

1.5 Approximation

Wie gut ist Greedy? (Gauzzahlige Variante von Alg 1.4, $\epsilon = \epsilon_0$)

Schon in Übung gesehen: beliebig schlecht!

Beispiel 1.21

Betrachte: $n=2$, $z_1=1$, $p_1=1+\epsilon$
 $z_2=N$, $p_2=N$
 $z=N$

Dann liefert Greedy eine Lösung von Nutzen $1+\epsilon$ (nur Objekt 1),
optimal ist aber Nutzen N (nur Objekt 2).

Für $N \gg 1+\epsilon$ ist das beliebig weit vom Optimum!

Relevante Größe ist dabei nicht die absolute Abweichung,
sondern die relative: Wieviele Prozent des Optimums können
wir erreichen?

ALGORITHMUS 1.22 (Greedy₀)

Eingabe : $z_1, \dots, z_n, Z, p_1, \dots, p_n > 0$

Ausgabe : Eine Lösung $S \subseteq \{1, \dots, n\}$ mit $\sum_{i \in S} z_i \leq Z$
und Lösungswert $G_0 := \sum_{i \in S} p_i$

① Sortiere $\{1, \dots, n\}$ nach $\frac{z_i}{p_i}$ aufsteigend;
dies ergibt Permutation $\pi(1), \dots, \pi(n)$.
Setze $j := 1$

② WHILE $(\sum_{i=1}^j z_{\pi(i)} \leq Z)$ DO {
 $x_{\pi(j)} := 1$;
 $j := j + 1$;

③ RETURN

(Vergleiche Algorithmus 1.4 ! $\triangle x_j \rightarrow x_{\pi(j)} \triangle$)

ALGORITHMUS 1.23

Eingabe : $z_1, \dots, z_n, Z, p_1, \dots, p_n > 0$ (mit $z_i \leq Z$)

Ausgabe : Eine Lösung $S \subseteq \{1, \dots, n\}$ mit $\sum_{i \in S} z_i \leq Z$

① Berechne G_0 mit Algorithmus 1.22

② Return $\tilde{G}_0 := \max \{ G_0, \max p_i \}$ und zugehöriges S

SATZ 1.24

Algorithmus 1.23 berechnet eine Lösung mit Wert \tilde{G}_0 , für den im Vergleich zum bestmöglichen Lösungswert gilt

$$\tilde{G}_0 \geq \frac{1}{2} OPT.$$

Beweis:

Nach Konstruktion ist $\tilde{G}_0 \geq G_0$
und $\tilde{G}_0 \geq \max P_i =: p^*$

Außerdem ist (vgl. Algorithmus 1.4)

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

Also ist $2\tilde{G}_0 \geq G_0 + p^* \geq OPT$, d.h. $\tilde{G}_0 \geq \frac{1}{2} OPT$.

Das motiviert die folgende

DEFINITION 1.25

- (1) Für ein Maximierungsproblem MAX ist ein Algorithmus ALG ein c-Approximationsalgorithmus, wenn für jede Instanz I von MAX
 - (i) ALG in polynomieller Zeit in der Größe der Instanz eine Lösung mit Wert ALG(I) liefert.
 - (ii) Für den Vergleich mit dem zugehörigen Optimalwert OPT(I) gilt

$$ALG(I) \geq c \cdot OPT(I)$$

(2) Entsprechend verlangt man für ein Minimierungsproblem wiederum: (39)

(i) polynomielle Laufzeit

(ii) $ALG(I) \leq c \cdot OPT(I)$

Merke: Maximieren $\rightarrow c \leq 1$, je größer c , um so besser!
Minimieren $\rightarrow c \geq 1$, je kleiner c , um so besser!

Wie gut kann man KNAPSACK approximieren?

ALGORITHMUS 1.26 (Greedy_k)

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

Ausgabe: Eine Lösung $S \subseteq \{1, \dots, n\}$ mit $\sum_{i \in S} z_i \leq Z$
und Lösungswert $G_k := \sum_{i \in S} p_i$

① $G_k := 0, S := \emptyset$

② Für alle $\bar{S} \subseteq \{1, \dots, n\}$ mit $|\bar{S}| \leq k$ DO {

②.1 IF $\sum_{i \in \bar{S}} z_i \leq Z$ DO {

②.1.1 $G_k := \max \left\{ G_k, \sum_{i \in \bar{S}} p_i + \text{GREEDY}_0 \left(\{z_i | i \in \bar{S}\}, Z - \sum_{i \in \bar{S}} z_i, \{p_i | i \notin \bar{S}\} \right) \right\}$

②.1.2 Update S

}

}

③ RETURN G_k, S

Satz 1.27

GREEDY_k ist ein $(1 - \frac{1}{k+1})$ -Approximationsalgorithmus.

Beweis:

(1) Man sieht leicht, dass die Laufzeit höchstens $O(n^{k+1})$ ist, für festes k also polynomiell.

(2) Sei OPT ein optimaler Lösungswert, erzielt mit einer Teilmenge $S^* \subseteq \{1, \dots, n\}$. Wir unterscheiden:

(A) $|S^*| \leq k$: Dann testet Greedy_k S^* als ein \bar{S} , findet also diese Lösung.

(B) $|S^*| > k$:

Seien i_1, \dots, i_k die k Objekte in S^* mit größten Nutzen. Diese Teilmenge wird von GREEDY_k als \bar{S} getestet und in (2.1) per Greedy erweitert.

Seien i_{k+1}, \dots, i_{k+q} die dabei hinzugenommenen Objekte und $i_{k+(q+1)}$ das erste Objekt, dass nicht mehr passt.

Dann wissen wir, dass

$$\sum_{j=1}^k p_{i_j} + \sum_{j=1}^q p_{i_{(k+j)}} + p_{i_{(k+q+1)}} > \text{OPT} \quad (*)$$

und

$$\sum_{j=1}^k p_{i_j} + \sum_{j=1}^q p_{i_{(k+j)}} \leq G_k \quad (**)$$

Da $P_{i_{(k+l+1)}} \leq P_{i_j}$ für alle $j = 1, \dots, k$, (4)

ist $P_{i_{(k+l+1)}} \leq \frac{1}{k} G_k$ (***)

Damit ist

$$\begin{aligned} (*) \quad \text{OPT} &\leq \underbrace{\sum_{j=1}^k P_{i_j}}_{(**)} + \sum_{j=1}^l P_{i_{(k+j)}} + \underbrace{P_{i_{(k+l+1)}}}_{(***)} \\ &\leq G_k + \frac{1}{k} G_k \\ &= \left(\frac{k+1}{k}\right) G_k. \end{aligned}$$

Also bekommen wir $G_k \geq \left(\frac{k}{k+1}\right) \text{OPT}$
 $= \left(1 - \frac{1}{k+1}\right) \text{OPT}.$

□

Man kann zeigen:

- Für Greedy_k ist der Faktor $\left(1 - \frac{1}{k+1}\right)$ bestmöglich, d.h. es gibt Instanzen...
- Wenn man G_0 durch \tilde{G}_0 ersetzt, bekommt man einen Faktor $\left(1 - \frac{1}{k+2}\right)$.

Man sieht:

Für jedes feste $\varepsilon > 0$ gibt es einen polynomiellen Algorithmus für KNAPSACK, der eine $(1-\varepsilon)$ -Approximation liefert.

Das motiviert die folgende

DEFINITION 1.28

Ein Polynomielles Approximationsschema (PTAS, für Polynomial-Time Approximation Scheme) für ein Optimierungsproblem ist eine Familie von Algorithmen, die für jedes beliebige aber feste $\epsilon > 0$ einen $(1-\epsilon)$ -Approximationsalgorithmus (bzw. $(1+\epsilon)$ -Approximationsalgorithmus) liefert.

Also:

KOROLLAR 1.29

Die Familie $\{GREEDY_k \mid k \in \mathbb{N}\}$ ist ein polynomielles Approximationsschema für KNAPSACK.

BEWEIS:

Wähle k groß genug, dass $\frac{1}{k+1} \leq \epsilon$ ist.