# Das Lastverteilungsproblem

## Approximationsalgorithmen


| | |
|---|---|
| Referent | Franz Brauße |
| Veranstaltung | Proseminar Theoretische Informatik |
| | Universität Trier, FB IV |
| Dozent | Prof. Dr. Henning Fernau |


23.02.2012

# Übersicht

# Übersicht

# Approximation

- exakte Lösung für $\mathbf{NP}$-vollständige Probleme oft zeitraubend
- vielleicht reicht eine angenäherte Lösung
  - Algorithmus $f$, $x \mapsto y$. Eine Optimallösung mag $y^*$ sein
- $\rightsquigarrow$ Gütebegriff für Lösung und Algorithmus
  - Güte der Lsg. durch Metrik $d(\cdot,\cdot) \in \mathbb{R}_{\geq}$ auf Lösungsraum, bspw. induziert durch Norm $\|\cdot\|$

Übliche Gütefunktion sind

- $v(y) = |y|$ für $y$ skalar.
- $v(y) = \|y\|_2 = \sqrt{\sum y_i^2}$ für $y$ in Vektorraum $B^k$.
- $v(y) = \|y\|_\infty = \max_i |y_i|$ für $y \in B^k$

## Güte von Approximationsalgorithmen

Güte der Lösung $v(y)$ in Relation zu Güte einer Optimallösung $v^* = v(y^*)$.

Mehrere Definitionsmöglichkeiten, teils je nach Problemart:

Minimierung $\frac{v(y)}{v^*} \geq 1$

Maximierung $\frac{v^*}{v(y)} \geq 1$

, auch üblich: $\max\left\{\frac{v^*}{v(y)}, \frac{v(y)}{v^*}\right\} \geq 1$

Güte $= 1 \iff$ Optimallösung erreicht

# Komplexitätsklassen

Klassifikation von Optimierungsalgorithmen $f$, die eine Approximation in polynomieller Zeit liefern:

**APX** *approximable*
Es gibt eine Näherungsschranke $\delta > 1$, die $f$ für jede Eingabe einhält $\rightsquigarrow \mathcal{O}(n^c)$, $c \gg 1$.

**PTAS** *polynomial time approximation scheme*
Für jede Näherungsschranke $\delta = 1 + \varepsilon$ lässt sich ein $f_\delta$ angeben, dass $\delta$ für jede Eingabe einhält $\rightsquigarrow \mathcal{O}(n^{1/\varepsilon})$.

**FPTAS** *fully polynomial time approximation scheme*
Für jedes $k \in \mathbb{N}$ liefert $f$ eine Näherung der Güte $\delta \leq 1 + \frac{1}{k}$ und ist polynomiell in Eingabelänge $n$ und $k$ $\rightsquigarrow \mathcal{O}(k^c n^d)$.

$$\mathbf{FPTAS} \subseteq \mathbf{PTAS} \subseteq \mathbf{APX}$$

Weitere Klassen in folgendem Vortrag.

# Übersicht

# Problem

- $m$ verschiedene Maschinen $M_1, \ldots, M_m$
- $n$ verschiedene Aufgaben zu erfüllen
- jede Aufgabe $j$ braucht bestimmte Zeit $t_j$ (gleich auf allen $M_i$)

## Beispiele

- Webserver:
  - HTTP + Dynamische Inhalte
  - FTP
- Ämter, Post, Friseur, ... (nur am Ende der Öffnungszeit)
- SMP-Betriebssysteme: Load-Balancer als Teil des Schedulers

Allgemein dort, wo $m$ vorhandene, nicht verbrauchbare Resourcen zeitweise allokiert werden.

$\rightsquigarrow$ Aufteilung sodass alle Maschinen gleichmäßig ausgelastet sind(?)

## Load-Balancing als Entscheidungsproblem

Gegeben die Anzahl der Resourcen $m \in \mathbb{N}$, die Zeiten $t_1, \ldots, t_n$ der Aufträge und ein Zeitlimit $T$.

Gibt es eine Verteilung der Aufträge, sodass die maximale Wartezeit $\hat{T} := \max_i T_i \leq T$ ?

Gesamtzeit für $M_i$: $T_i := \sum_{\text{Job } j \text{ in } M_i} t_j$

In dieser Allgemeinheit ist Load-Balancing $\mathbf{NP}$-vollständig:

- Reduktion von PARTITION: $m = 2$, $t_j = s_j$, $T = \frac{1}{2} \sum_{j=1}^{n} s_j$.

# Passendes Optimierungsproblem?

**Gegeben** #Maschinen $m \in \mathbb{N}$, Zeiten $t_j$ $(j = 1, \ldots, n)$

**Gesucht** Aufteilung Jobs $j$ auf Maschinen $i \rightsquigarrow$ Zeiten $T_i$

**Ziel** Minimiere $v(y)$, $y = (T_i)_i$

Wie *gut* ist eine Lösung für das Problem?

- gleichmäßige Auslastung: $\overline{T} = \frac{1}{m} \sum_j t_j$

$$v(y) = \max_{i=1,\ldots,m} |\overline{T} - T_i| \quad \stackrel{!}{\to} \quad 0$$

- kürzeste Gesamtzeit

$$v(y) = \hat{T} = \max_{i=1,\ldots,m} T_i \quad \stackrel{!}{\to} \quad \overline{T}$$

Gütefunktion $v$ hat Auswirkungen auf Faktor bei Approximation.

## Güte

$$w(y) = \max_i |\overline{T} - T_i| \qquad \text{statt} \qquad v(y) = \max_i T_i$$

im speziellen Fall $n = m$, $t_1 \gg t_2 = \ldots = t_m$:

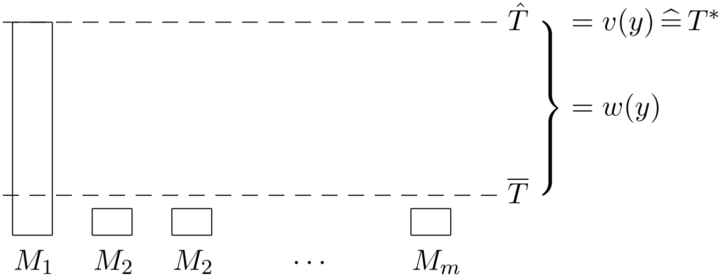



Abbildung: $\hat{T}$ versus $\overline{T}$

## LB als Optimierungsproblem

**Eingabe** $m$, $t_1, \ldots, t_n$.

**Ausgabe** $T_i$ für $i = 1, \ldots, m$, sodass $\hat{T}$ möglichst klein ist.

Triviale Einschränkungen für Optimalwert $T^*$:

- Nicht mehr als alle Elemente auf eine Maschine aufgeteilt.
- Auch größtes Elements muss (am Stück) bearbeitet werden.

$$\max_j t_j \quad \leq \quad T^* \quad \leq \quad \sum_{j=1}^{n} t_j$$

# Übersicht

# Greedy-Algorithmus

Approximation:

- benutze Job-Reihenfolge wie gegeben (Online-Verfahren)
- Zielmaschine ist die mit bisher kleinster Warteschlange

```
for i = 1 to m do
    T_i ← 0
    A_i ← ∅                          /* assigned Jobs */
end

for j = 1 to n do
    i ∈ argmin_{k=1,...,m} T_k       /* lowest workload so far */
    A_i ← A_i ∪ {j}                  /* assign job j to M_i */
    T_i ← T_i + t_j
end
```

Algorithmus macht größten Fehler, wenn

$$t_1 = \ldots = t_{n-1} = 1 \;\ll\; t_n,$$

dann:

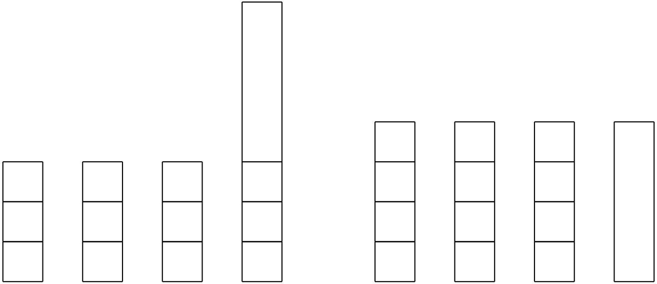


Abbildung: Links: Greedy ($\hat{T} = 7$); rechts: optimal ($T^* = 4$).

# Abschätzung max. Zeit $\hat{T}$ bzgl. $T^*$

Bekannt:

$$t_j \leq T^* \qquad j = 1, \ldots, n$$

Außerdem müssen die $m$ Maschinen auch wirklich die gesamte Arbeit erledigen, d.h.

$$\frac{1}{m} \sum_j t_j \leq T^*$$

# Greedy-Algorithmus hat Güte 2

## Beweis.

Betrachte Maschine $M_i$ mit längster Dauer $T_i = \hat{T}$ und ihren letzten Job $j$.

- $M_i$ hatte vorher minimale Last $T_i - t_j$

$$\implies \quad \sum_k T_k \geq m(T_i - t_j)$$

$$\iff \quad T_i - t_j \leq \frac{1}{m} \sum_k T_k \leq T^*$$

- Dann bekommt $M_i$ letzten Job $j$ zugewiesen, also

$$t_j \leq T^* \quad \implies \quad T_i = (T_i - t_j) + t_j \leq T^* + T^* = 2T^*$$

$M_i$ braucht maximale Zeit $\hat{T} \leq 2T^*$, also Güte 2. □

# Übersicht

# Sorted-Balance

- „Ausreißer“ störend $\rightsquigarrow$

**Eingabe sortieren**

$$t_1 \geq t_2 \geq t_3 \geq \ldots \geq t_{n-1} \geq t_n$$

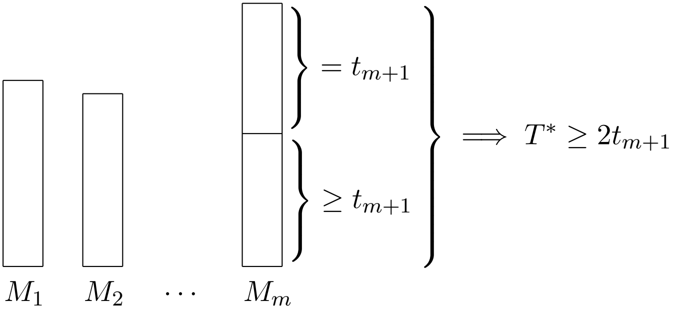
- anschließend gleicher Algorithmus:

```
for i = 1 to m do
    T_i ← 0
    A_i ← ∅                          /* assigned Jobs */
end

for j = 1 to n do
    i ∈ argmin_{k=1,...,m} T_k       /* lowest workload so far */
    A_i ← A_i ∪ {j}                  /* assign job j to M_i */
    T_i ← T_i + t_j
end
```

# Neue untere Schranke für $T^*$

Betrachte die $m+1$ ersten sortierten Jobs:



# Sorted-Balance hat Güte 1,5

## Beweis.

- $n \leq m \implies$ alle Jobs auf eigener Maschine, $\hat{T} = T^*$.
- $n > m \implies$ Maschine $M_i$ mit $T_i = \hat{T}$.
  - $M_i$ hat nur einen Job $\implies$ optimal, $\hat{T} = T^*$.
  - sonst wie im letzten Beweis sei $j$ der letzte $M_i$ zugewiesene Job und genauso

$$T_i - t_j \leq T^* \quad \text{und} \quad t_j \leq t_{m+1} \leq \tfrac{1}{2}T^*$$

damit

$$T_i = (T_i - t_j) + t_j \leq T^* + \tfrac{1}{2}T^*$$

Absteigende Sortierung $\rightsquigarrow \hat{T} \leq \frac{3}{2}T^*$. □