Technische Universität Braunschweig

Algorithmen und Datenstrukturen 2

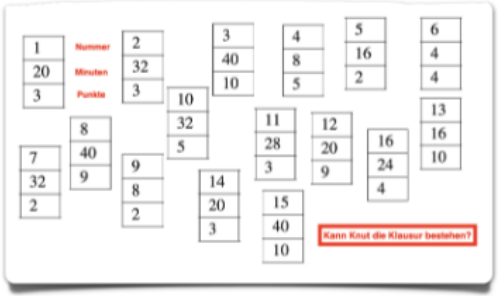
Arne Schmidt

Kapitel 4 – Approximationen

Kapitel 4.1 – Motivation

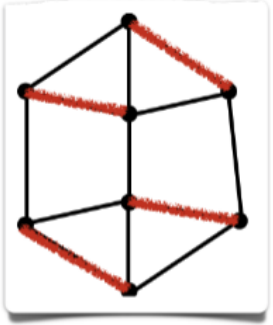
Optimierungsprobleme

0-1 Knapsack



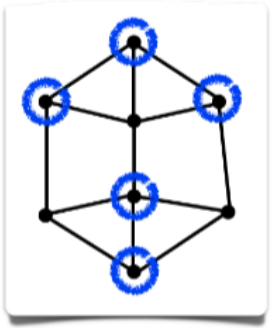
Max

Matching



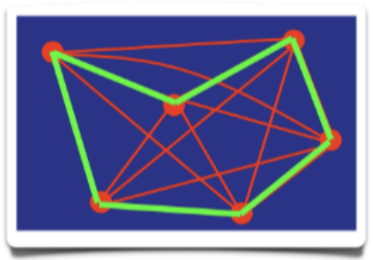
Max

Vertex Cover



Min

Traveling Salesman



Min

Ziel: Finde eine bestmögliche Lösung unter **exponentiell** vielen möglichen!

Laufzeiten

0-1 Knapsack

$O(nZ)$

$O(2^n f(n))$

Gegeben:

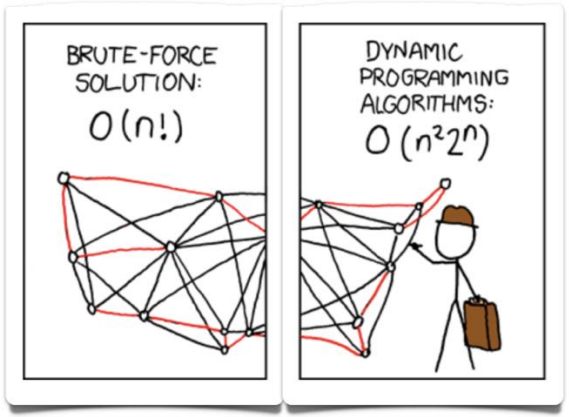
Aufgabe i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Zeit z_i	20	32	40	8	16	4	32	40	8	32	28	20	16	20	40	24
Punkte p_i	3	3	10	5	2	4	2	9	2	5	3	9	10	3	10	4

Zeitschranke: 120
Punktschranke: 44

Gesucht:

Eine Menge $S \subseteq \{1, \ldots, 16\}$ mit $\sum_{i \in S} z_i \leq 120$ und $\sum_{i \in S} p_i \geq 44$,
falls diese überhaupt existiert.

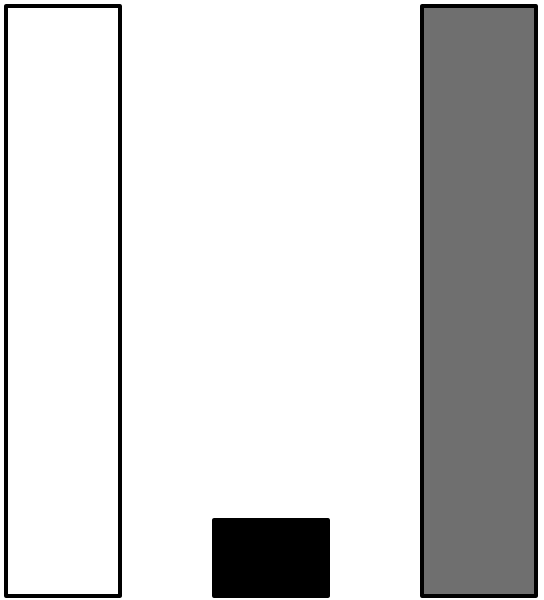
Travelling Salesman



Ziel: Vielleicht nicht die *bestmögliche* Lösung, sondern eine *gute*, aber dafür schneller?!

Kapitel 4.2 – Greedy und Approximation

Beispiel 4.1: Knapsack



$Z = N$ $z_1 = 1$ $z_2 = N$

$n = 2$ $p_1 = 2$ $p_2 = N$

Algorithmus 4.2 GREEDY$_0$

Eingabe: $z_1, ..., z_n, Z, p_1, ..., p_n$

Ausgabe: $S \subseteq \{1, ..., n\}$ und G_0

mit

$$\sum_{i \in S} z_i \leq Z$$

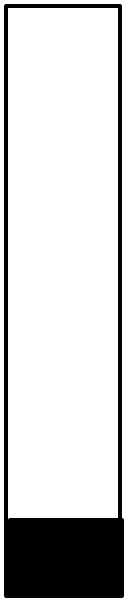
und

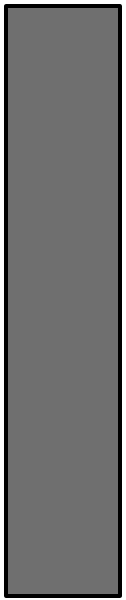
$$G_0 := \sum_{i \in S} p_i = Inklusionsmaximal$$

1: Sortiere {1,...,n} nach $\frac{z_i}{p_i}$ aufsteigend;
Dies ergibt die Permutation $\pi(1), ..., \pi(n)$.
Setze $j = 1$.
2: **while** $(j \leq n)$ **do**
3: **if** $\left(\sum_{i=1}^{j-1} z_{\pi(i)} x_{\pi(i)} + z_{\pi(j)} \leq Z\right)$ **then**
4: $x_{\pi(j)} := 1$
5: $j := j + 1$
6: **return**

Beispiel 4.1: Knapsack

Greedy: 2





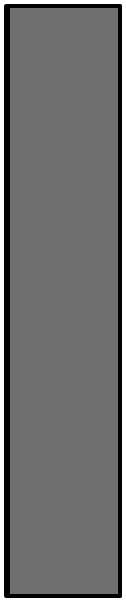
$Z = N$ $z_1 = 1$ $z_2 = N$

$n = 2$ $p_1 = 2$ $p_2 = N$

Greedy packt Objekt 1!

Man erkennt aber, dass das weit entfernt vom Optimum ist!

Beispiel 4.1: Knapsack



Greedy: 2

OPT: N

Greedy/OPT → 0

$Z = N$ $z_1 = 1$ $z_2 = N$

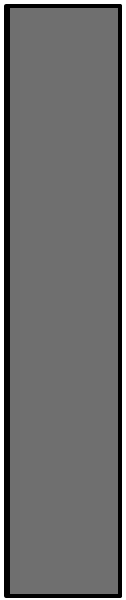
$n = 2$ $p_1 = 2$ $p_2 = N$

Greedy packt Objekt 1!

Man erkennt aber, dass das weit entfernt vom Optimum ist!

Greedy kann also beliebig schlecht sein!

Triviale Verbesserung



Greedy: 2

Greedy': N

OPT: N

Greedy'/OPT → 1

$Z = N$ $z_1 = 1$ $z_2 = N$

$n = 2$ $p_1 = 2$ $p_2 = N$

Was passiert, wenn wir einfach eine zweite Lösung betrachten…

Welches nur das beste Objekt enthält?

In diesem Beispiel wären wir optimal!

Triviale Verbesserung

Algorithmus 4.3 Greedy-Approximation

Eingabe: $z_1, ..., z_n, Z, p_1, ..., p_n$

Ausgabe: $S \subseteq \{1, ..., n\}$ mit $\sum_{i \in S} z_i \leq Z$ und Wert $G'_0 := \sum_{i \in S} p_i$

1: $G'_0 := max\{G_0, max\{p_i \mid z_i \leq Z, i \in \{1, ..., n\}\}\}$

2: **return**

Nimm das Maximum von…

Greedy0

Wertvollstem Objekt

Wie gut ist Algorithmus 4.3?

Satz 4.4:
Algorithmus 4.3 berechnet eine Lösung mit Wert G_0'. Im Vergleich mit dem Optimalwert OPT gilt:

$$G_0' \geq \frac{1}{2}\,\text{OPT}$$

Beweis:
Nach Konstruktion gilt:

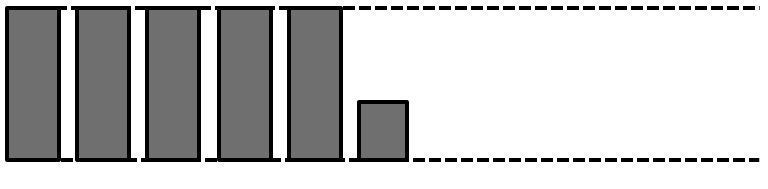
- $G_0' \geq G_0$
- $G_0' \geq \max p_i =: p^*$

Außerdem gilt

$$G_0 + p^* \geq \text{FractionKP} \geq \text{OPT}$$

Daher ist

$$2G_0' \geq G_0 + p^* \geq \text{OPT} \Longrightarrow G_0' \geq \frac{1}{2}\,\text{OPT}$$



Objekt j passt als erstes nicht mehr.
→ $p_j \leq p^*$

Also ist $G_0 + p^* \geq G_0 + p_j \geq \text{FractionalKP}$

Kapitel 4.3 – Approximationsalgorithmen

Definitionen

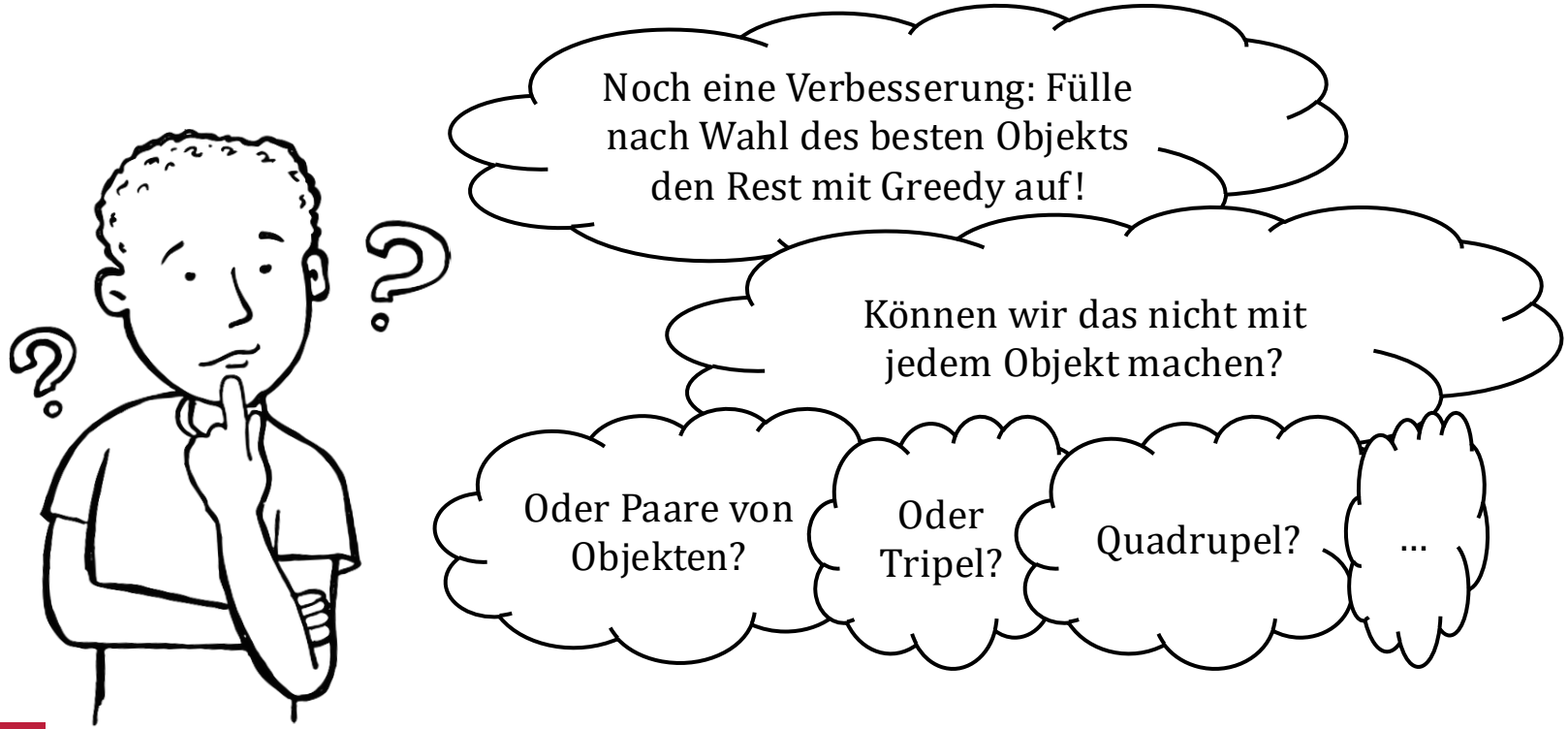
Definition 4.5 (Approximationsalgorithmus):

1. Für ein Maximierungsproblem MAX ist ein Algorithmus ALG ein c-Approximationsalgorithmus für MAX, wenn für jede Instanz I von MAX
   a) ALG in polynomieller Zeit in der Größe von I eine zulässige Lösung mit Wert $\text{ALG}(I)$ liefert und
   b) Für den Vergleich mit dem zugehörigen Optimalwert $\text{OPT}(I)$ gilt: $\text{ALG}(I) \geq c \cdot \text{OPT}(I)$ mit $c \leq 1$
2. Analog zu einem Minimierungsproblem MIN:
   a) ALG liefert in polynomieller Zeit in der Größe von I eine zulässige Lösung mit Wert $\text{ALG}(I)$ und
   b) Für den Vergleich mit dem zugehörigen Optimalwert $\text{OPT}(I)$ gilt: $\text{ALG}(I) \leq c \cdot \text{OPT}(I)$ mit $c \geq 1$

Frage: Wie gut kann man Maximum Knapsack approximieren?

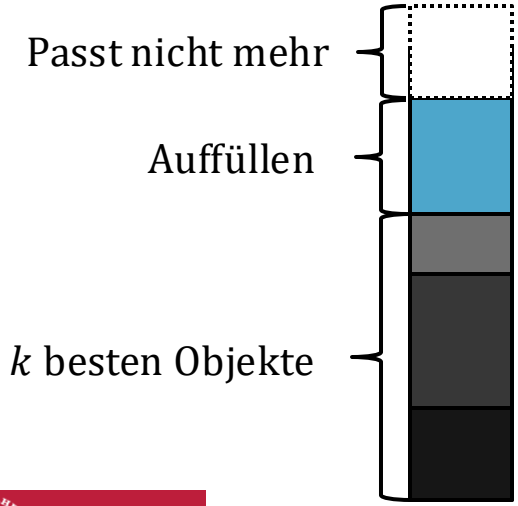
Kapitel 4.4 – Bessere Approximation

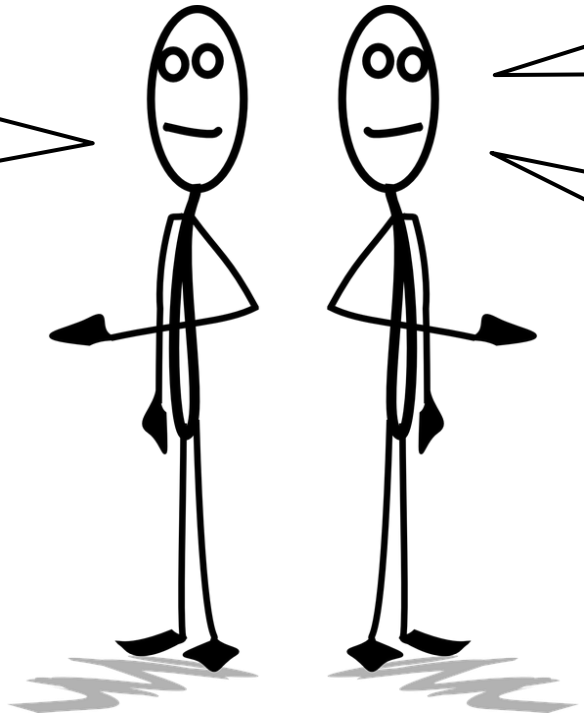
Idee


Noch eine Verbesserung: Fülle nach Wahl des besten Objekts den Rest mit Greedy auf!
Können wir das nicht mit jedem Objekt machen?
Oder Paare von Objekten?
Oder Tripel?
Quadrupel?
...

Teilmengen enumerieren

Benutze die k besten Objekt und fülle sie auf. Der Fehler ist durch höchstens 1 Objekt beschränkt!





Enumeriere alle Teilmengen der Größe höchstens $k \in \mathbb{N}$.

Das sind $O(n^k)$ viele.
Für fixes k ist das polynomiell!

Greedy-k

Algorithmus 4.6 GREEDY$_k$

Eingabe: $z_1, ..., z_n, Z, p_1, ..., p_n$ [Parameter k fixiert]

Ausgabe: $S \subseteq \{1, ..., n\}$ mit $\sum_{i \in S} z_i \leq Z$ und Wert $G_k := \sum_{i \in S} p_i$

1: $G_k := 0$, $S := \emptyset$
2: **for all** $\bar{S} \subseteq \{1, ..., n\}$ mit $|\bar{S}| \leq k$ **do**
3: **if** $(\sum_{i \in \bar{S}} z_i \leq Z)$ **then**
4: $G_k := max\{G_k, \sum_{i \in \bar{S}} p_i + \text{GREEDY}_0(\{z_i | i \notin \bar{S}\}, Z - \sum_{i \in \bar{S}} z_i, \{p_i | i \notin \bar{S}\})\}$;
5: Update S;
6: **return** G_k, S

Gütegarantie

Satz 4.7:
$\textsc{Greedy}_k$ ist $\left(1 - \frac{1}{k+1}\right)$-Approximationsalgorithmus.

Zu zeigen:

a) Die Laufzeit ist polynomiell.

b) Für jede Instanz gilt $\textsc{Greedy}_k(I) \geq \left(1 - \frac{1}{k+1}\right) \cdot \textsc{Opt}(I)$

Zusatzbemerkungen

Der Approximationsfaktor von GREEDY_k ist scharf, d.h. es gibt Instanzen, für die das Verhältnis Alg/Opt den Faktor annimmt.

Tauscht man G_0 mit G_0', erhält man einen Approximationsalgorithmus mit Gütegarantie $\left(1 - \frac{1}{k+2}\right)$.

Nächstes Kapitel