Ganzzahlige Programmierung

Branch & Bound – Grundlagen und Algorithmus

Deutsch

Wolfgang Böhme

wolfgang.boehme@economic-modeling.de

www.economic-modeling.de

25. Oktober 2026

Dieses Dokument führt systematisch in die ganzzahlige Programmierung ein. Voraussetzung: Grundkenntnisse der Simplex-Methode.

Dokumentationen dieser Webseite wurden mit Unterstützung der KI Claude (Anthropic) entwickelt.

Inhaltsverzeichnis

1 Einführung und Motivation **2**

2 Grundbegriffe **2**
2.1 Typen ganzzahliger Programme 2
2.2 LP-Relaxierung 3
2.3 Warum Runden nicht funktioniert 3

3 Der Branch & Bound Algorithmus **3**
3.1 Grundprinzip 3
3.2 Vollständiger Algorithmus 4
3.3 Strategien für Knotenauswahl und Variablenwahl 4

4 Vollständiges Beispiel **5**
4.1 Problemstellung 5
4.2 Knoten 0: LP-Relaxierung 5
4.3 Knoten 1: $x_1 \leq 1$ 6
4.4 Knoten 2: $x_1 \geq 2$ 6
4.5 Knoten 3: $x_1 \geq 2$, $x_2 \leq 2$ 6
4.6 Knoten 4: $x_1 \geq 2$, $x_2 \geq 3$ 6
4.7 Knoten 5: $x_1 \leq 2$, $x_2 \leq 2$ (und $x_1 \geq 2$) 6
4.8 Knoten 6: $x_1 \geq 3$, $x_2 \leq 2$ 6
4.9 Knoten 7: $x_1 \geq 3$, $x_2 = 0$ 7
4.10 Knoten 8: $x_1 \geq 3$, $x_2 \geq 1$, $x_2 \leq 2$ 7
4.11 Zusammenfassung: Branch & Bound Baum 7
4.12 Optimale Lösung 7

5 Korrektheit und Konvergenz **7**

6 Spezielle Strukturen: Binäre Programme **8**
6.1 0-1-Rucksackproblem 8
6.2 Cutting Planes 8

7 Branch & Bound + Cutting Planes: Branch & Cut **9**

8 Komplexität und praktische Hinweise **9**
8.1 NP-Schwere 9
8.2 Praktische Software 9

9 Zusammenfassung **10**

1 Einführung und Motivation

In vielen realen Optimierungsproblemen können die Entscheidungsvariablen keine beliebigen reellen Werte annehmen, sondern müssen **ganzzahlig** sein. Typische Beispiele:

Typische Anwendungen ganzzahliger Programmierung

- **Standortplanung:** Soll eine Fabrik an einem bestimmten Standort gebaut werden oder nicht? ($x_i \in \{0, 1\}$)
- **Rucksackproblem:** Wie viele Einheiten eines Gutes sollen eingepackt werden? ($x_i \in \mathbb{Z}_{\geq 0}$)
- **Personalplanung:** Wie viele Mitarbeiter werden in Schicht i eingesetzt? ($x_i \in \mathbb{Z}_{\geq 0}$)
- **Produktionsplanung:** Wie viele ganze Maschinen werden angeschafft? ($x_i \in \mathbb{Z}_{\geq 0}$)
- **Tourenplanung:** Wird Route j befahren oder nicht? ($x_j \in \{0, 1\}$)

Das einfache Runden der LP-Lösung führt in der Regel **nicht** zur optimalen ganzzahligen Lösung und kann sogar zu unzulässigen Lösungen führen. Es wird daher ein spezieller Algorithmus benötigt: **Branch & Bound**.

2 Grundbegriffe

2.1 Typen ganzzahliger Programme

Definition: Ganzzahlige Programme

- **Ganzzahliges lineares Programm (ILP):** Alle Variablen ganzzahlig.

$$\max \mathbf{c}^\top \mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{x} \in \mathbb{Z}^n$$

- **Gemischt-ganzzahliges lineares Programm (MILP):** Nur ein Teil der Variablen ganzzahlig.

$$\max \mathbf{c}^\top \mathbf{x} + \mathbf{d}^\top \mathbf{y} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} \leq \mathbf{b}, \quad \mathbf{x} \in \mathbb{Z}^p, \ \mathbf{y} \in \mathbb{R}^q$$

- **Binäres ganzzahliges Programm (BIP):** Alle Variablen binär ($x_i \in \{0, 1\}$).

2.2 LP-Relaxierung

Definition: LP-Relaxierung

Die **LP-Relaxierung** eines ILP entsteht durch Weglassen der Ganzzahligkeitsbedingungen:

$$\text{(LP-R)} \quad \max \mathbf{c}^\top \mathbf{x} \quad \text{s.t.} \quad \mathbf{Ax} \leq \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}$$

Satz 1: Schranke durch LP-Relaxierung

Für jedes ILP gilt: Der Optimalwert der LP-Relaxierung z^*_{LP} ist eine **obere Schranke** für den Optimalwert des ILP z^*_{ILP}:

$$z^*_{ILP} \leq z^*_{LP}$$

Proof. Jede ganzzahlig zulässige Lösung ist auch LP-zulässig (da die Ganzzahligkeitsbedingung eine stärkere Forderung ist). Das LP optimiert über eine größere zulässige Menge, daher ist sein Optimum mindestens so groß. □

Integrality Gap: Die Differenz $z^*_{LP} - z^*_{ILP}$ heißt **Integrality Gap**. Je größer diese Lücke, desto schwieriger ist das Problem typischerweise.

2.3 Warum Runden nicht funktioniert

Gegenbeispiel: Runden führt zu falschen Ergebnissen

$$\max \; z = 5x_1 + 4x_2 \quad \text{s.t.} \quad 6x_1 + 4x_2 \leq 21, \quad x_1 + 2x_2 \leq 7, \quad x_1, x_2 \geq 0, \; x_1, x_2 \in \mathbb{Z}$$

LP-Relaxierung: $x_1^* = 2{,}5$, $x_2^* = 2{,}25$, $z^*_{LP} = 21{,}5$
Gerundet: $\lfloor x_1 \rfloor = 2$, $\lfloor x_2 \rfloor = 2 \Rightarrow z = 18$ (zulässig, aber nicht optimal!)
Wahres Optimum: $x_1^* = 3$, $x_2^* = 0$, $z^*_{ILP} = 15$ *oder besser:* $x_1^* = 1$, $x_2^* = 3$, $z = 17$
Tatsächlich: $x_1^* = 3$, $x_2^* = 0$ mit $z = 15$ oder $x_1^* = 0$, $x_2^* = 3$ mit $z = 12$ – das gerundete Ergebnis $(2, 2)$ mit $z = 18$ ist zufällig besser, aber es ist nicht das wahre Optimum. Das wahre Optimum liegt bei $x_1 = 2$, $x_2 = 2$, $z = 18$ – in diesem Fall stimmt es zufällig, aber das ist nicht garantiert!

3 Der Branch & Bound Algorithmus

3.1 Grundprinzip

Branch & Bound (B&B) kombiniert drei Ideen:

1. **Relaxierung (Bound):** Löse das LP, um eine obere Schranke für den Teilbaum zu erhalten.
2. **Verzweigung (Branch):** Teile das Problem in zwei Teilprobleme durch Hinzufügen einer Nebenbedingung.

3. **Beschneiden (Cut/Prune):** Verwirf Teilprobleme, deren obere Schranke die bekannte beste ganzzahlige Lösung nicht übersteigt.

Definition: Branch-Variable und Verzweigung

Sei $\mathbf{x}^*$ die LP-optimale Lösung eines Teilproblems und x_j^* eine gebrochene Variable ($x_j^* \notin \mathbb{Z}$). Dann wird **auf** x_j verzweigt:

- **Linker Teilbaum:** füge $x_j \leq \lfloor x_j^* \rfloor$ hinzu
- **Rechter Teilbaum:** füge $x_j \geq \lceil x_j^* \rceil$ hinzu

Jede ganzzahlige Lösung erfüllt genau eine dieser beiden Bedingungen.

3.2 Vollständiger Algorithmus

Algorithmus: Branch & Bound (Maximierung)

1. **Initialisierung:** Setze $z^* = -\infty$ (beste bekannte ganzzahlige Lösung), $\mathbf{x}^* = \emptyset$. Füge das ursprüngliche LP als Knoten 0 zur Warteschlange hinzu.
2. **Knotenauswahl:** Wähle einen Knoten aus der Warteschlange (z.B. Best-First oder Depth-First).
3. **LP lösen:** Löse die LP-Relaxierung des Knotens.
4. **Fallunterscheidung:**
   (a) **LP unzulässig:** Knoten verwerfen. *Beschnitten (unzulässig).*
   (b) **LP-Schranke $\leq z^*$:** Knoten verwerfen. *Beschnitten (Bound).*
   (c) **LP-Lösung ganzzahlig und LP-Schranke $> z^*$:** Aktualisiere z^* und $\mathbf{x}^*$. *Neue beste Lösung.*
   (d) **LP-Lösung gebrochen:** Wähle gebrochene Variable x_j, erzeuge zwei Kindknoten. *Verzweige.*
5. Falls Warteschlange leer: $\mathbf{x}^*$ ist optimal. STOPP. Sonst: Gehe zu Schritt 2.

3.3 Strategien für Knotenauswahl und Variablenwahl

Knotenauswahl:

- **Best-First Search:** Wähle den Knoten mit der höchsten oberen Schranke. Führt schnell zur Optimallösung, benötigt aber viel Speicher.
- **Depth-First Search:** Verfolge den tiefsten Knoten. Speichersparend, findet schnell zulässige Lösungen.
- **Best-First + Backtracking:** Kombination beider Strategien.

Variablenwahl (Branching Rules):

- **Most Fractional:** Wähle die Variable mit der am stärksten gebrochenen Lösung (x_j^* am nächsten an 0,5).
- **Least Fractional:** Wähle die Variable mit der am wenigsten gebrochenen Lösung.
- **Pseudo-Cost Branching:** Schätze die Verbesserung je Verzweigung aus früheren Iterationen.

4 Vollständiges Beispiel

4.1 Problemstellung

Beispiel: Produktionsplanung mit ganzzahligen Mengen

Ein Unternehmen produziert zwei Produkte P_1 und P_2.

$$\max\ z = 5x_1 + 4x_2$$

$$\begin{aligned} 6x_1 + 4x_2 &\leq 21 \qquad \text{(Maschinenkapazität)} \\ x_1 + 2x_2 &\leq 7 \qquad \text{(Lagerkapazität)} \\ x_1, x_2 &\geq 0, \quad x_1, x_2 \in \mathbb{Z} \end{aligned}$$

4.2 Knoten 0: LP-Relaxierung

Wir lösen zunächst die LP-Relaxierung (ohne Ganzzahligkeitsbedingung).

Starttableau (Schlupfvariablen s_1, s_2):

Basis	x_1	x_2	s_1	s_2	b
s_1	6	4	1	0	21
s_2	1	2	0	1	7
z	-5	-4	0	0	0

Iteration 1: Pivotspalte x_1 ($\bar{c}_1 = -5$). Quotienten: $21/6 = 3{,}5$, $7/1 = 7$. Pivotzeile: s_1. Pivotelement: 6.

Tableau nach Iteration 1:

Basis	x_1	x_2	s_1	s_2	b
x_1	1	2/3	1/6	0	7/2
s_2	0	4/3	$-1/6$	1	7/2
z	0	$-2/3$	5/6	0	35/2

Iteration 2: Pivotspalte x_2. Quotienten: $(7/2)/(2/3) = 21/4 = 5{,}25$, $(7/2)/(4/3) = 21/8 = 2{,}625$. Pivotzeile: s_2. Pivotelement: 4/3.

Optimales Tableau (Knoten 0):

Basis	x_1	x_2	s_1	s_2	b
x_1	1	0	1/4	−1/2	7/4
x_2	0	1	−1/8	3/4	21/8
z	0	0	3/4	1/2	**21,5**

LP-Lösung: $x_1^* = 7/4 = 1{,}75$, $x_2^* = 21/8 = 2{,}625$, $z_{LP}^* = 21{,}5$
Beide Variablen sind **gebrochen** ⇒ Branch & Bound erforderlich.
Wir verzweigen auf x_1 (Most Fractional: $1{,}75 - 1 = 0{,}75 > 2{,}625 - 2 = 0{,}625$):

Knoten 1: $x_1 \leq 1$ Knoten 2: $x_1 \geq 2$

4.3 Knoten 1: $x_1 \leq 1$

Zusätzliche Nebenbedingung: $x_1 \leq 1$, d.h. $x_1 + s_3 = 1$.
LP-Lösung Knoten 1: Mit $x_1 \leq 1$ ergibt der Simplex: $x_1 = 1$, $x_2 = 3$, $z = 17$.
Beide Variablen ganzzahlig!
Neue beste Lösung: $z^* = 17$, $x_1^* = 1$, $x_2^* = 3$.

4.4 Knoten 2: $x_1 \geq 2$

Zusätzliche Nebenbedingung: $x_1 \geq 2$.
LP-Lösung Knoten 2: $x_1 = 2$, $x_2 = 2{,}25$, $z = 19$.
$x_2 = 2{,}25$ ist gebrochen ⇒ Verzweigung auf x_2:

Knoten 3: $x_2 \leq 2$ Knoten 4: $x_2 \geq 3$

4.5 Knoten 3: $x_1 \geq 2$, $x_2 \leq 2$

LP-Lösung: $x_1 = 2{,}5$, $x_2 = 2$, $z = 20{,}5$
$x_1 = 2{,}5$ gebrochen ⇒ Verzweigung auf x_1:

Knoten 5: $x_1 \leq 2$ Knoten 6: $x_1 \geq 3$

4.6 Knoten 4: $x_1 \geq 2$, $x_2 \geq 3$

LP-Lösung: $x_1 = 5/4$, $x_2 = 3$ – aber $x_1 \geq 2$ und $x_1 = 1{,}25 < 2$ ⇒ **Unzulässig!**
Knoten 4 wird verworfen.

4.7 Knoten 5: $x_1 \leq 2$, $x_2 \leq 2$ (und $x_1 \geq 2$)

Damit $x_1 = 2$ (fixiert). **LP-Lösung:** $x_1 = 2$, $x_2 = 2$, $z = 18$.
Beide ganzzahlig! $z = 18 > 17 = z^*$ ⇒ **Neue beste Lösung:** $z^* = 18$.

4.8 Knoten 6: $x_1 \geq 3$, $x_2 \leq 2$

LP-Lösung: $x_1 = 3$, $x_2 = 3/4$, $z = 18$.
$x_2 = 0{,}75$ gebrochen ⇒ Verzweigung auf x_2:

Knoten 7: $x_2 \leq 0$ Knoten 8: $x_2 \geq 1$

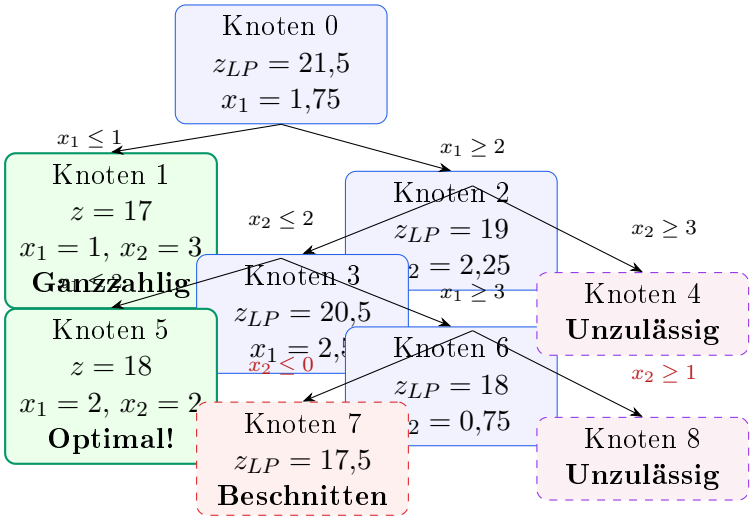
4.9 Knoten 7: $x_1 \geq 3,\ x_2 = 0$

LP-Lösung: $x_1 = 3{,}5,\ x_2 = 0,\ z = 17{,}5$
$z = 17{,}5 \leq z^* = 18 \Rightarrow$ **Beschnitten!**

4.10 Knoten 8: $x_1 \geq 3,\ x_2 \geq 1,\ x_2 \leq 2$

Prüfe: $6(3) + 4(1) = 22 > 21 \Rightarrow$ **Unzulässig!**

4.11 Zusammenfassung: Branch & Bound Baum



Farbcode:

- Blau: LP-Lösung, noch gebrochen
- Grün: Ganzzahlige Lösung gefunden
- Rot gestrichelt: Beschnitten (Schranke $\leq z^*$)
- Lila gestrichelt: Unzulässig

4.12 Optimale Lösung

$$\boxed{x_1^* = 2, \quad x_2^* = 2, \quad z_{ILP}^* = 18}$$

Integrality Gap: $z_{LP}^* - z_{ILP}^* = 21{,}5 - 18 = 3{,}5$ (16,3%)

5 Korrektheit und Konvergenz

Satz 2: Korrektheit des Branch & Bound

Der Branch-&-Bound-Algorithmus findet die optimale ganzzahlige Lösung (oder stellt die Unzulässigkeit fest), falls:

1. Jeder Knoten korrekt durch LP-Lösung bewertet wird,
2. Die Verzweigung vollständig ist (jede ganzzahlige Lösung liegt in genau

einem Teilbaum),

3. Der Baum vollständig ausgewertet wird (oder korrekt beschnitten wurde).

Proof. Durch vollständige Induktion über die Tiefe des B&B-Baums. Jede ganzzahlige Lösung $\mathbf{x} \in \mathbb{Z}^n$ liegt in genau einem Blatt des vollständig entwickelten Baums. Da $x_j \leq \lfloor x_j^* \rfloor$ und $x_j \geq \lceil x_j^* \rceil$ alle ganzzahligen Werte abdecken und disjunkt sind, wird keine zulässige ganzzahlige Lösung übersehen. Das Beschneiden verwirft nur Knoten, deren LP-Schranke die bekannte beste ganzzahlige Lösung nicht übersteigen kann – solche Knoten können keine bessere Lösung enthalten. □

Satz 3: Endlichkeit des Branch & Bound

Bei beschränktem zulässigem Bereich und ganzzahligen Koeffizienten terminiert B&B nach endlich vielen Schritten.

Beweisidee. Die Tiefe des Baums ist durch die Anzahl der Verzweigungsvariablen beschränkt. Bei beschränktem zulässigem Bereich gibt es nur endlich viele ganzzahlige Punkte. Das Beschneiden reduziert die Anzahl der zu untersuchenden Knoten drastisch. □

6 Spezielle Strukturen: Binäre Programme

6.1 0-1-Rucksackproblem

Das klassische **0-1-Rucksackproblem** ist ein wichtiger Spezialfall:

$$\max \sum_{j=1}^{n} c_j x_j \quad \text{s.t.} \quad \sum_{j=1}^{n} a_j x_j \leq b, \quad x_j \in \{0,1\}$$

Schranke durch lineare Relaxierung: Sortiere Gegenstände nach c_j/a_j (absteigend) und fülle den Rucksack der Reihe nach – das letzte Objekt wird gebrochen eingepackt (Greedy-Schranke).

6.2 Cutting Planes

Eine wichtige Ergänzung zu B&B sind **Schnittebenen** (Cutting Planes), die den zulässigen Bereich der LP-Relaxierung einengen, ohne ganzzahlige Lösungen auszuschließen.

Definition: Gomory-Schnitt

Sei Zeile i des Optimaltableaus: $x_{B_i} = \bar{b}_i - \sum_j \bar{a}_{ij} x_j$, wobei $\bar{b}_i \notin \mathbb{Z}$. Der **Gomory-Schnitt** lautet:

$$\sum_j \{\bar{a}_{ij}\} x_j \geq \{\bar{b}_i\}$$

wobei $\{v\} = v - \lfloor v \rfloor$ den gebrochenen Anteil bezeichnet.

Gomory-Schnitte garantieren, dass die LP-Relaxierung nach endlich vielen Schnitten die optimale ganzzahlige Lösung als LP-Lösung liefert.

7 Branch & Bound + Cutting Planes: Branch & Cut

Moderne ILP-Solver kombinieren B&B mit Schnittebenen zum **Branch-&-Cut-Algorithmus**:

1. Löse LP-Relaxierung im Knoten.
2. Füge Schnittebenen hinzu (Gomory, Clique Cuts, etc.).
3. Ist LP-Lösung ganzzahlig: Aktualisiere beste Lösung.
4. Ist LP-Schranke zu schwach: Beschneide.
5. Sonst: Verzweige.

Branch & Cut ist die Grundlage aller modernen ILP-Solver (CPLEX, Gurobi, SCIP).

8 Komplexität und praktische Hinweise

8.1 NP-Schwere

Ganzzahlige lineare Programme sind im Allgemeinen **NP-schwer**: Es gibt keinen bekannten Algorithmus mit polynomieller Laufzeit. Im schlechtesten Fall wächst die Anzahl der B&B-Knoten exponentiell.

Dennoch können moderne Solver (CPLEX, Gurobi) Probleme mit Millionen von Variablen in annehmbarer Zeit lösen – dank:

- **Starker LP-Relaxierung** durch viele Schnittebenen
- **Heuristiken** zur schnellen Findung guter Startlösungen
- **Presolving:** Vereinfachung des Problems vor dem Lösen
- **Parallelisierung** des B&B-Baums

8.2 Praktische Software

- **CPLEX, Gurobi:** Kommerzielle Solver, sehr schnell
- **SCIP, CBC (COIN-OR):** Open-Source-Alternativen
- **Python:** `scipy.optimize.milp`, `PuLP`, `OR-Tools`
- **MATLAB:** `intlinprog`

9 Zusammenfassung

Kernpunkte: Ganzzahlige Programmierung

1. **Ganzzahligkeitsbedingungen** entstehen, wenn Variablen nur ganze Werte annehmen können.
2. Die **LP-Relaxierung** liefert eine obere Schranke; einfaches Runden führt nicht zum Optimum.
3. **Branch & Bound** löst das ILP durch systematisches Verzweigen und Beschneiden.
4. **Bound:** LP-Relaxierung jedes Teilproblems liefert obere Schranke.
5. **Branch:** Verzweigung auf gebrochene Variable durch zwei neue Nebenbedingungen.
6. **Prune:** Knoten mit Schranke $\leq z^*$ werden verworfen.
7. **Integrality Gap** $= z^*_{LP} - z^*_{ILP}$ misst die Schwierigkeit des Problems.
8. **Schnittebenen** (Gomory) verbessern die LP-Schranke ohne ganzzahlige Lösungen auszuschließen.
9. **Branch & Cut** kombiniert B&B mit Schnittebenen – Grundlage moderner ILP-Solver.
10. ILP ist im Allgemeinen **NP-schwer**; moderne Solver lösen große Instanzen dennoch effizient.