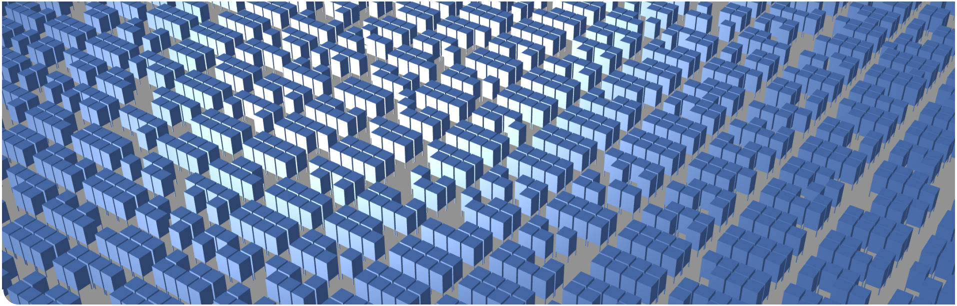
LEUPHANA
UNIVERSITÄT LÜNEBURG

Grundlagen des Operations Research

Teil 6 – Lösung von gemischt-ganzzahligen Optimierungsmodellen

Lin Xie | 16.11.2021

PROF. DR. LIN XIE - WIRTSCHAFTSINFORMATIK, INSBESONDERE OPERATIONS RESEARCH



Wiederholung

Richtig/Falsch-Fragen

1. Wir können das IP-Optimum immer durch die Rundung der optimalen Lösung der LP-Relaxation des IP-Modells finden.

Richtig/Falsch-Fragen

1. Wir können das IP-Optimum immer durch die Rundung der optimalen Lösung der LP-Relaxation des IP-Modells finden.
2. Duality Gap ist der Unterschied zwischen der Lösung der LP-Relaxation und der Lösung des Integer-Problem.

Richtig/Falsch-Fragen

1. Wir können das IP-Optimum immer durch die Rundung der optimalen Lösung der LP-Relaxation des IP-Modells finden.
2. Duality Gap ist der Unterschied zwischen der Lösung der LP-Relaxation und der Lösung des Integer-Problem.
3. Wir können Fixkosten bei der Planung von Produktionsanlagen und Schwellenwerte in Produktionsstrukturen mit Indikatorvariablen modellieren.

Lösung ganzzahliger Optimierungsprobleme

Lösungsprizipien

- Greedy-Methode
- Dynamische Programmierung
- Divide-and-conquer (teile-und-herrsche)
  - Binärsuche
- Backtracking
- Branch-and-Bound-Algorithmus
- Konstruktive Heuristiken
  - Eröffnungs- und Verbesserungsverfahren
- Heuristiken und Metaheuristiken

- Im Folgenden werden die **Greedy-Methode, Backtracking** und **Branch-and-Bound-Algorithmus** besprochen.
- Einige der anderen Techniken werden später in anderen Lehrveranstaltungen diskutiert.

Beispiel Antiquitäten

Beispiel Antiquitäten:
Das Rucksackproblem: Stellen Sie sich vor, Sie wollen die folgenden Antiquitäten gewinnbringend auf dem Flohmarkt verkaufen:

	Vase	Uhr	Radio
Verkaufswert	70 €	50 €	60 €
Gewicht	1 kg	2 kg	3 kg
Wert pro kg	70 €/kg	25 €/kg	20 €/kg

In Ihrem Rucksack können aber nur Gegenstände mit dem Gesamtgewicht von 4 Kilogramm Platz finden. Es ist offensichtlich, dass es nicht möglich ist, alle Antiquitäten auf einmal mitzunehmen. Finden Sie die Kombination mit maximalem Gewinn!

Beispiel Antiquitäten

Beispiel Antiquitäten:

- Entscheidungsvariablen:
  V: wird die Vase mitgenommen (1) oder nicht (0),
  U: wird die Uhr mitgenommen (1) oder nicht (0),
  R: wird das Radio mitgenommen (1) oder nicht (0)

- Mathematisches Modell:

$$\begin{array}{llllll} \text{Max} & 70\text{V} & +\,50\text{U} & +\,60\text{R} & & \\ \text{s.t.} & \text{V} & +\,2\text{U} & +\,3\text{R} & \leq 4 \\ & & & & V, U, R \in \{0, 1\} \end{array}$$

Beispiel Antiquitäten

Greedy-Algorithmus: Dieses „gierige“ Verfahren arbeitet sequentiell und konstruiert eine Lösung in Schritten, die nie revidiert werden.

Beispiel: Knapsack-Problem (Rucksackproblem):

- Ein Rucksack soll mit Gegenständen gefüllt werden, sodass deren Gesamtwert maximal ist.
- Das Gesamtgewicht des Rucksacks ist nach oben beschränkt.
- Gegenstände werden nacheinander eingepackt in der Reihenfolge der absteigenden Wert/Gewicht-Relation.
- Wenn kein Gegenstand mehr hineinpasst, ist eine zulässige Lösung gefunden.
- Der Algorithmus stoppt, obwohl es vielleicht eine bessere Lösung gibt!

Greedy-Algorithmus – Beispiel Antiquitäten

Beispiel Antiquitäten:

	Vase	Uhr	Radio
Verkaufswert	70 €	50 €	60 €
Gewicht	1 kg	2 kg	3 kg
Wert pro kg	70 €/kg	25 €/kg	20 €/kg

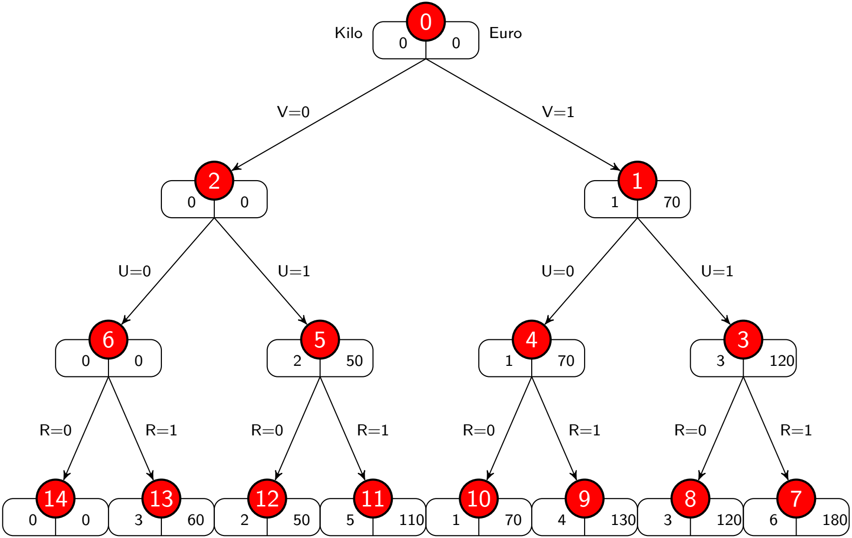
Mit dem Greedy-Algorithmus wird zunächst die Vase eingepackt, dann die Uhr, danach das Radio.
Nachdem die Vase und die Uhr eingepackt wurden, enthält der Rucksack 3 kg mit einem Wert von 120 €. Das Radio passt aber nun nicht mehr in den Rucksack. Also ist der Gewinn nach Anwendung des Greedy-Algorithmus: 120 €
Diese Lösung ist zulässig. Aber ist sie auch optimal?

Backtracking

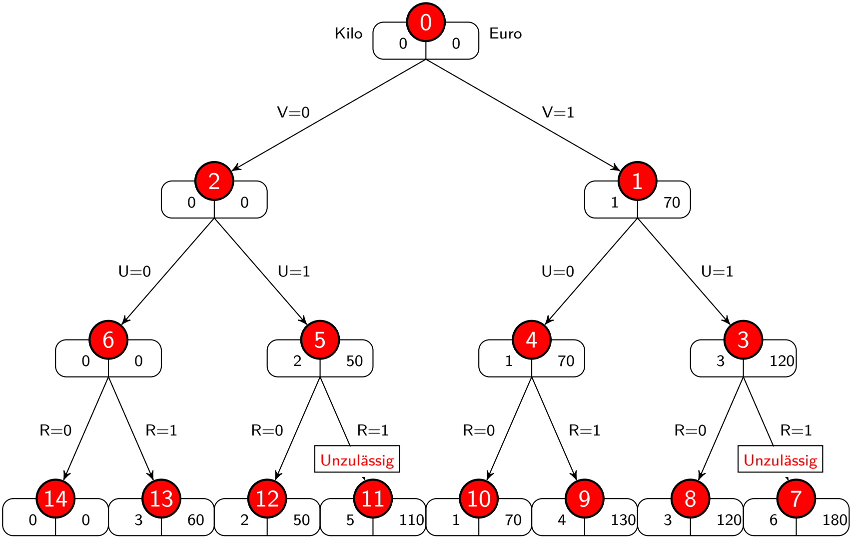
Backtracking:

- Getroffene Entscheidungen können revidiert werden.
- Wenn alle möglichen Entscheidungen (explizit oder implizit) untersucht sind, wurde mit Sicherheit eine optimale Lösung gefunden, falls eine existiert.
- Backtracking kann strukturiert organisiert werden, z. B.:
  - Tiefensuche (Depth-first-search (DFS))
  - Breitensuche (Breadth-first-search (BFS))

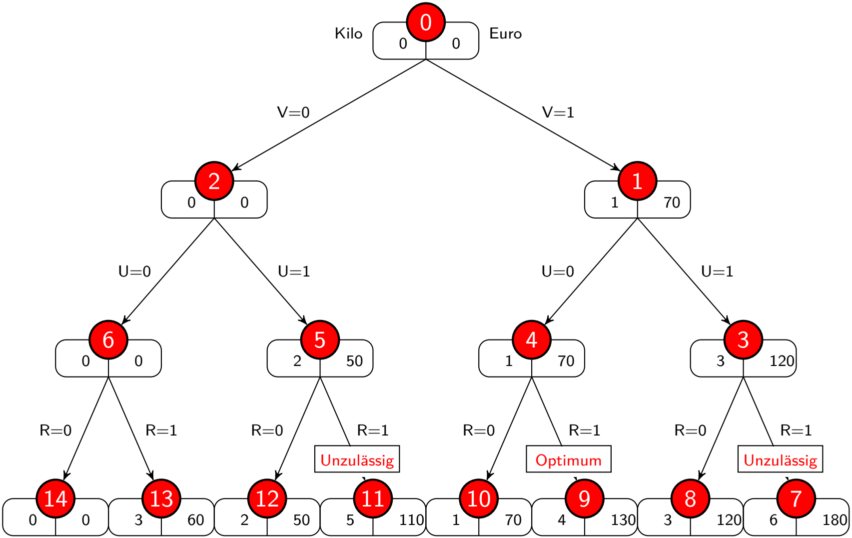
Backtracking



Backtracking



Backtracking



Backtracking mit Bounding

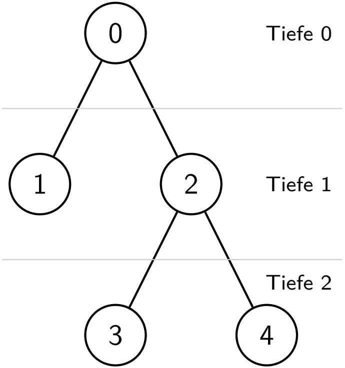
Backtracking mit Bounding:

- Es ist sehr aufwendig, alle möglichen Lösungen zu untersuchen und danach die beste auszuwählen.
- **Idee:** Bereits untersuchte Lösungen können als Schranke für noch zu untersuchende Bereiche genutzt werden.
- Somit muss nur ein (kleiner) Teilbaum untersucht werden.

Backtracking mit Bounding

- Bei einem Knoten K der Tiefe i im *Backtracking-Baum* kann eine obere Schranke der Zielfunktionswerte* aller möglichen Lösungen bestimmt werden, die im Teilbaum mit Wurzel K noch erreicht werden kann.
- **Beispiel:** Berechne bei Knoten 2, Tiefe 1, welcher Zielfunktionswert unterhalb von Knoten 2 noch erreicht werden kann, ohne den Teilbaum unter Knoten 2 zu durchlaufen.

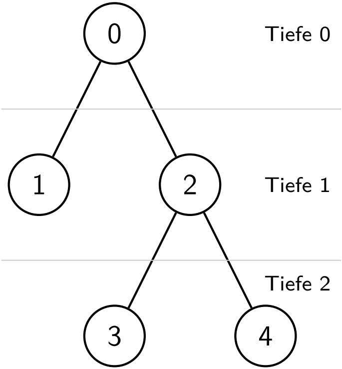
  *untere Schranke für den Fall einer Minimierungs-Zielfunktion



Backtracking mit Bounding

- Falls diese obere Schranke* den Zielfunktionswert der bisher besten Lösung unterschreitet**, so kann dieser Knoten K mit dem Vermerk „bounded“ versehen werden und auf weiteres Durchsuchen des Teilbaumes mit Wurzel K verzichtet werden (Bounding). Denn dieser Teilbaum kann keine bessere Lösung als die bisher beste gefundene mehr enthalten.
- **Beispiel:** Falls $s_2 < z_1$ gilt, so kann es in dem Teilbaum unter Knoten 2 keinen besseren Zielfunktionswert mehr geben als bei Knoten 1. Dieser Teilbaum muss nicht durchsucht werden.

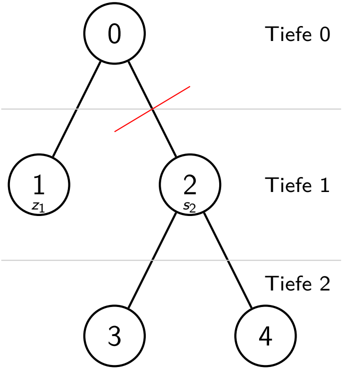
*untere Schranke bei Minimierung
** überschreitet bei Minimierung



Backtracking mit Bounding

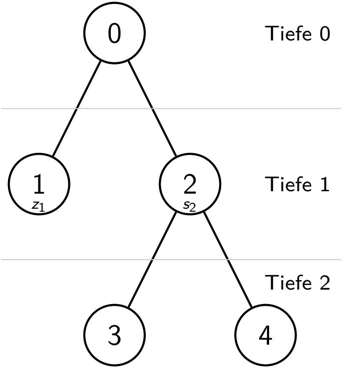
- Falls diese obere Schranke* den Zielfunktionswert der bisher besten Lösung unterschreitet**, so kann dieser Knoten K mit dem Vermerk „bounded“ versehen werden und auf weiteres Durchsuchen des Teilbaumes mit Wurzel K verzichtet werden (Bounding). Denn dieser Teilbaum kann keine bessere Lösung als die bisher beste gefundene mehr enthalten.
- **Beispiel:** Falls $s_2 < z_1$ gilt, so kann es in dem Teilbaum unter Knoten 2 keinen besseren Zielfunktionswert mehr geben als bei Knoten 1. Dieser Teilbaum muss nicht durchsucht werden.

*untere Schranke bei Minimierung
** überschreitet bei Minimierung



Backtracking mit Bounding

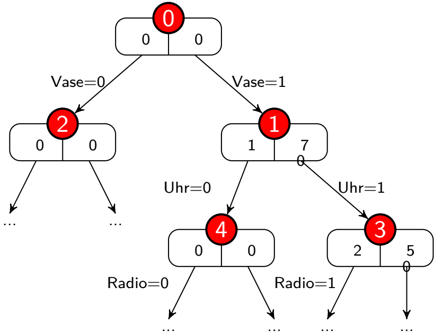
- Wichtig ist, dass die erforderlichen Berechnungen für die Schranke effizient durchgeführt werden (weit unter dem Aufwand für das vollständige Durchsuchen des Teilbaumes).
- Wie bekommen wir eine obere Schranke in effizienter Weise für einen beliebigen Knoten K?
- Z. B. kann die LP-Relaxation genutzt werden, um eine obere Schranke zu bestimmen.



Backtracking mit Bounding

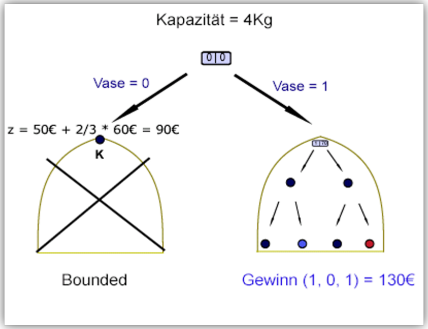
- **Beispiel Antiquitäten :**
  Ab Knoten 2 liegt ein Rucksackproblem kleinerer Größe vor:
  Restliche Güter: Uhr und Radio;
  Kapazität: immer noch 4 (frei ist noch die Restkapazität des ursprünglichen Rucksacks).

- Um eine obere Schranke für Knoten 2 zu bestimmen, kann die optimale Lösung der LP-Relaxation an Knoten 2 genutzt werden.



Backtracking mit Bounding – Beispiel Antiquitäten

- Der rechte Teilbaum wurde schon durchsucht und die bisher beste gefundene Lösung ist $z = 130$.
- Im linken Teilbaum kann bei der Lösung der LP-Relaxation nur ein Zielfunktionswert von höchstens 90 erreicht werden.



- Da $90 < 130$ gilt, kann im linken Teilbaum keine bessere Lösung gefunden werden und muss nicht weiter untersucht werden. Er kann abgeschnitten werden (Bounding).

Branch-and-Bound-Algorithmus – Grundidee

Branch-and-Bound-Algorithmus

- Suchalgorithmus mit Backtracking
- Grundprinzip: *implizite Enumeration* (Erklärung auf nächsten Folien)
- **Branching:** Unterteilung des Problems in Teilprobleme.
- **Bounds** (Schranken) werden durch die Lösung von LP-Relaxationen berechnet:
  1. Eine optimale Lösung für das ganzzahlige Problem kann nicht besser sein als eine optimale Lösung für die LP-Relaxation (weil der Lösungsraum des IP/MIPs kleiner ist)!
     Eine optimale Lösung für die LP-Relaxation bildet daher eine **obere Schranke** für das ganzzahlige Problem.
  2. Eine optimale Lösung für das ganzzahlige Problem muss mindestens so gut sein wie eine zulässige Lösung für das ganzzahlige Problem.
     Eine zulässige Lösung für das ganzzahlige Problem bildet daher eine **untere Schranke** für das ganzzahlige Problem.

Branch-and-Bound-Algorithmus – Grundidee

- Der optimale Zielfunktionswert des ganzzahligen Problems ist also mindestens so gut wie eine zulässige ganzzahlige Lösung (untere Schranke) und höchstens so gut wie die optimale Lösung der LP-Relaxation (obere Schranke):

  Zulässige Lösung MIP/IP $\leq$ Optimale Lösung MIP/IP $\leq$ Optimale Lösung der LP-Relaxation

- **Ziel:** Verkleinere die Differenz zwischen der oberen und unteren Schranke. Damit kann der optimale Zielfunktionswert eingegrenzt werden.
  Diese Informationen nutzt man beim Durchsuchen des Branch-and-Bound-Baumes.

Branch-and-Bound-Algorithmus – Implizite Enumeration

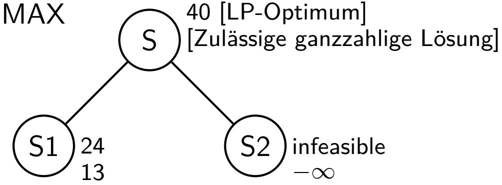
Implizite Enumeration:

- Man teilt den Lösungsraum des Optimierungsproblems in kleinere Teilmengen auf. Dann versucht man für diese Teilmengen zu entscheiden, ob die optimale Lösung in ihnen enthalten ist oder nicht.
- Da man damit nicht den gesamten Lösungsraum explizit durchsucht, heißt dieses Verfahren implizite Enumeration.

Branch-and-Bound-Algorithmus – Abschneiden von Teilbäumen

Was kann von oberen und unteren Schranken in den kleineren Mengen auf die oberen und unteren Schranken in S geschlossen werden?

Fall 1:

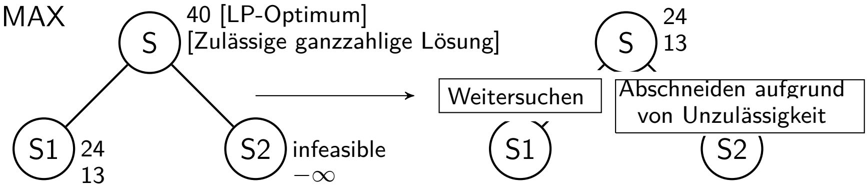


Branch-and-Bound-Algorithmus – Abschneiden von Teilbäumen

Was kann von oberen und unteren Schranken in den kleineren Mengen auf die oberen und unteren Schranken in S geschlossen werden?

Fall 1:

Neue Schranken von S:



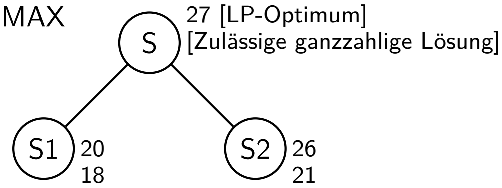
Der Zielfunktionswert von S1 ist mindestens 13 und höchstens 24.
Für die LP-Relaxation in S2 kann keine zulässige Lösung gefunden werden.
Daher muss in S2 nicht weitergesucht werden.

Der Zielfunktionswert von S muss also mindestens 13 sein und kann höchstens 24 sein.

Branch-and-Bound-Algorithmus – Abschneiden von Teilbäumen

Was kann von oberen und unteren Schranken in den kleineren Mengen auf die oberen und unteren Schranken in S geschlossen werden?

Fall 2:

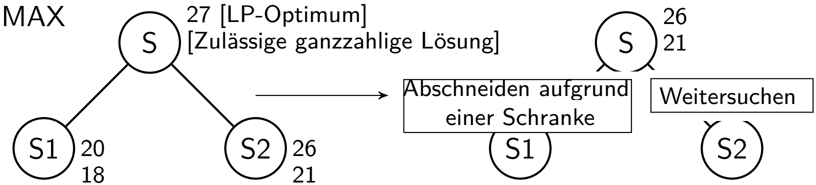


Branch-and-Bound-Algorithmus – Abschneiden von Teilbäumen

Was kann von oberen und unteren Schranken in den kleineren Mengen auf die oberen und unteren Schranken in S geschlossen werden?

Fall 2:

Neue Schranken von S:



Die obere Schranke von S1 ist schlechter als die untere Schranke von S2. Also kann es in S1 keine bessere Lösung geben als in S2. Daher muss in S1 nicht weiter gesucht werden.

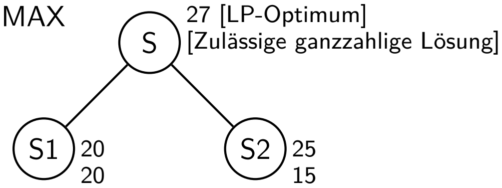
Der Zielfunktionswert von S2 ist mindestens 21 und höchstens 26.

Der Zielfunktionswert von S muss also mindestens 21 sein und kann höchstens 26 sein.

Branch-and-Bound-Algorithmus – Abschneiden von Teilbäumen

Was kann von oberen und unteren Schranken in den kleineren Mengen auf die oberen und unteren Schranken in S geschlossen werden?

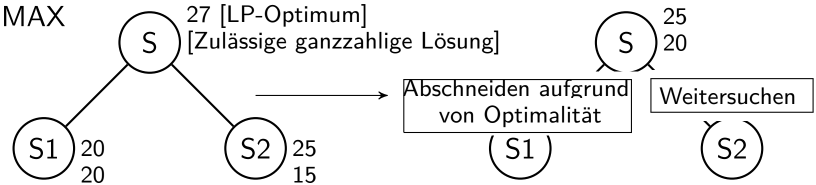
Fall 3:



Branch-and-Bound-Algorithmus – Abschneiden von Teilbäumen

Was kann von oberen und unteren Schranken in den kleineren Mengen auf die oberen und unteren Schranken in S geschlossen werden?

Fall 3: Neue Schranken von S:



S1 ist optimal gelöst, da obere und untere Schranke gleich sind.
Es braucht nicht weiter gesucht werden.
Der Zielfunktionswert von S2 ist mindestens 15 und höchstens 25.
Der Zielfunktionswert von S muss also mindestens 20 sein und kann höchstens 25 sein.

Branch-and-Bound-Algorithmus – Abschneiden von Teilbäumen

Zusammenfassung: Es gibt drei Fälle, bei denen ein Teilbaum abgeschnitten werden kann.

1. Abschneiden aufgrund von Unzulässigkeit (Fall 1)
   Das LP-Modell hat keine zulässige Lösungen. Also gibt es in dem Teilbaum keine zulässigen Lösungen für das IP bzw. MIP.

2. Abschneiden aufgrund einer Schranke (Fall 2)
   Der bestmögliche Zielfunktionswert in diesem Teilbaum kann nicht besser sein als eine bisher gefundene Lösung.

3. Abschneiden aufgrund von Optimalität (Fall 3)
   Die LP-Lösung, also die Belegung der Variablen, ist ganzzahlig.
   [Anmerkung: Der Zielfunktionswert muss nicht unbedingt ganzzahlig sein!]

Branch-and-Bound-Algorithmus

1. **Input:** MIP: maximiere $\{cx : x \in S\}$.
   Output: Eine optimale Lösung x* mit ZF-Wert z, oder MIP ist infeasible und z$= -\infty$.
2. **Initialisierung:** Füge Originalproblem zu der Liste noch zu bearbeitender Knoten L:$=\{S\}$ und setze zip (=beste bisher gefundene ganzzahlige Lösung) $= -\infty$.
3. **Prüfe Abbruchbedingung:** Falls L=$\{\}$, setze x*:=x und z:=zip. Stop.
4. **Knotenauswahl:** Wähle Knoten $S_k \in L$ und aktualisiere die Liste L:=L$\setminus\{S_k\}$.
5. **Berechne obere Schranke:** Löse die LP-Relaxation für S_k und erhalte $z^k := max\{cx : x \in S_k, relaxiert\}$
6. **Bounding/Abschneiden:**
   a) *Abschneiden aufgrund von Unzulässigkeit:* Falls S_k (relaxiert) infeasible, gehe zu Schritt 3. Sonst sei x^k die optimale Lösung der LP-Relaxation von S_k mit z^k als ZF-Wert.
   b) *Abschneiden aufgrund einer Schranke:* Falls $z^k \leq$ zip, gehe zu Schritt 3.
   c) *Abschneiden aufgrund von Optimalität:* Falls x^k zulässig für S_k (also ganzzahlig ist), aktualisiere x:=x^k und zip=z^k und gehe zu Schritt 3.
7. **Branching:** Erstelle zwei Unterprobleme $S_k = S_{k1} \cup S_{k2}$, setze L:=L$\cup\{S_{k1}, S_{k2}\}$. Diesen Unterproblemen wird jeweils eine neue Restriktion hinzugefügt: Wähle dazu eine Variable mit fraktionalem Wert aus:x_i=f.
   Neue Restriktion für $S_{k1} : x_i \leq \lfloor f \rfloor$
   Neue Restriktion für $S_{k2} : x_i \geq \lceil f \rceil$
   Gehe zu Schritt 3.

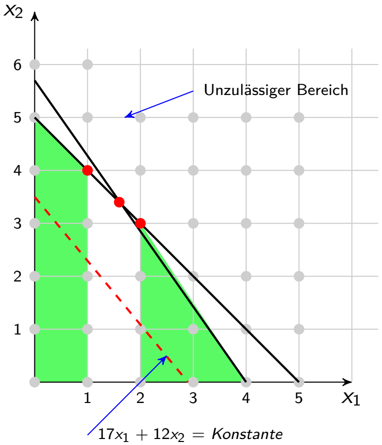
Branch-and-Bound-Algorithmus – Beispiel

Branching: Durch zusätzliche Restriktionen für die Variablen kann das Problem in zwei Teilprobleme unterteilt werden.

z(LP)=68,33

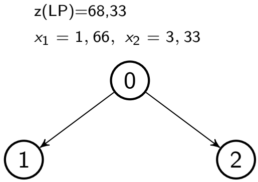
$x_1 = 1,66, \; x_2 = 3,33$

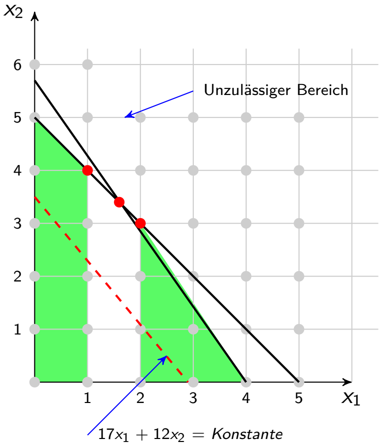
0



Branch-and-Bound-Algorithmus – Beispiel

Branching: Durch zusätzliche Restriktionen für die Variablen kann das Problem in zwei Teilprobleme unterteilt werden.





Branch-and-Bound-Algorithmus – Beispiel

Branching: Durch zusätzliche Restriktionen für die Variablen kann das Problem in zwei Teilprobleme unterteilt werden.

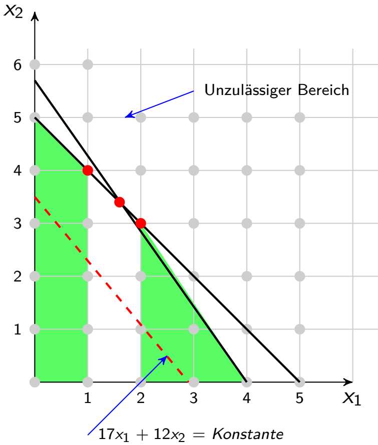
z(LP)=68,33

$x_1 = 1,66, \; x_2 = 3,33$

0

$x_1 \leq 1$

1 2

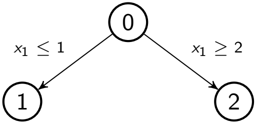


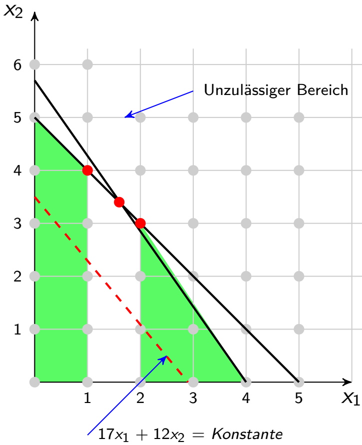
Branch-and-Bound-Algorithmus – Beispiel

Branching: Durch zusätzliche Restriktionen für die Variablen kann das Problem in zwei Teilprobleme unterteilt werden.

z(LP)=68,33

$x_1 = 1,66, \; x_2 = 3,33$



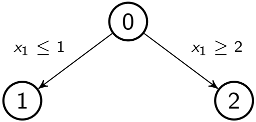


Branch-and-Bound-Algorithmus – Beispiel

Branching: Durch zusätzliche Restriktionen für die Variablen kann das Problem in zwei Teilprobleme unterteilt werden.

z(LP)=68,33

$x_1 = 1,66,\ \ x_2 = 3,33$



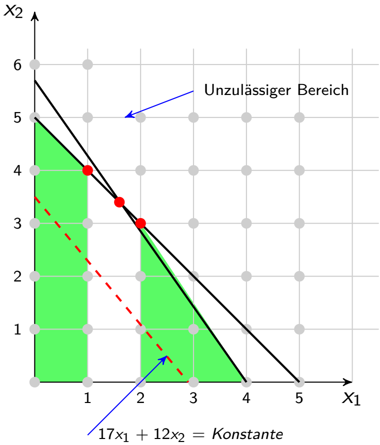
$\max 17x_1 + 12x_2$

$10x_1 + 7x_2 \leq 40$

$x_1 + x_2 \leq 5$

$x_1 \leq 1$

$x_1, x_2 \geq 0$

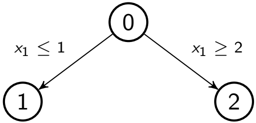


Branch-and-Bound-Algorithmus – Beispiel

Branching: Durch zusätzliche Restriktionen für die Variablen kann das Problem in zwei Teilprobleme unterteilt werden.

z(LP)=68,33

$x_1 = 1,66, \; x_2 = 3,33$



max $17x_1 + 12x_2$

$10x_1 + 7x_2 \leq 40$

$x_1 + x_2 \leq 5$

$x_1 \leq 1$

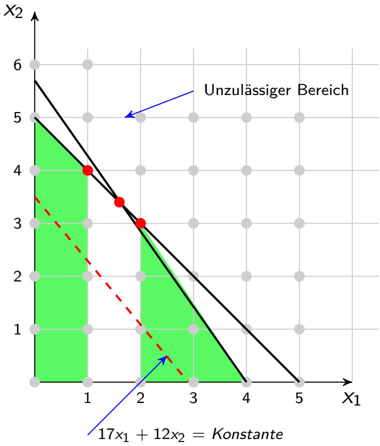
$x_1, x_2 \geq 0$

Lösung:

z(LP)=65

$x_1 = 1$

$x_2 = 4$

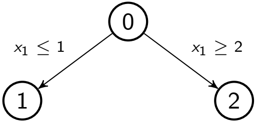


Branch-and-Bound-Algorithmus – Beispiel

Branching: Durch zusätzliche Restriktionen für die Variablen kann das Problem in zwei Teilprobleme unterteilt werden.

z(LP)=68,33

$x_1 = 1,66, \; x_2 = 3,33$



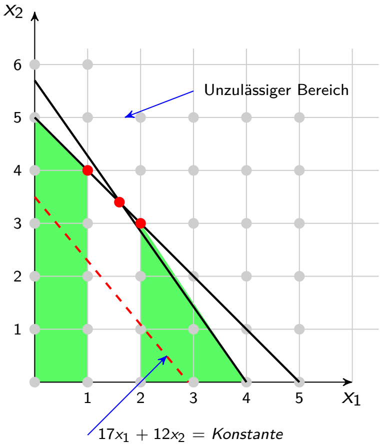
max $17x_1 + 12x_2$	max $17x_1 + 12x_2$
$10x_1 + 7x_2 \leq 40$	$10x_1 + 7x_2 \leq 40$
$x_1 + x_2 \leq 5$	$x_1 + x_2 \leq 5$
$x_1 \leq 1$	$x_1 \geq 2$
$x_1, x_2 \geq 0$	$x_1, x_2 \geq 0$

Lösung:

z(LP)=65

$x_1 = 1$

$x_2 = 4$

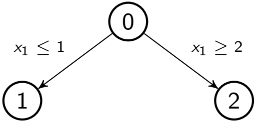


Branch-and-Bound-Algorithmus – Beispiel

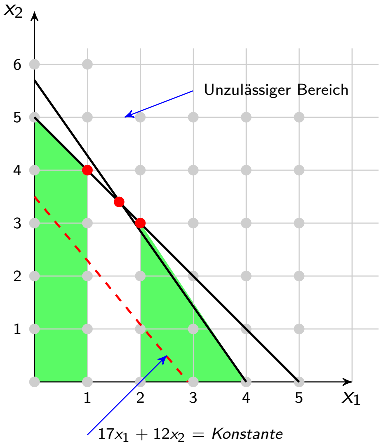
Branching: Durch zusätzliche Restriktionen für die Variablen kann das Problem in zwei Teilprobleme unterteilt werden.

z(LP)=68,33

$x_1 = 1,66, \; x_2 = 3,33$



Teilproblem 1		Teilproblem 2	
max $17x_1 + 12x_2$		max $17x_1 + 12x_2$	
$10x_1 + 7x_2 \leq 40$		$10x_1 + 7x_2 \leq 40$	
$x_1 + x_2 \leq 5$		$x_1 + x_2 \leq 5$	
$x_1 \leq 1$		$x_1 \geq 2$	
$x_1, x_2 \geq 0$	Lösung:	$x_1, x_2 \geq 0$	Lösung:
	z(LP)=65		z(LP)=68,29
	$x_1 = 1$		$x_1 = 2,00$
	$x_2 = 4$		$x_2 = 2,86$



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

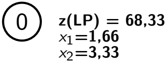
- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]

0

Branch-and-Bound-Algorithmus – Beispiel



Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]

Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

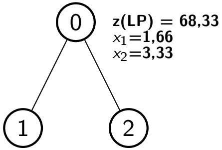
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

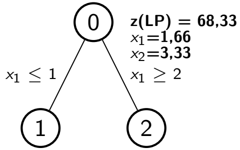
- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel



Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

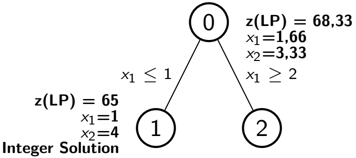
haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]

Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.



Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]

Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

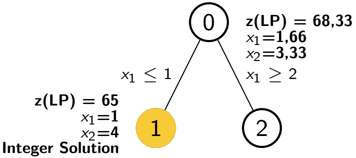
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert < zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

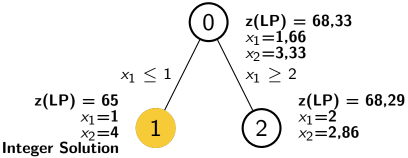
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert < zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

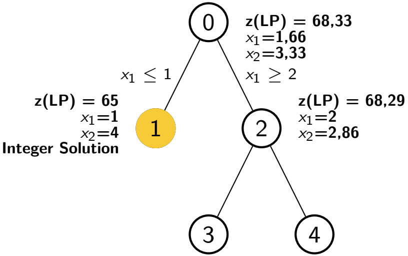
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

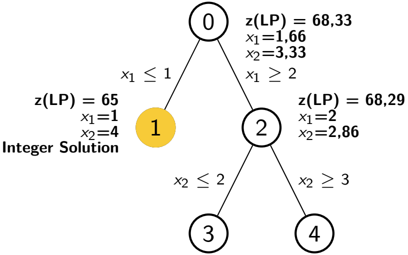
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert < zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

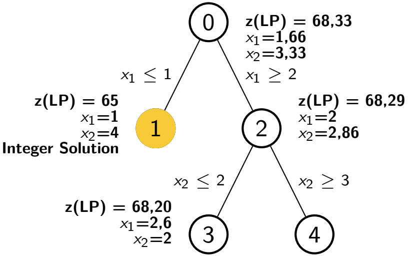
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

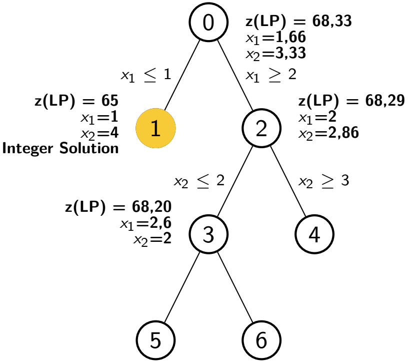
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

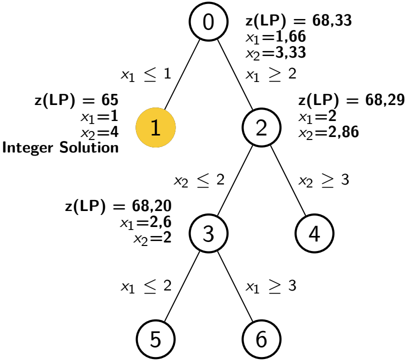
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert < zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

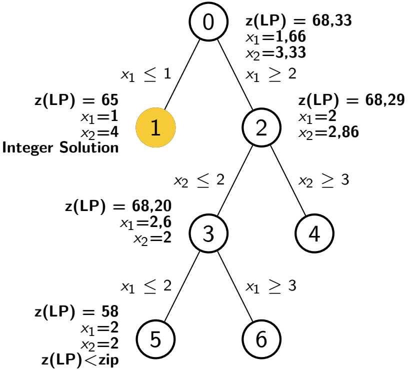
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

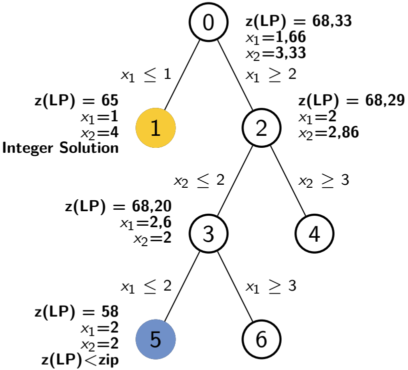
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

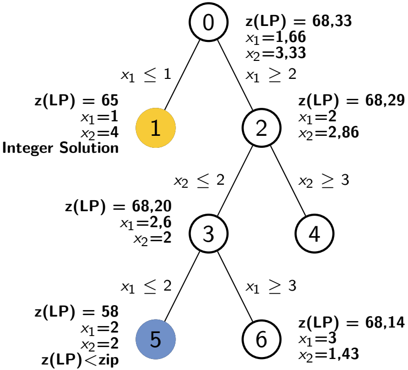
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert < zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

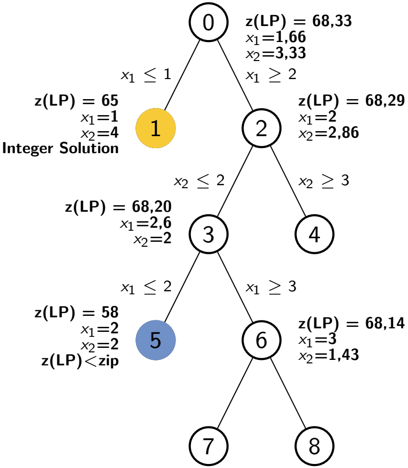
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert < zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

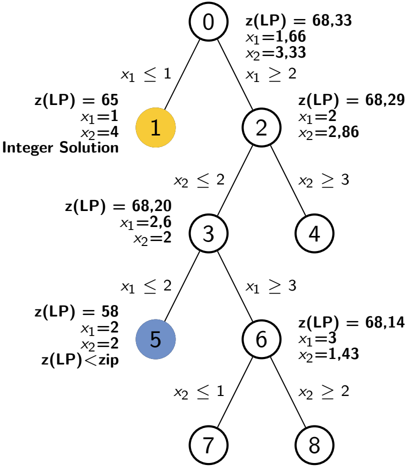
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert < zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

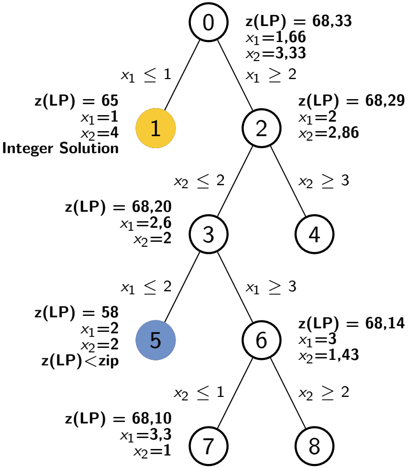
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

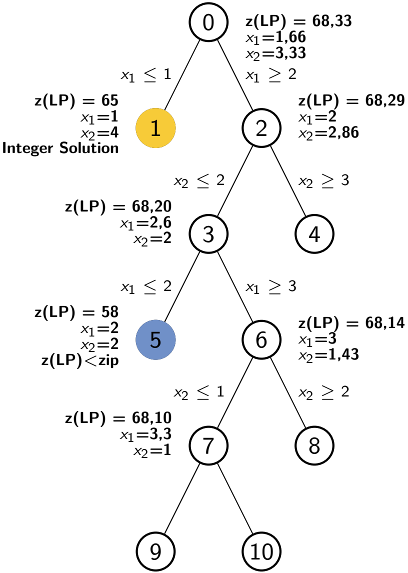
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

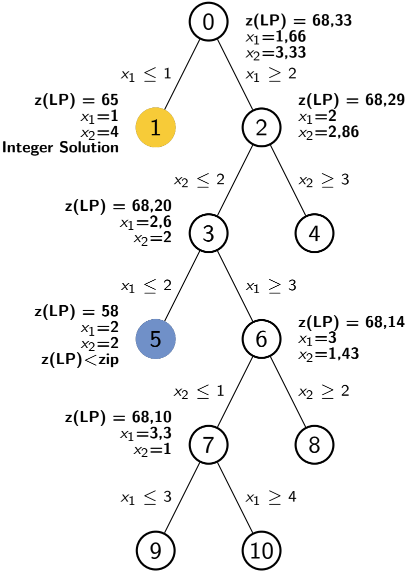
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

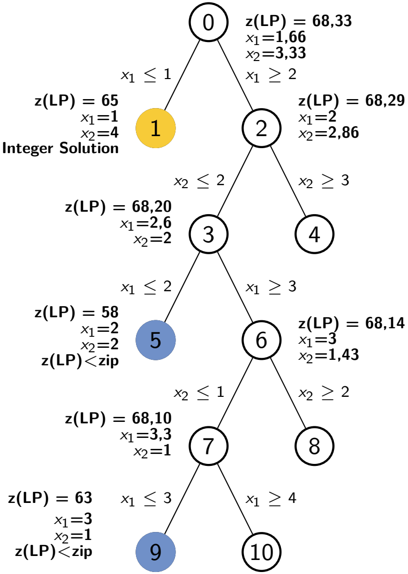
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

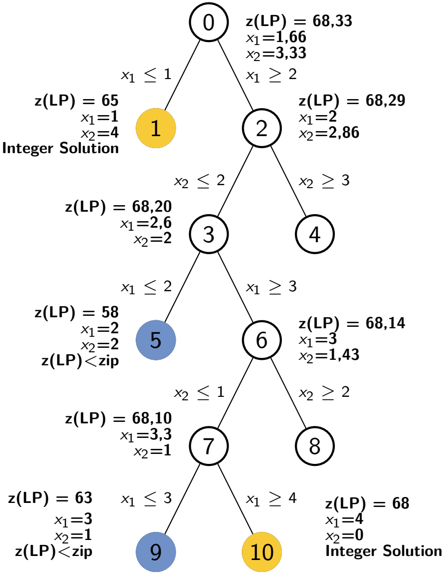
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

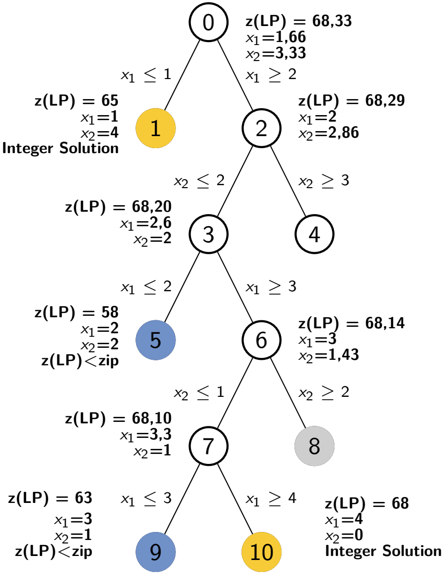
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert < zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus – Beispiel

Die Knotennummer gibt an, in welcher Reihenfolge die Teilprobleme **erstellt** wurden.

Auswahlregel der Knoten:
Tiefensuche, linker Knoten vor rechtem Knoten.

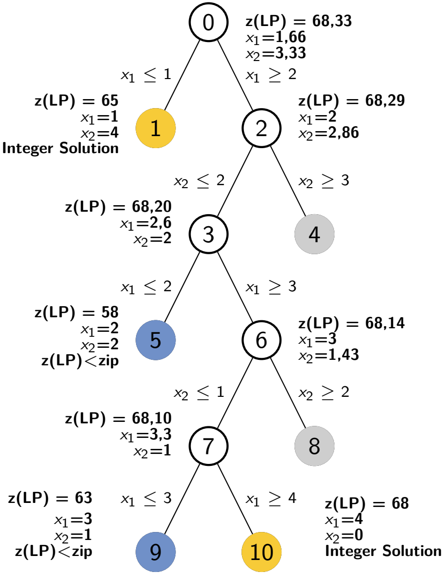
Reihenfolge der Bearbeitung:
0-1-2-3-5-6-7-9-10-8-4

Teilprobleme die

- unzulässig sind (Knoten 4, 8)
- LP-ZF-Wert $<$ zip aufweisen (Knoten 5, 9)
- ganzzahlig sind (Knoten 1, 10)

haben keine Nachfolger.

[zip = bisher beste gefundene ganzzahlige Lösung]



Branch-and-Bound-Algorithmus

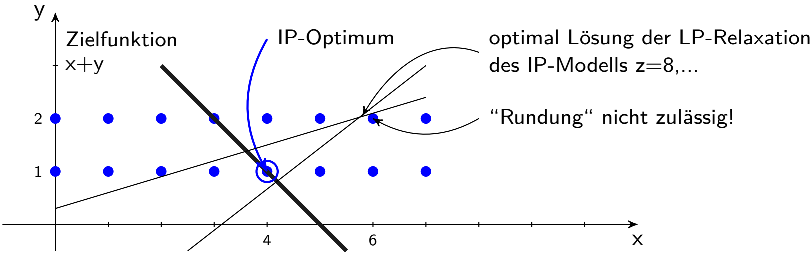
- In der Praxis gibt es viele Varianten zur Organisation des B&B-Verfahrens.
  Wichtigste Fragen:
  - Welche Variable wird zum Branching ausgewählt?
  - Welcher Knoten aus der Liste wird als nächstes bearbeitet?

- Branching-Variable
  - z.B. am meisten fraktional
  - Oder am wenigsten fraktional

Unterschiede zwischen LPs and MIPs

Unterschied zwischen LPs and MIPs

- Eine optimale Lösung für das ganzzahlige Problem kann nicht besser sein als eine optimale Lösung für die LP-Relaxation (weil der Lösungsraum verkleinert wird)!
- Rundung der optimalen Lösung der LP-Relaxation liefert in der Regel nicht die optimale Lösung des MIPs!



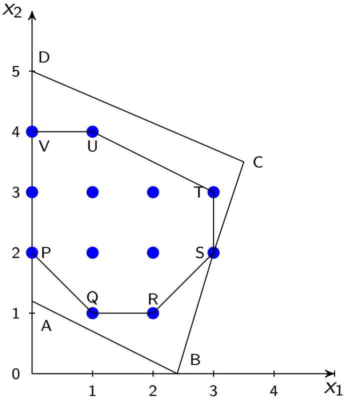
Unterschied zwischen LPs and MIPs

- Lösungsaufwand bei LPs: hängt stark von Anzahl der Variablen und Restriktionen im Modell ab.
- Lösungsaufwand bei MIPs: „gute“ Formulierung wichtig für geringen Lösungsaufwand. Oft werden zusätzliche Restriktionen hinzugefügt, um die Lösung zu erleichtern.
- Optimale Lösung eines LPs: liegt immer in einem Eckpunkt des Lösungsraumes.
- Optimale Lösung eines MIPs: Lösungsraum ist nicht zusammenhängend, daher gibt es auch keine Eckpunkte.
- Wenn „Lösungsraum“ von LP und MIP identisch wären, wäre die optimale Lösung des LPs gleichzeitig auch die optimale Lösung des MIPs.
  - Wünschenswert, da LPs im Allgemeinen leichter lösbar sind als IPs.

Unterschied zwischen LPs and MIPs

- **Daher ist das Ziel:** möglichst knappe Abgrenzung des zulässigen Bereichs des IPs.
  Kleinste Möglichkeit: *Konvexe Hülle* der zulässigen ganzzahligen Punkte.

  Konvexe Hülle: kleinste konvexe Menge, die alle zulässigen Punkte des IPs enthält.
  Dann sind die Lösung des IPs und die Lösung der LP-Relaxation des IPs identisch.



- Leider ist die Bestimmung der konvexen Hülle ein genauso schwieriges Problem wie das eigentliche Lösen des ganzzahligen Optimierungsproblems.

Fazit und Ausblick

Fazit und Ausblick

- Lernziele
  - Allgemeine Lösungsprinzipien für ganzzahlige Optimierungsprobleme
    - Greedy-Algorithmus
    - Backtracking
    - Branch-and-Bound

- Nächste Vorlesung
  - Logische Abhängigkeiten
  - Spezielle Modellierungstechniken

Literatur

- L. Suhl, T. Mellouli. Optimierungssysteme – Modelle, Verfahren, Software, Anwendungen. 3. Auflage, Springer Gabler, Berlin/Heidelberg, 2013, Seite 131–160.

Danke für Ihre Aufmerksamkeit!

Leuphana Universität Lüneburg
Wirtschaftsinformatik, insbesondere Operations Research
Prof. Dr. Lin Xie
Universitätsallee 1
Gebäude 4, Raum 314
21335 Lüneburg
Fon +49 4131 677 2305
Fax +49 4131 677 1749
xie@leuphana.de