



## Material zum Rucksackproblem

(Aufgabe 6 in Serie 4)

### Inhaltsverzeichnis

|   |          |
|---|----------|
| <b>1 Rucksackprobleme</b>   | <b>2</b> |
| 1.1 Gegebene Daten . . . . .  | 2        |
| 1.2 Problemstellung: Rucksackproblem mit oberen Schranken . . . . .                                   | 2        |
| 1.3 Mathematisches Modell: Rucksackproblem mit oberen Schranken . . . . .                             | 2        |
| 1.4 Voraussetzungen . . . . .   | 3        |
| <b>2 Greedy - Heuristik</b>   | <b>3</b> |
| <b>3 Ein spezieller Branch &amp; Bound - Algorithmus für das Rucksackproblem mit oberen Schranken</b> | <b>4</b> |
| 3.1 Schranken . . . . .   | 4        |
| 3.2 Algorithmus . . . . .   | 4        |
| <b>4 Beispiele</b>  | <b>6</b> |

# 1 Rucksackprobleme

## 1.1 Gegebene Daten

- Rucksack mit ganzzahligem Volumen  $V \in \mathbb{Z}$
- $n$  kleinere Gegenstände  $G_i = (w_i, v_i, b_i), i = 1, \dots, n$  mit
  - Volumen  $v_i \leq V, v_i \in \mathbb{Z}_+$
  - Wert  $w_i, w_i \in \mathbb{Z}, \mathbb{P}, \mathbb{R}$
  - Vorrat  $b_i, b_i \in \mathbb{Z}_+$

## 1.2 Problemstellung: Rucksackproblem mit oberen Schranken

Finde Häufigkeiten  $x^* = (x_1^*, \dots, x_n^*)^T \in \mathbb{Z}_+^n$  der Gegenstände  $G_i, i = 1, \dots, n$ , die den Wert  $w^T x^*$  der in den Rucksack gepackten Gegenstände maximiert, ohne dabei das Volumen des Rucksacks zu überschreiten ( $v^T x^* \leq V$ ) und ohne die Vorratsbedingungen  $x^* \leq b$  zu verletzen.

## 1.3 Mathematisches Modell: Rucksackproblem mit oberen Schranken

Ganzzahliges Lineares Optimierungsproblem

$$\begin{aligned} z = w^T x &\longrightarrow \max \\ \text{bei } v^T x &\leq V \\ x &\leq b \\ x &\in \mathbb{Z}_+^n \\ &\text{mit } V \in \mathbb{Z}_+, v, b \in \mathbb{Z}_+^n, w \in \mathbb{R}_+^n \end{aligned}$$

**Bemerkung:** Das ist ein Rucksackproblem mit oberen Schranken  $b_i$  für die Häufigkeiten  $x_i$  der Gegenstände  $G_i$  und nur eine mögliche Form eines Rucksackproblems. Andere wichtige Arten von Rucksackproblemen sind:

- $\max w^T x$  bei  $v^T x \leq V, x_i \in \{0, 1\}, v_i \in \mathbb{Z}_+$  Binär
- $\max w^T x$  bei  $v^T x \leq V, x_i \in \mathbb{Z}_+, v_i \in \mathbb{Z}_+$  Unbeschränkt
- $\max w^T x$  bei  $v^T x \leq V, x_i \in \mathbb{Z}_+, v_i \in \mathbb{P}_+$  Rational

Zum Teil kann man durch einfache Transformationen eine Form des Rucksackproblems in eine andere überführen. So kann man z.Bsp. obiges Rucksackproblem mit oberen Schranken  $b_i$  durch Einführung von Gegenständen  $\bar{G}_j$  und zugehörigen binären Variablen  $y_j, j = 1, \dots, m_n$  mit  $m_k = \sum_{i=1}^k b_i$  und  $\bar{w}_j = w_{\bar{i}}, \bar{v}_j = v_{\bar{i}}$  wobei  $\bar{i}(j) = \operatorname{argmax}\{k = 1, \dots, n \mid j \leq m_k = \sum_{i=1}^k b_i\}$  in ein Binäres Rucksackproblem umwandeln. Man hat also für jedes mögliche Exemplar des Gegenstandes  $G_i$  eine eigene Variable  $y_j, m_{j-1} < j \leq m_j$  mit  $m_0 = 0$ , eingeführt. Das entsprechende binäre Rucksackproblem lautet nun:

$$\begin{aligned} \max \quad & \sum_{j=1}^{m_n} \bar{w}_j^T y_j \\ \text{bei} \quad & \sum_{j=1}^{m_n} \bar{v}_j^T y_j \leq V \\ & x \in \{0, 1\} \\ & \text{mit } V \in \mathbb{Z}_+, \bar{v}_j \in \mathbb{Z}, \bar{w}_j \in \mathbb{R}, 1 \leq j \leq m_n \end{aligned}$$

Aus der Lösung  $y^* \in \{0, 1\}^{m_n}$  des binären Problems kann man wie folgt eine Lösung des Ausgangsproblems mit oberen Schranken konstruieren:

$$x_i^* = \sum_{m_{i-1}+1}^{m_i} y_i^*$$

## 1.4 Voraussetzungen

Im Folgenden wird ein einfacher spezieller Algorithmus<sup>1</sup> zur Lösung des Rucksackproblems mit oberen Schranken für die Variablen  $x_i$  angegeben. Dabei setzen wir insbesondere eine Sortierung der Gegenstände  $G_i$  nach abnehmenden Quotienten  $\frac{w_i}{v_i}$  voraus, es soll also gelten:

$$\frac{w_1}{v_1} \geq \frac{w_2}{v_2} \geq \dots \geq \frac{w_n}{v_n}.$$

Mit anderen Worten: Der erste Gegenstand hat den höchsten Wert pro Volumeneinheit, der letzte den kleinsten Wert.

Falls die Daten nicht dieser Sortierungsbedingung genügen, dann muss vor dem Anwerfen des Algorithmus entsprechend sortiert werden. Im Folgenden gehen wir davon aus, dass die Sortierung nach fallendem *Wert pro Volumeneinheit* vorliegt.

## 2 Greedy - Heuristik

Der Name „gefällig“ beschreibt das Vorgehen dieses einfachen Näherungsalgorithmus<sup>2</sup> sehr gut: Von dem am meisten Wert pro Volumeneinheit versprechenden Gegenstand werden so viele Exemplare wie möglich eingepackt und dann das verbleibende Volumen mit dem nächstbesten Gegenstand gefüllt. Die Anzahl, wie oft ein Gegenstand  $G_i$  eingepackt werden kann ist dabei begrenzt durch das verfügbare (Rest)volumen  $\tilde{V}$  und die maximale Anzahl  $b_i$ :

$$x_i = \min \left\{ \left\lfloor \frac{\tilde{V}}{v_i} \right\rfloor, b_i \right\},$$

wobei  $\lfloor a \rfloor = \max\{k \in \mathbb{Z} \mid k \leq a\}$  die größte *ganze* Zahl bezeichnet, die kleiner als  $a \in \mathbb{R}$  ist.

---

**Algorithm 1** Greedy – Heuristik für Rucksackprobleme mit oberen Schranken

---

1: **procedure** GREEDY( $\tilde{V}, v, b, g$ ) ▷ Näherungslösung mittels Greedy  
**Eingabe:**  
 $\tilde{V}$            Volumen des Rucksacks            $n \in \mathbb{Z}_+$    Anzahl der Gegenstände  
 $v \in \mathbb{Z}_+^n$     Volumen der Gegenstände        $b \in \mathbb{Z}_+^n$    Vorräte der Gegenstände  
**Ausgabe:**  
 $g \in \mathbb{Z}_+^n$    Häufigkeiten der Gegenstände  
**Require:** Sortierung der Gegenstände so dass  $\frac{w_1}{v_1} \geq \frac{w_2}{v_2} \geq \dots \geq \frac{w_n}{v_n}$  gilt.  
2:   **for**  $i \leftarrow 1, n$  **do**  
3:      $g_i \leftarrow \min \left\{ \left\lfloor \frac{\tilde{V}}{v_i} \right\rfloor, b_i \right\}$   
4:      $\tilde{V} \leftarrow \tilde{V} - v_i * g_i$   
5:   **end for**  
6: **end procedure**

---

Der Wert der Greedy-Lösung berechnet sich als  $\sum_{i=1}^n w_i g_i$ . Beachtenswert ist noch, dass die Werte  $w_i$  der Gegenstände erst bei der Berechnung des Wertes gebraucht werden nicht aber zur Bestimmung der Greedy-Lösung  $g \in \mathbb{Z}_+^n$  (das liegt an der speziellen Sortierung und den darauf abgestimmten Algorithmus, der den Wert nicht mit berechnet).

<sup>1</sup>In der Vorlesung wurde ein allgemeiner B&B - Algorithmus angegeben, hier werden wir zwar Grundkenntnisse über B&B aus der Vorlesung voraussetzen, ansonsten aber einen einfachen Algorithmus speziell für das Rucksackproblem mit oberen Schranken erarbeiten. Weiterhin ist zu beachten, dass in der Vorlesung ein B&B-Algorithmus für Minimierungsprobleme angegeben wurde, hier aber maximiert wird. Das heißt z.Bsp. dass hier *obere* und nicht *untere* Schranken für Teilprobleme benötigt werden.

<sup>2</sup>Eine *Heuristik* ist ein Verfahren, das für eine Problemklasse in vielen Fällen ziemlich gute Resultate liefert, allerdings ist dabei Optimalität nicht garantiert. Meist beruhen Heuristiken auf einfachen Beobachtungen an einigen Beispielen. Oft erhält man mit diesen im Vergleich zu exakten Verfahren (die die Optimalität garantieren) sehr einfachen Algorithmen sehr gute Resultate in einem Bruchteil der Rechenzeit exakter Verfahren.

## 3 Ein spezieller Branch & Bound - Algorithmus für das Rucksackproblem mit oberen Schranken

### 3.1 Schranken

Schranken (engl. *bounds*) werden im Branch & Bound – Verfahren benutzt, um Teilprobleme zu bewerten, die durch die Verzweigung (engl. *branch*) entstanden sind: Bei einem max - Problem wie dem Rucksackproblem will man durch die Schranke erfahren, ob die Abarbeitung eines Teilproblems keine bessere Lösung als die bisher beste gefundene Lösung liefern kann. Wenn das so ist, dann muss dieses Teilproblem nicht weiter untersucht werden (Schritt 6 im Algorithmus aus der Vorlesung).

Im vorliegenden Fall von nach fallendem Wert pro Volumeneinheit sortierten Gegenständen erhält man sehr einfach zu berechnende obere Schranken<sup>3</sup>:

$$\begin{aligned} B_1(\tilde{V}, i) &= \tilde{V} * \frac{w_i}{v_i} \\ B_2(\tilde{V}, i) &= \alpha * w_i + (\tilde{V} - \alpha * v_i) \frac{w_{i+1}}{v_{i+1}} \\ &\quad \text{mit } \alpha = \min \left\{ \left\lfloor \frac{\tilde{V}}{v_i} \right\rfloor, b_i \right\} \\ B_3(\tilde{V}, i) &= \alpha * w_i + \beta w_{i+1} + (\tilde{V} - \alpha * v_i - \beta * v_{i+1}) \frac{w_{i+2}}{v_{i+2}} \\ &\quad \text{mit } \alpha = \min \left\{ \left\lfloor \frac{\tilde{V}}{v_i} \right\rfloor, b_i \right\}, \quad \beta = \min \left\{ \left\lfloor \frac{\tilde{V} - \alpha v_i}{v_{i+1}} \right\rfloor, b_{i+1} \right\} \end{aligned}$$

Die erste Schranke *füllt* das komplette zur Verfügung stehende Volumen  $\tilde{V}$  mit dem  $i$ -ten Gegenstand, und zwar ohne Rücksicht auf den Vorrat  $b_i$  und Ganzzahligkeitsforderung. Diese beiden Relaxationen führen dazu, dass der Wert  $B_1(\tilde{V}, i)$  immer größer oder gleich dem optimalen Wert des Rucksackproblems mit dem Volumen  $\tilde{V}$  und den Gegenständen  $G_i, G_{i+1}, \dots, G_n$  ist. Damit ist  $B_1(\tilde{V}, i)$  eine obere Schranke für den aus dem Volumen  $\tilde{V}$  zu erwartenden maximalen Wert. Die beiden Schranken  $B_2(\tilde{V}, i)$  und  $B_3(\tilde{V}, i)$  sind verbesserte obere Schranken, d.h. es gilt

$$B_1(\tilde{V}, i) \geq B_2(\tilde{V}, i) \geq B_3(\tilde{V}, i).$$

Die Verkleinerung der Schranke wird dabei durch eine weniger starke Relaxation erreicht, indem ein bzw. zwei Gegenstände ( $G_i$  und  $G_{i+1}$ ) maximal oft unter Beachtung der entsprechenden Vorräte und der Ganzzahligkeit benutzt werden. Das jeweils verbleibende Volumen wird dann mit dem nächsten Gegenstand (also  $G_{i+1}$  bzw.  $G_{i+2}$ ) vollständig ausgefüllt (nun wieder ohne Vorrat und Ganzzahligkeit zu beachten). Allerdings werden die Verbesserungen der Schranken durch einen höheren Berechnungsaufwand erkauft.

Ob der höhere Aufwand bei der Schrankenberechnung gerechtfertigt ist oder nicht kann nicht im Voraus gesagt werden. Der Erfolg der Schranken hängt unter anderem von der Größe des Problems (Anzahl der Gegenstände), aber auch von den Daten (z.Bsp. Volumen des Rucksacks) und der sich daraus ergebenden Verzweigungsstruktur ab.

Bei der Berechnung der verbesserten Schranken ist zu beachten, dass  $B_2(\tilde{V}, i)$  nur für  $i = 1, \dots, n-1$  und  $B_3(\tilde{V}, i)$  nur für  $i = 1, \dots, n-2$  definiert sind. Bei der Implementation kann man entsprechende Fallunterscheidungen aber vermeiden, wenn man Hilfsgegenstände  $G_{n+1}$  und  $G_{n+2}$  einführt mit

$$v_{n+1} = v_{n+2} = 1, \quad w_{n+1} = w_{n+2} = 0, \quad b_{n+1} = b_{n+2} = 0.$$

### 3.2 Algorithmus

Ein einfaches Branch & Bound - Verfahren ist in Algorithmus 2 angegeben. Die benutzte rekursive Schreibweise erlaubt eine einfache Notation.

<sup>3</sup>Das ist nicht immer der Fall, oft ist die Schrankenberechnung viel teurer als hier, z.Bsp. kann die Schrankenberechnung die Lösung eines Linearen Optimierungsproblems bedeuten.

---

**Algorithm 2** Branch & Bound für Rucksackprobleme mit oberen Schranken

---

1: **procedure** BB( $i, \tilde{V}, v, w, b, x, z, x^*, z^*$ ) ▷ Exakte Lösung mittels Branch & Bound  
**Eingabe:**  
     $i \in \mathbb{Z}_+$     nächster/aktueller Gegenstand  
     $\tilde{V}$     aktuelles (Rest-) Volumen des Rucksacks     $n \in \mathbb{Z}_+$     Anzahl der Gegenstände  
     $v \in \mathbb{Z}_+^n$     Volumen der Gegenstände     $w \in \mathbb{R}_+^n$     Volumen der Gegenstände  
     $b \in \mathbb{Z}_+^n$     Vorräte der Gegenstände  
**Ein-/Ausgabe:**  
     $x \in \mathbb{Z}_+^n$     aktuelle Lösung  
     $z \in \mathbb{R}$     Wert der aktuellen Lösung  
     $x^* \in \mathbb{Z}_+^n$     bisher beste gefundene Lösung  
     $z^* \in \mathbb{R}$     Wert der bisher besten Lösung  
**Require:**  $\frac{w_1}{v_1} \geq \frac{w_2}{v_2} \geq \dots \geq \frac{w_n}{v_n}$ ,  $B$  sei eine der Schranken  $B_1, B_2$  oder  $B_3$   
2:    **if**  $i > n$  **then**  
3:        Exit  
4:    **end if**  
5:     $a \leftarrow \min \left\{ \left\lfloor \frac{\tilde{V}}{v_i} \right\rfloor, b_i \right\}$  ▷ maximale Anzahl von  $G_i$  in  $\tilde{V}$   
6:    **for**  $k \leftarrow a, a-1, \dots, 1, 0$  **do** ▷ Häufigkeit von  $G_i$  in jedem Schritt verringern  
7:         $x_i = k$  ▷ Gegenstand  $G_i$  k-mal einpacken  
8:         $\bar{z} = z + k \cdot w_i$  ▷ Wert dieser Lösung  
9:        **if**  $\bar{z} > z^*$  **then** ▷ Neue beste Lösung gefunden?  
10:           $z^* \leftarrow \bar{z}, x^* \leftarrow x$  ▷ Neue beste Lösung speichern  
11:        **end if**  
12:         $\bar{V} = \tilde{V} - k \cdot v_i$  ▷ Restliches Volumen  
13:         $b = B(\bar{V}, i+1)$  ▷ Schranke für verbleibendes Volumen berechnen  
14:        **if**  $z^* < z + b$  **then** ▷ Verzweigung verspricht Verbesserung von  $z^*$  ?  
15:          **Call** BB( $i+1, \bar{V}, v, w, b, x, z, x^*, z^*$ ) ▷ **Rekursiver Aufruf**  
16:        **end if**  
17:    **end for**  
18: **end procedure**

---

Der Aufruf dieser rekursiven Routine hat dabei folgendermaßen zu geschehen:

- 1:  $x = 0, x^* = 0$  ▷ Vektoren auf Null setzen
- 2:  $z = 0, z^* = 0$
- 3: **Call** BB( $1, V, v, w, b, x, z, x^*, z^*$ ) ▷ **Rekursiver Aufruf**

## 4 Beispiele

$$V = 26, \quad n = 3, \quad v = (3, 5, 7)^T, \quad w = (6, 8, 10)^T, \quad b = (4, 3, 2)^T$$

|            | $z^*$ | $x_1^*$ | $x_2^*$ | $x_3^*$ | Benutztes<br>Volumen<br>$v^T x^*$ | Anzahl<br>Schranken-<br>berechnung | Anzahl<br>Schleifenkörper-<br>durchläufe | Anzahl<br>rekursiver<br>Aufrufe |
|------------|-------|---------|---------|---------|-----------------------------------|------------------------------------|--|---------------------------------|
| Greedy     | 40    | 4       | 2       | 0       | 22                                |                                    |  |                                 |
| Voll.Enum. | 44    | 4       | 0       | 2       | 26                                | 0                                  | 68                                       | 25                              |
| B&B, $B_1$ | 44    | 4       | 0       | 2       | 26                                | 17                                 | 25                                       | 9                               |
| B&B, $B_2$ | 44    | 4       | 0       | 2       | 26                                | 14                                 | 19                                       | 6                               |
| B&B, $B_3$ | 44    | 4       | 0       | 2       | 26                                | 13                                 | 17                                       | 5                               |

$$V = 276, \quad n = 3, \quad v = (3, 5, 7)^T, \quad w = (6, 8, 10)^T, \quad b = (14, 33, 32)^T$$

|            | $z^*$ | $x_1^*$ | $x_2^*$ | $x_3^*$ | Benutztes<br>Volumen<br>$v^T x^*$ | Anzahl<br>Schranke<br>wirksam | Anzahl<br>Schleifenkörper-<br>durchläufe | Anzahl<br>rekursiver<br>Aufrufe |
|------------|-------|---------|---------|---------|-----------------------------------|-------------------------------|--|---------------------------------|
| Greedy     | 438   | 14      | 33      | 0       | 270                               |                               |  |                                 |
| Voll.Enum. | 444   | 14      | 30      | 12      | 276                               | 0                             | 13150                                    | 526                             |
| B&B, $B_1$ | 444   | 14      | 30      | 12      | 276                               | 507                           | 526                                      | 20                              |
| B&B, $B_2$ | 444   | 14      | 30      | 12      | 276                               | 134                           | 141                                      | 8                               |
| B&B, $B_3$ | 444   | 14      | 30      | 12      | 276                               | 114                           | 119                                      | 6                               |

phone: 030/2093-5820    fax: 030/2093-5859    e-mail: [griewank@math.hu-berlin.de](mailto:griewank@math.hu-berlin.de)    [riehme@math.hu-berlin.de](mailto:riehme@math.hu-berlin.de)  
<http://www.mathematik.hu-berlin.de/~gaggle/MATHINF>