

Kombinatorische Optimierung 1A 1

1)
a+b)

P1: Wenn ihr mit Logging geleskt
habe, lasst es das nächste mal
berner. Das macht es sehr langsam

	s	w	c
	1 1 1 1	$64 + 24 + 16 + 8 = 104$	
	0 1 1 1	$64 + 24 + 16 = 104$	
	0 0 1 1	$64 + 24 = 88$	$40 + 20 = 60$
	0 0 0 1	64	40
s^*	1 1 0 1	$64 + 16 + 8 = 88$	$40 + 20 + 15 = 75$
	0 1 0 1	$64 + 16 = 80$	$40 + 20 = 60$
	1 0 0 1	$64 + 8 = 72$	$40 + 15 = 55$
s^*	1 0 1 1	$64 + 24 + 8 = 96$	$40 + 20 + 15 = 75$
	1 1 1 0	$24 + 16 + 8 = 48$	$20 + 20 + 15 = 55$
	0 1 1 0	$24 + 16 = 40$	$20 + 20 = 40$
	1 0 1 0	$24 + 8 = 32$	$20 + 15 = 35$
	0 0 1 0	24	20
	1 1 0 0	$16 + 8 = 24$	$20 + 15 = 35$
	0 1 0 0	16	20
	1 0 0 0	8	15
	0 0 0 0	0	0

} nicht in S, da

$$\sum_i^4 w_i > 100 \text{ u. } \sum_i^4 v_i > 100$$

a) $\frac{15}{2}$

b) $\frac{3}{2}$

$\Rightarrow \frac{5}{6}$

c) $\frac{3}{2}$

(mit)	1	2	3	4
c_i	15	20	20	40
w_i	8	16	24	64

← ebenfalls zulässig -0.5

Optimal sind die Lösungen $s^* \in S$ mit $c(s^*) \geq c(s)$ für alle $s \in S$

c)

1 hat die höchste Nutzendifferenz: $d_i = c_i / w_i$

i	1	2	3	4
d_i	1.875	1.25	0.85	0.625

D.h. in Objekt 1 sind das geringste Gewicht mit dem höchsten Nutzen pro Gewichtseinheit verknüpft.

Dementsprechend ist die optimale Lösung für das ganzzahlige Rucksackproblem

$s^* \in S$ mit $[1, 1, 0, 0, 0]$, $w(s^*) = 96$ und $c(s^*) = 180$.

✓ 2 Pkt.

963067 Tobias Laske
951206 Franziska Becker
964188 Andrea Krusenbaum

A1	A2	Σ	P1
5,5	5,5	10,5	10,5

2.)

a)

Gegeben: n Objekte mit Gewichten $w_1, \dots, w_n \in \mathbb{N}$ und Nutzwerten $c_1, \dots, c_n \in \mathbb{N}$, Gewichtsbeschr. ~~ab~~ ...

Gesucht: Vektor $(x_1, \dots, x_n) \in \{0,1\}^n$, sodass

$$\sum_{i=1}^n w_i x_i \leq b \quad (\text{Gewichtsbeschränkung}) \text{ und}$$

$$\sum_i c_i x_i \text{ maximal.}$$

Idee:

1. Berechne Nutzendifferenz aller Objekte: $d_i = c_i / w_i, 1 \leq i \leq n$
2. Sortiere Objekte nach d_i fallend.
3. Packe möglichst viele ganze Objekte ($x_i = 1$) in den Rucksack der Reihenfolge der ~~absteigend~~ sortierten Objekte folgend bis das Ganzte Objekt, mit dessen Hinzufügen ~~ab~~ überschritten wird, zerfällt wird ($x_i \leq 1$)

Algorithmus:

for i from 1 to n do:

$d_i \leftarrow c_i / w_i$

$x_i \leftarrow 0$

Aufruf Sortieralgorithmus (nach d_i)

$i \leftarrow 0$

$r \leftarrow b$ // r ist das verfügbare Restvolumen

while $r > 0$ do:

$i++$

if ~~ab~~ $w_i \leq r$:

$x_i \leftarrow 1$

$r \leftarrow r - w_i$

else:

$x_i \leftarrow r / w_i$

$r \leftarrow 0$

Laufzeit?

Beweis Optimalität:

963067 Tobias Loske
951206 Franziska Becker
964188 Andrea Kivsenbaum

Fallunterscheidung:

Fall 1: $\sum w_i \leq b \rightarrow x_1 = \dots = x_n = 1$

→ Alle Objekte werden vollständig verwendet
 ↳ optimal

$$\text{Fall 2: } \sum_{i=1}^n w_i > b \rightarrow x_1 \geq \dots \geq x_n \quad \forall x_i \in [0,1]^n$$

Fall 2.1: $w_A > b$

$$x_1 = \frac{b}{w_1}$$

→ offensichtlich optimal (unter Berücksichtigung der Sortierung π), denn es wird durch $x_{\pi}^* = b/lw_i$ genau der Bruchteil zweier gegeben, der passt.

Fall 2.2: $w_1 \leq b$

5. Fall 1 $x_1 = 1 \rightarrow \text{optimal}$

Dann wird weiter iteriert durch: mit $d_1 \geq \dots \geq d_n$ (*) , bis $\sum w_i \geq b$

Dann trifft Fall 2.1 für w_1 ein:

$$\sum_{i=1}^{n-1} 1 \cdot c_i + \cancel{X_n} \cdot c_n$$

$$\text{mit } x_n = (b - \sum_{i=1}^{n-1} w_i) / w_n.$$

① nach Fall 1 optimal

② nach Fall 2.1 optimal

$$\Rightarrow \sum_{i=1}^n x_i c_i = \sum_{i=1}^n x_i^* c_i \quad (\text{mit } x_i^* \text{ optimal})$$

\therefore !

* Ein bedeutender Schritt des Algorithmus ist die Sortierung nach d_i :

Angenommen, S ist nicht optimal.

→ Es gibt einen optimalen Plan, wobei Sanktionierung nach d; nicht eingehalten wird.

$\rightarrow \exists i, j$ mit $\frac{c_i}{w_i} \geq \frac{c_j}{w_j}$ und j wird vor i betrachtet

→ Sei x_i die Anzahl, die von j mitgenommen wird, entsprechend x_i für i

$$\rightarrow x_j w_j = \underline{w} \frac{\underline{w_i}}{w_i} \geq \frac{\underline{w_i}}{w_j}$$

unklar: Was wird hier gezeigt?
Wie folgt nächster Schritt? -

→ Der Plan wird besser wenn Objekte nach $\frac{c_i}{v_i}$ betrachtet werden

5. ist optimal.

Beweis Zulässigkeit fehlt -2

9) $\frac{3}{5}$

b)

Idee:

1. Berechne Nutzendichte aller Objekte. ($d_i = c_i / w_i$)
2. Sortiere Objekte nach d_i fallend
3. Packe Objekte in den Rückrucksack bis die Grenze b erreicht wird.

✓

Algorithmus:

for i from 1 to n do:

$d_i \leftarrow c_i / w_i$

$sol \leftarrow []$ Lösungsarray

Aufruf Sortieralgorithmus nach d_i

$i \leftarrow 0$
 $r \leftarrow b$

while $r \geq 0$ do:

$i++$

if $w_i \leq r$:

$sol \leftarrow i$ (Füge i dem Array hinzu)

$r \leftarrow r - w_i$

else:

$r \leftarrow 0$ ✓

Gegenbeispiel aus Aufgabe 1:

Da die Objekte nach Nutzendichte sortiert sind wird die

Lösung $s_1 = \{1, 2, 3\}$ zurück gegeben mit $c(s_1) = 55$ und

$w(s_1) = 48$. Die in Nr. 1 ermittelten optimalen Lösungen sind

jedoch $s_2 = \{4, 3, 1\}$ ~~mit $c(s_2)$~~ und $s_3 = \{4, 2, 1\}$

mit $c(s_2) = c(s_3) = 75$ ✓

6) $\frac{3}{2}$

$\Rightarrow \frac{5}{8}$