

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Welche Probleme sind beispielsweise NP-vollständig?

- SAT/3SAT
- Minimales Vertex Cover
- Traveling Salesman Problem
- Shortest Superstring Problem
- Set Cover Problem
- Bin Packing Problem
- Rucksackproblem
- Hamiltonkreisproblem

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Was ist ein Approximationsalgorithmus und welche Definitionen (für ein Minimierungsproblem) gibt es dazu?

Ein polynomieller Algorithmus  $\mathcal{A}$  ist ein  $\delta$ -Approximationsalgorithmus, wenn für jede Problem Instanz  $I$  mit Optimalwert  $\text{OPT}(I)$

$$R_{\mathcal{A}}(I) = \frac{\mathcal{A}(I)}{\text{OPT}(I)} \leq \delta \quad \text{mit } \delta \geq 1$$

gilt.  $\delta$  ist die Gütegarantie (Approximationsfaktor).

Die absolute Gütegarantie (Performance) ist

$$R_{\mathcal{A}}^{\text{abs}}(I) = \inf\{r \geq 1 \mid R_{\mathcal{A}}(I) \leq r \text{ für alle Instanzen } I\}.$$

Die asymptotische Gütegarantie ist

$$R_{\mathcal{A}}^{\infty}(I) = \inf\{r \geq 1 \mid R_{\mathcal{A}}(I) \leq r \text{ für } \text{OPT}(I) \rightarrow \infty\}.$$

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Was versteht man unter dem „Set Cover“-Problem?

Gegeben sei eine Menge  $U = \{1, 2, \dots, n\}$  mit  $n$  Elementen, eine Familie  $\mathcal{S} = \{S_1, \dots, S_k\}$  von Teilmengen von  $U$  und eine Kostenfunktion  $c : \mathcal{S} \rightarrow \mathbb{Q}_+$ .

Gesucht ist eine Teilmenge  $S \subseteq \{1, \dots, k\}$  mit

$$\bigcup_{i \in S} S_i = U$$

so, dass  $\sum_{i \in S} c(S_i)$  minimal ist.

Unter der *Frequenz* eines Elementes versteht man die Anzahl an Mengen, in denen ein Element enthalten ist.  $f$  ist die Frequenz des am häufigsten auftretenden Elementes.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Wie lässt sich das „Vertex Cover“-Problem als „Set Cover“-Problem darstellen?

Sei  $U := E$  die Menge aller Kanten. Eine Teilmenge  $S_i$  von  $U$  besteht aus allen Kanten, die zu einem Knoten  $v_i$  inzident sind, also

$$S_i := \{e \in E \mid \{v_i, w\} \in E, w \in V\}.$$

Da jede Kante mit genau zwei Knoten verbunden ist, gilt für die Frequenz einer Kante  $f = 2$ .

Für dieses Problem existiert ein 2-Approximationsalgorithmus, nämlich alle zu einem maximalen Matching inzidenten Knoten.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Wie lautet der „Greedy Set Cover“-Algorithmus?

Sei  $C$  die Teilmenge, die zum Beginn der Iteration bereits überdeckt ist. Die *Kosteneffektivität* einer Menge  $S$  seien die durchschnittlichen Kosten, mit denen ein neues Element überlagert wird, also zum Beispiel  $\frac{c(S)}{|S-C|}$ . Der *Preis* eines Elementes sind die durchschnittlichen Kosten mit denen es überlagert wird.

---

**Algorithmus 1** : Greedy Set Cover

---

- 1  $C \leftarrow \emptyset$ ;
  - 2 **while**  $C \neq U$  **do**
    - Finde  $S$ , die kosteneffektivste Menge in der aktuellen Iteration;
    - Sei  $\alpha = \frac{c(S)}{|S-C|}$  die Kosteneffektivität von  $S$ ;
    - Wähle  $S$  aus und für alle  $e \in S - C$  setze  $\text{price}(e) = \alpha$ ;
    - $C \leftarrow C \cup S$ ;
  - 3 Gebe ausgewählte Mengen aus;
-

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Wie gut ist die Approximationsgüte des „Greedy Set Cover“-Algorithmus?

Für jedes  $k \in \{1, \dots, n\}$  gilt

$$\text{Preis}(e_k) \leq \frac{\text{OPT}}{n - k + 1}.$$

Der Greedy-Algorithmus für das „Minimum Set Cover“-Problem ist ein  $H_n$ -Approximationsalgorithmus, mit  $H_n := 1 + \frac{1}{2} + \dots + \frac{1}{n}$ . Er liefert also eine  $\mathcal{O}(\log n)$ -Approximation.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Welche Idee steckt hinter dem „Layering“-Prinzip?

Die Idee beim „Layering“ ist, eine gegebene Gewichtsfunktion auf den Knoten in geeignete Funktionen (gradgewichtete) auf einer verschachtelten Sequenzen von Teilgraphen zu zerlegen.

Sei  $w : V \rightarrow \mathbb{Q}_+$  die Funktion, die jedem Knoten ein Gewicht zuordnet. Diese Funktion ist gradgewichtet, wenn es eine Konstante  $c > 0$  so gibt, dass das Gewicht eines jeden Knotens  $v \in V$  gerade  $c \cdot \deg(v)$  ist.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Wie lautet der „Layering“-Algorithmus?

---

**Algorithmus 2** : Layering

---

- 1  $G_0 \leftarrow G; k \leftarrow 0;$
  - 2 **while** *es existiert*  $v \in G_k$  mit  $\deg(v) = 0$  **do**
    - $t(v) = c \cdot \deg(v)$  mit  $c = \min \left\{ \frac{w(v)}{\deg(v)} \right\}$  für alle  $v \in G_k;$
    - $w'(v) = w(v) - t(v)$  für alle  $v \in G_k;$
    - $D_k \leftarrow$  alle Knoten mit  $\deg(v) = 0$  aus  $G_k;$
    - $W_k \leftarrow$  alle Knoten mit  $w'(v) = 0;$
    - $k \leftarrow k + 1;$
    - $G_k = V - (D_{k-1} \cup W_{k-1});$
  - 3 **return**  $(C = W_0 \cup \dots \cup W_{k-1});$
- 

Hierbei ist  $t(v)$  die größte gradgewichtete Funktion in  $w$  und  $w'(v)$  die Residualgewichtsfunktion.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Was versteht man unter dem „Shortest Superstring“-Problem?

Gegeben sei ein endliches Alphabet  $\Sigma$  und eine Menge  $S = \{s_1, \dots, s_n\} \subseteq \Sigma_+$  von  $n$  Strings.

Gesucht ist ein kürzester String  $s$ , der jedes  $s_i$  als Teilstring enthält. Es wird angenommen, dass kein String  $s_i$  einen anderen String  $s_j$  ( $i \neq j$ ) als Teilstring enthält.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Wie lautet der „Shortest Superstring via Set Cover“-Algorithmus?

---

**Algorithmus 3** : Shortest Superstring via Set Cover

---

- 1 Benutze den *Greedy Set Cover Algorithmus* um ein Cover für die Instanz  $\mathcal{S}$  zu bekommen;  
Seien  $\text{set}(\pi_1), \dots, \text{set}(\pi_k)$  die durch dieses Cover ausgewählten Mengen;
  - 2 Verbinde die Strings  $\pi_1, \dots, \pi_k$  in beliebiger Reihenfolge;
  - 3 Gebe resultierenden String  $s$  aus;
- 

Die Instanz  $\mathcal{S}$  wird folgendermaßen konstruiert. Für alle  $s_i, s_j \in S$  und  $k > 0$  sei  $\sigma_{ijk}$  der String, den man erhält, wenn man  $s_i$  und  $s_j$  um  $k$  Stellen überlappen lässt. Für einen String  $\pi \in \Sigma_+$  sei  $\text{set}(\pi) = \{s \in S \mid s \text{ ist Teilstring von } \pi\}$ .  $S$  sei  $S$  und die Teilmengen sind  $\text{set}(\pi)$  für jeden String  $\pi \in S \cup I$  mit Kosten von  $|\pi|$ , der Länge von  $\pi$ .

# APPROXIMATIONSLGORITHMEN

*Prof. Dr. Fekete*

---

Was versteht man unter „PTAS“ und „FPTAS“?

Eine Familie  $\{\mathcal{A}_\varepsilon\}$  von Algorithmen ist ein *Approximationsschema* für ein NP-schweres Problem  $\Pi$ , wenn die Eingabe  $(I, \varepsilon)$  eine Lösung  $s$  liefert, sodass

- $f_\Pi(I, s) \leq (1 + \varepsilon) \cdot \text{OPT}$ , falls  $\Pi$  ein Minimierungsproblem ist,
- $f_\Pi(I, s) \geq (1 - \varepsilon) \cdot \text{OPT}$ , falls  $\Pi$  ein Maximierungsproblem ist,

gilt. Hierbei ist  $I$  eine Instanz von  $\Pi$  und  $\varepsilon > 0$  ein Fehlertherm.

$\mathcal{A}_\varepsilon$  ist ein *polynomielles Approximationsschema (PTAS)*, wenn für jedes feste  $\varepsilon > 0$  die Laufzeit polynomiell durch die Instanzgröße  $I$  begrenzt wird, also  $\mathcal{A}_\varepsilon$  ein  $(1 + \varepsilon)$ -Approximationsalgorithmus für  $\Pi$  ist.

Ist zusätzlich die Laufzeit polynomiell in  $\frac{1}{\varepsilon}$ , dann ist  $\mathcal{A}_\varepsilon$  ein *voll polynomielles Approximationsschema (FPTAS)*.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Was versteht man unter dem „Rucksack“-Problem?

Gegeben sei eine Menge  $S = \{a_1, \dots, a_n\}$  von Objekten mit festgelegter Größe  $\text{size}(a_i) \in \mathbb{Z}_+$  und festgelegtem Nutzen  $\text{profit}(a_i) \in \mathbb{Z}_+$  und die Rucksackkapazität  $B \in \mathbb{Z}_+$ .

Gesucht ist eine Teilmenge von Objekten, deren Gesamtgröße durch  $B$  beschränkt ist und deren Nutzen maximiert wird.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Was versteht man unter einem „Pseudo-Polynomial Time Algorithm“?

Die Instanzgröße  $|I|$  ist die Anzahl der benötigten Bits, um  $I$  zu kodieren.  $I_u$  kennzeichnet die Instanz  $I$ , bei der alle Zahlen unär codiert sind. Die unäre Instanzgröße  $|I_u|$  von  $I$  ist die Anzahl der benötigten Bytes, um  $I_u$  zu kodieren.

Ein Algorithmus für ein Problem  $\Pi$  wird *Pseudo-Polynomial Time Algorithm* genannt, wenn die Laufzeit der Instanz  $I$  polynomiell in  $|I_u|$  begrenzt wird.

Das Rucksack-Problem besitzt einen Pseudo-Polynomial Time Algorithm, basierend auf dynamischer Optimierung.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Wie lautet der „FPTAS für Rucksack“-Algorithmus?

---

**Algorithmus 4** : FPTAS für Rucksack

---

- 1 Gegeben sei  $\varepsilon > 0$ ; Sei  $K = \frac{\varepsilon P}{n}$ ;
  - 2 Für jedes Objekt  $a_i$  definiere  $\text{profit}'(a_i) = \left\lfloor \frac{\text{profit}(a_i)}{K} \right\rfloor$ ;
  - 3 Mit diesen Objektnutzen sucht man die nützlichste Menge  $S'$  mit dem *Dynamic Programming Algorithmus*;
  - 4 Gebe  $S'$  aus;
- 

Dieser Algorithmus ist ein Fully Polynomial Time Approximation Scheme (FPTAS) für das Rucksack-Problem, da  $\text{profit}(S') \geq (1 - \varepsilon) \cdot \text{OPT}$  ist und die Laufzeit durch

$$\mathcal{O} \left( n^2 \left\lfloor \frac{P}{K} \right\rfloor \right) = \mathcal{O} \left( n^2 \left\lfloor \frac{n}{\varepsilon} \right\rfloor \right)$$

begrenzt ist, was polynomiell in  $n$  und  $\frac{1}{\varepsilon}$  ist.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Was lässt sich im Zusammenhang mit stark NP-schweren Problemen zur Existenz eines FPTAS sagen?

Ein stark NP-schweres Optimierungsproblem kann keinen pseudo-polynomiellen Algorithmus besitzen, wenn  $P \neq NP$  ist.

Sei  $p$  ein Polynom und  $\Pi$  ein NP-schweres Minimierungsproblem so, dass der Zielfunktionswert  $f_{\Pi}$  ein ganzzahliger Wert ist und für jede Instanz  $I$   $OPT(I) < p(|I_u|)$  gilt.

- Besitzt  $\Pi$  ein FPTAS, dann besitzt es auch einen pseudo-polynomiellen Algorithmus.
- Ist  $\Pi$  stark NP-schwer, so besitzt  $\Pi$  kein FPTAS, wenn  $P \neq NP$  ist.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Was versteht man unter dem „Bin Packing“-Problem?

Gegeben seien  $n$  Objekte mit Größe  $a_1, \dots, a_n \in (0, 1]$ .

Gesucht ist eine Packung in Behälter der Größe eins mit minimaler Anzahl an benutzten Behältern.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Wie lautet der „First-Fit“-Algorithmus?

---

## Algorithmus 5 : First-Fit

---

1  $B_1 \leftarrow \emptyset; k \leftarrow 1;$

2 **for**  $i = 1, \dots, n$  **do**

    Versuche  $a_i$  in ein zum Teil gefülltes  $B_j$  ( $j = 1, \dots, k$ ) zu packen;

**if** *es existiert kein*  $B_j$  ( $j = 1, \dots, k$ ), *in das*  $a_i$  *hineinpasst* **then**

        └ Erstelle ein neues  $B_{k+1}$  und packe  $a_i$  hinein;

---

Dies ist ein 2-Approximationsalgorithmus, denn von den  $m$  benutzten Behältern ist höchstens eins nicht mindestens halbvoll.  $m \leq 2 \cdot \text{OPT}$ .

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Was versteht man unter „Asymptotic PTAS“?

Eine Familie von Algorithmen  $\{\mathcal{A}_\varepsilon\}$  bildet ein *asymptotisches polynomielles Approximationsschema (Asymptotic PTAS)*, wenn für jedes  $\varepsilon > 0$  ein  $N > 0$  und ein polynomieller Algorithmus  $\mathcal{B} \in \{\mathcal{A}_\varepsilon\}$  so existieren, dass  $\mathcal{B}$  eine Approximationsgüte von  $1 + \varepsilon$  für alle Instanzen mit  $\text{OPT} \geq N$  besitzt.

# APPROXIMATIONSALGORITHMEN

*Prof. Dr. Fekete*

---

Wie lautet der „ $\mathcal{A}_\epsilon$  für Bin Packing“-Algorithmus?

---

## Algorithmus 6 : $\mathcal{A}_\varepsilon$ für Bin Packing

---

- 1 Entferne alle Objekte der Größe  $< \varepsilon$ ;
  - 2 Runde, um eine konstante Zahl von Objektgrößen zu erhalten (Lemma 9.5);
  - 3 Finde optimale Packung (Lemma 9.4);
  - 4 Wende diese Packung auf die Originalgröße der Objekte an;
  - 5 Packe Objekte der Größe  $< \varepsilon$  mit *First-Fit*;
- 

Für jedes  $\varepsilon$ ,  $0 < \varepsilon \leq \frac{1}{2}$ , existiert ein Algorithmus  $\mathcal{A}_\varepsilon$ , der polynomiell in  $n$  ist und maximal  $(1 + 2\varepsilon) \cdot \text{OPT} + 1$  Behälter für ein Packung benutzt.