1.4 BRANCH AND BOUND

Im SUBSET-SUM-Beispiel 1.10 haben wir gesehen, dass sich auch beim Enumerieren Arbeit einsparen lässt!

Zur Erinnerung - wir hatten notiert:

- Vorwärtsrekursion statt Rückwärtsrekursion
  ↳ Dynamic Programming

- Schneide Teilbäume ab wenn möglich
  ↳ Branch and Bound

verzweige beschränke

Das betrachten wir in diesem Teilkapitel!

Grundidee:

(1) Enumeriere die möglichen Teilmengen in einen Enumerationsbaum

(2) Behalte dabei den Zielfunktionswert im Auge:
- Untere Schranke: Erreichter Zielfunktionswert im ganzen Enumerationsbaum
- Obere Schranke: Erreichbarer Zielfunktionswert im aktuellen Teilbaum

(3) Beide Schranken sollten einfach und schnell zu berechnen sein

(4) Wenn der erreichbare Wert kleiner bleiben muss als der bereits erreichte, können wir den aktuellen Teilbaum abschneiden

Wir betrachten noch einmal

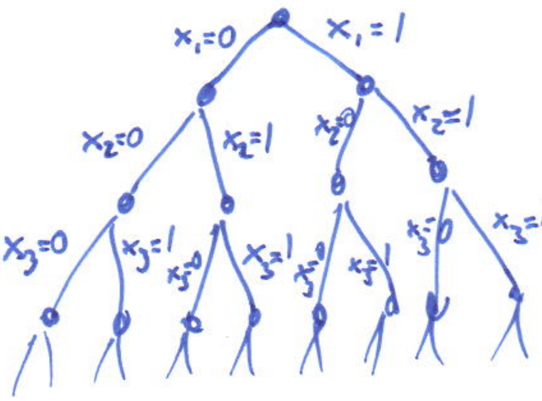
Beispiel 1.14 (Knapsack)

i	1	2	3	4	5	6	7
z_i	2	3	6	7	5	9	4
p_i	6	5	8	9	6	7	3

$Z = 9$, $n = 7$

(1) Wie können wir das enumerieren?

- Probiere nacheinander für $i = 1, \dots, 7$, ob $x_i = 0$ oder 1
- Gehe die Möglichkeiten baumartig durch:



- Laufe DFS-mäßig durch den Baum, d.h. arbeite die Entscheidungen im Stack ab:
  - Probiere $x_1 = 0$
    - Probiere $x_2 = 0$
      - Probiere $x_3 = 0$

        ⋮

| Wenn ein $x_i = 0$ nicht mehr funktioniert, probiere $x_i = 1$ |

(2) welche untere und obere Schranke haben wir zur Verfügung?

Untere Schranke: Greedy!

. Verwende unter den noch nicht zugewiesenen Objekten in aufsteigendem Kosten/Nutzen-Verhältnis Objekte, solange sie passen.

Hier ohne fixierte Objekte:

$i=1$: $\sum_{i \in S} z_i = 2$, $\sum_{i \in S} p_i = 6$ ✓

$i=2$: $\sum_{i \in S} z_i = 5$, $\sum_{i \in S} p_i = 11$ ✓

($i=3$: $\sum_{i \in S} z_i = 11$ ×)

($i=4$: $\sum_{i \in S} z_i$ 12 ×)

($i=5$: $\sum_{i \in S} z_i = 10$ ×)

($i=6$: $\sum_{i \in S} z_i = 14$ ×)

$i=7$: $\sum_{i \in S} z_i = 9$, $\sum_{i \in S} p_i = 14$

Gesamtnutzen 14 ist also erreichbar!

Obere Schranke:

Greedy für das einfachere Problem „Fractional Knapsack",
d.h. Algorithmus 1.4.

$i=1: \quad \sum_{i\in S} z_i = 2 \quad , \quad \sum_{i\in S} p_i = 6$

$i=2: \quad \sum_{i\in S} z_i = 5 \quad , \quad \sum p_i = 11$

Bleibt $Z - \sum z_i = 4$, also

$$x_3 = \frac{2}{3} \quad \left(= \frac{4}{6} = \frac{Z - \sum z_i}{z_3} \right)$$

Damit:
$$\sum_{i=1}^{3} x_i p_i = 11 + \frac{2}{3} \cdot 8 = 16\frac{1}{3} .$$

Wir wissen, dass die Optimallösung ganzzahlig sein muss, also haben wir eine obere Schranke von 16 !

(3) Diese Schranken sind beide einfach zu berechnen!

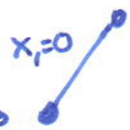
WICHTIG: Für die obere Schranke betrachten wir ein Hilfsproblem, das einfacher ist, weil wir weniger strenge Bedingungen verlangen.
So etwas nennt man eine Relaxierung !

(4) Betrachten wir den Baum
- und dabei jeweils

S: Positiv fixiert $(x_i = 1)$
$\bar{S}$: Negativ fixiert $(x_i = 0)$

• ← $S = \emptyset$, $\bar{S} = \emptyset$; $LB = 14$, $UB = 16$

Solange $LB \leq UB$, machen wir weiter
und $S \cup \bar{S} \neq \{1, \dots, 7\}$



$S = \emptyset$, $\bar{S} = \{1\}$ → ← Berechne UB!

$x_1 = 0$ bedeutet: Objekt 1
ist ausgeschlossen

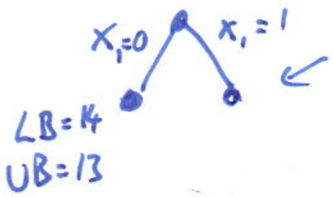
Damit Greedy für Fractional Knapsack:

$x_2 = 1 \Rightarrow \sum x_i z_i = 3$
$x_3 = 1 \Rightarrow \sum x_i z_i = 9 \Rightarrow x_4 = \dots = x_7 = 0$

Damit ist $\sum x_i p_i = 13 < 14$

Also ist dann nicht mehr als
Nutzen 13 erreichbar; das ist
schlechter als 14, $x_1 = 0$ ist also
eine Sackgasse!

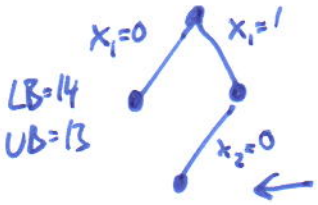
Also



Berechne UB : 16

Berechne LB : 14 (wie gehabt)

Dann



$S=\{1\}$, $\bar{S}=\{2\}$

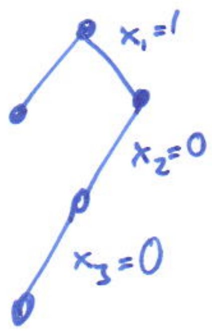
Berechne UB: $x_1=1$, $x_3=1$, $x_4=\frac{1}{7}$, $\sum_{i=1}^{7} x_i p_i = 15\frac{2}{7}$

Also UB = 15 !

Berechne LB: $x_1=1$, $x_3=1$ $\sum_{i=1}^{7} x_i p_i = 14$

Also LB = 14 !

Dann



Berechne UB: $x_1=1$, $x_4=1$, $\sum_{i=1}^{7} x_i p_i = 15$

Also UB = 15 !

Berechne LB: $x_1=1$, $x_4=1$, $\sum_{i=1}^{7} x_i p_i = 15$

Also LB = 15 !