Technische Universität Braunschweig

Mathematische Methoden der Algorithmik – Vorlesung #08

Arne Schmidt

Ein neues Kapitel

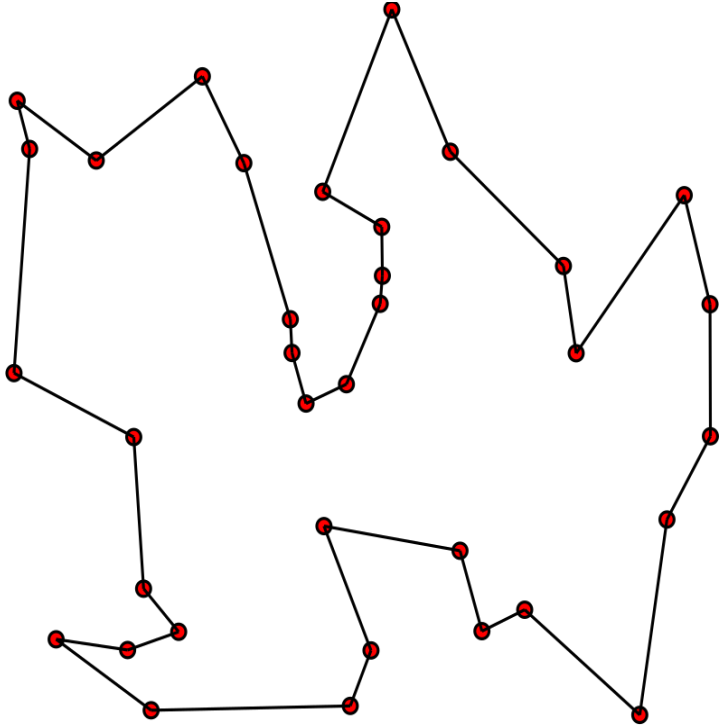
Ein Reisender

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?



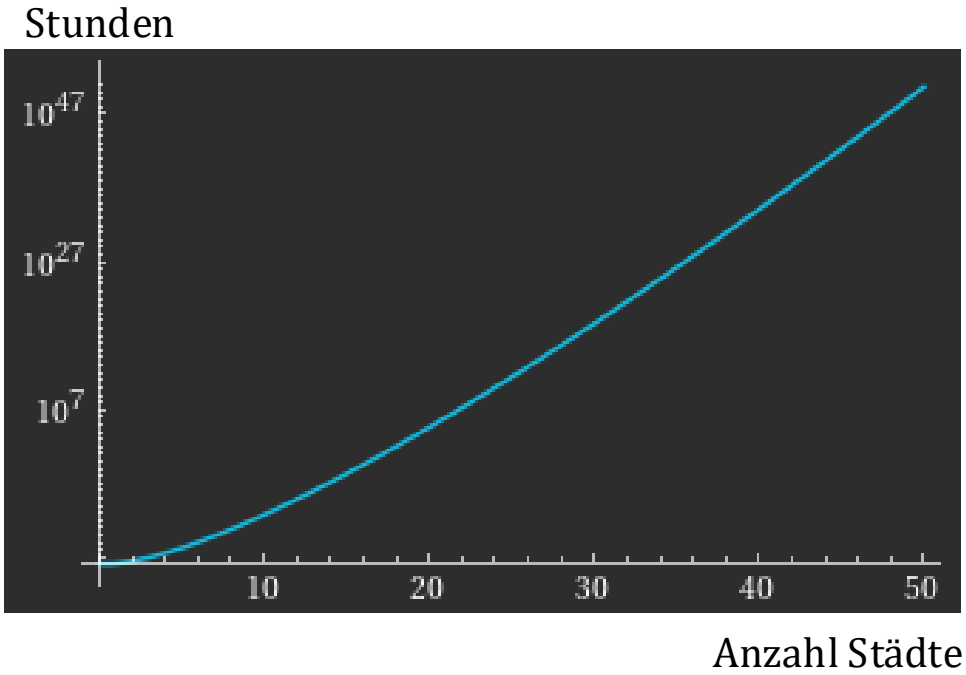
Brute Force

Es gibt $(n-1)!$ viele Touren durch n Städte.

Angenommen, wir können 18 Milliarden Touren in der Sekunde durchprüfen.

Der log-Plot zeigt dann die benötigte Zeit in Stunden an, um die beste Lösung zu finden.

Brute Force scheidet definitiv aus!



Moon helium deal is biggest purchase of natural resources from space

The feasibility of moon mining is not yet proven, but the future of supercomputing may depend on the ability to extract Helium-3 from the lunar surface.

September 16, 2025

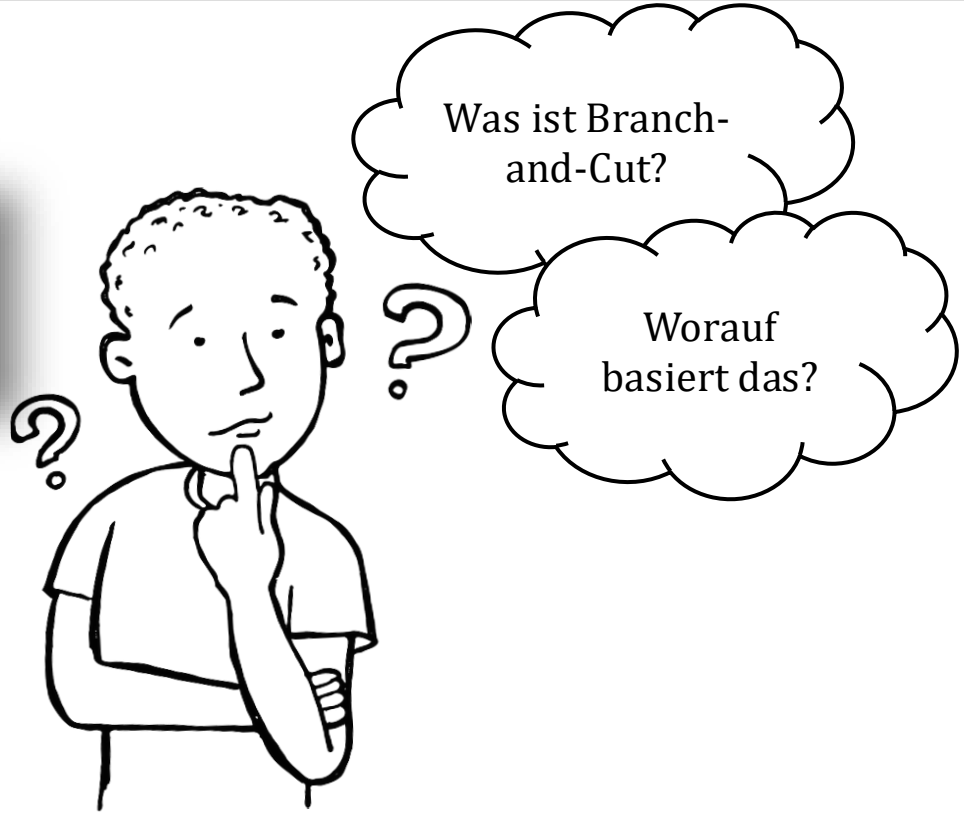
https://www.washingtonpost.com/technology/2025/09/16/moon-mining-helium-quantum-computing/

To understand the computer's potency, take this common thought experiment: A salesperson must travel to 22 cities, visiting each city only once and returning to the first city in the most efficient route. It sounds simple, but a laptop computer would take about 1,000 years to figure out the best course. A quantum computer could one day compute the solution in minutes, or maybe even seconds.

Wie bekommt man das schneller?

1,002 and 2.392 cities, respectively. The CPU times on the NBS CYBER 205 for the 1,002-city problem were 7 hours and 18 minutes and 27 hours and 20 minutes for the 2,392-city problem, respectively.

Aus: Optimization of a 532-City Symmetric Traveling Salesman Problem by Branch and Cut.
(1987, Padberg und Rinaldi)

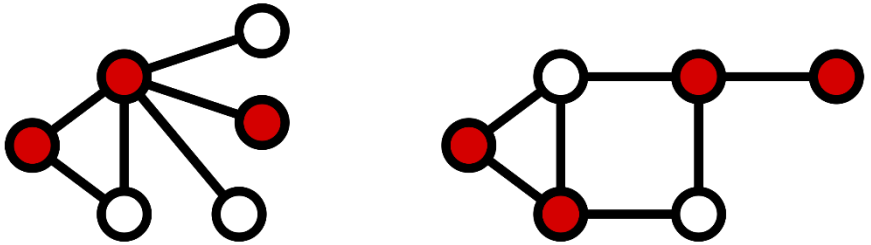


Integer Programming

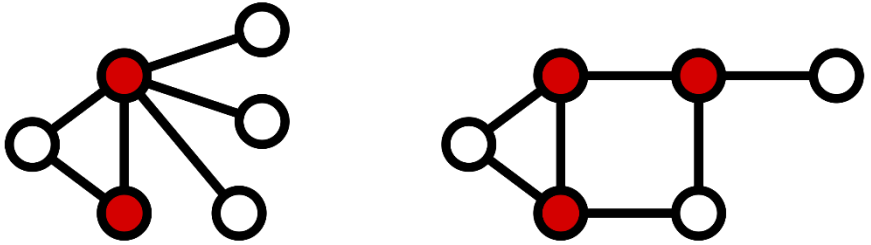
Ein anderes Problem: Minimum Vertex Cover

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.

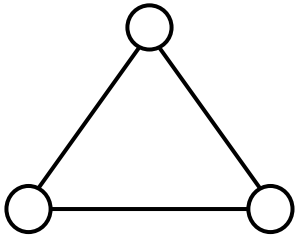
s.t.

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

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!

Komplexität von Vertex Cover und LPs

Simplex benötigt zwar exponentielle Laufzeit im Worst-Case, es gibt aber andere Algorithmen, die LPs in effizienter Zeit (polynomiell im Input) lösen können.

$\Rightarrow$ LP $\in P$

Vertex Cover auf der anderen Seite ist NP-schwer. D.h. es existiert vermutlich kein Algorithmus, der das Vertex Cover Problem in polynomieller Zeit lösen kann.

$\Rightarrow$ LPs können Vertex Cover nicht lösen. (Es sei denn $P = NP$ gilt.)

Können wir LPs etwas erweitern, sodass wir NP-schwere Probleme modellieren können?

Minimum Vertex Cover als LP

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

Minimiere Anzahl der ausgewählten Knoten.

s.t.

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

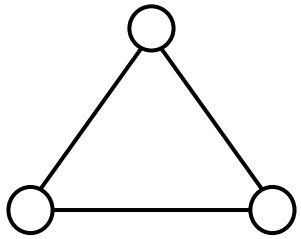
Jede Kante muss überdeckt sein.

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

Jede Kante 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!

3-Satisfiability

DAS Problem für NP-schwere ist 3-Satisfiability (3-SAT).

Gegeben: Eine boolesche Formel $\varphi(x) = \bigwedge_{i=1}^{m}(\ell_{1,i} \vee \ell_{2,i} \vee \ell_{3,i})$, wobei $\ell_{k,i}$ ein Literal in der Form x_j oder $\bar{x}_j$ einer der n Variablen ist.
Gesucht: Eine Belegung von x, sodass $\varphi(x)$ mit wahr evaluiert wird.

Um dieses Problem über ein LP zu modellieren, stellen wir folgende Fragen.

- Worüber treffen wir Entscheidungen? → Variablen x_j
- Wann ist eine Klausel $(\ell_{1,i} \vee \ell_{2,i} \vee \ell_{3,i})$ erfüllt? → z.B. $(x_1 \vee \bar{x}_2 \vee x_3)$ wird zu $x_1 + (1 - x_2) + x_3 \geq 1$
- Was ist die Zielfunktion? → min/max 0

Implikationen

Das Lösen von LPs mit ganzzahligen Variablen muss NP-schwer sein

Sogar das Entscheiden, ob es überhaupt eine Lösung gibt, muss NP-schwer sein!

Was können wir nun tun, um Probleme trotzdem damit zu lösen?

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 $\bar{I}$.

Branch-and-Bound

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.

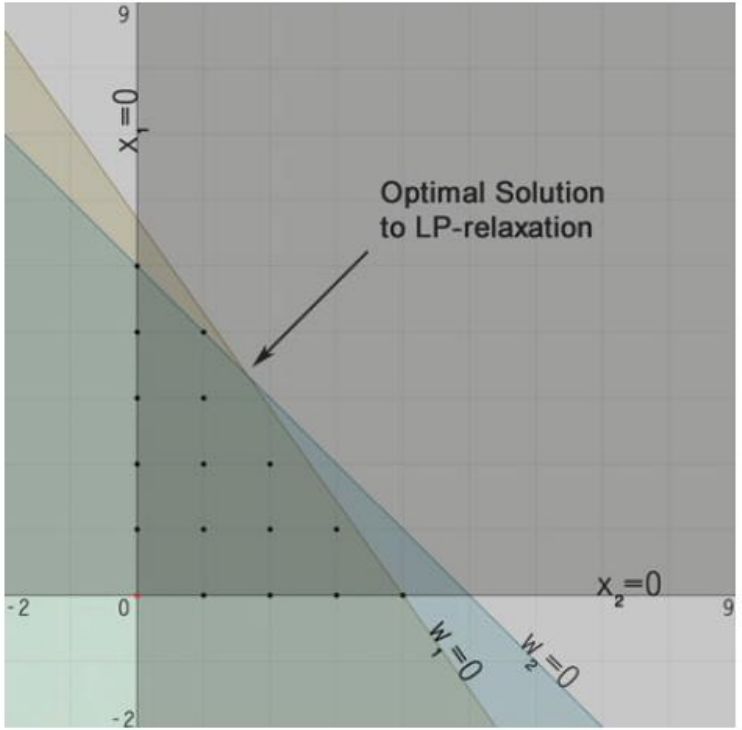
Beispiel (1)

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

Optimale Lösung:

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

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



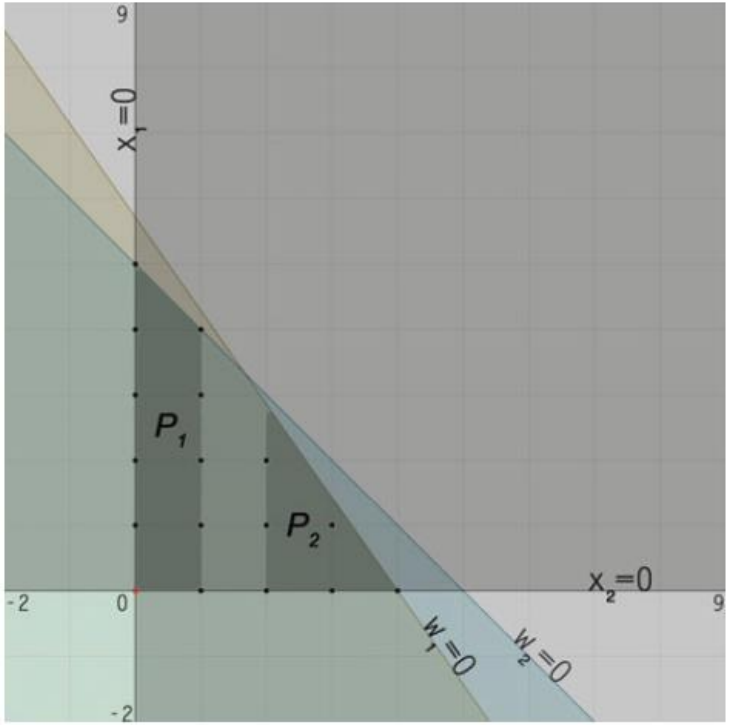
Beispiel (2)

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

Optimale Lösung:

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

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

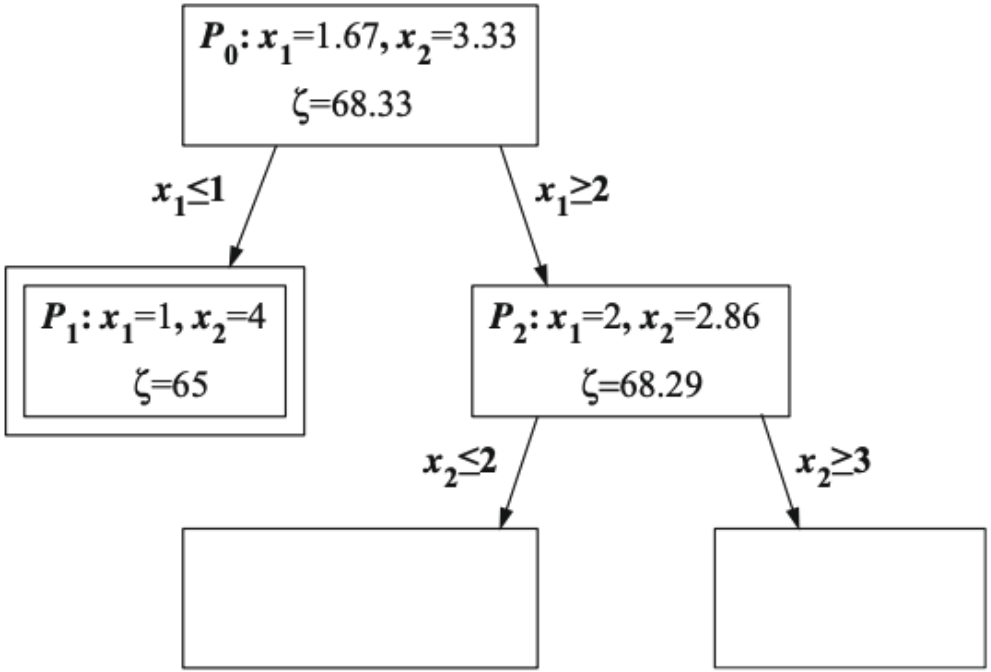


Beispiel (3)

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

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

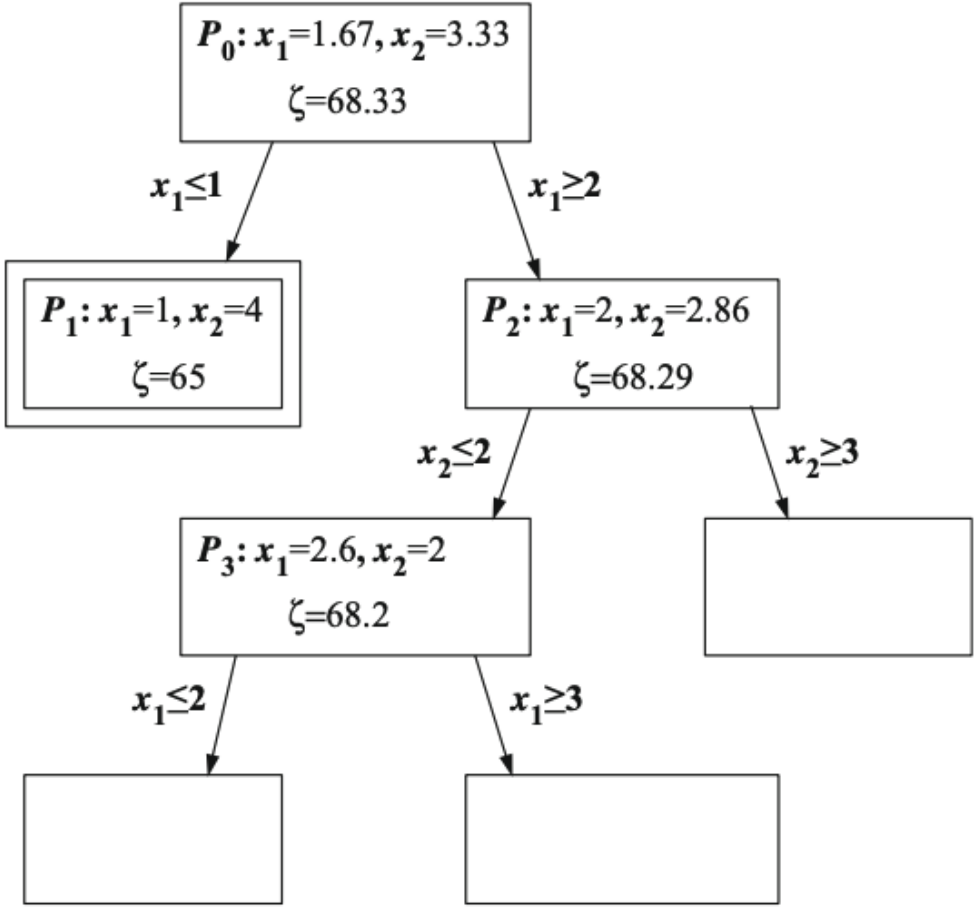
P1 besitzt ganzzahlige, optimale Lösung; P2 nicht.



Beispiel (4)

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

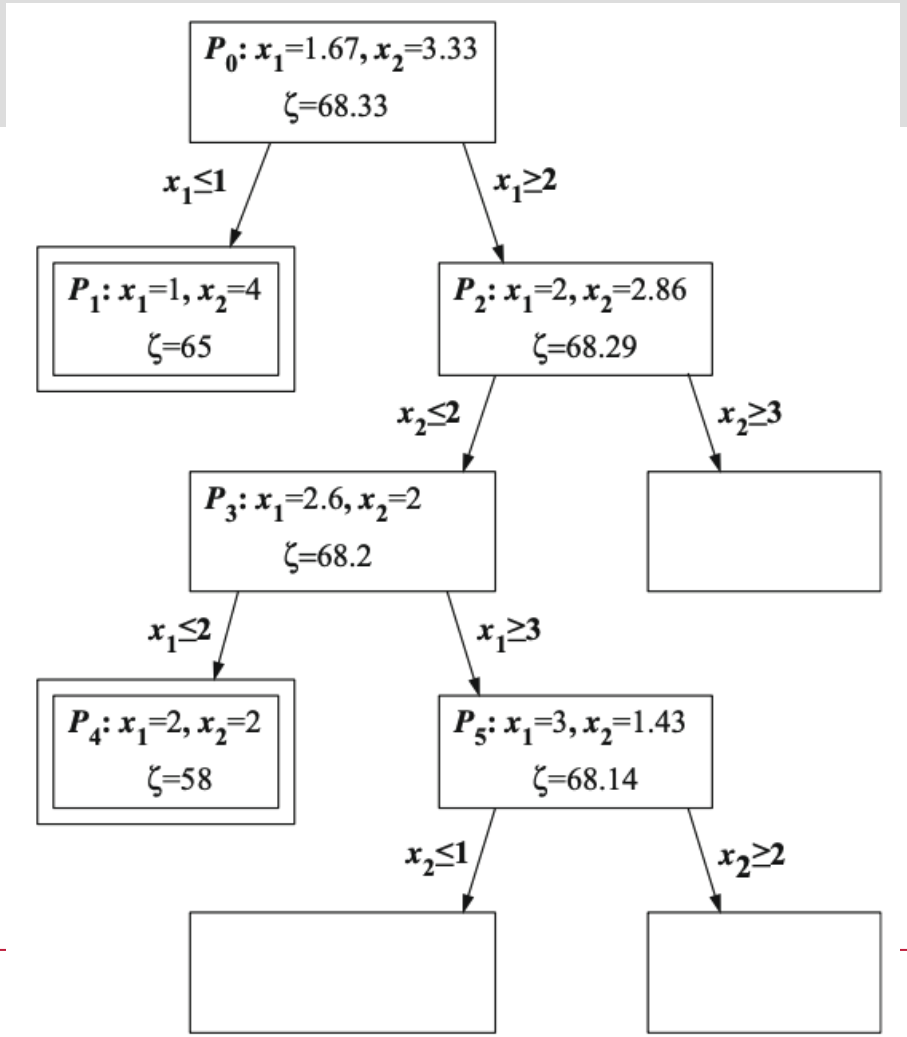
Wir bauen uns nach und nach einen **Enumerationsbaum** auf.



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.

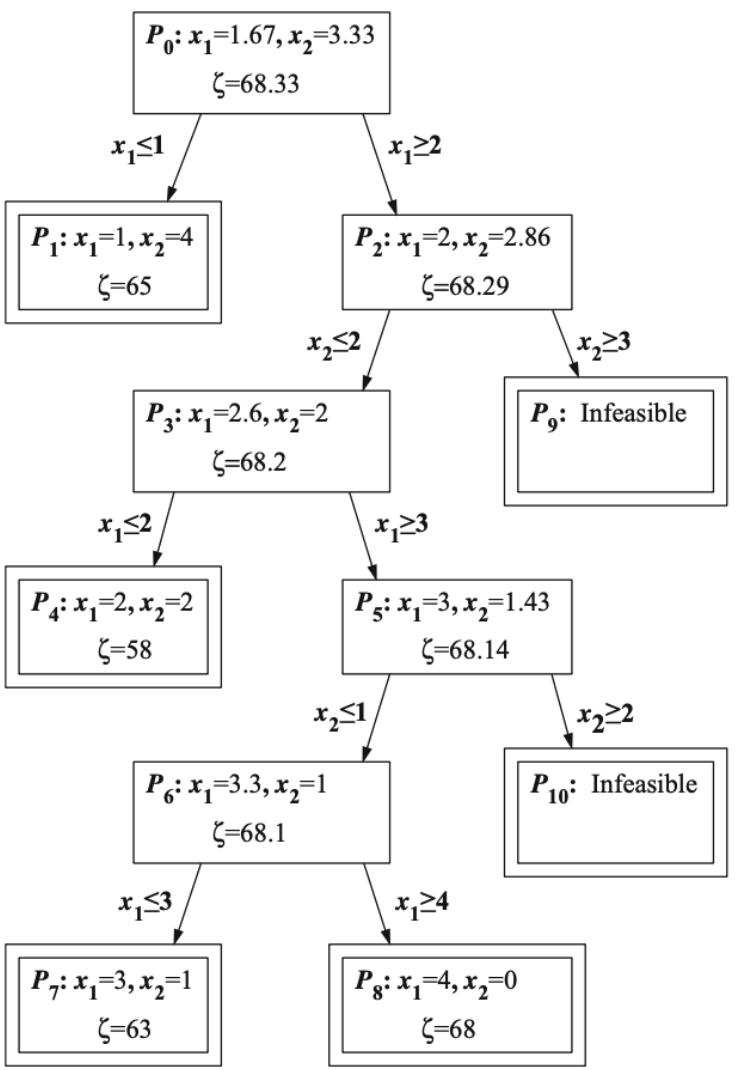


Beispiel (n)

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

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

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

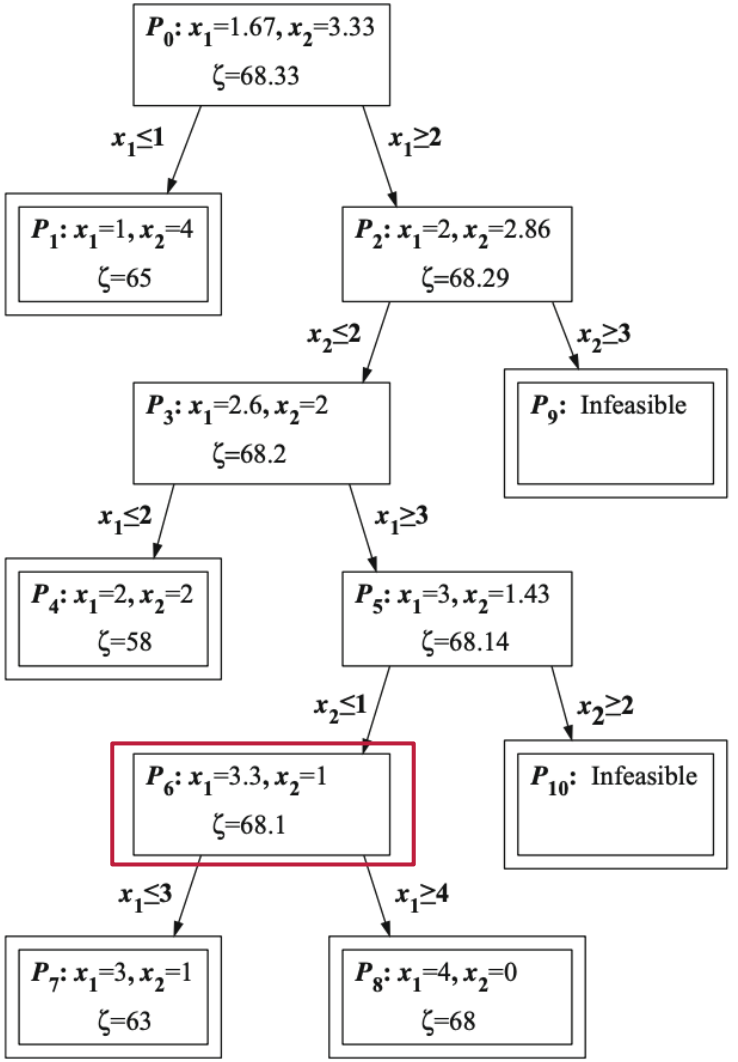


Was wäre wenn

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

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

Was wäre, wenn umrandete Lösung einen relaxierten Lösungswert von bspw. 57 hätte?
→ Wir hätten nicht weiter such müssen, wir kennen bereits bessere Lösungen!



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 Teilbäume so früh wie möglich ab („prune“).
  - Benutze Lösung des Vaterknotens wieder („Warmstart“).

Warmstart

Das optimale Dictionary im Wurzelknoten (P_0):

$$\begin{aligned} \zeta &= \frac{205}{3} - \frac{5}{3}w_1 - \frac{1}{3}w_2 \\ \hline x_1 &= \frac{5}{3} - \frac{1}{3}w_1 + \frac{7}{3}w_2 \\ x_2 &= \frac{10}{3} + \frac{1}{3}w_1 - \frac{10}{3}w_2 \end{aligned}$$

Fügen wir die Bedingung $x_1 \geq 2$ hinzu, erhalten wir die Slackvariable

$$\begin{aligned} g_1 &= x_1 - 2 \\ &= -\frac{1}{3} - \frac{w_1}{3} + \frac{7w_2}{3} \end{aligned}$$

$$\begin{aligned} \zeta &= \frac{205}{3} - \frac{5}{3}w_1 - \frac{1}{3}w_2 \\ \hline x_1 &= \frac{5}{3} - \frac{1}{3}w_1 + \frac{7}{3}w_2 \\ x_2 &= \frac{10}{3} + \frac{1}{3}w_1 - \frac{10}{3}w_2 \\ g_1 &= -\frac{1}{3} - \frac{1}{3}w_1 + \frac{7}{3}w_2 \end{aligned}$$

Führe einen dualen Pivotschritt aus!

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 ζ_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 infeasible.

Fragen


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

Branch-and-Cut

Idee

Wenn wir eine Lösung der LP Relaxierung besitzen...

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

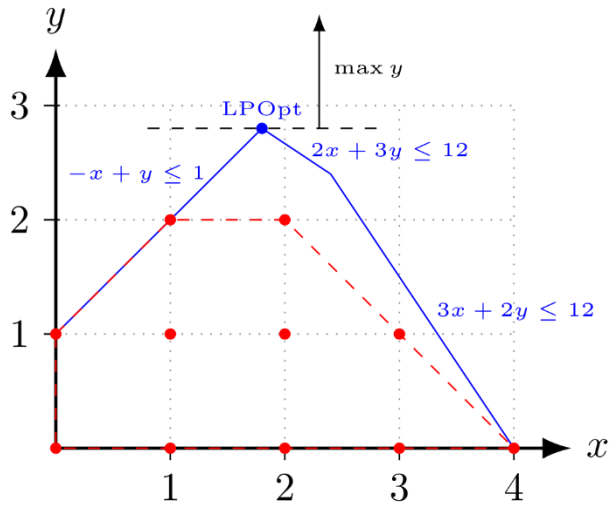
Durch alle ganzzahligen Lösungen erfüllt werden, aber...

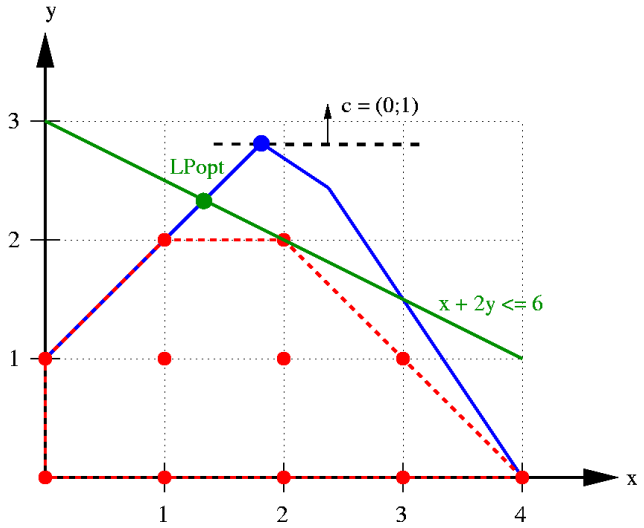
Nicht durch die Lösung der aktuellen LP Relaxierung?

Cutting Planes

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

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





Gomory-Cuts (1)

Betrachte eine Zeile des optimalen Dictionaries.

$$x_i = x_i^* - \sum_{j \in \mathcal{N}} \overline{a}_{ij} x_j \Leftrightarrow x_i^* = x_i + \sum_{j \in \mathcal{N}} \overline{a}_{ij} x_j$$

Wenn wir die Koeffizienten in einen ganzzahligen Teil und einen fractionalen Teil aufteilen, erhalten wir folgende Zeile.

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

Separieren von Ganzzahlen und fractionalen Werten liefert:

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

Gomory-Cuts (2)

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

< 1

≥ 0 für $x \geq 0$

$\in \mathbb{Z}$

< 1

Also können wir festhalten:

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

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

Damit haben wir einen neuen Constraint!

Gomory-Cut Beispiel – Ein neuer Contraint

Wie lautet ein möglicher Gomory-Cut?
x_1 ist nicht ganzzahlig. Betrachte also

$$x_1 + \frac{5}{54}w_1 + \frac{1}{54}w_2 = \frac{11}{3}$$

Füge also den Constraint

$$x_1 + 0w_1 + 0w_2 \leq 3 \Leftrightarrow x_1 \leq 3$$

hinzu.

$$\begin{aligned} \zeta &= \frac{179}{3} - \frac{7}{27}w_1 - \frac{73}{54}w_2 \\ \hline x_1 &= \frac{11}{3} - \frac{5}{54}w_1 - \frac{1}{54}w_2 \\ x_2 &= \frac{7}{3} + \frac{1}{27}w_1 + \frac{5}{54}w_2 \\ w_3 &= 13 - \frac{5}{9}w_1 - \frac{8}{9}w_2 \end{aligned}$$

Gomory-Cut Beispiel – Eine neue Zeile

Das Hinzufügen von $x_1 \leq 3$ liefert eine (ganzzahlige) Basis-Schlupfvariable

$$w_4 = 3 - x_1 = 3 - \frac{11}{3} + \frac{5}{54}w_1 + \frac{1}{54}w_2$$
$$= -\frac{2}{3} + \frac{5}{54}w_1 + \frac{1}{54}w_2$$

Führe eine Iteration des dualen Simplex aus.

$$\begin{aligned}
\zeta &= \frac{179}{3} - \frac{7}{27}w_1 - \frac{73}{54}w_2 \\
\hline
x_1 &= \frac{11}{3} - \frac{5}{54}w_1 - \frac{1}{54}w_2 \\
x_2 &= \frac{7}{3} + \frac{1}{27}w_1 + \frac{5}{54}w_2 \\
w_3 &= 13 - \frac{5}{9}w_1 - \frac{8}{9}w_2
\end{aligned}$$

Gomory-Cut Beispiel – Eine neue Iteration

Wie lautet ein möglicher Gomory-Cut?
x_2 ist nicht ganzzahlig. Betrachte also

$$x_2 - \frac{2}{5}w_4 - \frac{23}{270}w_2 = \frac{13}{5}$$

Also

$$x_2 - 1w_4 - 1w_2 \leq 2$$

Und damit die neue Zeile

$$w_5 = -\frac{3}{5} + \frac{3}{5}w_4 + \frac{247}{270}w_2$$

$$\begin{array}{rrrr} \zeta = & \frac{289}{5} - & \frac{14}{5}w_4 - & \frac{13}{10}w_2 \\ \hline x_1 = & 3 - & w_4 & \\ x_2 = & \frac{13}{5} + & \frac{2}{5}w_4 + & \frac{23}{270}w_2 \\ w_3 = & 9 - & 6w_4 - & \frac{7}{9}w_2 \\ w_1 = & \frac{36}{5} + & \frac{54}{5}w_4 - & \frac{1}{5}w_2 \end{array}$$

Gomory-Cut Beispiel – Eine neue Iteration

Wie lautet ein möglicher Gomory-Cut?
x_2 ist nicht ganzzahlig. Betrachte also

$$x_2 - \frac{2}{5}w_4 - \frac{23}{270}w_2 = \frac{13}{5}$$

Also

$$x_2 - 1w_4 - 1w_2 \leq 2$$

Und damit die neue Zeile

$$w_5 = -\frac{3}{5} + \frac{3}{5}w_4 + \frac{247}{270}w_2$$

Führe wieder einen dualen Pivotschritt aus!

$$\begin{array}{rrrr}
\zeta = & \frac{289}{5} - & \frac{14}{5}w_4 - & \frac{13}{10}w_2 \\
\hline
x_1 = & 3 - & w_4 & \\
x_2 = & \frac{13}{5} + & \frac{2}{5}w_4 + & \frac{23}{270}w_2 \\
w_3 = & 9 - & 6w_4 - & \frac{7}{9}w_2 \\
w_1 = & \frac{36}{5} + & \frac{54}{5}w_4 - & \frac{1}{5}w_2 \\
w_5 = & -\frac{3}{5} + & \frac{3}{5}w_4 + & \frac{247}{270}w_2
\end{array}$$

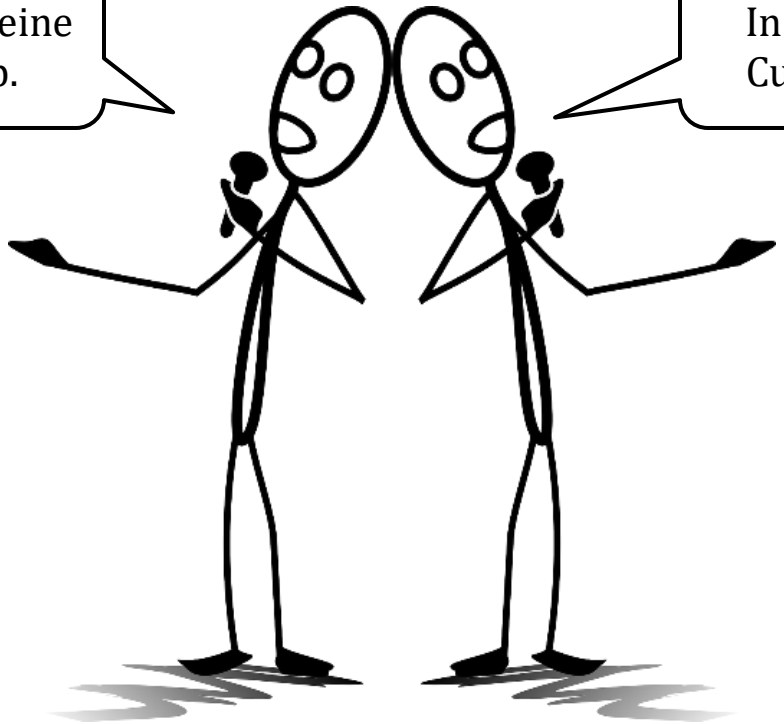
Gomory-Cuts: Vor- und Nachteile

Gomory-Cuts schneiden keine integralen Lösungen ab.

Mit jedem Cut erhalten wir eine andere Lösung

Erhalten wir ein infeasible LP, gibt es keine integrale Lösung!

Irgendwann werden wir tatsächlich terminieren!



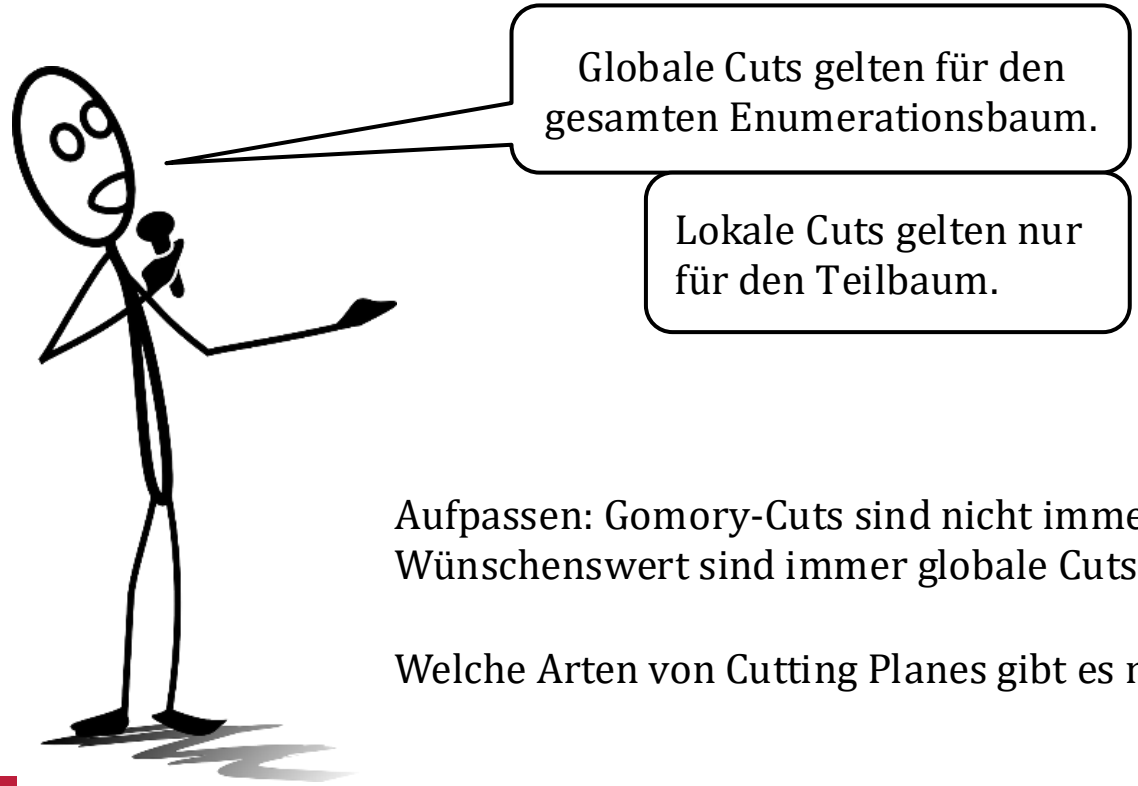
In der Praxis nur Gomory-Cuts nutzen ist ineffizient!

Numerische Probleme können auftauchen!

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.

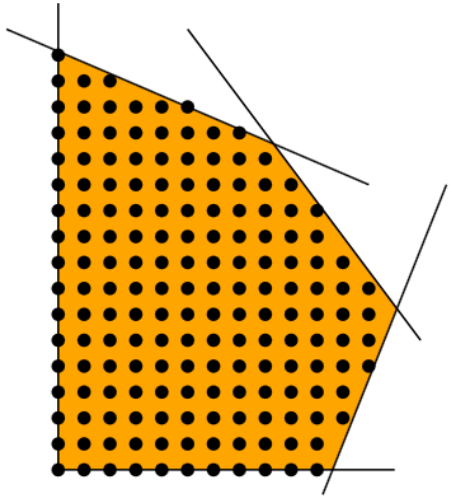
Globale vs. Lokale Cuts



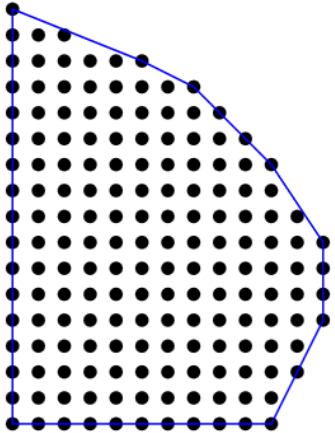
Aufpassen: Gomory-Cuts sind nicht immer globale Cuts!
Wünschenswert sind immer globale Cuts.

Welche Arten von Cutting Planes gibt es noch?

Konvexe Hülle



LP Lösungsraum



Konvexe Hülle der Integer-Lösungen

Idee: Finde Cuts, die die konvexe Hülle der integralen Lösungen beschreiben.
Die stärksten Cuts, die wir hinzufügen können, definieren die *Facetten der konvexen Hülle*.
Können wir diese immer finden?

Das Separationsproblem

Gegeben sei ein Polyeder $Q \subseteq \mathbb{R}^n$ und ein Vektor $\tilde{x} \in \mathbb{R}^n$. Entscheide, ob $\tilde{x} \in Q$ und falls nicht, gib einen linearen Constraint (a,b) an, sodass $a^T \tilde{x} > b$ und $a^T x \leq b$ für alle $x \in Q$.

Dabei ist in der Regel Q die konvexe Hülle der ganzzahligen Lösungen und $\tilde{x}$ die optimale Lösung der LP Relaxierung.

Die Ellipsoid Methode (löst LPs in polynomieller Zeit!), kann über Q ohne explizite Beschreibung von Q optimieren. Dabei wird nur ein Separationsorakel benötigt.

Diese Methode hat polynomielle Laufzeit, wenn das Separationsorakel eine polynomielle Laufzeit besitzt.

Korollar: Wenn wir das Separationsproblem in polynomieller Zeit lösen können, können lineare Optimierungsprobleme über Q in polynomieller Zeit gelöst werden.

Cutting Plane - Techniken

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!

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.

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!

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.

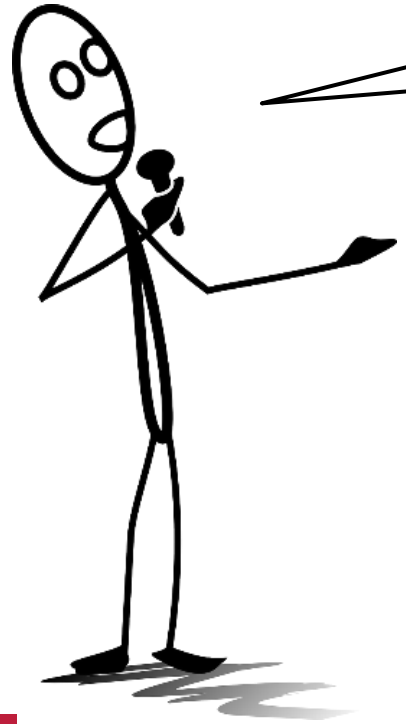
$$\sum_{i \in C} a_i x_i > z$$

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


Wir wissen, was wir allgemein für IPs ausführen können.
Können wir für spezielle Probleme noch andere Techniken ausschöpfen?