

APPROXIMATIVE ALGORITHMEN

”WIE GENAU IST UNGEFÄHR?”

Robert Bahmann

Hochschule **RheinMain** University of Applied Sciences
Wiesbaden Rüsselsheim Geisenheim

July 23, 2009

1 EINLEITUNG

2 GRUNDLEGENDE EIGENSCHAFTEN

- Güte/Performance
- Approximationsklassen

3 BEISPIEL

- JS Algorithmus
- Analyse
- Ergebnis

4 FRAGEN

Viele Probleme sind nicht effizient exakt lösbar
solange $\mathbb{P} \neq \mathbb{NP}$

WARUM APPROXIMATIONSALGORITHMEN?

- arbeiten in polynomieller Zeit
- produzieren Lösungen mit *garantierter* Güte

GÜTE/PERFORMANCE

ABSOLUT $\mathcal{AG}_A(\mathcal{I}) = |A(\mathcal{I}) - \text{OPT}(\mathcal{I})|$

RELATIV $\mathcal{RG}_A(\mathcal{I}) = \max \left\{ \frac{A(\mathcal{I})}{\text{OPT}(\mathcal{I})}, \frac{\text{OPT}(\mathcal{I})}{A(\mathcal{I})} \right\}$

APX

Approximable - Gut approximierbare Probleme.

PTAS

Polynomial Time Approximation Scheme

Beliebig gut Approximierbar auf Kosten des Fehlers.

FPTAS

Fully Polynomial Time Approximation Scheme

Beliebig gut und schnell Approximierbar auf Kosten des Fehlers un der Laufzeit.

JOB SCHEDULING

Auf p Prozessoren (P_1, P_2, \dots, P_n) sollen t Tasks (T_1, T_2, \dots, T_i) verteilt werden. Dabei hat jeder Task eine Laufzeit l_i die der Prozessor benötigt, um diesen abzuarbeiten.

Auf einem Rechner mit einem Dual-Core müssen folgende Tasks abgearbeitet werden:

TASK	ZEIT
1	1
2	2
3	2
4	4
5	1

JOB SCHEDULING - BEISPIEL

- Core 1: $(0, P_1), (1, P_3), (3, P_2)$
- Core 2: $(0, P_4), (4, P_5)$



Algorithmus 1 : JOB SCHEDULE

Input : p Prozessoren, t Taks und die Laufzeiten l_i für jeden Task

Output : Eine Prozessorbelegung S mit minimalen Makespan

$S = \text{null}$;

for $i = 1$ to m **do**

$a_i = \text{null}$;

for $k = 1$ to n **do**

 Berechne das kleinste i mit $a_i = \min\{a_j; j \in \{1, \dots, m\}\}$;

$S_k = a_i$;

$a_i = a_i + p_k$;

$S = S \cup \{(S_k, M_i)\}$

return S ;

INHALT

EINLEITUNG

GRUNDLEGENDE
EIGENSCHAFTENGÜTE/PERFORMANCE
APPROXIMATIONSKLASS

BEISPIEL

JS ALGORITHMUS

ANALYSE

ERGEBNIS

FRAGEN

ALGORITHMUSAUSGABE

- Core 1: $(0, P_1), (1, P_3), (4, P_5)$
- Core 2: $(0, P_2), (2, P_4)$

ALGORITHMUSAUSGABE

- Core 1: $(0, P_1), (1, P_3), (4, P_5)$
- Core 2: $(0, P_2), (2, P_4)$



- Sei $Z_k = s_k + l_k$ die Zeit in der der k -te Task endet.

- Sei $Z_k = s_k + l_k$ die Zeit in der der k -te Task endet.
- Sei T_{last} der Task der als letztes beendet wird.

- Sei $Z_k = s_k + l_k$ die Zeit in der der k -te Task endet.
- Sei T_{last} der Task der als letztes beendet wird.
- Dann ist die Gesamtlaufzeit $Z_{gesamt} = s_{last} + l_{last}$

- Sei $Z_k = s_k + l_k$ die Zeit in der der k -te Task endet.
- Sei T_{last} der Task der als letztes beendet wird.
- Dann ist die Gesamtlaufzeit $Z_{gesamt} = s_{last} + l_{last}$

- Sei $Z_k = s_k + l_k$ die Zeit in der der k -te Task endet.
- Sei T_{last} der Task der als letztes beendet wird.
- Dann ist die Gesamtlaufzeit $Z_{gesamt} = s_{last} + l_{last}$



- 1 Sei $Z_k = s_k + l_k$ die Zeit in der der k -te Task endet.
- 2 Sei T_{last} der Task der als letztes beendet wird.
- 3 Dann ist die Gesamtlaufzeit $Z_{gesamt} = s_{last} + l_{last}$
- 4 Damit ist die durchschnittliche Laufzeit bis zum Start des letzten Jobs höchstens: $\frac{1}{n} \sum_{k \in \{1, \dots, i\} \setminus \{last\}} l_k$

- 1 Sei $Z_k = s_k + l_k$ die Zeit in der der k -te Task endet.
- 2 Sei T_{last} der Task der als letztes beendet wird.
- 3 Dann ist die Gesamtlaufzeit $Z_{gesamt} = s_{last} + l_{last}$
- 4 Damit ist die durchschnittliche Laufzeit bis zum Start des letzten Jobs höchstens: $\frac{1}{n} \sum_{k \in \{1, \dots, i\} \setminus \{last\}} l_k$
- 5 Es existiert also mindestens ein Prozessor dessen Gesamtlaufzeit **höchstens** von dieser Größe ist, also:
$$s_{last} \leq \frac{1}{n} \sum_{k \in \{1, \dots, i\} \setminus \{last\}} l_k$$

- 1 $S_{last} \leq \frac{1}{n} \sum_{k \in \{1, \dots, i\} \setminus \{last\}} l_k$
- 2 offensichtlich gilt: $OPT \geq l_{last}$
- 3 ebenso: $OPT \geq \frac{1}{n} \sum_{k=1}^i l_k$

$$Z_{\text{gesamt}} = Z_{\text{last}} \quad (1)$$

$$= S_{\text{last}} + I_{\text{last}} \quad (2)$$

$$\leq \frac{1}{n} \sum_{k \in \{1, \dots, i\} \setminus \{\text{last}\}} l_k + I_{\text{last}} \quad (3)$$

$$= \frac{1}{n} \sum_{k=1}^i l_k + \left(1 - \frac{1}{n}\right) I_{\text{last}} \quad (4)$$

$$\leq OPT + \left(1 - \frac{1}{n}\right) * OPT \quad (5)$$

$$= \left(2 - \frac{1}{n}\right) * OPT \quad (6)$$

UND JETZT?

- Wir haben gerade gezeigt das LIST SCHEDULE eine relative Güte von $(2 - \frac{1}{n})$ hat. (juhu!)
- Kann als Basis genutzt werden um LIST SCHEDULE zu verbessern
 - Eingabe komplett erfassen und Sortieren
 - Alternativ nur vorhandene Eingabe sortieren und den rest nach altem Verfahren sortieren usw.

FRAGEN?

Fragen?

QUELLEN

Diese Präsentation beruht auf der Ausarbeitung im Rahmen dieses Fachseminars. Sämtliche Quellen sind dort zu finden.