Algorithmen und Datenstrukturen

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Tanya Braun (Übungen)
sowie viele Tutoren

Hm... NP-schwere Probleme sind trickreich

- Was machen wir, wenn Heuristiken nicht greifen und unserer Algorithmus sich im Backtracking verirrt?
- Approximation der Lösung?
  - Vielleicht in nebenläufiger Berechnung?
  - Wenn die Bestimmung der optimalen Lösung zu lange dauert, nimm approximative Lösung, sofern sie bereitsteht
- Wir haben allerdings gesehen, dass Approximation manchmal beliebig schlecht sein kann (unbegrenztes Rucksackproblem)
- Können wir bei Approximation Garantien für die Güte geben?

Danksagung

Die nachfolgenden Präsentationen wurden mit ausdrücklicher Erlaubnis des Autors mit nur kleinen Änderungen übernommen aus:

- „Effiziente Algorithmen und Datenstrukturen“ (Kapitel 12 Approximation) gehalten von Christian Scheideler an der TUM http://www14.in.tum.de/lehre/2008WS/ea/index.html.de

Approximationsalgorithmen

Frage: Ich will ein NP-schweres Problem lösen. Was muss ich tun?

Antwort: Polynomialzeitalgorithmus dafür wohl nicht möglich.

Annahme:

Entwurfsmuster zur Aufwandsreduktion (siehe SAT)
bzw. Reduktion auf SAT (o.ä.) nicht offensichtlich,

Eine der drei Eigenschaften muss aufgegeben werden:

- Löse das Problem optimal.
- Löse das Problem in polynomieller Zeit
- Löse beliebige Instanzen des Problems

ρ-Approximationsalgorithmus:

- Läuft in polynomieller Zeit.
- Löst beliebige Instanzen des Problems.
- Findet eine Lösung, die höchstens Faktor ρ weg von Optimum ist.

Herausforderung: Lösung sollte möglichst nah an Optimum sein.

Lastbalancierung

Eingabe: m identische Maschinen, n Jobs. Job i hat Laufzeit t_i.

Einschränkungen:

- Ein einmal ausgeführter Job muss bis zum Ende auf derselben Maschine ausgeführt werden.
- Jede Maschine kann höchstens einen Job gleichzeitig bearbeiten.

Definition: Sei $J(i)$ die Teilmenge der Jobs, die Maschine i zugewiesen werden. Dann ist $L_i = \sum_{j \in J(i)} t_j$ die Last der Maschine i.

Definition: Der Makespan L ist die maximale Last einer Maschine, d.h. $L = \max_i L_i$

Lastbalancierung: finde Zuweisung, die Makespan minimiert

Lastbalancierung: List Scheduling

List-Scheduling Algorithmus:

- Betrachte n Jobs in einer festen Reihenfolge
- Weise Job j der Maschine mit z.Zt. geringster Last zu

List-Scheduling$(m, n, (t_1,\ldots,t_n))$:

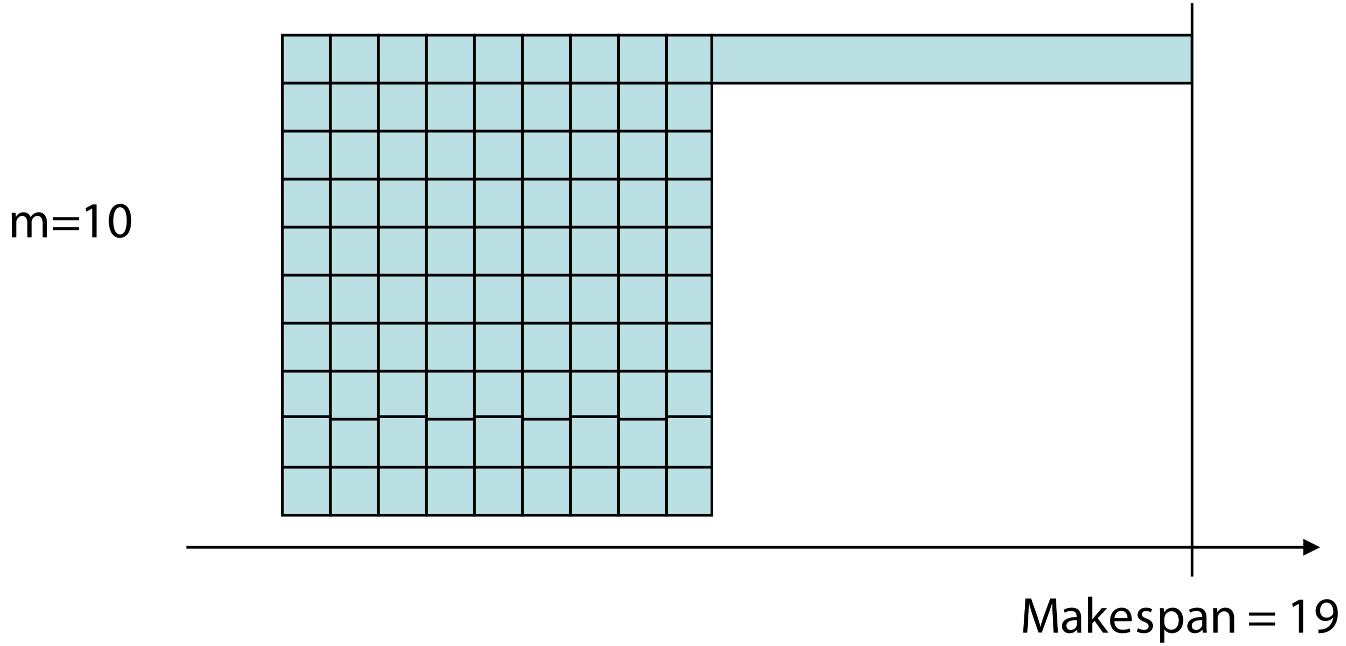
for $i:=1$ to m do
  $L_i := 0; J(i):=\emptyset$
for $j:=1$ to n do
  $i:=\mathrm{argmin}_{k \in [1..m]} L_k$
  $J(i):=J(i) \cup \{j\}$
  $L_i:=L_i + t_j$
return $(J(1),\ldots,J(m))$

Laufzeit: $O(n \log m)$ mit Priority Queue

Lastbalancierung: List Scheduling

Ist die Analyse scharf? Ja!

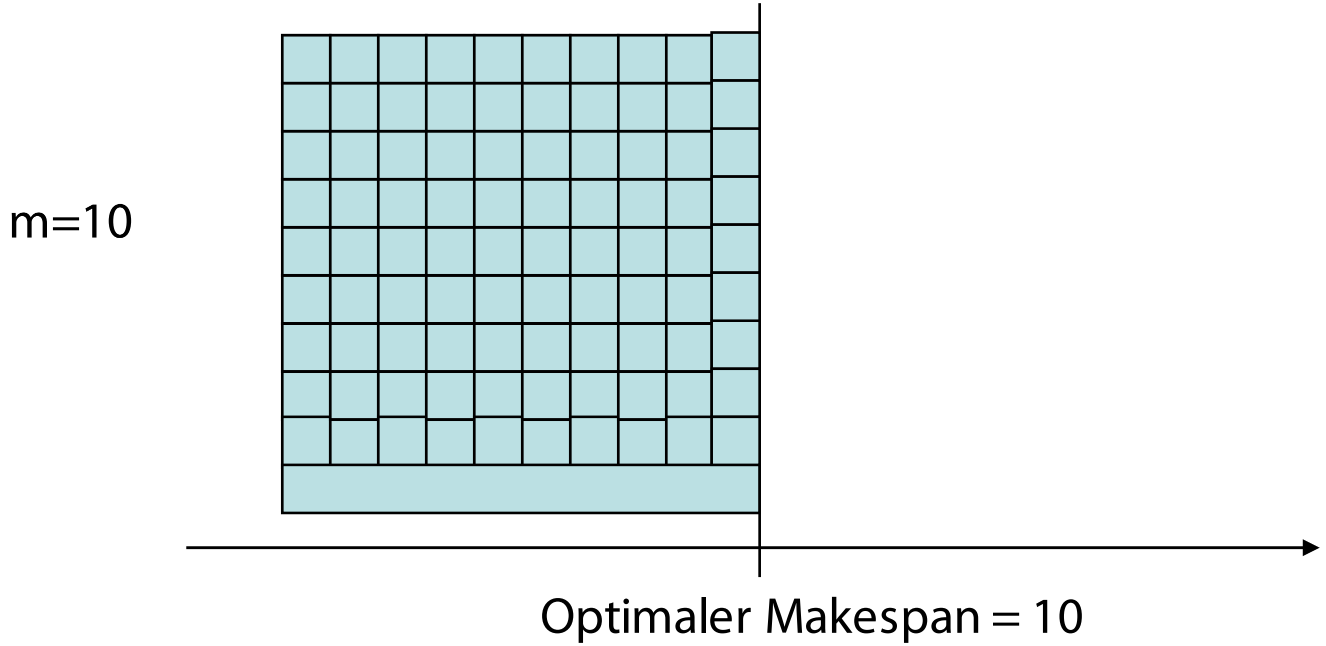
Beispiel: m Maschinen, m(m-1) Jobs der Länge 1, ein Job der Länge m



Lastbalancierung: List Scheduling

Ist die Analyse scharf? Ja!

Beispiel: m Maschinen, m(m-1) Jobs der Länge 1, ein Job der Länge m



Lastbalancierung: List Scheduling

Theorem (Graham): Der Greedy Algorithmus ist 2-approximativ.

Lastbalancierung: List Scheduling

Theorem (Graham): Der Greedy Algorithmus ist 2-approximativ.

$\longrightarrow$ Vergleiche Güte des Algorithmus mit optimalem Makespan L^*

Lemma 1: $L^* \geq \max_j t_j$

Beweis: Eine Maschine muss den zeitintensivsten Job bearbeiten.

Lemma 2: $L^* \geq (1/m) \sum_j t_j$

Beweis:

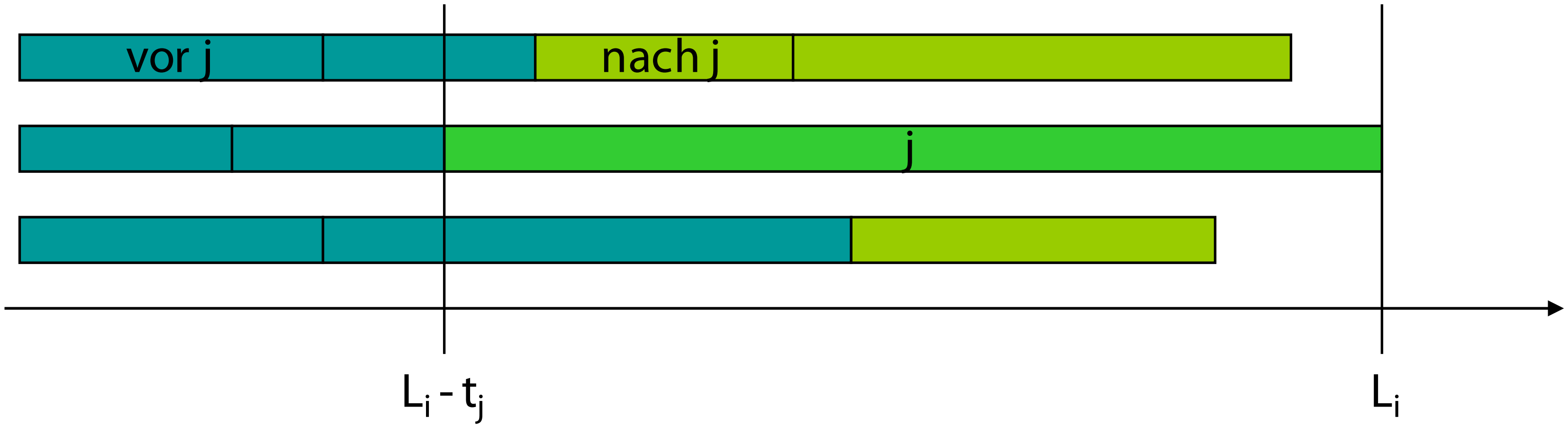
- Die Gesamtlast ist $\sum_j t_j$
- Eine der m Maschinen muss mindestens $1/m$ der Gesamtlast bekommen.

Lastbalancierung: List Scheduling

Theorem: Der Greedy Algorithmus ist 2-approximativ.

Beweis:

- Betrache Maschine i mit höchster Last L_i.
- Sei j der letzte Job in Maschine i.
- Da Job j Maschine i zugeordnet wurde, hatte i vorher die kleinste Last. Es gilt also $L_i - t_j \leq L_k$ für alle k.



Lastbalancierung: List Scheduling

Beweis (Forsetzung):

- Es gilt: $L_i\text{-}t_j \leq L_k$ für alle k
- Daraus folgt:

$$\begin{aligned} L_i - t_j &\leq (1/m) \textstyle\sum_k L_k \\ &= (1/m) \textstyle\sum_k t_k \\ &\leq L^* \qquad \text{wegen Lemma 1 } (L^* \geq \max_j t_j) \end{aligned}$$

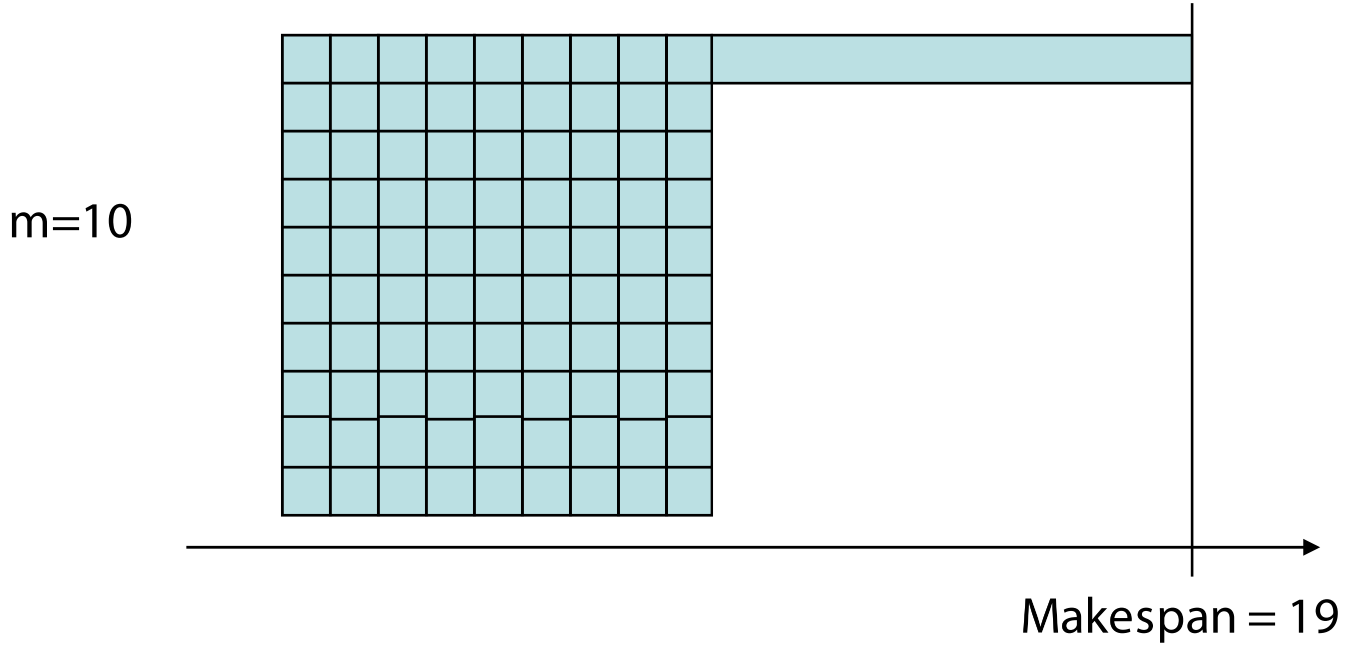
- Also gilt wegen Lemma 2 ($L^* \geq (1/m) \sum_j t_j$):

$$L_i = (L_i\text{-}t_j) + t_j \leq L^* + L^* = 2 \cdot L^*$$

Lastbalancierung: List Scheduling

Ist die Analyse scharf? Ja!

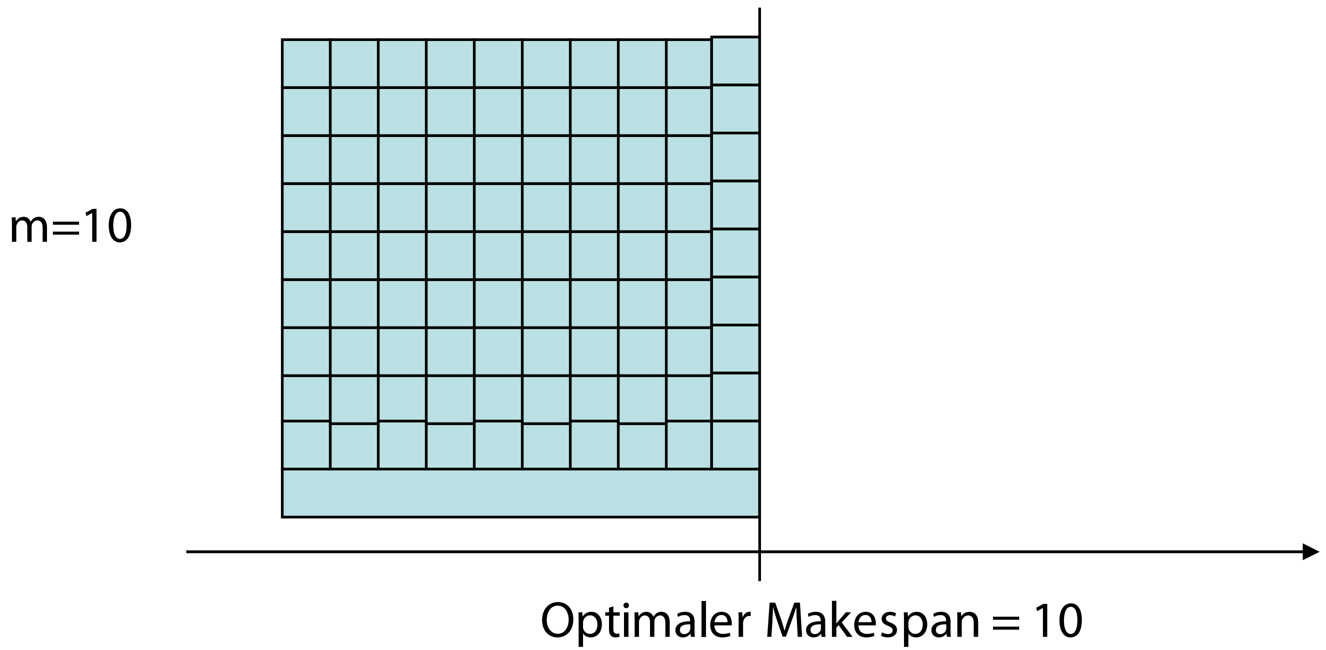
Beispiel: m Maschinen, m(m-1) Jobs der Länge 1, ein Job der Länge m



Lastbalancierung: List Scheduling

Ist die Analyse scharf? Ja!

Beispiel: m Maschinen, m(m-1) Jobs der Länge 1, ein Job der Länge m



Übersicht

- P und NP
- Approximationsalgorithmen
- Güte der Approximation
- Lastbalancierungsproblem