Satz 1.27

Greedy_k ist ein $\left(1 - \frac{1}{k+1}\right)$-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 $i_{k+1}, \dots, i_{k+\ell}$ die dabei hinzugenommenen Objekte und $i_{k+(\ell+1)}$ das erste Objekt, dass nicht mehr passt.

Dann wissen wir, dass

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

und

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

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

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

Damit ist

(*) $$OPT \leq \underbrace{\sum_{j=1}^{k} p_{i_j} + \sum_{j=1}^{l} p_{i_{(k+j)}}}_{(**)} + \underbrace{p_{i_{(k+l+1)}}}_{(***)}$$

$$\leq G_k + \frac{1}{k} G_k$$

$$= \left(\frac{k+1}{k}\right) G_k .$$

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

$$= \left(1 - \frac{1}{k+1}\right) OPT .$$

□

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 über feste $\varepsilon > 0$ einen $(1-\varepsilon)$-Approximationsalgorithmus (bzw. $(1+\varepsilon)$-Approximationsalgorithmus) liefert.

Also:

KOROLLAR 1.29

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

BEWEIS:

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

1.6 KOMPLEXITÄT

1.6.1 Einstieg

Wir haben bislang eine ganze Reihe von Algorithmen kennengelernt, die KNAPSACK unterschiedlich angehen:

(A) Heuristisch: Einfache „Probiermethoden", die oft ganz ordentliche Lösungen liefern – manchmal sogar optimale (Beispiel: GREEDY!)

(B) Exakt: Algorithmen, die immer optimale Lösungen liefern, aber manchmal sehr lange dafür brauchen (Beispiele: - Dynamic Programming
- Branch-and-Bound)

(C) Approximierend: Algorithmen, die in polynomieller Zeit Lösungen liefern, die nicht unbedingt optimal sind, aber zumindest mit einer „Gütegarantie".

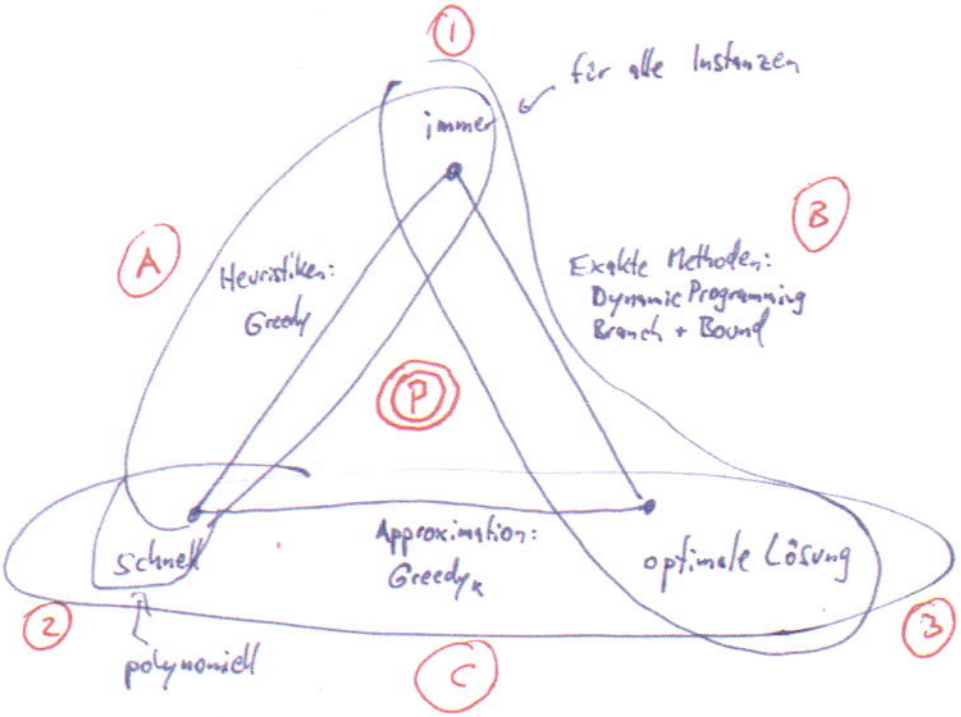
Geht das noch besser? Können wir einen „perfekten" Algorithmus finden, der

(1) immer
(2) in polynomieller Zeit
(3) eine optimale Lösung

} Klasse P

berechnet?

Bild:



Geht das noch besser?

Gibt es einen Algorithmus, der

(1) immer für alle Instanzen
(2) in polynomieller Zeit
(3) eine optimale Lösung

berechnet?

Definition 1.30 (Klasse P)

Ein algorithmisches (oder logisches) Problem gehört zur Klasse P, wenn dafür ein „perfekter" Algorithmus existiert, der

(1) für jede Instanz
(2) in polynomieller Zeit
(3) eine optimale (oder korrekte) Lösung findet.

Schwächer; d.h. potentiell größere Klasse:

Definition 1.31 (Klasse NP)

Ein algorithmisches (oder logisches) Problem gehört zur Klasse NP, wenn sich die Existenz einer Lösung in polynomieller Zeit Nachprüfen lässt.

Beobachtung 1.32

Klar: SUBSET SUM $\in$ NP
Unklar: SUBSET SUM $\in$ P

Lösungswert mindestens so groß wie eine Schranke!

Klar: KNAPSACK$_{\geq}$ $\in$ NP
Unklar: KNAPSACK$_{\geq}$ $\in$ P

Problem 1.33

$P \neq NP$?