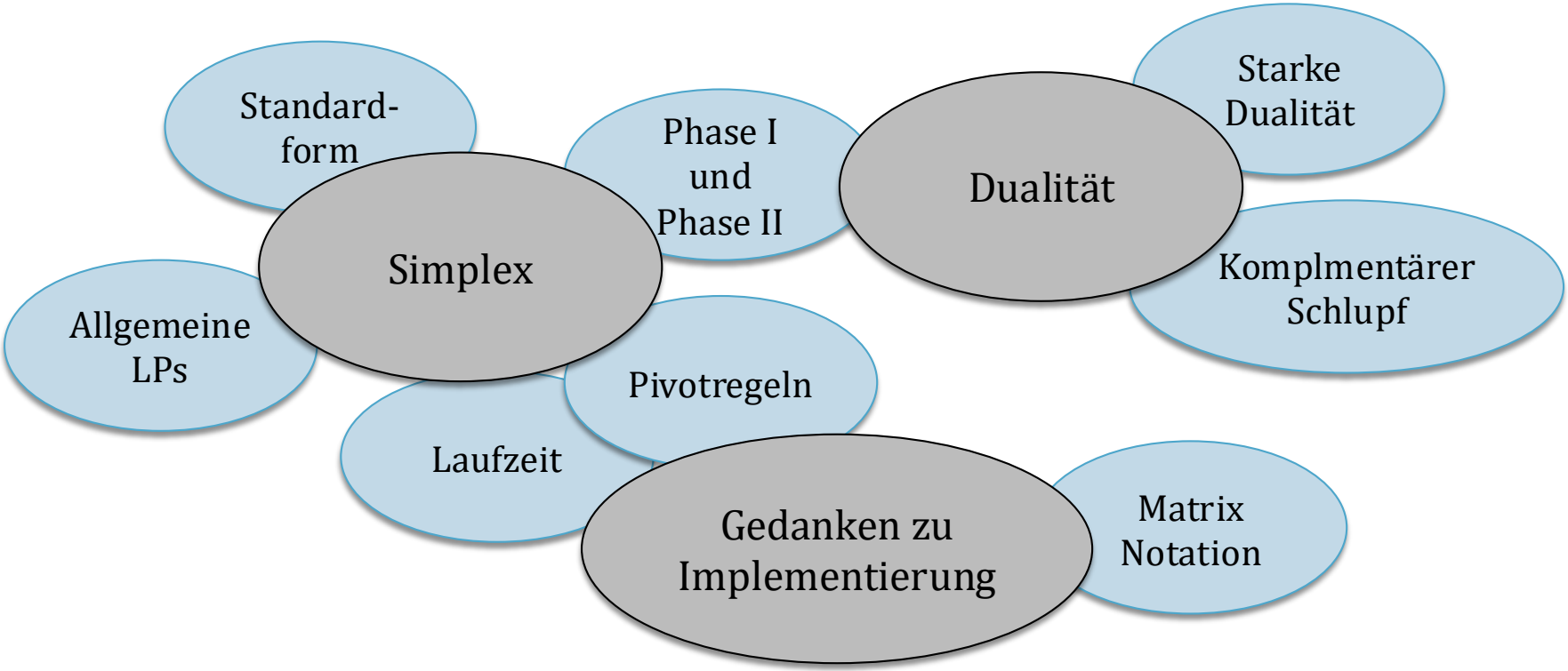
Technische Universität Braunschweig

Mathematische Methoden der Algorithmik

Arne Schmidt

Zusammenfassung

Grober Überblick Part 1

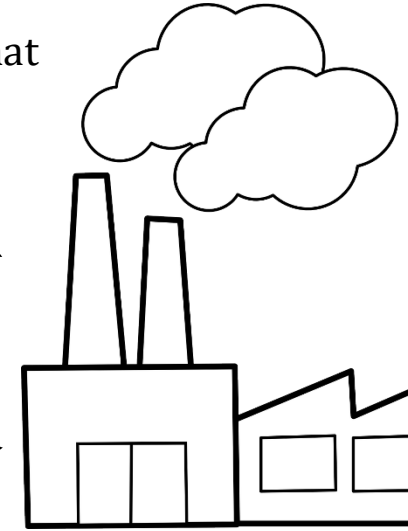


LPs und Simplex

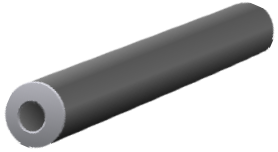
Fabrik – Bestmöglicher Profit?

Eisen

240t pro Monat



Stahlrohre (1000€ / t)

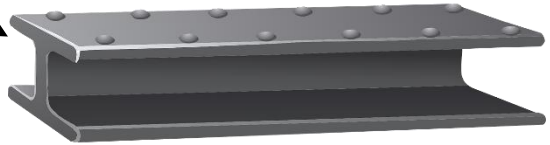


0.8t Eisen + 0.4t Kohle pro t

0.9t Eisen + 0.7t Kohle pro t

Kohle

180t pro Monat



Stahlträger (1400€ / t)

Etwas anders

Maximiere $(1000, 1400) \cdot \begin{pmatrix} Rohre \\ Träger \end{pmatrix}$ **Profit**

$$\begin{pmatrix} 0.8 & 0.9 \\ 0.4 & 0.7 \end{pmatrix} \begin{pmatrix} Rohre \\ Träger \end{pmatrix} \leq \begin{pmatrix} 240 \\ 180 \end{pmatrix}$$

Eisenlimit

$$\begin{pmatrix} Rohre \\ Träger \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Kohlelimit

Allgemein

Maximiere $c^T x$

Unter Bedingungen

$$Ax \leq b$$

$$x \geq 0$$

Feasibility und Optimalität

Eine Belegung von Variablen für ein LP heißt Lösung (**solution**). Weiter
- Eine Lösung ist gültig (**feasible**), wenn alle Constraints erfüllt werden.
- Eine Lösung heißt **optimal**, wenn der Lösungswert dem Maximum entspricht.

Besitzt ein LP keine Lösung, so ist das LP ungültig (**infeasible**).
Besitzt ein LP Lösungen mit beliebig hohen Werten, dann ist das LP unbeschränkt (**unbounded**).

Lemma:
Ist ein LP feasible und bounded, dann hat das LP ein Optimum.

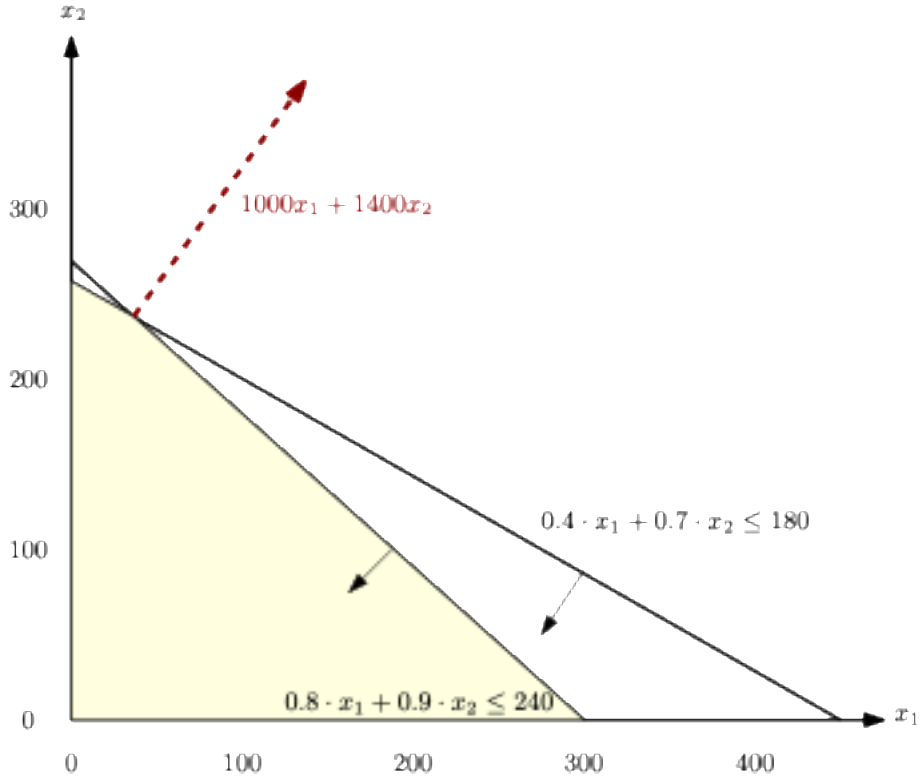
Beweis:
Sei X die Lösungsmenge. Da das LP beschränkt ist, existiert ein Supremum s.
Da c linear ist, wird die abgeschlossene Menge X auf eine abgeschlossene Menge c[X] abgebildet.
Da s entweder in c[X] oder ein Häufungspunkt. Da c[x] abgeschlossen ist, muss s in c[X] sein.

Graphische Darstellung

Maximiere: $(1000, 1400) \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

$$\begin{pmatrix} 0.8 & 0.9 \\ 0.4 & 0.7 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 240 \\ 180 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$



Ein Beispiel-LP mit Slack Variablen

$$
\begin{aligned}
max \quad & \zeta = && 5x_1 + && 4x_2 + && 3x_3 \\
\text{s.t.} \quad & w_1 = && 5 - && 2x_1 - && 3x_2 - && x_3 \\
& w_2 = && 11 - && 4x_1 - && x_2 - && 2x_3 \\
& w_3 = && 8 - && 3x_1 - && 4x_2 - && 2x_3 \\
& && && w_1, w_2, w_3, x_1, x_2, x_3 && \geq 0
\end{aligned}
$$

Bei jeder gültigen Lösung ist die Differenz zwischen der linken und rechten Seite der Ungleichungen nicht-negativ.

$\Rightarrow$ Führe Variablen ein, die diesen Unterschied angeben!

Diese Variablen heißen **Slack-Variablen**.

Idee: Starte nun mit irgendeiner Lösung und schaue, wie diese verbessert werden kann!

Startlösung und erste Verbesserung

$$
\begin{aligned}
max \quad & \zeta = && 5x_1 + && 4x_2 + && 3x_3 \\
\text{s.t.} \quad & w_1 = && 5 - && 2x_1 - && 3x_2 - && x_3 \\
& w_2 = && 11 - && 4x_1 - && x_2 - && 2x_3 \\
& w_3 = && 8 - && 3x_1 - && 4x_2 - && 2x_3 \\
& && w_1, w_2, w_3, x_1, x_2, x_3 && && && \geq 0
\end{aligned}
$$

Eine gültige Lösung ist
$(\dot{x}_1, \dot{x}_2, \dot{x}_3, \dot{w}_1, \dot{w}_2, \dot{w}_3) = (0,0,0,5,11,8)$

Wie können wir das verbessern?

Betrachte bspw. x_1. Um wieviel dürfen wir die Variable erhöhen, sodass alle Slack-Variablen nicht negativ werden?

$w_1 \geq 0 \Rightarrow x_1 \leq \frac{5}{2}$, $\quad w_2 \geq 0 \Rightarrow x_1 \leq \frac{11}{4}$, $\quad w_3 \geq 0 \Rightarrow x_1 \leq \frac{8}{3}$

Setze $x_1 = \frac{5}{2}$, damit wird $w_1 = 0, w_2 = 1, w_3 = \frac{1}{2}$

Schritt 1: LP zu Dictionary

Umwandeln in ein Dictionary:

$$\zeta = \bar{\zeta} + \sum_{j \in \mathcal{N}}^{n} \bar{c}_j x_j$$

$$x_i = \bar{b}_i - \sum_{j=1}^{n} \bar{a}_{ij} x_j \,, \qquad i \in \mathcal{B}$$

Die Koeffizienten im Vektor $\bar{c}$ heißen auch **reduzierte Kosten**.

Sei nun $\mathcal{N}, \mathcal{B} \subseteq \{1, 2, \dots, n+m\}$ die Indexmenge der (Nicht-)Basisvariablen.

Dann ist zu Beginn:
$\mathcal{N} = \{1, 2, \dots, n\}$
$\mathcal{B} = \{n+1, n+2, \dots, n+m\}$

Mit dieser Notation lässt sich das Dictionary noch etwas anders aufschreiben!

Der Balken über Konstanten bedeutet, dass diese sich über Iterationen verändern.

Schritt 2: Pivotschritt

Das Dictionary:

$$\zeta = \bar{\zeta} + \sum_{j \in \mathcal{N}}^{n} \bar{c}_j x_j$$

$$x_i = \bar{b}_i - \sum_{j=1}^{n} \bar{a}_{ij} x_j\,, \qquad i \in \mathcal{B}$$

- Wähle nun $k \in \{j \in \mathcal{N} \mid \bar{c}_j > 0\}$.
- Existiert k nicht, dann sind wir optimal!
- Erhöhe nun x_k so weit wie möglich.

- Es muss für alle $i \in \mathcal{B}$ gelten:
$$x_i \leq \bar{b}_i - \bar{a}_{ik} x_k$$
- Da für ein $i \in \mathcal{B}$ der Wert x_i auf 0 gesetzt wird, erhalten wir $x_k \leq \frac{\bar{b}_i}{\bar{a}_{ik}}$.
- Wähle $\ell \in \mathcal{B}$ mit $\bar{a}_{\ell k} > 0$ und $\frac{\bar{b}_\ell}{\bar{a}_{\ell k}}$ kleinstmöglich.
- Setze $x_\ell = 0, x_k = \frac{\bar{b}_\ell}{\bar{a}_{\ell k}}$
- Passe Dictionary über den Variablentausch an!

Ein weiteres Beispiel

$$
\begin{array}{lrrl}
max & -2x_1 - & x_2 & \\
\text{s.t.} & -x_1 + & x_2 & \leq -1 \\
& -x_1 - & 2x_2 & \leq -2 \\
& & x_2 & \leq 1 \\
& & x_1, x_2 & \geq 0
\end{array}
\quad \longrightarrow \quad
\begin{array}{rrrr}
\zeta = & & -2x_1 - & x_2 \\
w_1 = & -1 + & x_1 - & x_2 \\
w_2 = & -2 + & x_1 + & 2x_2 \\
w_3 = & 1 - & & x_2
\end{array}
$$

Die Lösung (0,0,-1,-2,1) ist nicht gültig!
Das LP hat aber eine Lösung (z.B. $x_1 = 2, x_2 = 0$)

Generell: Das Dictionary ist genau dann gültig, wenn alle $\bar{b}_i > 0$.
Wie bekommen wir ein gültiges Dictionary?

Ein Hilfsproblem

Wir erstellen uns ein Hilfsproblem, für welches

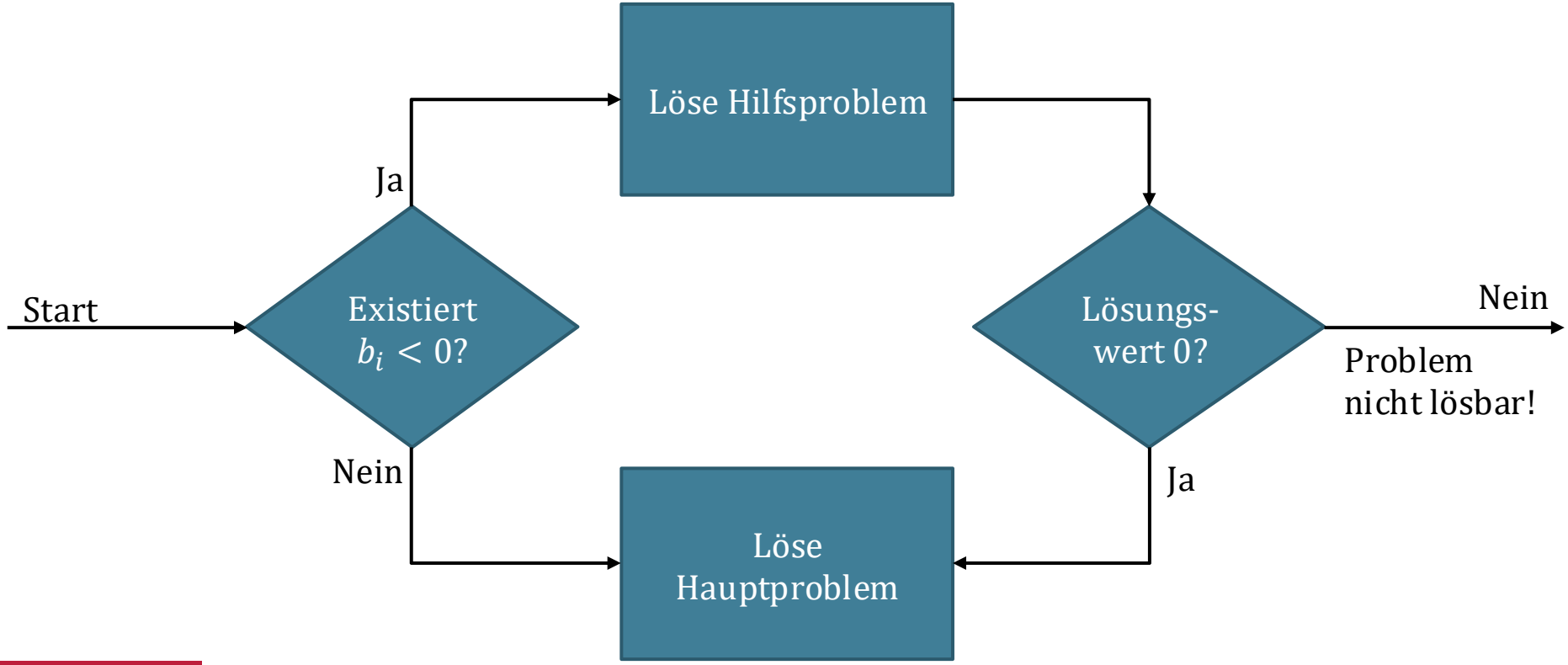
1) ein gültiges Dictionary einfach zu finden ist.
2) eine gültige Lösung für unser originales Problem bereitstellt, oder zeigt, dass keine gültige Lösung existiert

$$\max -x_0$$
$$\text{s.t.}$$
$$\left(\sum_{j=1}^{n} a_{ij} x_j\right) - x_0 \leq b_i, \qquad i \in \{1, 2, \dots, m\}$$
$$x_j \geq 0, \qquad j \in \{0, 1, 2, \dots, n\}$$

Warum hat dieses LP **immer** eine Lösung?

Wann zeigt dieses LP, dass für das **Ursprungsproblem keine Lösung** existiert?

2-Phasen Simplex



Degenerierte Dictionaries und Pivots

$$
\begin{aligned}
\zeta &= 0 + x_1 + x_2 \\
w_1 &= 5 - 2x_1 + 3x_2 \\
w_2 &= 7 - 3x_1 - 7x_2 \\
w_3 &= 0 - x_2
\end{aligned}
$$

$$
\begin{aligned}
\zeta &= 0 + x_1 - w_3 \\
w_1 &= 5 - 2x_1 + 3w_3 \\
w_2 &= 7 - 3x_1 - 7w_3 \\
x_2 &= 0 - w_3
\end{aligned}
$$

Ein solches Dictionary heißt **degeneriert**. Es enthält ein $\bar{b}_i = 0$.

Dadurch entstehen **degenerierte Pivots**: Für ein $k \in \mathcal{B}$ existiert ein $i \in \mathcal{N}$ mit

$$\frac{\bar{b}_i}{\bar{a}_{ik}} = 0$$

Frage: Welche Pivots sind hier degeneriert und welche nicht?

x_1 nicht degeneriert, x_2 ist degeneriert.

Zykeln

Um zykeln zu vermeiden, werden spezielle Regeln eingeführt. Aber sogar die folgende Regel kann zum Zykeln führen:

- Wähle als Nichtbasisvariable die mit dem größten positiven Koeffizienten $\bar{c}_k$.
- Wähle die Basisvariable nach lexikographischer Ordnung, wenn mehrere zur Auswahl stehen. (Dabei in der Regel Schlupfvariablen als letztes, also $x_1, \dots, x_n, w_1, \dots, w_m$)

$$
\begin{array}{rrrrrrr}
\zeta = & & + & x_1 - & 2x_2 & - & 2x_4 \\
\hline
w_1 = & & - & 0.5x_1 + & 3.5x_2 + & 2x_3 - & 4x_4 \\
w_2 = & & - & 0.5x_1 + & x_2 + & 0.5x_3 - & 0.5x_4 \\
w_3 = & 1 - & & x_1 & & &
\end{array}
$$

Probiert es selbst aus; nach 6 Iterationen sollte man wieder bei diesem Dictionary ankommen (ggf. sind die Spalten / Zeilen permutiert).

Pertubation

$$
\begin{aligned}
\zeta &= 0 + 6x_1 + 4x_2 \\
w_1 &= 0 + 9x_1 + 4x_2 \\
w_2 &= 0 - 4x_1 - 2x_2 \\
w_3 &= 1 - x_2
\end{aligned}
$$

Notiz: Man kann auch einfach jede Zeile pertubieren.

$$
\begin{aligned}
\zeta &= 0 + 6x_1 + 4x_2 \\
w_1 &= 0 + \epsilon_1 + 9x_1 + 4x_2 \\
w_2 &= 0 + \epsilon_2 - 4x_1 - 2x_2 \\
w_3 &= 1 - x_2
\end{aligned}
$$

Fundamentalsatz

Theorem (Fundamentalsatz):
Für jedes LP in Standardform, gelten die folgenden Aussagen:

- Wenn es keine optimale Lösung gibt, ist das LP infeasible oder unbeschränkt.
- Wenn es eine Lösung gibt, dann existiert es eine Basislösung.
- Wenn es eine optimale Lösung gibt, dann existiert eine optimale Basislösung.

Andere Pivotregeln

Bland‘s Rule:
Wähle aus allen möglichen Pivots immer den mit dem kleinsten Index.

Simplex zykelt mit dieser Regel nicht! Benötigt aber ggf. länger zum Konvergieren.

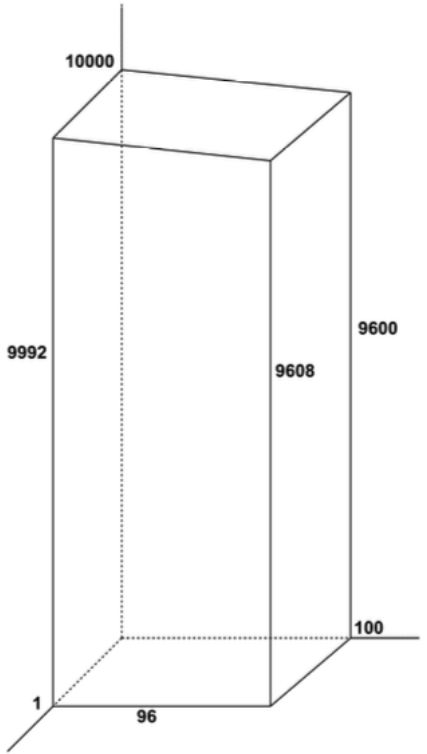
Random Rule:
Wähle aus allen möglichen Pivots immer zufällig einen aus.

Greatest Increase Rule / Steepest Edge:
Pivotisiere so, dass die Zielfunktion am stärksten wächst.

Klee-Minty-Würfel

Mit $\beta_2 = 98, \beta_3 = 9800$ sieht der Lösungsraum so aus:

$$
\begin{aligned}
1x_1 &&&\leq 1\\
4x_1 + &\quad 1x_2 &&\leq 100\\
8x_1 + &\quad 4x_2 + &\quad 1x_3 &\leq 10000\\
x_1, &\quad x_2, &\quad x_3 &\geq 0
\end{aligned}
$$



Lemma:
Mit der Größte-Pivot-Regel werden $2^n - 1$ Iterationen benötigt.

Dualität

Ableiten oberer Schranken

$$
\begin{array}{llrrl}
max & 4x_1 + & x_2 + & 3x_3 & \\
\text{s.t.} & x_1 + & 4x_2 & & \leq 1 \\
& 3x_1 - & x_2 + & x_3 & \leq 3 \\
& & & x_1, x_2, x_3 & \geq 0
\end{array}
$$

$$
\begin{array}{llrrl}
min & y_1 + & 3y_2 & & \\
\text{s.t.} & y_1 + & 3y_2 & & \geq 4 \\
& 4y_1 - & y_2 & & \geq 1 \\
& & y_2 & & \geq 3 \\
& & & y_1, y_2 & \geq 0
\end{array}
$$

Betrachte nun

$$
\begin{array}{l}
y_1 \times (x_1 + 4x_2 \leq 1) \\
+y_2 \times (3x_1 - x_2 + x_3 \leq 3)
\end{array}
$$

$$
(y_1 + 3y_2)x_1 + (4y_1 - y_2)x_2 + y_2 x_3 \leq y_1 + 3y_2
$$

≥ 4 $\quad$ ≥ 1 $\quad$ ≥ 3 $\quad$ Minimiere!

Duales LP

Zu jedem LP existiert ein **duales** LP. Das duale LP eines dualen LPs wird auch **primales** LP genannt. Ist das LP in Standardform, ist die Dualisierung ganz einfach.

Primal

Maximiere $c^T x$

Unter Bedingungen

$$Ax \leq b$$
$$x \geq 0$$

Dual

Minimiere $b^T y$

Unter Bedingungen

$$A^T y \geq c$$
$$y \geq 0$$

Schwache Dualität

Theorem (Schwache Dualität)
Sind x bzw. y zulässige Lösungen eines primalen bzw. dualen LPs, dann gilt

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

Beweis:

$$c^T x = \sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j$$

$$= \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i$$

$$\leq \sum_{i=1}^{m} b_i y_i = b^T y$$

Primal	Dual
Maximiere $c^T x$	Minimiere $b^T y$
Unter Bedingungen $Ax \leq b$ $x \geq 0$	Unter Bedingungen $A^T y \geq c$ $y \geq 0$

Dualität – Theoreme

Theorem (Starke Dualität)
Besitzt das primale LP eine optimale Lösung x^*, dann besitzt auch das duale LP eine optimale Lösung y^* und es gilt

$$c^T x^* = b^T y^*$$

Theorem (Komplementärer Schlupf)
Angenommen, $x = (x_1, \dots, x_n)$ und $y = (y_1, \dots, y_m)$ sind gültige Lösungen für das primale bzw. duale LP. Dann sind diese genau dann optimal, wenn:

1. $\forall j = 1, \dots, n: \; x_j z_j = 0$
2. $\forall i = 1, \dots, m: \; y_i w_i = 0$

Wobei $w = (w_1, \dots, w_m)$ und $z = (z_1, \dots, z_n)$ die zugehörigen primalen bzw. dualen Schlupfvariablen sind.

Dual pivotisieren I

Betrachte komplementären Schlupf: Wenn wir bspw. y_2 in die Basis aufnehmen, muss w_2 den Wert 0 annehmen, geht also aus der Basis heraus!

Durch **negativ-transponierte Matrix A**: Nimm Basisvariable heraus, die die kleinste Konstante besitzt.

$\zeta =$	$0 -$	$1x_1 -$	$1x_2$
$w_1 =$	$4 +$	$2x_1 +$	$1x_2$
$w_2 =$	$-8 +$	$2x_1 -$	$4x_2$
$w_3 =$	$-7 +$	$1x_1 -$	$3x_2$

Infeasible Dictionary!

$-\xi =$	$0 -$	$4y_1 +$	$8y_2 +$	$7y_3$
$z_1 =$	$1 -$	$2y_1 -$	$2y_2 -$	$1y_3$
$z_2 =$	$1 -$	$1y_1 +$	$4y_2 +$	$3y_3$

Feasible Dictionary!

Dual pivotisieren II

Wenn w_2 heraus geht, wer geht in die Basis rein?
Über das duale erhalten wir z_1 oder z_2. Da einer der beiden auf 0 geht, darf das entsprechende x_1 bzw x_2 größer werden!

Hier: Tausche y_2 mit z_1 im dualen Dictionary, also tausche w_2 mit x_1 im primalen Dictionary!

$$\begin{array}{rrrr} \zeta = & 0 - & 1x_1 - & 1x_2 \\ \hline w_1 = & 4 + & 2x_1 + & 1x_2 \\ w_2 = & -8 + & 2x_1 - & 4x_2 \\ w_3 = & -7 + & 1x_1 - & 3x_2 \end{array}$$

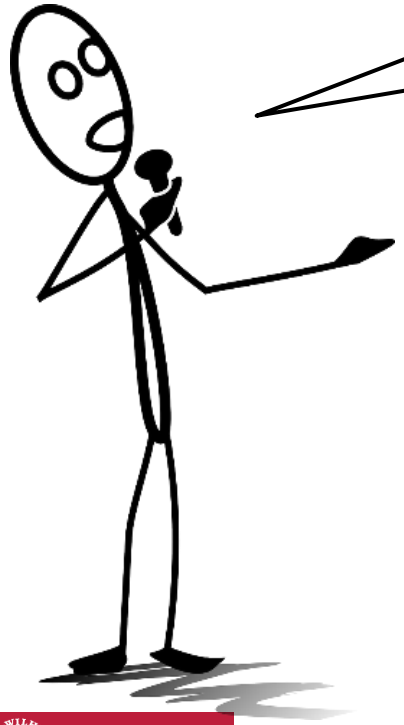
Infeasible Dictionary!

$$\begin{array}{rrrrr} -\xi = & 0 - & 4y_1 + & 8y_2 + & 7y_3 \\ \hline z_1 = & 1 - & 2y_1 - & 2y_2 - & 1y_3 \\ z_2 = & 1 - & 1y_1 + & 4y_2 + & 3y_3 \end{array}$$

Feasible Dictionary!

Größter negativer Quotient $\left(\frac{a_{ij}}{c_j}\right)$

Möglichkeiten für Phase I



Für die Phase I stehen 3 Möglichkeiten zur Verfügung!

Benutze das Hilfsproblem aus Vorlesung 2 und führe primalen Simplex aus.

Ersetze $c_{\mathcal{N}}$ durch einen nicht-positiven Vektor und führe dualen Simplex aus.

Phase II: Primaler Simplex

Ersetze b durch einen nicht-negativen Vektor und führe primalen Simplex aus.

Phase II: Dualer Simplex

LPs

Wie kann man über Simplex erkennen, ob ...

... das primale LP zulässig, unzulässig, oder unbeschränkt ist?

Was erkennt man wie über Phase 1?
Was erkennt man wie über primalen Simplex?
Was erkennt man wie über dualen Simplex?

Simplex in Matrixform

Simplex in Matrixform

- Was kostet an meisten Zeit?
- Wo kann man etwas Zeit sparen?
- Was für Möglichkeiten gibt es?

Primal Simplex	Dual Simplex
Suppose $x^*_{\mathcal{B}} \geq 0$	Suppose $z^*_{\mathcal{N}} \geq 0$
while $(z^*_{\mathcal{N}} \not\geq 0)$ {	while $(x^*_{\mathcal{B}} \not\geq 0)$ {
pick $j \in \{j \in \mathcal{N} : z^*_j < 0\}$	pick $i \in \{i \in \mathcal{B} : x^*_i < 0\}$
$\Delta x_{\mathcal{B}} = B^{-1}Ne_j$	$\Delta z_{\mathcal{N}} = -(B^{-1}N)^T e_i$
$t = \left(\max_{i \in \mathcal{B}} \frac{\Delta x_i}{x^*_i} \right)^{-1}$	$s = \left(\max_{j \in \mathcal{N}} \frac{\Delta z_j}{z^*_j} \right)^{-1}$
pick $i \in \operatorname{argmax}_{i \in \mathcal{B}} \frac{\Delta x_i}{x^*_i}$	pick $j \in \operatorname{argmax}_{j \in \mathcal{N}} \frac{\Delta z_j}{z^*_j}$
$\Delta z_{\mathcal{N}} = -(B^{-1}N)^T e_i$	$\Delta x_{\mathcal{B}} = B^{-1}Ne_j$
$s = \frac{z^*_j}{\Delta z_j}$	$t = \frac{x^*_i}{\Delta x_i}$
$x^*_j \leftarrow t$	$x^*_j \leftarrow t$
$x^*_{\mathcal{B}} \leftarrow x^*_{\mathcal{B}} - t\Delta x_{\mathcal{B}}$	$x^*_{\mathcal{B}} \leftarrow x^*_{\mathcal{B}} - t\Delta x_{\mathcal{B}}$
$z^*_i \leftarrow s$	$z^*_i \leftarrow s$
$z^*_{\mathcal{N}} \leftarrow z^*_{\mathcal{N}} - s\Delta z_{\mathcal{N}}$	$z^*_{\mathcal{N}} \leftarrow z^*_{\mathcal{N}} - s\Delta z_{\mathcal{N}}$
$\mathcal{B} \leftarrow \mathcal{B} \setminus \{i\} \cup \{j\}$	$\mathcal{B} \leftarrow \mathcal{B} \setminus \{i\} \cup \{j\}$
}	}

Idee: LU-Zerlegung

Faktorisiere die $m \times m$-Matrix B in zwei $m \times m$-Matrizen L und U, sodass:

$$L = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & 0 \\ \ell_{2,1} & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \ell_{m,1} & \cdots & \ell_{m,m-1} & 1 \end{pmatrix}}_{\text{Untere Dreiecksmatrix}}, U = \underbrace{\begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,m} \\ 0 & u_{2,2} & \cdots & u_{2,m} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{m,m} \end{pmatrix}}_{\text{Obere Dreiecksmatrix}}$$

Beispiel: Bestimme die LU-Zerlegung von

$$A = \begin{pmatrix} 2 & 4 & 1 \\ 4 & 9 & 3 \\ 1 & 3 & 2 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0.5 & 1 & 1 \end{pmatrix}, U = \begin{pmatrix} 2 & 4 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0.5 \end{pmatrix}$$

Wichtig: $B^T = (LU)^T = U^T L^T$

Berechnen von Gleichungssystemen mit LU

$\Delta x_{\mathcal{B}} = B^{-1}Ne_j = B^{-1}a_j$ ist die Lösung zu $Bx = LUx = a_j$

Sei $y \coloneqq Ux$, dann löse
- erst $Ly = a_j$,
- dann $Ux = y$

$\Delta z_{\mathcal{N}} = -N^T v$, wobei v die Lösung zu $B^T v = (LU)^T v = U^T L^T v = e_j$ ist.

Sei $y \coloneqq L^T v$, dann löse
- erst $U^T y = e_j$,
- dann $L^T v = y$

Sparsity

$$\begin{pmatrix} 1 & 4 & 0 & 1 & 0 \\ 2 & 1 & 2 & 3 & 4 \\ 0 & 4 & 1 & 2 & 1 \\ 2 & 5 & 8 & 3 & 2 \\ 9 & 5 & 0 & 5 & 1 \end{pmatrix}$$

„Voll“ (dense)

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

„Dünn“ (sparse)

Generell: Je mehr 0en eine Matrix besitzt, desto **dünnbesetzter** (sparser) ist die Matrix.

Dünnbesetzte Matrizen lassen sich einfacher und schneller in eine LU-Zerlegung teilen, und benötigen deutlich weniger Speicher.

Ziel: Halte L und U Matrizen so dünnbesetzt wie möglich!

Viele Pivotschritte

Wenn wir k Iterationen durchgeführt haben, können wir den nächsten Schritt über die erste Basis berechnen

Dann ist $B_{new} = BE_0E_1 \dots E_{k-1}$

Für B ist die LU-Zerlegung bekannt und die E-Matrizen sind sehr einfach.

Wenn wir B_{new} nicht explizit berechnen, dauert das ggf. irgendwann sehr lange.

Basis- und Eta-Matrizen

Bei einem Pivotschritt (Basistausch) kann sich nur eine Zeile in B ändern, d.h.

$$B_{new} = B + (a_j - a_i)e_i^T = B\underbrace{(I + B^{-1}(a_j - a_i)e_i^T)}_{E :=}.$$

Also, nach k Iterationen müssen wir folgende Informationen speichern:

- Basismatrix B
- „Eta-Matrizen“ $E_0, \dots, E_{k-1}$ (dafür reicht es, $\Delta x_{\mathcal{B}}^j$ und einen Integer für die Spalte zu merken, die sogenannte eta-file)

Irgendwann sollte man eine neue Basis mit seiner LU-Zerlegung neu berechnen!

Viele Variablen, wenig Constraints → Partial Pricing!

Viele Variablen erzeugen viel Arbeit:

Welches Pivot als nächstes?

Man muss die Länge des Pivotschritts berechnen.

Man muss reduzierte Kosten berechnen.

Müssen wir immer alle Variablen anschauen?

Interessant sind doch nur die „guten“ Variablen.

Allgemeine LP-Darstellung

LPs in allgemeiner Form

$$\begin{aligned}&\max c^T x\\ &\text{s.t.}\\ &\quad a \le Ax \le b\\ &\quad \ell \le x \le u\end{aligned}$$

Sowohl die Contraints als auch die Variablen besitzen eine obere und untere Schranke.

Das verdoppelt aber die Constraints!

$$\begin{aligned}&\max c^T x\\ &\text{s.t.}\\ &\quad Ax \le b\\ &\quad -Ax \le -a\\ &\quad x \le u\\ &\quad x \ge \ell\end{aligned}$$

$$\begin{array}{lrcrcr}
\text{maximize} & & & 3x_1 - & x_2 & \\
\text{subject to} & 1 & \le & -x_1 + & x_2 & \le 5\\
& 2 & \le & -3x_1 + & 2x_2 & \le 10\\
& & & 2x_1 - & x_2 & \le 0\\
& -2 & \le & x_1 & & \\
& 0 & \le & & x_2 & \le 6
\end{array}$$

l		$\boxed{-2}$	$\boxed{0}$
u		∞	6
	$\zeta =$	$3x_1 -$	$x_2 = -6$
1 5	$w_1 =$	$-x_1 +$	$x_2 = 2$
2 10	$w_2 =$	$-3x_1 + 2x_2 =$	6
$-\infty$ 0	$w_3 =$	$2x_1 -$	$x_2 = -4$

Betrachte, was passiert, wenn Nicht-Basisvariablen sich in Richtung ihrer anderen Schranke bewegen!

Ein Dictionary ist degeneriert, wenn eine Basisvariable eine seiner Schranken annimmt.

Part II – Integer Programming

Definitionen und Sätze

Ein LP, welches nur ganzzahlige Variablen enthält, heißt **Integer Program (IP)**.

LPs mit ganzzahligen und reellen Variablen, werden **Mixed Integer Program (MIP)** genannt.

LP mit nur 0-1-Variablen, werden **0-1-Program** oder **Binary Program (BP)** genannt.

Theorem:
Das Lösen von IPs, MIPs und BPs ist NP-schwer.

Ein LP, welches entsteht, wenn man die Ganzzahligkeit aus einem (M)IP entfernt, heißt **Lineare Relaxierung des (M)IPs**.

Lemma:
Sei I ein (M)IP und $\bar{I}$ das dazugehörige relaxierte LP für ein Maximierungsproblem. Dann ist der Lösungswert von I höchstens dem Lösungswert von $\bar{I}$.

Grundidee - Branch

Das Lösen von LPs ist einfach. Löse also zunächst die Relaxierung.

Nimm dann eine nicht-ganzzahlige Variable x_i mit Wert a.

Für eine optimale, ganzzahlige Lösung muss dann gelten:
$\boldsymbol{x_i \leq \lfloor a \rfloor}$ **oder** $\boldsymbol{x_i \geq \lceil a \rceil}$

Prüfe beide Optionen!

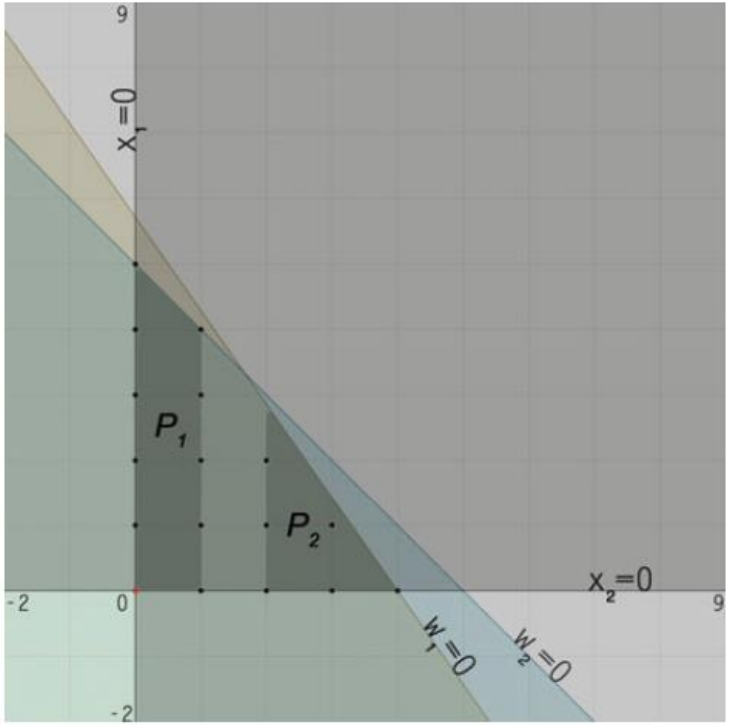
Beispiel (2)

$$
\begin{aligned}
\text{maximize} \quad & 17x_1 + 12x_2 \\
\text{subject to} \quad & 10x_1 + 7x_2 \leq 40 \\
& x_1 + x_2 \leq 5 \\
& x_1,\ x_2 \geq 0 \\
& x_1,\ x_2 \text{ integers}
\end{aligned}
$$

Optimale Lösung:

$$x = \left(\frac{5}{3}, \frac{10}{3}\right)$$

Betrachte Subprobleme mit
$x_1 \leq 2$ und $x_1 \geq 3$



Branch-and-Bound-Idee

Also:

- Löse die LP Relaxierung unter den aktuell gebranchten Variablenbedingunen.
- Entscheide, ob und wie gebrancht werden muss.

Dazu:

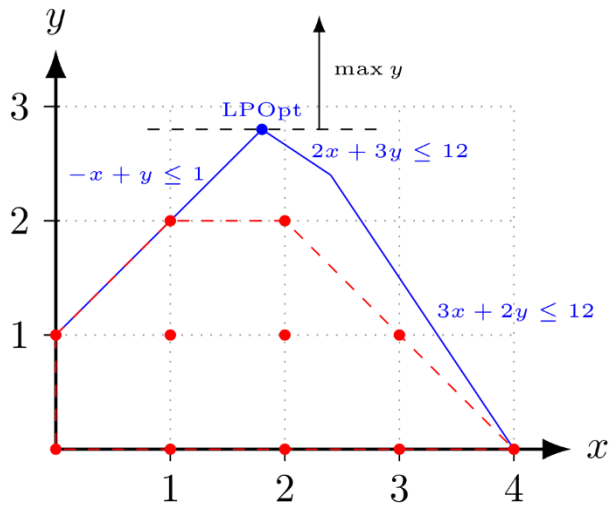
- Gehe Knoten DFS-basiert durch. Das erfordert das Speichern von nur O(Tiefe) vielen Knoten.
- BFS würde pro Level $\Omega(2^{level})$ Knoten speichern.
- Wie können wir weiter Arbeit sparen?
  - Schneide den Teilbäume so früh wie möglich ab („prune“).
  - Benutze Lösung des Vaterknotens wieder („Warmstart“).

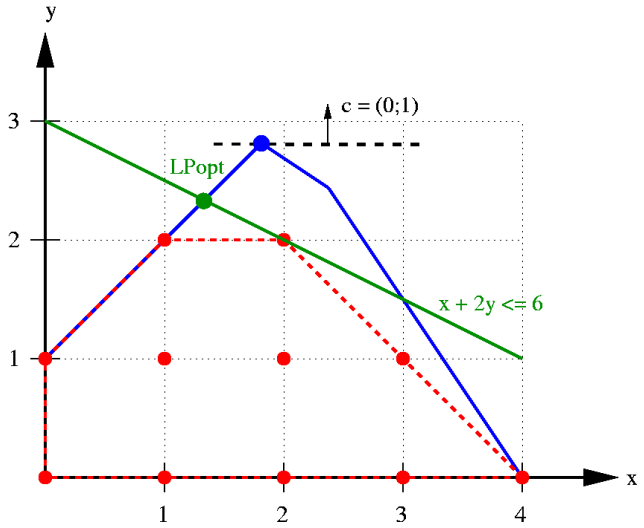
Branch-and-Cut

Cutting Planes

Eine **Cutting Plane** C (oder kurz **Cut**) ist ein Constraint für LP P, welcher folgende Bedingungen erfüllt

- Jede integrale Lösunge zu P ist eine gültige Lösung in $P \cup C$.
- Die optimale Lösung zu P ist ungültig in $P \cup C$.





Gomory-Cuts (2)

$$x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor x_i^* \rfloor = (x_i^* - \lfloor x_i^* \rfloor) - \sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j$$

< 1

≥ 0 für $x \geq 0$

$\in \mathbb{Z}$

< 1

Also können wir festhalten:

$$x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor x_i^* \rfloor \leq 0$$

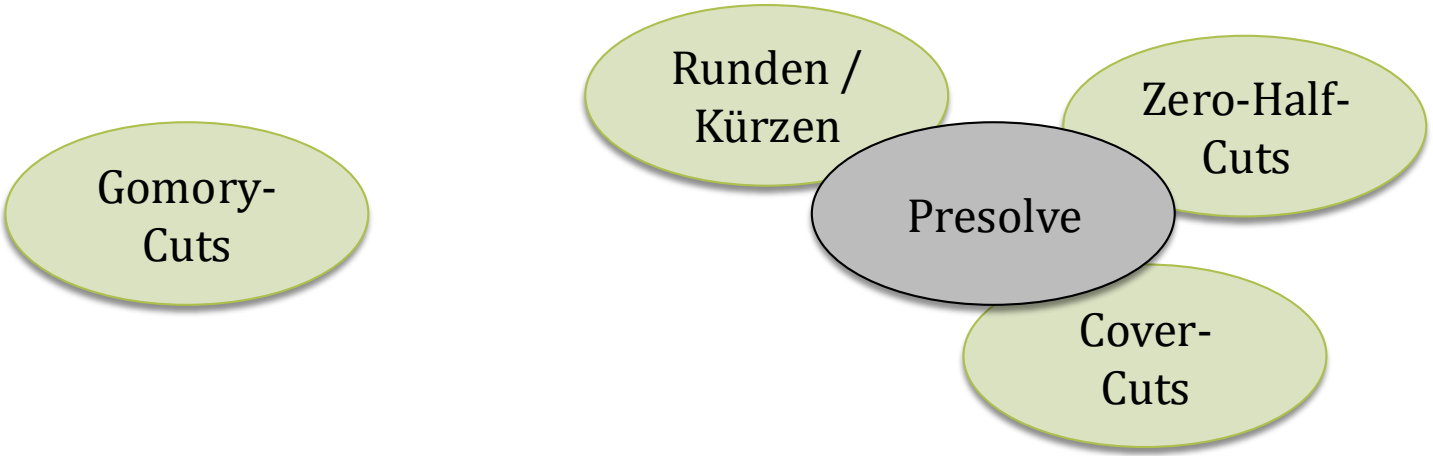
$$\Rightarrow x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor x_i^* \rfloor$$

Damit haben wir einen neuen Constraint!

Branch-and-Cut

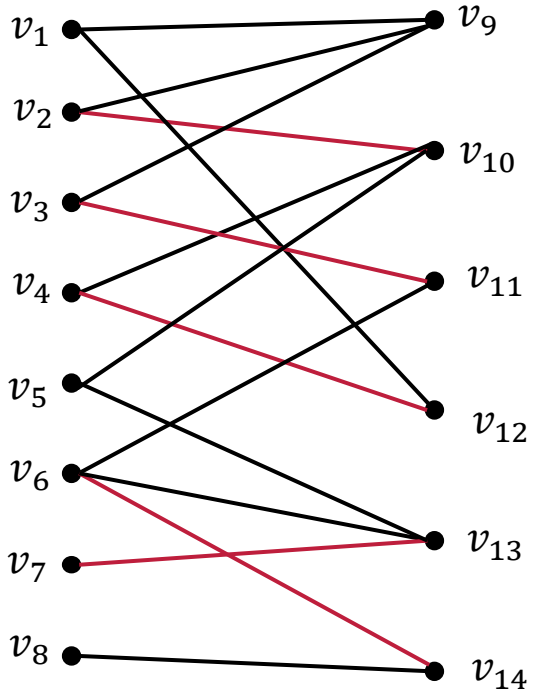
1. Initialisiere eine (Priority-)Queue Q mit P_0
2. Initialisiere B und v_B (Beste bekannte Lösung und Lösungswert); Oder setze $v_B = -\infty, B = \perp$
3. Setze Menge von globalen Cuts $C = \emptyset$
4. Wiederhole, solange Q nicht leer:
   1. Nimm P_i aus Q
   2. Setze Menge lokaler Cuts $C' = \emptyset$
   3. Wiederhole „so lange wie es sinnvoll erscheint“
      1. Bestimme optimale Lösung x_i mit Wert ζ_i der LP-Relaxierung von $P_i \cup C \cup C'$
      2. Falls P_i infeasible, oder $\zeta_i \leq v_B$, starte nächste Iteration.
      3. Falls x_i ganzzahlig ist, setze $v_B = \zeta_i, B = x_i$ und starte nächste Iteration.
      4. Suche einen guten globalen / lokalen Cut (a,b) mit $a^T x_i > b$
      5. Falls gefunden, füge (a,b) zu C bzw C' hinzu.
   4. Wähle nicht-ganzzahligen Wert $\hat{x}$ mit Wert a.
   5. Füge Probleme $P_{i+1} \coloneqq P_i \cup \{\hat{x} \leq \lfloor a \rfloor\} \cup C'$ und $P_{i+2} \coloneqq P_i \cup \{\hat{x} \leq \lceil a \rceil\} \cup C'$ zu Q hinzu.
5. Gib (B, v_B) zurück.

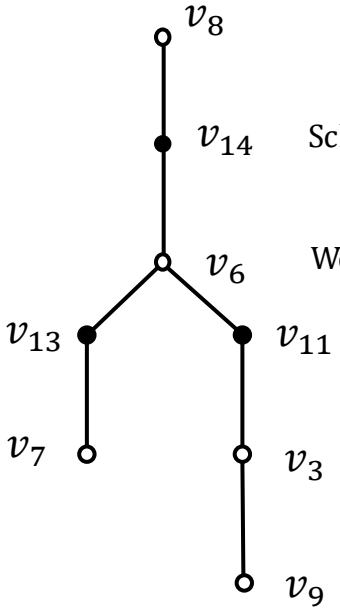
Cutting Plane - Techniken



Verschiedene Probleme

Matchings – Bipartite Graphen





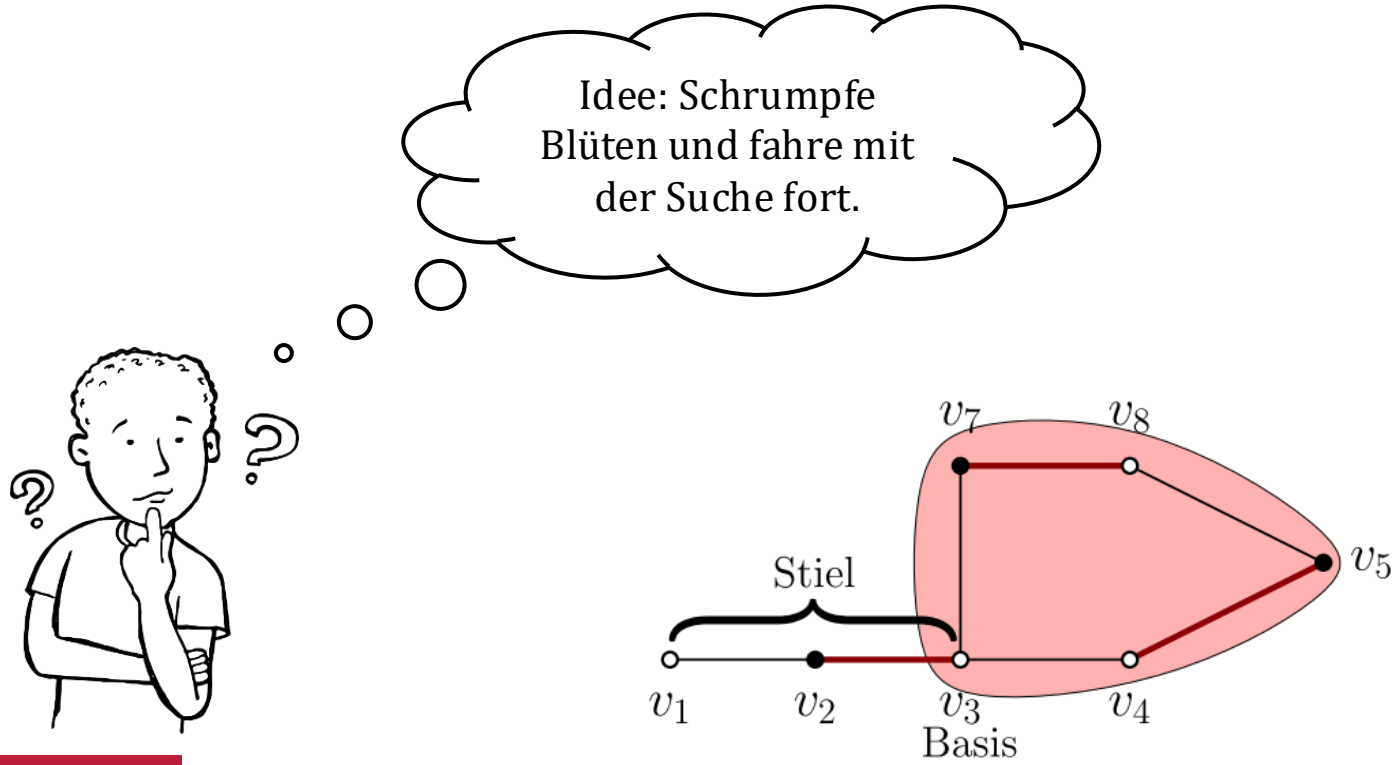
Alternierender Baum

Weiß: v_8 muss gematcht werden.

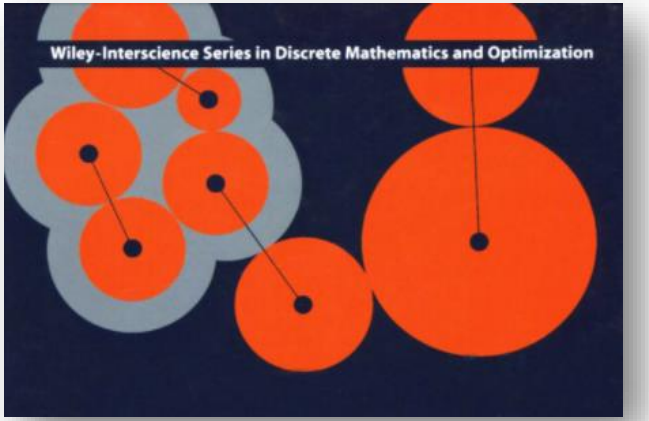
Schwarz: v_{14} hat dann neuen Partner.

Weiß: v_6 muss neu gematcht werden.

Blüten in allgemeinen Graphen – Shrink it!



Gewichtete Matchings – Das duale Problem



Lass Kreise um Knoten
möglichst groß wachsen,
sodass sie sich nicht schneiden.

$$\max \sum_{v \in V} y_v + \sum_{B \in \mathrm{odd}(V)} y_B$$

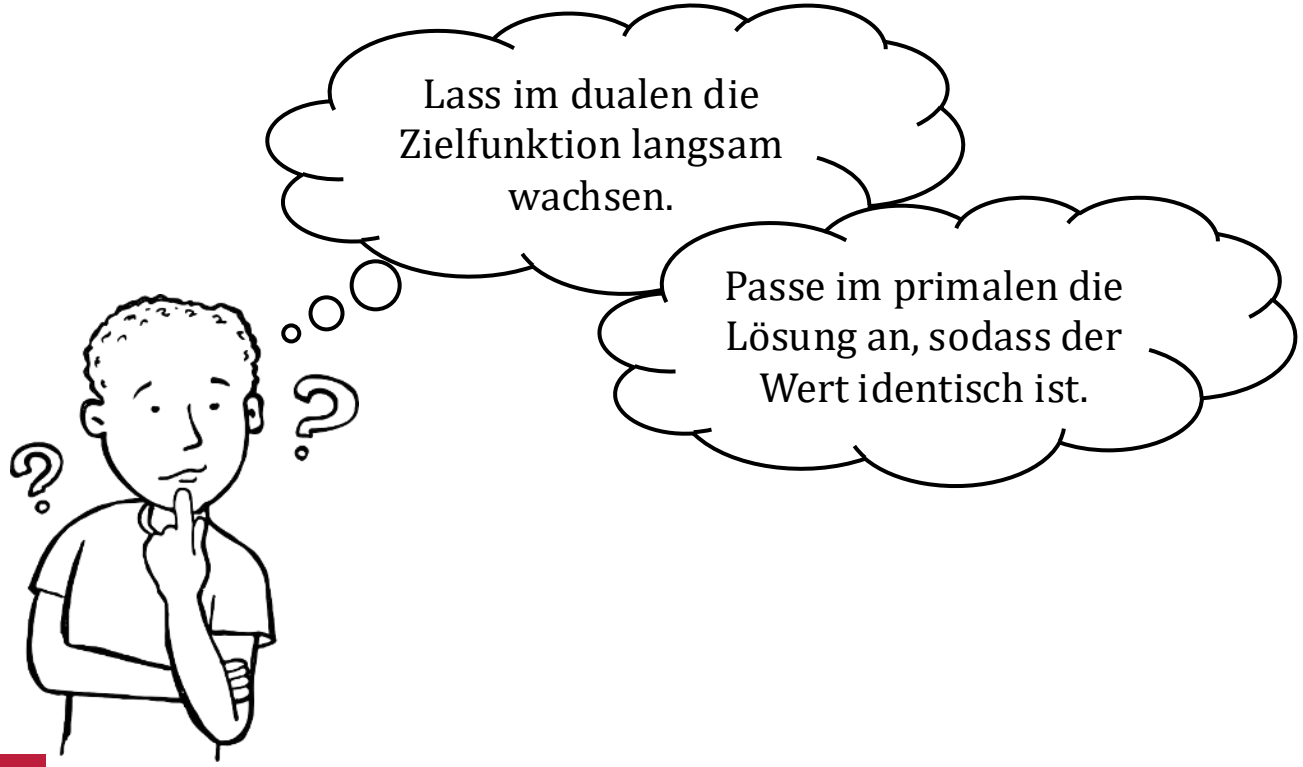
s. t.

$$y_v + y_w + \sum_{\substack{e \in E(B, V \setminus B) \\ B \in \mathrm{odd}(V)}} y_B \leq c(e), \qquad \forall e \in E$$

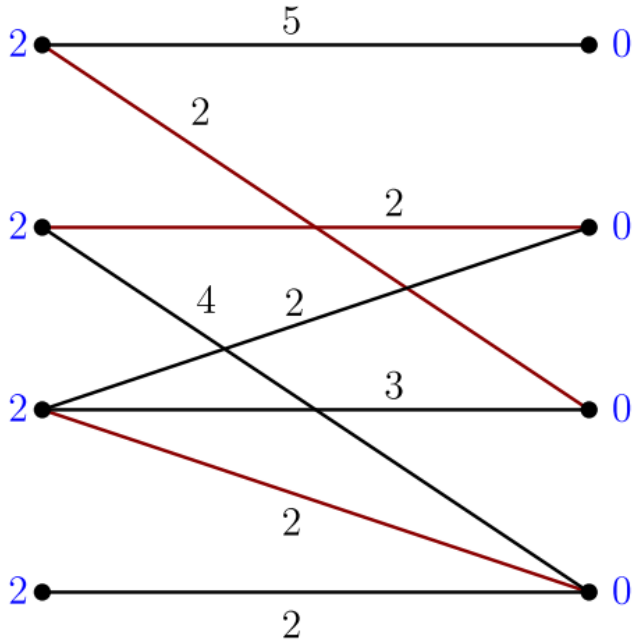
$$y_B \geq 0, \qquad \forall B \in \mathrm{odd}(V)$$

Kanten dürfen nur dann im Matching sein, wenn die Kreise sich berühren!

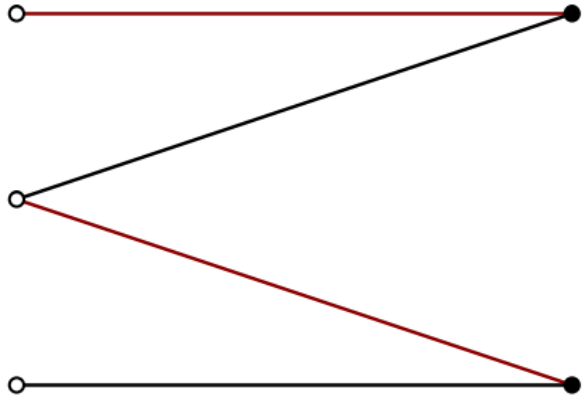
Idee – Primal-Dual-Algorithmus



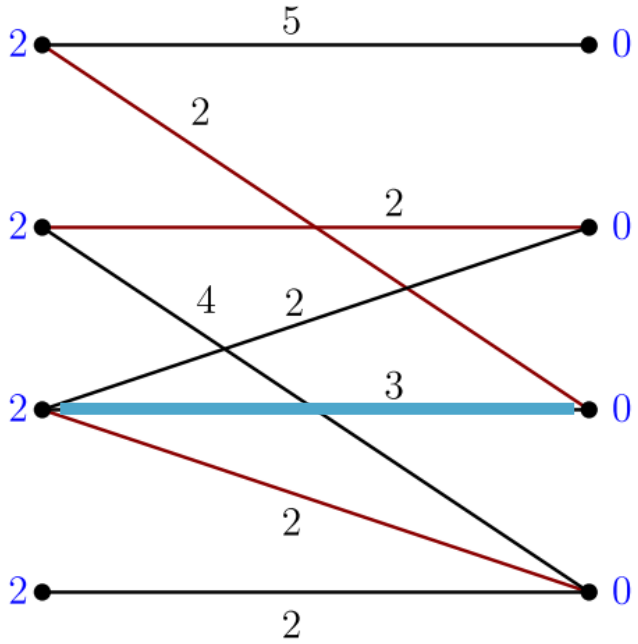
Finden von augmentierenden Pfaden



Konstruiere alternierenden Baum nur über überdeckte Kanten.



Aktualisieren der Dual-Werte



Suche kleinste reduzierte Kosten von Kanten, die herausführen.

$\Rightarrow 1$

Erhöhe/Verringere Werte von Knoten um 1.

Aber wie?

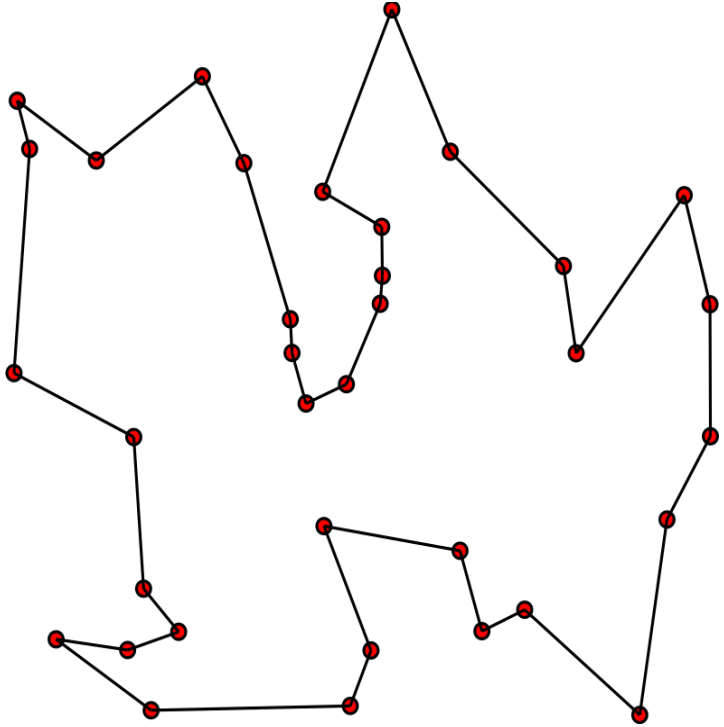
Ein Reisender

Ein Reisender möchte alle großen Städte eines Landes besuchen, dabei aber so wenig wie möglich Zeit mit dem tatsächlichen Reisen verbringen.

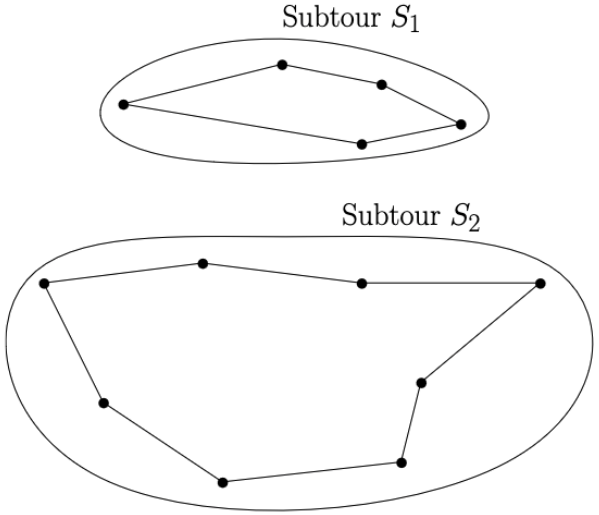
Klar ist:
Der Reisende kann sich nicht aufteilen.

Also:
Entweder gehen wir von Stadt x zu Stadt y, oder wir lassen es sein.

Wie kann der Reisende nun eine kürzeste Tour bestimmen?



TSP-Formulierung – Danzig, Fulkerson, Johnson



Degree-Constraint

Subtour-Constraint

$$\min \sum_{e \in P \times P} c_e x_e$$

s.t.

$$\sum_{e \in \delta(p)} x_e = 2, \qquad \forall p \in P$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall \emptyset \neq S \backslash \subsetneq P$$

$$x_e \in \{0,1\}$$

Lazy Constraints

Problem:
Das IP hat exponentiell viele Constraints!

Wir brauchen aber u.U. nicht alle.

- Löse zunächst das IP ohne die Subtour-Constraints.
- Nach Erhalt der Lösung, prüfe, ob Constraints verletzt sind. Füge verletzte Subtour-Constraints hinzu!

$$\min \sum_{e \in P \times P} c_e x_e$$

s.t.

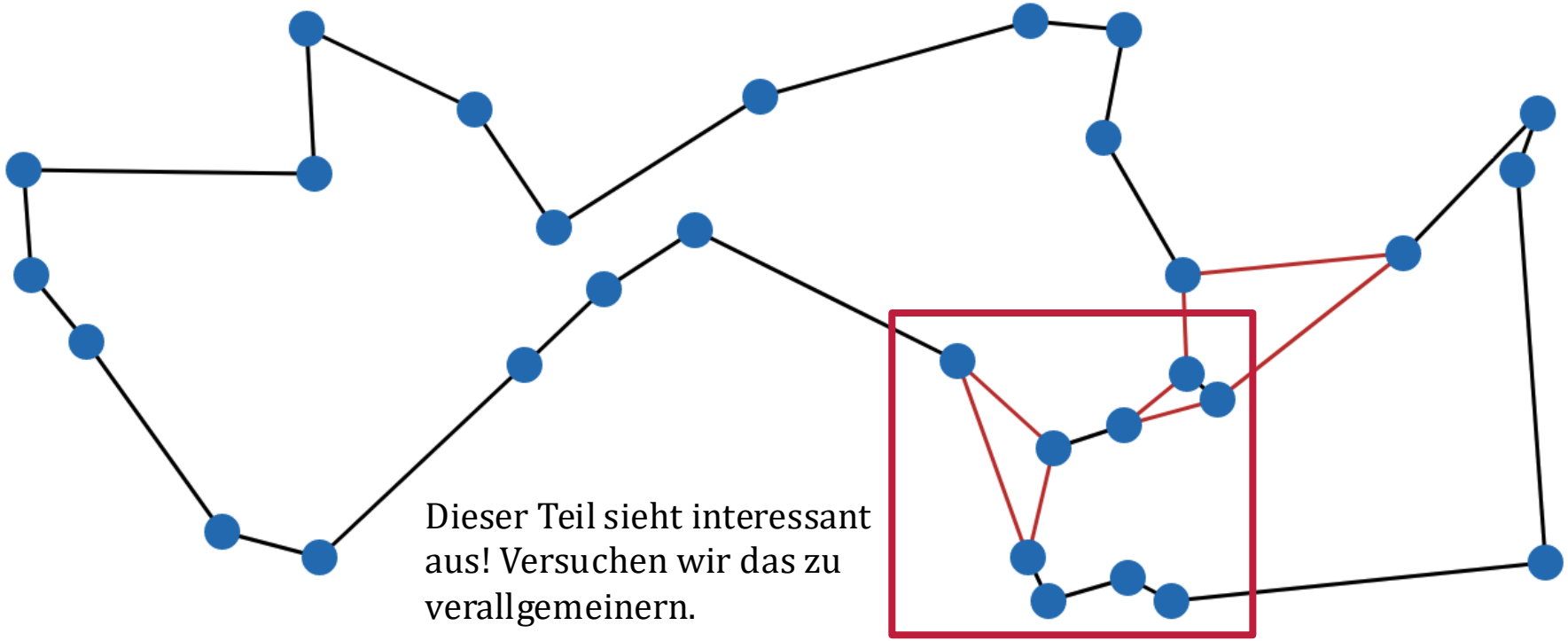
$$\sum_{e \in \delta(p)} x_e = 2, \qquad \forall p \in P$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall \emptyset \neq S \subsetneq P$$

$$x_e \in \{0,1\}$$

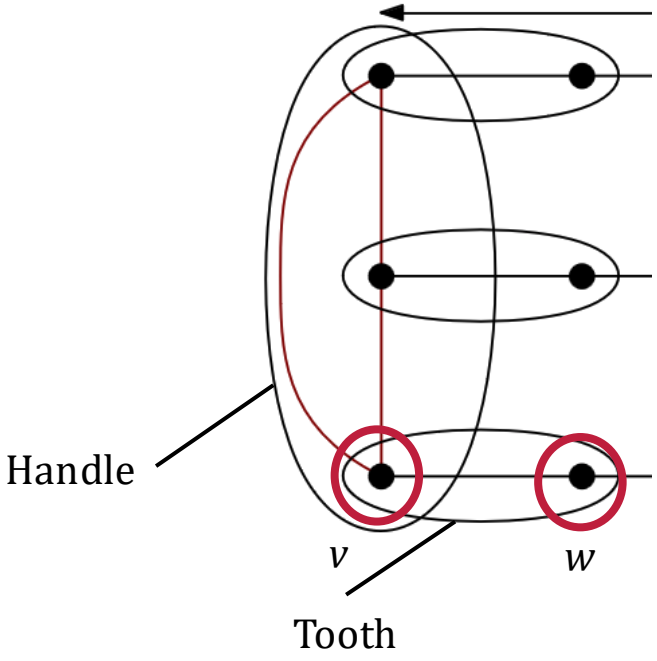
Wir fügen Constraints also nur hinzu, wenn es unbedingt sein muss. Die Constraints werden auch „lazy constraints“ genannt.

TSP – Cutting Planes



Dieser Teil sieht interessant aus! Versuchen wir das zu verallgemeinern.

Comb-Ungleichungen



Damit erhalten wir als Constraint:

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e \geq 3k + 1$$

Pricing

Betrachte eine Teilmenge von Kanten.

Nimm Kanten nach und nach auf, wenn sie für duale Unzulässigkeit sorgen

Lösche ggf. Kanten / Constraints wieder heraus, wenn sie lange nicht mehr betrachtet wurden.