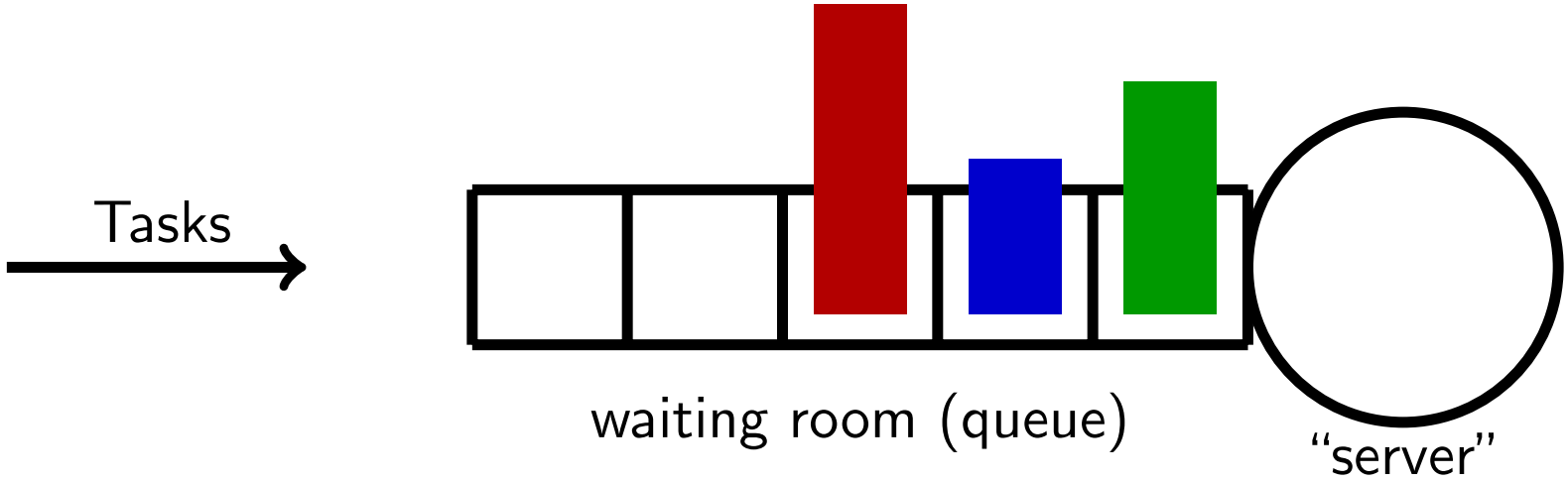
Stochastic Scheduling and Index Policies

Nicolas Gast

Inria

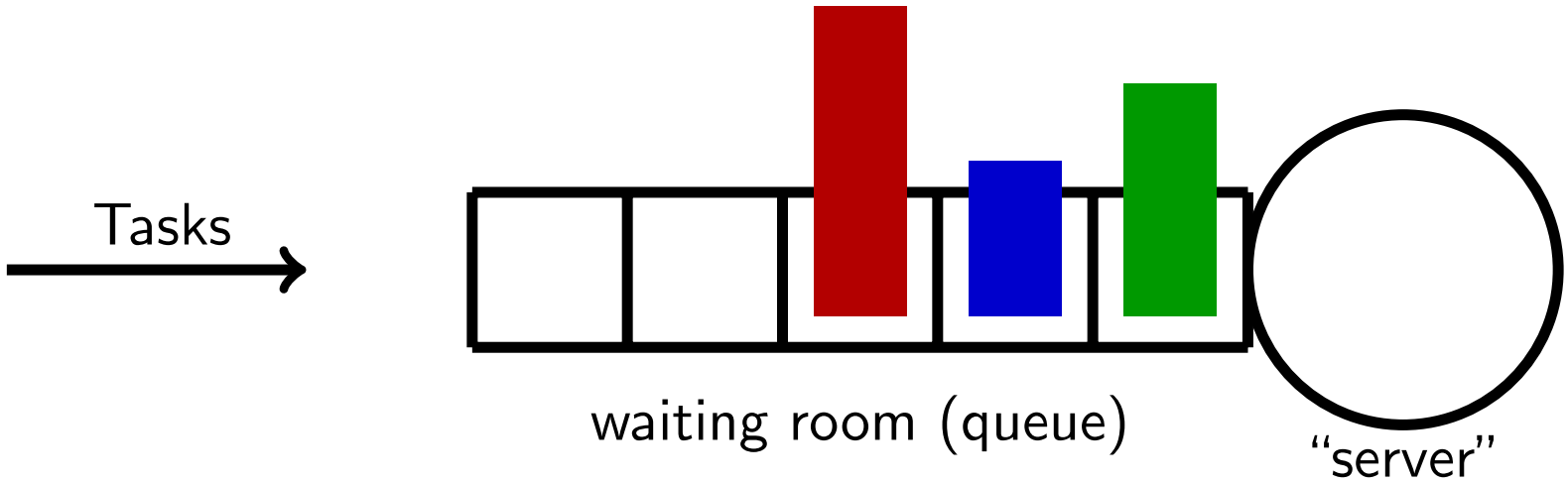
LIG WAX – May, 27, 2021.

Motivation: Single-server scheduling



- We allow preemptive scheduling (preempt-resume).
- Examples: tasks allocations, electric vehicle charging, wireless scheduling, research projects,...

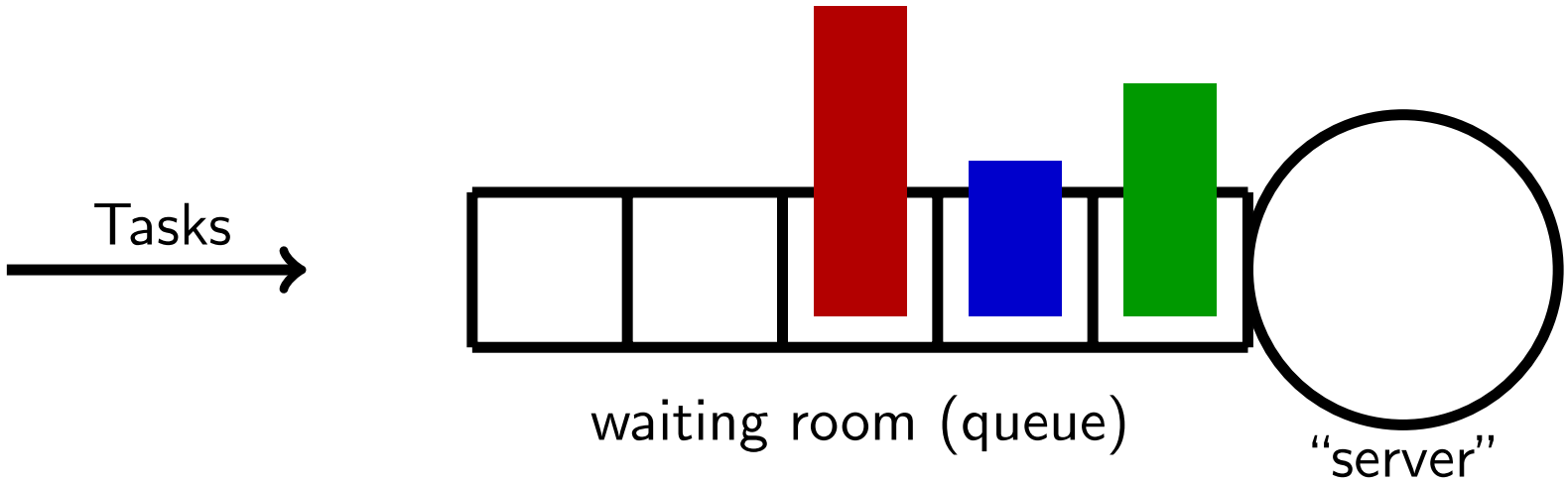
Motivation: Single-server scheduling



Focus of the talk

How to you schedule jobs to minimize the average response time?

Motivation: Single-server scheduling



Focus of the talk

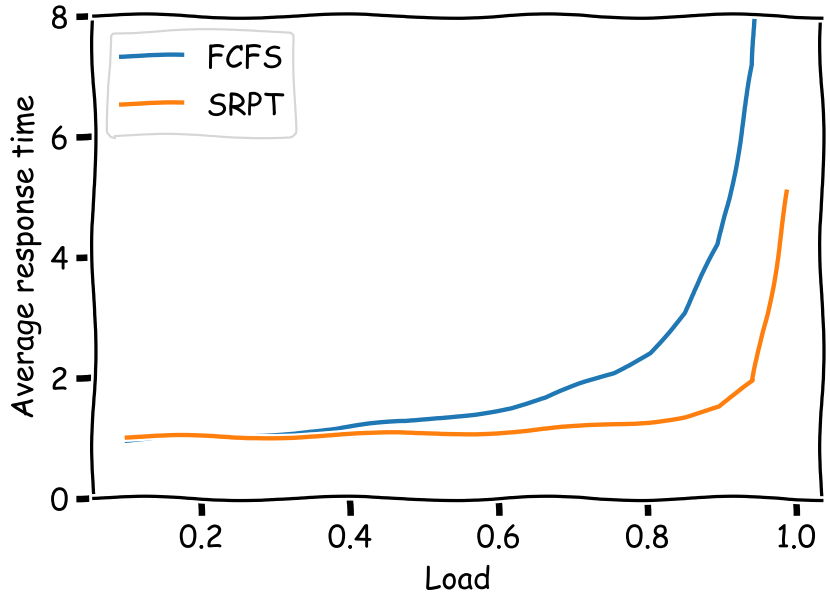
How to you schedule jobs to minimize the average response time?

If you know the job sizes: SRPT (Shortest Remaining Processing Time)

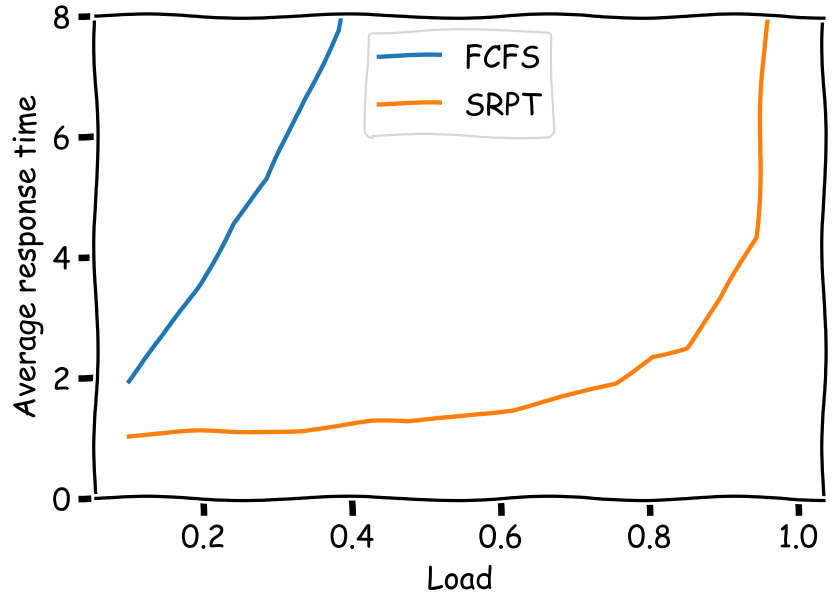
- "Strongly optimal" [Schrage, 1966]

Scheduling does matter

SRPT can lead to huge performance gain compared to FCFS



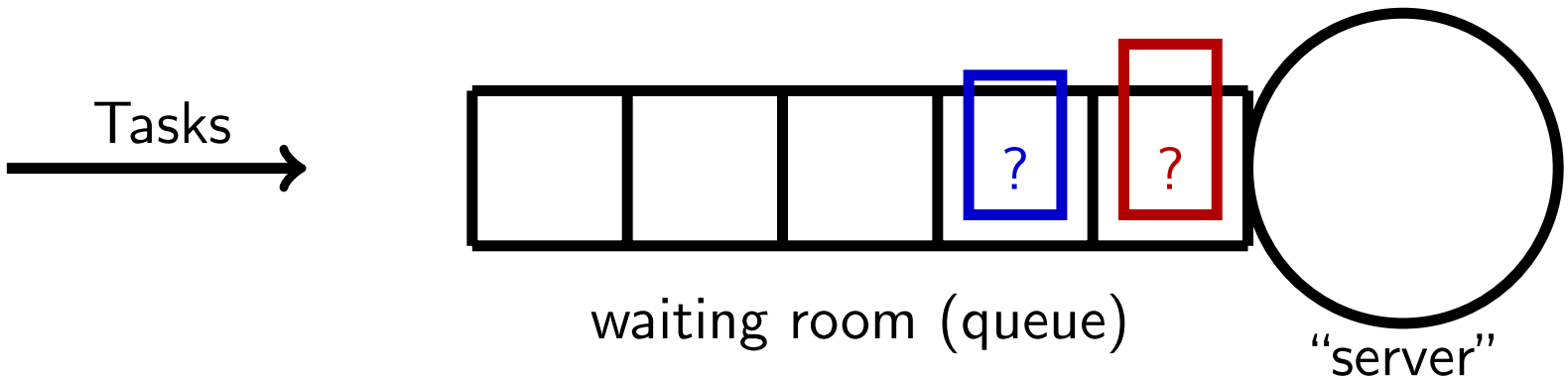
Low varying jobs
Job size $\in \{1, 10\}$
99% of small jobs.



High varying jobs
Job size $\in \{1, 100\}$
99% of small jobs.

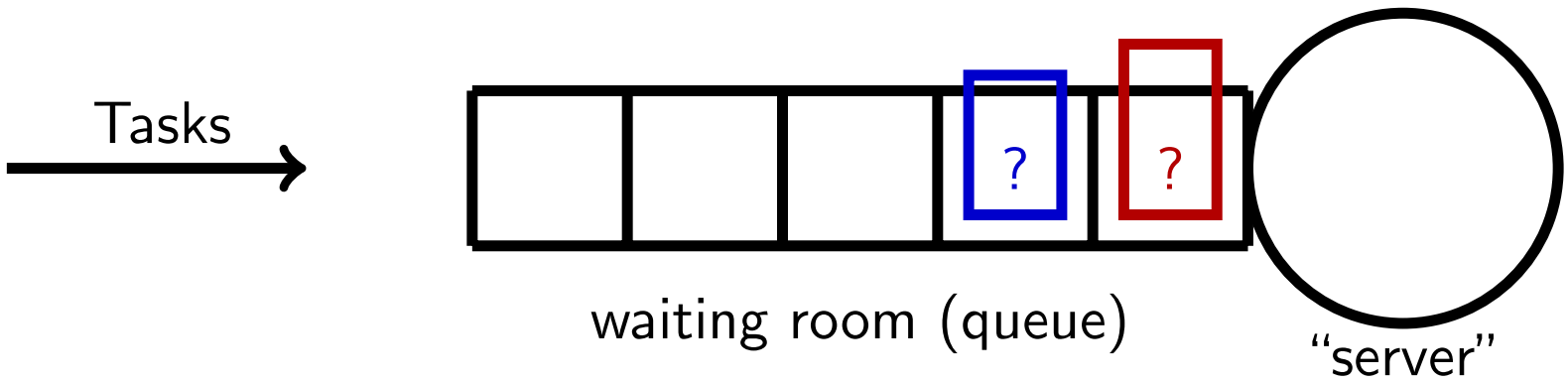
How to schedule if you do not know jobs sizes?

But you know their distribution



How to schedule if you do not know jobs sizes?

But you know their distribution

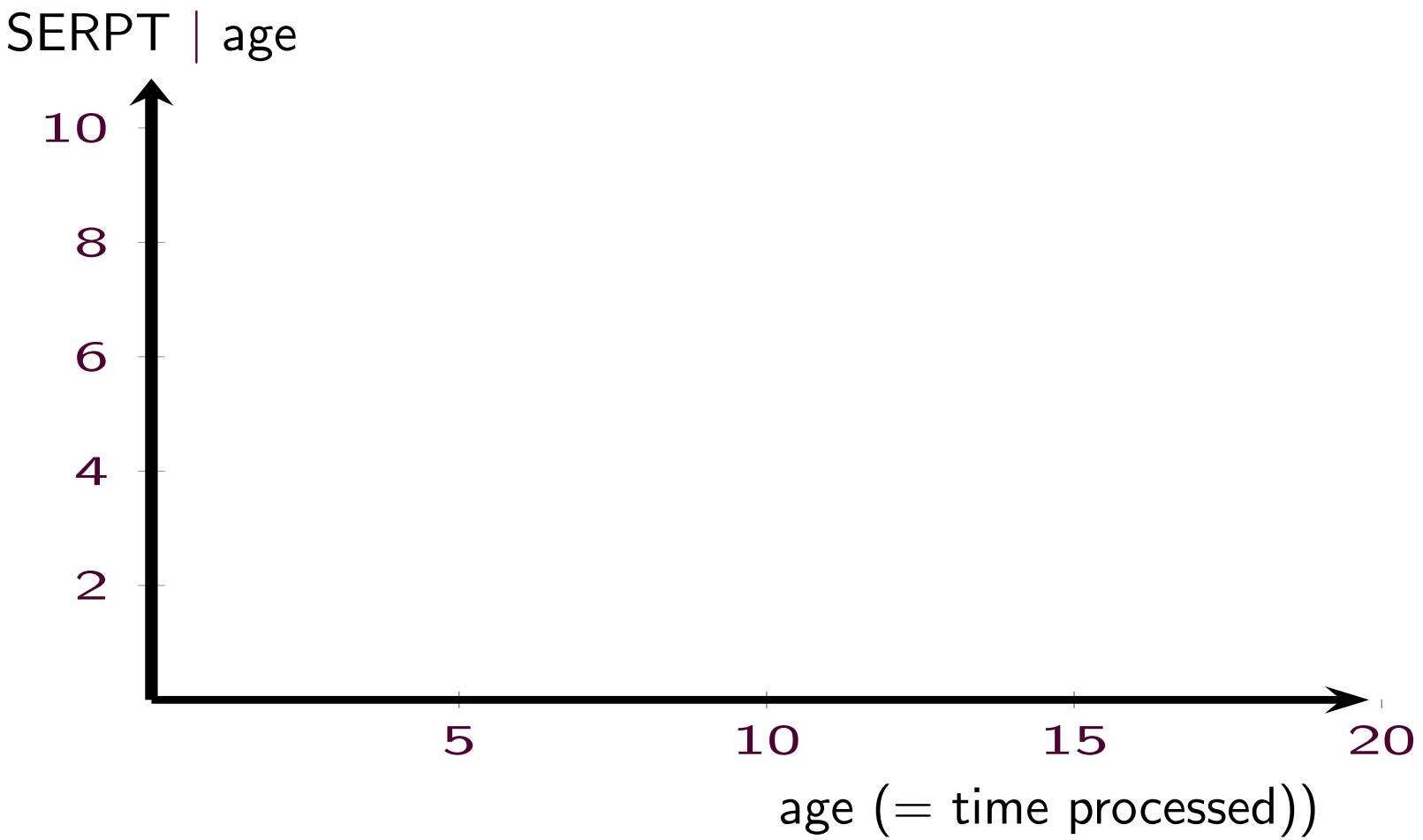


Tentative: SERPT : shortest **expected** remaining processing time.

Is it optimal?

SERPT is not optimal.

Example: two jobs with $X = 10 - \varepsilon$ and $Y = \begin{cases} 2 & \text{proba } 1/2 \\ 18 & \text{proba } 1/2 \end{cases}$

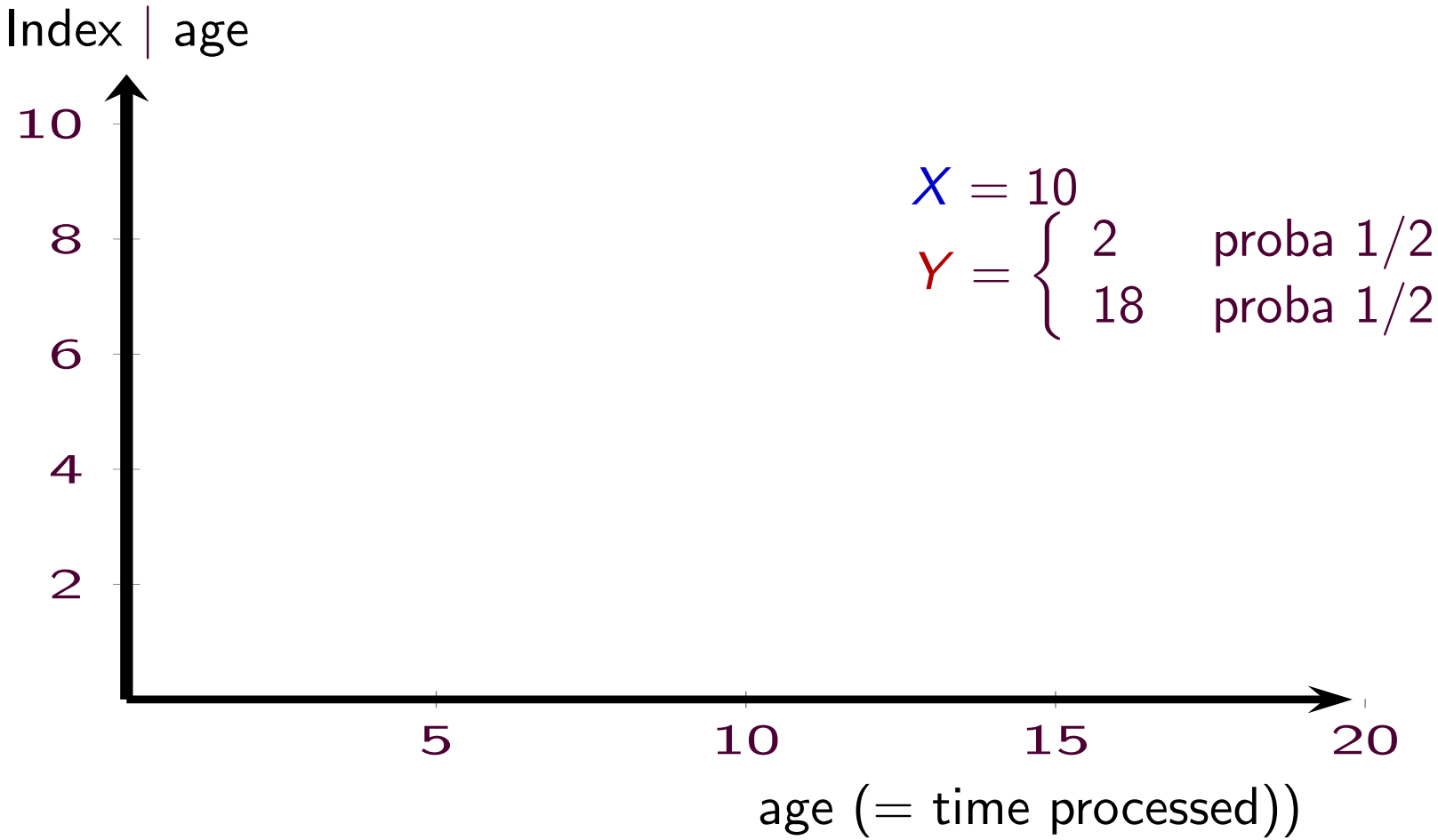


Gittins' index

If a finished job is subsidized x€ and running it costs 1€/sec.
Gittins index of a job = smallest x so that you start running it.

Gittins' index

If a finished job is subsidized x€ and running it costs 1€/sec.
Gittins index of a job = smallest x so that you start running it.



Optimality of Gittins index

Theorem. (Gittins, Glazebrook, Weber, 90s). Scheduling smallest index first is optimal.

The result is powerful but not very robust result: Gittins is not optimal as soon as:

- Multi-server, Impatient jobs,. . .

Optimality of Gittins index

Theorem. (Gittins, Glazebrook, Weber, 90s). Scheduling smallest index first is optimal.

The result is powerful but not very robust result: Gittins is not optimal as soon as:

- Multi-server, Impatient jobs,. . .

(Recently) closed questions:

- Can we analyze the performance of this Gittins index? (SOAP: [Scully et al 2018]).
- Multi-server? (Close to optimal: [Grosof et al 2019]).
- Impatient customers? We need to change the index: [Whittle 88]. (Very close to optimal [G,G,Yan 2021])
- Unkown distribution (No regret learning.[G,G,Khun, 2021])

Impatient Jobs

Suppose that at each time, a job can leave with some probability.

- The problem is in general PSPACE-hard.

Impatient Jobs

Suppose that at each time, a job can leave with some probability.

- The problem is in general PSPACE-hard.

Whittle' index is defined similarly to Gittins' index: Find the smallest subsidy so that you start running it.

- Not always well defined.
- Not optimal in general.
- Works very well in practice.

Impatient Jobs

Suppose that at each time, a job can leave with some probability.

- The problem is in general PSPACE-hard.

Whittle' index is defined similarly to Gittins' index: Find the smallest subsidy so that you start running it.

- Not always well defined.
- Not optimal in general.
- Works very well in practice.

"Therem" [G,G,Yan, 2021] Under quite general condition, when the number of servers n grow, Whittle index becomes asymptotically optimal exponentially fast in n.

Conclusion

Index policies are very efficient to share a single resource among tasks.

- Idea: for a given task, find the largest subsidy so that you are willing to execute this task.
- Schedule cheapest tasks in priority.
- This scales well and performs very well in practice.

Some open questions:

- Does not work "as is" when you have multiple resources : need for other relaxations.
- Does not work for "bin-packing"-like constraints.