Technische Universität Braunschweig

Algorithmen und Datenstrukturen II

2. Vorlesung

Linda Kleist, 24.04.2019

Lösen von Fractional Knapsack - Wiederholung

Problem 3 (Fractional Knapsack)

Teilpunkte

Gegeben:
- Objekte $\{O_1, ..., O_n\}$, je mit Größe z_i und Gewinn p_i
- Größenschranke Z

Gesucht: Ein Wert $x_i \in [0, 1]$ für jedes Objekt O_i, so dass

$$\sum_{i=1}^{n} x_i \cdot z_i \leq Z \text{ und}$$

$$\sum_{i=1}^{n} x_i \cdot p_i \text{ maximal.}$$

Lösen von Fractional Knapsack - Wiederholung

Algorithmus 1 (Greedy-Algorithmus für FRACTIONAL KNAPSACK)

Eingabe: $z_1, z_2, \ldots, z_n, Z, p_1, p_2, \ldots, p_n$

Ausgabe: $x_1, x_2, \ldots, x_n \in [0,1]$ mit $\sum_{i=1}^{n} x_i \cdot z_i \leq Z$ und $\sum_{i=1}^{n} x_i \cdot p_i$ maximal.

1: sortiere $\{1, \ldots, n\}$ nach $\frac{z_i}{p_i}$ aufsteigend $\rightarrow$ Permutation $\pi(1), \ldots, \pi(n)$
2: **for** $k = 1$ to n **do** ▷ Betrachte jedes Element
3: **if** $(\sum_{i=1}^{k} z_{\pi(i)} \leq Z)$ **then** ▷ falls es hineinpasst
4: $x_{\pi(k)} := 1$ ▷ wähle es ganz,
5: **else**
6: $x_{\pi(k)} := \dfrac{Z - \sum_{i=1}^{k-1} x_{\pi(i)} z_{\pi(i)}}{z_{\pi(k)}}$ ▷ ansonsten anteilig.
7: **return** $x_1, x_2, \ldots, x_n$ ▷ Gib Lösung zurück.

Lösen von Fractional Knapsack

Satz 1

Algorithmus 1 liefert eine optimale Lösung für Fractional Knapsack. Die Laufzeit ist in $O(n \cdot \log n)$.

Example

Tabelle, sortiert nach wertvollen Aufgaben:

i	6	4	13	12	9	3	15	8	16	10	1	14	5	11	2	7
z_i	4	8	16	20	8	40	40	40	24	32	20	20	16	28	32	32
p_i	4	5	10	9	2	10	10	9	4	5	3	3	2	3	3	2
$z_i/p_i \approx$	1.0	1.6	1.6	2.2	4.0	4.0	4.0	4.4	6	6.4	6.7	6.7	8.0	9.3	10.7	16.0

Beweis.

Tafel... □

Greedy-Algorithmen

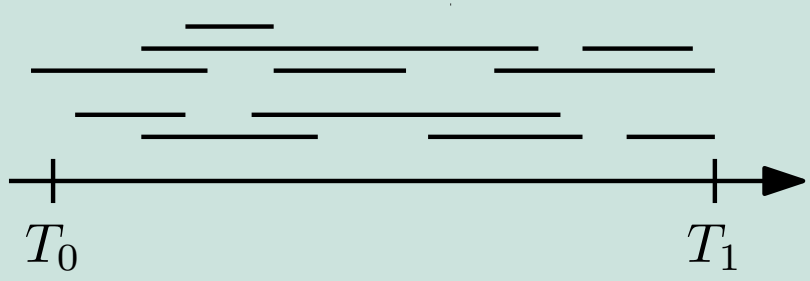
Greedy-Algorithmus

Ein Greedy-Algorithmus wählt in jedem Schritt den besten Kandidaten (bzgl. einer Bewertungsfunktion).

Beispiel: Hörsaal-Belegung

Gegeben:
- Zeitspanne (T_0, T_1)
- Veranstaltungen mit Start- und Endzeiten (s_i, e_i), $i \in \{1, \ldots, n\}$

Gesucht: Auswahl möglichst vieler Veranstaltungen mit disjunkten Zeitspannen



Frage: Was ist eine gute Bewertungsfunktion?

Greedy-Algorithmen

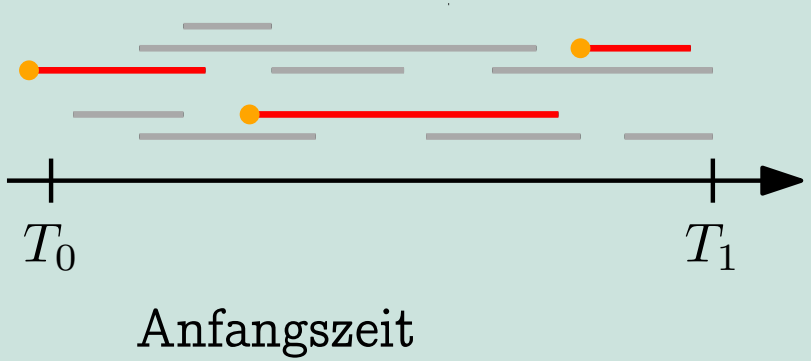
Greedy-Algorithmus

Ein Greedy-Algorithmus wählt in jedem Schritt den besten Kandidaten (bzgl. einer Bewertungsfunktion).

Beispiel: Hörsaal-Belegung

Gegeben:
- Zeitspanne (T_0, T_1)
- Veranstaltungen mit Start- und Endzeiten (s_i, e_i), $i \in \{1, \ldots, n\}$

Gesucht: Auswahl möglichst vieler Veranstaltungen mit disjunkten Zeitspannen



Frage: Was ist eine gute Bewertungsfunktion?

Greedy-Algorithmen

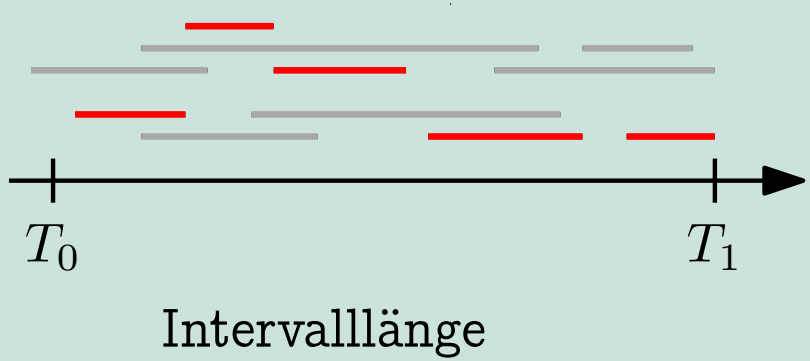
Greedy-Algorithmus

Ein Greedy-Algorithmus wählt in jedem Schritt den besten Kandidaten (bzgl. einer Bewertungsfunktion).

Beispiel: Hörsaal-Belegung

Gegeben:
- Zeitspanne (T_0, T_1)
- Veranstaltungen mit Start- und Endzeiten (s_i, e_i), $i \in \{1, \ldots, n\}$

Gesucht: Auswahl möglichst vieler Veranstaltungen mit disjunkten Zeitspannen



Frage: Was ist eine gute Bewertungsfunktion?

Greedy-Algorithmen

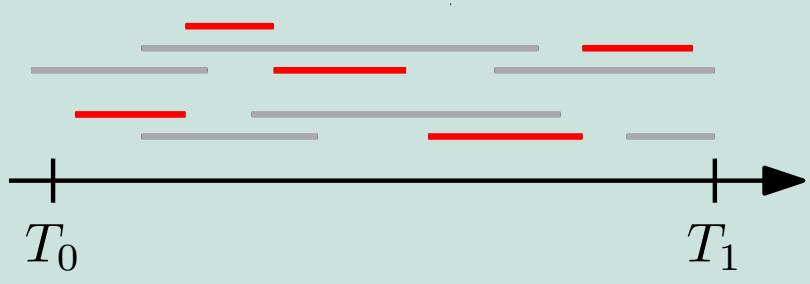
Greedy-Algorithmus

Ein Greedy-Algorithmus wählt in jedem Schritt den besten Kandidaten (bzgl. einer Bewertungsfunktion).

Beispiel: Hörsaal-Belegung

Gegeben:
- Zeitspanne (T_0, T_1)
- Veranstaltungen mit Start- und Endzeiten (s_i, e_i), $i \in \{1, \ldots, n\}$

Gesucht: Auswahl möglichst vieler Veranstaltungen mit disjunkten Zeitspannen



Überlappungsgrad

Frage: Was ist eine gute Bewertungsfunktion?

Greedy-Algorithmen

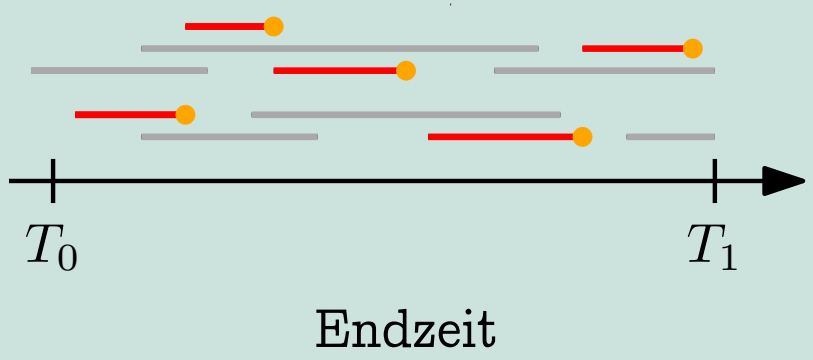
Greedy-Algorithmus

Ein Greedy-Algorithmus wählt in jedem Schritt den besten Kandidaten (bzgl. einer Bewertungsfunktion).

Beispiel: Hörsaal-Belegung

Gegeben:
- Zeitspanne (T_0, T_1)
- Veranstaltungen mit Start- und Endzeiten (s_i, e_i), $i \in \{1, \ldots, n\}$

Gesucht: Auswahl möglichst vieler Veranstaltungen mit disjunkten Zeitspannen



Frage: Was ist eine gute Bewertungsfunktion?

Lösen der Hörsaal-Belegung

Algorithmus 2 (Greedy-Algorithmus für Hörsaal-Belegung)

Eingabe: $(s_1, e_1), (s_2, e_2), \ldots, (s_n, e_n)$

Ausgabe: $S \subseteq \{1, \ldots, n\}$, sodass für alle $i, j \in S$ sich (s_i, e_i) und (s_j, e_j) nicht überschneiden und $|S|$ maximal

---

1: sortiere $\{1, ..., n\}$ nach e_i aufsteigend $\rightarrow$ Permutation $\pi(1), ..., \pi(n)$
2: $S := \{\pi(1)\}$
3: $e := e_{\pi(1)}$
4: **for** $k = 2$ to n **do**
5: **if** $(s_{\pi(i)} \geq e)$ **then**
6: $S := S \cup \{\pi(i)\}$
7: $e := e_{\pi(i)}$
8: **return** S

Lösen der Hörsaal-Belegung

Satz 2

Algorithmus 2 liefert eine optimale Lösung für Hörsaal-Belegung. Die Laufzeit ist in $O(n \cdot \log n)$.

Beweis.

Übung. □