The dynamics of backfilling: Solving the mystery of why increased inaccuracy may help…

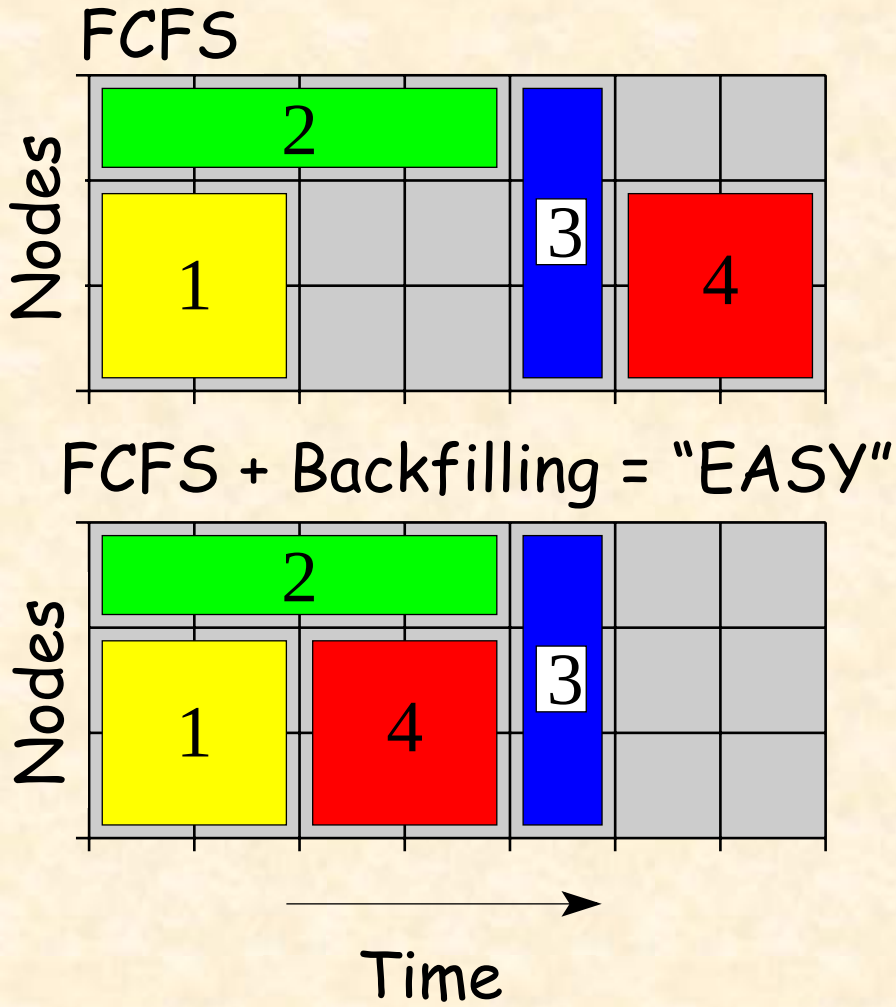
Dan Tsafrir* Dror G. Feitelson

The Hebrew University of Jerusalem

*(*current affiliation: IBM Watson Research Center)*

Supercomputer scheduling

- FCFS
  - Causes fragmentation
- The "**backfilling**" optimization
  - Can jump over 1st queued job if not delaying it



Backfilling: Pros

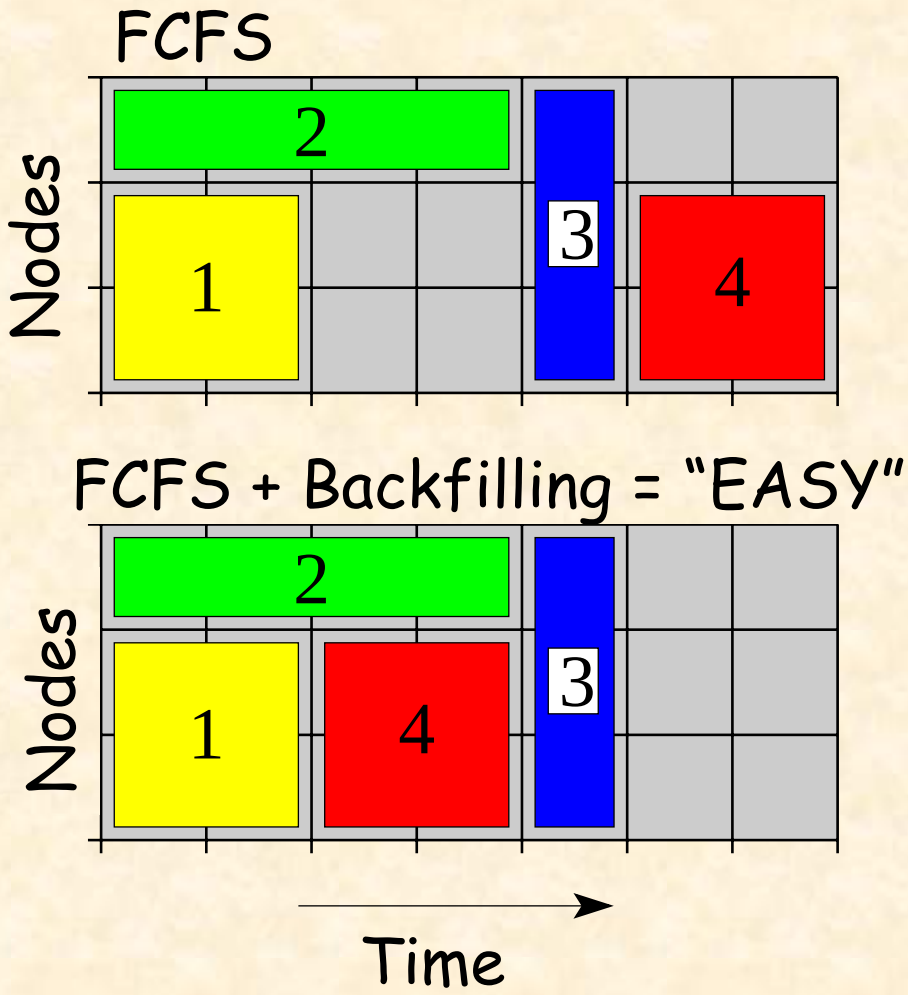
- Simple and easy
  - For users to understand ("it's FCFS with…")
  - For developers to implement
- Batch
  - Scientific apps. often tailored to use entire memory
  - No need to co-schedule
- Significantly improves performance
  - Utilization (from 40-60% to 70%), throughput, response time,…
- Comparable to more sophisticated alternatives
  - Involving preemption & migration

Pros' consequences

- Backfilling is very popular in production systems
Company	*product*	
	commercial	*free*
IBM	LoadLeveler	
Cluster Resources	Moab	Maui
Platform	LSF	
Altair	PBS-Pro	OpenPBS
Sun	GridEngine	
  - 60% of the top-500 use backfilling
- And the focus of many research efforts
  - Dozens of variations
  - Dozens of papers

Backfilling: The price

- Need to know jobs' runtimes **a-priori**
- Thus, users must provide **estimates** of how long their jobs will run
- Jobs attempting to exceed their estimates are **killed**



The *f*-model

Notation	$f \geq 0$	predefined "**badness factor**"
	R	**input**: a job's runtime
	E	**output**: the job's estimate

***f*-model**	random	E uniform in $[R, (f+1) \cdot R]$
	deterministic	$E = (f+1) \cdot R$

Intuition	$f = 0$	complete accuracy $(E=R)$
	increased f	increasingly inaccurate estimates

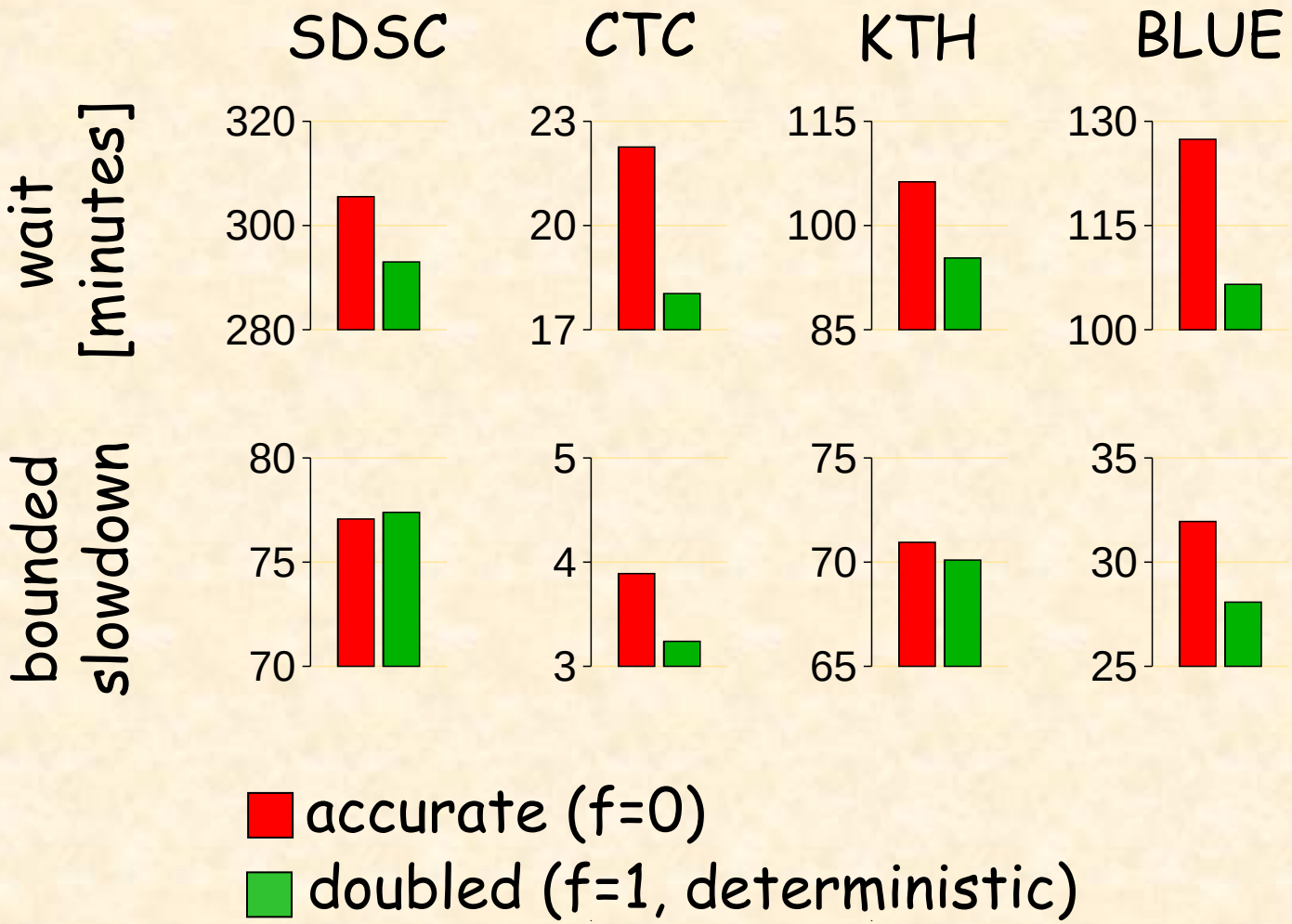
Methodology: Using the f-model

- **Logs**: from the "Parallel Workload Archive"
  - We use four (SDSC, CTC, KTH, and BLUE)
  - 10s to 100s of thousands of jobs, spanning 1-3 years
  - Replacing user-estimates with model-values

job ID	*arrival time*	*size [CPU#]*	*runtime*	*estimate*
1	Aug 24, 12:00:01	2	00:15:37	00:30:00
2	Aug 24, 12:05:37	128	01:50:01	02:00:00
3	Aug 24, 13:25:20	49	18:00:00	18:00:00
...	...	...	...	...

- **Simulator's input**: the modified workload
- **Simulator's output**: performance
  - avg. wait-time and slowdown

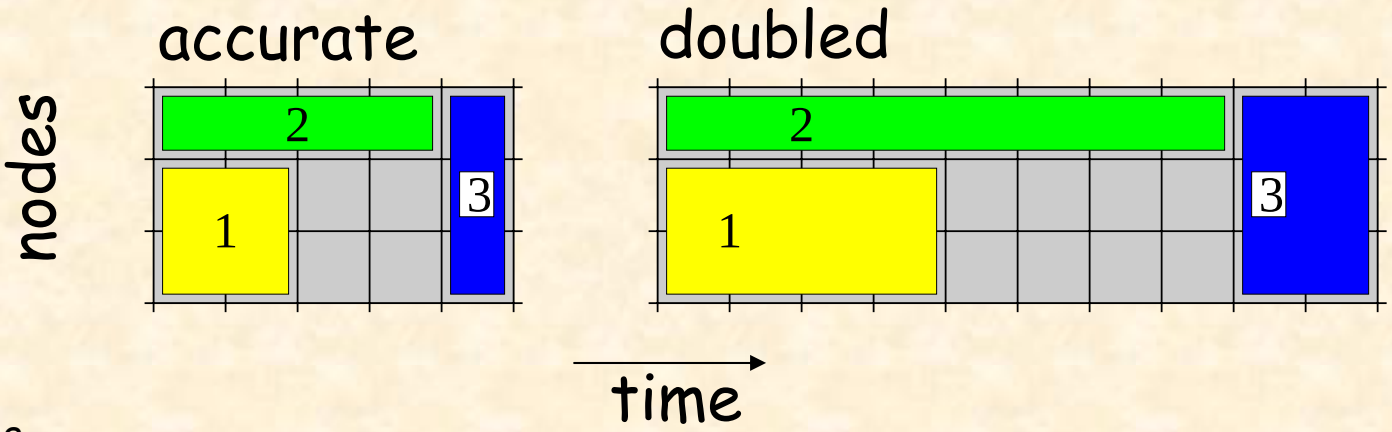
Claim 1:
Inaccuracy improves performance



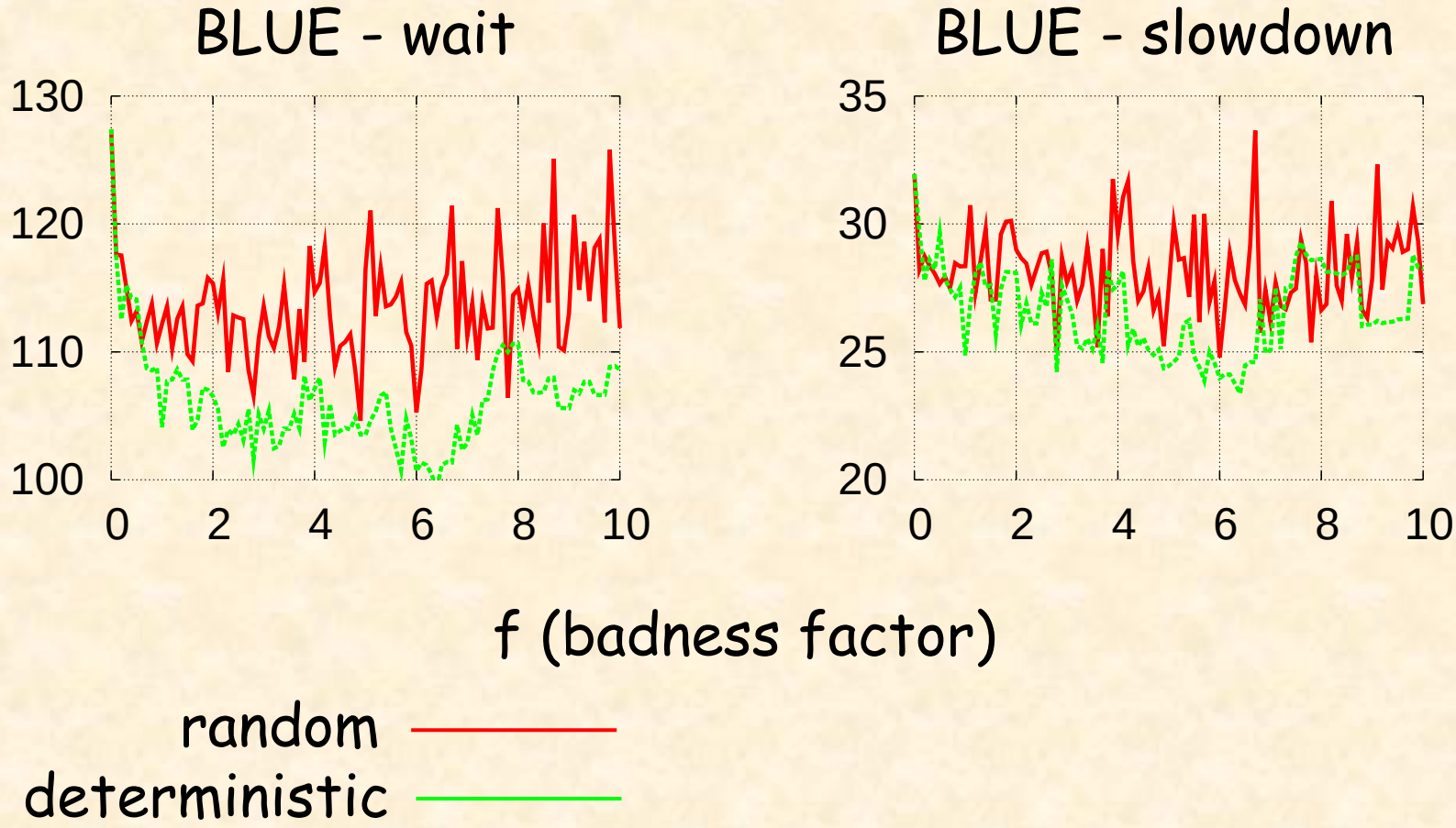
Explanation 1: The "holes argument"

S. Chiang, A. Arpaci-Dusseau, and M. Vernon [JSSPP, 2002]:

> *"...when multiplying estimates by two, job with long runtimes can have very large overestimation, which leaves larger holes for backfilling shorter jobs.*
>
> *As a result, average slowdown and wait may be lower."*



Claim 2:
Performance is independent of *f*

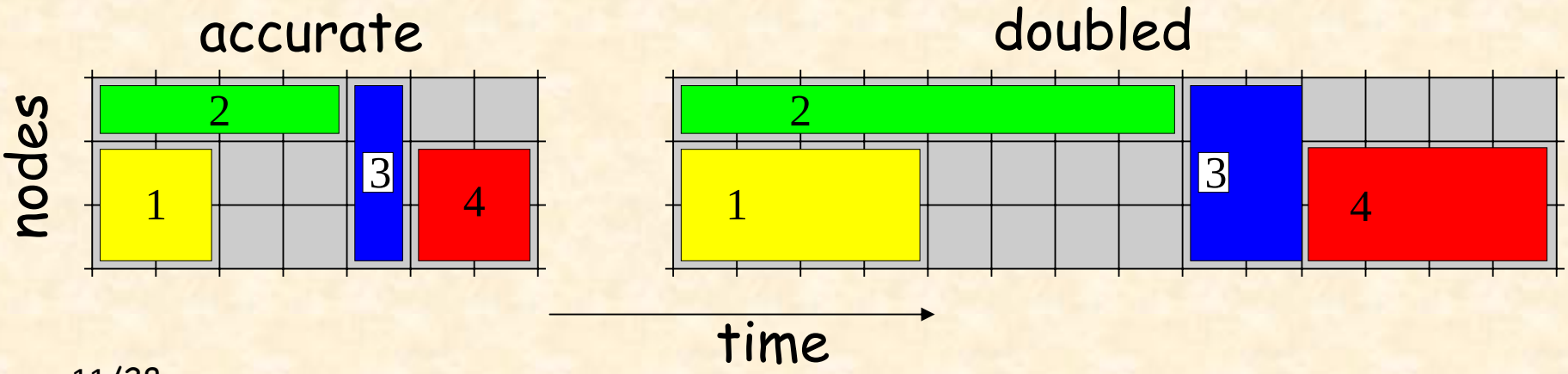


Explanation 2: The "balance argument"

Y. Zhang, H. Franke, J. Moreira, and A. Sivasubramaniam [IPDPS, 2000]:

"On average, overestimation impacts both the jobs that are running and the jobs that are waiting…

…the probability of finding a backfilling candidate effectively does not change with the overestimation."



The robustness claim

D. England, J. Weissman, and J. Sadagopan [HPDC, 2005]:

"…Our results support those of previous work and also indicate that backfilling is robust to inaccurate estimates in general.

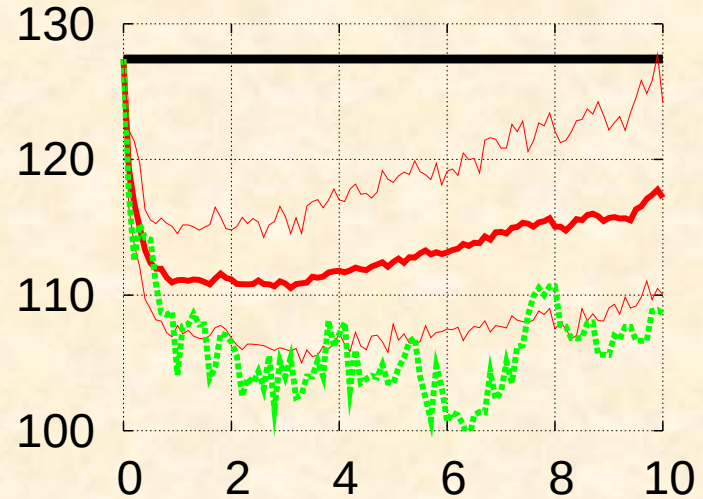
It seem that, with respect to backfilling, what the scheduler doesn't know won't hurt it."

Intermediate summary

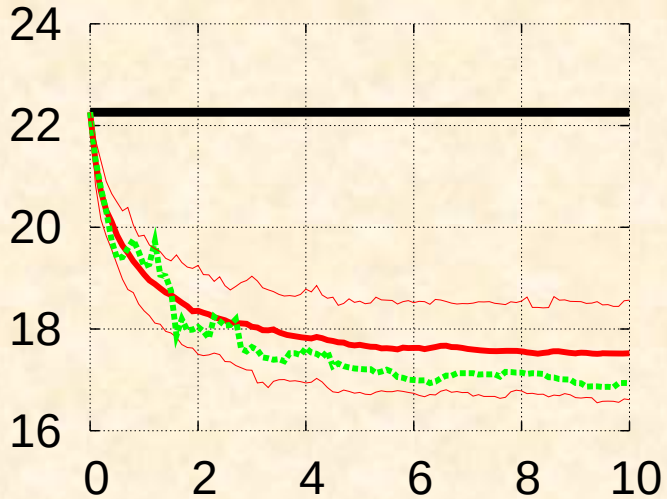
- Two observations
  - Inaccuracy improves performance
  - Inaccuracy doesn't affect performance
- Two contradictory explanations
  - The holes argument
  - The balance argument
- One mystery
  - Improved accuracy should result in better packing
  - How come the opposite is true?

Using mean & confidence exposes a clear trend

BLUE: V-**shape** (the norm)



CTC: **L-shape**

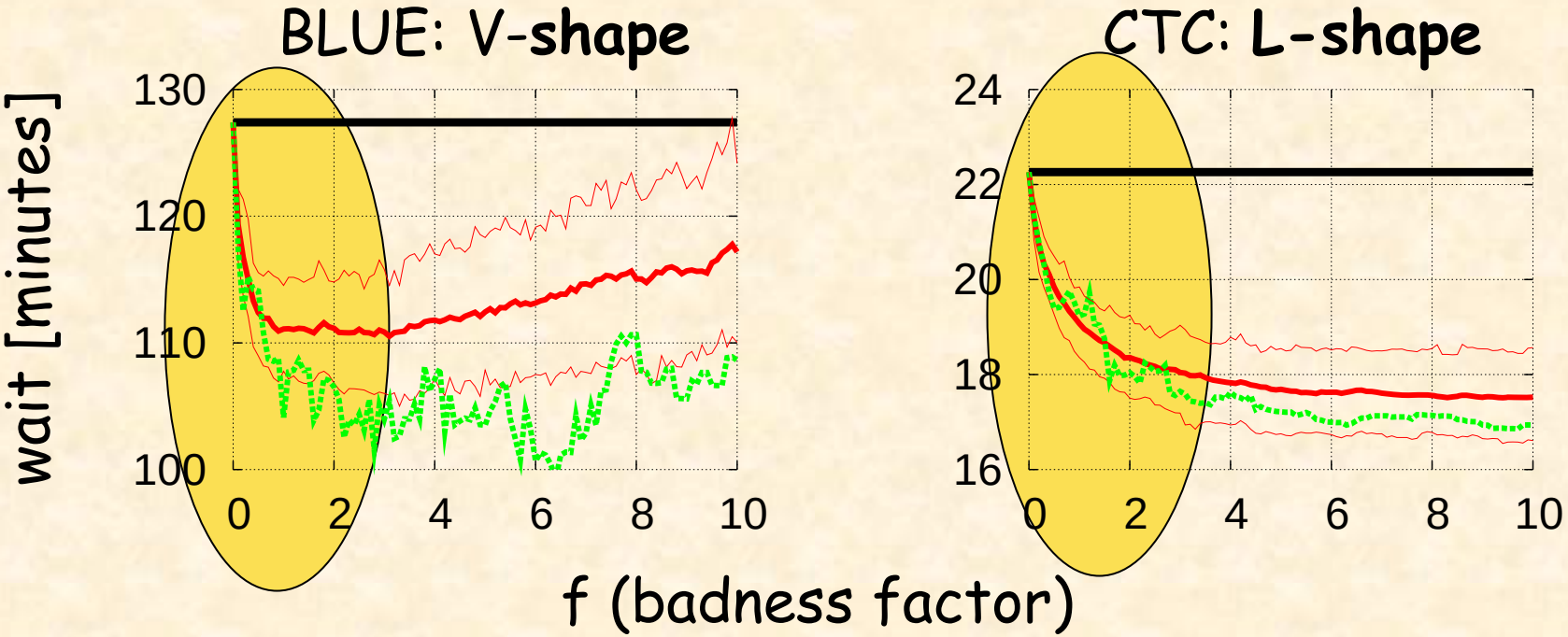


f (badness factor)

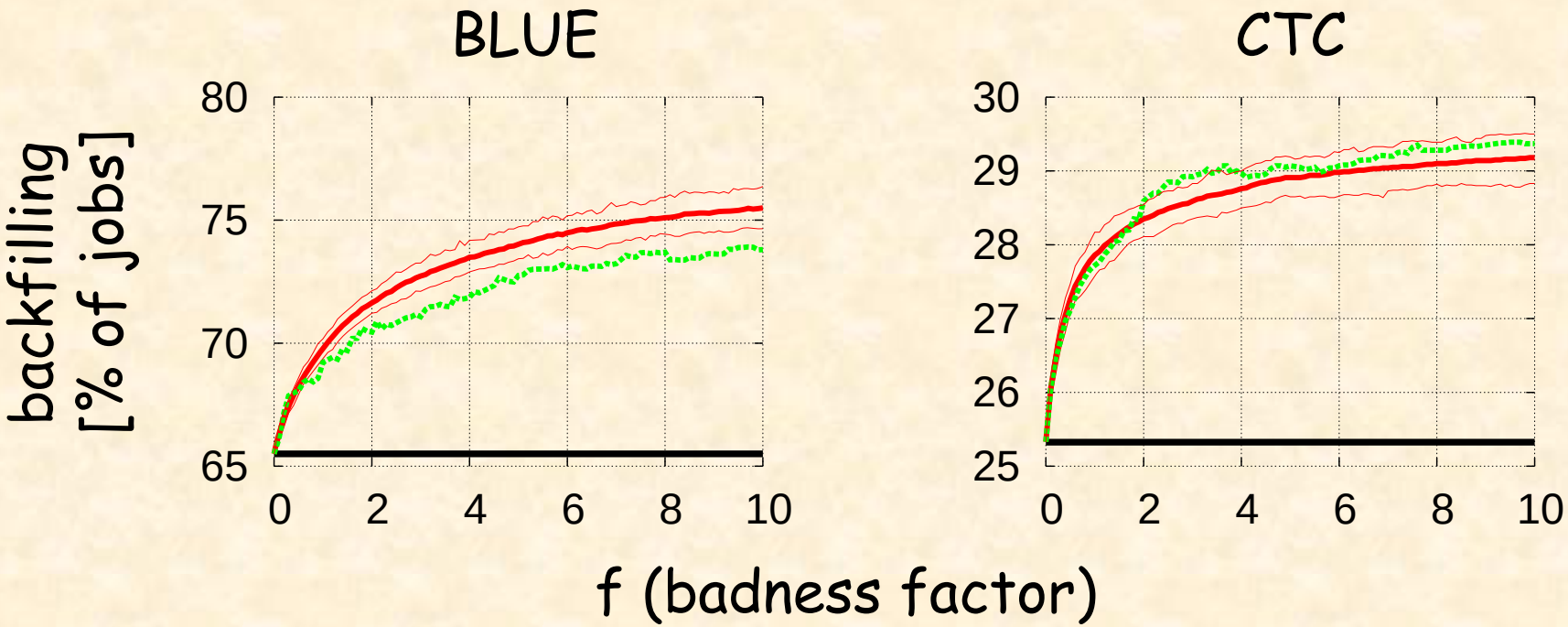
E is uniform in *[R, (1+f) · R]*

f=0
random (mean)
random (90% confidence)
deterministic

Control: The descending part



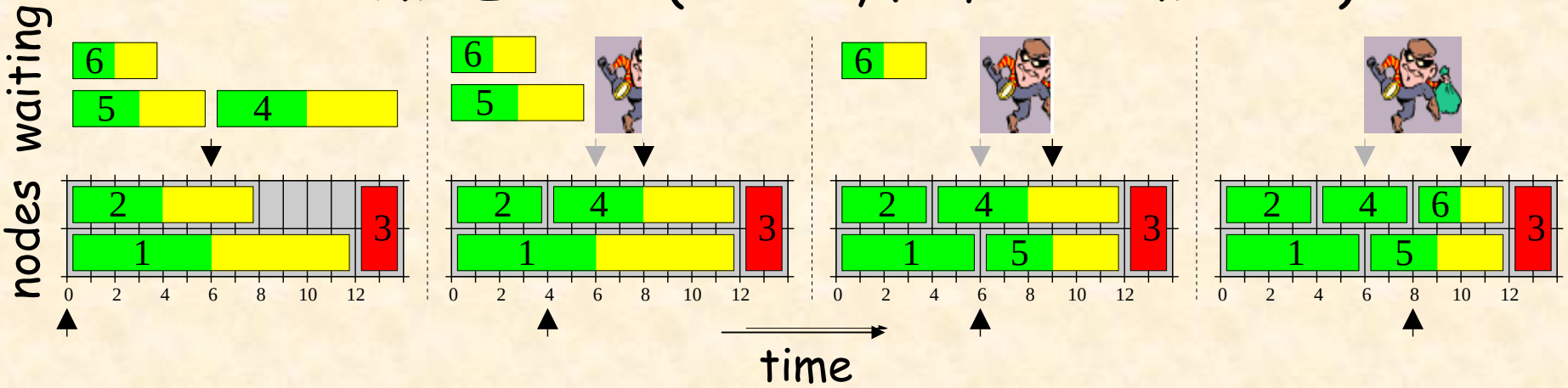
What's going on? Balance? Holes?

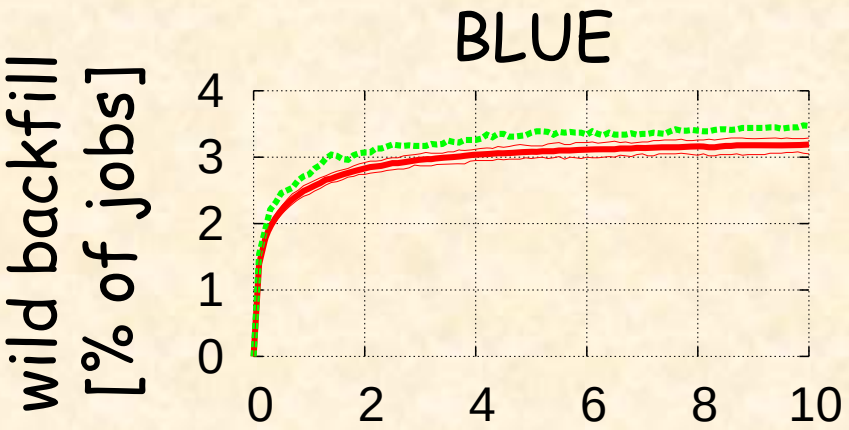


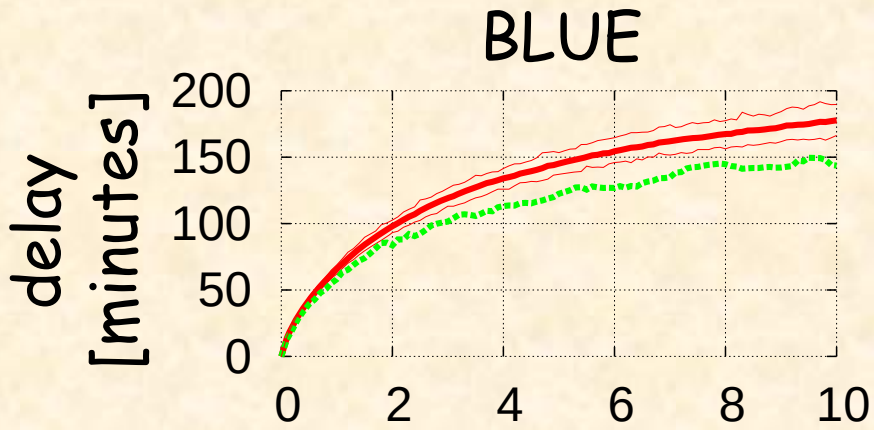
certainly more backfilling activity

"Heel & toe" dynamics

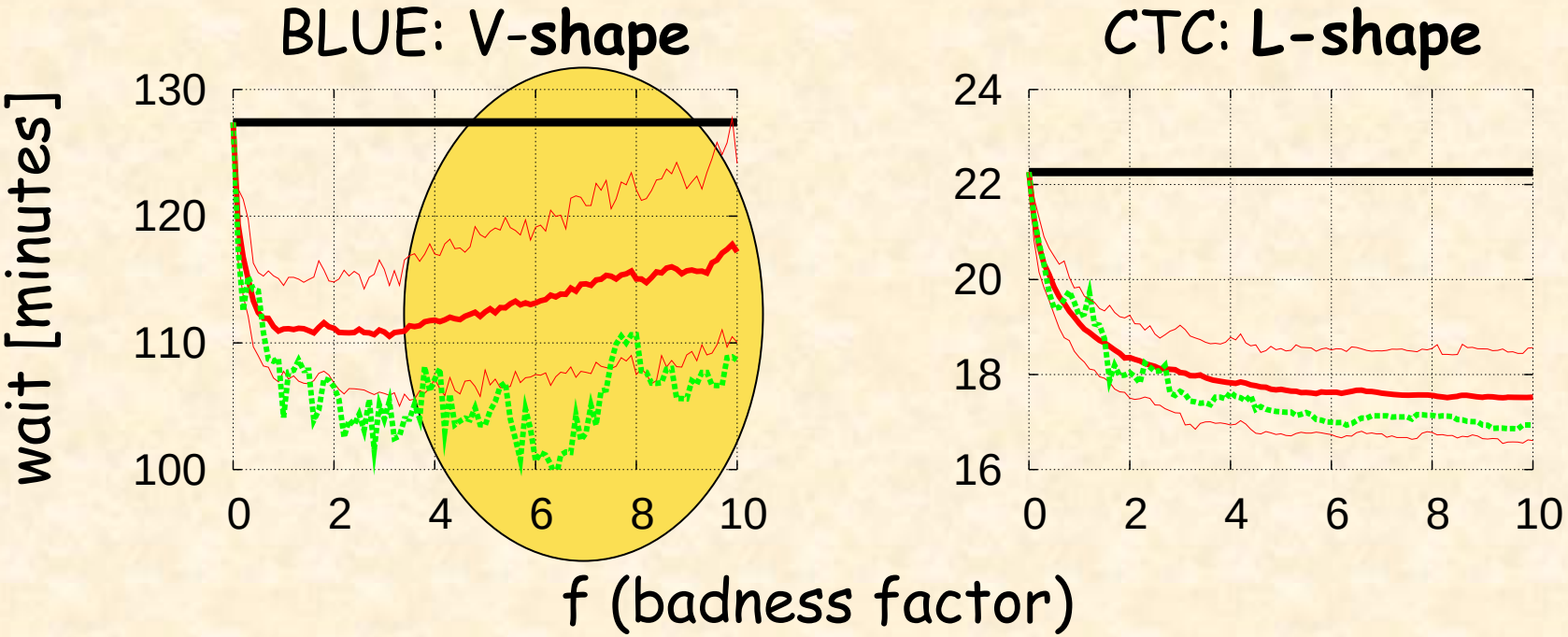
Assume $E = 2R$ (that is, f=1; deterministic)



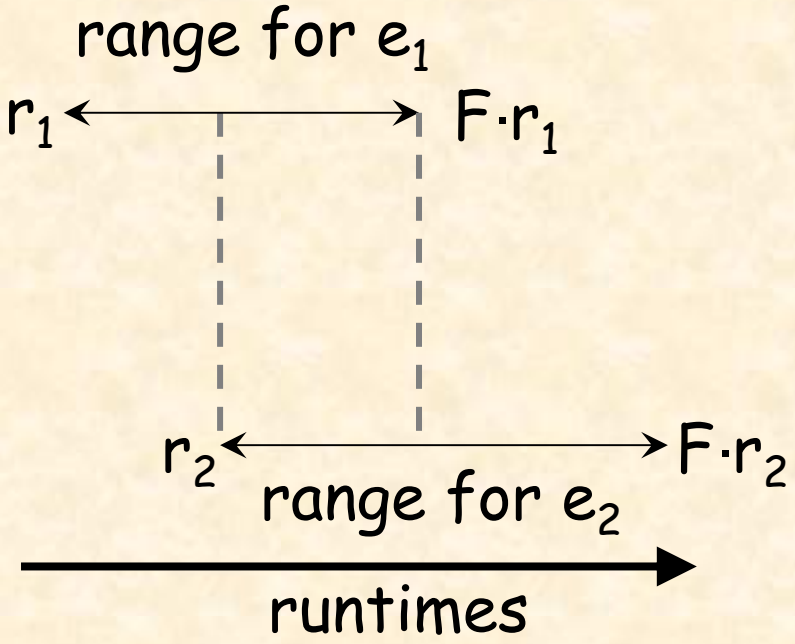


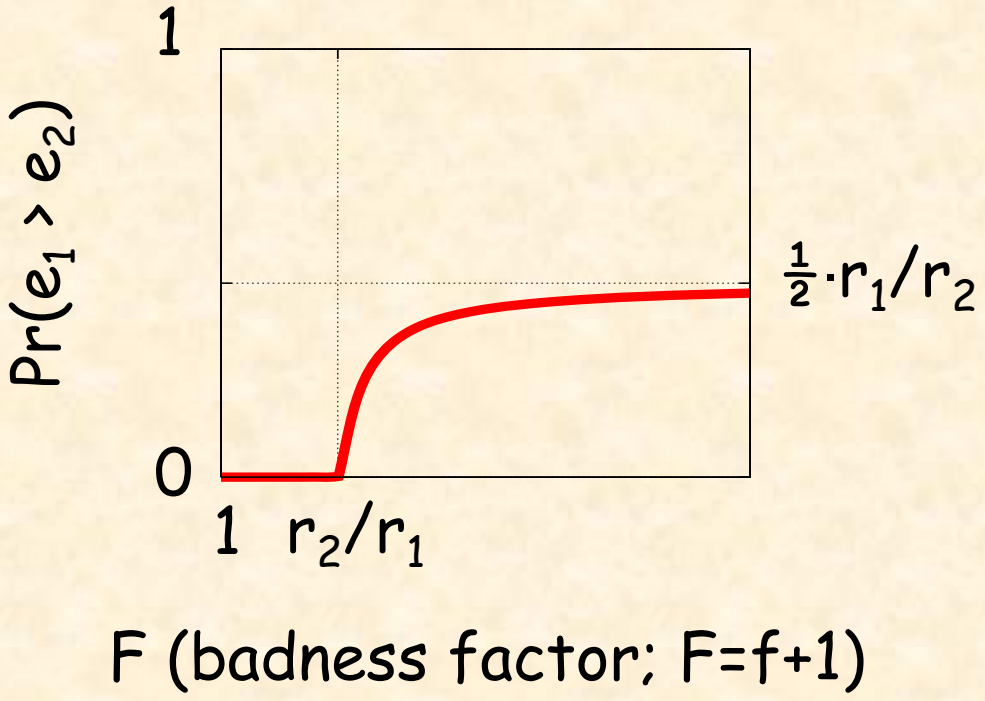


Control: The ascending part



Bigger f => more long jobs masquerade as short & vise versa

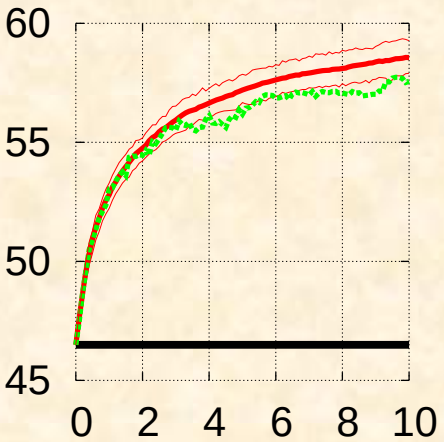




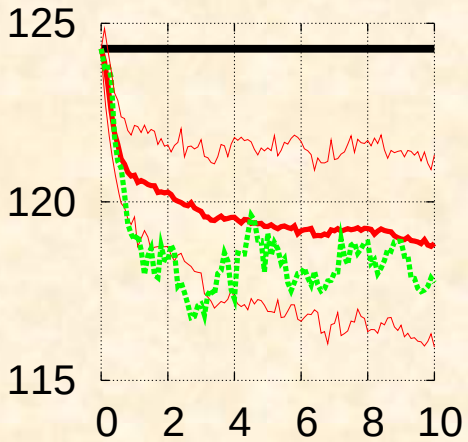
The probability the scheduler is erroneously told j_1 is longer than j_2, is monotonically increasing with f

Bigger *f* => wider holes => longer jobs enjoy backfilling

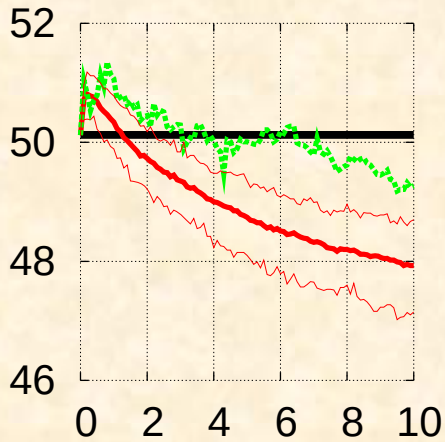
runtime of backfilled



runtime of non-backfilled



SJFness (% of jobs)



f (badness factor)

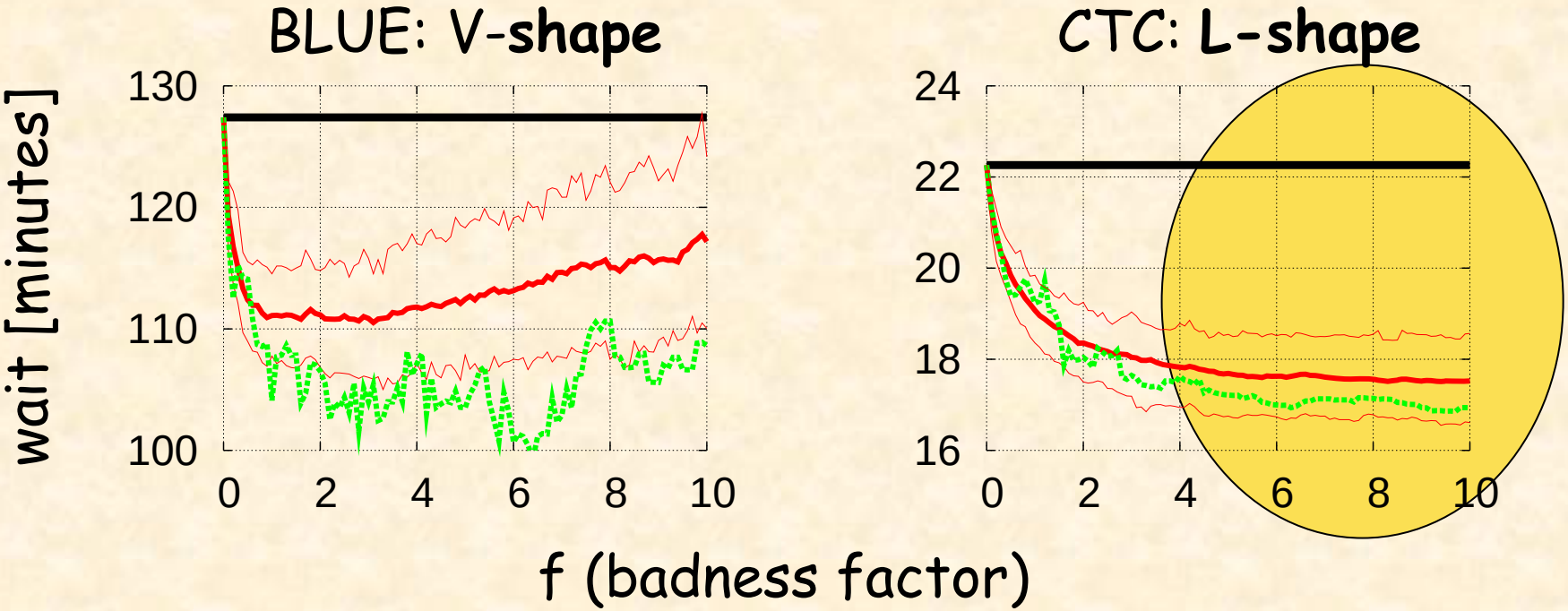
f=0

random (mean)

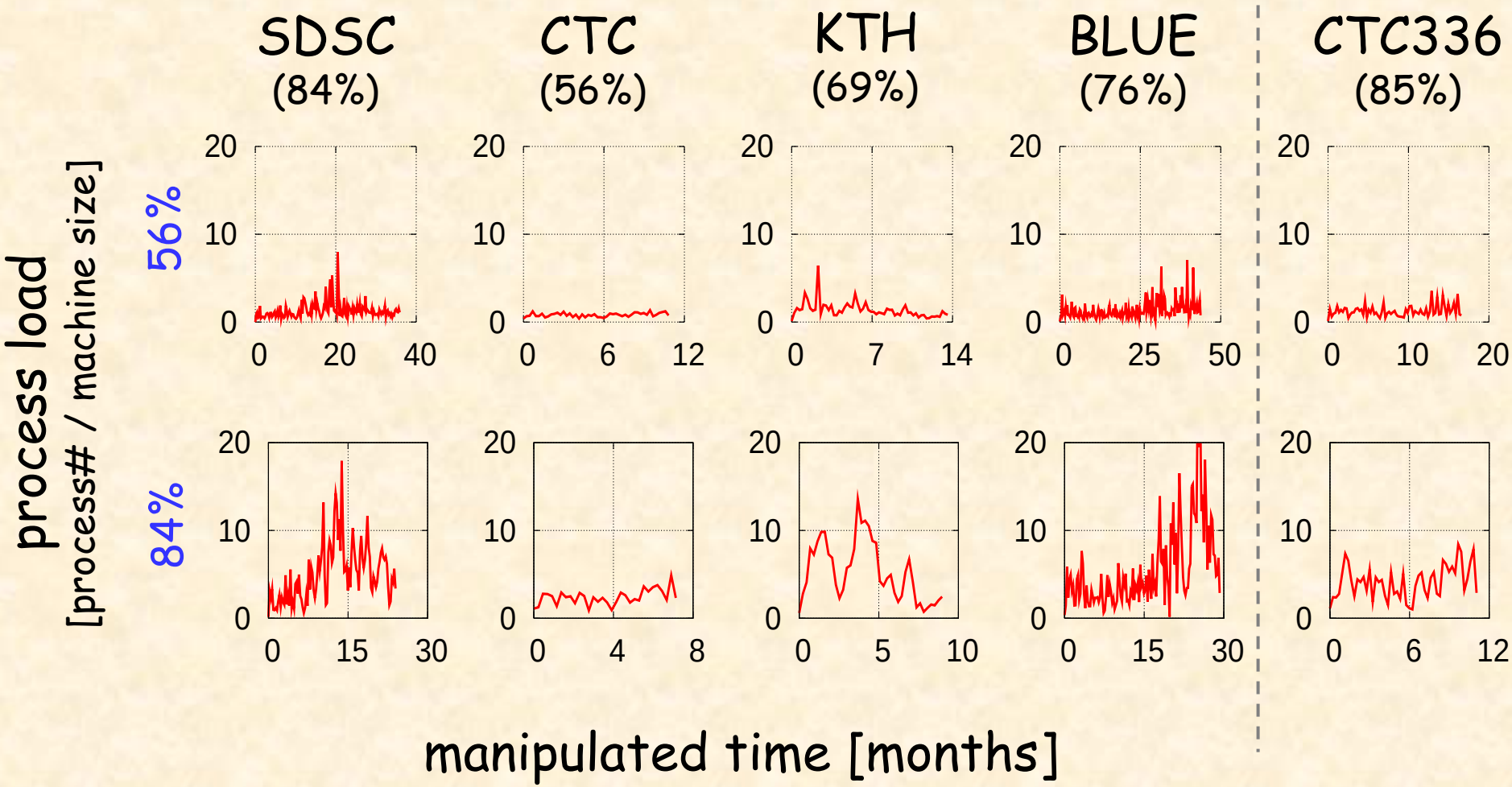
random (90% confidence)

deterministic

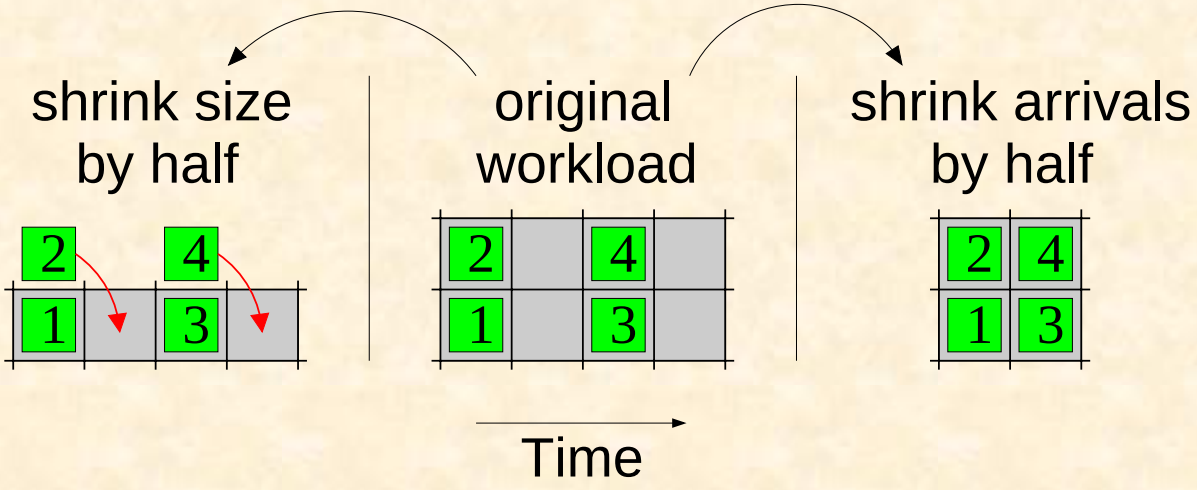
Control: Why is CTC different?

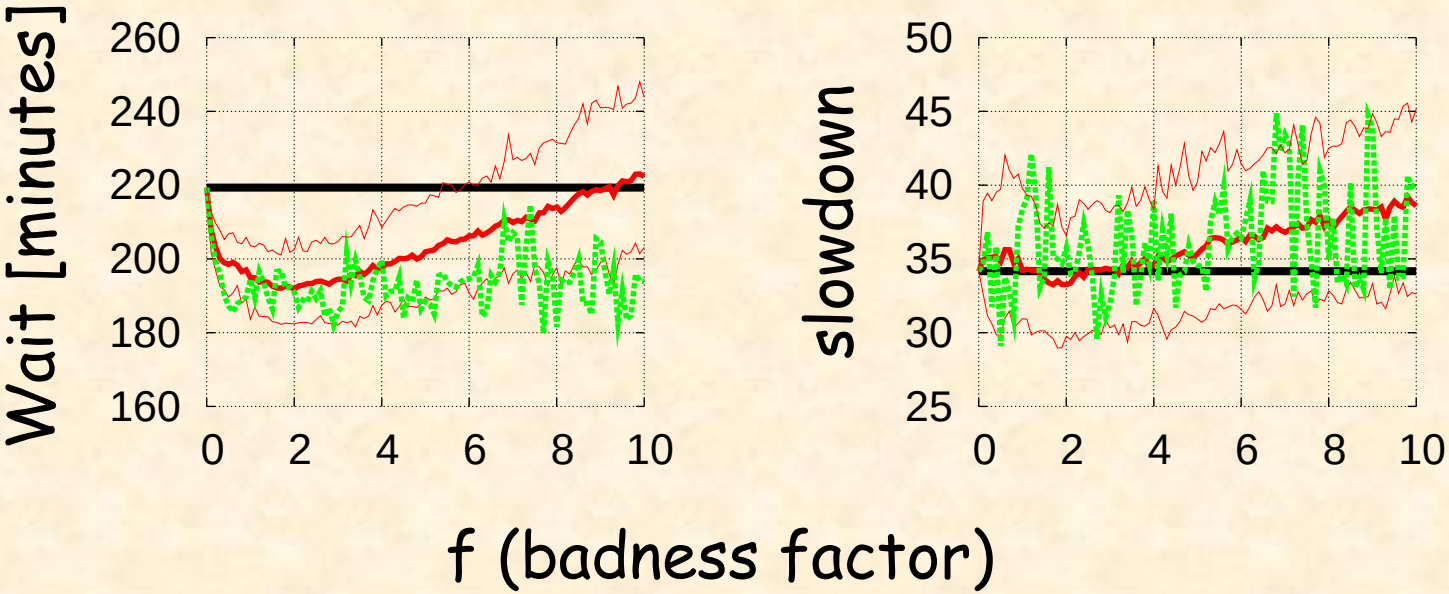


The role of burstiness



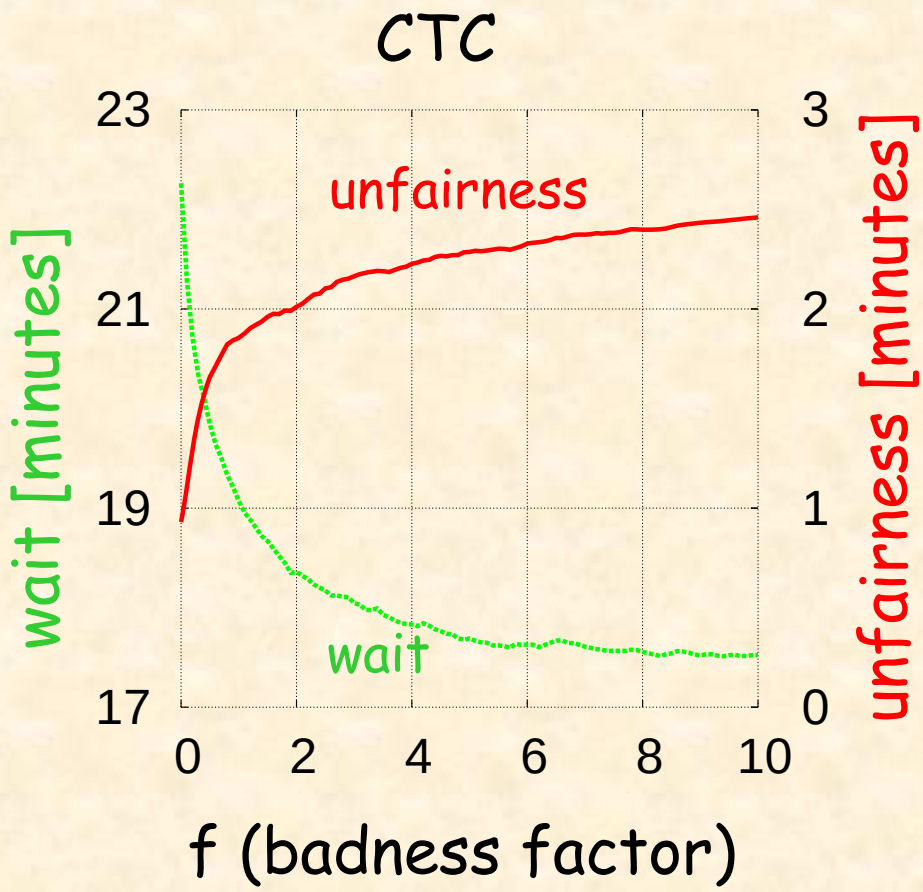
CTC-336


shrink size
by half
original
workload
shrink arrivals
by half
2
4
1
3
Time


Wait [minutes]
260
240
220
200
180
160
0
2
4
6
8
10
slowdown
50
45
40
35
30
25
f (badness factor)

Fairness/performance tradeoff

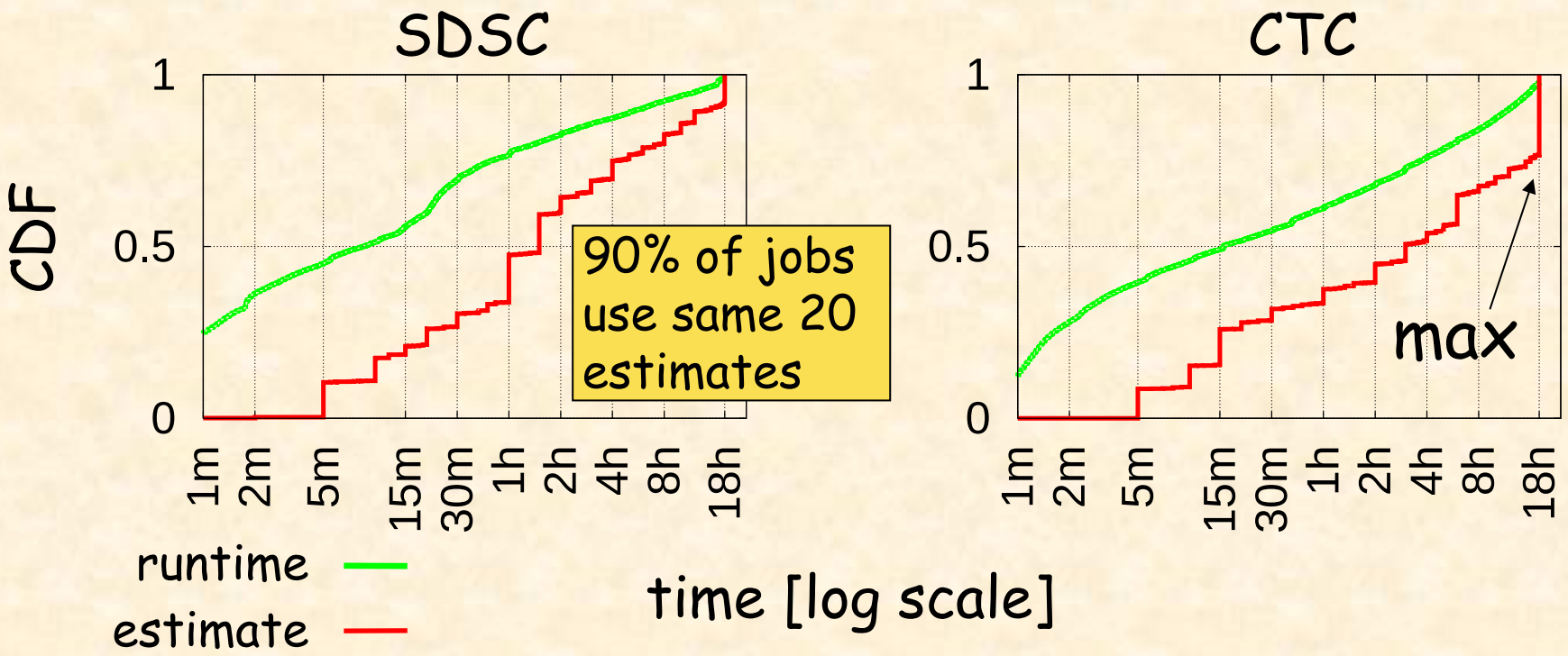
- "Unfairness" is the avg. delay of jobs beyond their "Fair Start Time"
- Jobs that start before that time contribute zero to the avg.
- **Multiplying by a factor simply means trading off fairness for performance**



Not just a theoretical result…

- The more accurate the value we multiply, the better the performance & fairness
- **Increased <u>accuracy</u> actually <u>improves</u> performance**

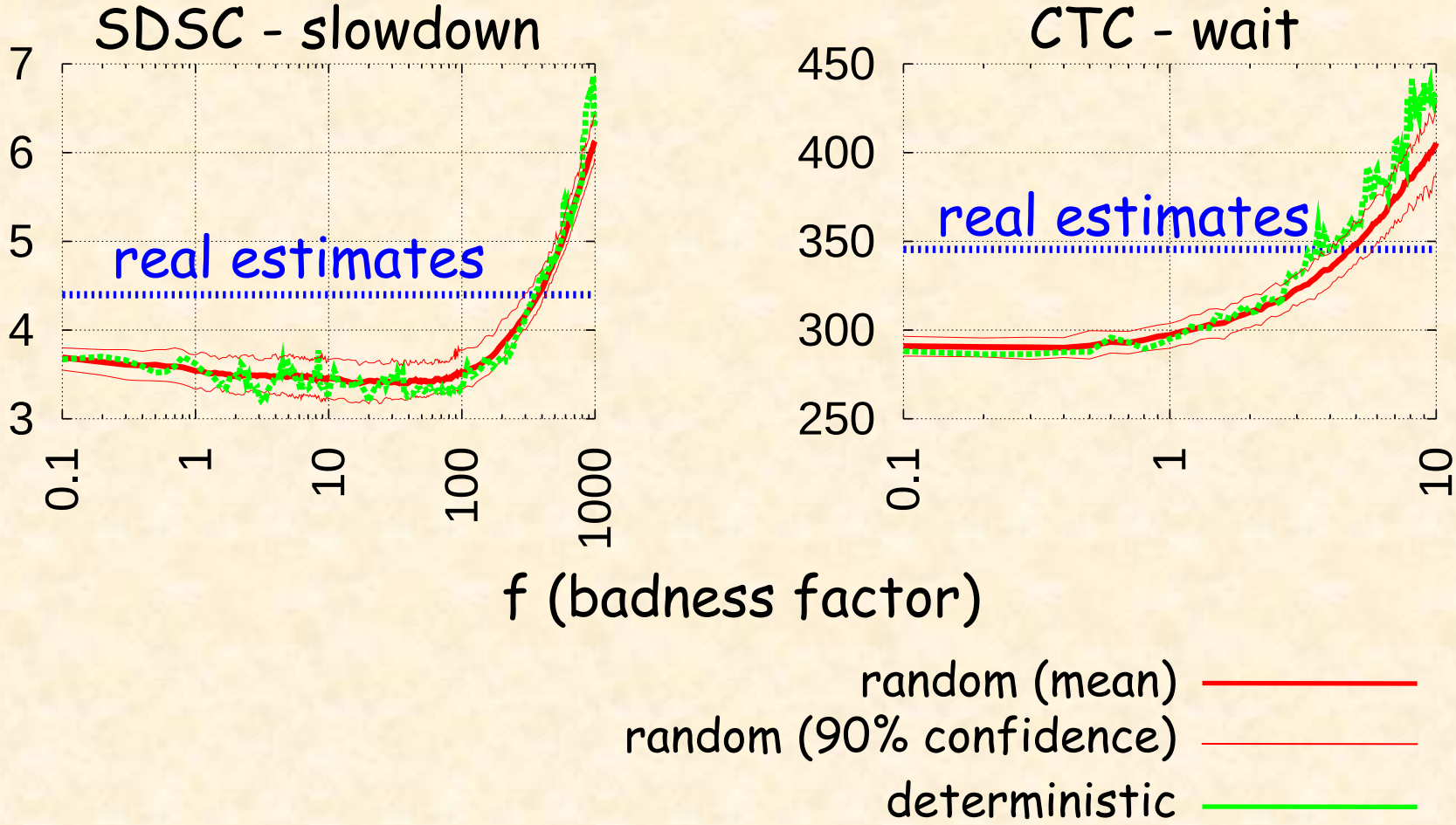
The source of users' badness: **Modality**



1. Modality => many "identical" jobs => bad scheduling info.
2. 'Max' is especially popular => such jobs never backfill !

=> **Increased <u>inaccuracy</u> means increased <u>modality</u>**

The truncated f-model:
min((f+1)·R , 'Max')



Conclusions

- Should distinguish between 2 inaccuracy types:

type	*property of*	*nature*	*performance*
real	users	modal and favors 'max'	worsened
artificial (f)	schedulers	promotes heel & toe	improves

- "Inaccuracy helps" is a myth: the *f*-model is
  - Inadequate to study the impact of **real** inaccuracy
  - Inadequate workload-model for performance eval.
- Need a realistic model
  - *"Modeling user runtime estimates" [JSSPP, 2005]*
- Have strong motivation to improve estimates
  - *"Backfilling with system predictions" [TPDS, 2007]*