On the randomized competitive ratio of bounded delay buffer management

ongoing work with Spyros Angelopoulos and Shahin Kamali

Christoph Dürr

January 2020

Sorbonne Université, CNRS, LIP6

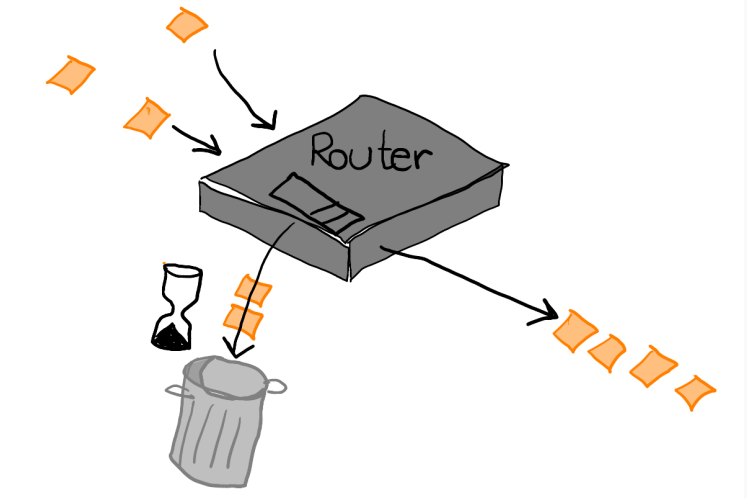
The problem

Online Buffer management

- Packets arrive online into a network router and need to be send on an outcoming link.
- Every packet represents a unit length job with

  release time $r_j \in \mathbb{N}$, (=time job is revealed to algorithm)
  deadline $d_j \in \mathbb{N}$,
  weight $w_j \in \mathbb{R}^+$. (=quality of service)

- Goal is to maximize the total weight of packets sent on time.

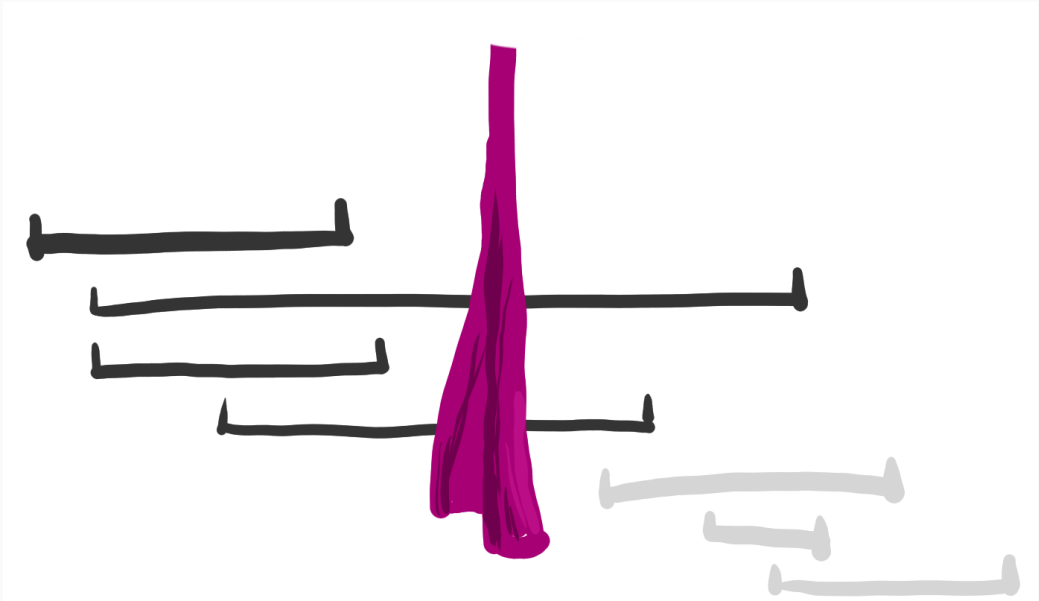


The competitive ratio

How bad is the algorithm ALG performing compared with the optimal solution?

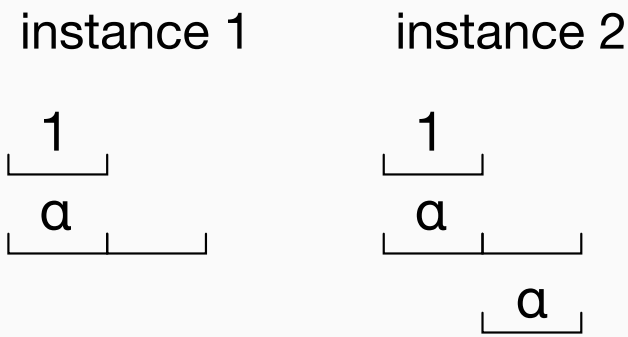
$$\max_{I} \frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}$$

- ALG(I) = obj. value of the solution produced by algorithm on instance I
- OPT(I) = obj. value of the optimal solution
- the ratio measures the *price of not knowing* the future part of the instance



Example

Let $\alpha > 1$. At time 0 instance 1 and instance 2 are indistiguishable.



What would you do?

- Send the heavy job. In instance 1 this gives ratio $(1 + \alpha)/\alpha$.
- Send the urgent job. In instance 2 this gives ratio $(2\alpha)/(1 + \alpha)$

This shows that for $\alpha = 1 + \sqrt{2}$, any deterministic online algorithm has ratio at least $\sqrt{2} \approx 1.414$.

By the way
At time 1 this dilemma could be repeated with heavier jobs, eventually leading to a lower bound of $\varphi \approx 1.618$.

Related work

Deterministic algorithms

2001. *[Hajek],[Kesselman et al]* Introduced problem. Deterministic lower bound φ.

- since then, many upper bounds for variants.

2018. *[Veselý, Chrobak, Jez, Sgall]* Optimal deterministic online algorithm.

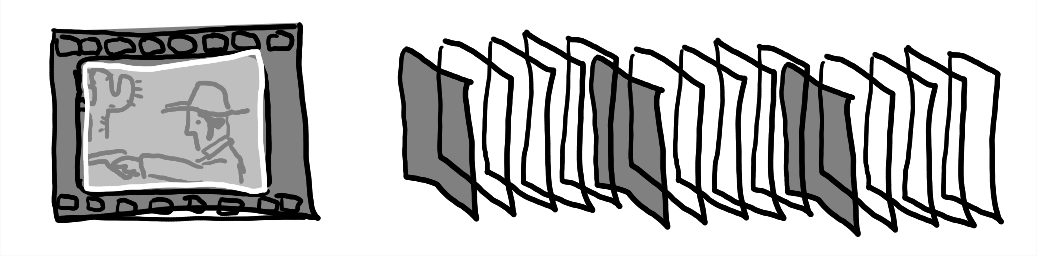
Randomized algorithms

2006. *[Chin, Chrobak, Fung, Jawor, Sgall, Tichý]* Randomized algorithm, 1.582 competitive against oblivious adversary.

2007. *[Bienkowski, Chrobak, Jez]* Randomized algorithms against adaptive adversary.

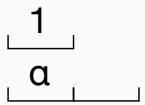
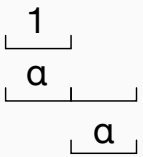
Roadmap

- Find optimal randomized algorithm (against adaptive adv.)
- Try to understand already the case when there are only two possible weights (light=1, heavy=α)



Lower bound on randomized competitive ratio

See it as a game played between algorithm and adversary.

	instance 1	instance 2
		
start with heavy job	$\frac{1+\alpha}{\alpha}$	$\frac{2\alpha}{2\alpha}$
start with urgent job	$\frac{1+\alpha}{1+\alpha}$	$\frac{2\alpha}{1+\alpha}$

Unique Nash equilibrium shows lower bound of $4 - 2\sqrt{2} \approx 1.1716$ for worst $\alpha = 1 + \sqrt{2}$.

Two simple deterministic (known) algorithms

The lower bound suggestes the following algorithms.

Greedy 

Wants to maximize profit now, in case more heavy jobs are to come.

Schedule a heavy job if available (tie break with deadline).

Commit 

Wants to make the most profit from the pending job, in case no more jobs are to come.

Compute a maximum weight schedule for the pending jobs (tie break heavy first).
Schedule the first one from it.

Barely random

After an initial biased coin flip, execute Greedy with probability p and Commit with probability $1 - p$, for

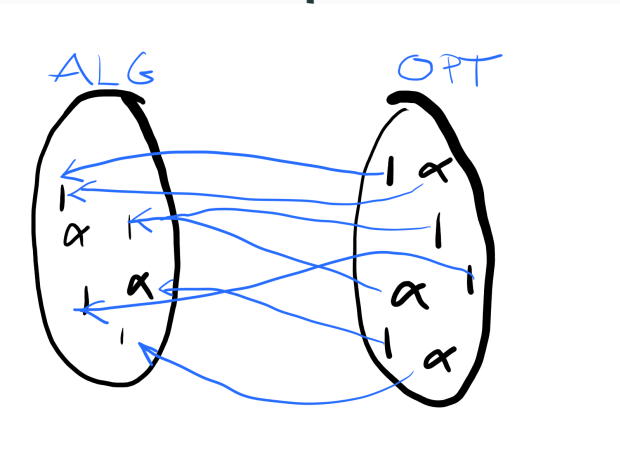
$$p = \frac{\alpha^2 - 1}{\alpha^2 + 2\alpha - 1}.$$

Our theorem

Barely Random is optimal.

The proof uses a standard charging scheme.

General technique



In order to show that the total weight of a set OPT is at most R times the total weight of a set ALG, charge items of OPT to items of ALG, and show that every item in ALG is charged at most R times its value.

Experiments

Two test beds:
For both: Generate δH jobs with release time choosen uniformly in $[0, H)$.

1. **Lower bound distribution** Generate weight and span (=deadline - release time) with same proportion as in lower bound.
2. **Exponential distribution** Generate weight with equal probability. Generate span with exponential distribution of mean 1.

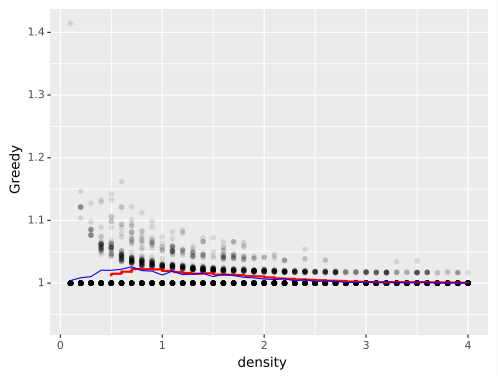
Questions

- What time horizon should we choose?
- What would be the worst density?
- How are the algorithms performing with respect to the theoretical upper bound?
- Tuning BarelyRandom: Could another distribution over Greedy and Commit be better?

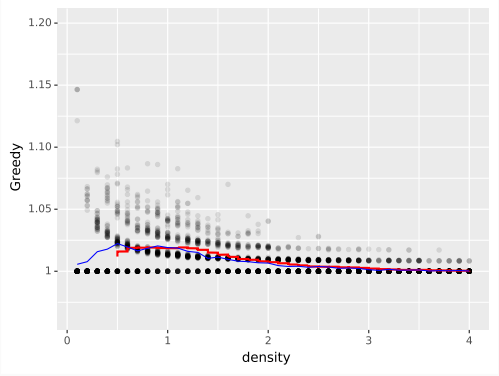
Which horizon suits?

Here $\alpha = 1 + \sqrt{2}$, lower bound distribution. Points = random instances (horizontally=density, vertically=competitive ratio).

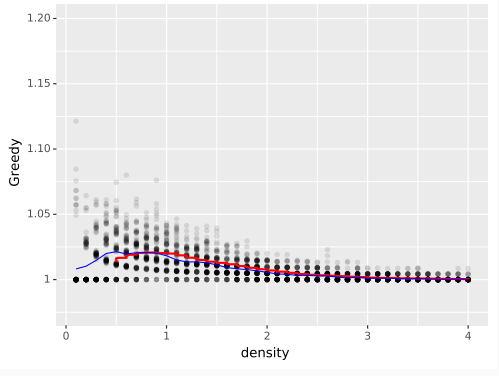
horizon = 25



horizon = 50



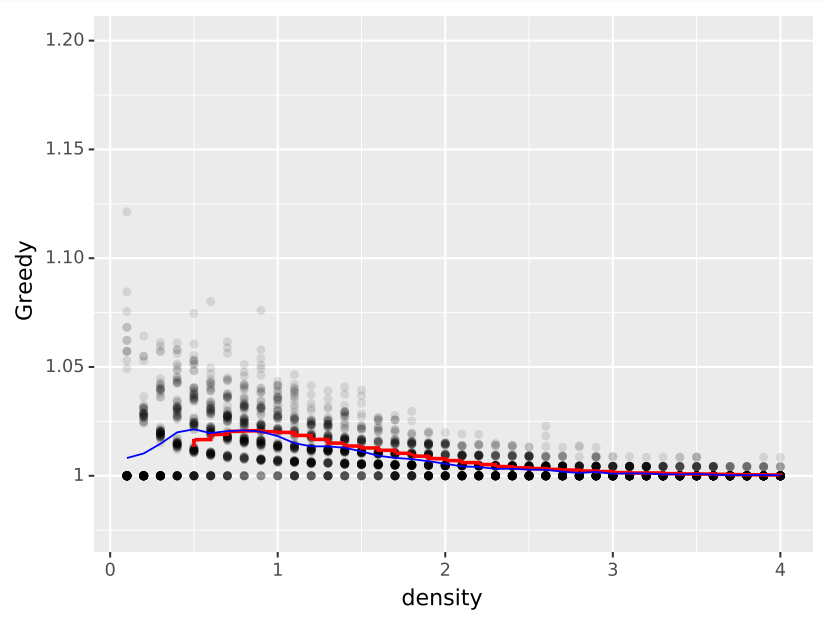
horizon = 100

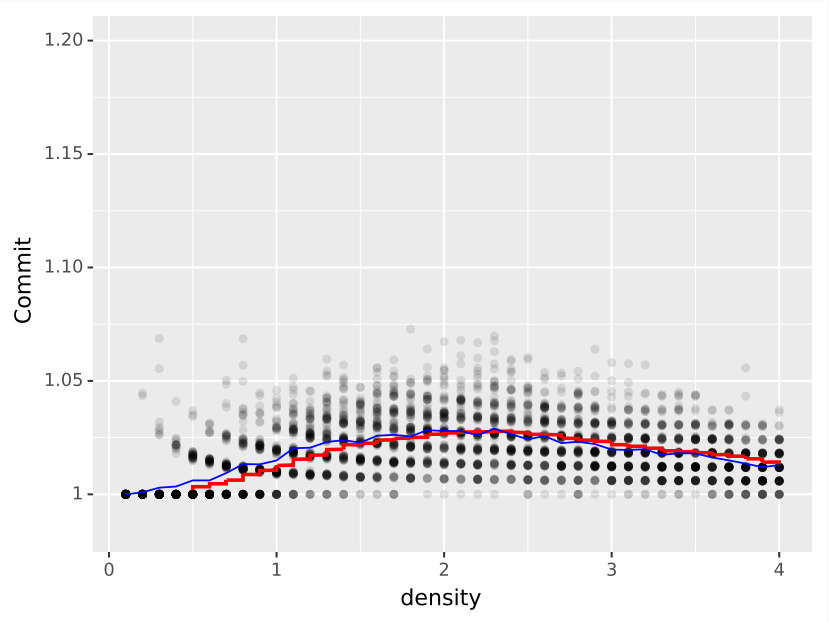


Every point is an instance (x=density, y=ratio), in blue the average ratio per density. The algorithms perform differently at the beginning of the instances, since no jobs are pending yet, thus the dependency on the time horizon.

Conclusion: Horizon 100 seems to smooth out enough the “stripes”.

Which density suits?

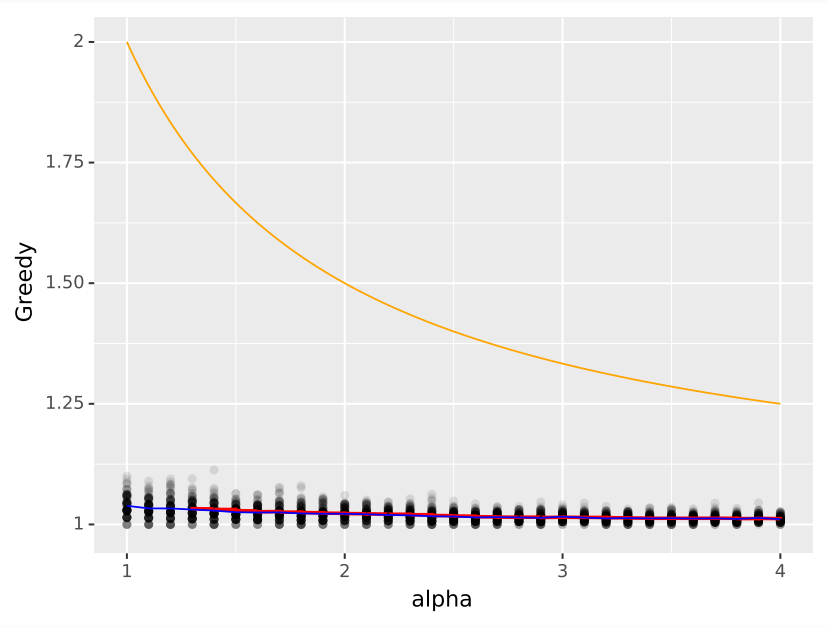


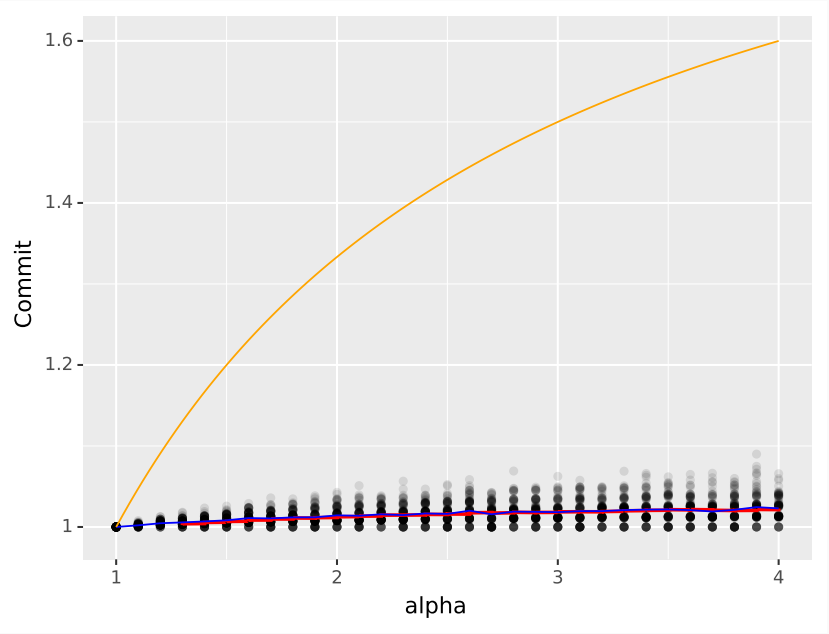


Blue=Average, Red=Smoothed

Conclusion: no δ is worst case for both algorithms. Hence we decide for $\delta = 1$ for simplicity.

What ratio do the algorithms achieve?

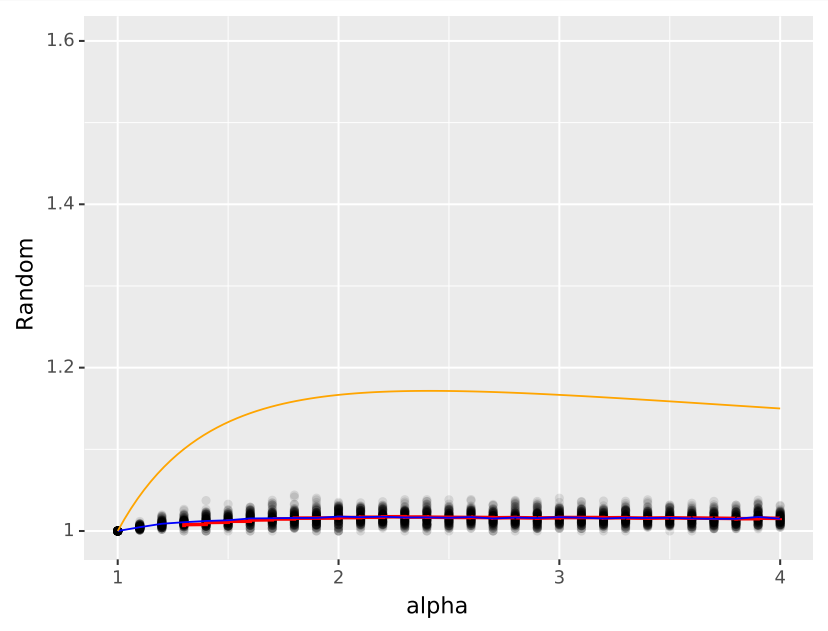




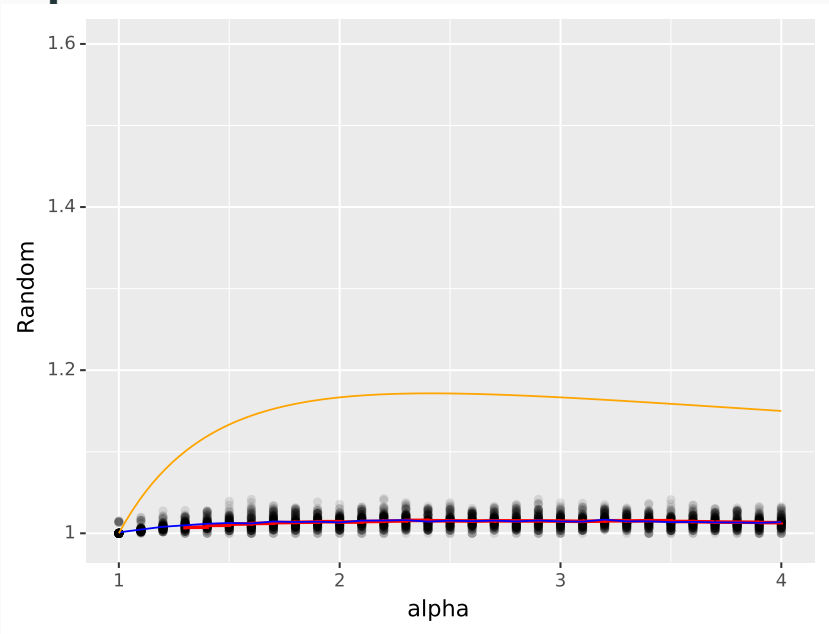
Measured competitive ratio has similar dependency in α as the theoretical upper bound (in orange), but is quite better.

What ratio does BarelyRandom achieve?

Lower bound distribution



Exponential distribution



Conclusion

- for practical purposes, one could imagine working in phases, using multiplicative weight update to tune probability of choosing Greedy or Commit during each phase.

Conclusion

- for practical purposes, one could imagine working in phases, using multiplicative weight update to tune probability of choosing Greedy or Commit during each phase.
- Next step: consider arbitrary weights between 1 and α.

Conclusion

- for practical purposes, one could imagine working in phases, using multiplicative weight update to tune probability of choosing Greedy or Commit during each phase.
- Next step: consider arbitrary weights between 1 and α.
- Eventually determine the randomized competitive ratio, by extending the optimal deterministic algorithm

Conclusion

- for practical purposes, one could imagine working in phases, using multiplicative weight update to tune probability of choosing Greedy or Commit during each phase.
- Next step: consider arbitrary weights between 1 and α.
- Eventually determine the randomized competitive ratio, by extending the optimal deterministic algorithm
- *Thank you*