The Knapsackproblem

Ferienakademie im Sarntal — Kurs 1
Moderne Suchmethoden der Informatik

Florian Pawlik

Fakultät für Informatik
TU München

23. September 2014

Inhalt

Notationen

"Knapsackproblem" (KP)

- Was ist das Rucksackproblem?

Notationen

"Knapsackproblem" (KP)

- Was ist das Rucksackproblem?

"simple Knapsackproblem" (SKP)

- Was ist das SKP?

Begrifflichkeiten

- Was ist PTAS?

Begrifflichkeiten

- Was ist PTAS?
- Was ist FPTAS?

greedy Algorithmus für SKP

Algorithmus 1

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$.

greedy Algorithmus für SKP

Algorithmus 1

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$.

Schritt 1: Sortiere w_1 bis w_n nach Größe, sodass $w_1 \geq w_2 \geq ... \geq w_n$ gilt.

greedy Algorithmus für SKP

Algorithmus 1

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$.

Schritt 1: Sortiere w_1 bis w_n nach Größe, sodass $w_1 \geq w_2 \geq ... \geq w_n$ gilt.

Schritt 2: $T := \emptyset$; $cost_T := 0$

greedy Algorithmus für SKP

Algorithmus 1

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$.

Schritt 1: Sortiere w_1 bis w_n nach Größe, sodass $w_1 \geq w_2 \geq ... \geq w_n$ gilt.

Schritt 2: $T := \emptyset$; $cost_T := 0$

Schritt 3: **for** i=1 **to** n **do**

if $cost_T + w_i < b$ **then**

do begin $T := T \cup \{i\}$;

$cost_T := cost_T + w_i$;

end

greedy Algorithmus für SKP

Algorithmus 1

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$.

Schritt 1: Sortiere w_1 bis w_n nach Größe, sodass $w_1 \geq w_2 \geq ... \geq w_n$ gilt.

Schritt 2: $T := \emptyset$; $cost_T := 0$

Schritt 3: **for** i=1 **to** n **do**

```
    if cost_T + w_i < b then
        do begin T := T ∪ {i};
                 cost_T := cost_T + w_i;
        end
```

Output: T

Güte des Approximationsalgorithmus

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j + 1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j+1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Fall 1: $j = 1$: $w_1 + w_2 > b$.

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j+1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Fall 1: $j = 1$: $w_1 + w_2 > b$.
$\Rightarrow cost_T \geq w_1 > \frac{b}{2}$.

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j + 1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Fall 1: $j = 1$: $w_1 + w_2 > b$.
$\Rightarrow cost_T \geq w_1 > \frac{b}{2}$.

Fall 2: $j \geq 2$:

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j+1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Fall 1: $j = 1$: $w_1 + w_2 > b$.
$\Rightarrow cost_T \geq w_1 > \frac{b}{2}$.

Fall 2: $j \geq 2$: $cost_T + w_{j+1} > b \geq OPT_{SKP}(w_1, ..., w_n, b)$.

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j+1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Fall 1: $j = 1$: $w_1 + w_2 > b$.
$\Rightarrow cost_T \geq w_1 > \frac{b}{2}$.

Fall 2: $j \geq 2$: $cost_T + w_{j+1} > b \geq OPT_{SKP}(w_1, ..., w_n, b)$.
Wegen $w_1 \geq w_2 \geq ... \geq w_n$ gilt:
$w_{j+1} \leq w_j \leq \frac{w_1+w_2+...+w_j}{j} \leq \frac{b}{j}$.

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j+1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Fall 1: $j = 1$: $w_1 + w_2 > b$.
$\Rightarrow cost_T \geq w_1 > \frac{b}{2}$.

Fall 2: $j \geq 2$: $cost_T + w_{j+1} > b \geq OPT_{SKP}(w_1, ..., w_n, b)$.
Wegen $w_1 \geq w_2 \geq ... \geq w_n$ gilt:
$w_{j+1} \leq w_j \leq \frac{w_1+w_2+...+w_j}{j} \leq \frac{b}{j}$.
$\Rightarrow \; cost_T > b - w_{j+1} \geq b - \frac{b}{j} \geq \frac{b}{2}$ für $j \geq 2$

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j+1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Fall 1: $j = 1$: $w_1 + w_2 > b$.
$\Rightarrow cost_T \geq w_1 > \frac{b}{2}$.

Fall 2: $j \geq 2$: $cost_T + w_{j+1} > b \geq OPT_{SKP}(w_1, ..., w_n, b)$.
Wegen $w_1 \geq w_2 \geq ... \geq w_n$ gilt:
$w_{j+1} \leq w_j \leq \frac{w_1 + w_2 + ... + w_j}{j} \leq \frac{b}{j}$.
$\Rightarrow \; cost_T > b - w_{j+1} \geq b - \frac{b}{j} \geq \frac{b}{2}$ für $j \geq 2$

$\Rightarrow$ Es ist eine 2-Approximation.

Güte des Approximationsalgorithmus

ObdA gilt $b \geq w_1 \geq w_2 \geq ... \geq w_n$.
Sei $j+1$ als erster Index nicht in T. Dabei ist $j \geq 1$, da $w_1 \leq b$ und damit $1 \in T$.

Fall 1: $j = 1$: $w_1 + w_2 > b$.
$\Rightarrow cost_T \geq w_1 > \frac{b}{2}$.

Fall 2: $j \geq 2$: $cost_T + w_{j+1} > b \geq OPT_{SKP}(w_1, ..., w_n, b)$.
Wegen $w_1 \geq w_2 \geq ... \geq w_n$ gilt:
$w_{j+1} \leq w_j \leq \frac{w_1+w_2+...+w_j}{j} \leq \frac{b}{j}$.
$\Rightarrow \; cost_T > b - w_{j+1} \geq b - \frac{b}{j} \geq \frac{b}{2}$ für $j \geq 2$

$\Rightarrow$ Es ist eine 2-Approximation.

Idee für PTAS

$T_{OPT} = \{i_1, i_2, ..., i_r\}$ mit $w_{i_1} \geq w_{i_2} \geq ... \geq w_{i_r}$ ist eine optimale Lösung für ein SKP.

Idee für PTAS

$T_{OPT} = \{i_1, i_2, ..., i_r\}$ mit $w_{i_1} \geq w_{i_2} \geq ... \geq w_{i_r}$ ist eine optimale Lösung für ein SKP.
Angenommen Lösung T enthält j höchsten Gewichte von T_{OPT} und $cost_T + w_{i_{j+1}} > b$.

Idee für PTAS

$T_{OPT} = \{i_1, i_2, ..., i_r\}$ mit $w_{i_1} \geq w_{i_2} \geq ... \geq w_{i_r}$ ist eine optimale Lösung für ein SKP.
Angenommen Lösung T enthält j höchsten Gewichte von T_{OPT} und $cost_T + w_{i_{j+1}} > b$.
Dann gilt:

$$cost(T_{OPT}) - cost_T \leq w_{i_{j+1}} \leq \frac{b}{i_{j+1} - 1} \leq \frac{b}{j}$$

PTAS für SKP

Algorithmus 2

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

PTAS für SKP

Algorithmus 2

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

Schritt 1: Sortiere w_1 bis w_n nach Größe, sodass $w_1 \geq w_2 \geq ... \geq w_n$ gilt.

PTAS für SKP

Algorithmus 2

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

Schritt 1: Sortiere w_1 bis w_n nach Größe, sodass $w_1 \geq w_2 \geq ... \geq w_n$ gilt.

Schritt 2: $k := \lceil \frac{1}{\epsilon} \rceil$.

PTAS für SKP

Algorithmus 2

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

Schritt 1: Sortiere w_1 bis w_n nach Größe, sodass $w_1 \geq w_2 \geq ... \geq w_n$ gilt.

Schritt 2: $k := \lceil \frac{1}{\epsilon} \rceil$.

Schritt 3: Für jede Untermenge $S \subseteq \{1, 2, ..., n\}$ mit $|S| \leq k$ und $\sum_{i \in S} w_i \leq b$, erweitere S zu S^* durch Schritt 3 von Algorithmus 1(Greedy).
Speichere das momentan teuerste S^*.

PTAS für SKP

Algorithmus 2

Input: Positive Integer $w_1, w_2, ..., w_n, b$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

Schritt 1: Sortiere w_1 bis w_n nach Größe, sodass $w_1 \geq w_2 \geq ... \geq w_n$ gilt.

Schritt 2: $k := \lceil \frac{1}{\epsilon} \rceil$.

Schritt 3: Für jede Untermenge $S \subseteq \{1, 2, ..., n\}$ mit $|S| \leq k$ und $\sum_{i \in S} w_i \leq b$, erweitere S zu S^* durch Schritt 3 von Algorithmus 1(Greedy).
Speichere das momentan teuerste S^*.

Output: S^* mit einem Maximum von $cost_{S^*}$ aus allen in Schritt 3 gebildeten Mengen.

Algoithmus 2 ist ein PTAS

Satz 1

Algorithmus 2 ist ein PTAS für das SKP.

Algoithmus 2 ist ein PTAS

Satz 1

Algorithmus 2 ist ein PTAS für das SKP.

Beweis

Zu Zeigen: Laufzeitverhalten ist polynomiell in n für fixes ϵ und für die Lösung S^* gilt $S^* \cdot (1 + \epsilon) = T_{OPT}$.

Zeitbeweis für PTAS

zu zeigen:

Laufzeitverhalten ist polynomiell in n für fixes ϵ.

Zeitbeweis für PTAS

zu zeigen:

Laufzeitverhalten ist polynomiell in n für fixes ϵ.

Beweis

Schritt 1: $O(n\, log(n))$.

Zeitbeweis für PTAS

zu zeigen:

Laufzeitverhalten ist polynomiell in n für fixes ϵ.

Beweis

Schritt 1: $O(n\, log(n))$.
Schritt 2: $O(1)$.

Zeitbeweis für PTAS

zu zeigen:

Laufzeitverhalten ist polynomiell in n für fixes ϵ.

Beweis

Schritt 1: $O(n\, log(n))$.
Schritt 2: $O(1)$.
Schritt 3: Anzahl Mengen S mit $|S| \leq k$:

$$\sum_{0 \leq i \leq k} \binom{n}{i} \leq \sum_{0 \leq i \leq k} n^i = \frac{n^{k+1}-1}{n-1} = O(n^k).$$

Zeitbeweis für PTAS

zu zeigen:

Laufzeitverhalten ist polynomiell in n für fixes ϵ.

Beweis

Schritt 1: $O(n\, log(n))$.
Schritt 2: $O(1)$.
Schritt 3: Anzahl Mengen S mit $|S| \leq k$:
$\sum_{0 \leq i \leq k} \binom{n}{i} \leq \sum_{0 \leq i \leq k} n^i = \frac{n^{k+1}-1}{n-1} = O(n^k)$.
Dauer für das Finden der nächsten Menge: $O(1)$.

Zeitbeweis für PTAS

zu zeigen:

Laufzeitverhalten ist polynomiell in n für fixes ϵ.

Beweis

Schritt 1: $O(n\,log(n))$.
Schritt 2: $O(1)$.
Schritt 3: Anzahl Mengen S mit $|S| \leq k$:
$\sum_{0 \leq i \leq k} \binom{n}{i} \leq \sum_{0 \leq i \leq k} n^i = \frac{n^{k+1}-1}{n-1} = O(n^k)$.
Dauer für das Finden der nächsten Menge: $O(1)$.
Erweitern von S zu S^* mithilfe von Algorithmus 1 Schritt 3: $O(n)$

Zeitbeweis für PTAS

zu zeigen:

Laufzeitverhalten ist polynomiell in n für fixes ϵ.

Beweis

Schritt 1: $O(n\,log(n))$.
Schritt 2: $O(1)$.
Schritt 3: Anzahl Mengen S mit $|S| \leq k$:
$\sum_{0 \leq i \leq k} \binom{n}{i} \leq \sum_{0 \leq i \leq k} n^i = \frac{n^{k+1}-1}{n-1} = O(n^k)$.
Dauer für das Finden der nächsten Menge: $O(1)$.
Erweitern von S zu S^* mithilfe von Algorithmus 1 Schritt 3: $O(n)$
$\Rightarrow$
$Time(n) \leq \left[\sum_{0 \leq i \leq k} \binom{n}{i}\right] \cdot O(n) = O(n^{k+1}) = O(n^{\lceil \frac{1}{\epsilon} \rceil + 1})$.

Gütebeweis für PTAS

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1+\epsilon$

$R_{Algorithmus2}(I, \epsilon) \leq 1 + \frac{1}{k} \leq 1 + \epsilon$

Gütebeweis für PTAS

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1+\epsilon$
$R_{Algorithmus2}(I, \epsilon) \leq 1 + \frac{1}{k} \leq 1 + \epsilon$

Beweis

Sei $M = \{i_1, i_2, ..., i_p\}, i_1 \leq i_2 \leq ... \leq i_p$ eine optimale Lösung.
2 Möglichkeiten:

Gütebeweis für PTAS

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1+\epsilon$

$R_{Algorithmus2}(I,\epsilon) \leq 1 + \frac{1}{k} \leq 1 + \epsilon$

Beweis

Sei $M = \{i_1, i_2, ..., i_p\}, i_1 \leq i_2 \leq ... \leq i_p$ eine optimale Lösung.

2 Möglichkeiten:

$p \leq k$: M ist eine der in Schritt 3 vorgeschlagenen Mengen und damit ist S^* optimal.

Gütebeweis für PTAS

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1+\epsilon$

$R_{Algorithmus2}(I, \epsilon) \leq 1 + \frac{1}{k} \leq 1 + \epsilon$

Beweis

Sei $M = \{i_1, i_2, ..., i_p\}, i_1 \leq i_2 \leq ... \leq i_p$ eine optimale Lösung.

2 Möglichkeiten:

$p \leq k$: M ist eine der in Schritt 3 vorgeschlagenen Mengen und damit ist S^* optimal.

$k < p$: Algorithmus 2 Schritt 3 liefert eine Menge $P = \{i_1, i_2, ..., i_k\}$ mit den Indizes der k größten Gewichten von M. Falls $P^* = M$ gilt, sind wir fertig.

Gütebeweis für PTAS-2

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1 + \epsilon$

$R_{Algorithmus2}(I, \epsilon) \leq 1 + \frac{1}{k} \leq 1 + \epsilon.$

Gütebeweis für PTAS-2

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1 + \epsilon$

$R_{Algorithmus2}(I, \epsilon) \leq 1 + \frac{1}{k} \leq 1 + \epsilon.$

Beweis Teil 2

Sei nun $P^* \neq M$.

Gütebeweis für PTAS-2

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1+\epsilon$
$R_{Algorithmus2}(I,\epsilon) \leq 1 + \frac{1}{k} \leq 1 + \epsilon$.

Beweis Teil 2

Sei nun $P^* \neq M$.
$\exists i_q \in M - P^*$ mit $i_q > i_k \geq k$ und $cost_{P*} + w_{i_q} > b \geq cost_M$.

Gütebeweis für PTAS-2

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1 + \epsilon$
$R_{Algorithmus2}(I, \epsilon) \leq 1 + \frac{1}{k} \leq 1 + \epsilon$.

Beweis Teil 2

Sei nun $P^* \neq M$.
$\exists i_q \in M - P^*$ mit $i_q > i_k \geq k$ und $cost_{P^*} + w_{i_q} > b \geq cost_M$.
Außerdem gilt $w_{i_q} \leq \frac{w_{i_1} + w_{i_2} + \ldots + w_{i_k} + w_{i_q}}{k+1} \leq \frac{cost_M}{k+1}$

Gütebeweis für PTAS-2

zu zeigen:

Genauigkeit des Algorithmus liegt immer unter $1+\epsilon$
$R_{Algorithmus2}(I,\epsilon) \leq 1+\frac{1}{k} \leq 1+\epsilon$.

Beweis Teil 2

Sei nun $P^* \neq M$.
$\exists i_q \in M - P^*$ mit $i_q > i_k \geq k$ und $cost_{P^*} + w_{i_q} > b \geq cost_M$.
Außerdem gilt $w_{i_q} \leq \frac{w_{i_1}+w_{i_2}+\ldots+w_{i_k}+w_{i_q}}{k+1} \leq \frac{cost_M}{k+1}$
Damit erhalten wir:
$R(I,\epsilon) = \frac{cost_M}{cost_{S^*}} \leq \frac{cost_M}{cost_{P^*}} \leq \frac{cost_M}{cost_M - w_{i_q}} \leq \frac{cost_M}{cost_M-(cost_M/k+1)} = \frac{1}{1-(1/k+1)} = \frac{k+1}{k} = 1+\frac{1}{k} \leq 1+\epsilon$

DIST-Funktion

Erweiterung der Eingabe um $c_1, ..., c_n$=" Kosten/Wert".
bisher: $w_i = c_i \forall i \in \{1, ..., n\}$.

DIST-Funktion

Erweiterung der Eingabe um $c_1, ..., c_n$="Kosten/Wert".
bisher: $w_i = c_i \forall i \in \{1, ..., n\}$.
DIST-Funktion: relativer Abstand von neuer Eingabe zu SKP.

$$DIST(w_1, ..., w_n, b, c_1, ..., c_n) =$$
$$max\{max\{\tfrac{c_i - w_i}{w_i} | c_i \geq w_i, i \in \{1, ..., n\}\},$$
$$max\{\tfrac{w_i - c_i}{c_i} | w_i \geq c_i, i \in \{1, ..., n\}\}\}.$$

KP_δ

Benennungen

KP_δ ist die Teilmenge der Rucksackprobleme mit der Bedingung, dass die Kosten und Gewichte der Gegenstände maximal um Faktor $1+\delta$ auseinander liegen.

KP_δ

Benennungen

KP_δ ist die Teilmenge der Rucksackprobleme mit der Bedingung, dass die Kosten und Gewichte der Gegenstände maximal um Faktor $1+\delta$ auseinander liegen.
$\{ASKP_\epsilon\}_{\epsilon>0}$ Menge der $(1+\epsilon)$-Approximationsalgorithmen entsprechend Algorithmus 2.

KP_δ

Benennungen

KP_δ ist die Teilmenge der Rucksackprobleme mit der Bedingung, dass die Kosten und Gewichte der Gegenstände maximal um Faktor $1+\delta$ auseinander liegen.
$\{ASKP_\epsilon\}_{\epsilon>0}$ Menge der $(1+\epsilon)$-Approximationsalgorithmen entsprechend Algorithmus 2.

Lemma

Für jedes $\epsilon > 0$ und jedes $\delta > 0$ ist der Algorithmus $ASKP_\epsilon$ ein $(1+\epsilon+\delta(2+\delta)\cdot(1+\epsilon))$-Approximationsalgorithmus für KP_δ.

Beweis

Lemma

Für jedes $\epsilon > 0$ und jedes $\delta > 0$ ist der Algorithmus $ASKP_\epsilon$ ein $(1 + \epsilon + \delta(2 + \delta) \cdot (1 + \epsilon))$-Approximationsalgorithmus für KP_δ.

Beweis

Lemma

Für jedes $\epsilon > 0$ und jedes $\delta > 0$ ist der Algorithmus $ASKP_\epsilon$ ein $(1 + \epsilon + \delta(2 + \delta) \cdot (1 + \epsilon))$-Approximationsalgorithmus für KP_δ.

Beweis

$w_1 \geq w_2 \geq ... \geq w_n$ für Input $I = w_1, ..., w_n, b, c_1, ..., c_n$.
$k = \lceil \frac{1}{\epsilon} \rceil$.
$U = \{i_1, ..., i_l\} \subseteq \{1, 2, ..., n\}$ ist eine optimale Lösung für I.

Beweis

Lemma

Für jedes $\epsilon > 0$ und jedes $\delta > 0$ ist der Algorithmus $ASKP_\epsilon$ ein $(1 + \epsilon + \delta(2 + \delta) \cdot (1 + \epsilon))$-Approximationsalgorithmus für KP_δ.

Beweis

$w_1 \geq w_2 \geq ... \geq w_n$ für Input $I = w_1, ..., w_n, b, c_1, ..., c_n$.
$k = \lceil \frac{1}{\epsilon} \rceil$.
$U = \{i_1, ..., i_l\} \subseteq \{1, 2, ..., n\}$ ist eine optimale Lösung für I.

$l \leq k$ $ASKP_\epsilon$ hat eine optimale Lösung für I mit $cost_U$ gefunden.

Beweis

Lemma

Für jedes $\epsilon > 0$ und jedes $\delta > 0$ ist der Algorithmus $ASKP_\epsilon$ ein $(1 + \epsilon + \delta(2 + \delta) \cdot (1 + \epsilon))$-Approximationsalgorithmus für KP_δ.

Beweis

$w_1 \geq w_2 \geq ... \geq w_n$ für Input $I = w_1, ..., w_n, b, c_1, ..., c_n$.
$k = \left\lceil \frac{1}{\epsilon} \right\rceil$.
$U = \{i_1, ..., i_l\} \subseteq \{1, 2, ..., n\}$ ist eine optimale Lösung für I.

$l \leq k$ $ASKP_\epsilon$ hat eine optimale Lösung für I mit $cost_U$ gefunden.

$l > k$ $ASKP_\epsilon$ hat eine greedy-Erweiterung von $T = \{i_1, i_2, ..., i_k\}$ zu $T^* = \{i_1, i_2, ..., i_k, j_{k+1}, ..., j_{k+r}\}$ gefunden.

Beweis

Beweis-2

Zeige nun dass $cost_U - cost_{T^*}$ relativ zu $cost_U$ klein ist:

Beweis

Beweis-2

Zeige nun dass $cost_U - cost_{T^*}$ relativ zu $cost_U$ klein ist:
Betrachte die Gewichte von U und T^*:
Es gelte $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j \leq 0$.

Beweis

Beweis-2

Zeige nun dass $cost_U - cost_{T^*}$ relativ zu $cost_U$ klein ist:
Betrachte die Gewichte von U und T^*:
Es gelte $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j \leq 0$.
für jedes i gilt:$(1+\delta)^{-1} \leq \frac{c_i}{w_i} \leq 1+\delta$.

Beweis

Beweis-2

Zeige nun dass $cost_U - cost_{T^*}$ relativ zu $cost_U$ klein ist:
Betrachte die Gewichte von U und T^*:
Es gelte $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j \leq 0$.
für jedes i gilt:$(1+\delta)^{-1} \leq \frac{c_i}{w_i} \leq 1+\delta$.
deshalb gilt: $cost_U = \sum_{i \in U} c_i \leq (1+\delta) \cdot \sum_{i \in U} w_i$

Beweis

Beweis-2

Zeige nun dass $cost_U - cost_{T^*}$ relativ zu $cost_U$ klein ist:
Betrachte die Gewichte von U und T^*:
Es gelte $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j \leq 0$.
für jedes i gilt:$(1+\delta)^{-1} \leq \frac{c_i}{w_i} \leq 1+\delta$.
deshalb gilt: $cost_U = \sum_{i \in U} c_i \leq (1+\delta) \cdot \sum_{i \in U} w_i$
und $cost_{T^*} = \sum_{j \in T^*} c_j \geq (1+\delta)^{-1} \cdot \sum_{j \in T^*} w_j$.

Beweis

Beweis-3

Somit erhalten wir:

$cost_U - cost_{T^*}$

Beweis

Beweis-3

Somit erhalten wir:

$cost_U - cost_{T*}$
$\leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{j \in T*} w_j$

Beweis

Beweis-3

Somit erhalten wir:

$cost_U - cost_{T*}$
$\leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{j \in T*} w_j$
$\leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{i \in U} w_i$

Beweis

Beweis-3

Somit erhalten wir:

$$cost_U - cost_{T^*}$$
$$\leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{j \in T^*} w_j$$
$$\leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{i \in U} w_i$$
$$= \frac{\delta \cdot (2+\delta)}{1+\delta} \cdot \sum_{i \in U} w_i$$

Beweis

Beweis-3

Somit erhalten wir:

$$
\begin{aligned}
& cost_U - cost_{T^*} \\
& \leq (1+\delta) \cdot \textstyle\sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{j \in T^*} w_j \\
& \leq (1+\delta) \cdot \textstyle\sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{i \in U} w_i \\
& = \tfrac{\delta \cdot (2+\delta)}{1+\delta} \cdot \textstyle\sum_{i \in U} w_i \\
& \leq \tfrac{\delta \cdot (2+\delta)}{1+\delta} \textstyle\sum_{i \in U} (1+\delta) c_i
\end{aligned}
$$

Beweis

Beweis-3

Somit erhalten wir:

$$
\begin{aligned}
& cost_U - cost_{T^*} \\
& \leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{j \in T^*} w_j \\
& \leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{i \in U} w_i \\
& = \frac{\delta \cdot (2+\delta)}{1+\delta} \cdot \sum_{i \in U} w_i \\
& \leq \frac{\delta \cdot (2+\delta)}{1+\delta} \sum_{i \in U} (1+\delta) c_i \\
& = \delta \cdot (2+\delta) \cdot \sum_{i \in U} c_i
\end{aligned}
$$

Beweis

Beweis-3

Somit erhalten wir:

$$
\begin{aligned}
& cost_U - cost_{T^*} \\
& \leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{j \in T^*} w_j \\
& \leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{i \in U} w_i \\
& = \frac{\delta \cdot (2+\delta)}{1+\delta} \cdot \sum_{i \in U} w_i \\
& \leq \frac{\delta \cdot (2+\delta)}{1+\delta} \sum_{i \in U} (1+\delta) c_i \\
& = \delta \cdot (2+\delta) \cdot \sum_{i \in U} c_i \\
& = \delta \cdot (2+\delta) \cdot cost_U
\end{aligned}
$$

Beweis

Beweis-3

Somit erhalten wir:

$$
\begin{aligned}
& cost_U - cost_{T^*} \\
& \leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{j \in T^*} w_j \\
& \leq (1+\delta) \cdot \sum_{i \in U} w_i - (1+\delta)^{-1} \cdot \sum_{i \in U} w_i \\
& = \frac{\delta \cdot (2+\delta)}{1+\delta} \cdot \sum_{i \in U} w_i \\
& \leq \frac{\delta \cdot (2+\delta)}{1+\delta} \sum_{i \in U} (1+\delta) c_i \\
& = \delta \cdot (2+\delta) \cdot \sum_{i \in U} c_i \\
& = \delta \cdot (2+\delta) \cdot cost_U
\end{aligned}
$$

Und damit: $\frac{cost_U - cost_{T^*}}{cost_U} \leq \frac{\delta \cdot (2+\delta) \cdot cost_U}{cost_U} = \delta \cdot (2+\delta)$.

Beweis

Beweis-4

Gelte nun $d = \sum_{i \in U} w_i - \sum_{j \in T^*} w_j > 0$.
c :=kosten des ersten Teils von U mit Gewicht $\sum_{j \in T^*} w_j$.

Beweis

Beweis-4

Gelte nun $d = \sum_{i \in U} w_i - \sum_{j \in T*} w_j > 0$.
c :=kosten des ersten Teils von U mit Gewicht $\sum_{j \in T*} w_j$.
Wie vorher gilt nun: $\frac{c - cost_{T*}}{c} \leq \delta \cdot (2 + \delta)$.

Beweis

Beweis-4

Gelte nun $d = \sum_{i \in U} w_i - \sum_{j \in T^*} w_j > 0$.
c :=kosten des ersten Teils von U mit Gewicht $\sum_{j \in T^*} w_j$.
Wie vorher gilt nun: $\frac{c - cost_{T^*}}{c} \leq \delta \cdot (2 + \delta)$.
Offensichtlich gilt: $d \leq b - \sum_{j \in T^*} w_j \leq w_{i_r}$ für einige $r > k, i_r \in U$.

Beweis

Beweis-4

Gelte nun $d = \sum_{i \in U} w_i - \sum_{j \in T^*} w_j > 0$.
c :=kosten des ersten Teils von U mit Gewicht $\sum_{j \in T^*} w_j$.
Wie vorher gilt nun: $\frac{c - cost_{T^*}}{c} \leq \delta \cdot (2 + \delta)$.
Offensichtlich gilt: $d \leq b - \sum_{j \in T^*} w_j \leq w_{i_r}$ für einige $r > k, i_r \in U$.
Damit gilt: $d \leq w_{i_r} \leq \frac{w_{i_1} + w_{i_2} + \ldots + w_{i_r}}{r} \leq \frac{\sum_{i \in U} w_i}{k+1} \leq \epsilon \cdot \sum_{i \in U} w_i$.

Beweis

Beweis-5

Mit $cost_U \leq c + d \cdot (1 + \delta)$ erhalten wir:

Beweis

Beweis-5

Mit $cost_U \leq c + d \cdot (1 + \delta)$ erhalten wir:
$\frac{cost_U - cost_{T*}}{cost_U}$

Beweis

Beweis-5

Mit $cost_U \leq c + d \cdot (1 + \delta)$ erhalten wir:

$\frac{cost_U - cost_{T*}}{cost_U}$

$\leq \frac{c + d \cdot (1+\delta) - cost_{T*}}{cost_U}$

Beweis

Beweis-5

Mit $cost_U \leq c + d \cdot (1+\delta)$ erhalten wir:

$$\frac{cost_U - cost_{T*}}{cost_U}$$

$$\leq \frac{c + d \cdot (1+\delta) - cost_{T*}}{cost_U}$$

$$\leq \frac{c - cost_{T*}}{cost_U} + \frac{(1+\delta) \cdot \epsilon \cdot \sum_{i \in U} w_i}{cost_U}$$

Beweis

Beweis-5

Mit $cost_U \leq c + d \cdot (1 + \delta)$ erhalten wir:

$$\frac{cost_U - cost_{T*}}{cost_U}$$
$$\leq \frac{c + d \cdot (1+\delta) - cost_{T*}}{cost_U}$$
$$\leq \frac{c - cost_{T*}}{cost_U} + \frac{(1+\delta) \cdot \epsilon \cdot \sum_{i \in U} w_i}{cost_U}$$
$$\leq \delta \cdot (2 + \delta) + (1 + \delta) \cdot \epsilon \cdot (1 + \delta)$$

Beweis

Beweis-5

Mit $cost_U \leq c + d \cdot (1 + \delta)$ erhalten wir:

$$\frac{cost_U - cost_{T*}}{cost_U}$$
$$\leq \frac{c + d \cdot (1 + \delta) - cost_{T*}}{cost_U}$$
$$\leq \frac{c - cost_{T*}}{cost_U} + \frac{(1 + \delta) \cdot \epsilon \cdot \sum_{i \in U} w_i}{cost_U}$$
$$\leq \delta \cdot (2 + \delta) + (1 + \delta) \cdot \epsilon \cdot (1 + \delta)$$
$$= 2\delta + \delta^2 + \epsilon \cdot (1 + \delta)^2$$

Beweis

Beweis-5

Mit $cost_U \leq c + d \cdot (1+\delta)$ erhalten wir:

$$\frac{cost_U - cost_{T*}}{cost_U}$$
$$\leq \frac{c + d \cdot (1+\delta) - cost_{T*}}{cost_U}$$
$$\leq \frac{c - cost_{T*}}{cost_U} + \frac{(1+\delta) \cdot \epsilon \cdot \sum_{i \in U} w_i}{cost_U}$$
$$\leq \delta \cdot (2+\delta) + (1+\delta) \cdot \epsilon \cdot (1+\delta)$$
$$= 2\delta + \delta^2 + \epsilon \cdot (1+\delta)^2$$
$$= \epsilon + \delta \cdot (2+\delta) \cdot (1+\epsilon)$$

Stabilität

Korollar

Algorithmus 2 ist stabil bezüglich $DIST$, aber nicht superstabil.

Stabilität

Korollar

Algorithmus 2 ist stabil bezüglich $DIST$, aber nicht superstabil.

Beweis

Stabilität falls gilt $||\hat{f}(x) - f(\tilde{x})|| = O(\epsilon_{mach})$.
Erfüllt, da es eine $1 + \epsilon + \delta \cdot (2 + \delta) \cdot (1 + \epsilon)$-Approximation ist, und es somit für jedes ϵ ein δ gibt, sodass die Approximation beliebig nahe an der Approximation von SKP ist.

Stabilität

Beweis-2

Für superstabil betrachte die Eingabe $w_1, w_2, ..., w_m, u_1, u_2, ..., u_m, b, c_1, c_2, ..., c_{2m}$, mit $w_1 = w_2 = ... = w_m,\ u_1 = u_2 = ... = u_m,\ w_i = u_i + 1$ für $i = 1, ..., m.\ b = \sum_{i=1}^{m} w_i = m \cdot w_1,\ c_1 = c_2 = ... = c_m = (1 - \delta)w_1,\ c_{m+1} = c_{m+2} = ... = c_{2m} = (1 + \delta)u_1$.

Stabilität

Beweis-2

Für superstabil betrachte die Eingabe $w_1, w_2, ..., w_m, u_1, u_2, ..., u_m, b, c_1, c_2, ..., c_{2m}$, mit $w_1 = w_2 = ... = w_m,\ u_1 = u_2 = ... = u_m,\ w_i = u_i + 1$ für $i = 1, ..., m$. $b = \sum_{i=1}^{m} w_i = m \cdot w_1,\ c_1 = c_2 = ... = c_m = (1-\delta)w_1,\ c_{m+1} = c_{m+2} = ... = c_{2m} = (1+\delta)u_1$.
$\nexists\delta$ sodass die Kosten des Algorithmus für KP_δ nahe der Kosten von Algorithmus für SKP für alle ϵ liegen.

Stabilität

Beweis-2

Für superstabil betrachte die Eingabe $w_1, w_2, ..., w_m, u_1, u_2, ..., u_m, b, c_1, c_2, ..., c_{2m}$, mit $w_1 = w_2 = ... = w_m,\ u_1 = u_2 = ... = u_m,\ w_i = u_i + 1$ für $i = 1, ..., m$. $b = \sum_{i=1}^{m} w_i = m \cdot w_1,\ c_1 = c_2 = ... = c_m = (1-\delta)w_1,\ c_{m+1} = c_{m+2} = ... = c_{2m} = (1+\delta)u_1$.
$\nexists \delta$ sodass die Kosten des Algorithmus für KP_δ nahe der Kosten von Algorithmus für SKP für alle ϵ liegen.

Folgerung

Neuer Algorithmus mit anderer Form der Sortierung.

PTAS für Modifiziertes SKP

Algorithmus 3

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

PTAS für Modifiziertes SKP

Algorithmus 3

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

Schritt 1: Sortiere $\frac{c_1}{w_1}, \frac{c_2}{w_2}, ..., \frac{c_n}{w_n}$, sodass $\frac{c_i}{w_i} \geq \frac{c_{i+1}}{w_{i+1}}$ für $i = 1, ..., n-1$ gilt.

PTAS für Modifiziertes SKP

Algorithmus 3

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

Schritt 1: Sortiere $\frac{c_1}{w_1}, \frac{c_2}{w_2}, ..., \frac{c_n}{w_n}$, sodass $\frac{c_i}{w_i} \geq \frac{c_{i+1}}{w_{i+1}}$ für $i = 1, ..., n-1$ gilt.

Schritt 2: $k := \left\lceil \frac{1}{\epsilon} \right\rceil$.

PTAS für Modifiziertes SKP

Algorithmus 3

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

Schritt 1: Sortiere $\frac{c_1}{w_1}, \frac{c_2}{w_2}, ..., \frac{c_n}{w_n}$, sodass $\frac{c_i}{w_i} \geq \frac{c_{i+1}}{w_{i+1}}$ für $i = 1, ..., n-1$ gilt.

Schritt 2: $k := \lceil \frac{1}{\epsilon} \rceil$.

Schritt 3: (Wie in Algorithmus 2 nur mit anderer Ordnung)
Für jede Untermenge $S \subseteq \{1, 2, ..., n\}$ mit $|S| \leq k$ und $\sum_{i \in S} w_i \leq b$, erweitere S zu S^* durch Schritt 3 von Algorithmus 1(Greedy).
Speichere das momentan teuerste S^*.

PTAS für Modifiziertes SKP

Algorithmus 3

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}$ mit $0 < \epsilon < 1$.

Schritt 1: Sortiere $\frac{c_1}{w_1}, \frac{c_2}{w_2}, ..., \frac{c_n}{w_n}$, sodass $\frac{c_i}{w_i} \geq \frac{c_{i+1}}{w_{i+1}}$ für $i = 1, ..., n-1$ gilt.

Schritt 2: $k := \lceil \frac{1}{\epsilon} \rceil$.

Schritt 3: (Wie in Algorithmus 2 nur mit anderer Ordnung)
Für jede Untermenge $S \subseteq \{1, 2, ..., n\}$ mit $|S| \leq k$ und $\sum_{i \in S} w_i \leq b$, erweitere S zu S^* durch Schritt 3 von Algorithmus 1(Greedy).
Speichere das momentan teuerste S^*.

Output: Das beste T^* das in Schritt 3 gebildet wurde.

Approximationsgüte von Algorithmus 3

$MOD - SKP_\epsilon$ sei der Algorithmus für ein bestimmtes $\epsilon > 0$.

Approximationsgüte von Algorithmus 3

$MOD - SKP_\epsilon$ sei der Algorithmus für ein bestimmtes $\epsilon > 0$.

Lemma

Algorithmus 3 ist eine $1 + \epsilon \cdot (1 + \delta)^2$-Approximation für KP_δ.

Approximationsgüte von Algorithmus 3

$MOD - SKP_\epsilon$ sei der Algorithmus für ein bestimmtes $\epsilon > 0$.

Lemma

Algorithmus 3 ist eine $1 + \epsilon \cdot (1 + \delta)^2$-Approximation für KP_δ.

Beweis-1

Sei U=$\{i_1, i_2, ..., i_l\} \subseteq \{1, 2, ..., n\}$ eine optimale Lösung.

Approximationsgüte von Algorithmus 3

$MOD - SKP_{\epsilon}$ sei der Algorithmus für ein bestimmtes $\epsilon > 0$.

Lemma

Algorithmus 3 ist eine $1 + \epsilon \cdot (1 + \delta)^2$-Approximation für KP_{δ}.

Beweis-1

Sei U=$\{i_1, i_2, ..., i_l\} \subseteq \{1, 2, ..., n\}$ eine optimale Lösung.

$l \leq k$: Der Algorithmus liefert eine optimale Lösung.

Approximationsgüte von Algorithmus 3

$MOD - SKP_\epsilon$ sei der Algorithmus für ein bestimmtes $\epsilon > 0$.

Lemma

Algorithmus 3 ist eine $1 + \epsilon \cdot (1 + \delta)^2$-Approximation für KP_δ.

Beweis-1

Sei U$=\{i_1, i_2, ..., i_l\} \subseteq \{1, 2, ..., n\}$ eine optimale Lösung.

$l \leq k$: Der Algorithmus liefert eine optimale Lösung.

$l > k$: T^* ist die greedy-Erweiterung von $T = \{i_1, i_2, ..., i_k\}$.

Approximationsgüte von Algorithmus 3

Beweis-2

Betrachte die beiden Möglichkeiten abhängig von den Größen $\sum_{i \in U} w_i$ und $\sum_{j \in T^*} w_j$:

Approximationsgüte von Algorithmus 3

Beweis-2

Betrachte die beiden Möglichkeiten abhängig von den Größen $\sum_{i \in U} w_i$ und $\sum_{j \in T^*} w_j$:
Sei $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j < 0$.

Approximationsgüte von Algorithmus 3

Beweis-2

Betrachte die beiden Möglichkeiten abhängig von den Größen $\sum_{i \in U} w_i$ und $\sum_{j \in T^*} w_j$:
Sei $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j < 0$.
Nicht möglich, da dann $cost_U < cost_{T^*}$ und damit U nicht optimal wäre.

Approximationsgüte von Algorithmus 3

Beweis-2

Betrachte die beiden Möglichkeiten abhängig von den Größen $\sum_{i \in U} w_i$ und $\sum_{j \in T^*} w_j$:
Sei $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j < 0$.
Nicht möglich, da dann $cost_U < cost_{T^*}$ und damit U nicht optimal wäre.
Sei demnach $d = \sum_{i \in U} w_i - \sum_{j \in T^*} w_j \geq 0$.

Approximationsgüte von Algorithmus 3

Beweis-2

Betrachte die beiden Möglichkeiten abhängig von den Größen $\sum_{i \in U} w_i$ und $\sum_{j \in T^*} w_j$:
Sei $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j < 0$.
Nicht möglich, da dann $cost_U < cost_{T^*}$ und damit U nicht optimal wäre.
Sei demnach $d = \sum_{i \in U} w_i - \sum_{j \in T^*} w_j \geq 0$.
Sei c die Kosten des Teiles von U mit dem Gewicht $\sum_{j \in T^*} w_j$.
Wegen der Optimalität von T^* bezüglich der Kosten pro Gewicht gilt $c - cost_{T^*} \leq 0$.

Approximationsgüte von Algorithmus 3

Beweis-3

Da $i_1, i_2, ..., i_k$ in U und T^* vorhanden sind und $w_{i_1}, ..., w_{i_k}$ die größten Gewichte in beiden sind, gilt wie vorhin gezeigt für das Restgewicht d von U:
$d \leq \epsilon \cdot \sum_{i \in U} w_i$.

Approximationsgüte von Algorithmus 3

Beweis-3

Da $i_1, i_2, ..., i_k$ in U und T^* vorhanden sind und $w_{i_1}, ..., w_{i_k}$ die größten Gewichte in beiden sind, gilt wie vorhin gezeigt für das Restgewicht d von U:
$d \leq \epsilon \cdot \sum_{i \in U} w_i$.
Auch gilt $cost_U \leq c + d \cdot (1 + \delta)$.

Approximationsgüte von Algorithmus 3

Beweis-3

Da $i_1, i_2, ..., i_k$ in U und T^* vorhanden sind und $w_{i_1}, ..., w_{i_k}$ die größten Gewichte in beiden sind, gilt wie vorhin gezeigt für das Restgewicht d von U:
$d \leq \epsilon \cdot \sum_{i \in U} w_i$.
Auch gilt $cost_U \leq c + d \cdot (1 + \delta)$.
Daraus folgt:

Approximationsgüte von Algorithmus 3

Beweis-3

Da $i_1, i_2, ..., i_k$ in U und T^* vorhanden sind und $w_{i_1}, ..., w_{i_k}$ die größten Gewichte in beiden sind, gilt wie vorhin gezeigt für das Restgewicht d von U:
$d \leq \epsilon \cdot \sum_{i \in U} w_i$.
Auch gilt $cost_U \leq c + d \cdot (1 + \delta)$.
Daraus folgt:
$\frac{cost_U - cost_{T*}}{cost_U}$

Approximationsgüte von Algorithmus 3

Beweis-3

Da $i_1, i_2, ..., i_k$ in U und T^* vorhanden sind und $w_{i_1}, ..., w_{i_k}$ die größten Gewichte in beiden sind, gilt wie vorhin gezeigt für das Restgewicht d von U:
$d \leq \epsilon \cdot \sum_{i \in U} w_i$.
Auch gilt $cost_U \leq c + d \cdot (1 + \delta)$.
Daraus folgt:
$\frac{cost_U - cost_{T*}}{cost_U}$
$\leq \frac{c + d \cdot (1+\delta) - cost_{T*}}{cost_U}$

Approximationsgüte von Algorithmus 3

Beweis-3

Da $i_1, i_2, ..., i_k$ in U und T^* vorhanden sind und $w_{i_1}, ..., w_{i_k}$ die größten Gewichte in beiden sind, gilt wie vorhin gezeigt für das Restgewicht d von U:

$d \leq \epsilon \cdot \sum_{i \in U} w_i$.

Auch gilt $cost_U \leq c + d \cdot (1 + \delta)$.

Daraus folgt:

$$\frac{cost_U - cost_{T*}}{cost_U} \leq \frac{c + d \cdot (1+\delta) - cost_{T*}}{cost_U} \leq \frac{d \cdot (1+\delta)}{cost_U}$$

Approximationsgüte von Algorithmus 3

Beweis-3

Da $i_1, i_2, ..., i_k$ in U und T^* vorhanden sind und $w_{i_1}, ..., w_{i_k}$ die größten Gewichte in beiden sind, gilt wie vorhin gezeigt für das Restgewicht d von U:

$d \leq \epsilon \cdot \sum_{i \in U} w_i$.

Auch gilt $cost_U \leq c + d \cdot (1 + \delta)$.

Daraus folgt:

$\frac{cost_U - cost_{T*}}{cost_U}$

$\leq \frac{c + d \cdot (1+\delta) - cost_{T*}}{cost_U}$

$\leq \frac{d \cdot (1+\delta)}{cost_U}$

$\leq \epsilon \cdot (1 + \delta) \cdot \frac{\sum_{i \in U} w_i}{cost_U}$

Approximationsgüte von Algorithmus 3

Beweis-3

Da $i_1, i_2, ..., i_k$ in U und T^* vorhanden sind und $w_{i_1}, ..., w_{i_k}$ die größten Gewichte in beiden sind, gilt wie vorhin gezeigt für das Restgewicht d von U:

$d \leq \epsilon \cdot \sum_{i \in U} w_i$.

Auch gilt $cost_U \leq c + d \cdot (1 + \delta)$.

Daraus folgt:

$\frac{cost_U - cost_{T*}}{cost_U}$

$\leq \frac{c + d \cdot (1+\delta) - cost_{T*}}{cost_U}$

$\leq \frac{d \cdot (1+\delta)}{cost_U}$

$\leq \epsilon \cdot (1 + \delta) \cdot \frac{\sum_{i \in U} w_i}{cost_U}$

$\leq \epsilon \cdot (1 + \delta)^2$

PTAS für KP

PTAS für KP

Satz

$MOD - SKP$ ist superstabil bezüglich $DIST$ und damit ist Algorithmus 3 ein PTAS für das KP.

PTAS für KP

Satz

$MOD - SKP$ ist superstabil bezüglich $DIST$ und damit ist Algorithmus 3 ein PTAS für das KP.

FPTAS für KP

Algorithmus 4

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

FPTAS für KP

Algorithmus 4

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Schritt 1: $c_{max} := max\{c_1, ..., c_n\}$.
$t := \left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon) \cdot n} \right\rfloor$.

FPTAS für KP

Algorithmus 4

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Schritt 1: $c_{max} := max\{c_1, ..., c_n\}$.
$t := \left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon) \cdot n} \right\rfloor$.

Schritt 2: für alle $i \in \{1, ..., n\} : c_i' := \lfloor c_i \cdot 2^{-t} \rfloor$.

FPTAS für KP

Algorithmus 4

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Schritt 1: $c_{max} := max\{c_1, ..., c_n\}$.
$t := \left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon) \cdot n} \right\rfloor$.

Schritt 2: für alle $i \in \{1, ..., n\} : c'_i := \lfloor c_i \cdot 2^{-t} \rfloor$.

Schritt 3: Berechne ein optimales Ergebnis T' für die Eingabe $I' = w_1, ..., w_n, b, c'_1, ..., c'_n$.

FPTAS für KP

Algorithmus 4

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Schritt 1: $c_{max} := max\{c_1, ..., c_n\}$.
$t := \left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon) \cdot n} \right\rfloor$.

Schritt 2: für alle $i \in \{1, ..., n\} : c_i' := \lfloor c_i \cdot 2^{-t} \rfloor$.

Schritt 3: Berechne ein optimales Ergebnis T' für die Eingabe $I' = w_1, ..., w_n, b, c_1', ..., c_n'$.

Output: T.

Methode der Optimalen Berechnung

Algorithmus 4.1

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Methode der Optimalen Berechnung

Algorithmus 4.1

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Schritt 1: $TRIPLE(1) := \{(0, 0, \emptyset)\} \cup \{(c_1, w_1, \{1\}) | if \;\; w_1 \leq b)\}$.

Methode der Optimalen Berechnung

Algorithmus 4.1

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Schritt 1: $TRIPLE(1) := \{(0,0,\emptyset)\} \cup \{(c_1, w_1, \{1\}) | if \ \ w_1 \leq b)\}$.

Schritt 2: **for** $i = 1$ **to** $n - 1$**do**

begin $Set(i+1) := TRIPLE(i)$

for every $(kw, T) \in TRIPLE(i)$ **do**

if $w + w_{i+1} \leq b$ **then**

$SET(i+1) := SET(i+1) \cup \{(k+c_{i+1}, w+w_{i+1}, T \cup \{i+1\})\}$

SET(i+1) enthält nur ein Tripel (k,w,T) für jedes k.

Dieses hat das geringste Gewicht w.

Methode der Optimalen Berechnung

Algorithmus 4.1

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Schritt 1: $TRIPLE(1) := \{(0, 0, \emptyset)\} \cup \{(c_1, w_1, \{1\}) | if \;\; w_1 \leq b)\}$.

Schritt 2: **for** $i = 1$ **to** $n - 1$**do**
  begin $Set(i + 1) := TRIPLE(i)$
    for every $(kw, T) \in TRIPLE(i)$ **do**
      if $w + w_{i+1} \leq b$ **then**
$SET(i+1) := SET(i+1) \cup \{(k+c_{i+1}, w+w_{i+1}, T \cup \{i+1\})\}$
SET(i+1) enthält nur ein Tripel (k,w,T) für jedes k.
Dieses hat das geringste Gewicht w.

Schritt 3: Suche das größte k.

Methode der Optimalen Berechnung

Algorithmus 4.1

Input: Positive Integer $w_1, w_2, ..., w_n, b, c_1, ..., c_n$ für ein bestimmtes n$\in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$.

Schritt 1: $TRIPLE(1) := \{(0, 0, \emptyset)\} \cup \{(c_1, w_1, \{1\}) | if \;\; w_1 \leq b)\}$.

Schritt 2: **for** $i = 1$ **to** $n - 1$**do**
begin $Set(i+1) := TRIPLE(i)$
for every $(kw, T) \in TRIPLE(i)$ **do**
if $w + w_{i+1} \leq b$ **then**
$SET(i+1) := SET(i+1) \cup \{(k+c_{i+1}, w+w_{i+1}, T \cup \{i+1\})\}$
SET(i+1) enthält nur ein Tripel (k,w,T) für jedes k.
Dieses hat das geringste Gewicht w.

Schritt 3: Suche das größte k.

Output: T

FPTAS für KP

Satz

Algorithmus 4 ist ein FPTAS für das KP.

Beweis-1

Nachdem T' eine optimale Lösung für I' ist, und damit eine mögliche Lösung für I'. I und I' unterscheiden sich nicht in den Gewichten. Damit ist T' auch eine mögliche Lösung für I.

FPTAS für KP

Satz

Algorithmus 4 ist ein FPTAS für das KP.

Beweis-1

Nachdem T' eine optimale Lösung für I' ist, und damit eine mögliche Lösung für I'. I und I' unterscheiden sich nicht in den Gewichten. Damit ist T' auch eine mögliche Lösung für I.
Sei T eine optimale Lösung für I.

FPTAS für KP

Beweis-2

Somit gilt:

$cost(T, I) = \sum_{j \in T} c_j$

FPTAS für KP

Beweis-2

Somit gilt:

$cost(T, I) = \sum_{j \in T} c_j$

$\geq \sum_{j \in T'} c_j = cost(T', I)$

FPTAS für KP

Beweis-2

Somit gilt:

$cost(T, I) = \sum_{j \in T} c_j$
$\geq \sum_{j \in T'} c_j = cost(T', I)$
$\geq 2^t \cdot \sum_{j \in T'} c'_j$

FPTAS für KP

Beweis-2

Somit gilt:

$cost(T, I) = \sum_{j \in T} c_j$
$\geq \sum_{j \in T'} c_j = cost(T', I)$
$\geq 2^t \cdot \sum_{j \in T'} c'_j$
$\geq 2^t \cdot \sum_{j \in T} c'_j$

FPTAS für KP

Beweis-2

Somit gilt:

$$\begin{aligned}
& cost(T, I) = \sum_{j \in T} c_j \\
& \geq \sum_{j \in T'} c_j = cost(T', I) \\
& \geq 2^t \cdot \sum_{j \in T'} c_j' \\
& \geq 2^t \cdot \sum_{j \in T} c_j' \\
& = \sum_{j \in T} 2^t \cdot \lfloor c_j \cdot 2^{-t} \rfloor
\end{aligned}$$

FPTAS für KP

Beweis-2

Somit gilt:

$$cost(T, I) = \sum_{j \in T} c_j$$
$$\geq \sum_{j \in T'} c_j = cost(T', I)$$
$$\geq 2^t \cdot \sum_{j \in T'} c'_j$$
$$\geq 2^t \cdot \sum_{j \in T} c'_j$$
$$= \sum_{j \in T} 2^t \cdot \lfloor c_j \cdot 2^{-t} \rfloor$$
$$\geq \sum_{j \in T} 2^t (c_j \cdot 2^{-t} - 1)$$

FPTAS für KP

Beweis-2

Somit gilt:

$$cost(T, I) = \sum_{j \in T} c_j$$
$$\geq \sum_{j \in T'} c_j = cost(T', I)$$
$$\geq 2^t \cdot \sum_{j \in T'} c'_j$$
$$\geq 2^t \cdot \sum_{j \in T} c'_j$$
$$= \sum_{j \in T} 2^t \cdot \lfloor c_j \cdot 2^{-t} \rfloor$$
$$\geq \sum_{j \in T} 2^t (c_j \cdot 2^{-t} - 1)$$
$$\geq \left(\sum_{j \in T} c_j\right) - n \cdot 2^t$$

FPTAS für KP

Beweis-2

Somit gilt:

$cost(T, I) = \sum_{j \in T} c_j$
$\geq \sum_{j \in T'} c_j = cost(T', I)$
$\geq 2^t \cdot \sum_{j \in T'} c_j'$
$\geq 2^t \cdot \sum_{j \in T} c_j'$
$= \sum_{j \in T} 2^t \cdot \lfloor c_j \cdot 2^{-t} \rfloor$
$\geq \sum_{j \in T} 2^t (c_j \cdot 2^{-t} - 1)$
$\geq (\sum_{j \in T} c_j) - n \cdot 2^t$
$= cost(T, I) - n \cdot 2^t$

FPTAS für KP

Beweis-3

Damit zeigen wir:
$cost(T, I) \geq cost(T', I) \geq cost(T, I) - n \cdot 2^t$

FPTAS für KP

Beweis-3

Damit zeigen wir:

$cost(T, I) \geq cost(T', I) \geq cost(T, I) - n \cdot 2^t$

$0 \leq cost(T, I) - cost(T', I) \leq n \cdot 2^t$

FPTAS für KP

Beweis-3

Damit zeigen wir:

$cost(T, I) \geq cost(T', I) \geq cost(T, I) - n \cdot 2^t$

$0 \leq cost(T, I) - cost(T', I) \leq n \cdot 2^t$

$\leq n \cdot \frac{\epsilon \cdot c_{max}}{(1+\epsilon) \cdot n}$

FPTAS für KP

Beweis-3

Damit zeigen wir:

$cost(T, I) \geq cost(T', I) \geq cost(T, I) - n \cdot 2^t$

$0 \leq cost(T, I) - cost(T', I) \leq n \cdot 2^t$

$\leq n \cdot \frac{\epsilon \cdot c_{max}}{(1+\epsilon) \cdot n}$

$= \epsilon \cdot \frac{c_{max}}{1+\epsilon}$

FPTAS für KP

Beweis-3

Damit zeigen wir:

$cost(T, I) \geq cost(T', I) \geq cost(T, I) - n \cdot 2^t$

$0 \leq cost(T, I) - cost(T', I) \leq n \cdot 2^t$

$\leq n \cdot \frac{\epsilon \cdot c_{max}}{(1+\epsilon) \cdot n}$

$= \epsilon \cdot \frac{c_{max}}{1+\epsilon}$

Somit können wir mit der Annahme, dass $cost(T, I) \geq c_{max}$ gilt erhalten, dass:

FPTAS für KP

Beweis-3

Damit zeigen wir:

$cost(T, I) \geq cost(T', I) \geq cost(T, I) - n \cdot 2^t$

$0 \leq cost(T, I) - cost(T', I) \leq n \cdot 2^t$

$\leq n \cdot \frac{\epsilon \cdot c_{max}}{(1+\epsilon) \cdot n}$

$= \epsilon \cdot \frac{c_{max}}{1+\epsilon}$

Somit können wir mit der Annahme, dass $cost(T, I) \geq c_{max}$ gilt erhalten, dass:

$cost(T', I) \geq c_{max} - \epsilon \cdot \frac{c_{max}}{1+\epsilon}$

FPTAS für KP

Beweis-4

Somit erhalten wir für die Approximationsgüte:

$R(I) =$

FPTAS für KP

Beweis-4

Somit erhalten wir für die Approximationsgüte:

$R(I) = \frac{cost(T,I)}{cost(T',I)}$

FPTAS für KP

Beweis-4

Somit erhalten wir für die Approximationsgüte:

$$R(I) = \frac{cost(T,I)}{cost(T',I)}$$

$$= \frac{cost(T',I) + cost(T,I) - cost(T',I)}{cost(T',I)}$$

FPTAS für KP

Beweis-4

Somit erhalten wir für die Approximationsgüte:

$$R(I) = \frac{cost(T,I)}{cost(T',I)}$$

$$= \frac{cost(T',I)+cost(T,I)-cost(T',I)}{cost(T',I)}$$

$$\leq 1 + \frac{\epsilon \cdot (c_{max}/(1+\epsilon))}{cost(T',I)}$$

FPTAS für KP

Beweis-4

Somit erhalten wir für die Approximationsgüte:

$$R(I) = \frac{cost(T,I)}{cost(T',I)}$$

$$= \frac{cost(T',I) + cost(T,I) - cost(T',I)}{cost(T',I)}$$

$$\leq 1 + \frac{\epsilon \cdot (c_{max}/(1+\epsilon))}{cost(T',I)}$$

$$\leq 1 + \frac{\epsilon \cdot (c_{max}/(1+\epsilon))}{c_{max} - \epsilon \cdot \epsilon \cdot (c_{max}/(1+\epsilon))}$$

FPTAS für KP

Beweis-4

Somit erhalten wir für die Approximationsgüte:

$$R(I) = \frac{cost(T,I)}{cost(T',I)}$$

$$= \frac{cost(T',I)+cost(T,I)-cost(T',I)}{cost(T',I)}$$

$$\leq 1 + \frac{\epsilon\cdot(c_{max}/(1+\epsilon))}{cost(T',I)}$$

$$\leq 1 + \frac{\epsilon\cdot(c_{max}/(1+\epsilon))}{c_{max}-\epsilon\cdot\epsilon\cdot(c_{max}/(1+\epsilon))}$$

$$= 1 + \frac{\epsilon}{1+\epsilon}\cdot\frac{1}{1-(\epsilon/(1+\epsilon))}$$

FPTAS für KP

Beweis-4

Somit erhalten wir für die Approximationsgüte:

$$R(I) = \frac{cost(T,I)}{cost(T',I)}$$

$$= \frac{cost(T',I) + cost(T,I) - cost(T',I)}{cost(T',I)}$$

$$\leq 1 + \frac{\epsilon \cdot (c_{max}/(1+\epsilon))}{cost(T',I)}$$

$$\leq 1 + \frac{\epsilon \cdot (c_{max}/(1+\epsilon))}{c_{max} - \epsilon \cdot \epsilon \cdot (c_{max}/(1+\epsilon))}$$

$$= 1 + \frac{\epsilon}{1+\epsilon} \cdot \frac{1}{1-(\epsilon/(1+\epsilon))}$$

$$= 1 + \frac{\epsilon}{1+\epsilon} \cdot (1+\epsilon) = 1 + \epsilon$$

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.
Schritt 3 läuft in der Zeit von $O(n \cdot Opt_{KP}(I'))$.

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.
Schritt 3 läuft in der Zeit von $O(n \cdot Opt_{KP}(I'))$.
$Opt_{KP}(I)$ kann wie folgt abgeschätzt werden:

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.
Schritt 3 läuft in der Zeit von $O(n \cdot Opt_{KP}(I'))$.
$Opt_{KP}(I)$ kann wie folgt abgeschätzt werden:
$Opt_{KP}(I')$

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.
Schritt 3 läuft in der Zeit von $O(n \cdot Opt_{KP}(I'))$.
$Opt_{KP}(I)$ kann wie folgt abgeschätzt werden:
$Opt_{KP}(I')$
$\leq \sum_{i=1}^{n} c_i' = \sum_{i=1}^{n} \left\lfloor c_i \cdot 2^{-\left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon)\cdot n} \right\rfloor} \right\rfloor$

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.
Schritt 3 läuft in der Zeit von $O(n \cdot Opt_{KP}(I'))$.
$Opt_{KP}(I)$ kann wie folgt abgeschätzt werden:
$Opt_{KP}(I')$
$\leq \sum_{i=1}^{n} c_i' = \sum_{i=1}^{n} \left\lfloor c_i \cdot 2^{-\left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon)\cdot n} \right\rfloor} \right\rfloor$
$\leq \sum_{i=1}^{n} (c_i \cdot 2 \cdot \frac{(1+\epsilon)\cdot n}{\epsilon \cdot c_{max}})$

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.
Schritt 3 läuft in der Zeit von $O(n \cdot Opt_{KP}(I'))$.
$Opt_{KP}(I)$ kann wie folgt abgeschätzt werden:

$$Opt_{KP}(I')$$
$$\leq \sum_{i=1}^{n} c_i' = \sum_{i=1}^{n} \left\lfloor c_i \cdot 2^{-\left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon)\cdot n} \right\rfloor} \right\rfloor$$
$$\leq \sum_{i=1}^{n} (c_i \cdot 2 \cdot \frac{(1+\epsilon)\cdot n}{\epsilon \cdot c_{max}})$$
$$= 2 \cdot (1+\epsilon) \cdot \epsilon^{-1} \cdot \frac{n}{c_{max}} \cdot \sum_{i=1}^{n} c_i$$

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.
Schritt 3 läuft in der Zeit von $O(n \cdot Opt_{KP}(I'))$.
$Opt_{KP}(I)$ kann wie folgt abgeschätzt werden:

$$
\begin{aligned}
&Opt_{KP}(I') \\
&\leq \sum_{i=1}^{n} c_i' = \sum_{i=1}^{n} \left\lfloor c_i \cdot 2^{-\left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon)\cdot n} \right\rfloor} \right\rfloor \\
&\leq \sum_{i=1}^{n} (c_i \cdot 2 \cdot \frac{(1+\epsilon)\cdot n}{\epsilon \cdot c_{max}}) \\
&= 2 \cdot (1+\epsilon) \cdot \epsilon^{-1} \cdot \frac{n}{c_{max}} \cdot \sum_{i=1}^{n} c_i \\
&\leq 2 \cdot (1+\epsilon) \cdot \epsilon^{-1} \cdot n^2 \in O(\epsilon^{-1} \cdot n^2)
\end{aligned}
$$

FPTAS für KP

Beweis-5

Für die Zeitkomplexität des Algorithmus ist beobachtbar, dass Schritt 1 und Schritt 2 in Zeit $O(n)$ laufen.
Schritt 3 läuft in der Zeit von $O(n \cdot Opt_{KP}(I'))$.
$Opt_{KP}(I)$ kann wie folgt abgeschätzt werden:

$$
\begin{aligned}
& Opt_{KP}(I') \\
& \leq \sum_{i=1}^{n} c'_i = \sum_{i=1}^{n} \left\lfloor c_i \cdot 2^{-\left\lfloor \log_2 \frac{\epsilon \cdot c_{max}}{(1+\epsilon)\cdot n} \right\rfloor} \right\rfloor \\
& \leq \sum_{i=1}^{n} (c_i \cdot 2 \cdot \frac{(1+\epsilon)\cdot n}{\epsilon \cdot c_{max}}) \\
& = 2 \cdot (1+\epsilon) \cdot \epsilon^{-1} \cdot \frac{n}{c_{max}} \cdot \sum_{i=1}^{n} c_i \\
& \leq 2 \cdot (1+\epsilon) \cdot \epsilon^{-1} \cdot n^2 \in O(\epsilon^{-1} \cdot n^2)
\end{aligned}
$$

Algorithmus 4 läuft in $O(\epsilon^{-1} \cdot n^3)$ und ist somit ein FPTAS.

Conclusion

Verwendung des KP

- Verschlüsselungsalgorithmen
- Bankräuber
- Transportfirmen
- uvm.

Vielen Dank für eure Aufmerksamkeit

noch Fragen?