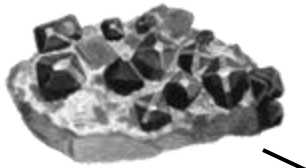
Technische Universität Braunschweig

# Algorithmen und Datenstrukturen 2

Arne Schmidt

# Kapitel 3.1 – Lineare Programme

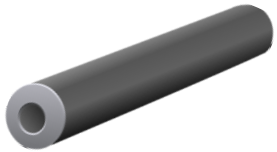
# Fabrik – Bestmöglicher Profit?



Eisen

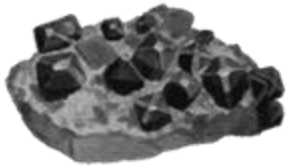
240t pro Monat

Stahlrohre (1000€ / t)



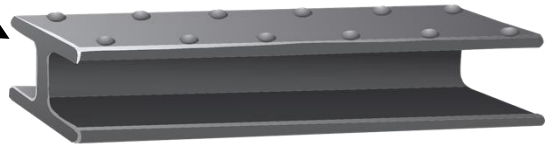
0.8t Eisen + 0.4t Kohle pro t

0.9t Eisen + 0.7t Kohle pro t



Kohle

180t pro Monat



Stahlträger (1400€ / t)

# Etwas mathematischer

Maximiere: $1000 * \text{Rohre} + 1400 * \text{Träger}$ **Profit**

$0.8 * \text{Rohre} + 0.9 * \text{Träger} \leq 240$ **Eisenlimit**

$0.4 * \text{Rohre} + 0.7 * \text{Träger} \leq 180$ **Kohlelimit**

Reicht das zur Beschreibung?

$\text{Rohre} \geq 0$
$\text{Träger} \geq 0$

**Eine Lösung:**
Rohre = 0
Träger = 257,14285…
Umsatz = 360000

**Andere Lösung:**
Rohre = 30
Träger = 240
Umsatz = 366000

Geht das besser?

# 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

$$Ax \leq b$$

$$x \geq 0$$

# Definition

Maximiere $c^T x$

Unter Bedingungen

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

Ein lineares Programm (LP) besteht aus einer linearen Zielfunktion (**objective function**) $\zeta = c_1x_1 + c_2x_2 + \cdots + c_nx_n = c^T x$, welche maximiert oder minimiert werden soll.
Dabei ist:

- c der **Kostenvektor**
- x ein **Variablenvektor** (Entscheidungsvariablen)

Zusätzlich besitzt ein LP $m$ Nebenbedingungen (Constraints) in der Form

$$Ax \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b$$

Dabei ist

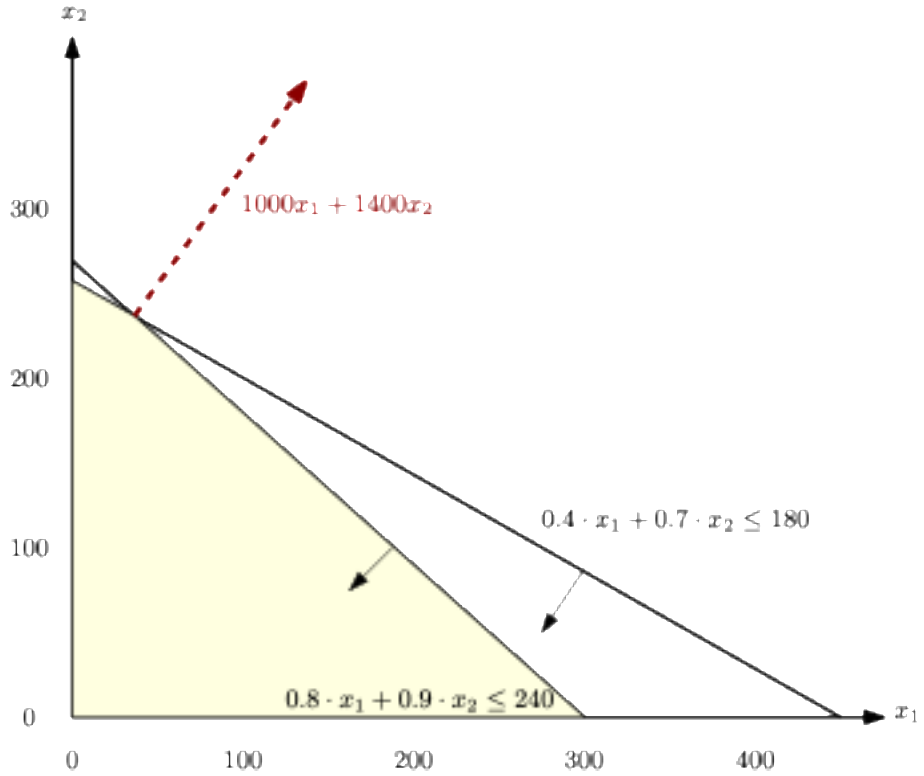
- A eine $m \times n$-**Koeffizientenmatrix**
- b ein n-dimensionaler **Vektor von Konstanten**

# Graphische Darstellung

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

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

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

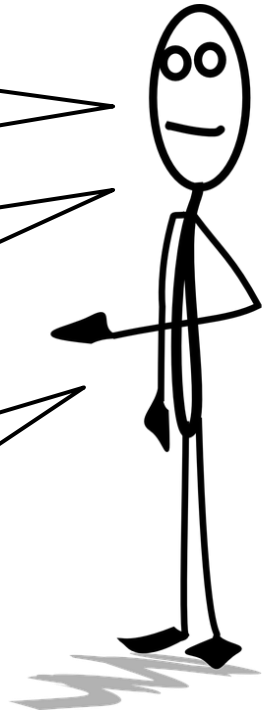


# Lösen von linearen Programmen

Solche linearen Ungleichungssysteme können effizient optimiert werden.

In der Praxis nutzt man oft den Simplex-Algorithmus, der theoretisch exponentielle Laufzeit haben **kann**.

Wir nutzen einen solchen „Solver“ im Folgenden als Black-Box.

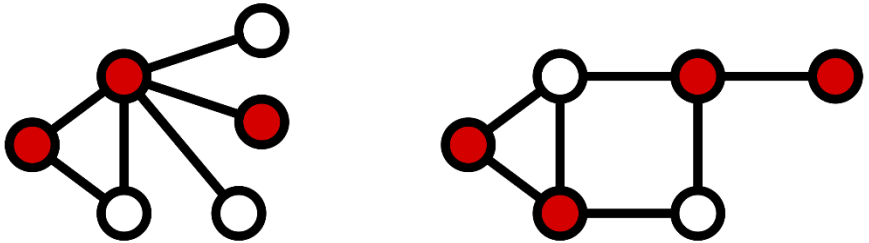


# Kapitel 3.2 – Integer Programming

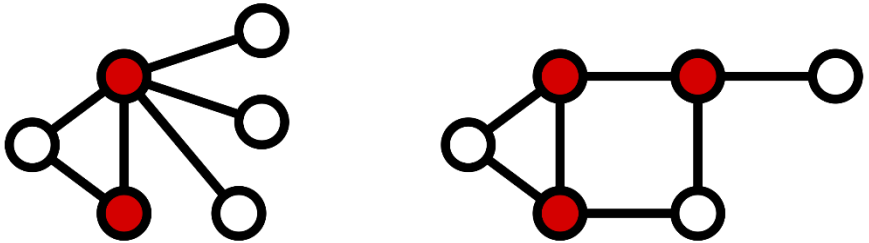
# Ein anderes Problem: Minimum Vertex Cover

**Gegeben**: Ein Graph $G = (V, E)$.
**Gesucht**: Eine kleinste Menge $VC \subseteq V$, sodass für jede Kante $e = \{u, v\}$ entweder $u \in VC$ oder $v \in VC$.

Vertex Cover



Minimum Vertex Cover



# Minimum Vertex Cover als LP

$$\min \sum_{v \in V} x_v$$

Minimiere Anzahl der ausgewählten Knoten.

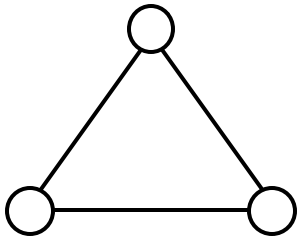
s.t.

$x_v + x_w \geq 1, \qquad \forall e = \{v, w\} \in V$

Jede Kante muss überdeckt sein.

$1 \geq x_v \geq 0, \qquad \forall v \in V$

Jeder Knoten darf maximal einmal ausgewählt sein.



Wie lautet eine optimale Lösung des LPs für diesen Graphen?
=> Jeder Knoten wird zur Hälfte gewählt!

# Minimum Vertex Cover als IP

$$\min \sum_{v \in V} x_v$$

Minimiere Anzahl der ausgewählten Knoten.

s.t.

$x_v + x_w \geq 1, \quad \forall e = \{v, w\} \in V$

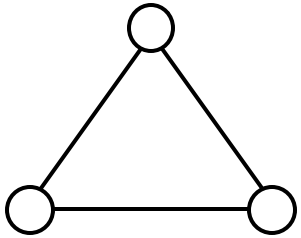
Jede Kante muss überdeckt sein.

$1 \geq x_v \geq 0, \quad \forall v \in V$

Jeder Knoten darf maximal einmal ausgewählt sein.

$x_v \in \mathbb{Z}, \quad \forall v \in V$

Nur ganzzahlige Werte sind erlaubt!



Wie lautet jetzt eine optimale Lösung für diesen Graphen?
=> Zwei Knoten werden gewählt!

# Lösen von ganzzahligen linearen Programmen

Solche IPs können sehr wahrscheinlich nicht effizient gelöst werden.

Das Lösen eines IPs ist „NP-schwer“ (demnächst mehr dazu!)

# Relaxierung

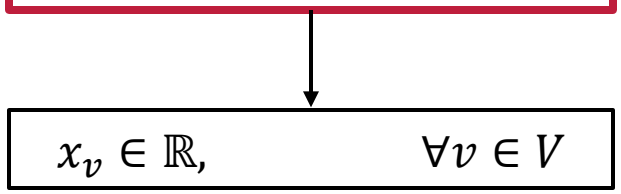
$$\min \sum_{v \in V} x_v$$

s.t.

$x_v + x_w \geq 1, \qquad \forall e = \{v, w\} \in V$

$1 \geq x_v \geq 0, \qquad \forall v \in V$

$x_v \in \mathbb{Z}, \qquad \forall v \in V$



$x_v \in \mathbb{R}, \qquad \forall v \in V$

Das macht uns das Leben schwer!

Damit können wir das einfacher Lösen.
Wir **relaxieren** das Problem.
Es wird dann auch oft die **LP-Relaxierung** genannt.

# Der Blick zurück

Moment mal. Das kommt doch bekannt vor!

Für Maximum Knapsack machen wir das Gleiche!

Löse das fraktionale Problem und nutze dann Branch-and-Bound!

Wie können wir allgemein Branchen?

Einfach irgendeine Variable?

Variablen müssen nicht nur binär sein!

# 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!

# Grundidee – Branch: Etwas allgemeiner

Angenommen, wir haben ein Subproblem $P_i$, dessen LP Relaxierung eine nicht-ganzzahlige, optimale Lösung besitzt.

Splitte $P_i$ in $k$ neue Subprobleme $P_{i+1}, \dots, P_{i+k}$, sodass

1. Jede ganzzahlige Lösung von $P_i$ ist in einem der Subprobleme $P_{i+1}, \dots, P_{i+k}$ enthalten.
2. Keines der Subprobleme $P_{i+1}, \dots, P_{i+k}$ enthält die nicht-ganzzahlige Lösung von $P_i$.
3. (Optional) Die Lösungsräume aller Subprobleme $P_{i+1}, \dots, P_{i+k}$ sind disjunkt.

Am einfachsten:
Erstelle zwei neue Subprobleme, in welchen für ein $x$ mit Wert $a$ die Constraints $x \leq \lfloor a \rfloor$ bzw. $x \geq \lceil a \rceil$ für die Probleme $P_{i+1}$ bzw. $P_{i+2}$ hinzugefügt werden.

Die beste Lösung von $P_i$ ist dann die beste Lösung, die rekursiv in den Subproblemen $P_{i+1}, \dots, P_{i+k}$ gefunden wird.

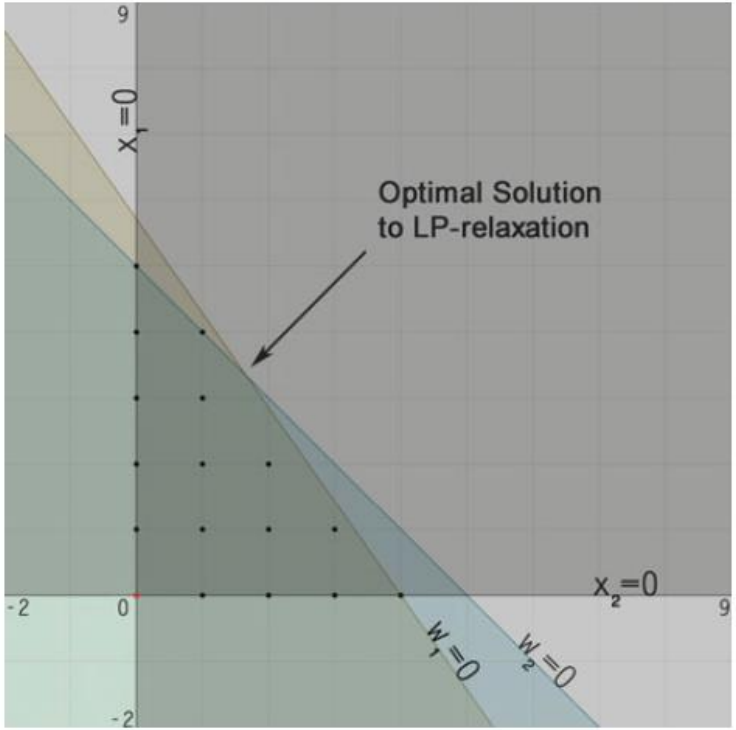
# Beispiel (1)

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

Optimale Lösung der Relaxierung:

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

Betrachte Subprobleme mit
$x_1 \leq 1$ und $x_1 \geq 2$



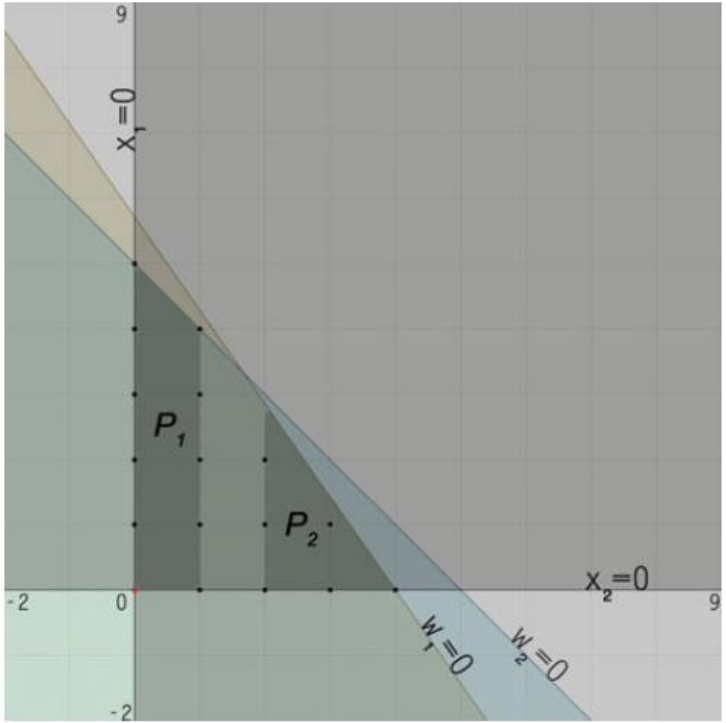
# Beispiel (2)

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

Optimale Lösung der Relaxierung:

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

Betrachte Subprobleme mit
$x_1 \leq 1$ und $x_1 \geq 2$

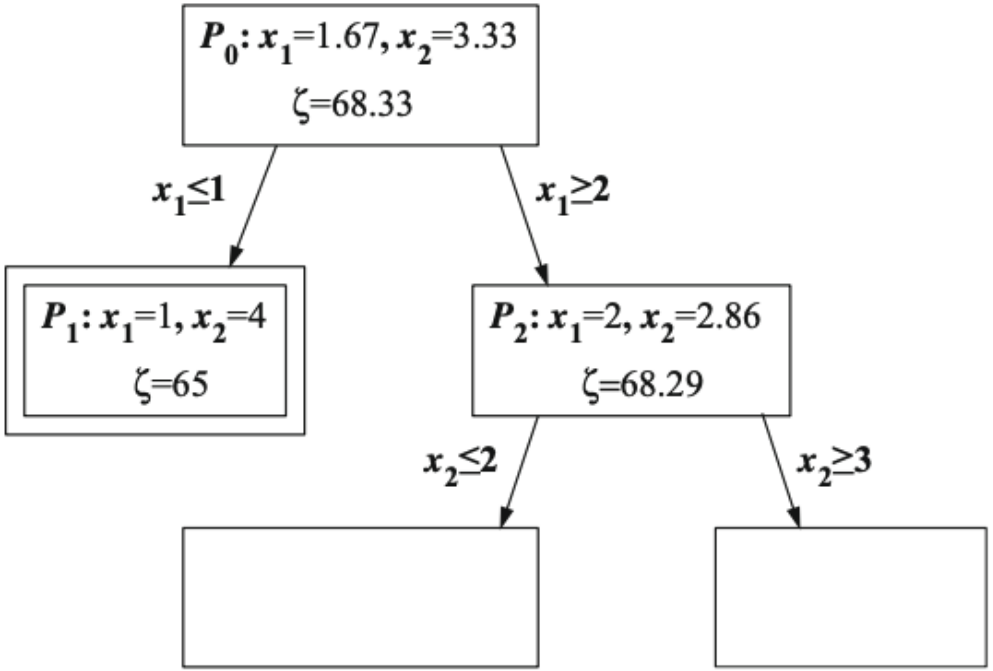


# Beispiel (3)

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

Wir bauen uns nach und nach einen **Enumerationsbaum** auf.

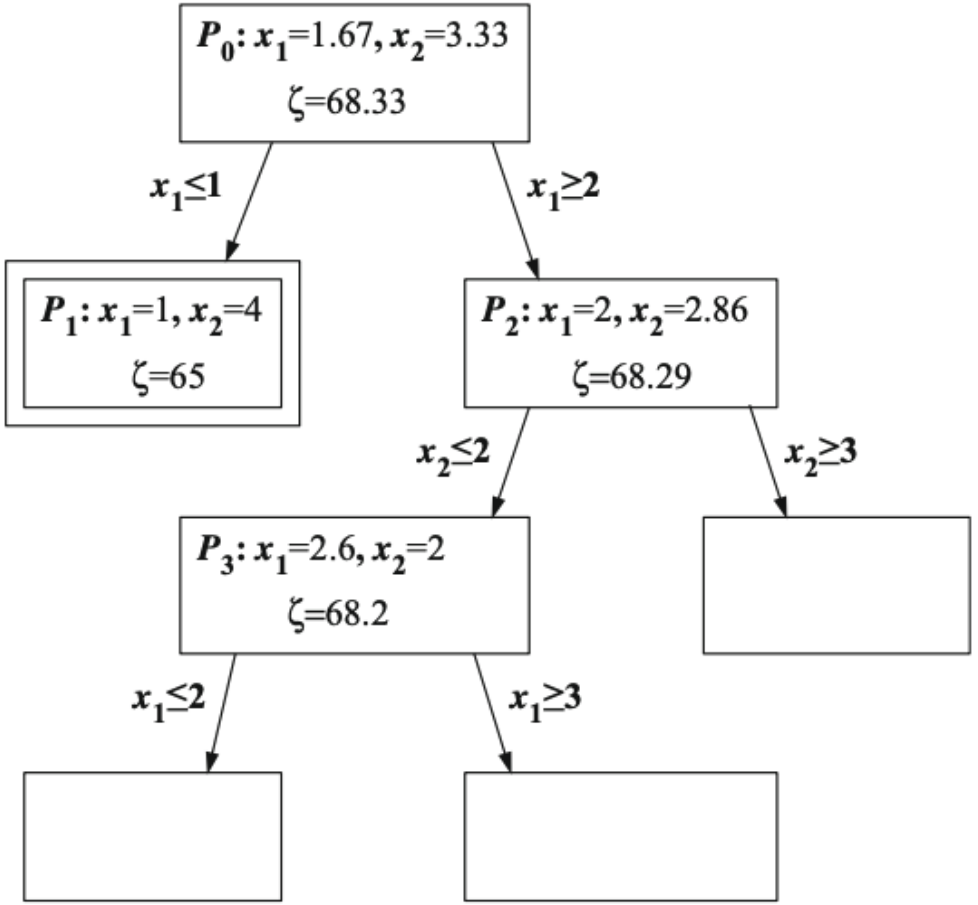
P1 besitzt ganzzahlige, optimale Lösung; P2 nicht.



# Beispiel (4)

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

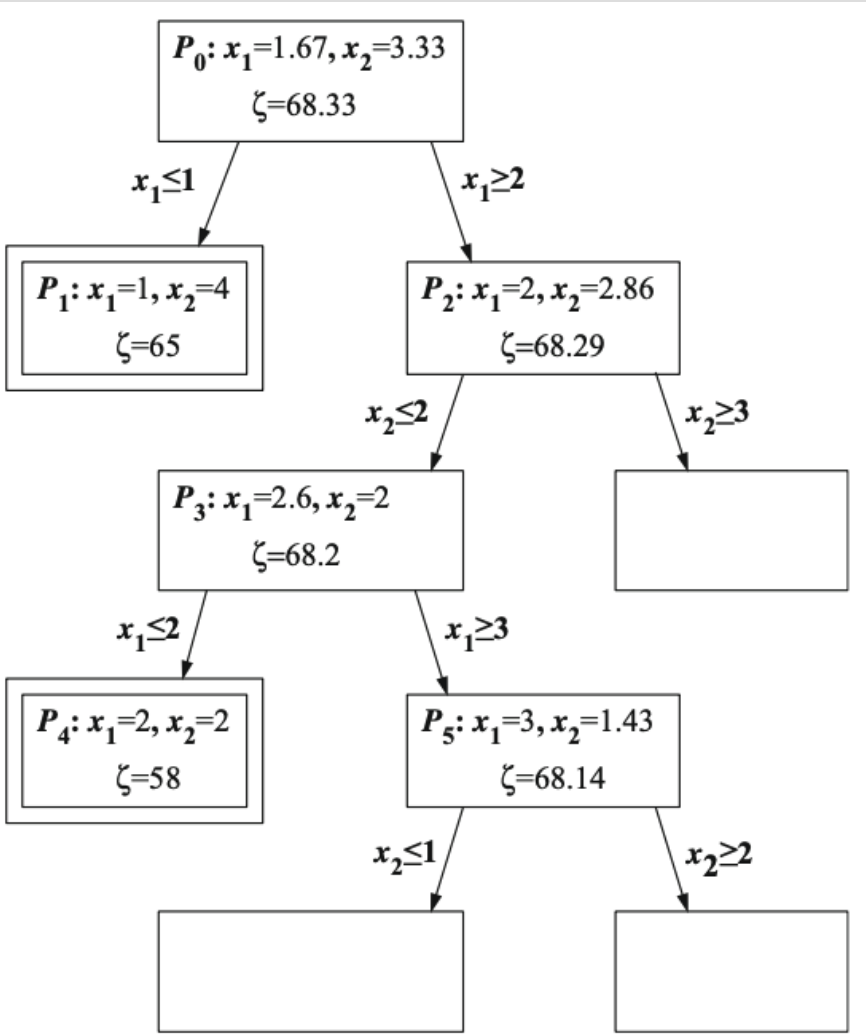
Wir bauen uns nach und nach einen **Enumerationsbaum** auf.



# Beispiel (5)

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

Wir bauen uns nach und nach einen **Enumerationsbaum** auf.

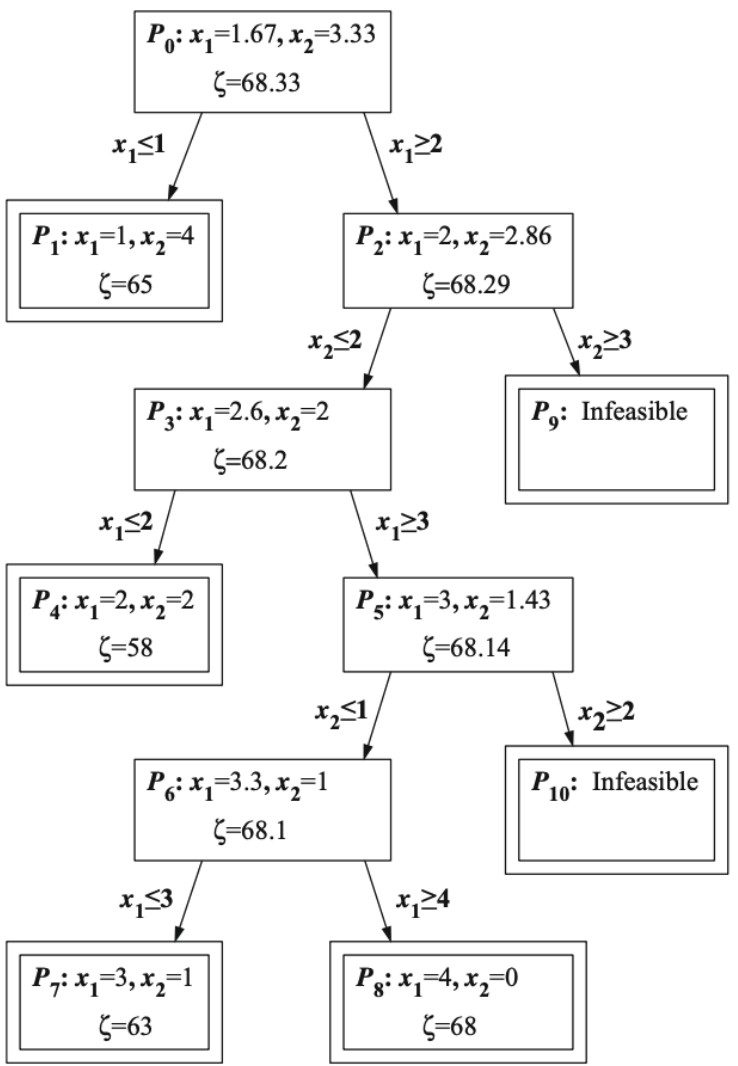


# Beispiel (n)

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

Wir bauen uns nach und nach einen **Enumerationsbaum** auf.

Die beste ganzzahlige Lösung ist also $x = (4,0)$.



# Branch-and-Bound-Algorithmus

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. Wiederhole, solange $Q$ nicht leer:
    1. Nimm $P_i$ aus $Q$
    2. Bestimme optimale Lösung $x_i$ mit Wert $\zeta_i$ der LP-Relaxierung von $P_i$
    3. Falls $P_i$ infeasible, oder $\zeta_i \leq v_B$, starte nächste Iteration.
    4. Falls $x_i$ ganzzahlig ist, setze $v_B = \zeta_i, B = x_i$ und starte nächste Iteration.
    5. Wähle nicht-ganzzahligen Wert $\hat{x}$ mit Wert $a$.
    6. Füge Probleme $P_{i+1} \coloneqq P_i \cup \{\hat{x} \leq \lfloor a \rfloor\}$ und $P_{i+2} \coloneqq P_i \cup \{\hat{x} \geq \lceil a \rceil\}$ zu $Q$ hinzu.
4. Gib $(B, v_B)$ zurück.

Wichtig: Ist am Ende $B = \perp$, dann ist das IP nicht lösbar.

# Fragen


Damit haben wir eine generelle Technik, um IPs zu lösen.
So ein Baum kann sehr groß werden!
Wie können wir Teilbäume frühzeitig abschneiden?

Ab hier nur noch Exkurs!

# Idee

Wenn wir eine Lösung der LP Relaxierung besitzen...

Können wir darin erkennen, ob Constraints existieren, die...

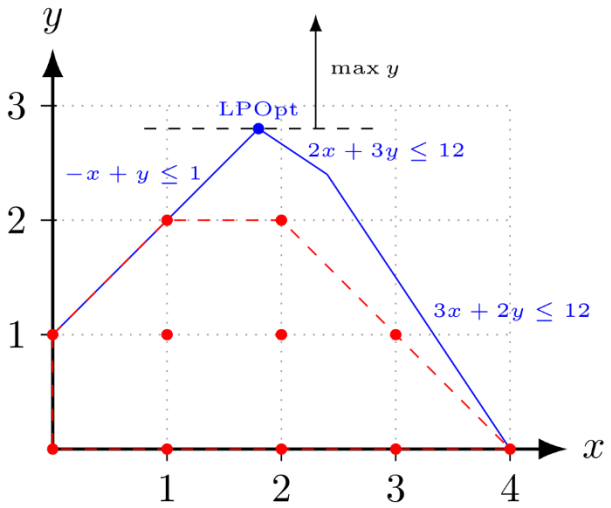
Durch alle ganzzahligen Lösungen erfüllt werden, aber...

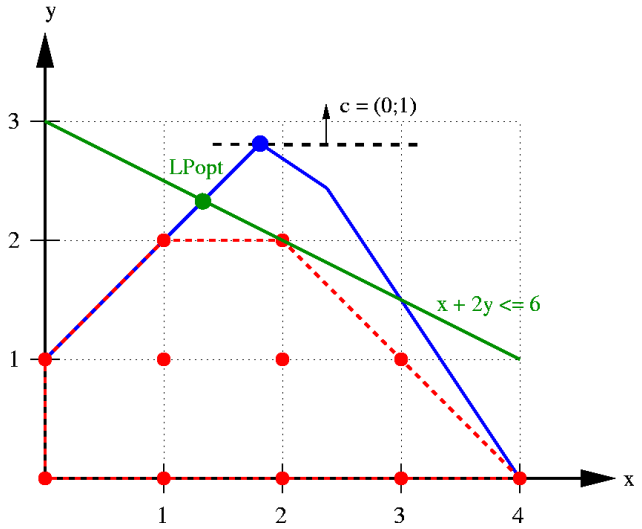
Nicht durch die Lösung der aktuellen LP Relaxierung?

# Cutting Planes

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

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



# Runden und kürzen

Gibt es fraktionale Bounds zu Variablen, z.B. $x \leq \frac{5}{3}$, dann kann dieser Wert abgerundet werden.

$$\Rightarrow x \leq 1$$

Schwieriger wird es, wenn mehrere Variablen in einem Constraint auftauchen:

$$x + 2y + 4z = 4$$

Sind $x, y$ nur binär Variablen, muss $z \geq \frac{4 - \sup(2y+x)}{4} = \frac{1}{4}$, also $z \geq 1$ gelten.

Besitzt ein Constraint nur ganzzahlige Variablen und ganzzahlige Koeffizienten, können wir diesen Constraint vereinfachen. Sei dazu $g$ der ggT der Koeffizienten.
Dann ist ein neuer (oder nur vereinfachter) Constraint:

$$\sum \frac{a_j}{g} x_j \leq \left\lfloor \frac{b}{g} \right\rfloor$$

Die linke Seite besitzt immer noch nur ganzzahlige Koeffizienten, also muss die rechte Seite auch ganzzahlig sein!

# Clique Cuts

Zwei binäre Variablen heißen **inkompatibel**, wenn sie nicht gleichzeitig den Wert 1 annehmen können.

Eine **Clique** ist eine Menge paarweise inkompatibler Binärvariablen $B$:

$$C \subseteq B: \forall x_i, x_j \in C: x_i, x_j \text{ sind inkompatibel}$$

Ein **Clique-Cut** ist dann der Constraint $\sum_{j \in C} x_j \leq 1$

Beispiel:

$$x + y \leq 1, x + z \leq 1, y + z \leq 1 \Rightarrow x + y + z \leq 1$$

Die Constraints müssen nicht immer so einfach zu erkennen sein, sondern man muss *propagieren* (betrachte, was passiert, wenn eine Variable auf 0 bzw. 1 gesetzt wird). Etwas verallgemeinert kann die Kombination von 0-1-Belegungen betrachtet werden!

# Zero-Half-Cuts

Sei $x_1, \dots, x_5 \in \mathbb{Z}$ und betrachte folgende Constraints.

$$x_1 + x_2 + x_3 + 3x_4 + 2x_5 \leq 10$$
$$x_1 + x_2 + 3x_3 + x_4 + 2x_5 \leq 5$$

Addieren wir beide Constraints, erhalten wir

$$2x_1 + 2x_2 + 4x_3 + 4x_4 + 4x_5 \leq 15$$

Linke Seite ist gerade, rechte Seite ist ungerade. Wir den folgenden Constraint hinzufügen:

$$x_1 + x_2 + 2x_3 + 2x_4 + 2x_5 \leq 7$$

Generell: Versuche Constraints so zu addieren, dass die Koeffizienten gerade sind, die rechte Seite aber ungerade.

# Cover Cuts

Betrachte einen Constraint der Form

$$a^T x \leq z$$

Wenn $x \in \{0,1\}^n$, ist es einfach kleine Mengen $C$ zu finden, die folgende Eigenschaft hat.

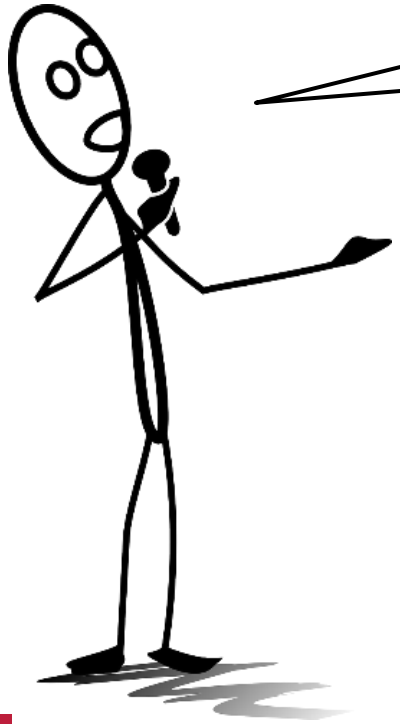
$$\sum_{i \in C} a_i x_i > z$$

Ein Cover Cut bzgl. $C$ ist dann ein Constraint der Form

$$\sum_{i \in C} x_i \leq |C| - 1$$

Gerade beim Knapsack Problem tauchen diese Cuts häufig auf („welche Auswahl ist sowieso zu schwer“), daher werden sie oft auch Knapsack Cut genannt.

# Presolve



Viele dieser Cuts können bereits vor dem eigentlich Lösungsprozess eingebunden werden

Dieser Schritt nennt sich ***presolve***.

Ein Presolve dient dazu:

- Redundante Informationen zu entfernen
- Stärkere Constraints einzufügen

# Nächstes Kapitel

Bisher nur langsame Algorithmen im Worst-Case

Was passiert, wenn ich unbedingt effiziente Algorithmen haben möchte?