The Power of Online Reordering

Matthias Westermann
University of Bonn

Online algorithms

- They are used in practice for a potentially infinite run-time.
- During run-time, new requests for service are permanently issued, e.g.:
  - Routing requests for data packets in a network
  - Access requests in a storage system

- They are crucial for ambitious computer applications processing huge amounts of data with increasingly complex hardware.

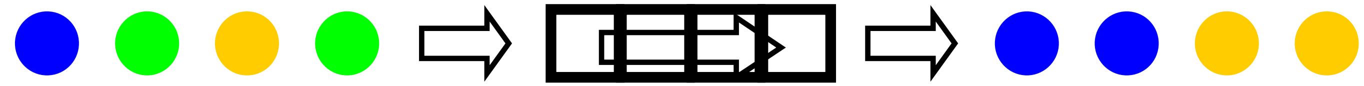
Online model

- An online algorithm gets to know the input sequence of requests for service incrementally, one request at a time, without knowledge about the future.
- Classic model:
  A new request is not issued
  until the previous one is served.

Online reordering paradigm

- In real applications, request can usually be delayed for a short amount of time.
- As a consequence, the input sequence of requests can be reordered in a limited fashion in order to optimize the performance.

Application:
Rendering in computer graphics



sequence of primitives buffer graphics hardware

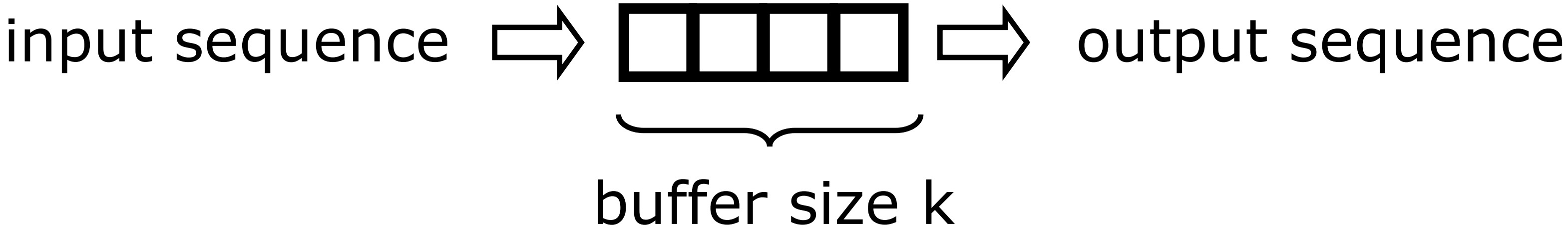
□ Given: Sequence of primitives with state changes (consecutive primitives differ in their attribute values).

□ Objective: Reorder the sequence of primitives in such a way that the number of state changes is reduced.

Applications

- Rendering in computer graphics
- Paint shop in car manufacturing
- Disk scheduling
- Machine scheduling
- …

Online reordering paradigm



input sequence ⇨ buffer ⇨ output sequence

buffer size k

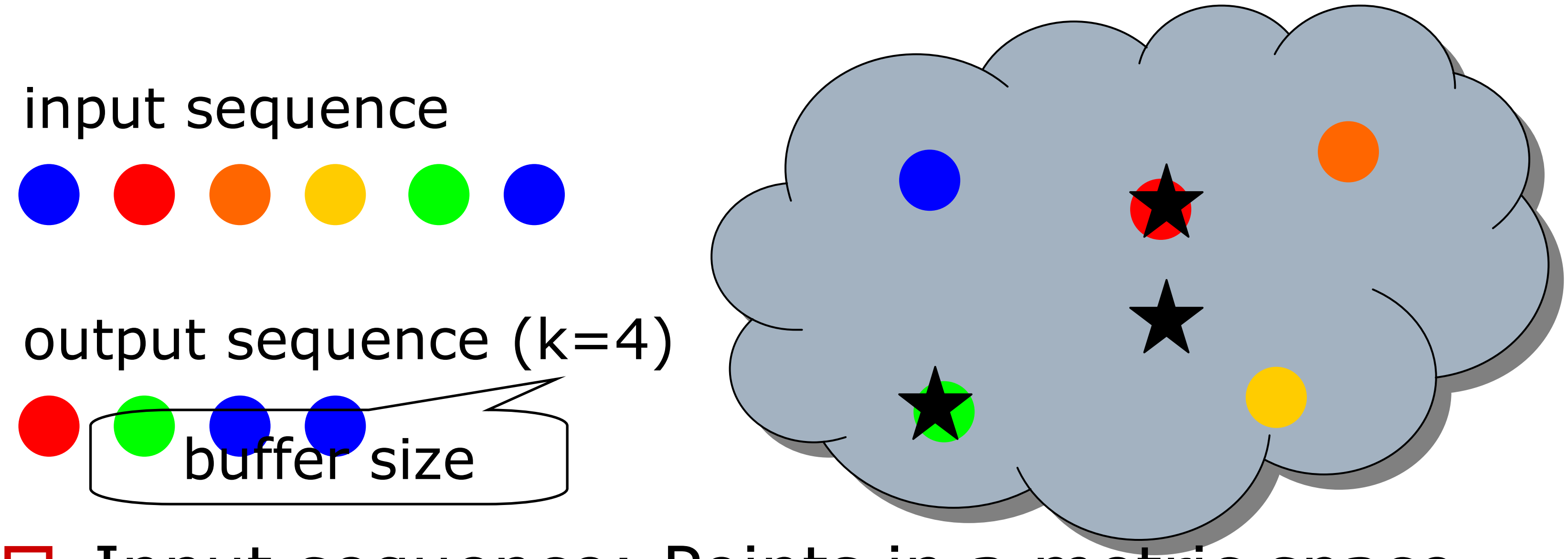
Buffer can be used to reorder the input:

- Buffer contains the first k requests of the input that are not serviced so far.
- Online algorithm selects a request contained in the buffer for service.
- Thereafter the next request in the input takes the place of the serviced request.

Competitive analysis

- Comparison
  - Online algorithm
    (without knowledge about the future)
  - Offline algorithm
    (knows the whole input in advance)
- An online algorithm is c-competitive, if its cost are at most c times the cost of an optimal offline algorithm.

Reordering buffers for general metric spaces



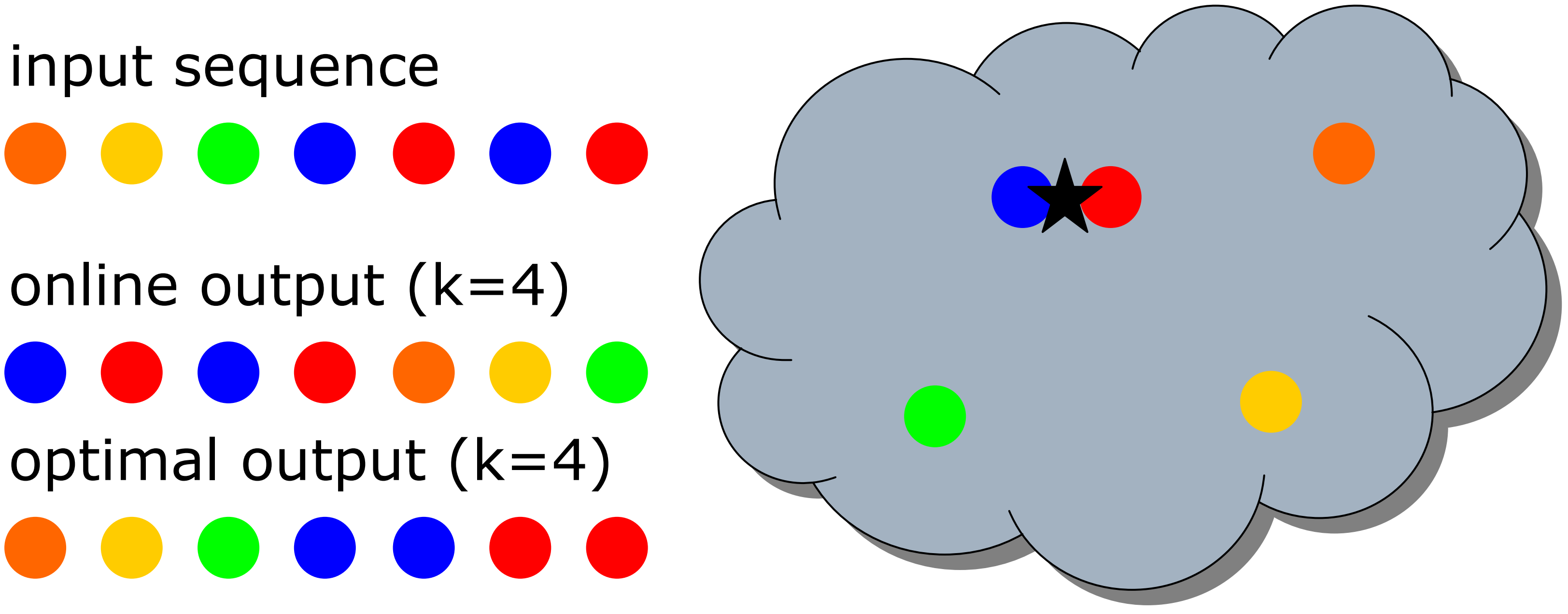
input sequence

output sequence (k=4)

buffer size

- □ Input sequence: Points in a metric space.
- □ Objective: Move the server to the points such that the total traveled distance is minimized.

Lower bound for shortest distance first



input sequence

online output (k=4)

optimal output (k=4)

- □ No memoryless algorithm can achieve a competitive ratio of o(k) [Khandekar, Pandit STACS’06].

Approximation of metric spaces

- □ Tree metric space:
  Shortest path metric induced by a tree.
- □ Each n-point metric space can randomly be approximated by tree metric spaces with an approximation ratio of O(log n) [Fakcharoenphol, Rao, Talwar STOC’03].
- □ We only need an algorithm for (hierarchical well-separated) trees.

Our results [Englert, Räcke, W. STOC’07]

- □ Algorithm for general trees: Competitive ratio O(D log k).
- □ Improved analysis for hierarchical well-separated trees: Competitive ratio O($\log^2$ k).

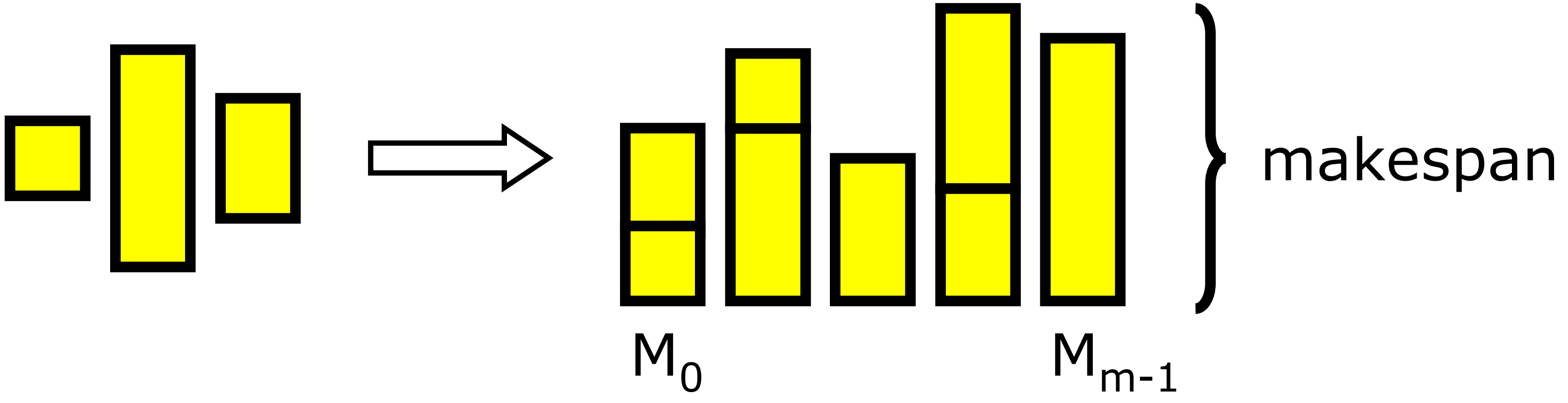
hop-diameter of the tree

- □ Randomized algorithm for general n-point metric spaces: Competitive ratio O(log n $\log^2$ k).

Open questions

- □ Can the competitive ratio be reduced to O(ploylog k) for line metric spaces or arbitrary trees?
- □ Can the competitive ratio be reduced to O(1) for any non-trivial metric space?
- □ Can the competitive ratio be reduced to O(polylog k) for general metric spaces?

Minimum makespan scheduling



- □ Input sequence: Jobs with processing times.
- □ Objective: Assign the jobs
  to m parallel machines without preemption
  such that the makespan is minimized.

m identical machines: Previous work (no reordering)

- 1.986 [Bartal, Fiat, Karloff, Vohra STOC'92]
- 1.945 [Karger, Phillips, Torng SODA'94]
- 1.923 [Albers STOC'97]
- 1.920 [Fleischer, Wahl ESA'00]

- 1.880 [Rudin PhDThesis'01]
- 1.853 [Gormley et al. SODA'00]
- 1.852 [Albers STOC'97]
- 1.837 [Bartal, Karloff, Rabani IPL'94]

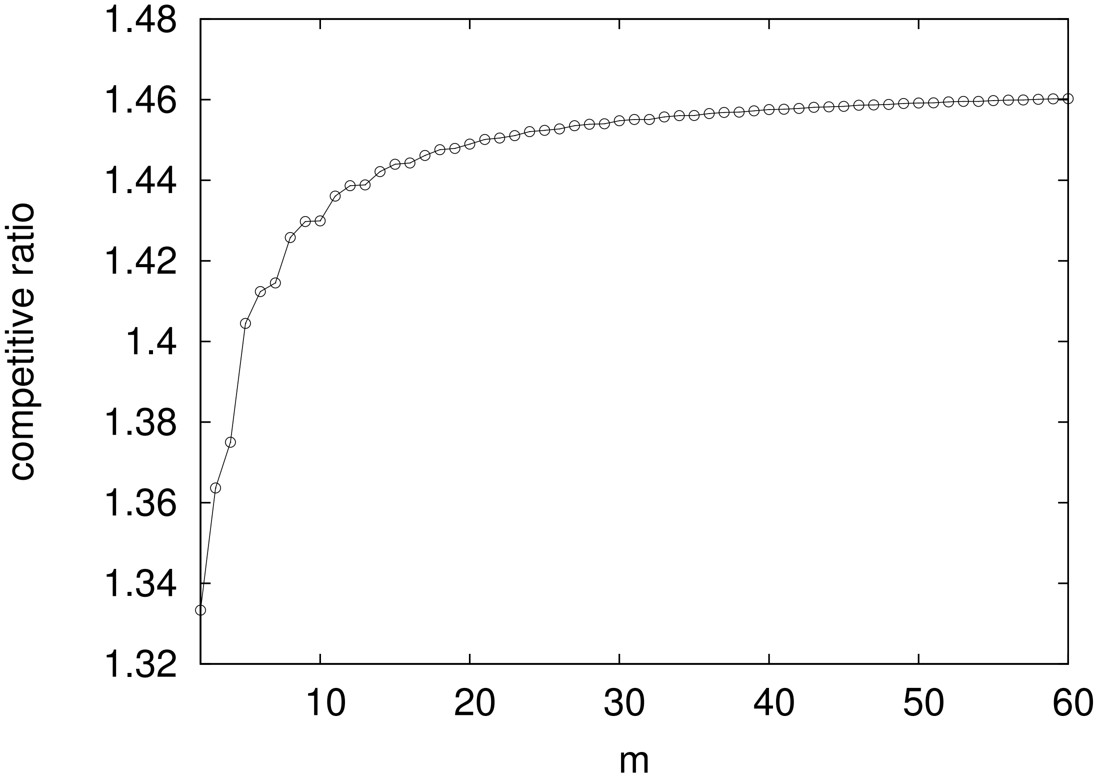
m identical machines: Our main results [Englert, Özmen, W. FOCS’08]

- □ Lower bound of r_m, if the size of the buffer does not depend on the input sequence.
  - $r_2 = 4/3$
  - $\lim_{m\to\infty} r_m = \text{LambertW}_{-1}(-1/e^2) / (1+\text{LambertW}_{-1}(-1/e^2)) \approx 1.4659$

smallest real solution to $x \cdot e^x = -1/e^2$

- □ Scheduling algorithm matching the lower bound with a buffer of size $\lceil (1+2/r_m)\cdot m \rceil + 2$.
  - $1+2/r_2 = 2.5$
  - $\lim_{m\to\infty} 1+2/r_m \approx 2.36$

Values of r_m

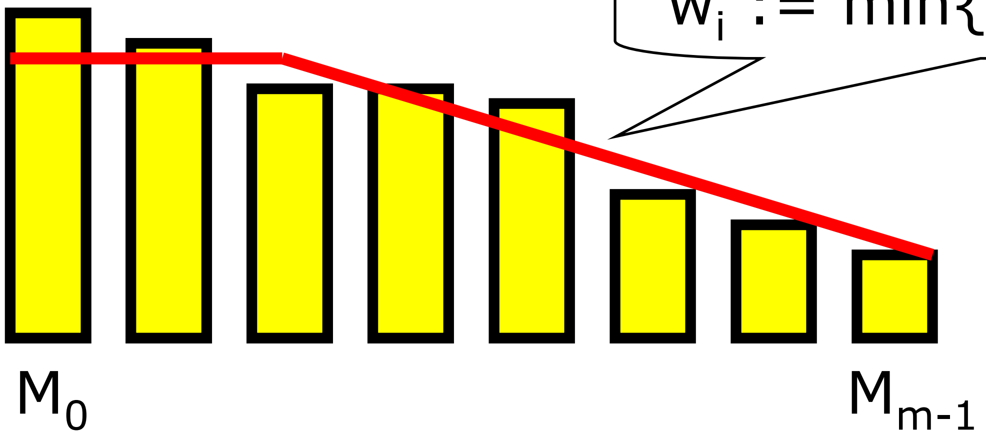

1.48
1.46
1.44
1.42
1.4
1.38
1.36
1.34
1.32
competitive ratio
10
20
30
40
50
60
m

m identical machines: Overview

m	our results reordering buffer	lower bounds no reordering	upper bounds no reordering
2	1.3333	1.5	1.5
3	1.3636	1.6667	1.6667
4	1.375	1.7321	1.7333
$\to \infty$	1.4659	1.8800	1.9201

m identical machines: The lower bound of r_m

- □ Assume for contradiction that algorithm A achieves a competitive ratio $r < r_m$ with a buffer of size k.
- □ $1/\varepsilon+k$ jobs of size ε arrive.

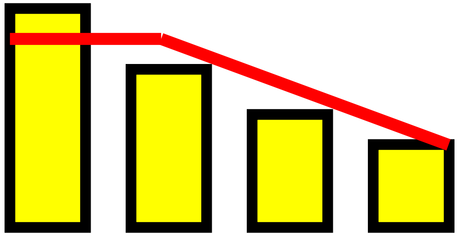


$w_i := \min\{r_m/m, (r_m-1)/i\}$

r_m is the solution to $\sum w_i = 1$.

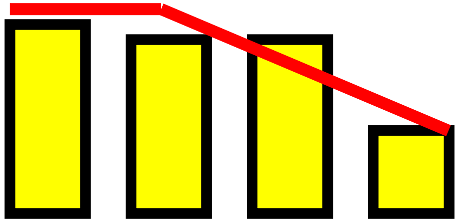
m identical machines: The lower bound of r_m

- There exists a machine M_j with load $\geq w_j$.
- If $w_j = r_m/m$, no more jobs arrive.
  - Optimal makespan $\leq (1+k\cdot\varepsilon)/m+\varepsilon = (1+(k+m)\cdot\varepsilon)/m$.
  - Competitive ratio of A $\geq r_m/(1+(k+m)\cdot\varepsilon) > r$.



m identical machines: The lower bound of r_m

- If $w_j = (r_m-1)/j$, (m-j) large jobs of size 1/j arrive.



  - Optimal makespan $\leq (1+k\cdot\varepsilon)/j+\varepsilon = (1+(k+j)\cdot\varepsilon)/j$.
  - If A schedules two large jobs on the same machine, competitive ratio of A $\geq 2/(1+(k+j)\cdot\varepsilon) > r$.
  - Otherwise, i.e., A schedules at least one of the large jobs on a machine with load $\geq (r_m-1)/j$, competitive ratio of A $\geq r_m/(1+(k+j)\cdot\varepsilon) > r$.

m identical machines: The optimal algorithm

- When a new job arrives:
  - Store this job in the buffer and remove a job J of smallest size from the buffer.
  - Schedule J on a machine M_i with load $\leq w_i \cdot (T + m \cdot p(J)) - p(J)$.

total scheduled load

size of J

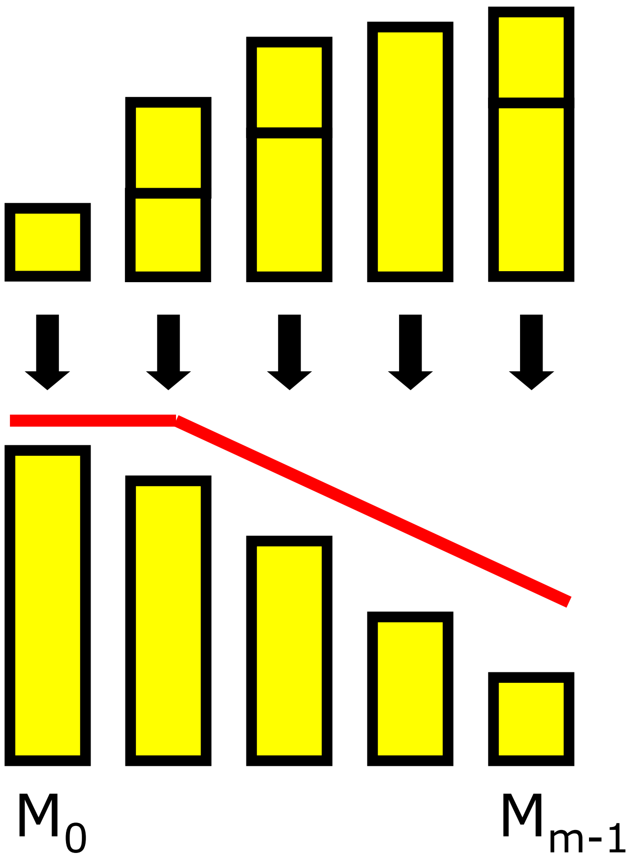
- After all jobs have arrived:
  - Schedule the remaining jobs in the buffer optimally on the machines.

Assume for contradiction that such a machine does not exist.
Then $T > \sum (w_i \cdot (T + m \cdot p(J)) - p(J)) = T$.

m identical machines: The optimal algorithm

Efficient final phase:

- Schedule virtually some of the remaining jobs on m empty machines according to LPT.
  Abort when the makespan is at least three times the size of the smallest job assigned so far.
- Schedule the jobs from the virtual machines on the real machines.
- Schedule the remaining jobs according to Greedy.



m identical machines: Further results

- □ Lower bounds of $3/2 > r_m$, if the buffer size is at most $\lfloor m/2 \rfloor$.
- □ Lower bound of $1+1/2^{1/2} \approx 1.7071$, if the buffer size is at most $\lfloor m/8 \rfloor$.
- □ Algorithms for different buffer sizes:

competitive ratio	buffer size
$3/2$	$\approx 1.6197 \cdot m+1$
$(1+r_m)/2 \approx 1.733$	$m+1$
$2-1/(m-k+1)$	$k \in [1,(m+1)/2]$

m related machines: Our result

□ Scheduling algorithm achieving the competitive ratio 2 with a buffer of size m.

our result reordering buffer	lower bound no reordering	upper bound no reordering
2	2.438	5.828

m related machines: The algorithm

- When a new job arrives:
  - Store this job in the buffer and remove a job J of smallest size from the buffer.
  - Schedule J on a machine M_i with load $\leq \underbrace{\alpha_i}_{\text{speed of } M_i}/\sum\alpha_j\cdot(\underbrace{T}_{\text{total scheduled load}}+m\cdot p(J))-\underbrace{p(J)}_{\text{size of J}}$.
- After all jobs have arrived:
  - Schedule the remaining jobs optimally on m corresponding empty machines.
  - Schedule the jobs from the virtual machines on the respective real machines.

m related machines: Analysis

- At the end of the arrival phase, the completion time of machine M_i is $\leq 1/\sum\alpha_j \cdot (T+(m-1)\cdot p(J_i))$

  $\leq$ OPT.

  last Job scheduled on M_i

- In the final phase, the completion time of each virtual machine is at most OPT.

Open questions

- □ Our algorithm for identical machines achieves the optimal competitive ratio. What buffer size is necessary to obtain this result?
  $\lfloor m/2 \rfloor \leq \ldots \leq \lceil (1+2/r_m)\cdot m \rceil + 2$
- □ Can our result for related machines be improved or can a better lower bound be shown in this case?
- □ Reordering for other scheduling problems?