuche wurden

1. Bestimme zwei Kanten (i,j) und (p,q) in der Tour, für welche der Austausch noch nicht versucht wurde. Wobei $i \neq j \neq p \neq q$.
2. Vertausche die Kanten (i,j) und (p,q) durch (i,p) und (j,q).
3. Berechne die Änderung Δ der Tourlänge $\Delta = d_{ip} + d_{jq} - d_{ij} - d_{pq}$
4. Akzeptiere den Tausch falls $\Delta < 0$.



3-opt Heuristik

Analog zu 2-opt Heuristik

ILP-Formulierung für das symmetrische TSP

Bestimme die minimale Tour im ungerichteten Graphen $G = (V, E)$ mit

- positiven Gewichten $d_{vw} > 0 \quad (v,w) \in E$

Zielfunktion

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

Nebenbedingungen

1. Jeder Knoten muss genau mit zwei Kanten verbunden werden.

$$\sum_{w\in V:(v,w)\in E} x_{vw} = 2, \qquad v \in V$$

2. Aus jeder Subtour S mit Anzahl Knoten $|S|$ dürfen höchstens $|S| - 1$ Kanten verwendet werden.

$$\sum_{v,w\in S:(v,w)\in E} x_{vw} \leq |S| - 1, \qquad \emptyset \subset S \subset V$$

3. Knoten v und w sind verbunden $\rightarrow x_{vw} = 1$

$$x_{vw} \in \{0,1\}, \qquad (v,w) \in E$$

Optimale Bäume

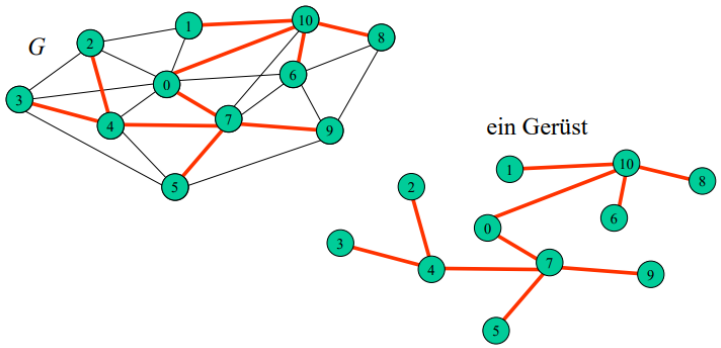
Definitionen

Sei $G = (V, E)$ ein ungerichteter Graph. Ein *Baum* in G ist ein Subgraph von G, der

- Zusammenhängend ist
- Keine Zyklen enthält

Ein *Gerüst / Spannbaum* (*spanning tree*) ist ein Baum, der alle Knoten des Graphen enthält und dessen Kanten im Graphen vorhanden waren.

Das *minimale Gerüst* ist das Gerüst dessen Länge minimal ist unter allen möglichen Gerüsten.



Formulierung als ILP und LP

$$\min \sum_{alle\ Kanten\ (i,j)} d_{ij} x_{ij}$$

Für jede Partition* von V in Mengen $S_1, \dots, S_k$, wobei $2 \leq k \leq n$

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

$$x_{ij} \geq 0$$

Minimales Gerüst

Das Problem des *minimalen Gerüsts* ist eines der einfachsten Probleme der Kombinatorischen Optimierung! Es kann mit dem Kruskal-Algorithmus auf einfache Weise gelöst werden.

Kruskal-Algorithmus

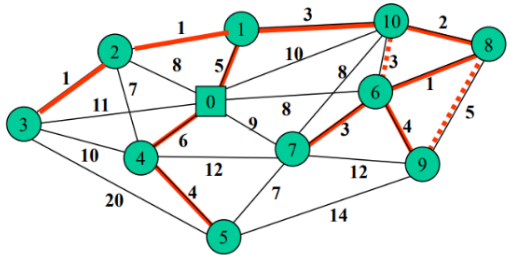
Sei n die Anzahl Knoten und m die Anzahl Kanten des Graphen.

Initialisierung:

- Ordne die Kanten nach aufsteigenden Gewichten $e_1, e_2, e_3, \dots, e_m$ das heisst $d_{e1} \leq d_{e_2} \leq \dots \leq d_{e_m}$.
- Menge der gewählten Kanten $J := \emptyset, i := 0$

Iterationen $(i = 0, \dots, n-1)$

1. Falls $\{e_i\} \cup J$ keinen Zyklus enthält.
   - Füge e_i zu den gewählen Kanten hinzu $J := \{e_i\} \cup J$



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

Optimale Bäume

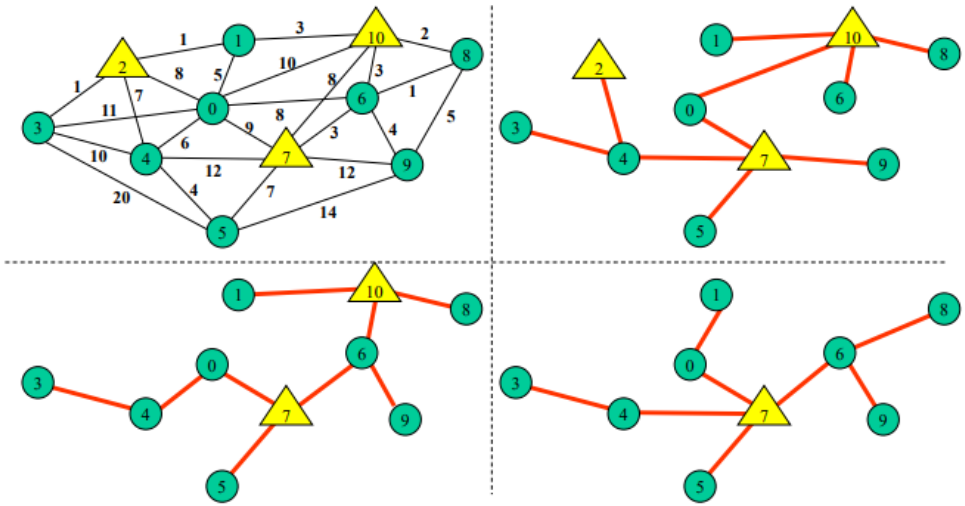
Steiner-Bäume

Gegeben sei ein zusammenhängender ungerichteter Graph $G = (V, E)$. T sei eine Teilmenge von V.

Gesucht ist nun ein Teilgraph, der alle Knoten aus T verbindet und bei dem die Summe der Kantengewichte minimal ist. Falls nötig können auch Knoten aus $V \backslash T$ (Steiner-Knoten) verwendet werden.

Ein Steiner Baum hat

- Obligatorische Knoten $:= T$
- Fakultative Knoten $:= V \backslash T$ → *Steiner-Knoten*



Lösungsansatz (exakt / naiv)

1. Alle möglichen Kombinationen von Steiner-Knoten prüfen.
2. Für jede Auswahl von Steiner-Knoten wird das entsprechende Problem des *minimalen Gerüsts* gelöst.
3. Unter diesen minimalen Gerüsten, wähle dasjenige mit kleinstem Gewicht.

Bemerkung: Anzahl Kombinationen von Steiner-Knoten wächst exponentiell mit der Anzahl Steiner Knoten $|V - N|$: $2^{|V-N|}$ Auswahlen.

Transformation des Problems

Falls die Anzahl obligatorischer Knoten $|N|$ klein ist, kann der obige Ansatz mittels einer Transformation verbessert werden.

Lemma:

Sei $H = (V, U)$ ein vollständiger Graph, dessen Kantengewichte die Dreiecksungleichung erfüllen $(h_{ij} + h_{jk} \geq h_{ik})$. Dann existiert ein minimaler Steiner-Baum in H, welche höchstens $|N| - 2$ Steiner-Knoten enthält