Die Algorithmen von Bartal und Karger zur Lösung des Graham-Problems

Christian Lennerz
lennerz@mpi-sb.mpg.de

12. November 2002

Seminarvortrag im Vertiefungsfach
Informations- und Technologiemanagement
an der Wirtschaftswissenschaftlichen Fakultät
der Universität des Saarlandes

Saarbrücken 1999

Gliederung des Vortrags

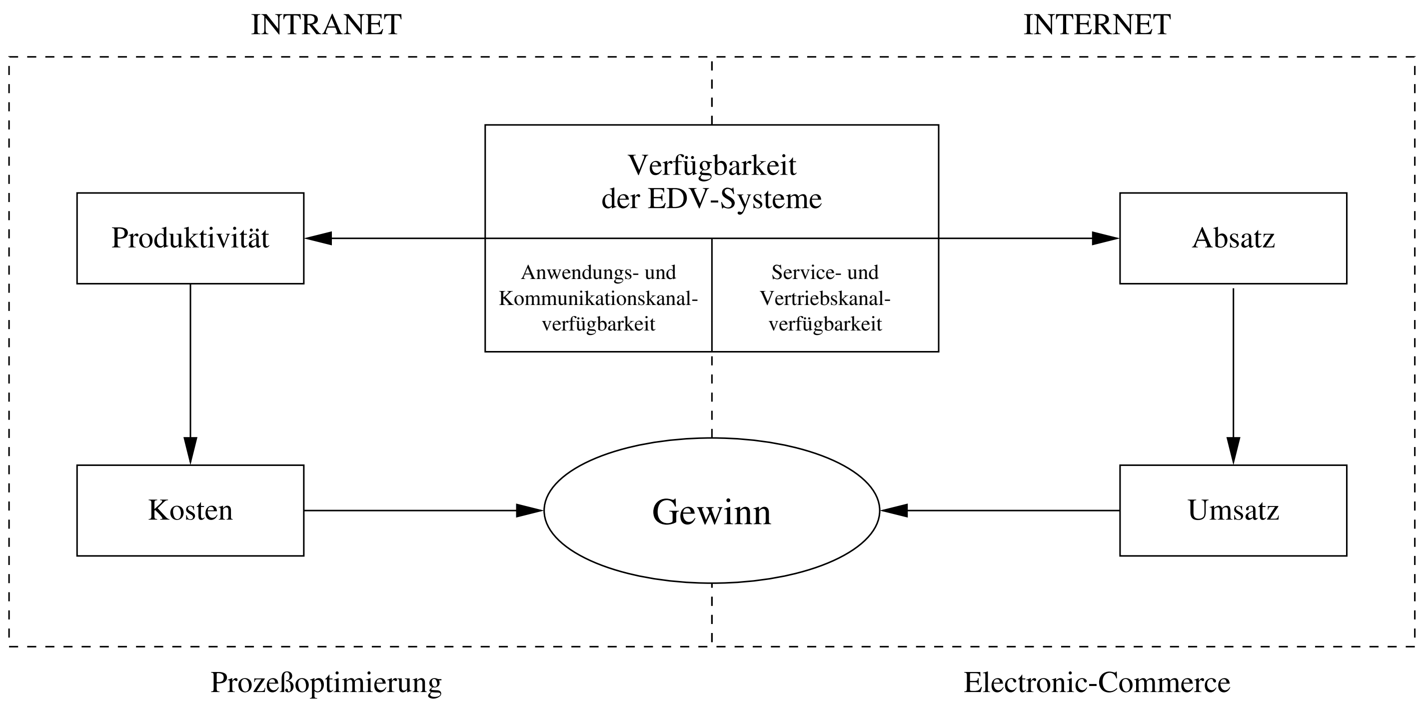
1. Das Graham-Problem – motiviert durch Alltag, Forschung und Kommerz
2. Ein Anwendungsbeispiel aus dem Bereich *Electronic Commerce*
3. Die worst-case-Analyse von LIST als Ausgangspunkt kompetitiverer Algorithmen
4. Der Algorithmus von Bartal, Fiat, Karloff und Vohra
5. Der Algorithmus von Karger, Phillips, und Torng
6. Bewertung der Verfahren

Das GRAHAM-Problem – motiviert durch Alltag, Forschung und Kommerz

- *Alltag:* Welche Kasse soll ich im Supermarkt wählen, so daß in möglichst kurzer Zeit möglichst viele Kunden bedient werden können?
- *Informationstechnik:* Von welcher Platte eines redundanten Festplattensystems soll das angeforderte File gelesen werden, so daß der Durchsatz des Systems maximiert wird?
- *Unternehmen:* Wie sollen beim Versand Pakete auf Verpackungsstationen verteilt werden, so daß an einem Arbeitstag möglichst viele Kunden beliefert werden können?
- *Mathematik/Informatik:* Wie effizient kann ein solch einfach zu formulierendes aber schwer zu lösendes Problem beantwortet werden?

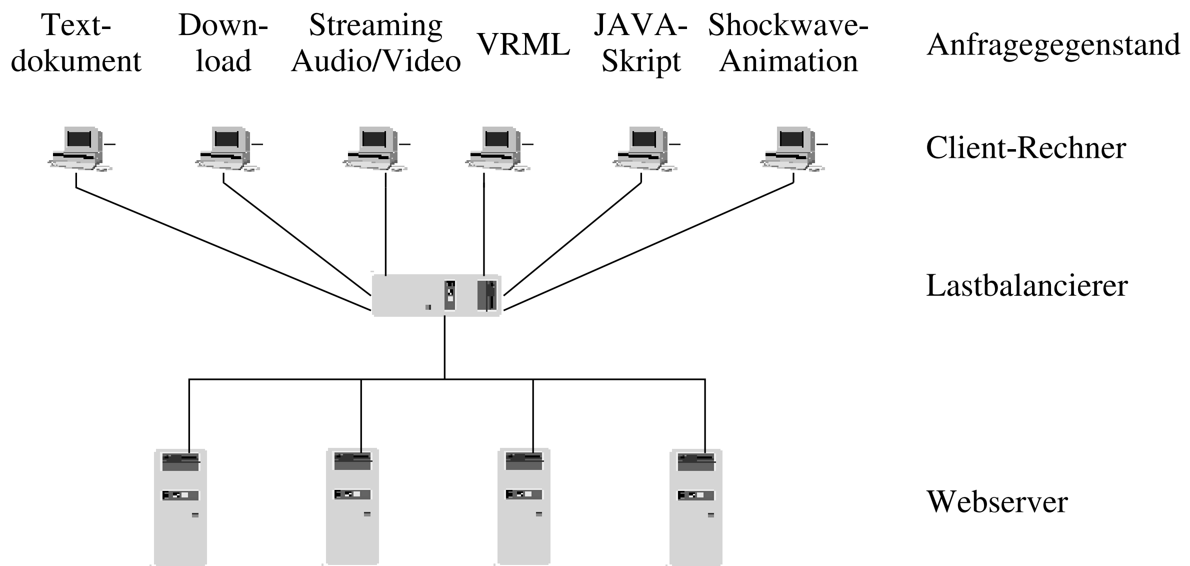
Die betriebswirtschaftliche Bedeutung der EDV-Verfügbarkeit im E-Commerce-Umfeld

Im E-Commerce-Zeitalter ist die Verfügbarkeit der EDV-Systeme gleichbedeutend mit Service- und Vertriebsverfügbarkeit.



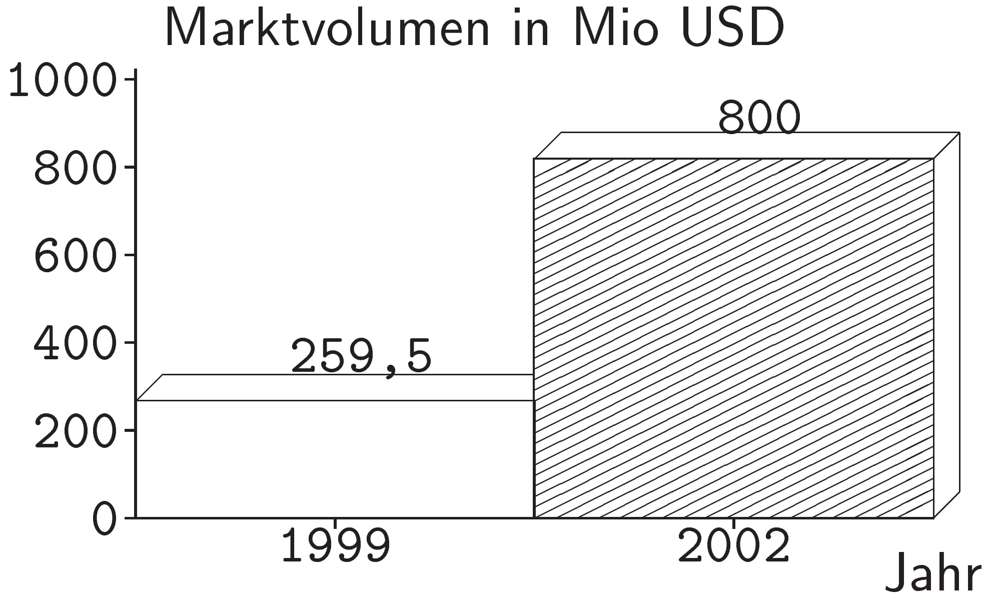
Der Lastbalancierer als technische Lösung

- Erhaltung der minimalen Verfügbarkeit durch Redundanz.
- Erlangung optimaler Verfügbarkeit (Durchsatzmaximierung) durch lastbalancierte Verteilung der Internetanfragen auf die Webserver.

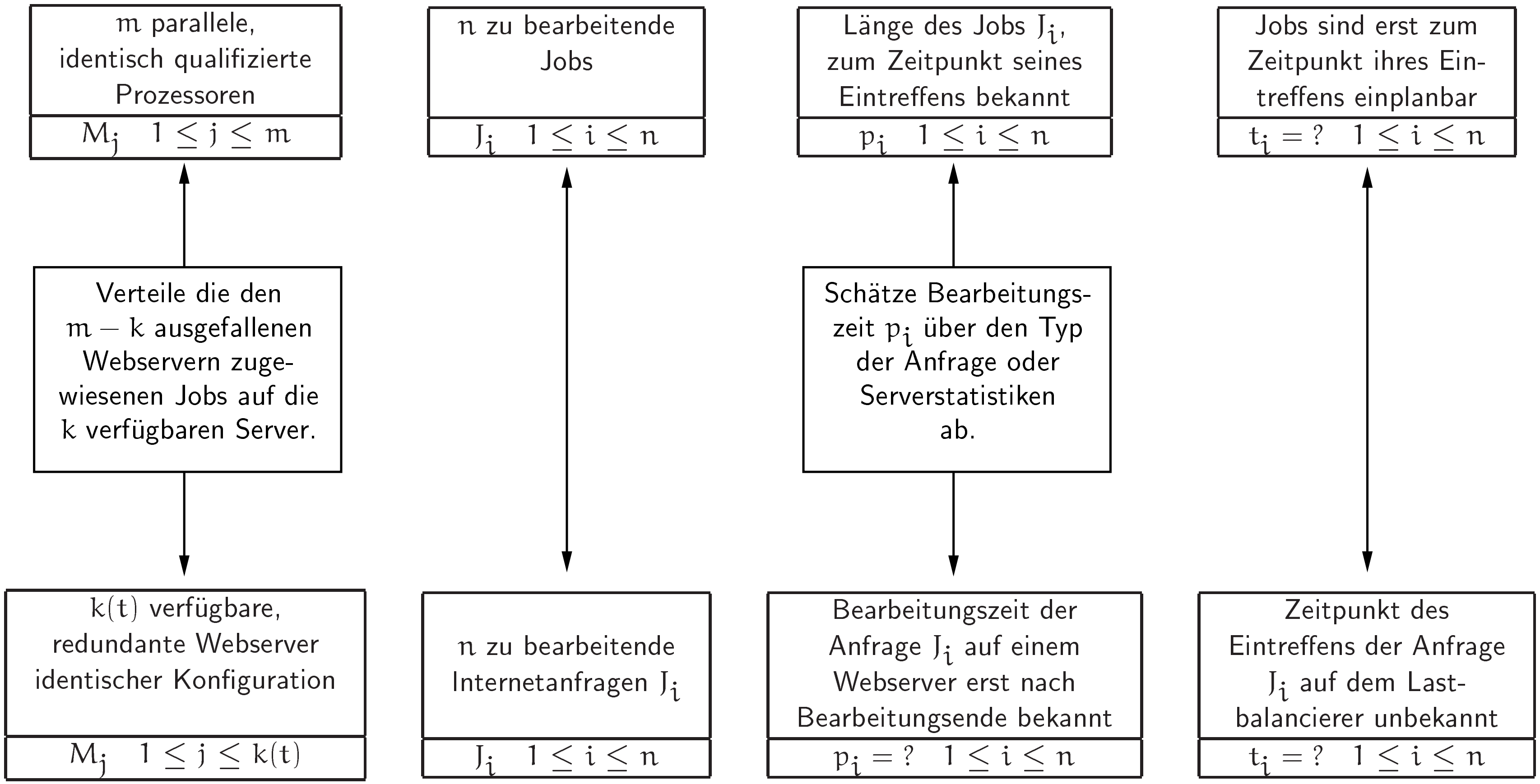


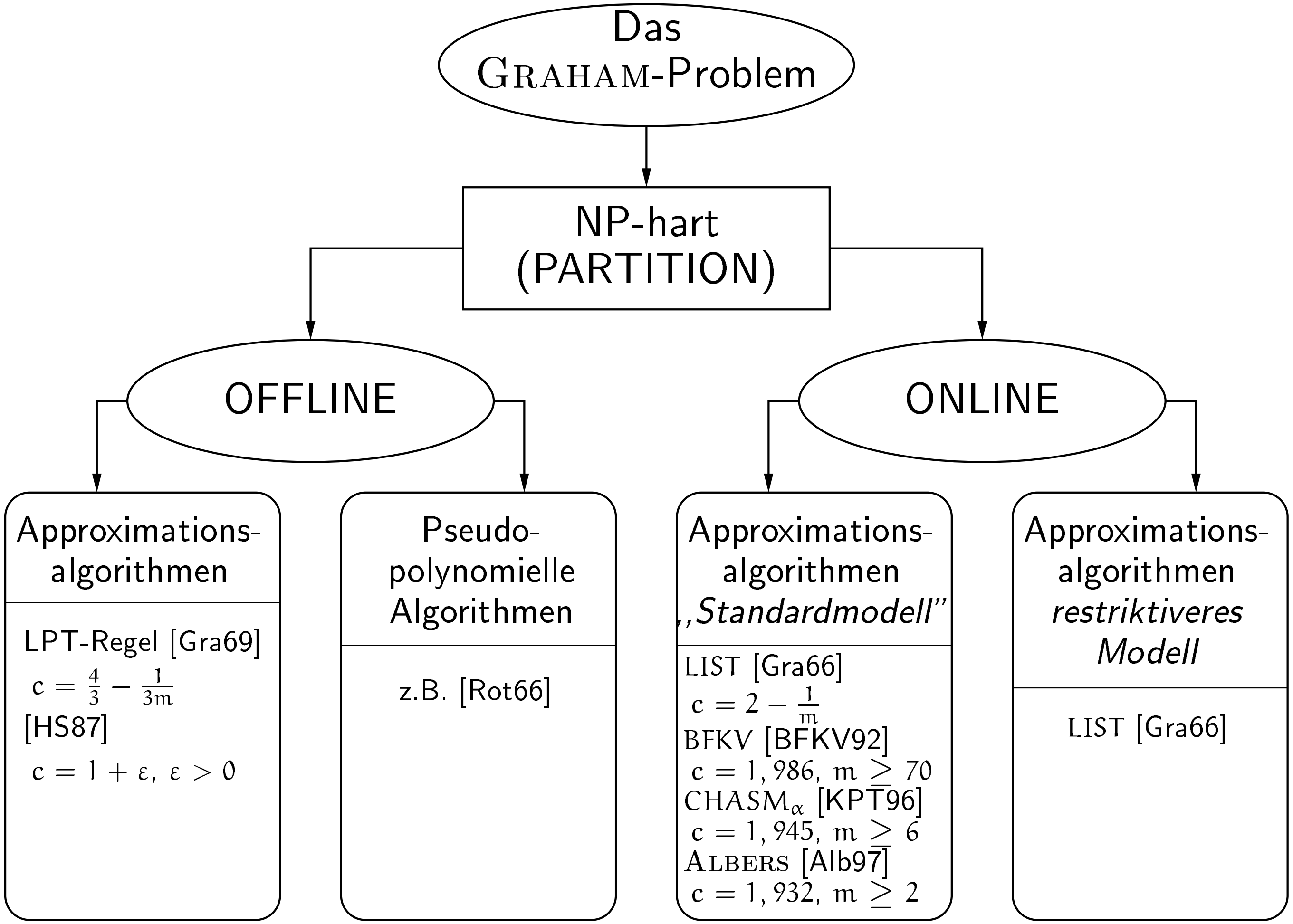
Marktpotential

Prognose von *Collaborative Research* [Tör99] zur Entwicklung des Marktvolumens für (IP-basierte) Lastbalancierer



Das Schedulingmodell



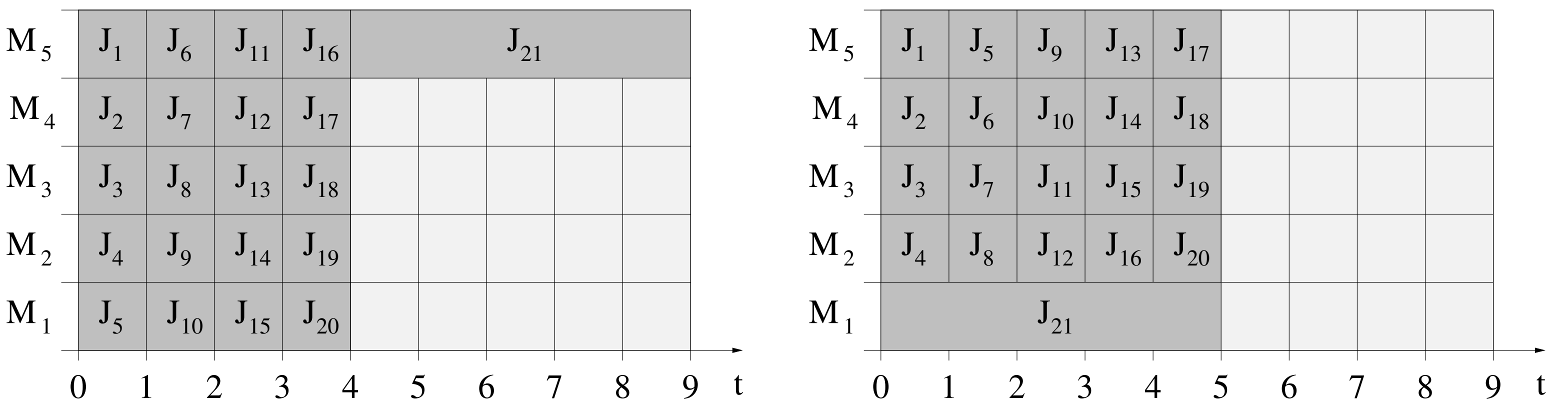

Das GRAHAM-Problem
NP-hart (PARTITION)
OFFLINE
ONLINE
Approximations-algorithmen
LPT-Regel [Gra69]
$c = \frac{4}{3} - \frac{1}{3m}$
[HS87]
$c = 1 + \varepsilon,\ \varepsilon > 0$
Pseudo-polynomielle Algorithmen
z.B. [Rot66]
Approximations-algorithmen *„Standardmodell"*
LIST [Gra66]
$c = 2 - \frac{1}{m}$
BFKV [BFKV92]
$c = 1{,}986,\ m \geq 70$
CHASM$_\alpha$ [KPT96]
$c = 1{,}945,\ m \geq 6$
ALBERS [Alb97]
$c = 1{,}932,\ m \geq 2$
Approximations-algorithmen *restriktiveres Modell*
LIST [Gra66]

Die worst-case-Analyse von $\mathcal{LIST}$ als Ausgangspunkt kompetitiverer Algorithmen

Perfekte Lastbalancierung, wie sie vom $\mathcal{LIST}$-Algorithmus angestrebt wird, führt zu schlechtem worst-case-Verhalten:

Die folgende Jobsequenz verursacht einen kompetitiven Faktor von $2 - \frac{1}{m}$:

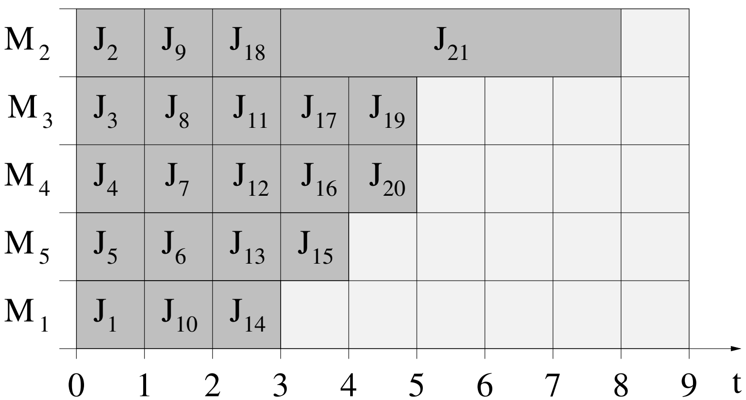
$$\sigma_{m(m-1)+1} = (J_1, \ldots, J_{m(m-1)}, J_{m(m-1)+1}) \quad \pi_{m(m-1)+1} = (\underbrace{1, \ldots, 1}_{m(m-1)}, m)$$



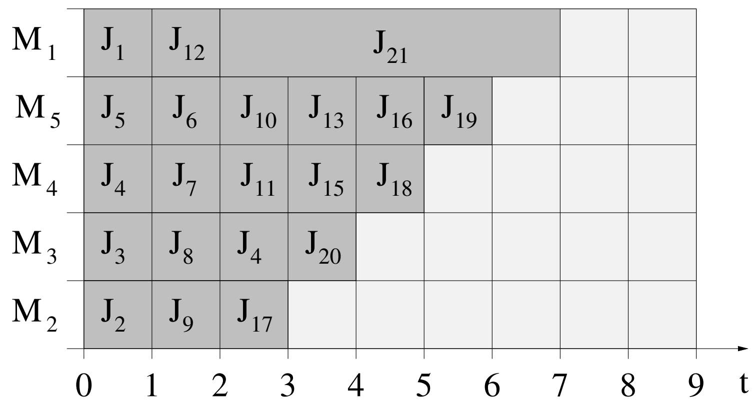
Bewußte Lastdebalancierung zur Verbessserung des kompetitiven Faktors

Idee: Vermeide optimal balancierte Pläne!

Vorsicht: Es existiert einen *Trade-Off* :
Lastdebalancierung führt bei einer Sequenz von (nahezu) identischen Joblängen zu kompetitiv schlechten Plänen.



Der Algorithmus von Bartal et al. : diskrete Lastdebalancierung $(44,5\%)$



Der Algorithmus von Karger et al. : stetige Lastdebalancierung

Der Algorithmus von Bartal et al.

Strategie:

- Halte $d = \langle 0,445m \rangle$ Maschinen unter ,,niedriger" Last.
- Plane ,,*kleine Jobs*" solange auf Maschinen mit ,,*hoher*" *Last* ein, bis der kompetitive Faktor seinen avisierten Wert überschreitet.
- Weise ,,*kleine*" *Jobs* nur dann Maschinen mit ,,*niedriger*" *Last* zu, falls das Lastungleichgewicht zu groß geworden ist.
- Plane ,,*große*" *Jobs* auf Maschinen mit ,,*niedriger Last*" ein.

Einplanungsregel:

Sei J_t der zum Zeitpunkt $t-1$ einzuplanende Job und p_t seine Länge. Des weiteren bezeichne M_{d+1}^{t-1} die kleinste, niedrig belastete Maschine mit Last l_{d+1}^{t-1}. A_d^{t-1} sei die mittlere Auslastung aller Maschinen mit niedriger Last. Dann plane J_t auf M_{d+1}^{t-1} ein, falls gilt:

$$l_{d+1}^{t-1} + p_t \leq (2-\varepsilon)A_d^{t-1}$$

Andernfalls weise J_t der kleinsten Maschine M_1^{t-1} zu.

Lösungsgüte:

Theorem. [Kompetitiver Faktor] *Sei $m \geq m_0 = 70$ und $\varepsilon := \frac{1}{m_0}$, dann ist der BFKV-Algorithmus $(2-\varepsilon) = 1.986$-kompetitiv.*

Der BFKV-Algorithmus:

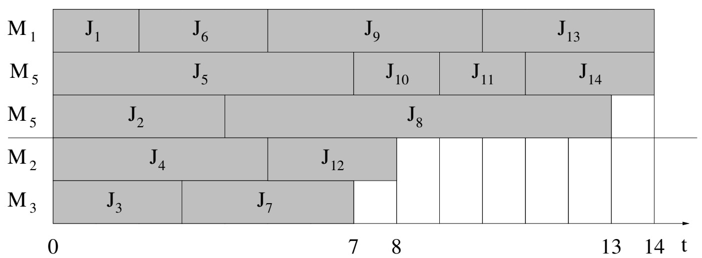
```
d := ⟨0,445m⟩
ε := 1/70
for j := 1 to m do
   A_j := 0 (* Initialisierung: t = 0 *)
end for
for all jobs J_t in job sequence do
   if l_{d+1} + p_t ≤ (2 − ε)A_d then
      Assign job J_t to machine M_{d+1}
      l_{d+1} := l_{d+1} + p_t
   else
      Assign job J_t to machine M_1
      l_1 := l_1 + p_t
   end if(* l_j, A_j, M_j, 1 ≤ j ≤ m zu t − 1 *)
   Order M according to non-decreasing machine loads
   A_d := (1/d) Σ_{i=1}^{d} l_i (* l_j, A_j und M_j, 1 ≤ j ≤ m zu t *)
end for(* Laufzeit: O(n log m) *)
```

Der Algorithmus von Karger et al.

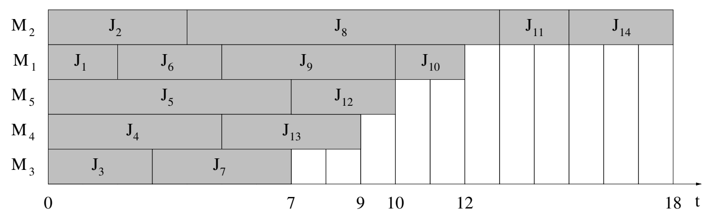
Strategie:

- Halte ein im besten Fall m-*stufiges Treppenmuster* der Lastdebalancierung aufrecht.
- Plane einen Job abhängig von seiner Länge auf der größten Maschine ein, die die Einhaltung des avisierten kompetitiven Faktors gestattet.

Vergleich: $\sigma_{14} = (J_1, \ldots J_{12}) \quad \pi_{14} = (2, 4, 3, 5, 7, 3, 4, 9, 5, 2, 2, 3, 4, 3)$



Der Algorithmus von Bartal et al.



Der Algorithmus von Karger et al.

Einplanungsregel:

Sei J_t der zum Zeitpunkt $t-1$ einzuplanende Job und p_t seine Länge. Des weiteren bezeichne M_i^{t-1}, $1 \leq i \leq m$ die Maschine mit i-t kleinster Last l_i^{t-1}. A_i^{t-1} sei die mittlere Auslastung der i kleinsten Maschinen.
Dann plane J_t auf der größten Maschine M_k^{t-1} ein, für die gilt:

$$l_k^{t-1} + p_t \leq \alpha A_{k-1}^{t-1}, \quad \alpha = 1,945$$

Andernfalls weise J_t der kleinsten Maschine M_1^{t-1} zu.

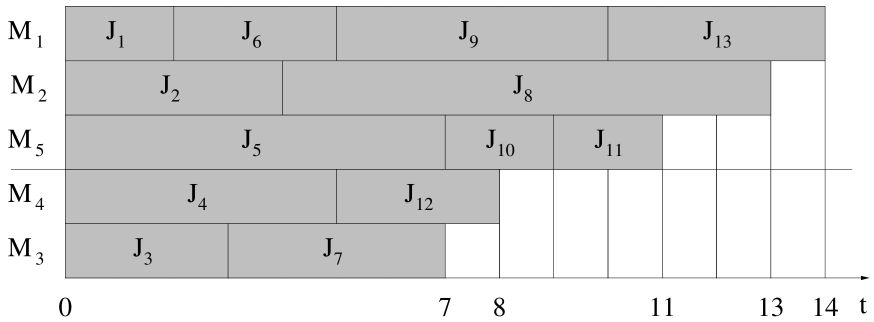
Lösungsgüte:

Theorem. [Kompetitiver Faktor] *Sei $\alpha := 1,945$ und $m \geq 6$, dann ist der CHASM_α-Algorithmus α-kompetitiv.*

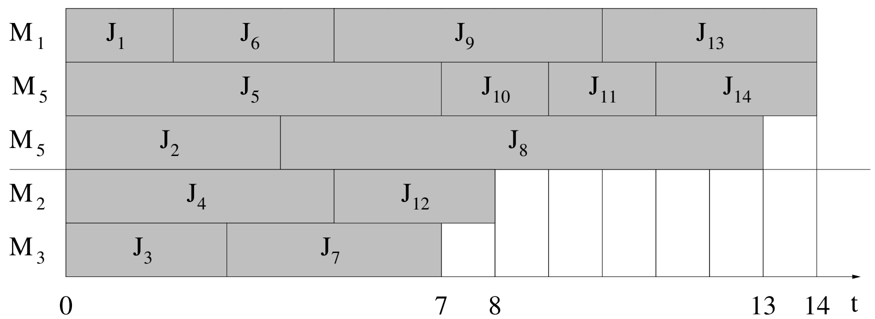
Der $CHASM_\alpha$-Algorithmus:

$\alpha := 1{,}945$
$A_0 := \infty$
for $j := 1$ to m **do**
  $A_j := 0$ (* Initialisierung: $t = 0$ *)
end for
for all jobs J_t in job sequence **do**
  $k := \max\{j \mid l_j + p_t \leq \alpha A_{j-1}\}$
  Assign job J_t to machine M_k (* l_j, A_j, M_j, $1 \leq j \leq m$ zu $t-1$ *)
  $l_k := l_k + p_t$
  Order M according to non-decreasing machine loads
  for $j := 1$ to m **do**
    $A_j := \frac{1}{j}\sum_{i=1}^{j} l_i$
  end for(* l_j, A_j und M_j, $1 \leq j \leq m$ zu t *)
end for(* Laufzeit: $O(mn)$ *)

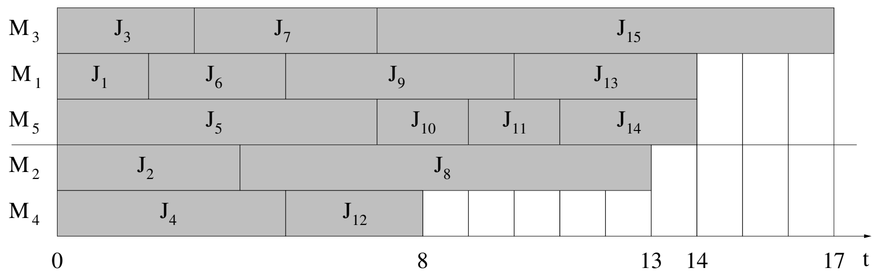
Beispiel für die Einplanungsregel des BFKV-Algorithmus



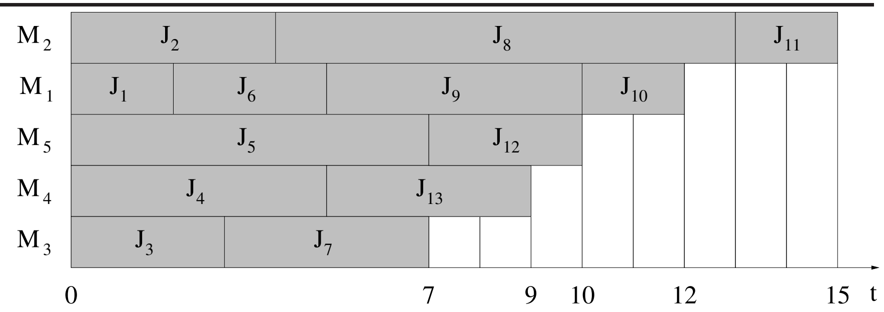
$t = 13$	$l_{d+1}^{13} = 11$ $p_{14} = 3$ $A_d^{13} = 7,5$	$l_{d+1}^{13} + p_{14}$ $\leq$ $1,986 A_d^{13}$
J_{14} auf M_5 einplanen		



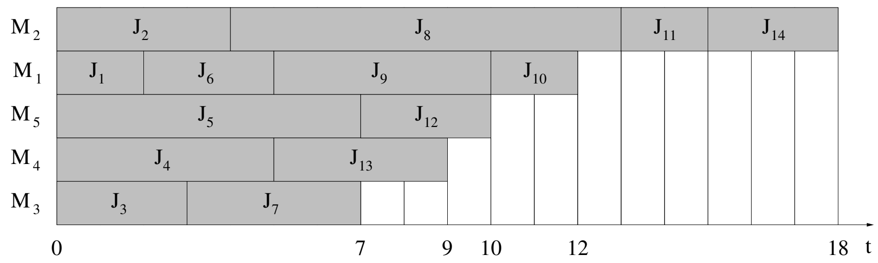
$t = 14$	$l_{d+1}^{14} = 13$ $p_{15} = 10$ $A_d^{14} = 7,5$	$l_{d+1}^{14} + p_{15}$ $>$ $1,986 A_d^{14}$
J_{15} auf M_3 einplanen		



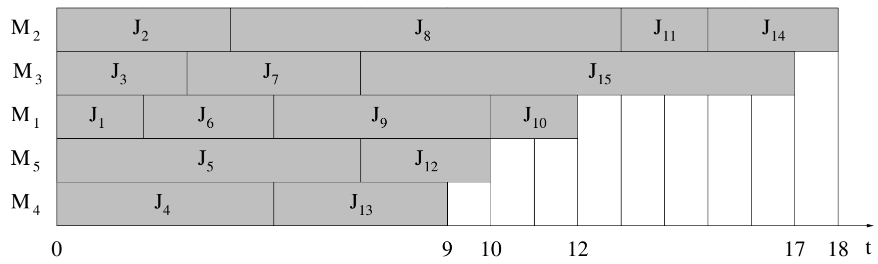
Beispiel für die Einplanungsregel des $CHASM_\alpha$-Algorithmus



M_j^{t-1}	l_j^{t-1}	A_{j-1}^{t-1}	$l_j^{t-1} + p_t$	αA_{j-1}^{t-1}
2	15	9,5	18	≈ 19
1	12	$8,\overline{6}$	15	$\approx 17,\overline{3}$
5	10	8	13	≈ 16
4	9	7	12	≈ 14
3	7	∞	10	∞
$t = 13$:	J_{14} auf M_2 einplanen			



M_j^{t-1}	l_j^{t-1}	A_{j-1}^{t-1}	$l_j^{t-1} + p_t$	αA_{j-1}^{t-1}
2	18	9,5	28	¡ 19
1	12	$8,\overline{6}$	22	¡ $17,\overline{3}$
5	10	8	20	¡ 16
4	9	7	19	¡ 14
3	7	∞	17	∞
$t = 14$:	J_{15} auf M_3 einplanen			



Literatur

[Alb97] S. Albers. Better bounds for online scheduling. In *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC97)*, pages 130–139, 1997.

[BFKV92] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. In N. Alon, editor, *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 51–58, Victoria, B.C., Canada, May 1992. ACM Press.

[Gra66] R. L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[Gra69] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Computing*, 17:263–269, 1969.

[HS87] D. Hochbaum and D Shmoys. Using dual approximation algorithms for scheduling problems. *Journal of the ACM*, 34:144–162, 1987.

[KPT96] D. R. Karger, S. J. Philips, and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20:400–430, 1996.

[Rot66] M. H. Rothkopf. Scheduling independent tasks on parallel processors. *Management Science*, 1966.

[Tör99] E. Török. Spiegelkabibnett. *ct – Magazin für Computertechnik*, (6):302–308, 1999.