Technische Universität Braunschweig

Algorithmen und Datenstrukturen II

5. Vorlesung

Linda Kleist, 22.05.2019

Erinnerung

Problem 6 (Partition)

Gegeben: - Objekte $\{O_1, ..., O_n\}$, je mit Größe z_i

Gesucht: Eine Menge $S \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in S} z_i = \sum_{i \notin S} z_i$.

Beispiel 2

Hat die Zahlenreihe $(7, 13, 17, 20, 29, 31, 31, 35, 57)$ mit Summe 240 eine Teilmenge mit Summe 120?

Antwort: Nein! *...Wie kann man das beweisen?...*

strukturiertes Ausprobieren

- Aufbauen der Lösung auf Teilproblemen → Dynamic Programming
- Schneide Teilbäume ab, wenn möglich → Branch & Bound

Verallgemeinerung für Maximum Knapsack

Problem 2 (Maximum Knapsack)

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

Gesucht: Eine Menge $S \subseteq \{1, ..., n\}$ mit $\sum_{i \in S} z_i \leq Z$ und

$\sum_{i \in S} p_i$ maximal.

Beispiel

Gegeben sei $Z = 15$ und fünf Objekte mit den Werten:

i	1	2	3	4	5
z_i	5	4	12	2	3
p_i	3	4	10	5	2

Verallgemeinerung für Maximum Knapsack

Problem 2 (Maximum Knapsack)

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

Gesucht: Eine Menge $S \subseteq \{1, ..., n\}$ mit $\sum_{i \in S} z_i \leq Z$ und

$\sum_{i \in S} p_i$ maximal.

Alternative Formulierung: Gesucht ist ein Vektor $(x_1, \ldots, x_n) \in \{0, 1\}^n$ mit

$$\sum_{i=1}^{n} x_i z_i \leq Z \text{ so, dass } \sum_{i=1}^{n} x_i p_i \text{ maximal.}$$

("Übersetzungsvorschrift": $x_i = 1 \iff i \in S$)

Verallgemeinerung für Maximum Knapsack

Problem 2 (Maximum Knapsack)

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

Gesucht: Eine Menge $S \subseteq \{1, ..., n\}$ mit $\sum_{i \in S} z_i \leq Z$ und

$\sum_{i \in S} p_i$ maximal.

Eine Maximum Knapsack-Instanz $I := (z_1, ..., z_n, Z, p_1, ..., p_n)$, bei der die ersten $\ell - 1$ Variablen auf $b_1, \ldots, b_{\ell-1}$ fixiert sind, hat den Lösungswert $\text{OPT}(I, b_1 \ldots, b_{\ell-1}) :=$

$$\max \left\{ \sum_{j=1}^{\ell-1} b_j p_j + \sum_{j=\ell}^{n} x_j p_j \,\middle|\, \sum_{j=1}^{\ell-1} b_j z_j + \sum_{j=\ell}^{n} x_j z_j \leq Z, x_j \in \{0, 1\} \right\}$$

Algorithmus 5 BRANCH-AND-BOUND als Unterroutine

Eingabe: $I := (z_1, ..., z_n, Z, p_1, ..., p_n)$, ▷ Instanz
P, ▷ aktuell bester Lösungswert
ℓ, ▷ Verzweigungsindex
$b_1, \ldots, b_{\ell-1}$ ▷ Werte fixierter Binärvariablen
Ausgabe: $\max\{P, \text{OPT}(I, b_1, ..., b_{\ell-1})\}$ ▷ bester bekannter Lösungswert

---

1: **procedure** BRANCH-AND-BOUND$(I, P, \ell, b_1, \ldots, b_{\ell-1})$
2: **if** $(\sum_{j=1}^{\ell-1} b_j z_j > Z)$ **then return** P ▷ unzulässig
3: Compute $L := LB(I, b_1, \ldots, b_{\ell-1})$ ▷ untere Schranke berechnen
4: **if** $L > P$ **then** $P := L$ ▷ Lösungswert verbessert
5: **if** $(\ell > n)$ **then return** P ▷ Blatt im Baum erreicht
6: Compute $U := UB(I, b_1, \ldots, b_{\ell-1})$ ▷ obere Schranke berechnen
7: **if** $(U > P)$ **then**
8: $b_\ell := 0$; $P :=$ BRANCH-AND-BOUND$(I, P, \ell + 1, b_1, \ldots, b_\ell)$;
9: $b_\ell := 1$; $P :=$ BRANCH-AND-BOUND$(I, P, \ell + 1, b_1, \ldots, b_\ell)$;
10: **return** P

Berechnung der Schranken

Sei $I := (z_1, ..., z_n, Z, p_1, ..., p_n)$ eine Instanz von Maximum Knapsack und $b_1, ..., b_{\ell-1}$ Werte bereits fixierter Variablen x_i für $i = 1, \ldots, \ell - 1$.

- Eine untere Schranke $LB(I, b_1, ..., b_{\ell-1})$ ist jeder Wert aus

$$\left\{ \sum_{j=1}^{\ell-1} b_j p_j + \sum_{j=\ell}^{n} x_j p_j \;\middle|\; \sum_{j=1}^{\ell-1} b_j z_j + \sum_{j=\ell}^{n} x_j z_j \leq Z, x_j \in \{0, 1\} \right\}$$

  - Berechnung mit ganzzahligem Greedy-Algorithmus Greedy_0

- Eine obere Schranke $UB(I, b_1, \ldots, b_{\ell-1})$ ist

$$\max \left\{ \sum_{j=1}^{\ell-1} b_j p_j + \sum_{j=\ell}^{n} x_j p_j \;\middle|\; \sum_{j=1}^{\ell-1} b_j z_j + \sum_{j=\ell}^{n} x_j z_j \leq Z, x_j \in [0, 1] \right\}$$

  - Berechnung mit Greedy-Algorithmus für Fractional Knapsack

Korrektheit und Laufzeit

Satz 5

Algorithmus 5 berechnet eine optimale Lösung für Maximum Knapsack in einer Laufzeit $O(2^n f(n))$, wobei $f(n)$ die Laufzeit zur Berechnung der Schranken ist.

Beweis.

Tafel... □