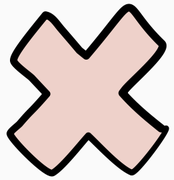
Scheduling and dispatching problems in queues:

 recently closed,

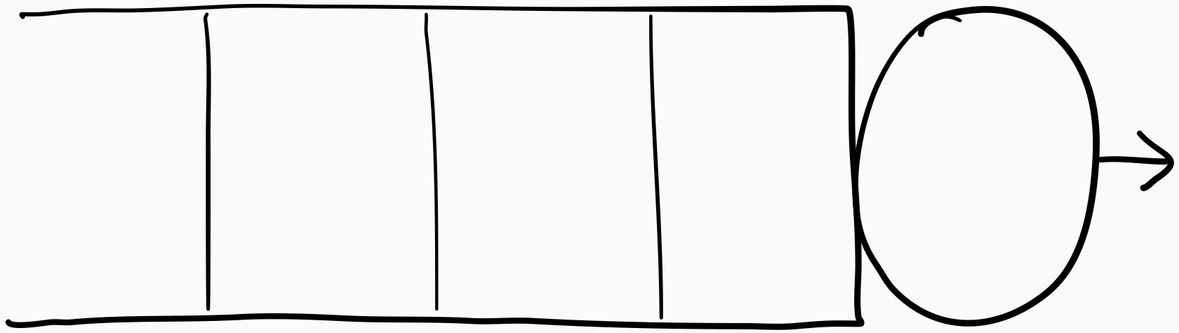
 hopelessly open, and

 promisingly ajar

Ziv Scully
Cornell ORIE

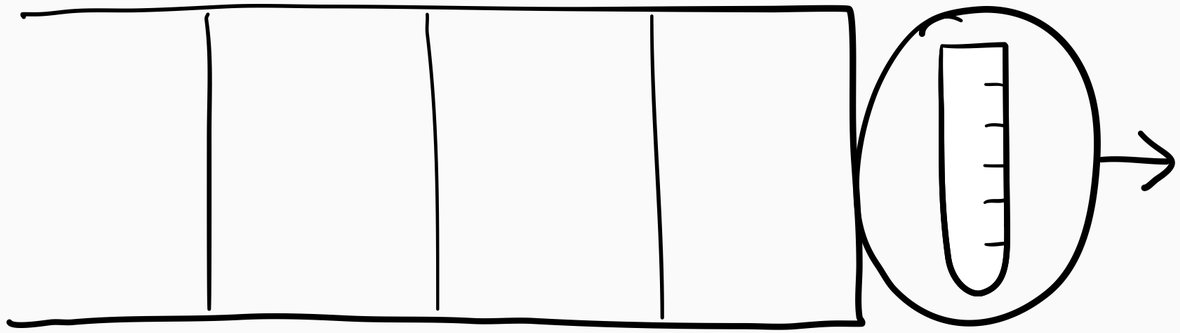
The easiest scheduling problem

The easiest scheduling problem



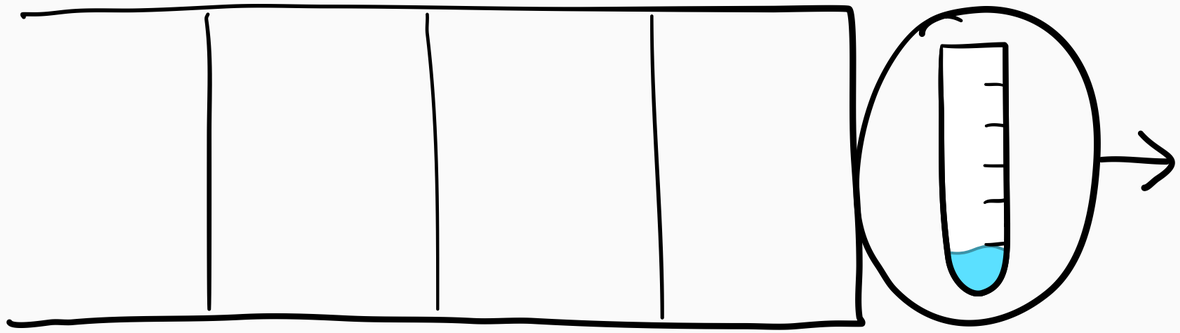
Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



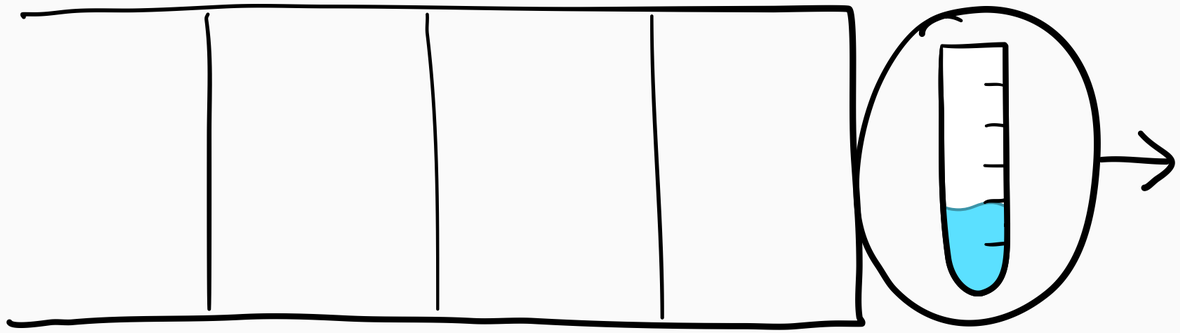
Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



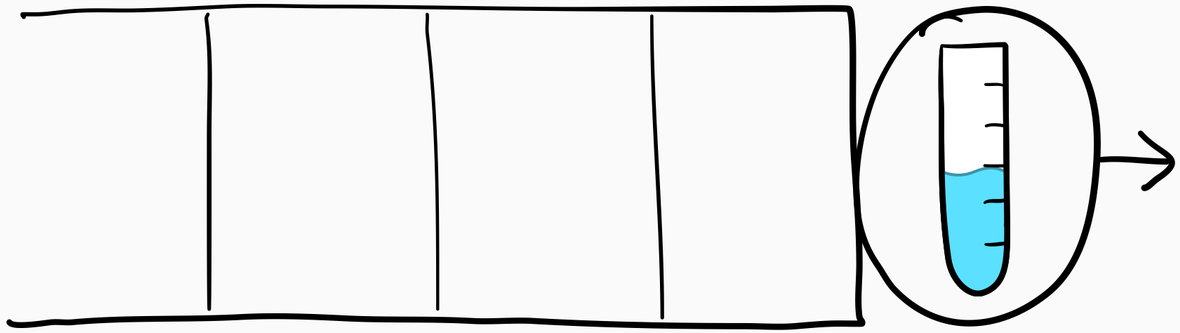
Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



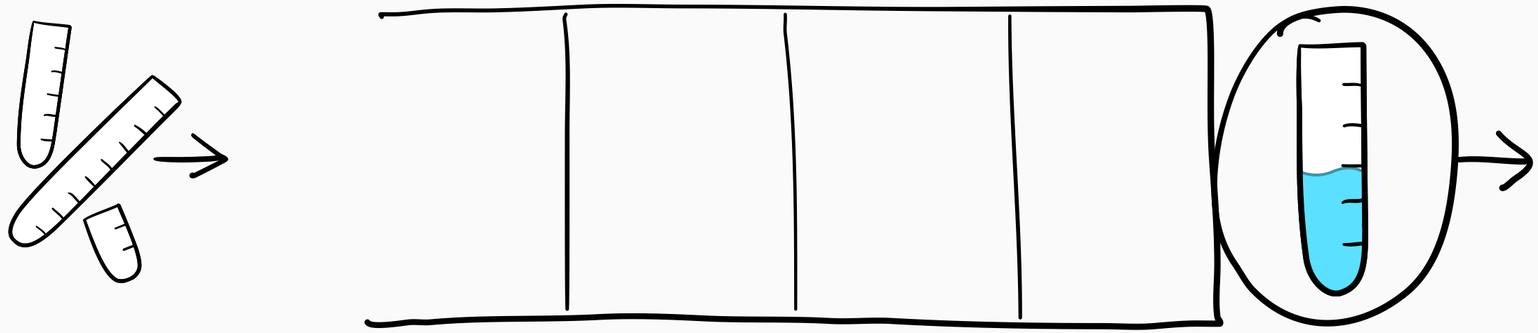
Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



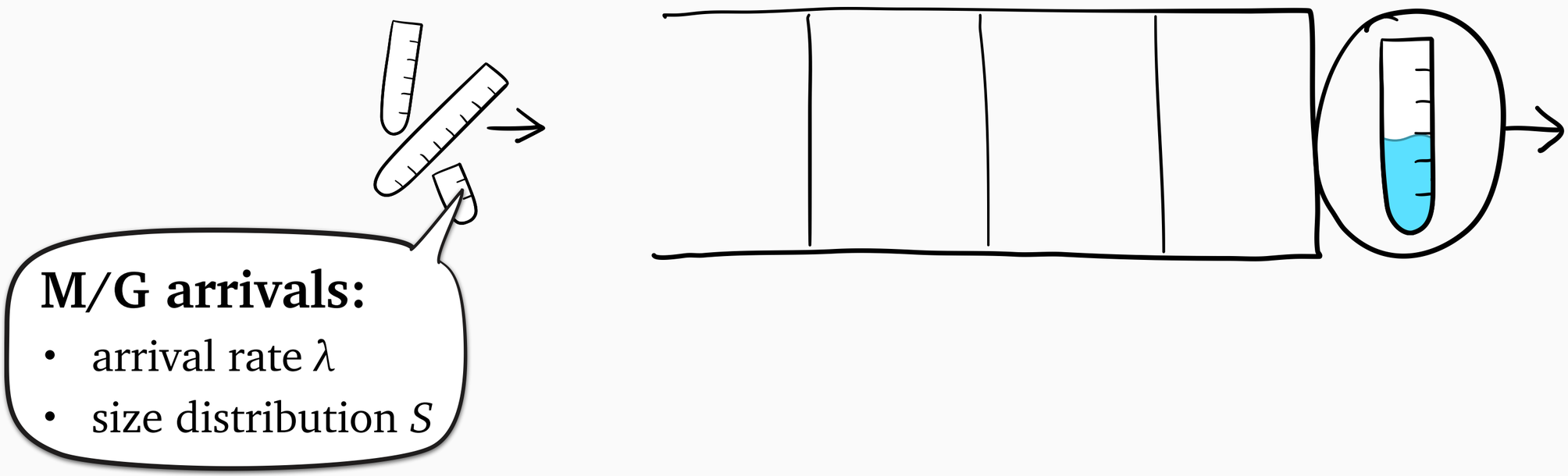
Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

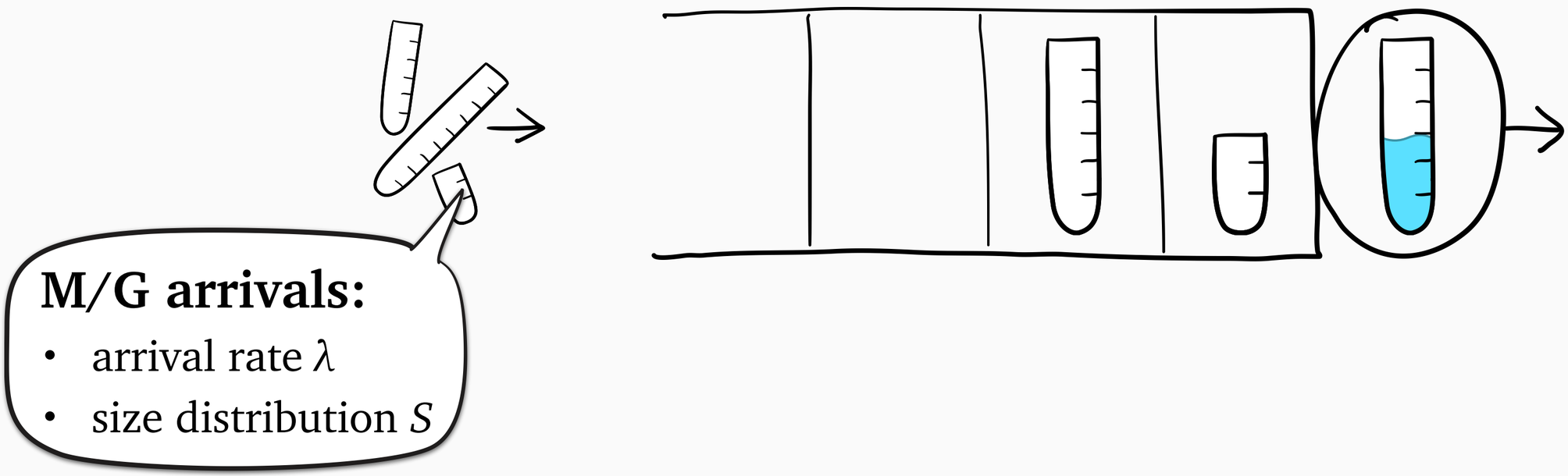


M/G arrivals:

- arrival rate λ
- size distribution S

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

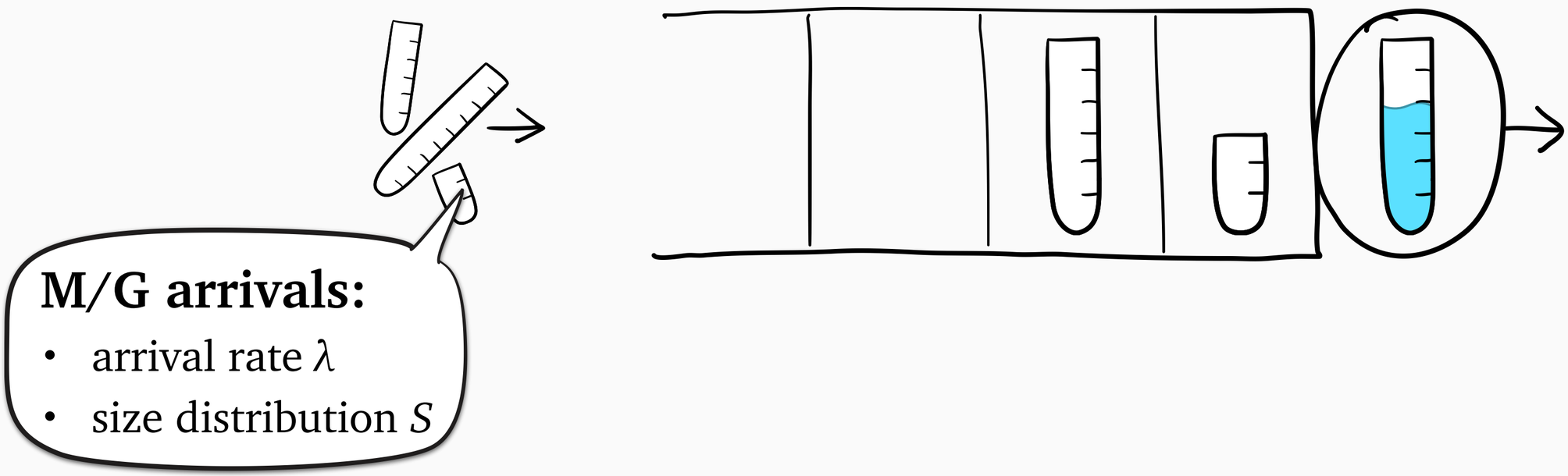


M/G arrivals:

- arrival rate λ
- size distribution S

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

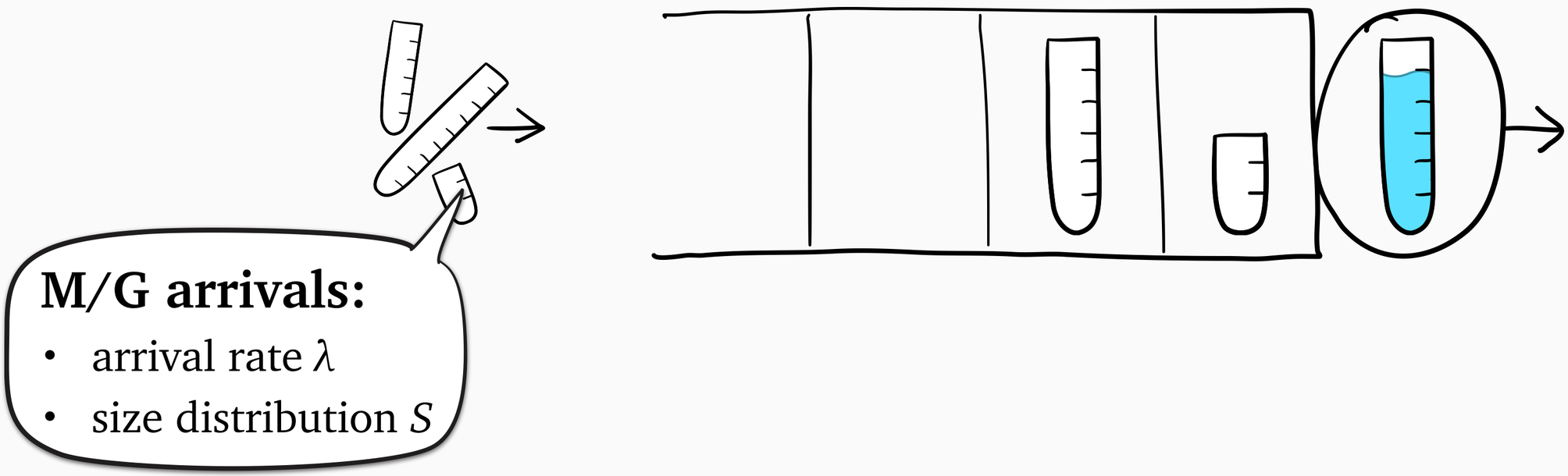


M/G arrivals:

- arrival rate λ
- size distribution S

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

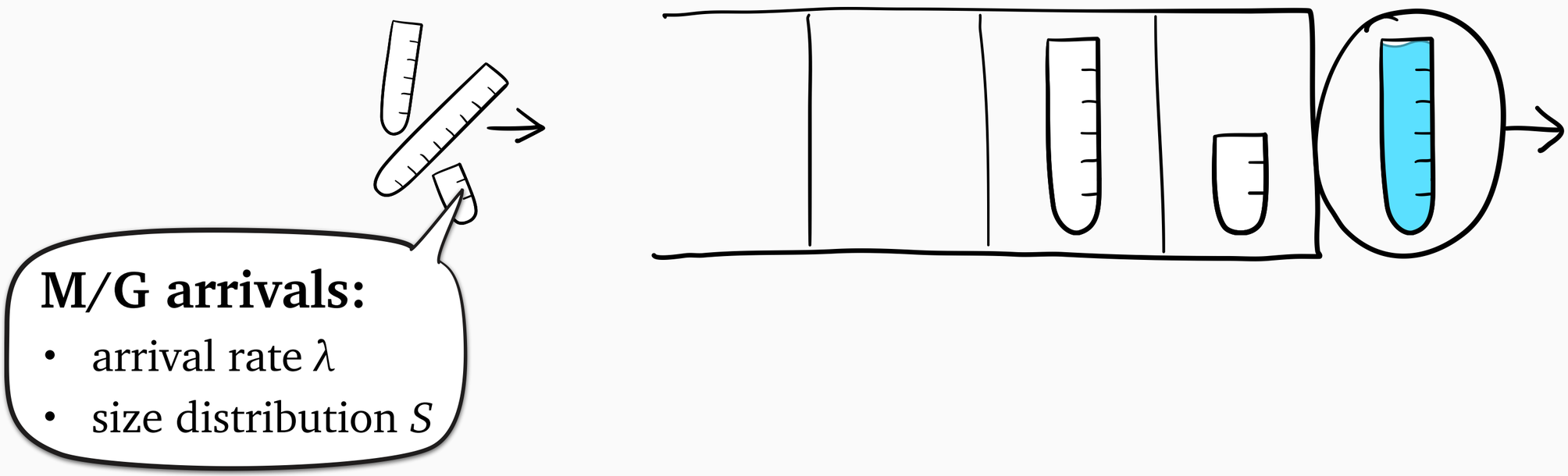


M/G arrivals:

- arrival rate λ
- size distribution S

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

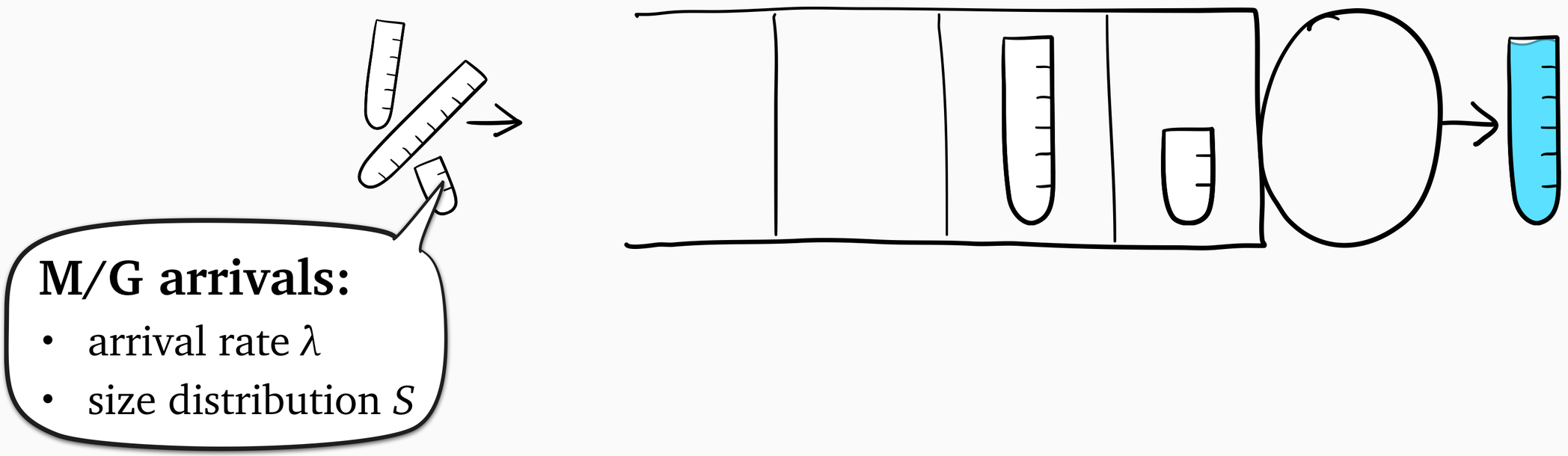


M/G arrivals:

- arrival rate λ
- size distribution S

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

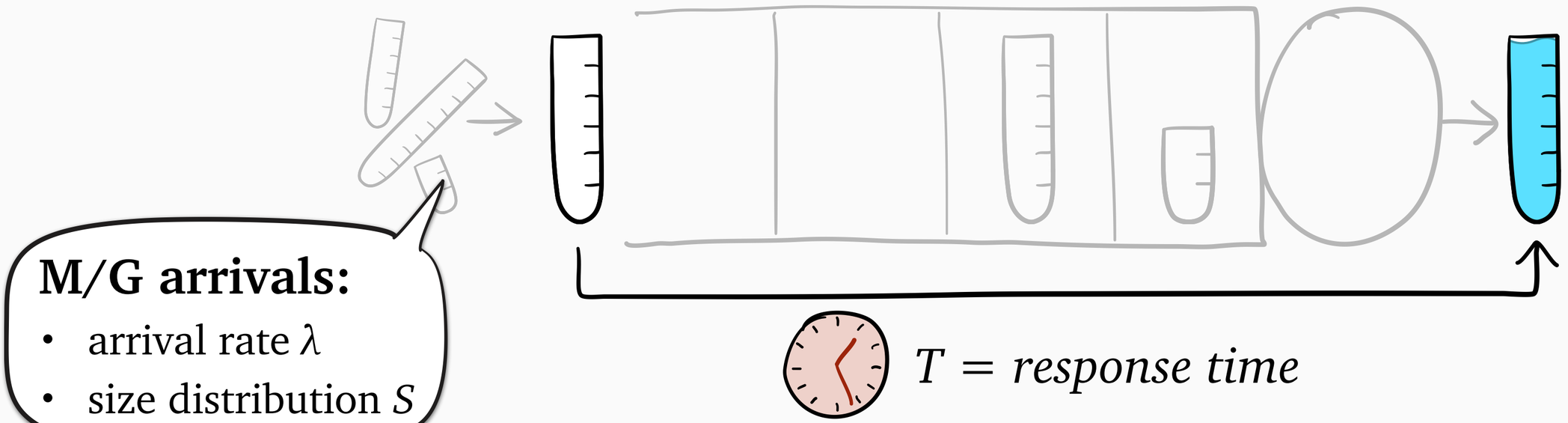


M/G arrivals:

- arrival rate λ
- size distribution S

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



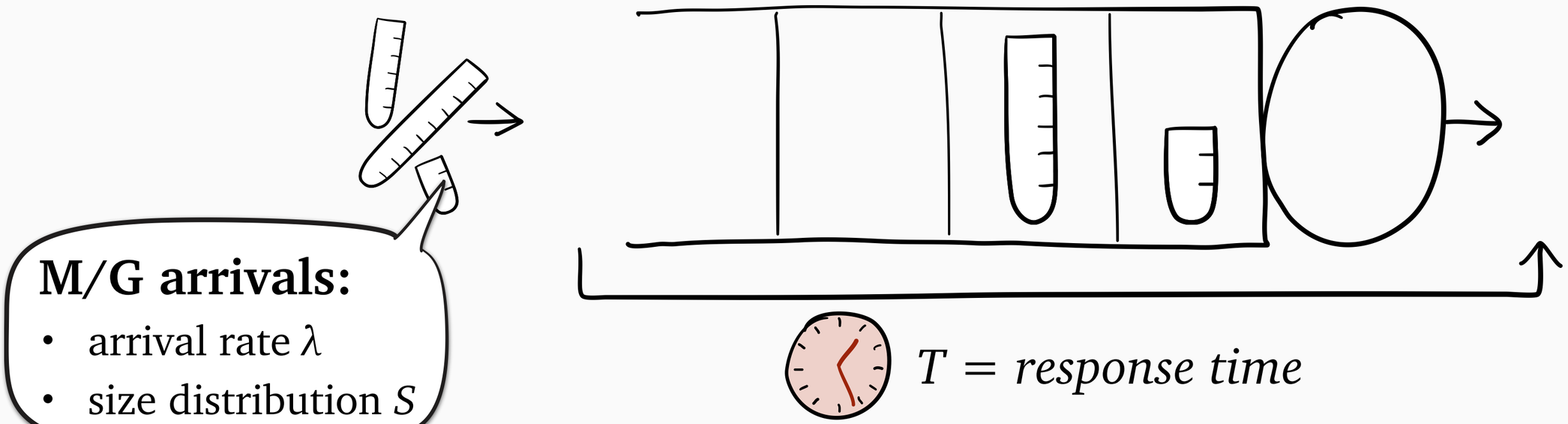
M/G arrivals:

- arrival rate λ
- size distribution S

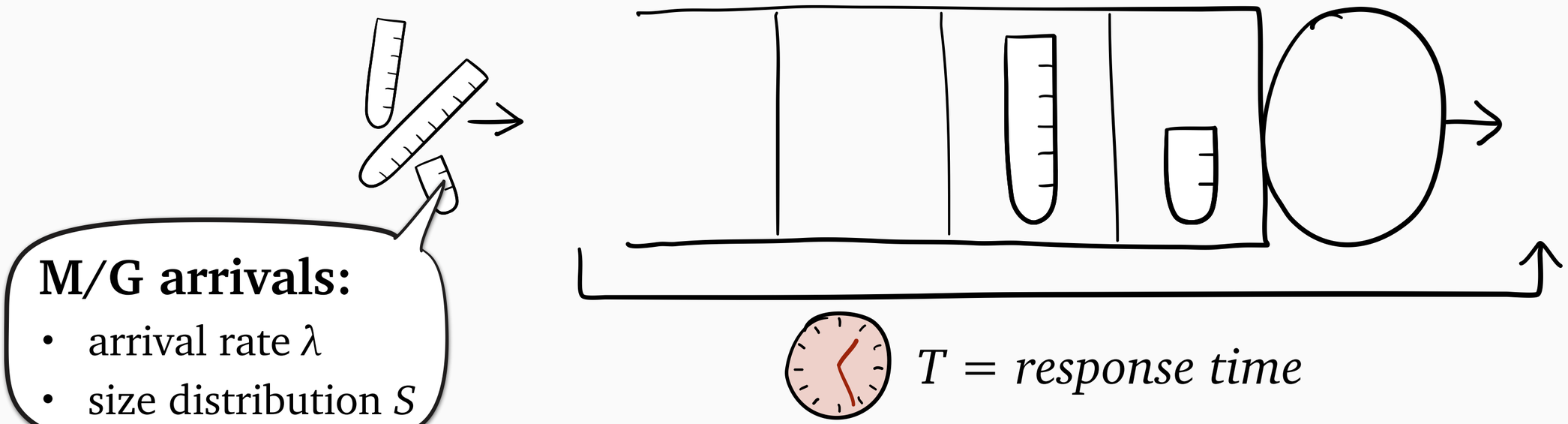
T = *response time*

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



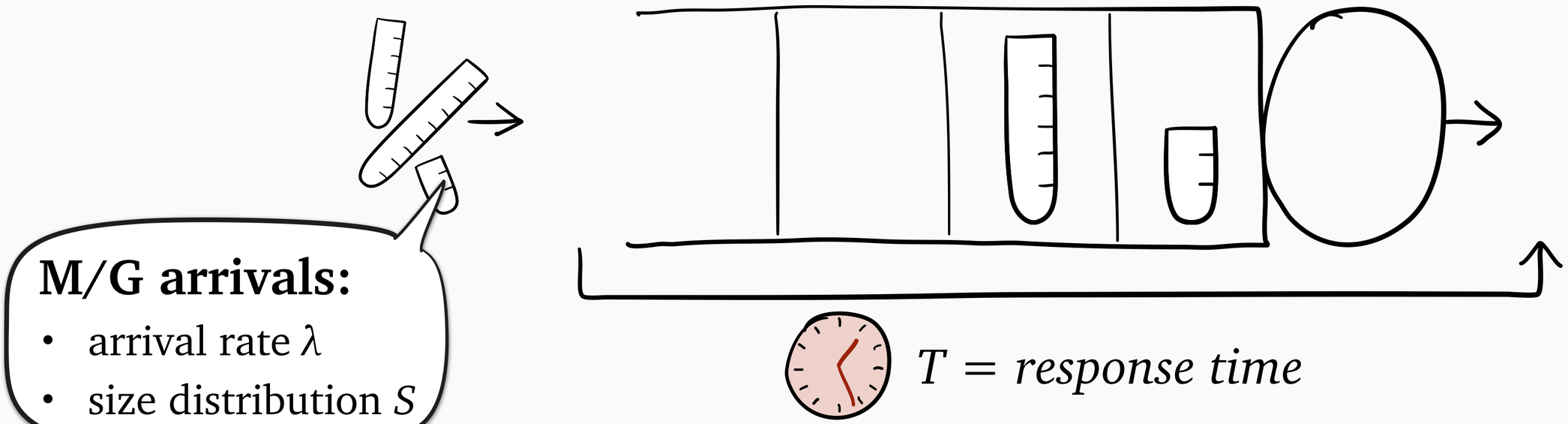
The easiest scheduling problem



Minimize $\mathbf{E}[T]$?

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

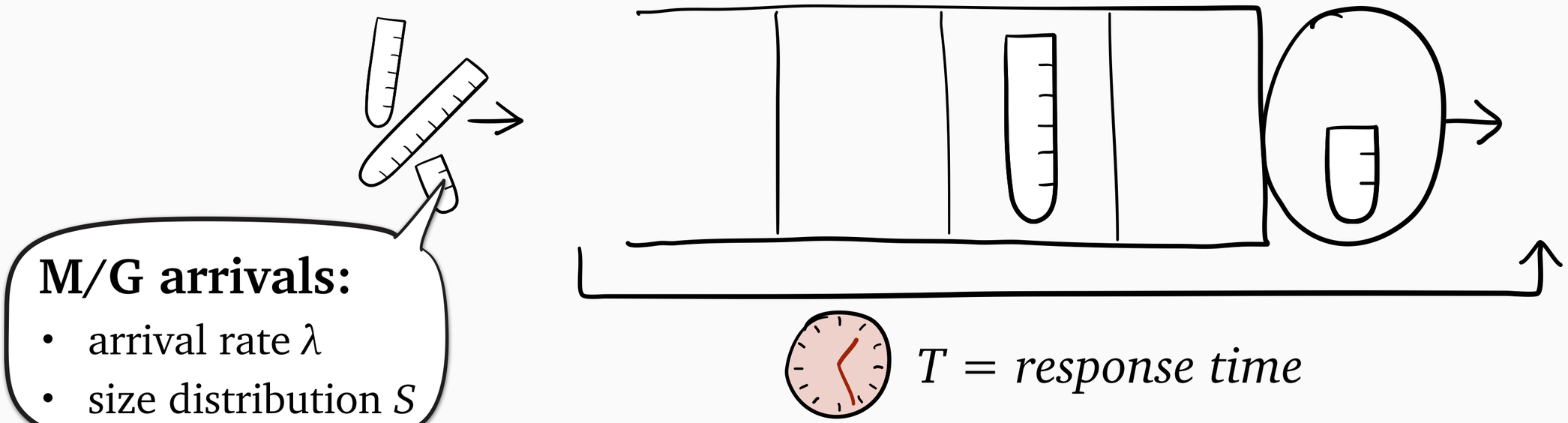


Minimize $\mathbf{E}[T]$?

Serve short jobs
before long jobs

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

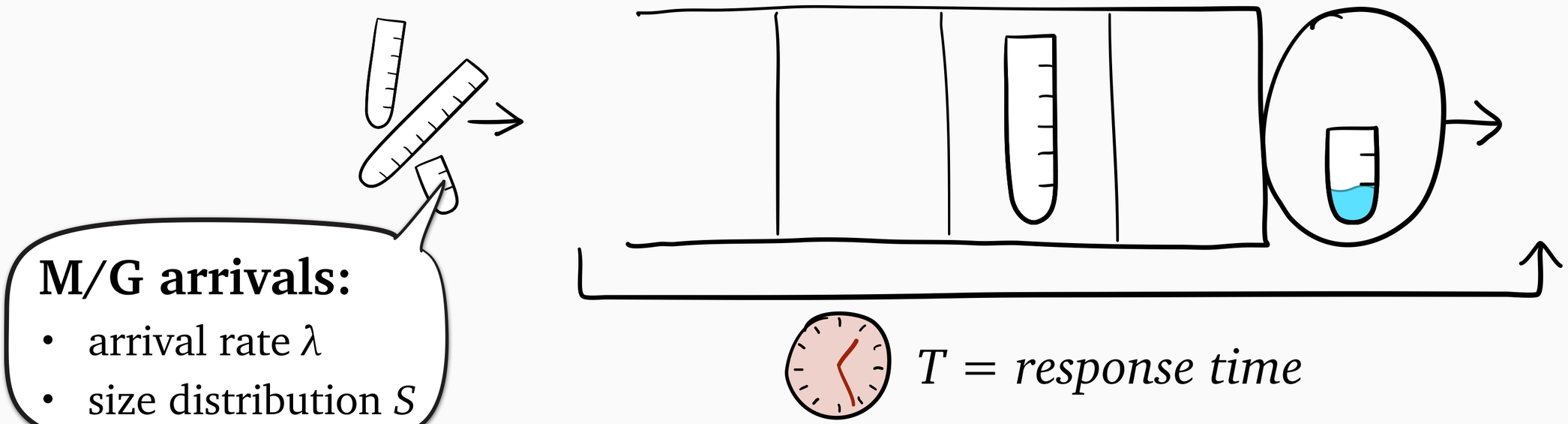


Minimize $\mathbf{E}[T]$?

Serve short jobs
before long jobs

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem

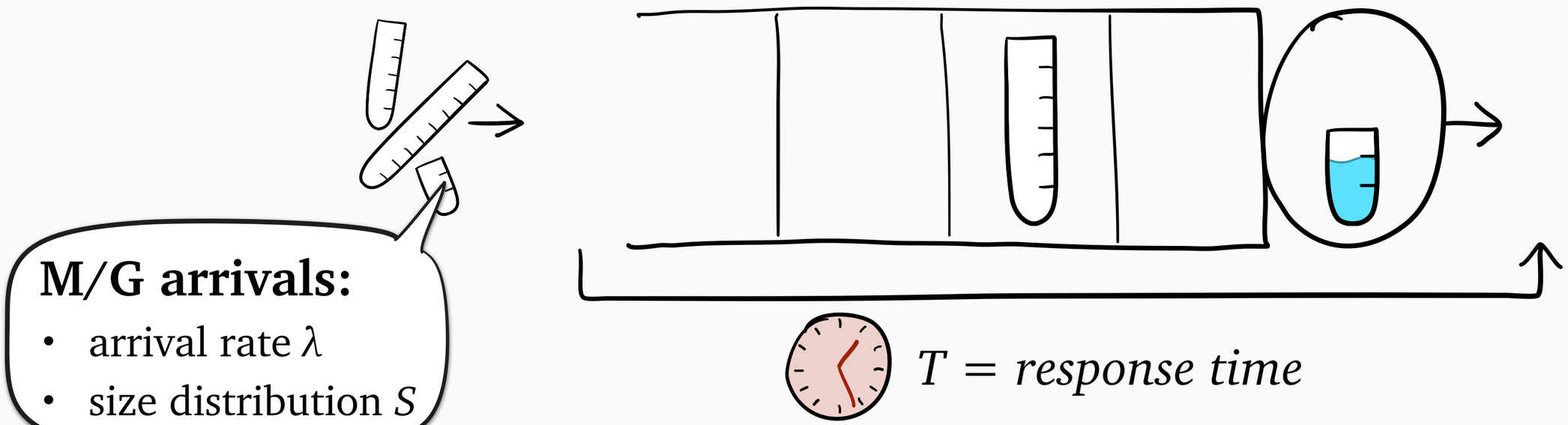


Minimize $\mathbf{E}[T]$?

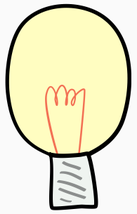
Serve short jobs before long jobs

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



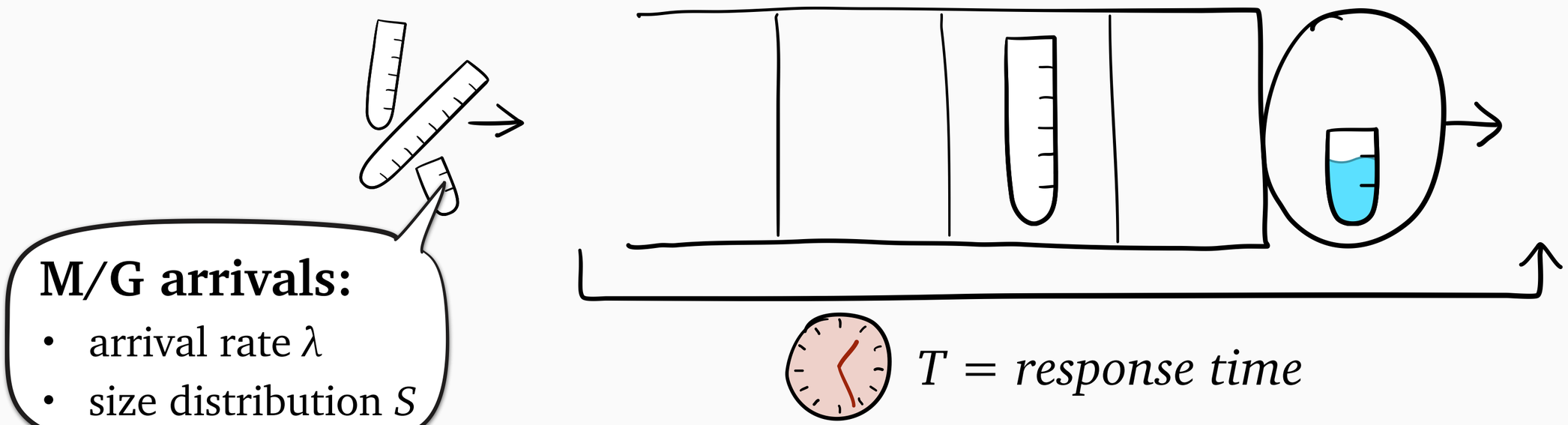
Minimize $\mathbf{E}[T]$?



Serve short jobs
before long jobs

Schrage (1968). doi:10.1287/opre.16.3.687.

The easiest scheduling problem



M/G arrivals:

- arrival rate λ
- size distribution S

T = *response time*

Minimize $\mathbf{E}[T]$?

Serve short jobs before long jobs

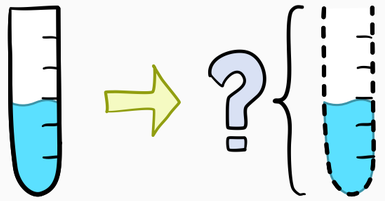
shortest remaining processing time

SRPT: minimizes $\mathbf{E}[T]$

Schrage (1968). doi:10.1287/opre.16.3.687.

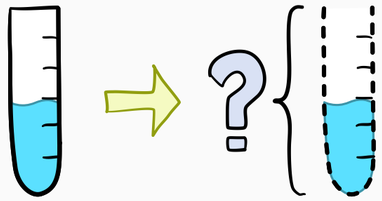
Hard scheduling questions

Hard scheduling questions

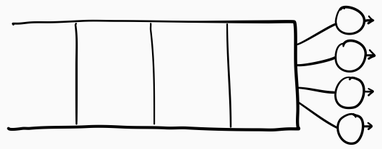


What if we have
unknown job sizes?

Hard scheduling questions

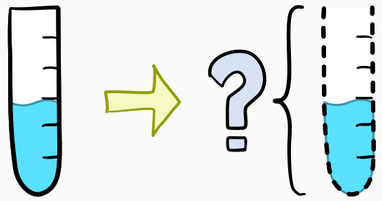


What if we have
unknown job sizes?

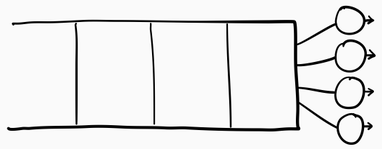


What if there are
multiple servers?

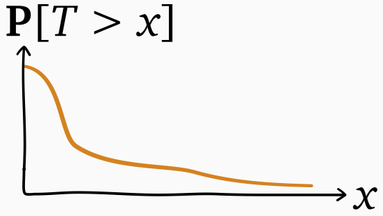
Hard scheduling questions



What if we have
unknown job sizes?

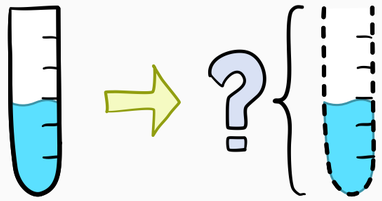


What if there are
multiple servers?

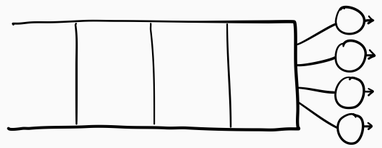


What if we want to optimize
tails instead of means?

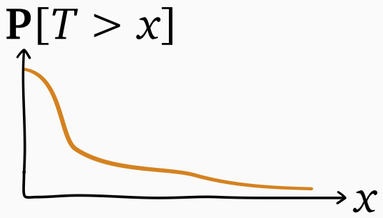
Hard scheduling questions



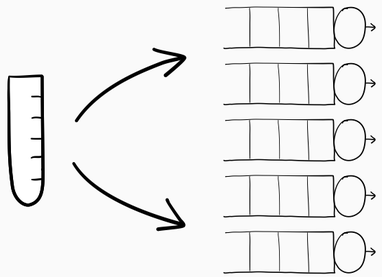
What if we have
unknown job sizes?



What if there are
multiple servers?

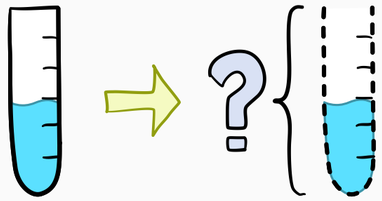


What if we want to optimize
tails instead of means?

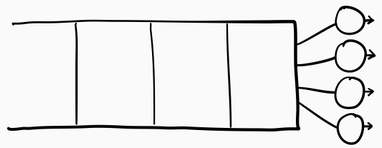


What if we can only use
dispatching without fancy scheduling?

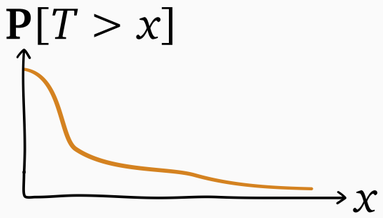
Hard scheduling questions



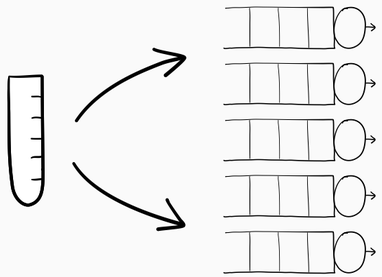
What if we have
unknown job sizes?



What if there are
multiple servers?



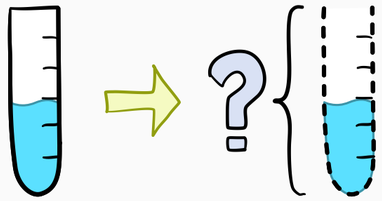
What if we want to optimize
tails instead of means?



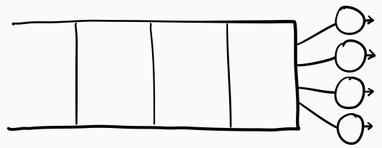
What if we can only use
dispatching without fancy scheduling?

What are the underlying
theoretical tools?

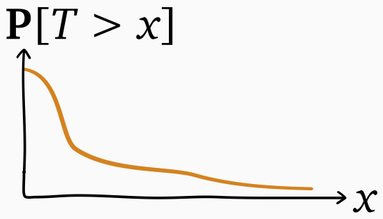
Hard scheduling questions



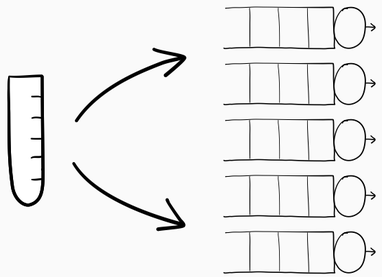
What if we have
unknown job sizes?



What if there are
multiple servers?

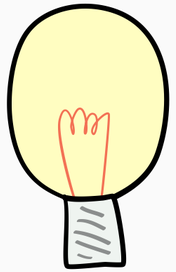


What if we want to optimize
tails instead of means?



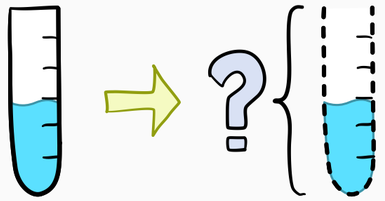
What if we can only use
dispatching without fancy scheduling?

What are the underlying
theoretical tools?

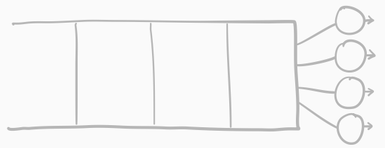


What are the resulting
practical lessons?

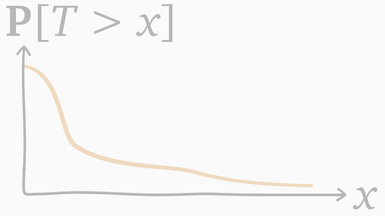
Hard scheduling questions



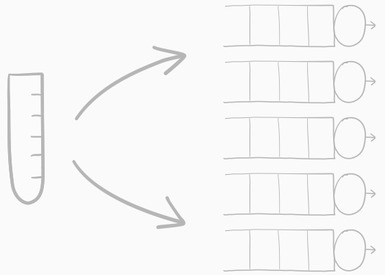
What if we have
unknown job sizes?



What if there are
multiple servers?

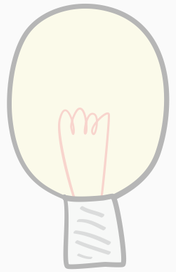


What if we want to optimize
tails instead of means?



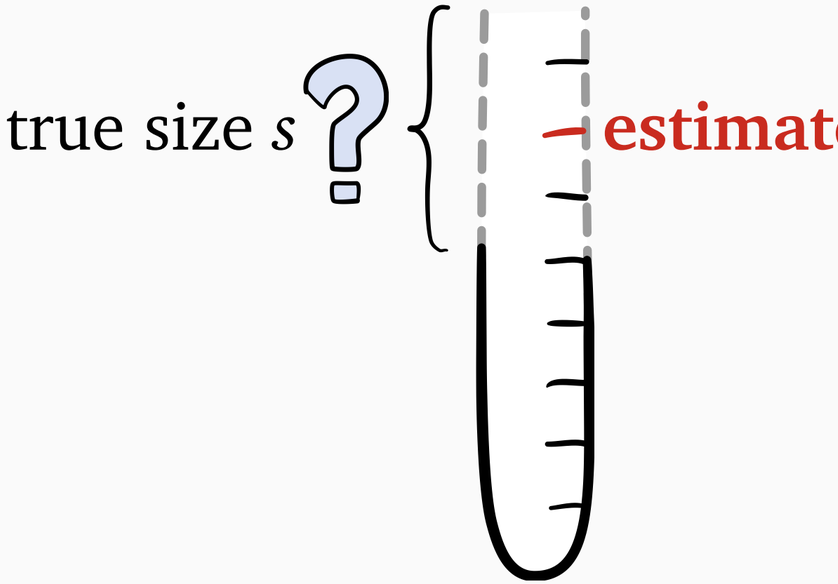
What if we can only use
dispatching without fancy scheduling?

What are the underlying
theoretical tools?

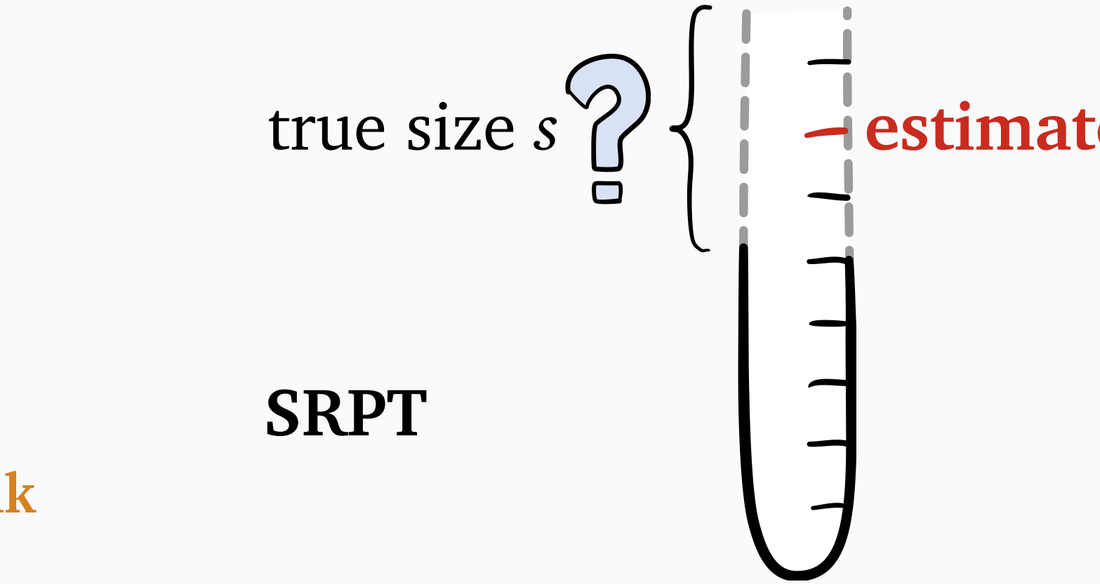


What are the resulting
practical lessons?

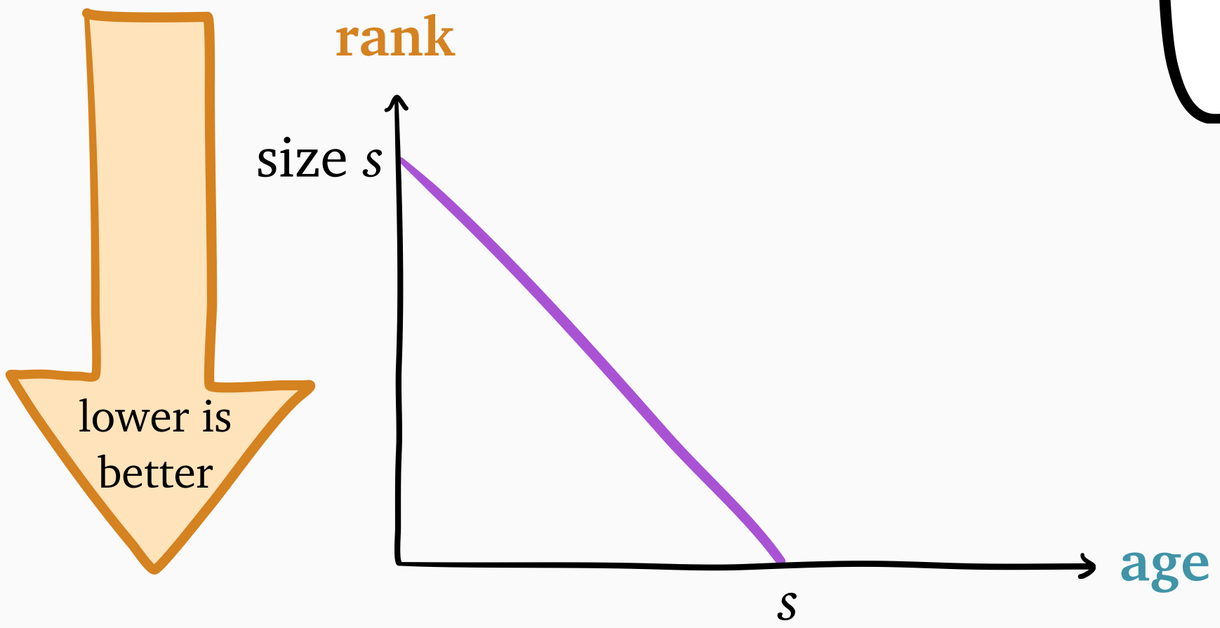
Scheduling with **noisy size estimates**



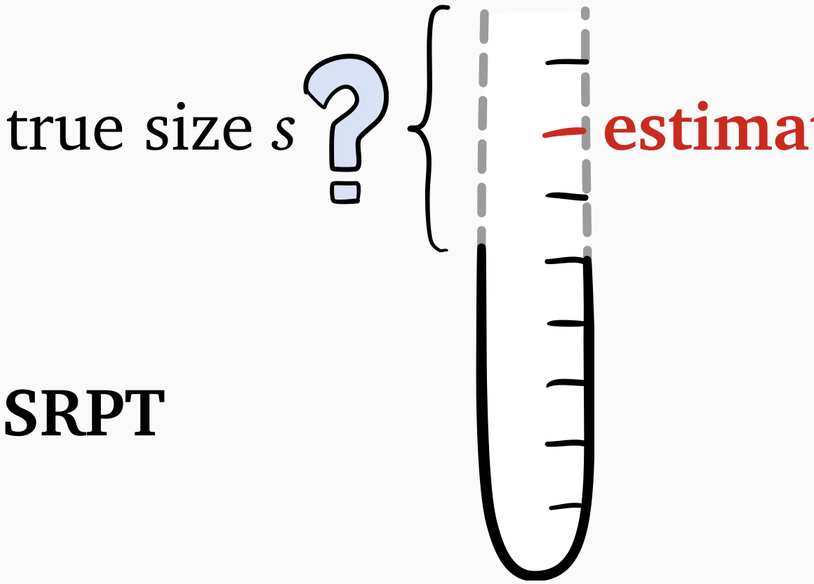
Scheduling with **noisy size estimates**



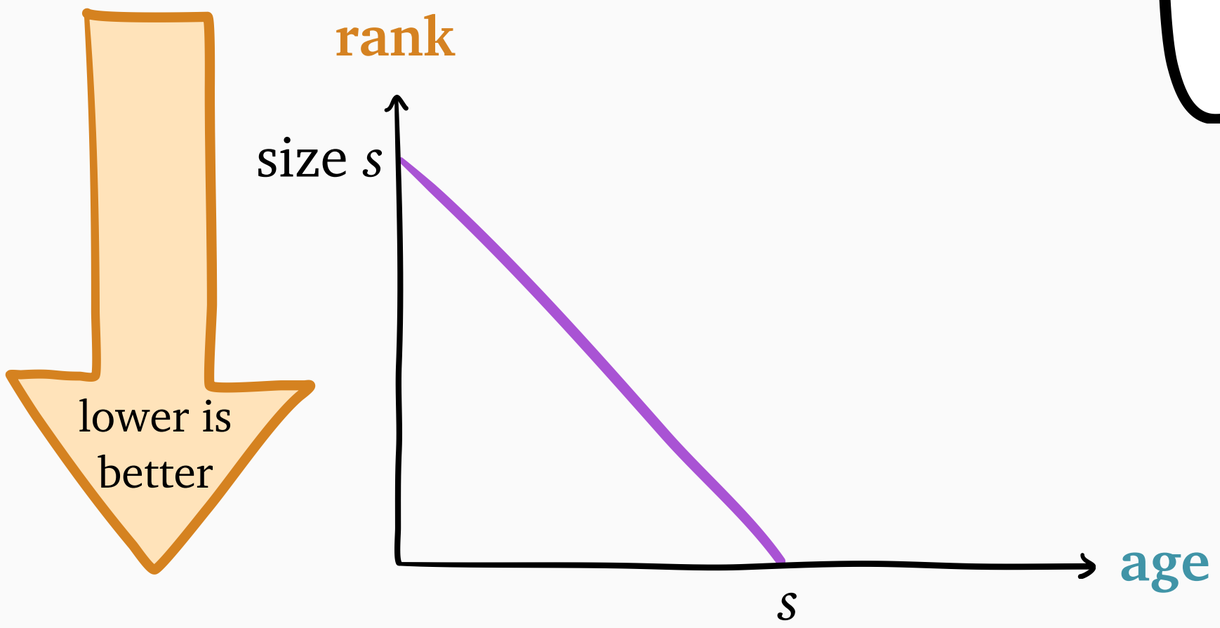
SRPT



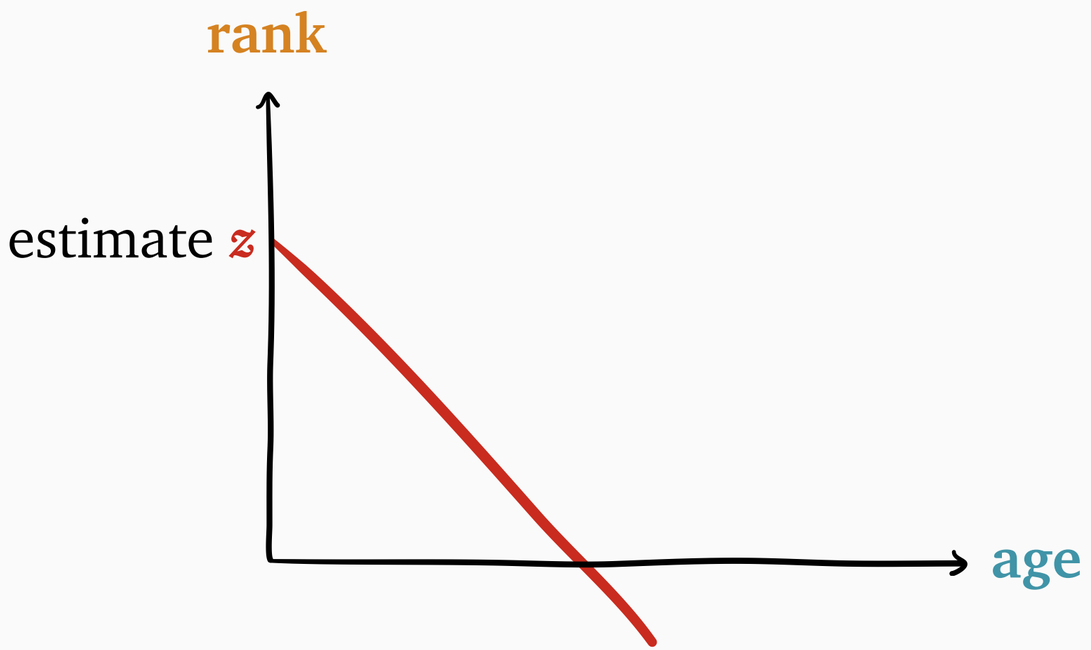
Scheduling with **noisy size estimates**



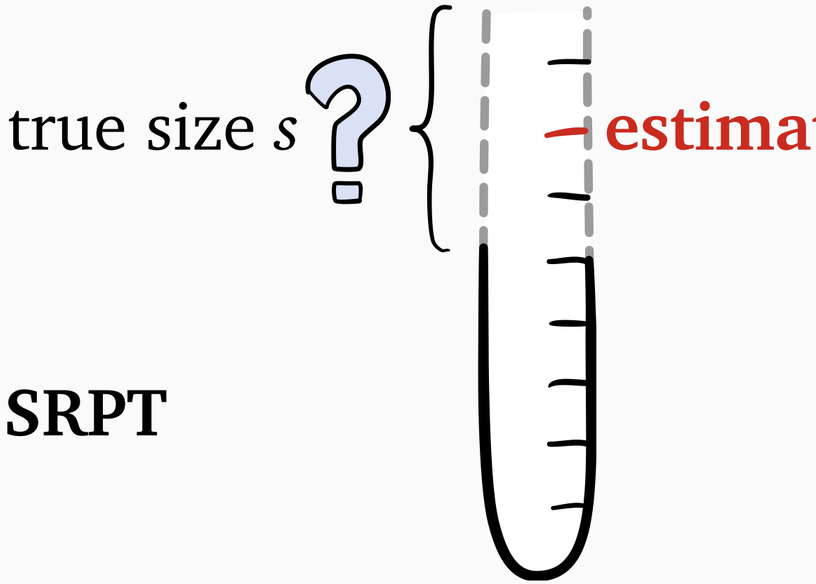
SRPT



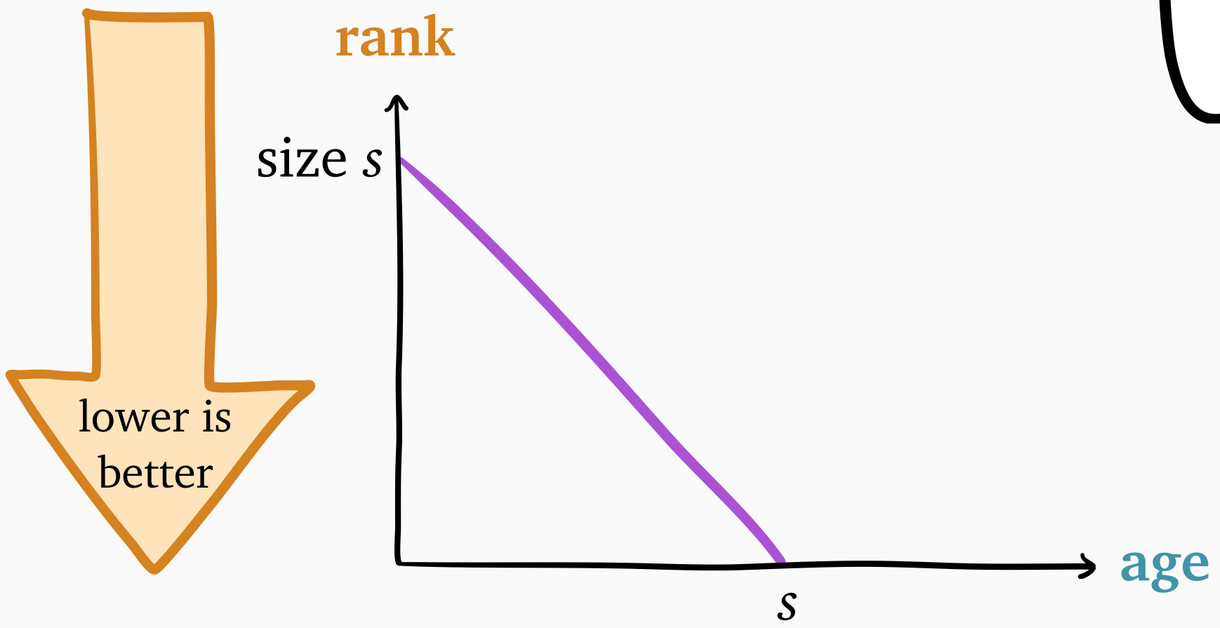
Noisy SRPT



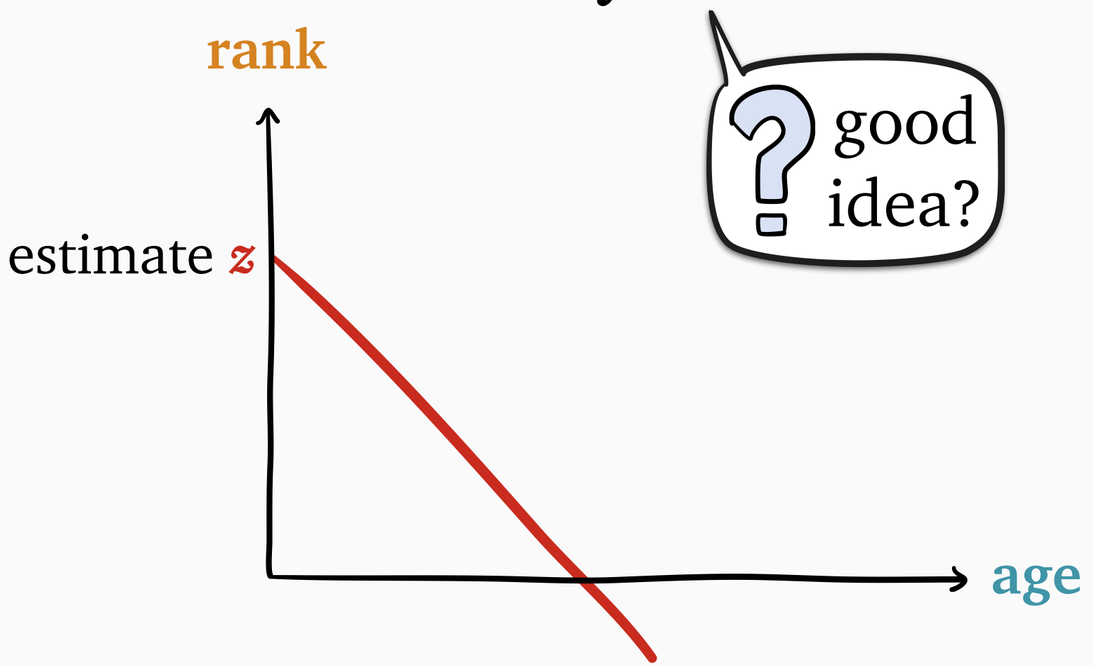
Scheduling with noisy size estimates

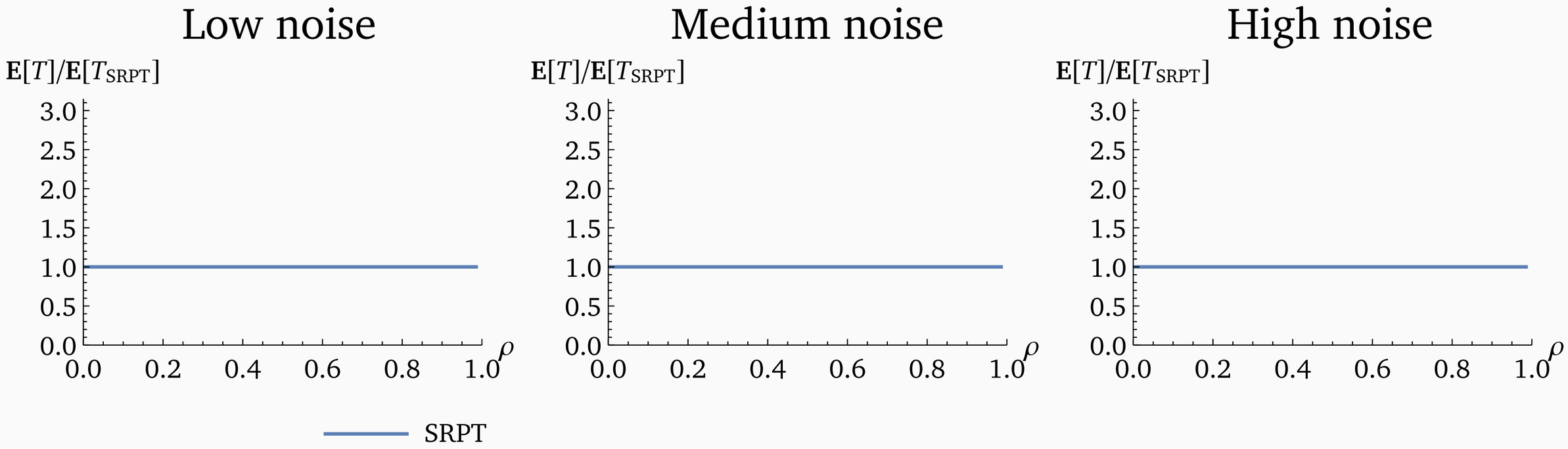


SRPT

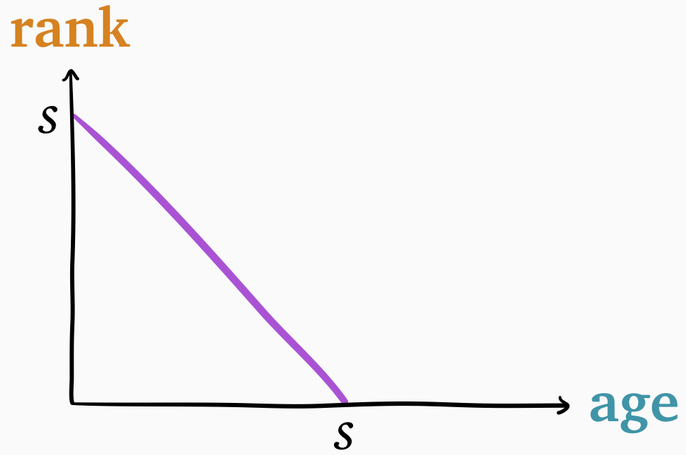


Noisy SRPT



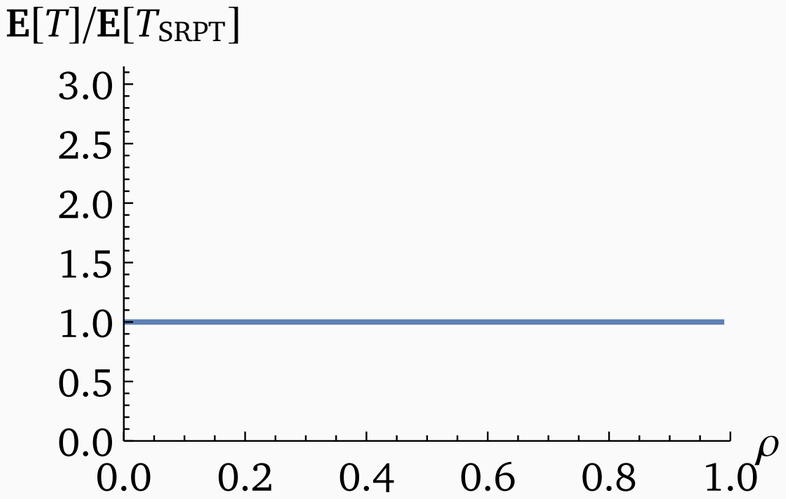


SRPT

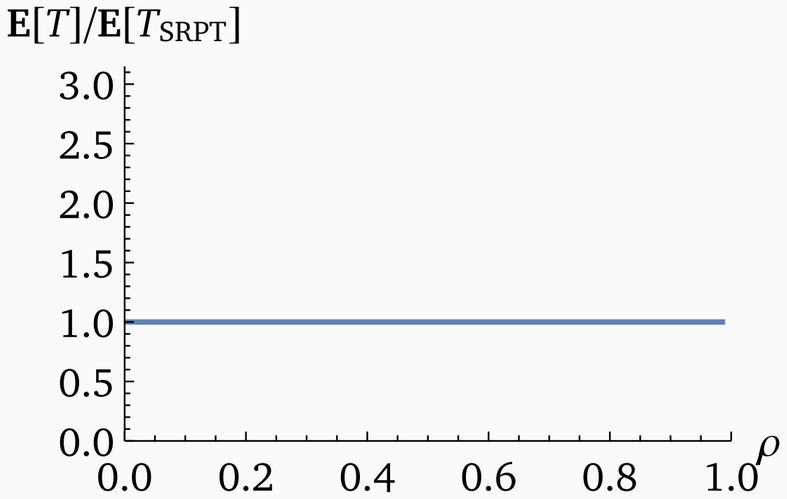


Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.

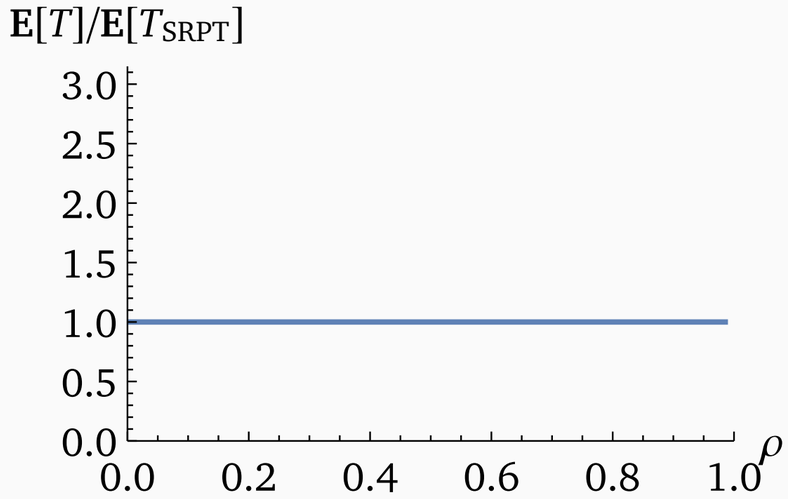
Low noise



Medium noise

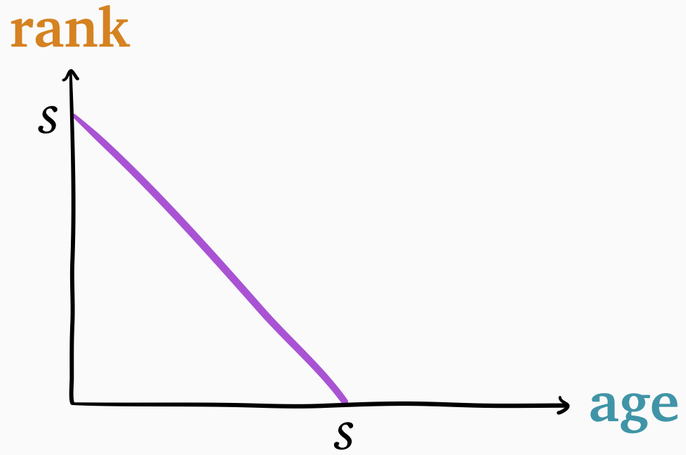


High noise

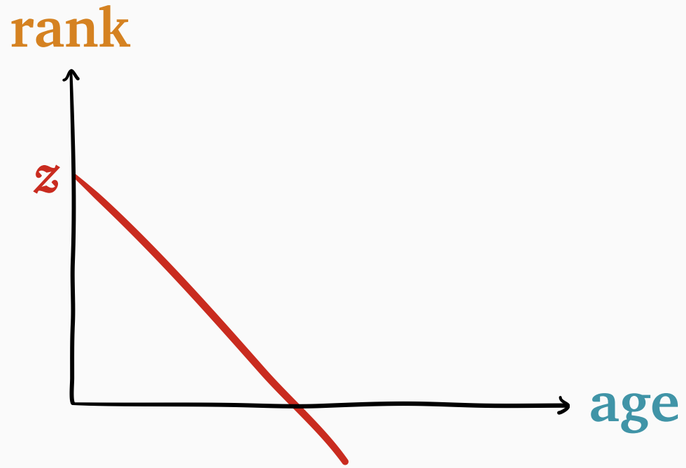


SRPT

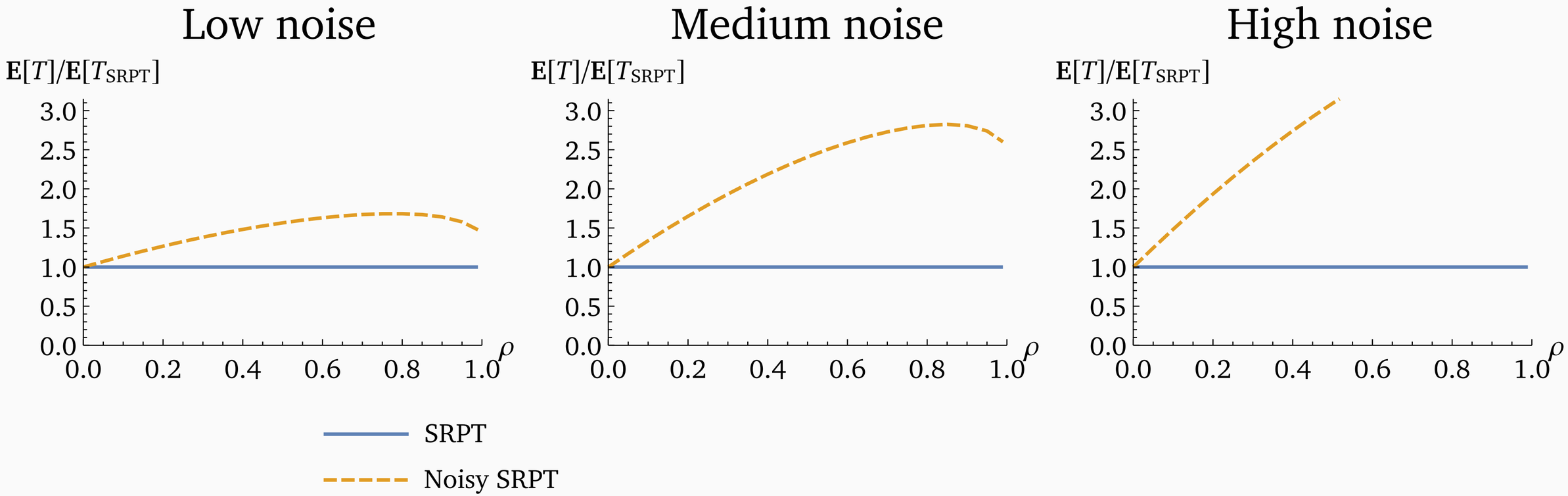
SRPT

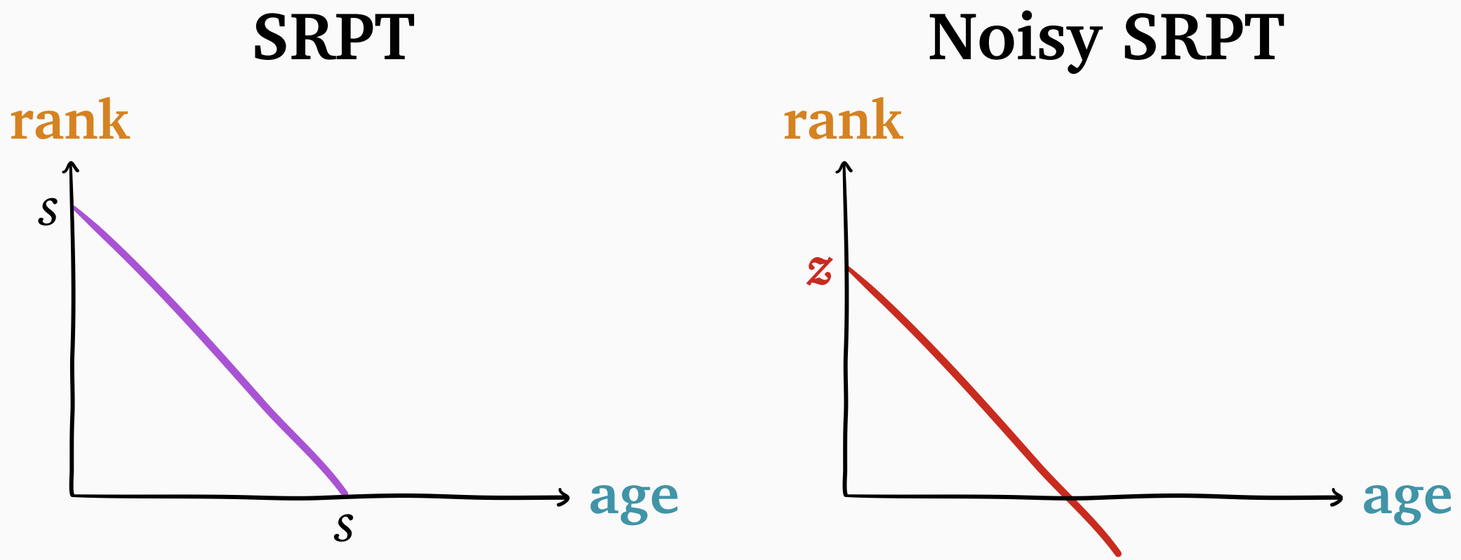


Noisy SRPT



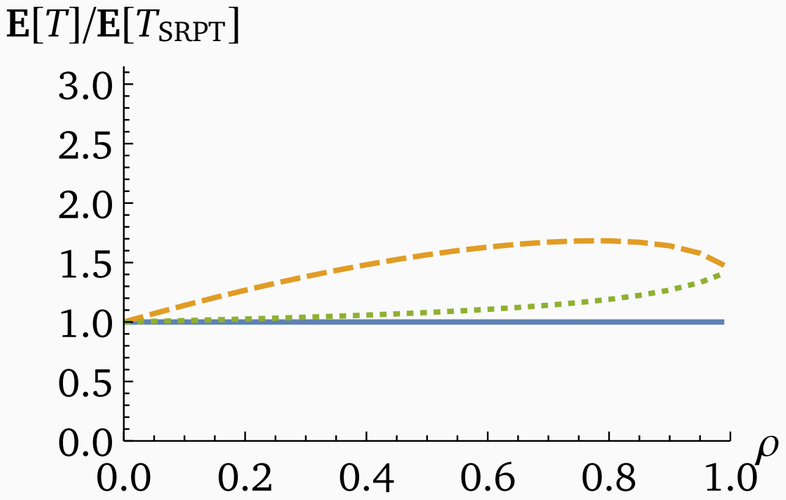
Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.



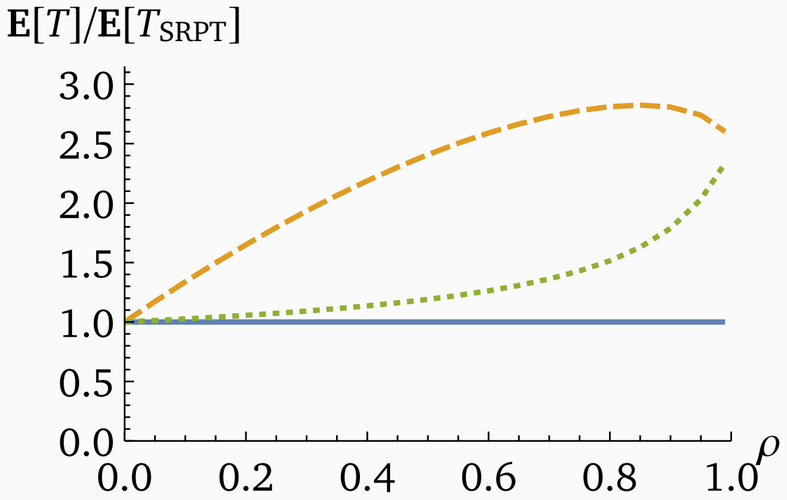


Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.

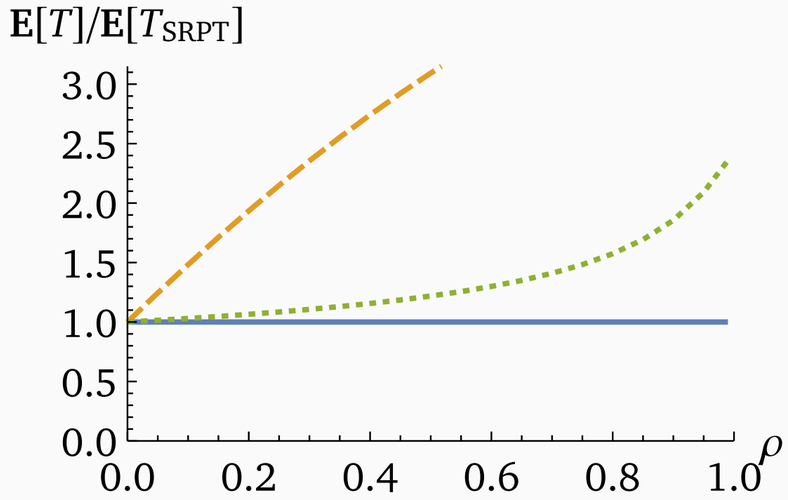
Low noise



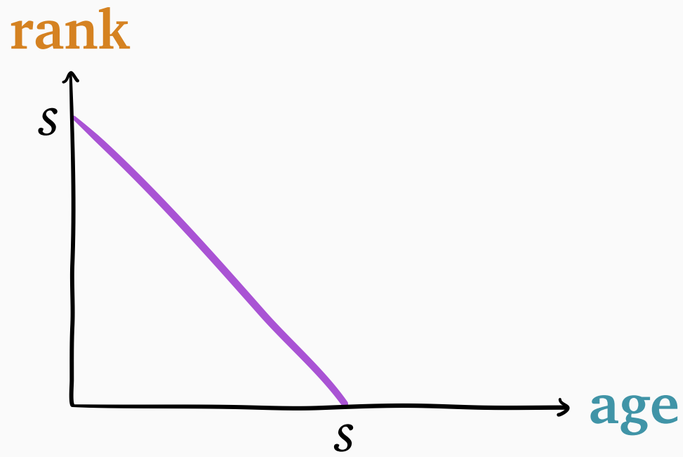
Medium noise



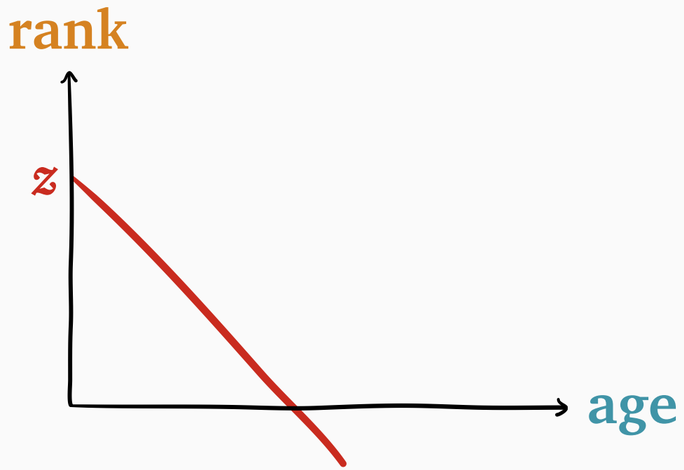
High noise



SRPT

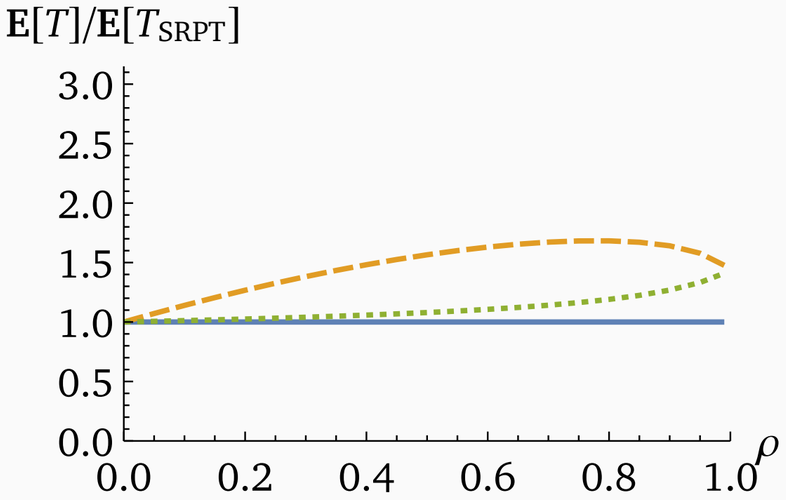


Noisy SRPT

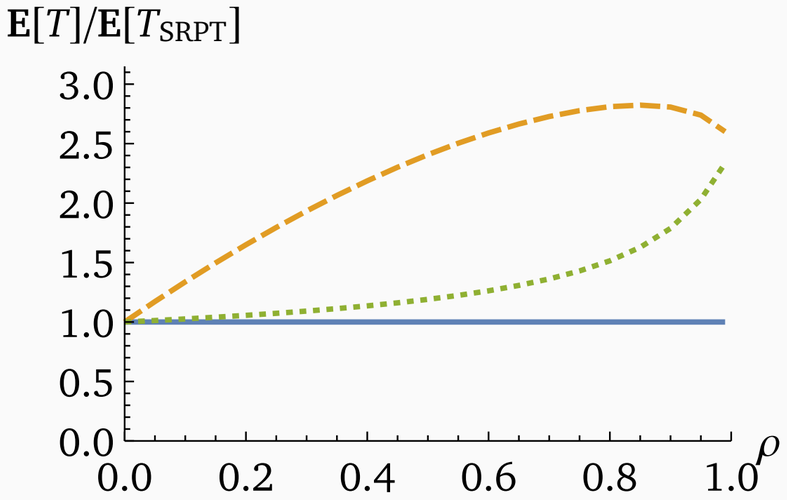


Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.

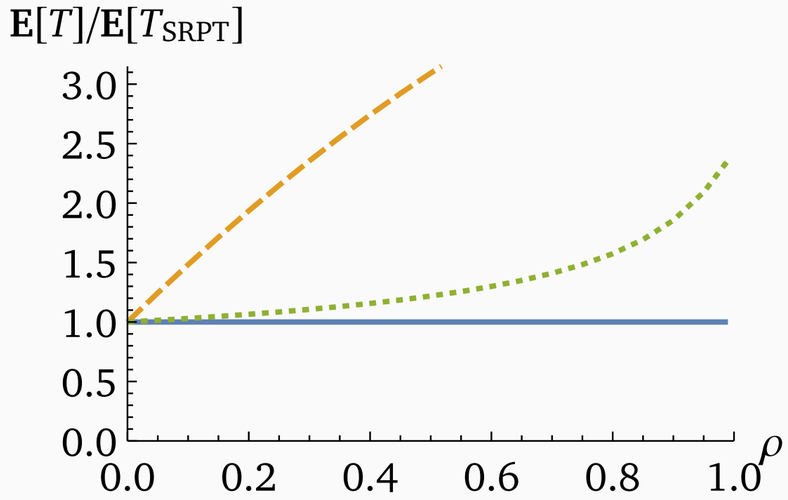
Low noise



Medium noise



High noise

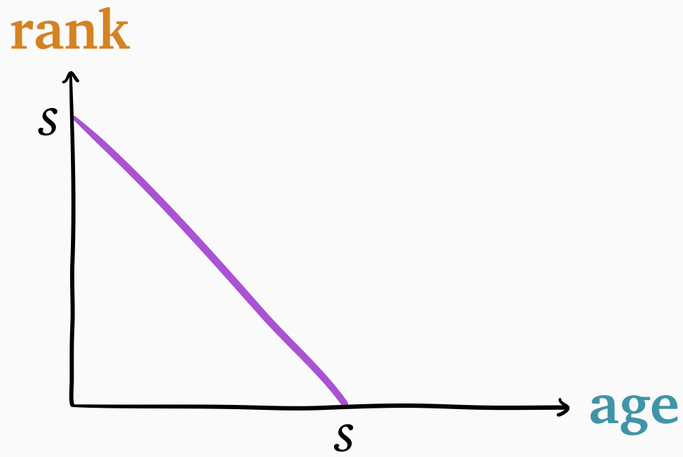


SRPT

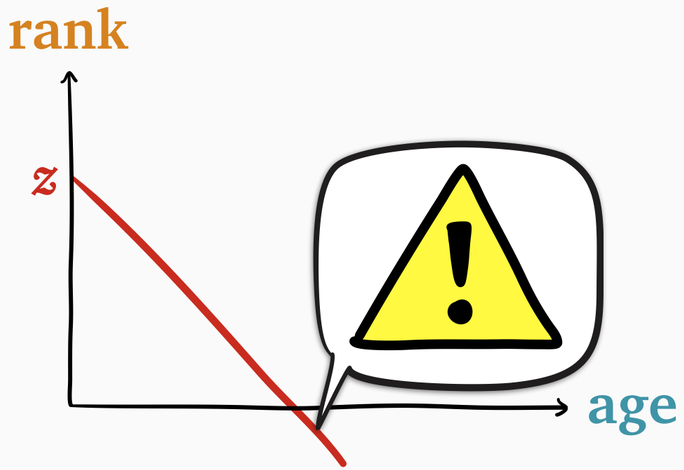
Opt. w/ estimates

Noisy SRPT

SRPT

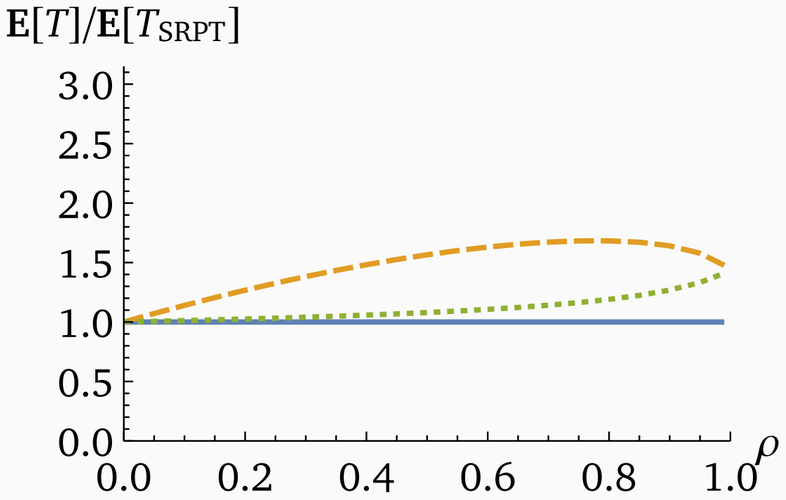


Noisy SRPT

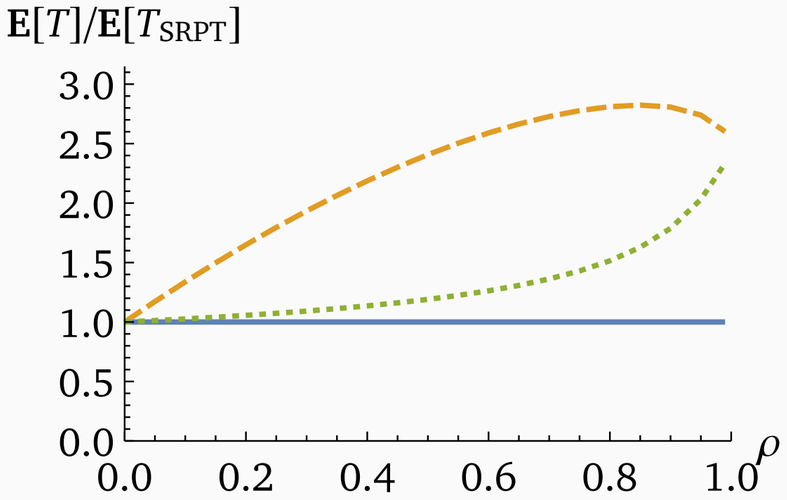


Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.

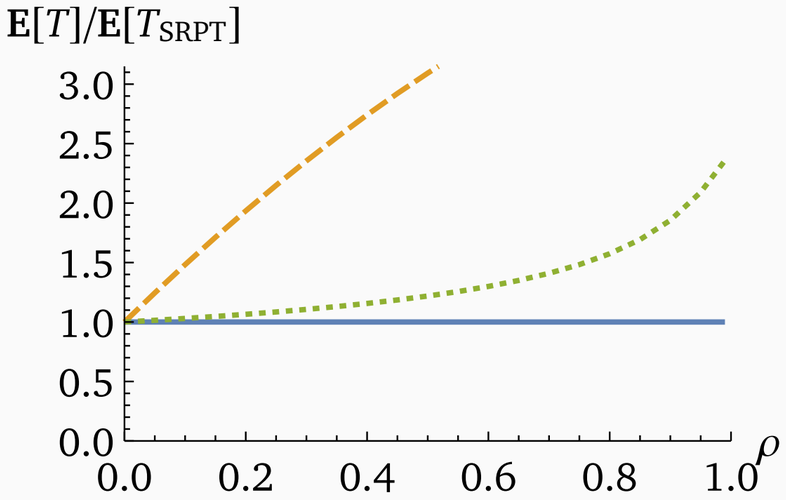
Low noise



Medium noise



High noise

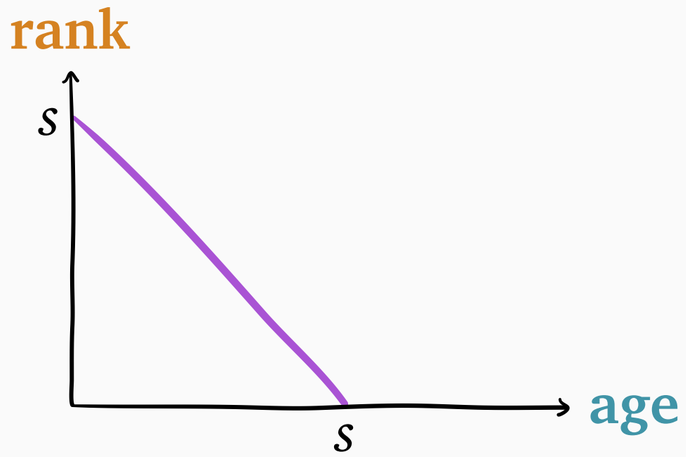


SRPT

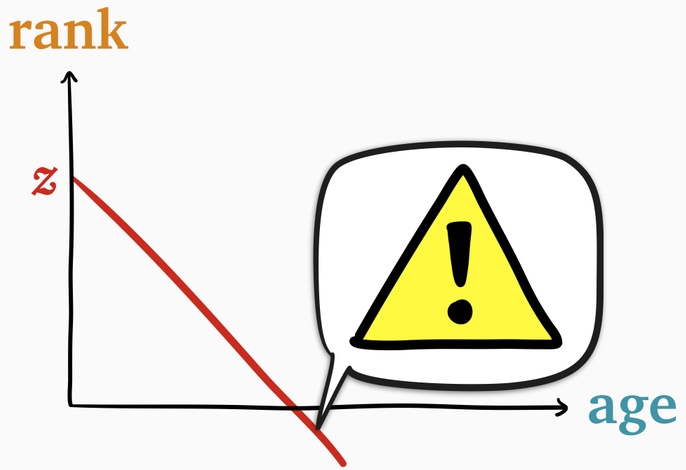
Opt. w/ estimates

Noisy SRPT

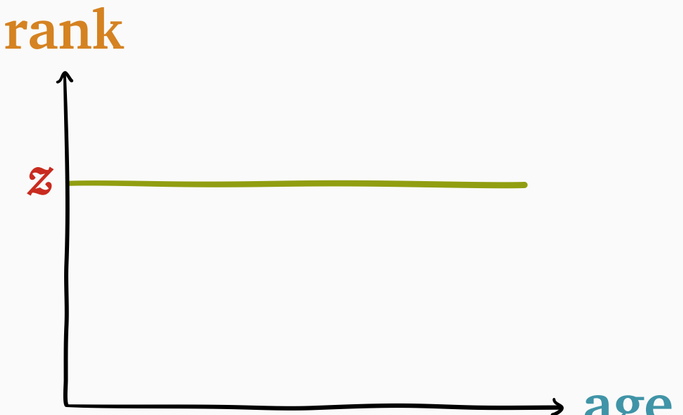
SRPT



Noisy SRPT



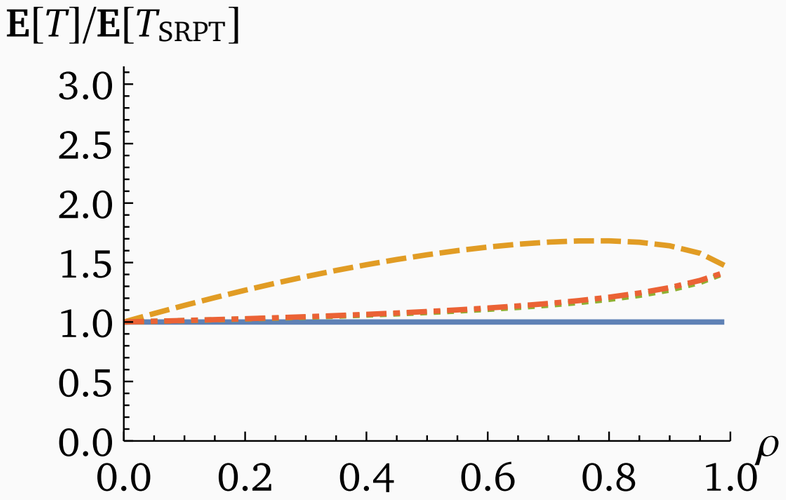
Noisy PSJF



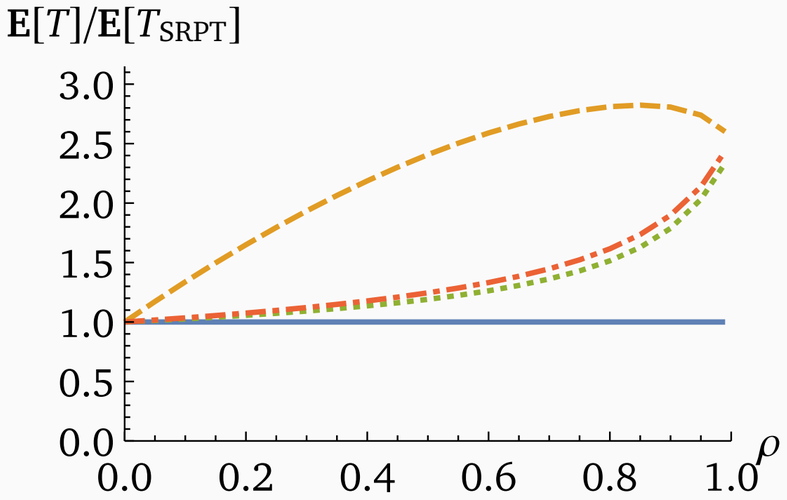
Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.

Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.

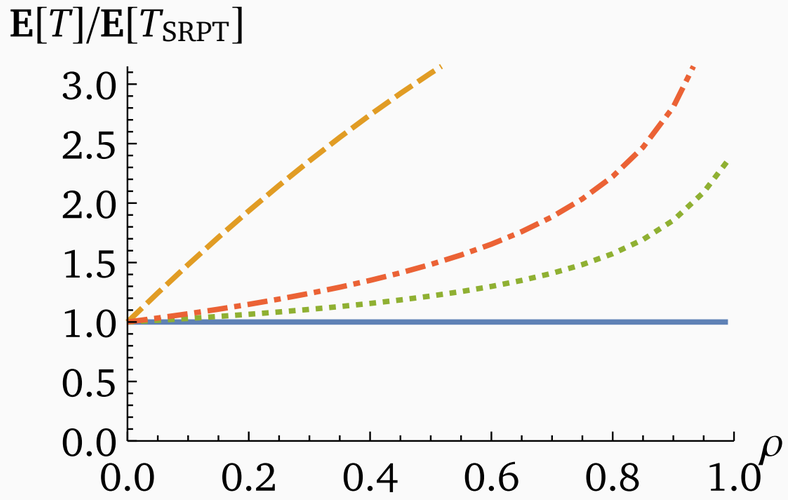
Low noise



Medium noise

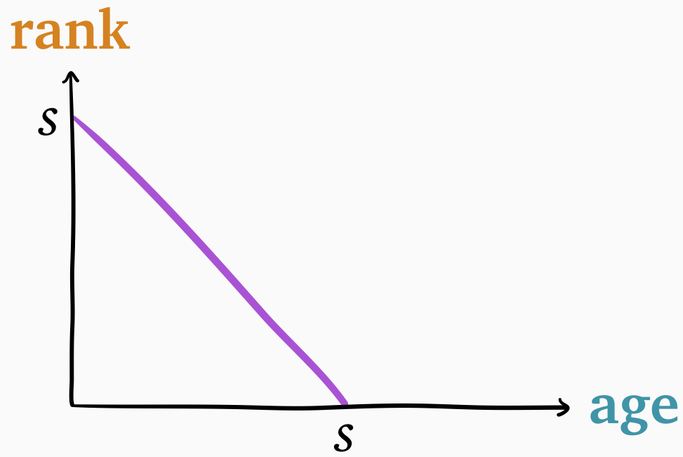


High noise

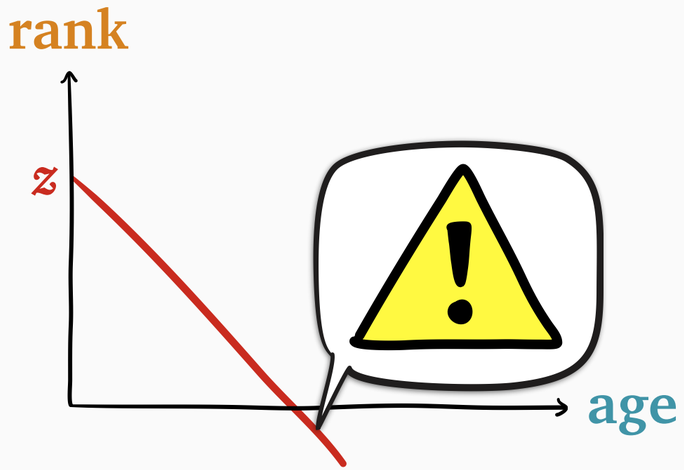


SRPT
Opt. w/ estimates
Noisy SRPT
Noisy PSJF

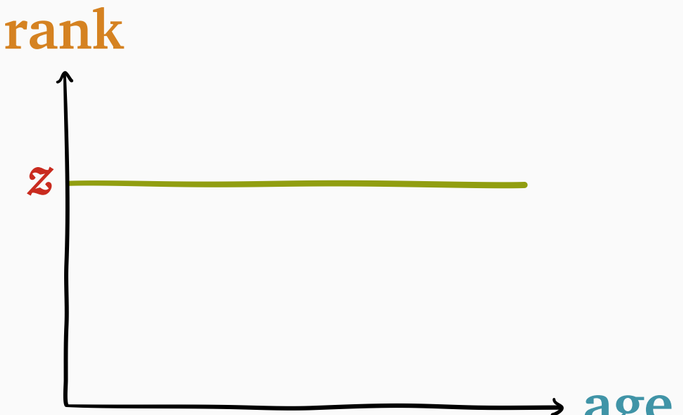
SRPT



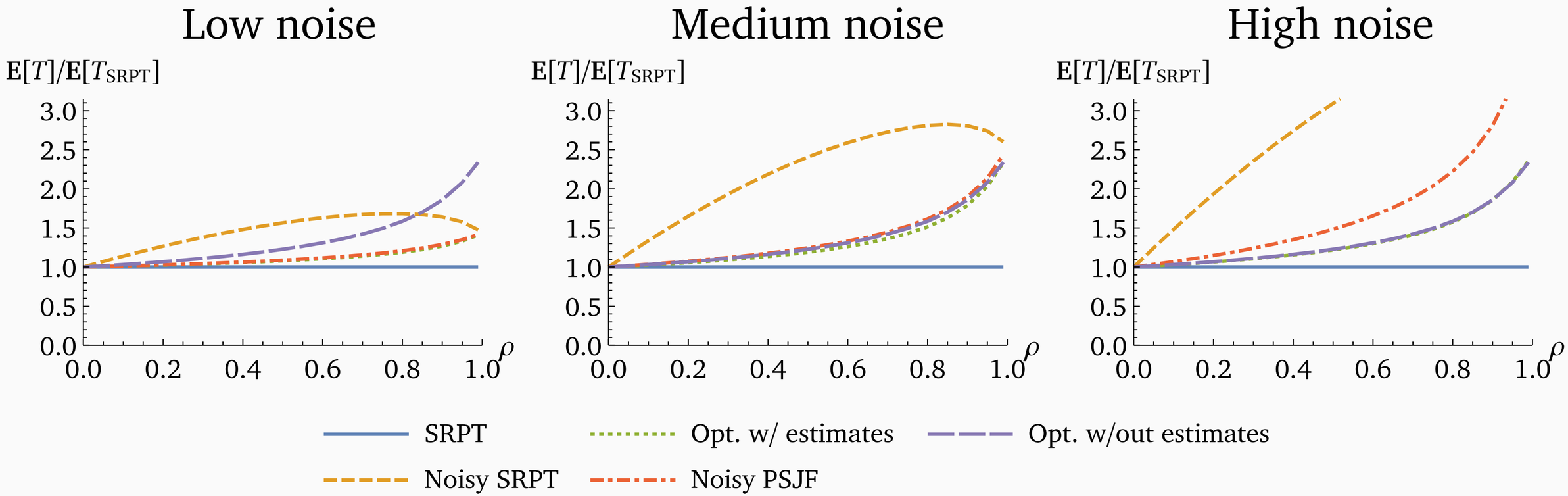
Noisy SRPT

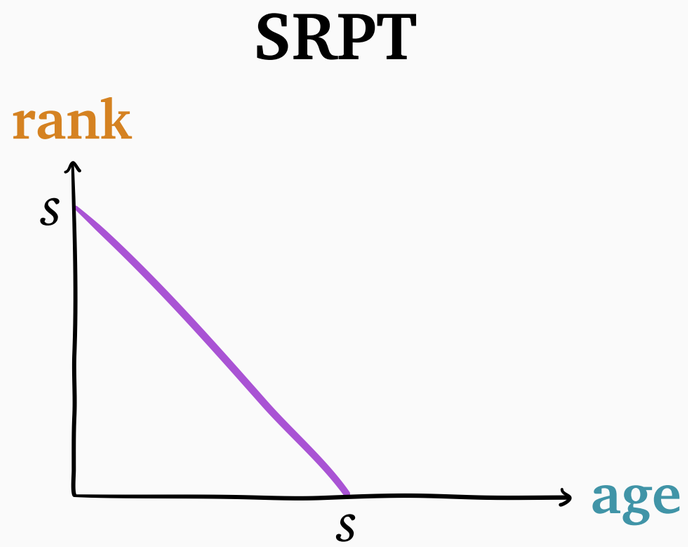


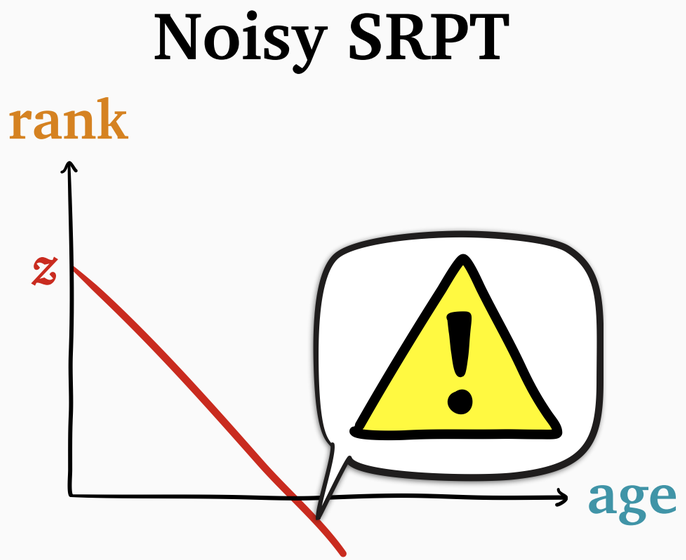
Noisy PSJF

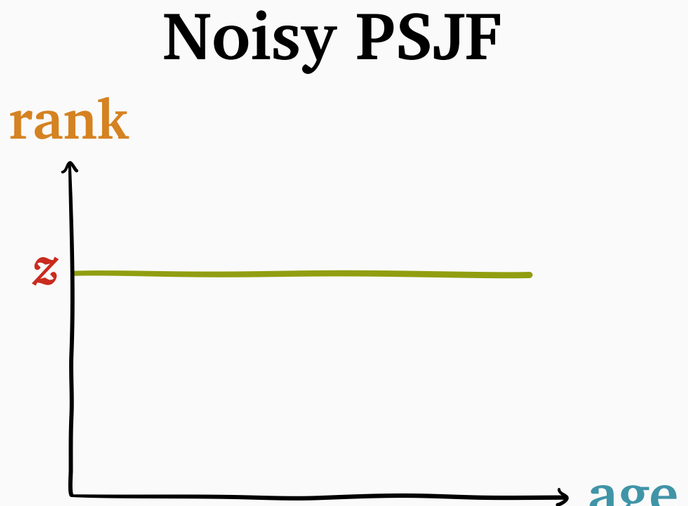


Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.

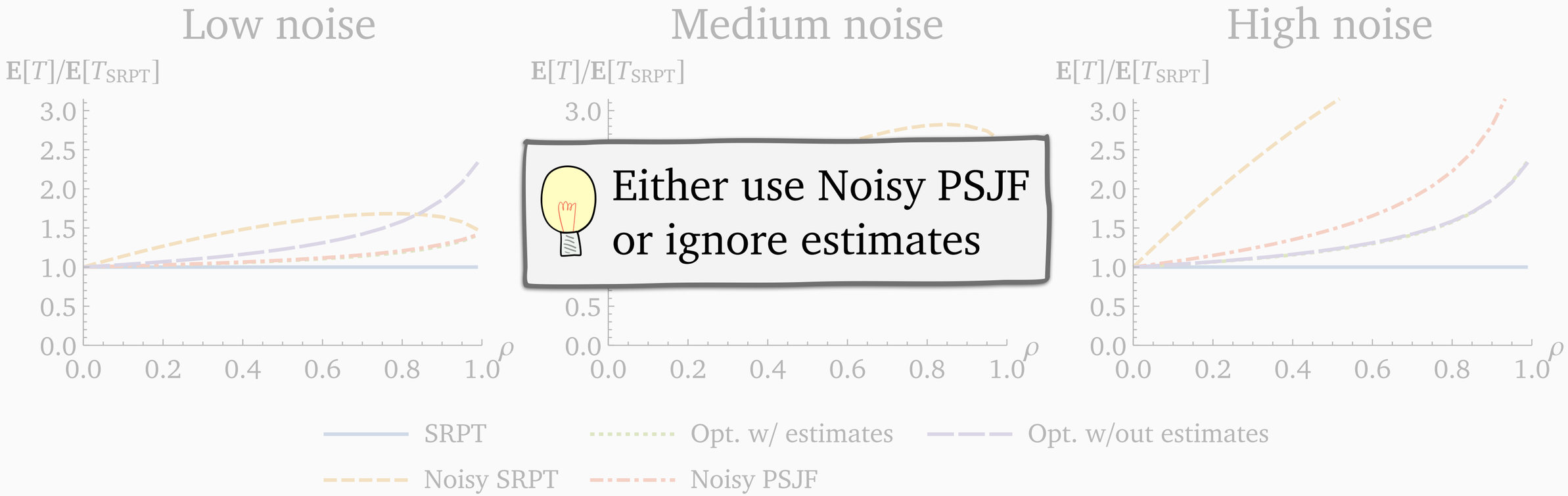


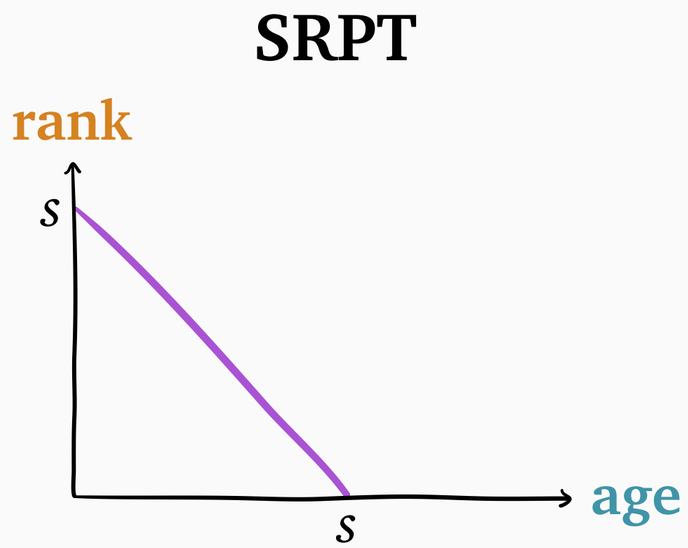


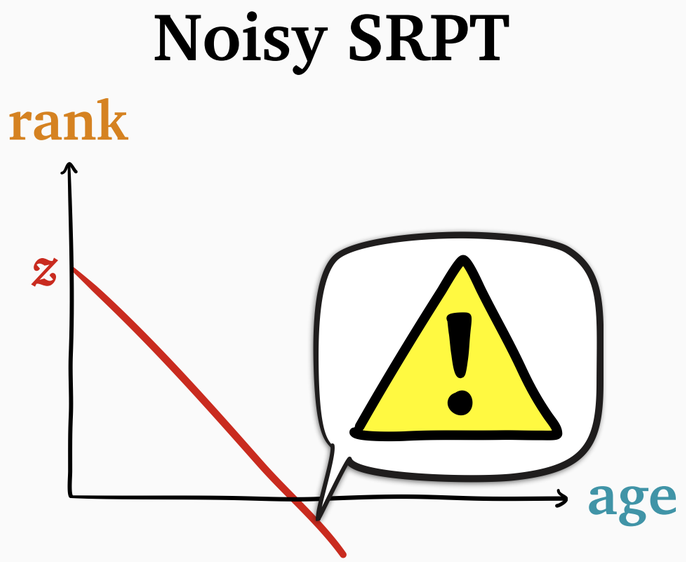


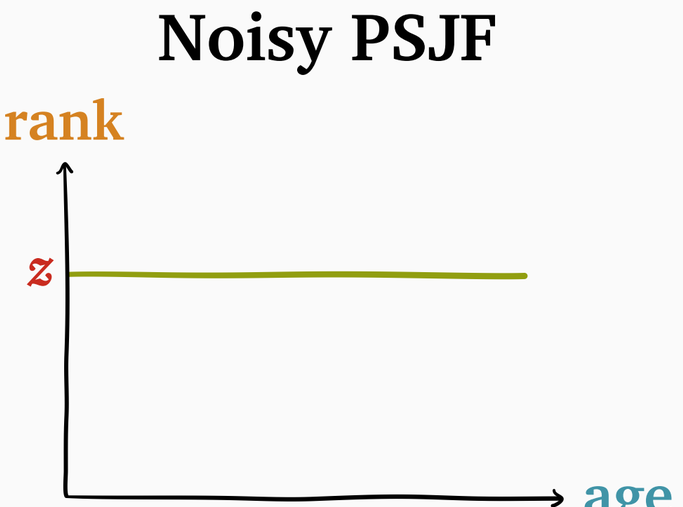


Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.

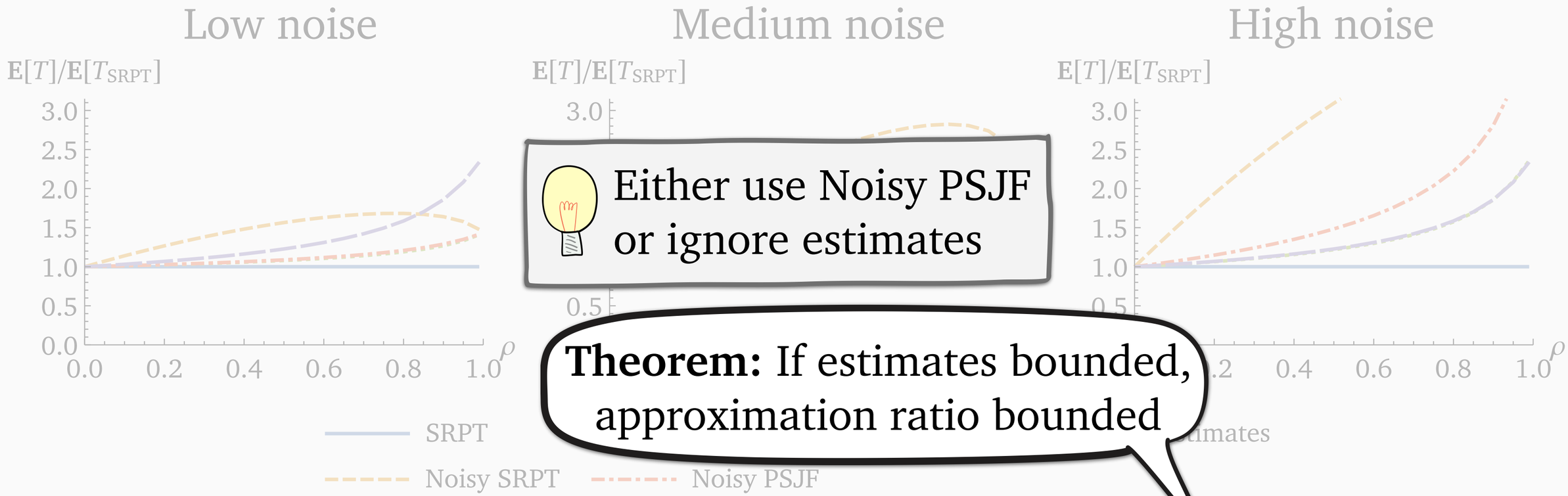








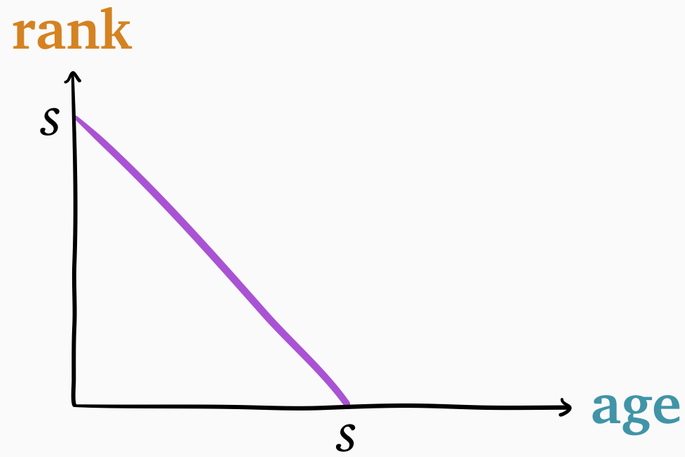
Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.



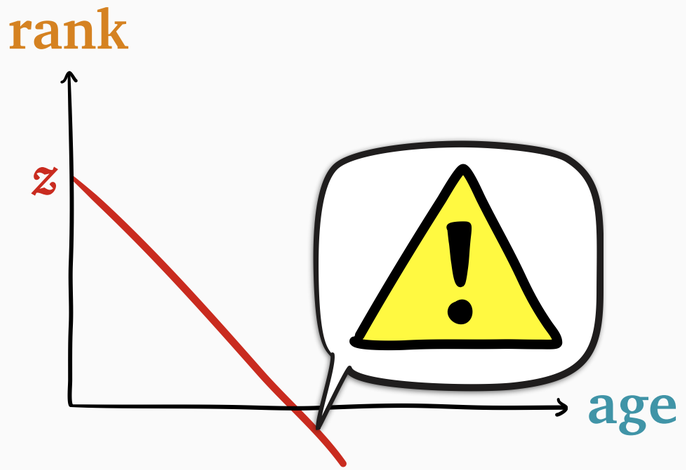
Either use Noisy PSJF or ignore estimates

Theorem: If estimates bounded, approximation ratio bounded

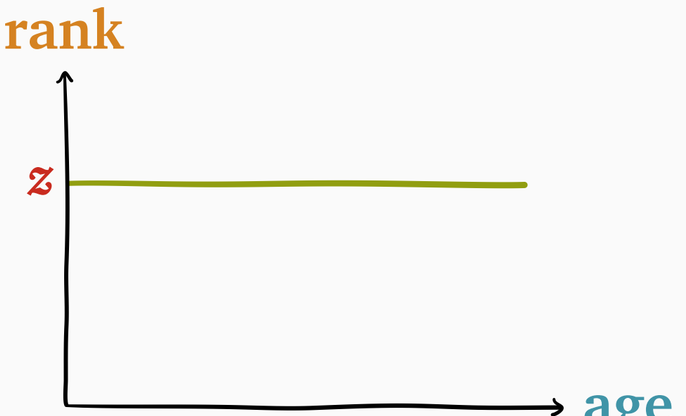
SRPT



Noisy SRPT

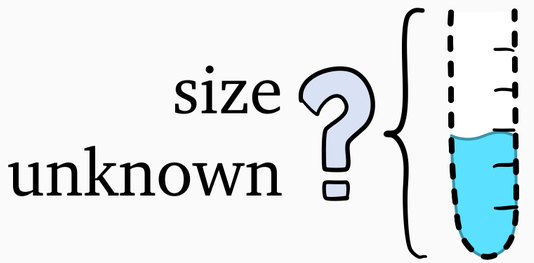


Noisy PSJF



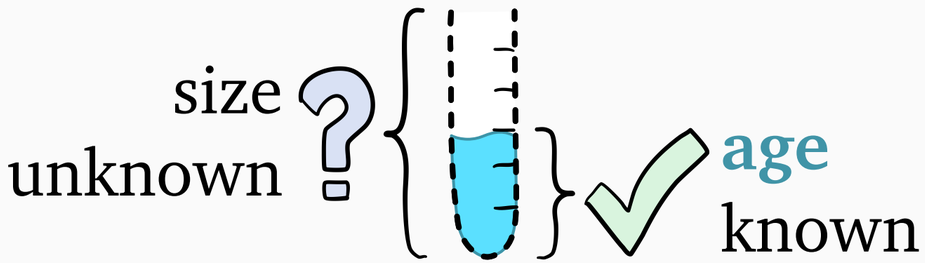
Scully, Grosof, & Mitzenmacher (2022). doi:10.4230/LIPIcs.ITCS.2022.114.
Scully (2022). https://ziv.codes/pdf/scully-thesis.pdf.

Optimal policy with no estimates



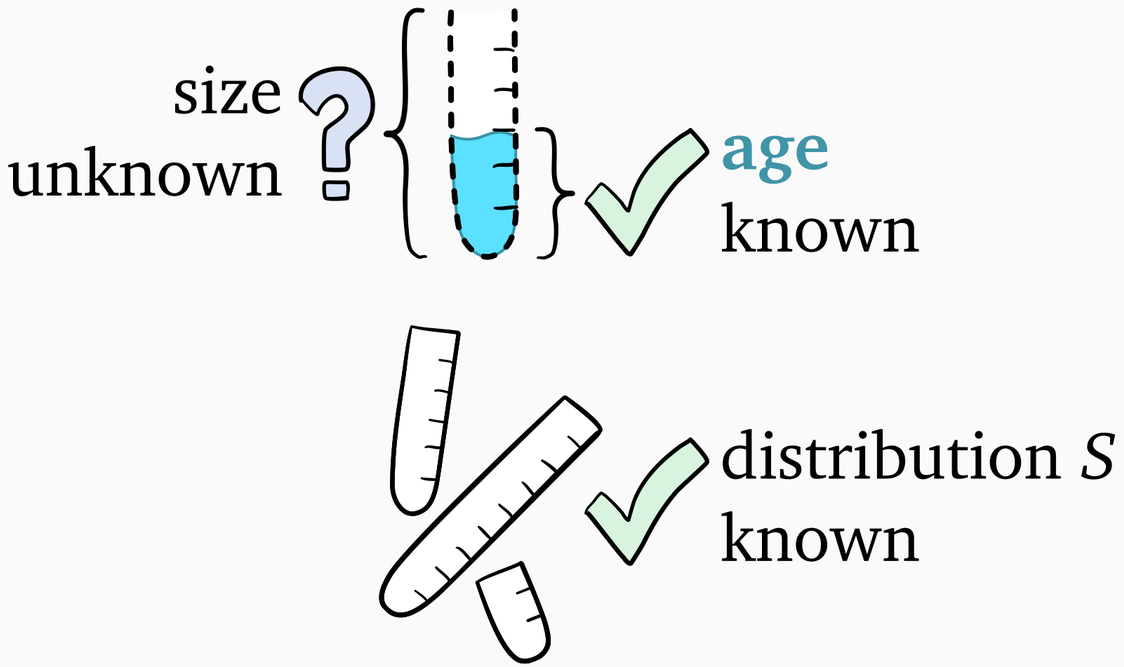
Gittins, Glazebrook, & Weber (2011). doi:10.1002/9780470980033.
Ramakrishna, Harlev, & Scully (2025). https://ziv.codes/pdf/mama2025-ramakrishna.pdf.

Optimal policy with no estimates



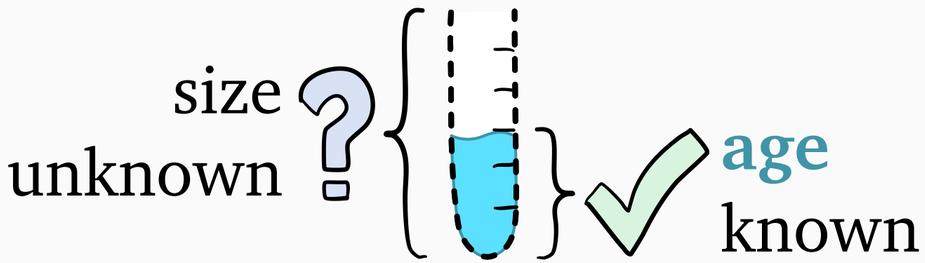
Gittins, Glazebrook, & Weber (2011). doi:10.1002/9780470980033.
Ramakrishna, Harlev, & Scully (2025). https://ziv.codes/pdf/mama2025-ramakrishna.pdf.

Optimal policy with no estimates

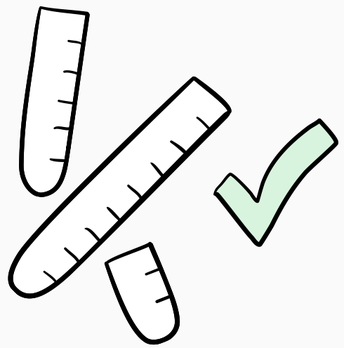


Gittins, Glazebrook, & Weber (2011). doi:10.1002/9780470980033.
Ramakrishna, Harlev, & Scully (2025). https://ziv.codes/pdf/mama2025-ramakrishna.pdf.

Optimal policy with no estimates



age
known

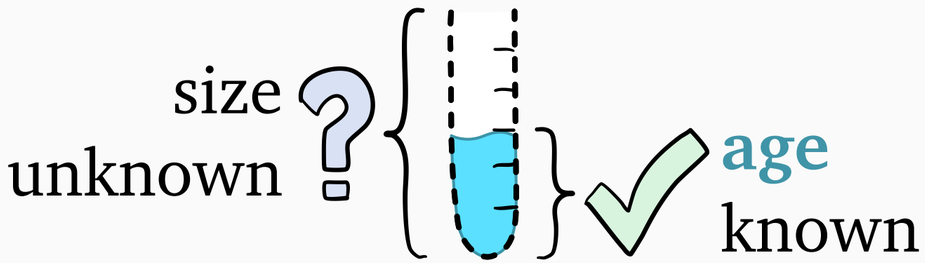


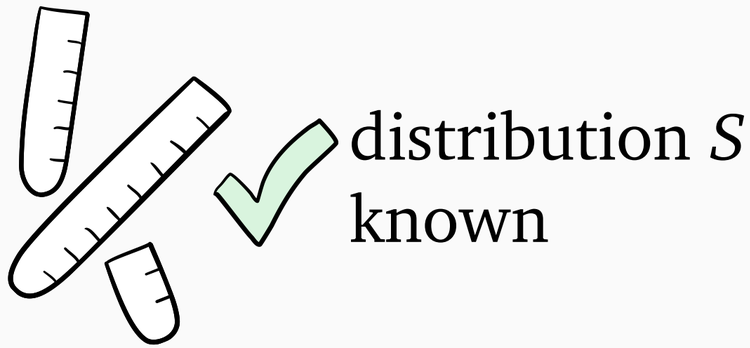
distribution S
known

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Gittins, Glazebrook, & Weber (2011). doi:10.1002/9780470980033.
Ramakrishna, Harlev, & Scully (2025). https://ziv.codes/pdf/mama2025-ramakrishna.pdf.

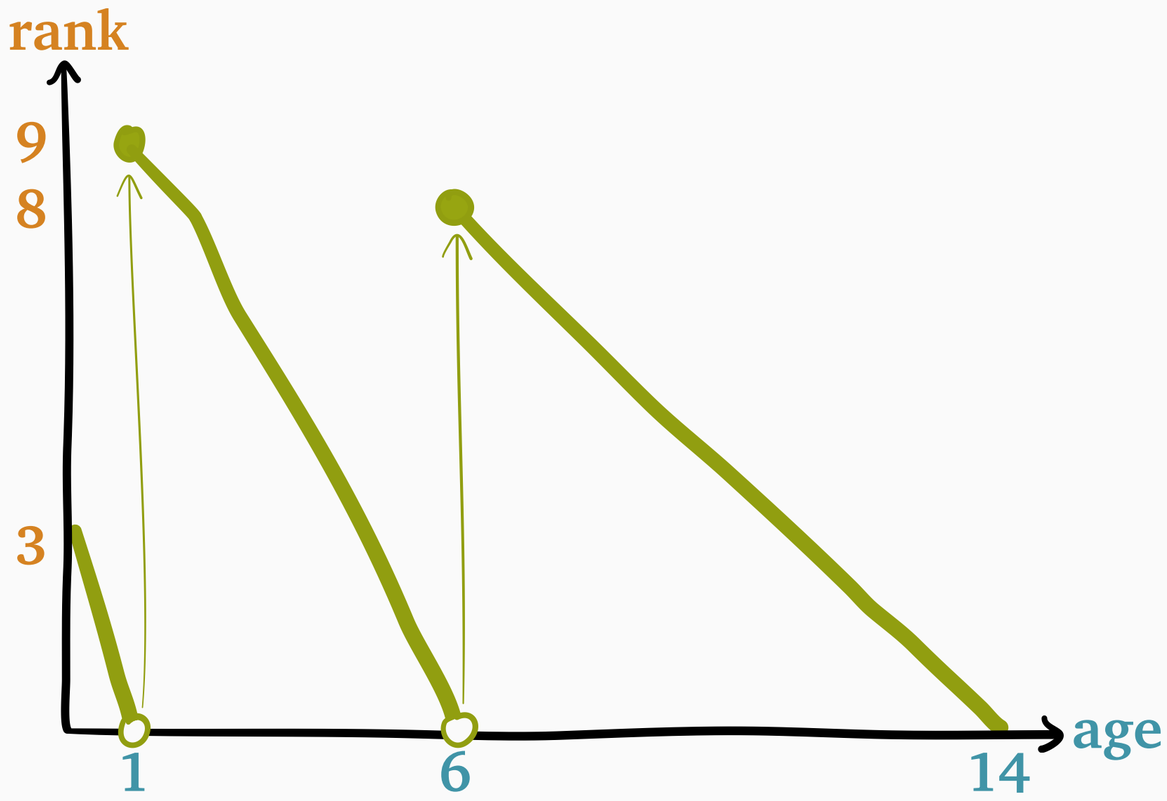
Optimal policy with no estimates





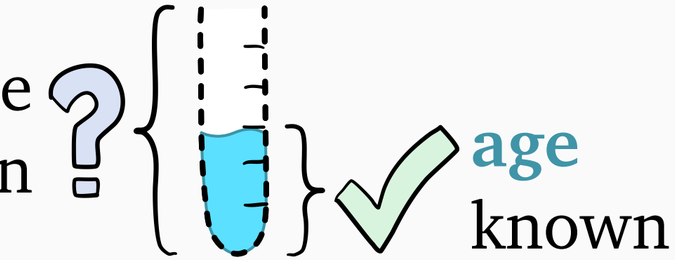
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

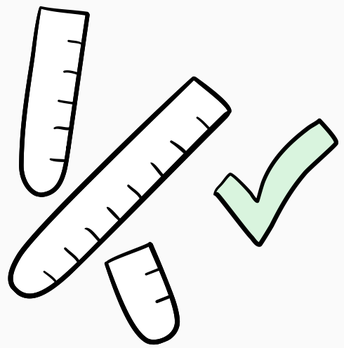
Gittins policy



Gittins, Glazebrook, & Weber (2011). doi:10.1002/9780470980033.
Ramakrishna, Harlev, & Scully (2025). https://ziv.codes/pdf/mama2025-ramakrishna.pdf.

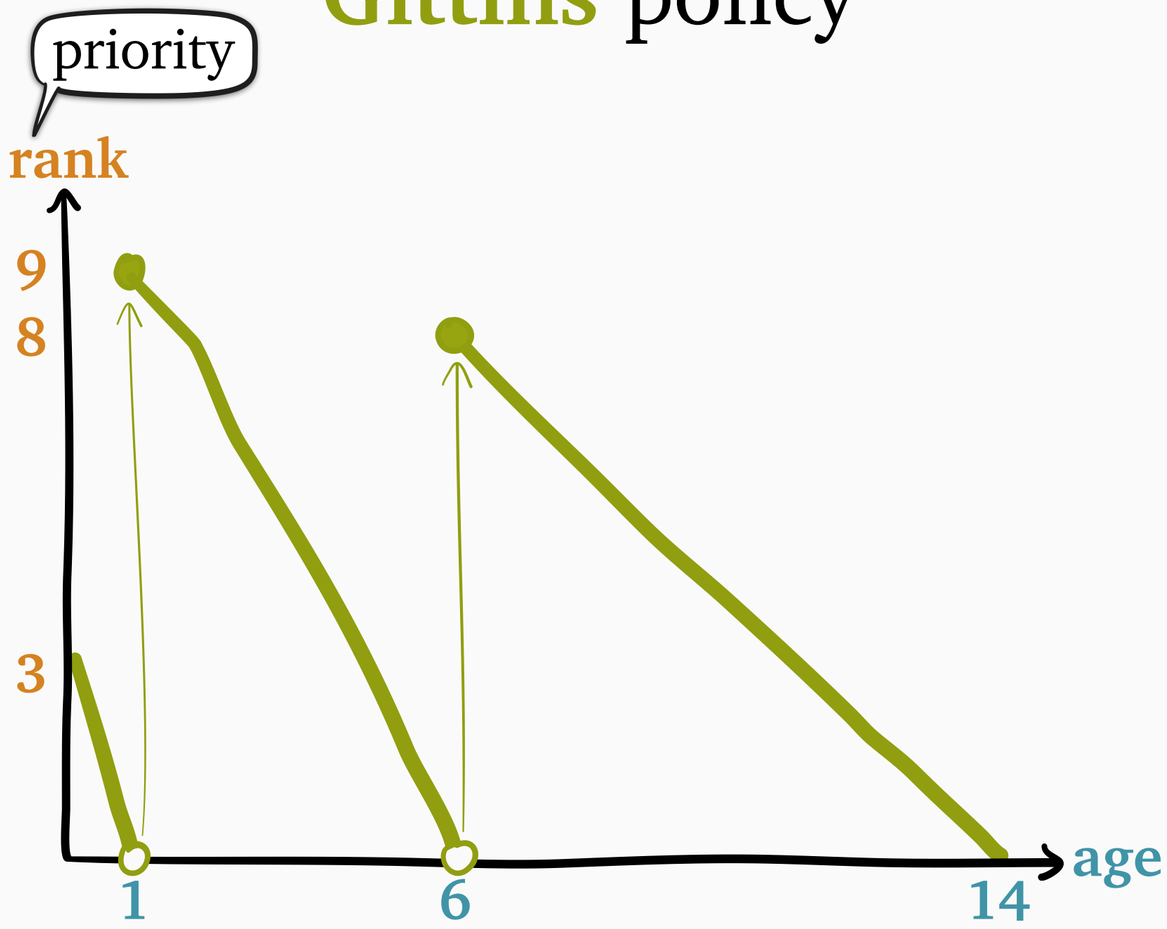
Optimal policy with no estimates





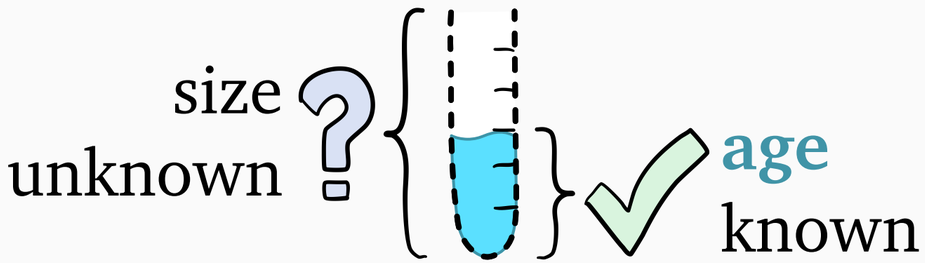
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

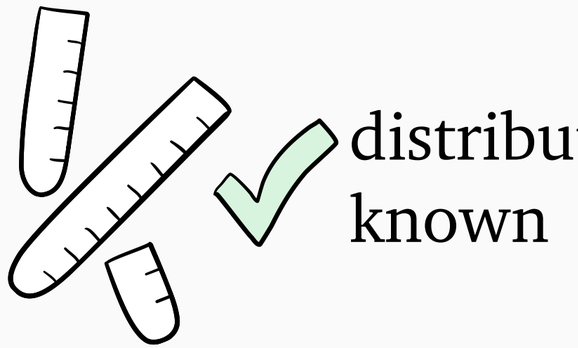
Gittins policy



Gittins, Glazebrook, & Weber (2011). doi:10.1002/9780470980033.
Ramakrishna, Harlev, & Scully (2025). https://ziv.codes/pdf/mama2025-ramakrishna.pdf.

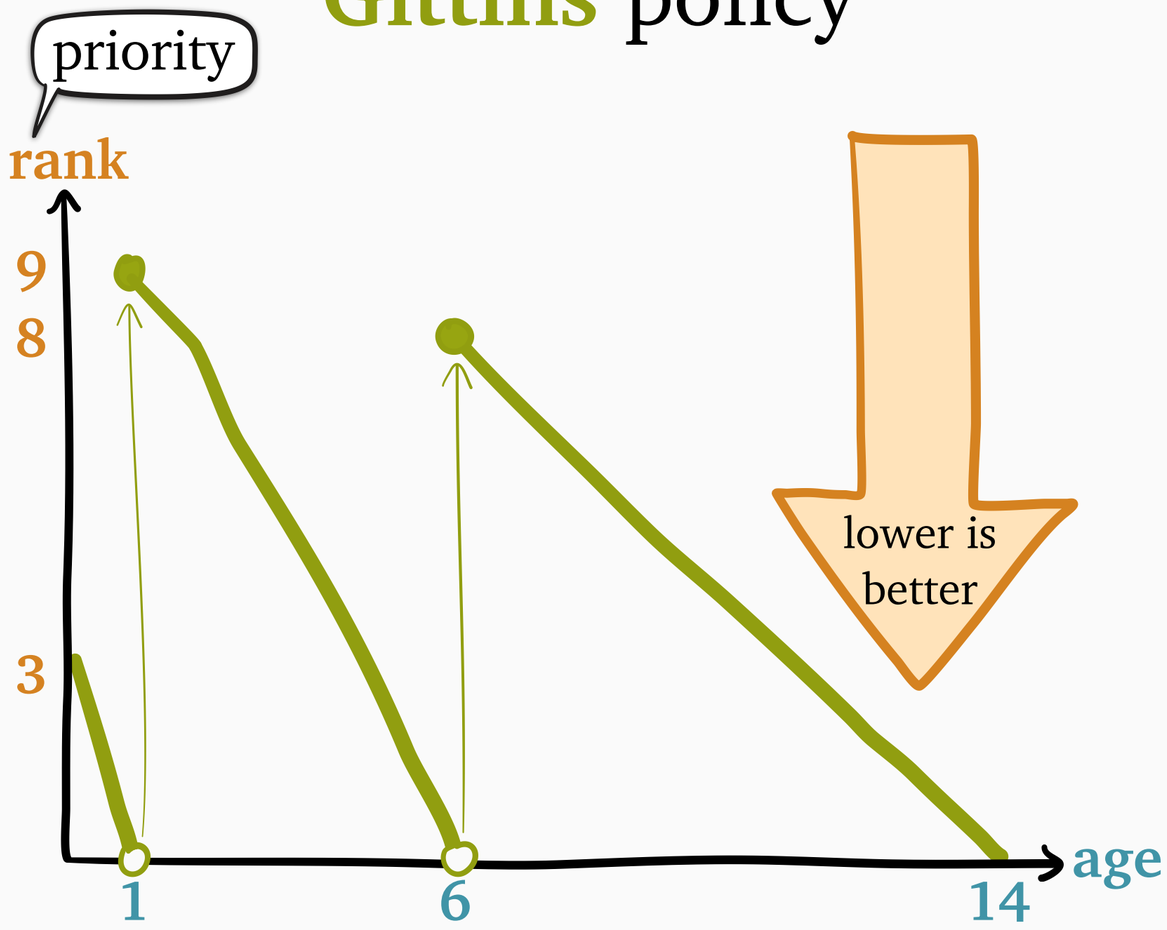
Optimal policy with no estimates





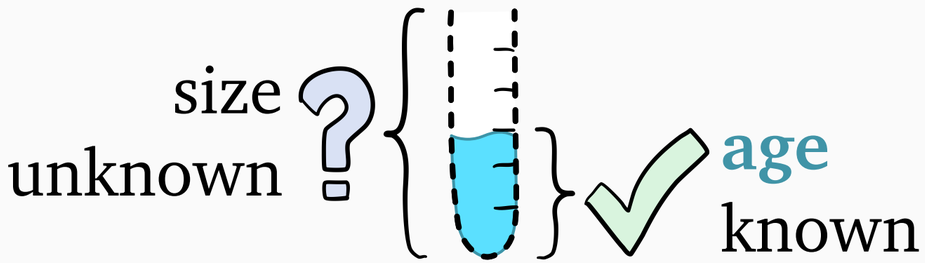
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

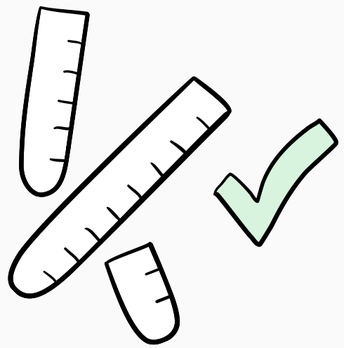
Gittins policy



Gittins, Glazebrook, & Weber (2011). doi:10.1002/9780470980033.
Ramakrishna, Harlev, & Scully (2025). https://ziv.codes/pdf/mama2025-ramakrishna.pdf.

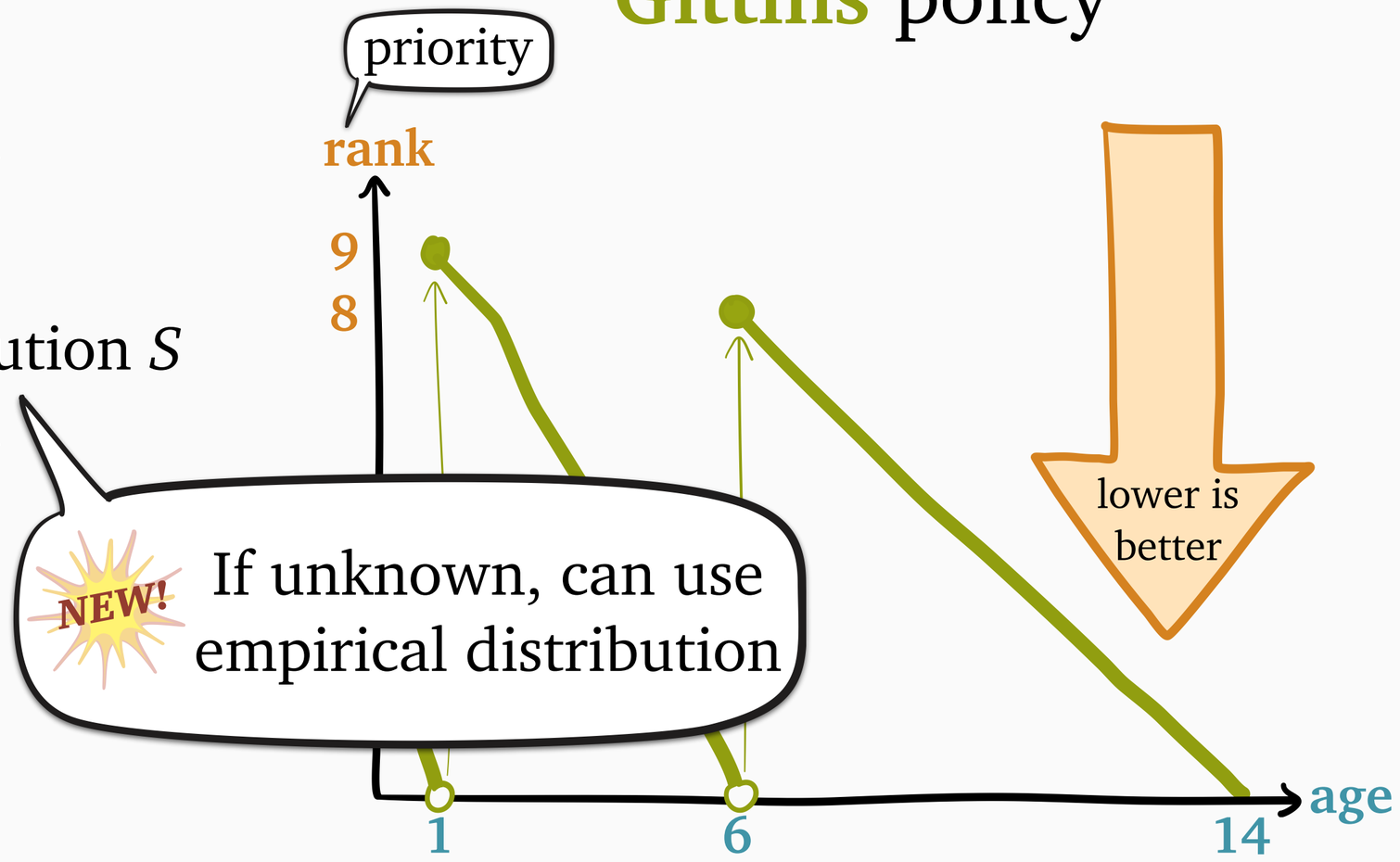
Optimal policy with no estimates





$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

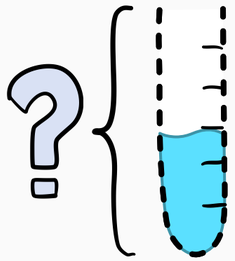
Gittins policy



Gittins, Glazebrook, & Weber (2011). doi:10.1002/9780470980033.
Ramakrishna, Harlev, & Scully (2025). https://ziv.codes/pdf/mama2025-ramakrishna.pdf.

Gittins construction generalizes

state = age

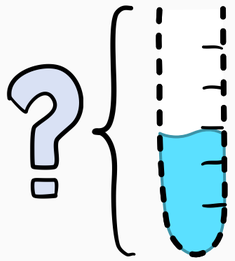


minimize

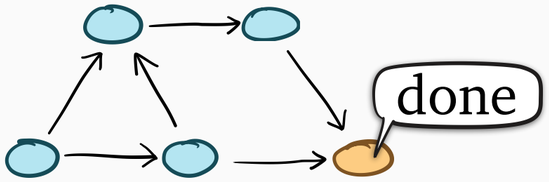
$$\mathbf{E}[N] = \lambda \mathbf{E}[T]$$

Gittins construction generalizes

state = age



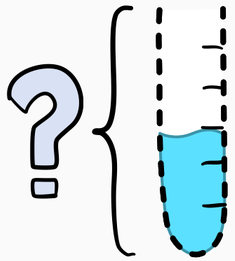
model service as
Markov process



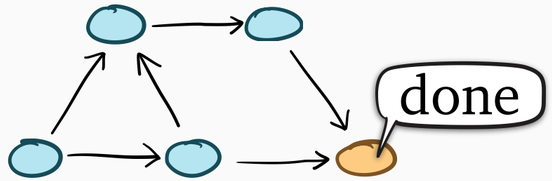
minimize
$\mathbf{E}[N] = \lambda \mathbf{E}[T]$

Gittins construction generalizes

state = **age**



model service as
Markov process

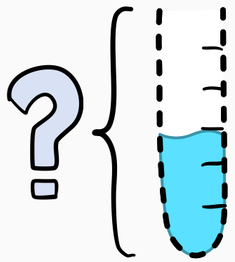


minimize
$\mathbf{E}[N] = \lambda \mathbf{E}[T]$

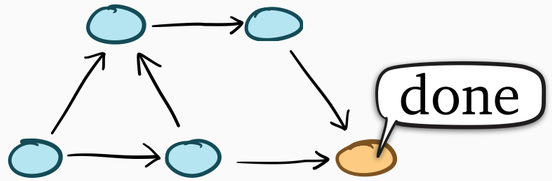
minimize mean sum of
state-based holding costs

Gittins construction generalizes

state = **age**

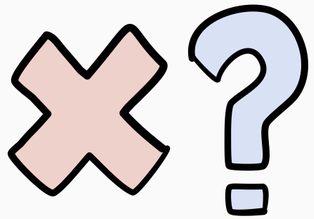


model service as
Markov process



minimize
$\mathbf{E}[N] = \lambda \mathbf{E}[T]$

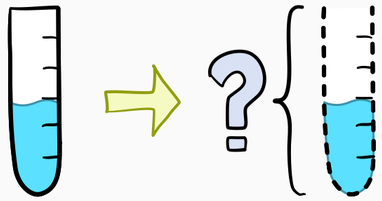
minimize mean sum of
state-based holding costs



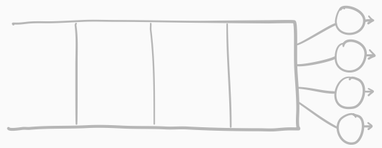
Open: *time-based* holding costs

Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52861.2021.9589051.

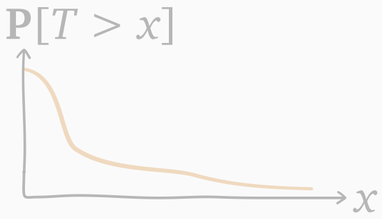
Hard scheduling questions



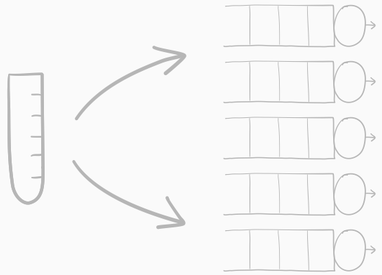
What if we have
unknown job sizes?



What if there are
multiple servers?

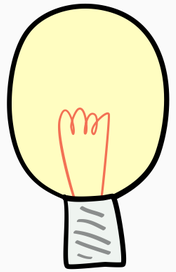


What if we want to optimize
tails instead of means?



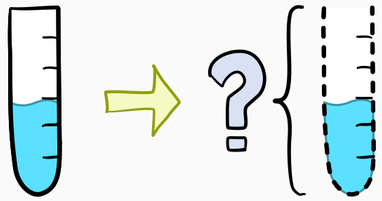
What if we can only use
dispatching without fancy scheduling?

What are the underlying
theoretical tools?

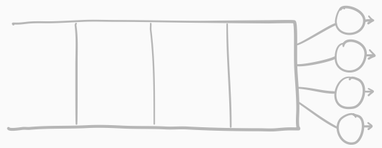


What are the resulting
practical lessons?

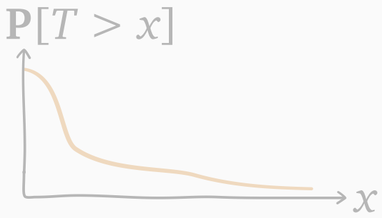
Hard scheduling questions



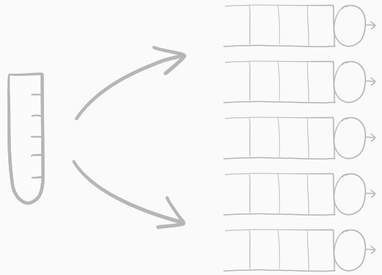
What if we have
unknown job sizes?



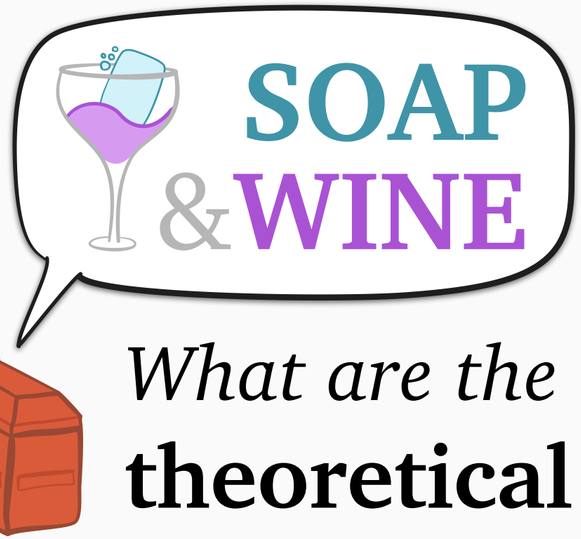
What if there are
multiple servers?



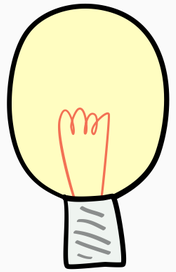
What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

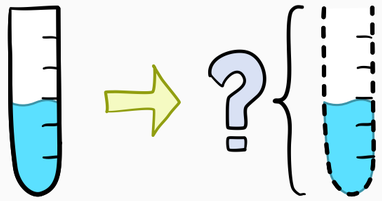


What are the underlying
theoretical tools?

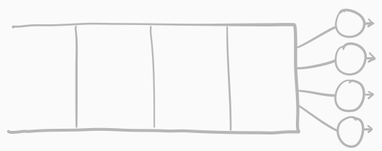


What are the resulting
practical lessons?

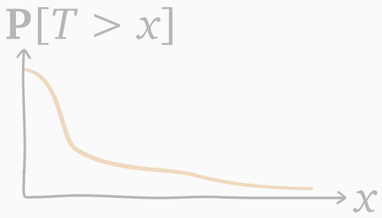
Hard scheduling questions



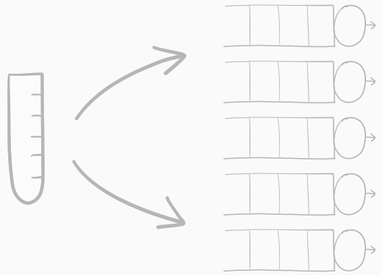
What if we have
unknown job sizes?



What if there are
multiple servers?

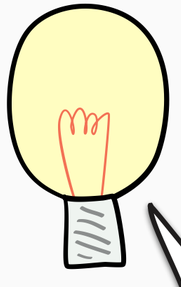


What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

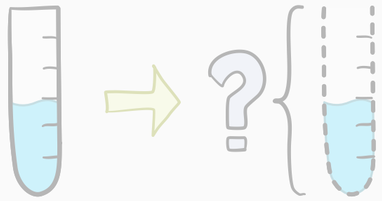
What are the underlying
theoretical tools?



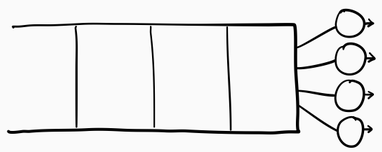
What are the resulting
practical lessons?

- PSJF is safer than SRPT
- Ignore terrible estimates
- **Gittins** works with data

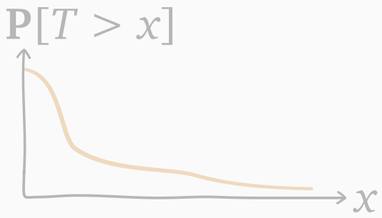
Hard scheduling questions



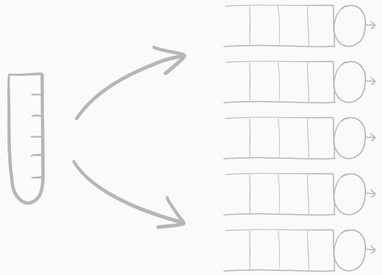
What if we have
unknown job sizes?



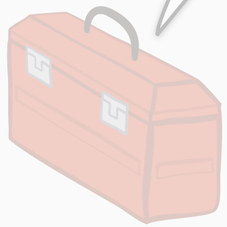
What if there are
multiple servers?



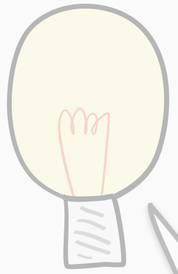
What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?



What are the underlying
theoretical tools?



What are the resulting
practical lessons?

- PSJF is safer than SRPT
- Ignore terrible estimates
- **Gittins** works with data

WINE and r-work

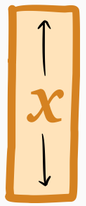
$W(r)$ = work relevant to **rank r**

Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.
Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52861.2021.9589051.

WINE and r-work

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = {

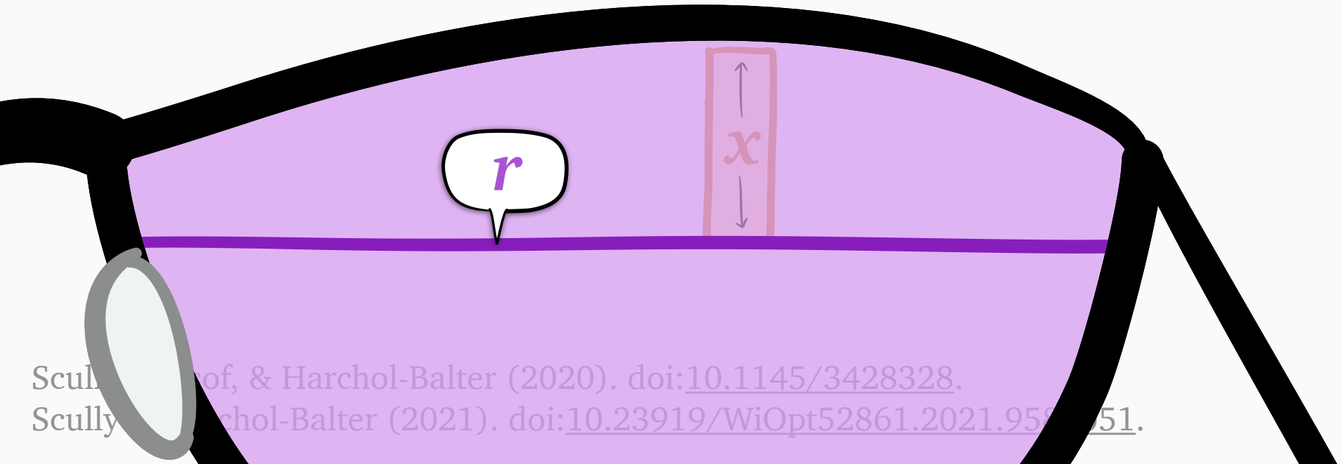


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.
Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52861.2021.9589051.

WINE and r-work

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases}$

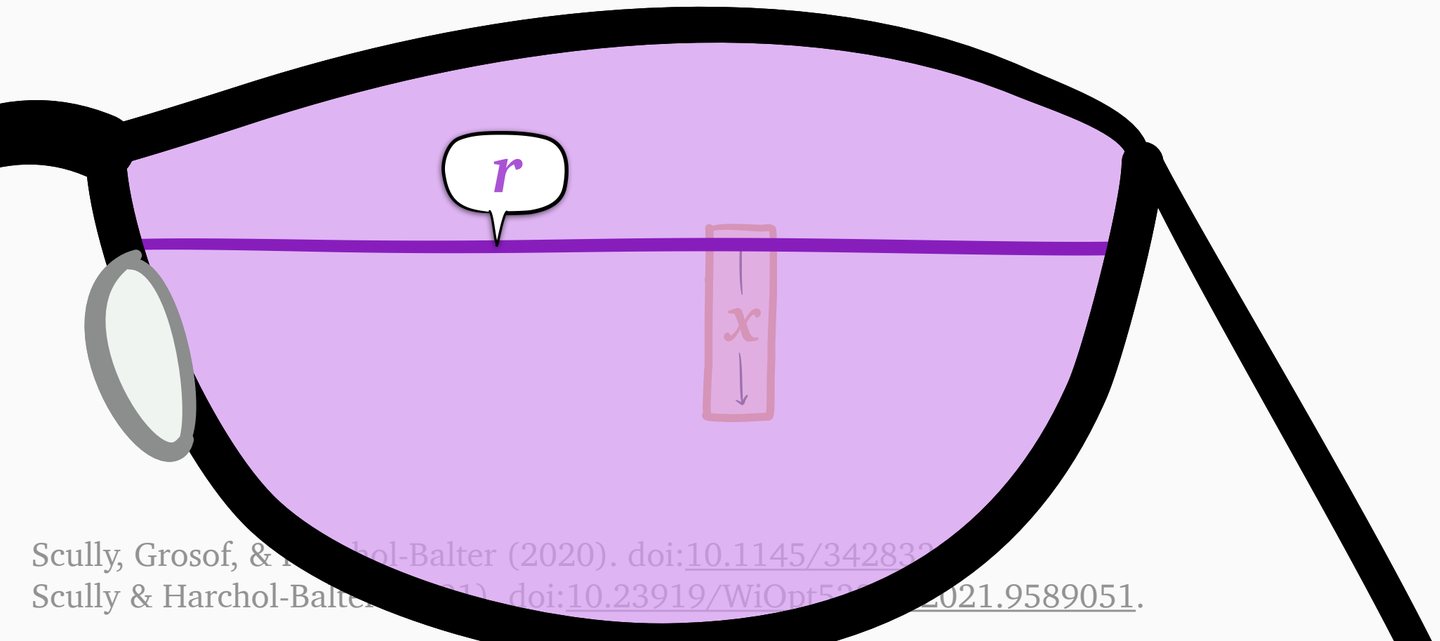


Scull[illegible]of, & Harchol-Balter (2020). doi:10.1145/3428328.
Scully[illegible]chol-Balter (2021). doi:10.23919/WiOpt52861.2021.95[illegible]51.

WINE and r-work

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases}$

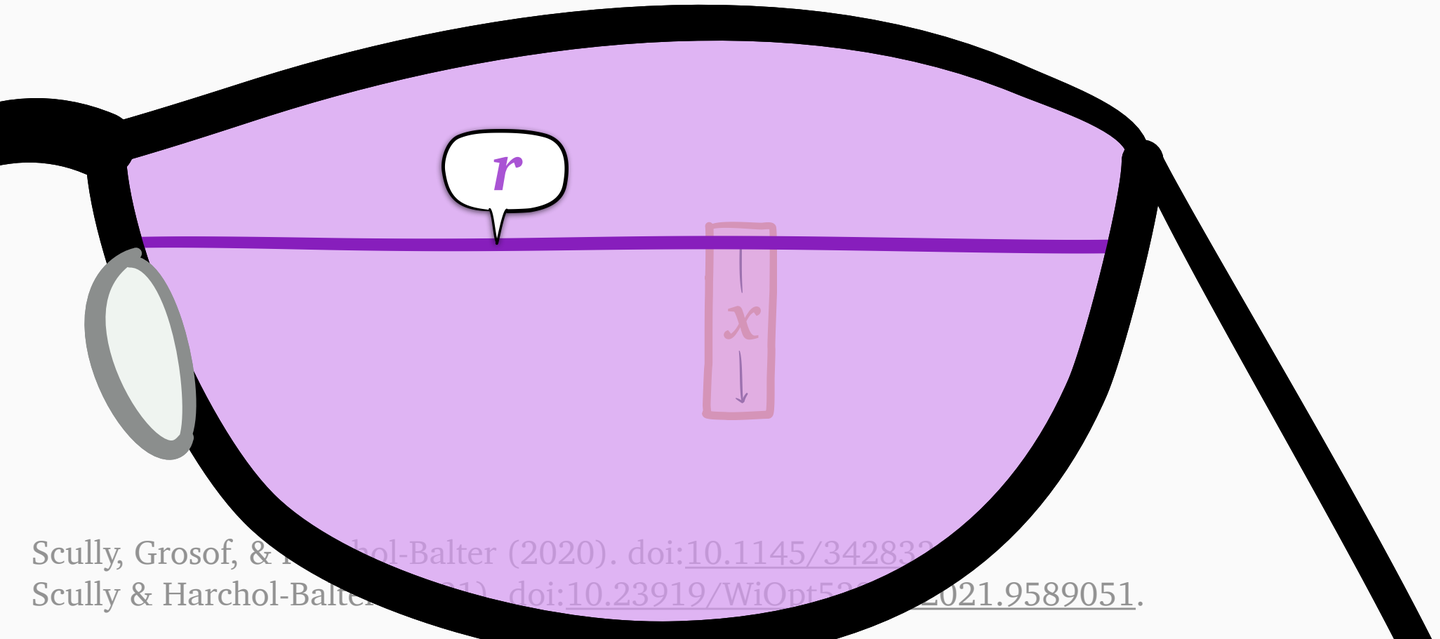


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428330.
Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52691.2021.9589051.

WINE and r-work

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \end{cases}$

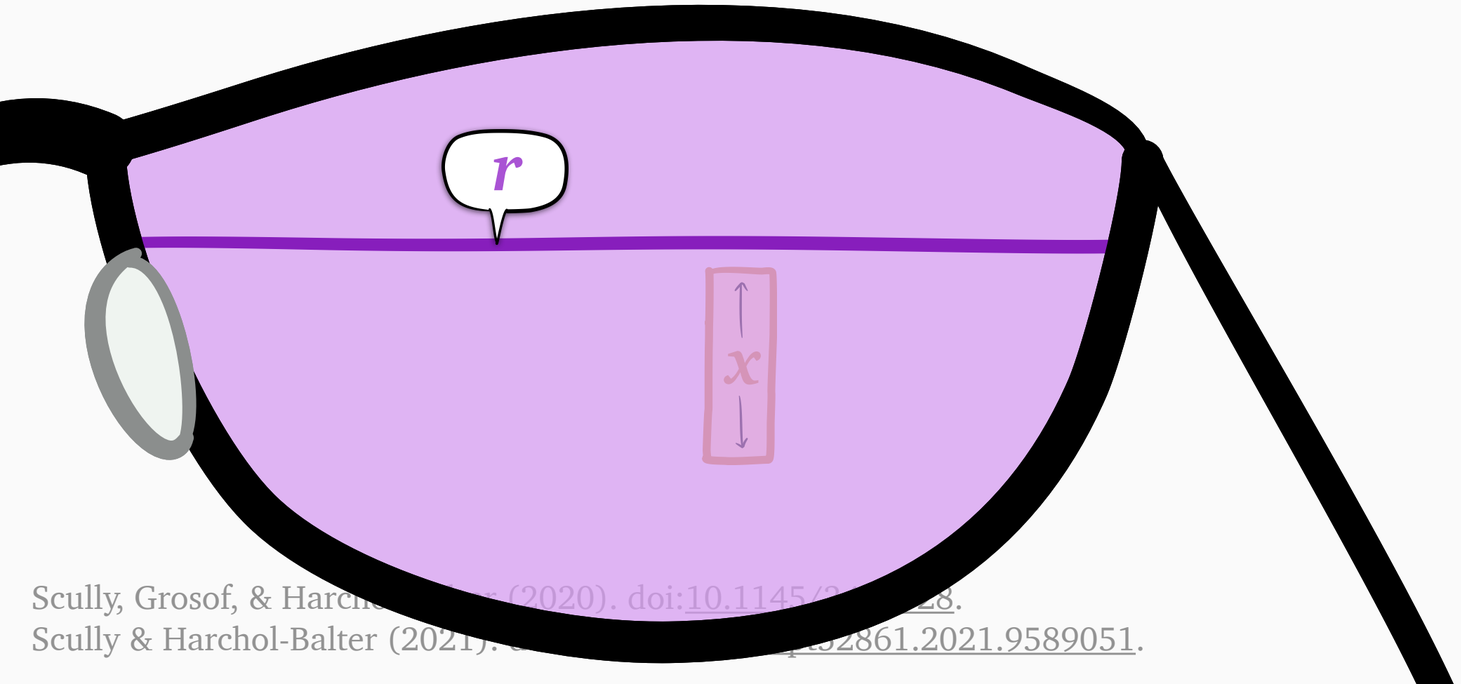


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428330.
Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52691.2021.9589051.

WINE and r-work

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \end{cases}$

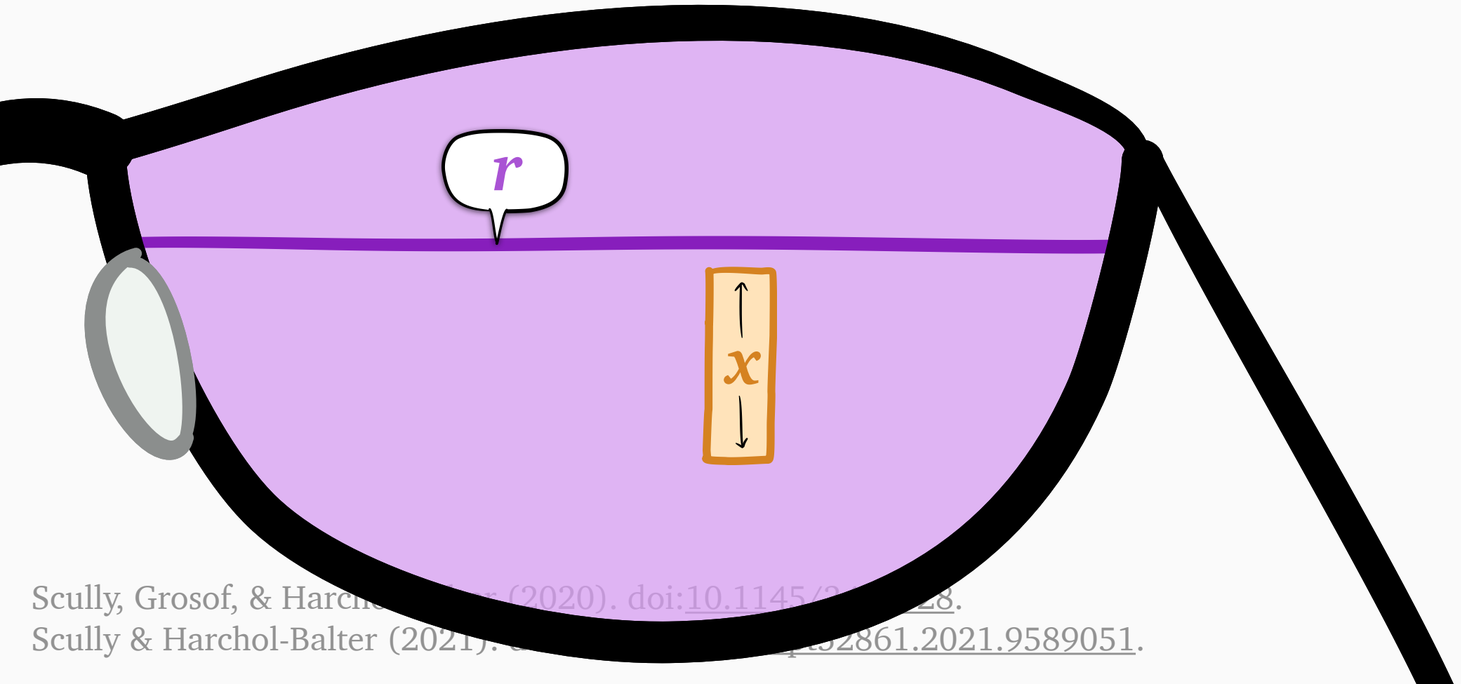


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3[illegible]28.
Scully & Harchol-Balter (2021). [illegible]4[illegible]52861.2021.9589051.

WINE and r-work

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \end{cases}$

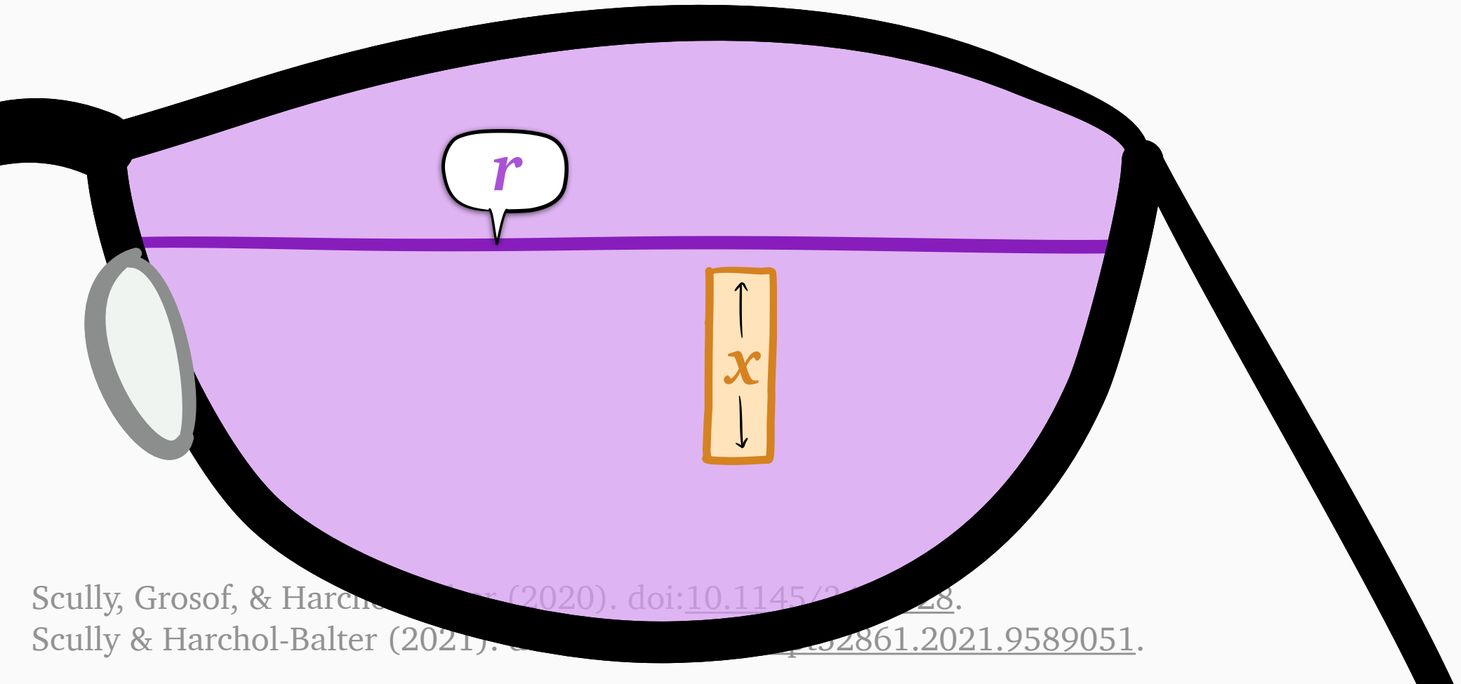


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3[illegible]28.
Scully & Harchol-Balter (2021). [illegible]4[illegible]52861.2021.9589051.

WINE and r-work

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$

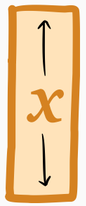


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3[illegible]28.
Scully & Harchol-Balter (2021). [illegible]4[illegible]52861.2021.9589051.

WINE and r-work

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$

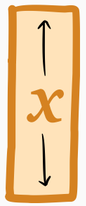


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.
Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52861.2021.9589051.

WINE and r-work

$$W(r) = \text{work relevant to rank } r$$
$$= \text{total } r\text{-work of all jobs}$$

$$w_x(r) = r\text{-work of } \textit{single job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

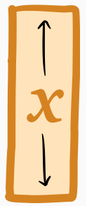


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.
Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52861.2021.9589051.

WINE and r-work

$$W(r) = \text{work relevant to rank } r$$
$$= \text{total } r\text{-work of all jobs}$$

$$w_x(r) = r\text{-work of } \textit{single job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$



Theorem: in *any* system,

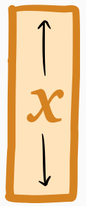
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.
Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52861.2021.9589051.

WINE and r-work

$W(r)$ = work relevant to **rank r**
= total r-work of all jobs

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$



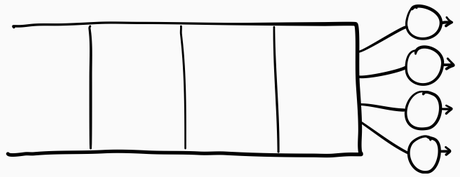
Theorem: in *any* system,

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

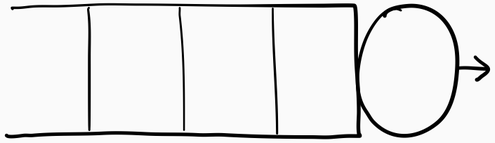
Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.
Scully & Harchol-Balter (2021). doi:10.23919/WiOpt52861.2021.9589051.

SRPT in the M/G/*k*

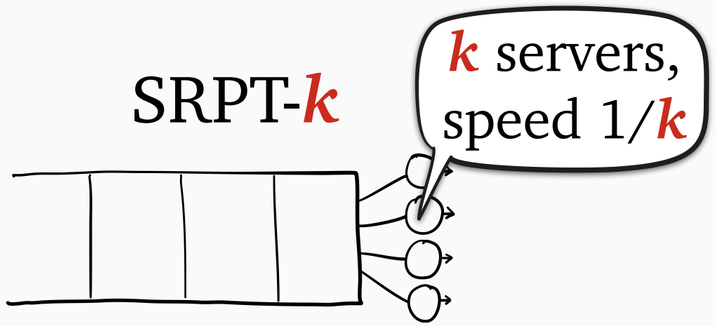
SRPT-*k*

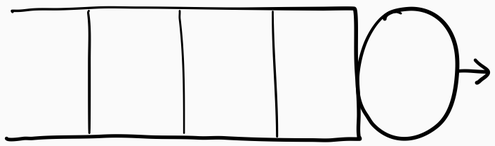


SRPT-1



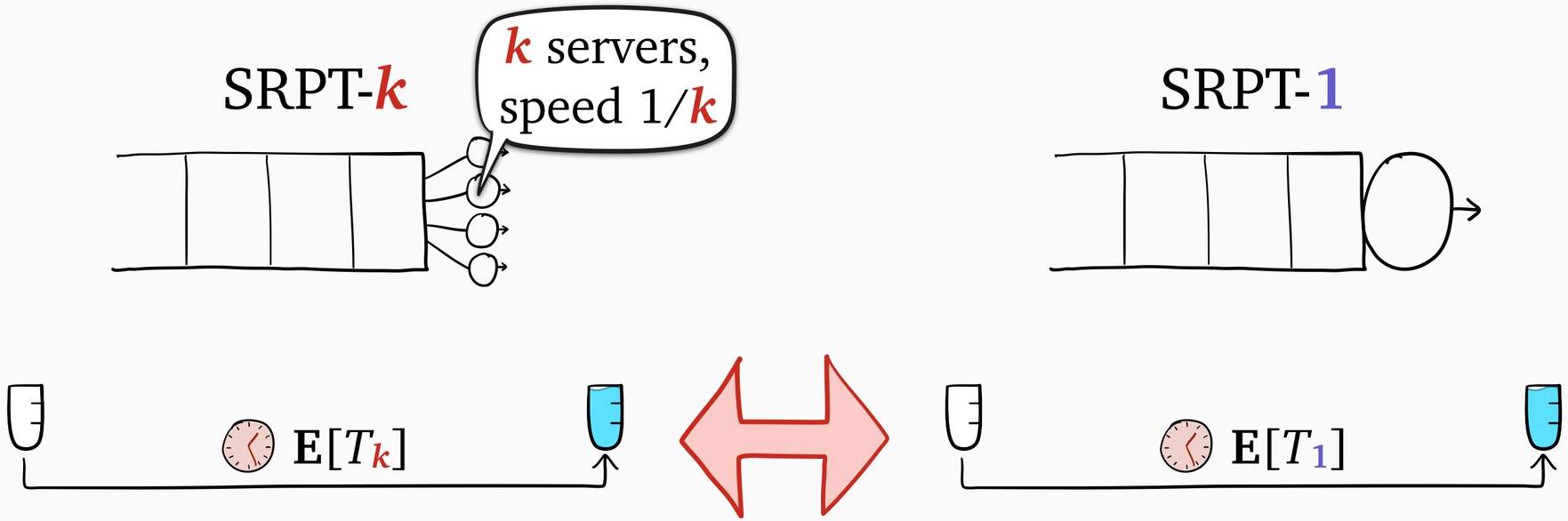
SRPT in the M/G/k



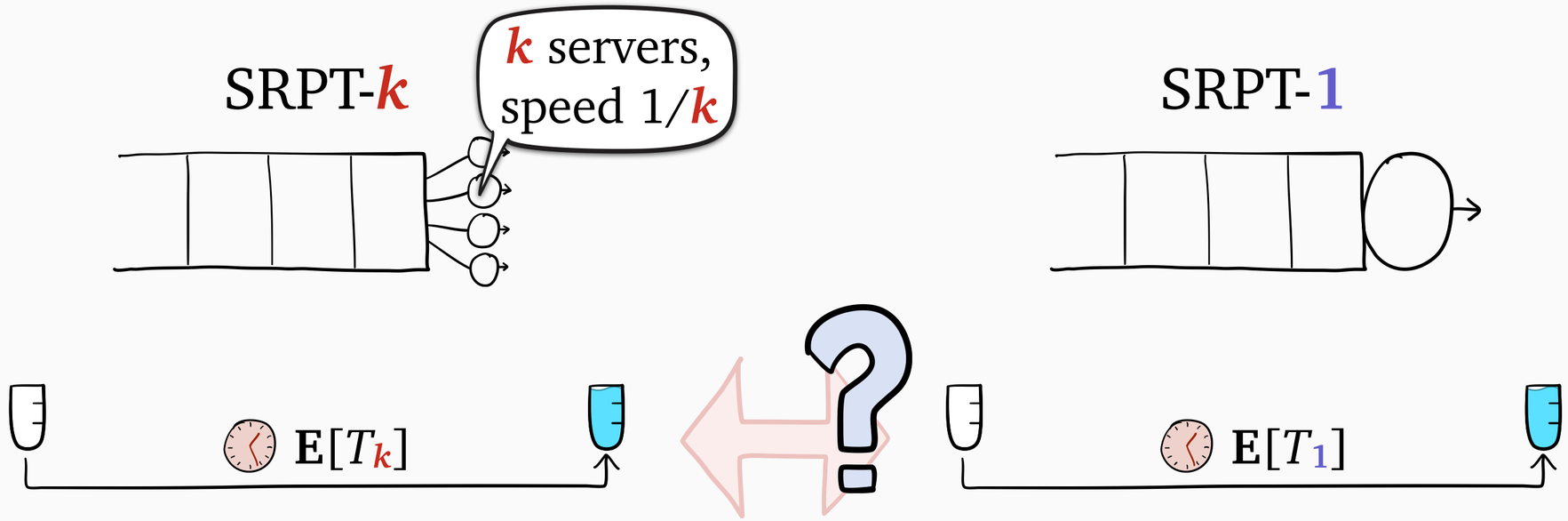


Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.

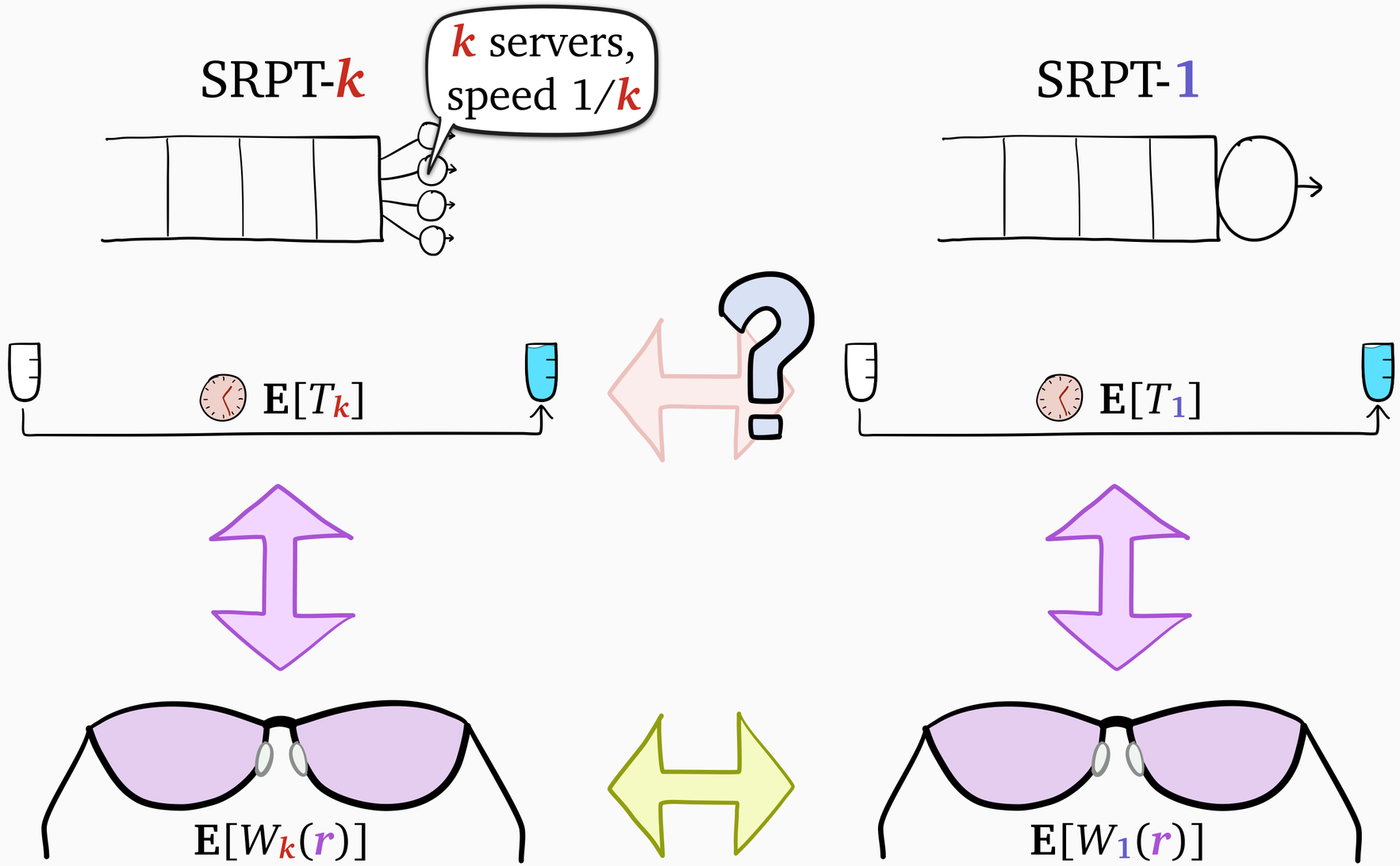
SRPT in the M/G/k



SRPT in the M/G/k

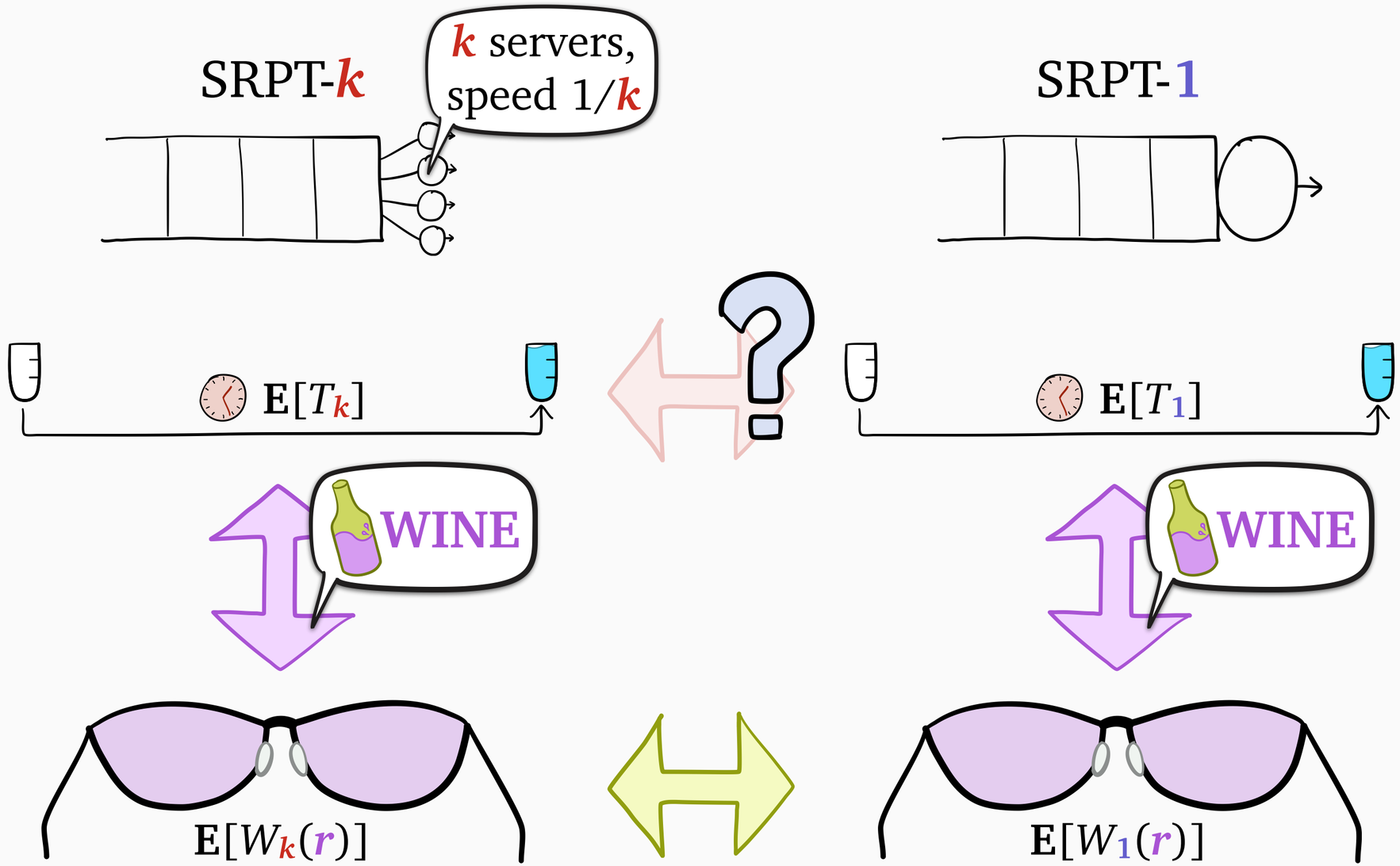


SRPT in the M/G/k



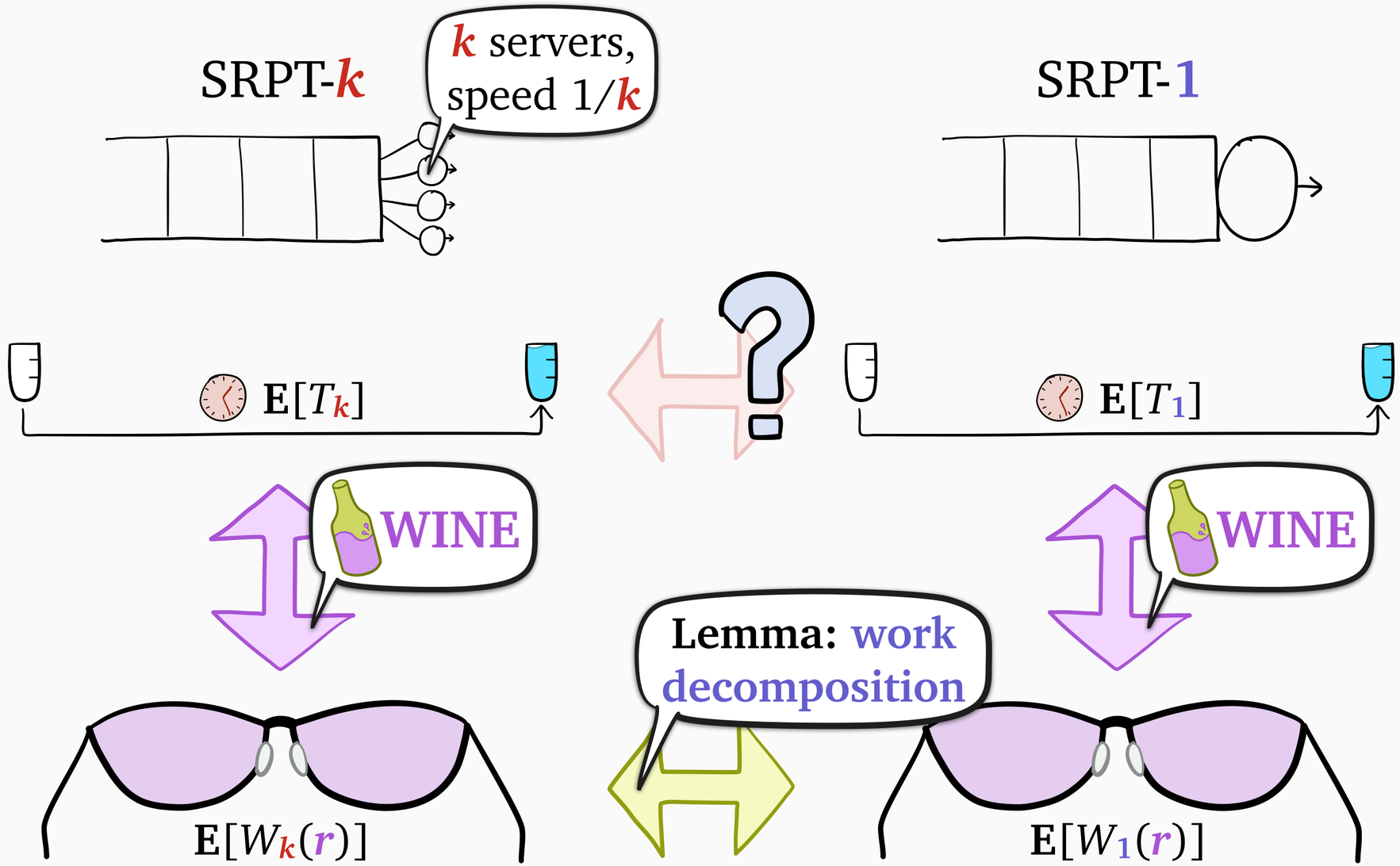
Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.

SRPT in the M/G/k



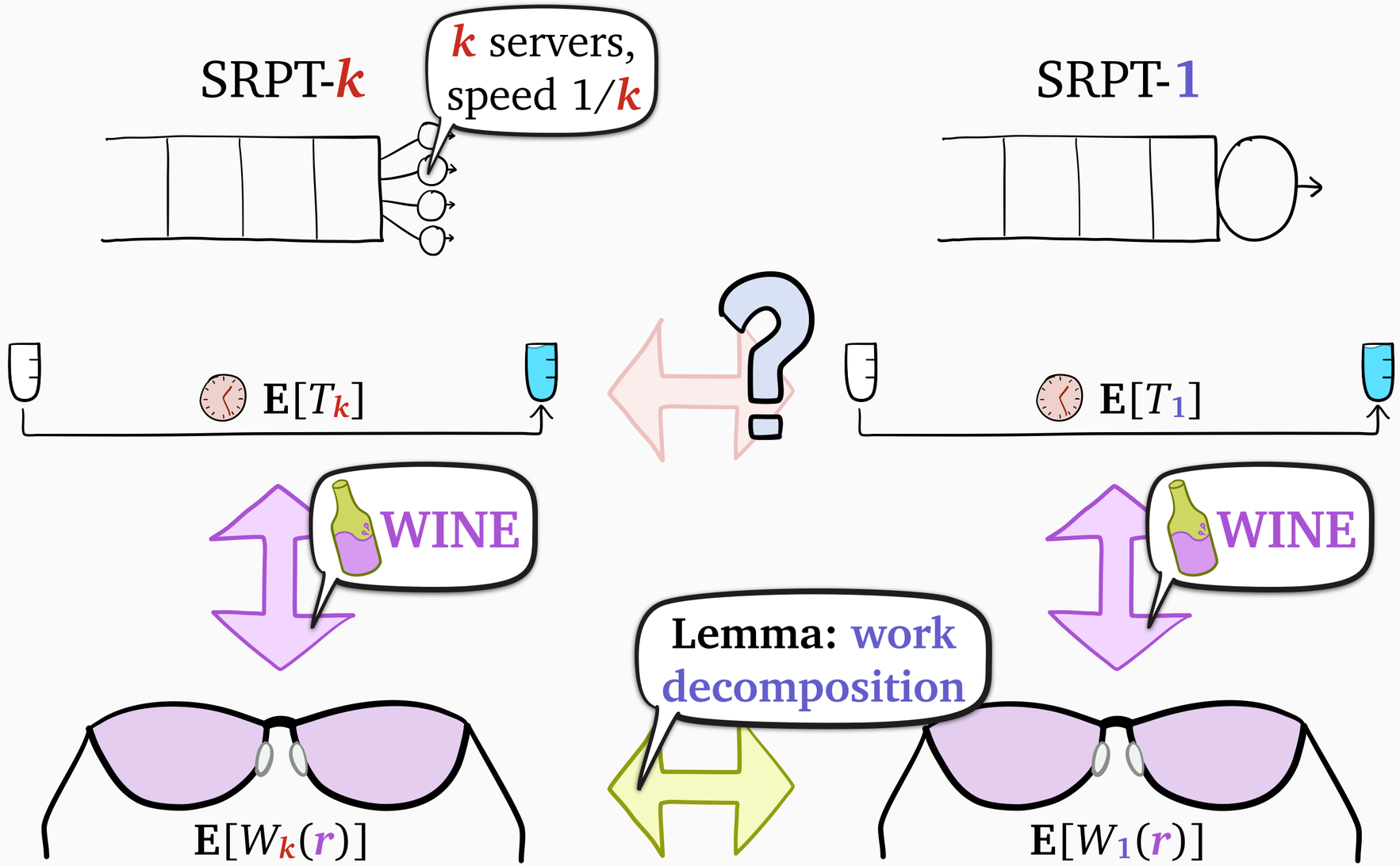
Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.

SRPT in the M/G/k



Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.

SRPT in the M/G/k



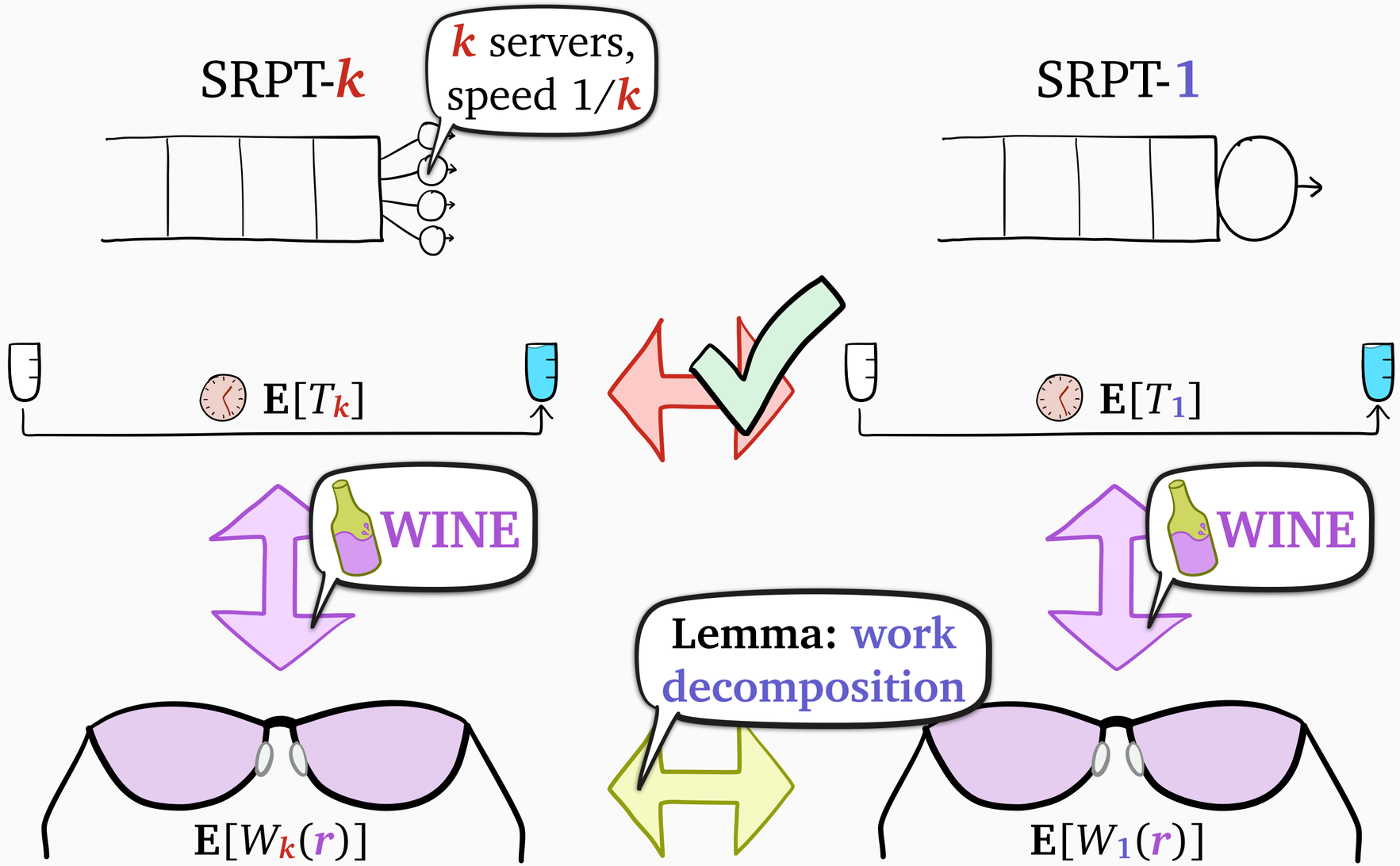
Theorem:

$$\mathbf{E}[S(\log S)^+] < \infty$$

$$\Downarrow$$

$$\lim_{\rho \to 1} \frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{SRPT-1}}]} = 1$$

SRPT in the M/G/k



Theorem:

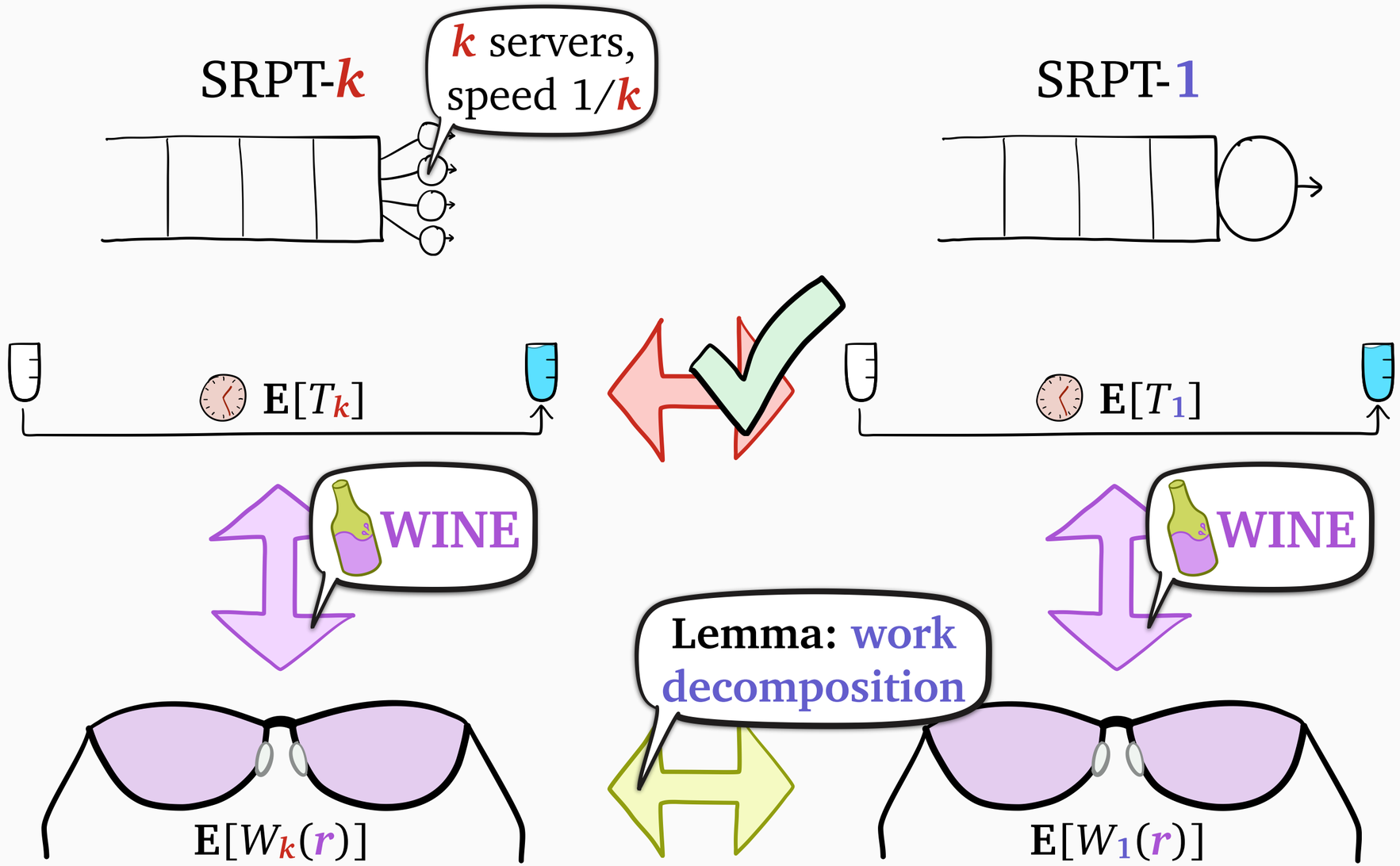
$$\mathbf{E}[S(\log S)^+] < \infty$$

$$\Downarrow$$

$$\lim_{\rho \to 1} \frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{SRPT-1}}]} = 1$$

Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.

SRPT in the M/G/k



Gittins, too

Theorem:

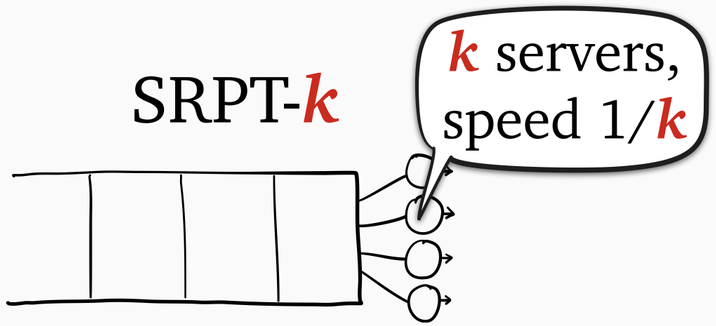
$$\mathbf{E}[S(\log S)^+] < \infty$$

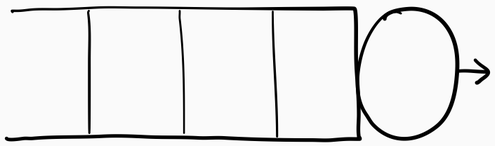
$$\Downarrow$$

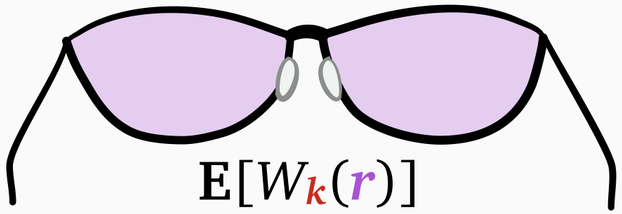
$$\lim_{\rho \to 1} \frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{SRPT-1}}]} = 1$$

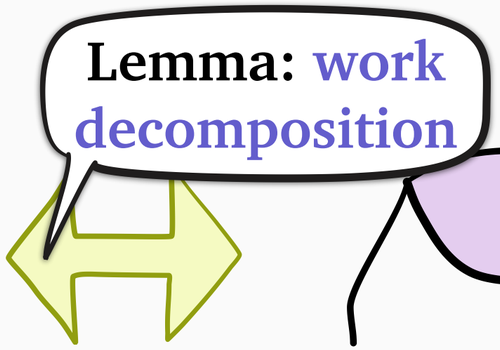
Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.

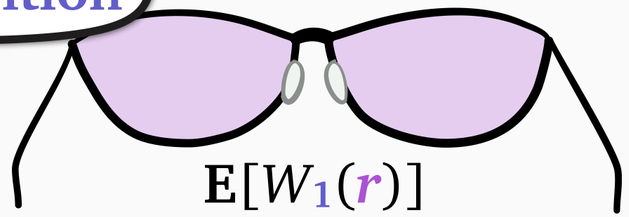
SRPT in the M/G/k











Gittins, too

Theorem:

$$\mathbf{E}[S(\log S)^+] < \infty$$

$$\Downarrow$$

$$\lim_{\rho \to 1} \frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{SRPT-1}}]} = 1$$

Open: general holding costs

Scully, Grosof, & Harchol-Balter (2020). doi:10.1145/3428328.

Beyond the M/G/k

Beyond the M/G/k

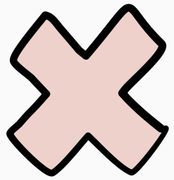
Dispatching with SRPT

Beyond the M/G/k

Dispatching with SRPT

Beyond the M/G/k

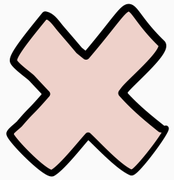
Dispatching with SRPT



Dispatching with **Gittins**

Beyond the M/G/k

Dispatching with SRPT

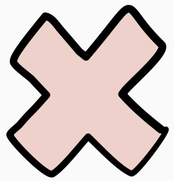


Dispatching with **Gittins**

Multiserver jobs and other service constraints

Beyond the M/G/*k*

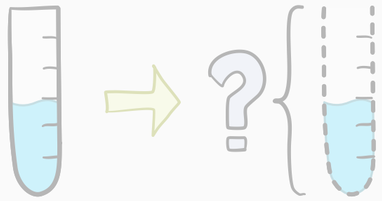
Dispatching with SRPT



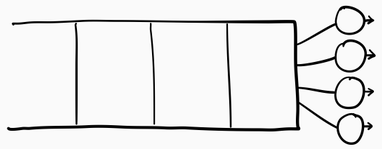
Dispatching with **Gittins**

Multiserver jobs and other service constraints

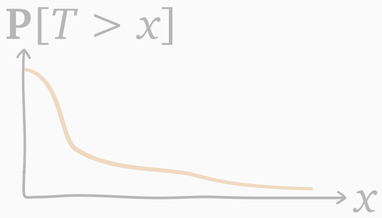
Hard scheduling questions



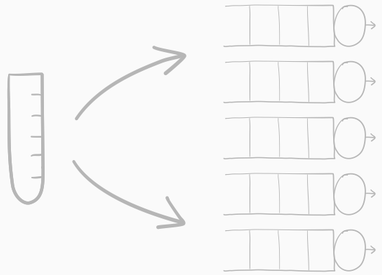
What if we have
unknown job sizes?



What if there are
multiple servers?

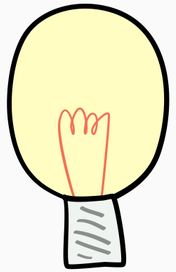


What if we want to optimize
tails instead of means?



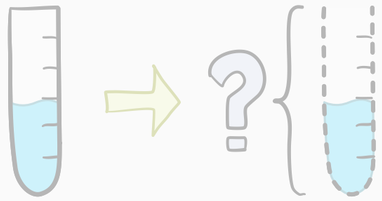
What if we can only use
dispatching without fancy scheduling?

What are the underlying
theoretical tools?

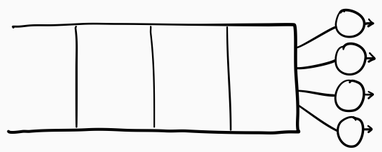


What are the resulting
practical lessons?

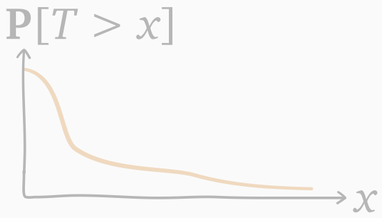
Hard scheduling questions



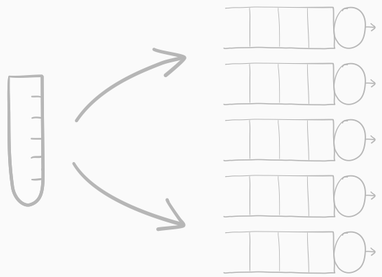
What if we have
unknown job sizes?



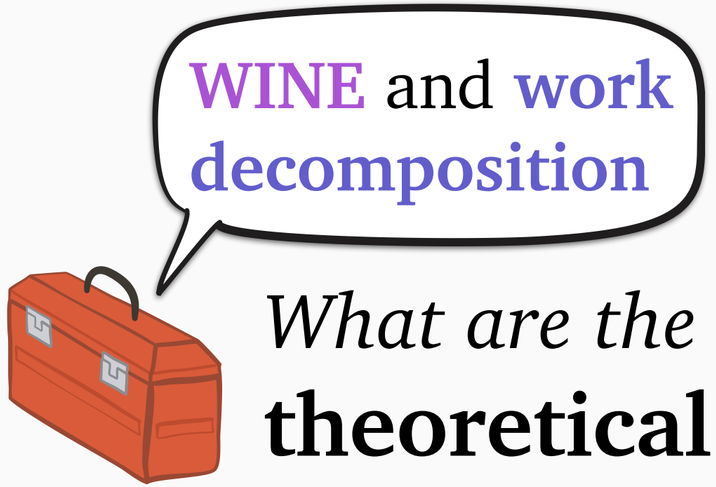
What if there are
multiple servers?



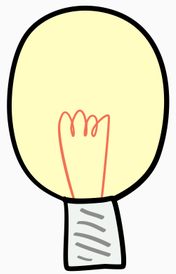
What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

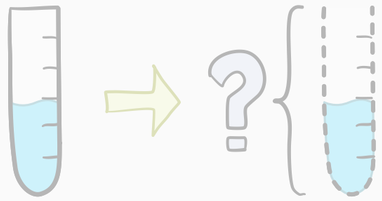


What are the underlying
theoretical tools?

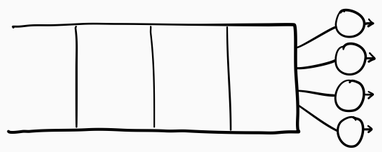


What are the resulting
practical lessons?

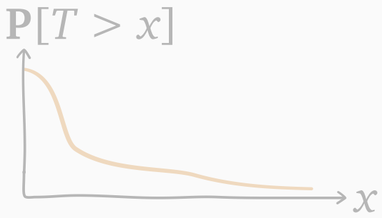
Hard scheduling questions



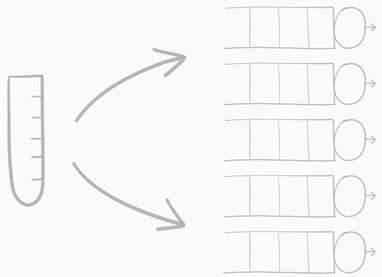
What if we have
unknown job sizes?



What if there are
multiple servers?



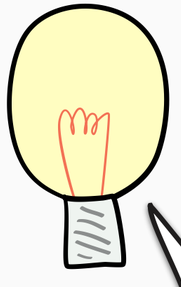
What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

WINE and **work decomposition**

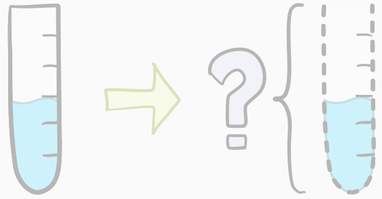
What are the underlying
theoretical tools?



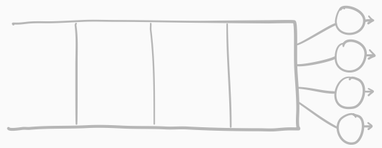
What are the resulting
practical lessons?

- Prefer central queueing
- When dispatching, spread out the small jobs

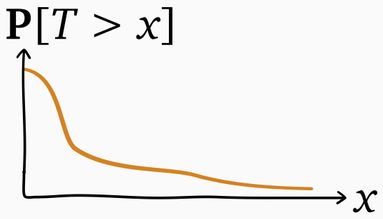
Hard scheduling questions



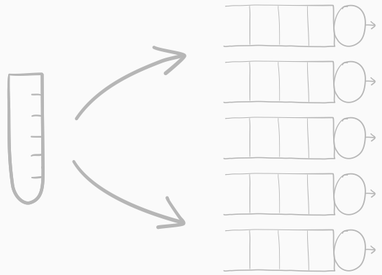
What if we have
unknown job sizes?



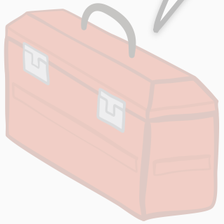
What if there are
multiple servers?



What if we want to optimize
tails instead of means?

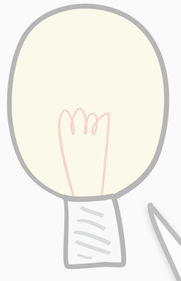


What if we can only use
dispatching without fancy scheduling?



WINE and **work decomposition**

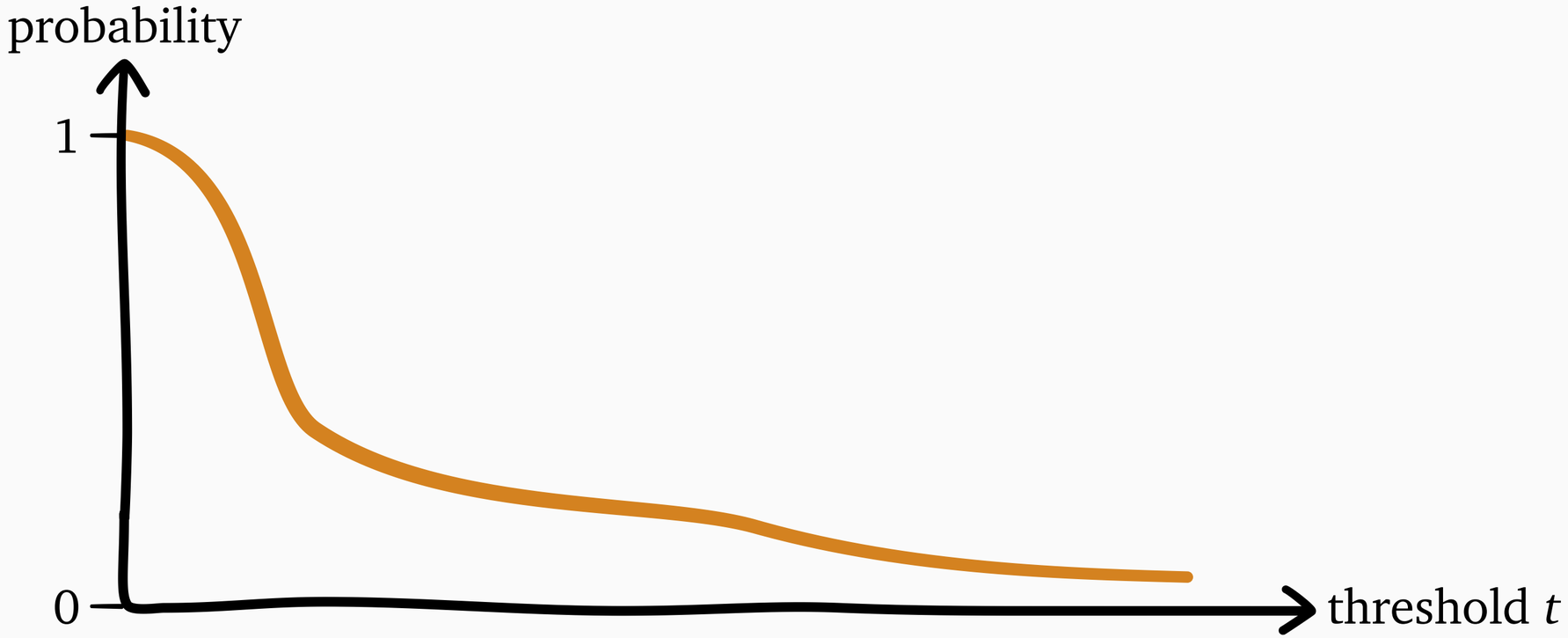
What are the underlying
theoretical tools?



What are the resulting
practical lessons?

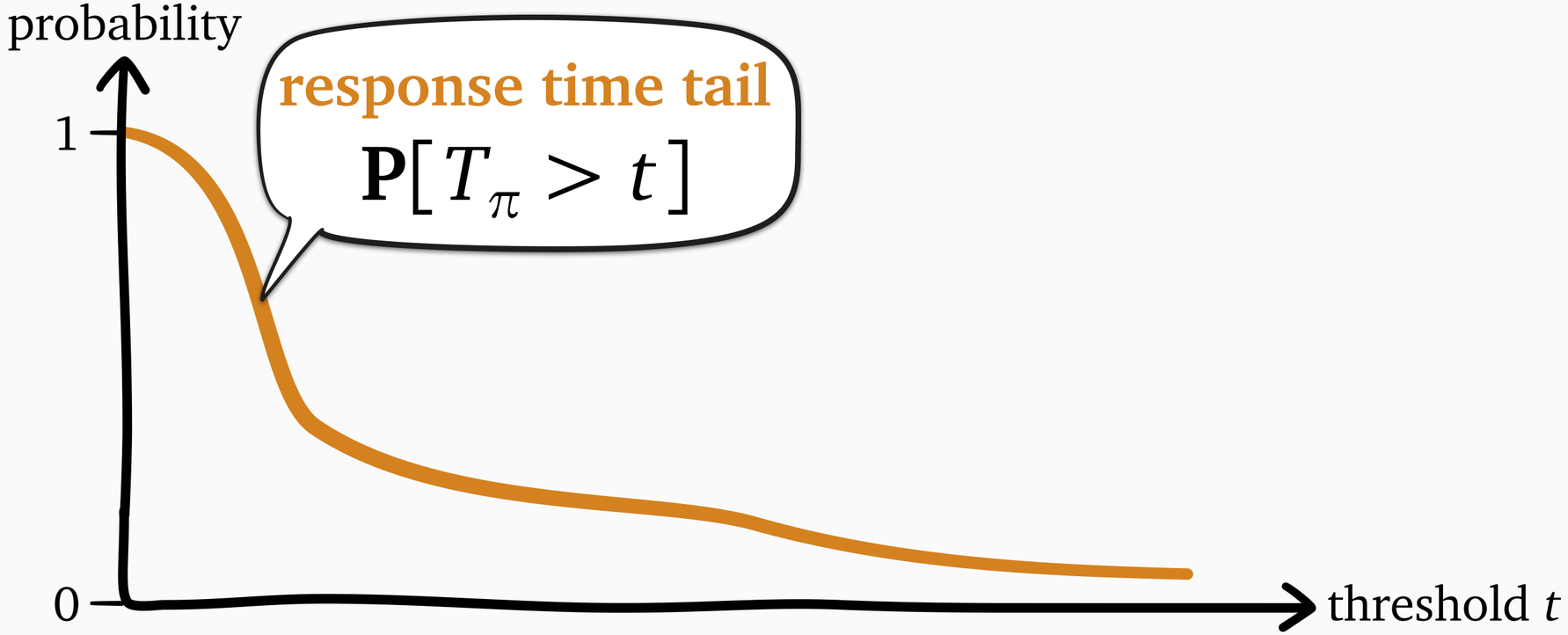
- Prefer central queueing
- When dispatching, spread out the small jobs

Asymptotic response time tail



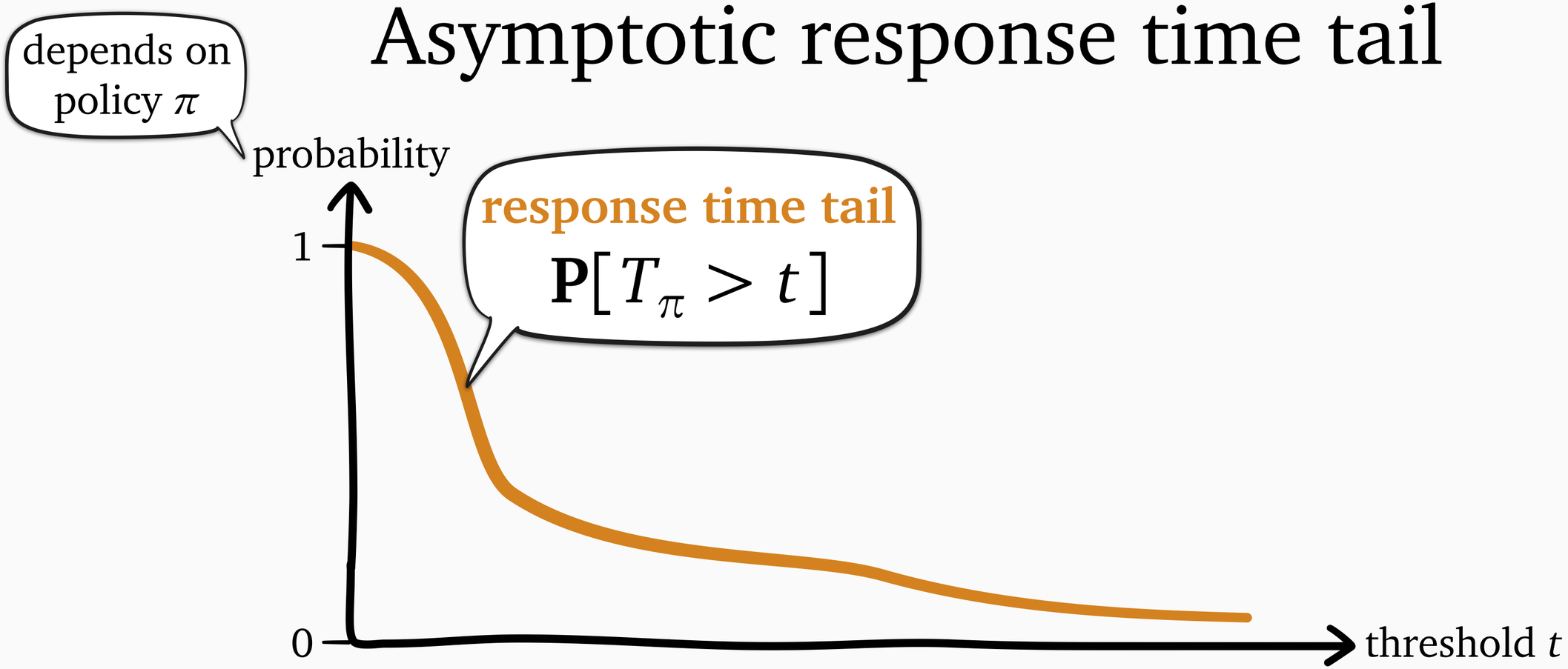
Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

Asymptotic response time tail



Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

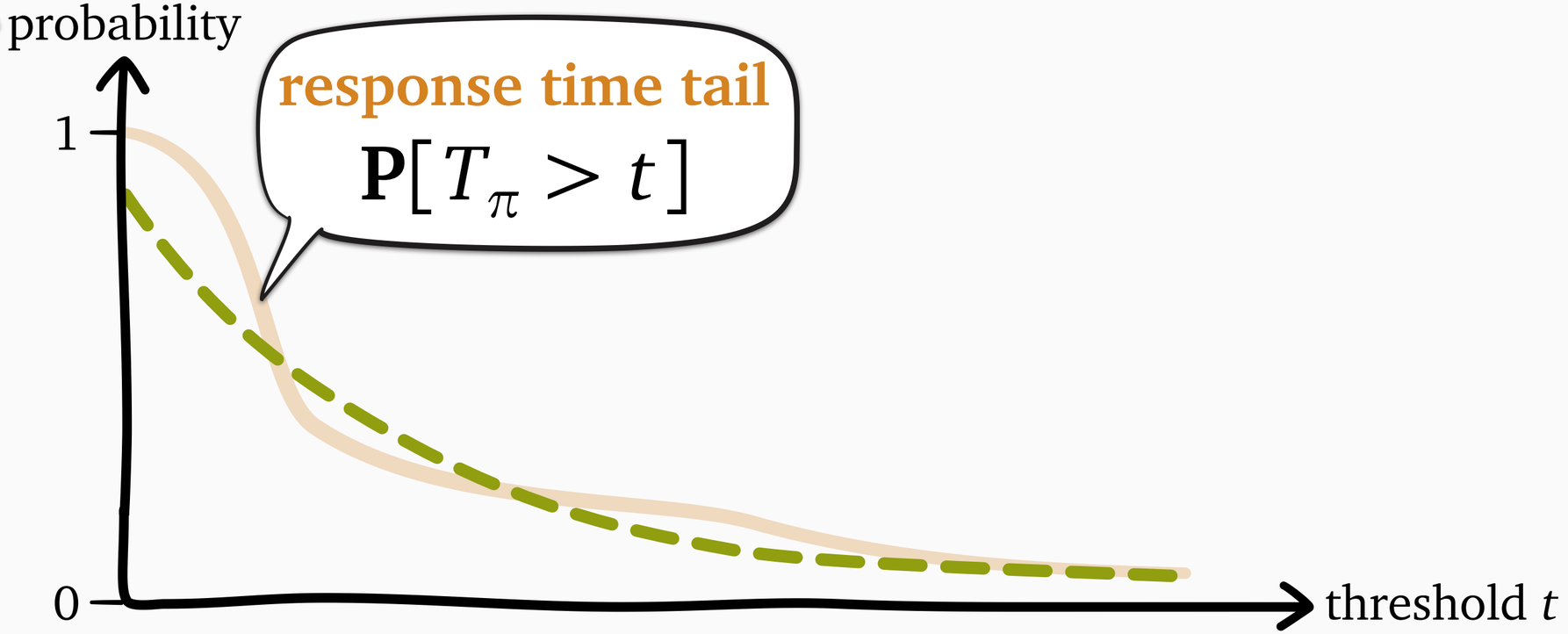
Asymptotic response time tail



Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

Asymptotic response time tail

depends on policy π

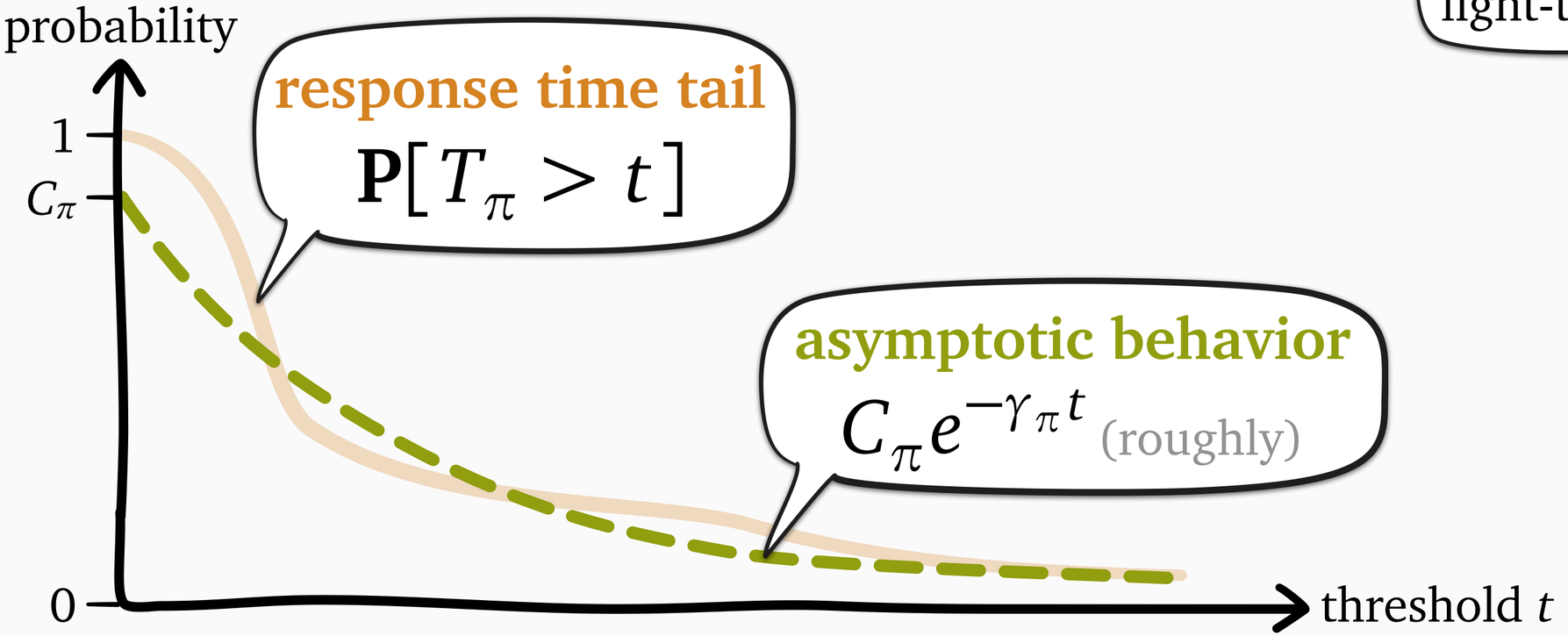


Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

Asymptotic response time tail

depends on policy π

when S is light-tailed

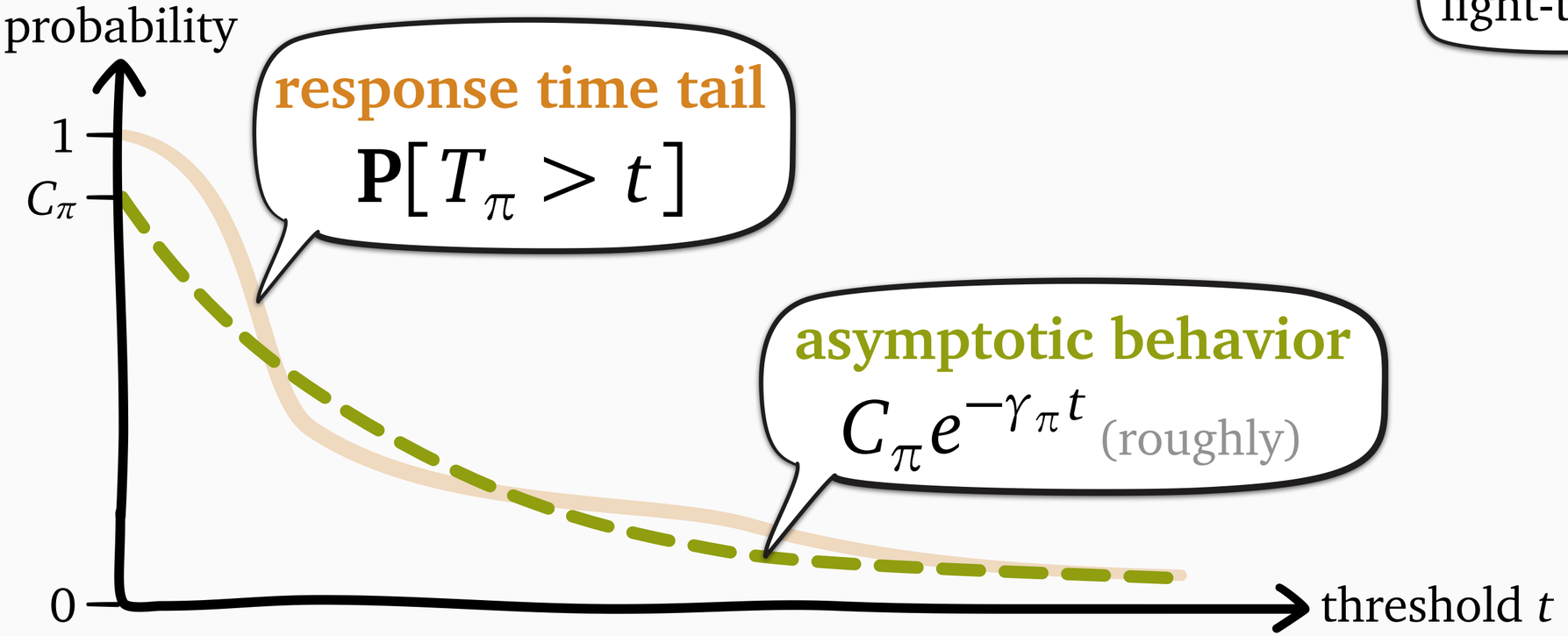


Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

Asymptotic response time tail

depends on policy π

when S is light-tailed



γ_π = *decay rate* of π

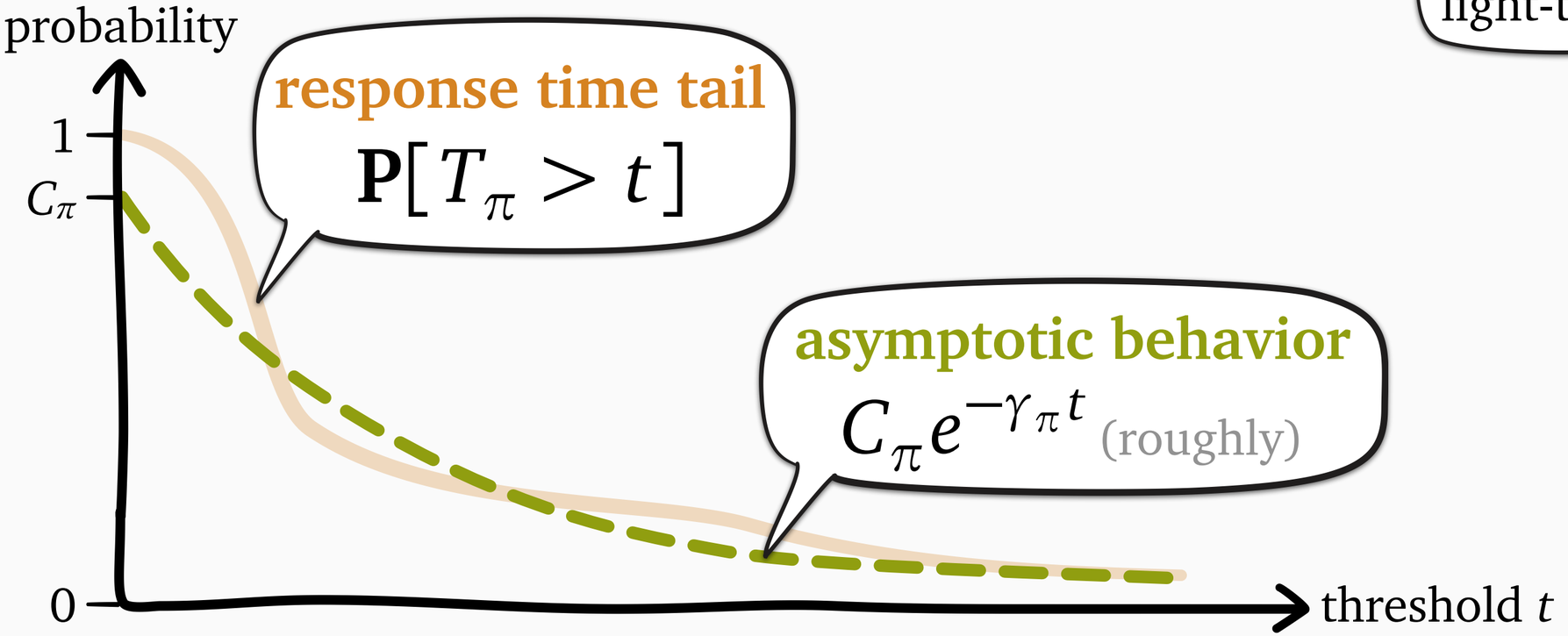
C_π = *tail constant* of π

Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

Asymptotic response time tail

depends on policy π

when S is light-tailed



Weak optimality: optimal γ_π

γ_π = *decay rate* of π

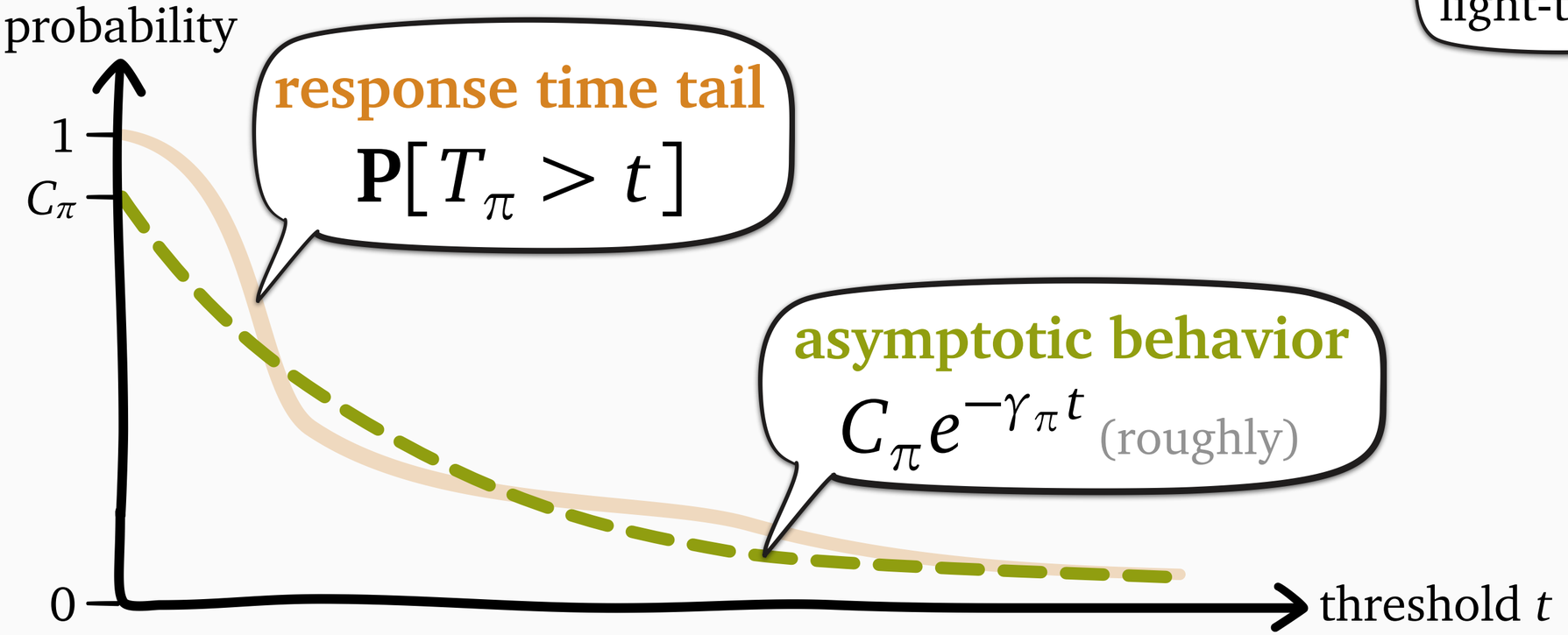
C_π = *tail constant* of π

Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

Asymptotic response time tail

depends on policy π

when S is light-tailed



γ_π = *decay rate* of π

C_π = *tail constant* of π

Weak optimality: optimal γ_π

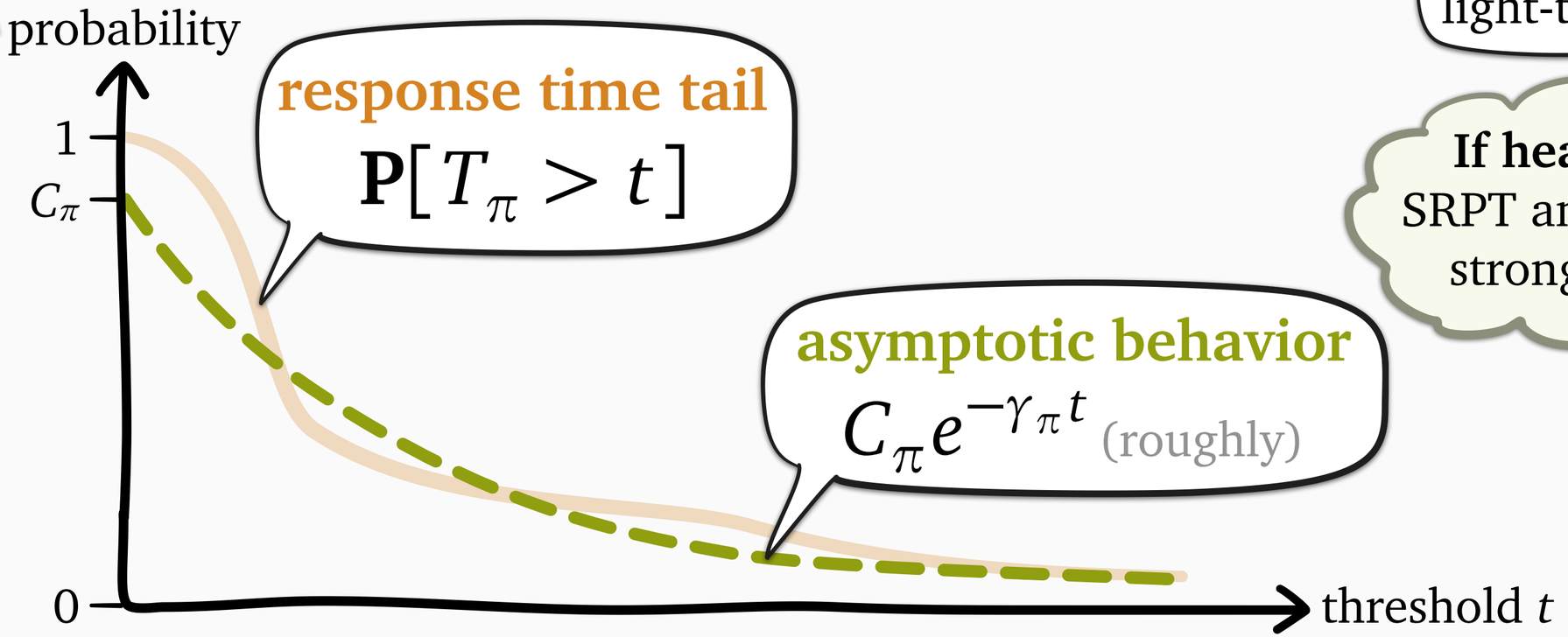
Strong optimality: optimal γ_π and C_π

Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

Asymptotic response time tail

depends on policy π

when S is light-tailed



If heavy-tailed:
SRPT and Gittins are strongly optimal

Weak optimality:
optimal γ_π

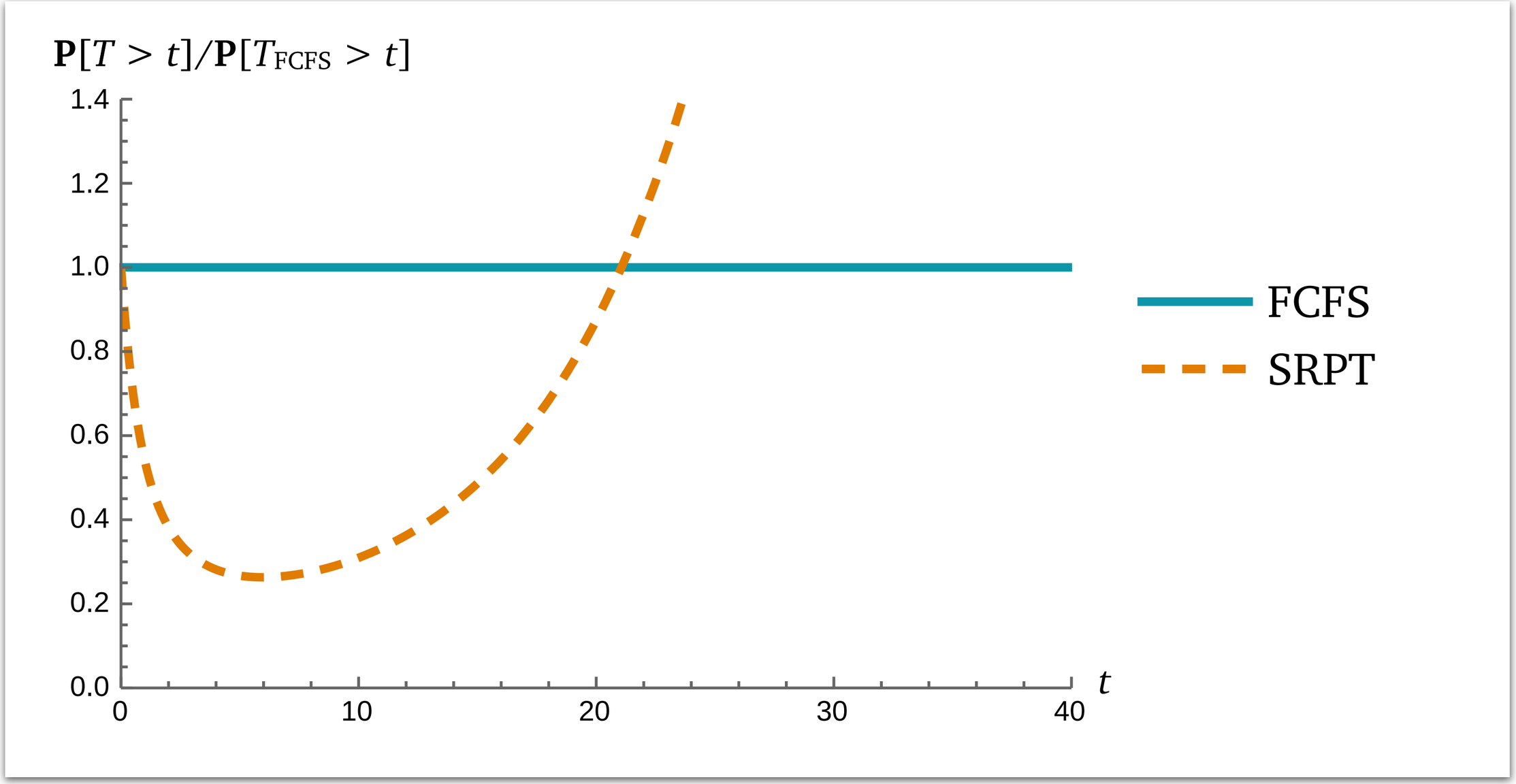
$\gamma_\pi = decay\ rate$ of π

$C_\pi = tail\ constant$ of π

Strong optimality:
optimal γ_π and C_π

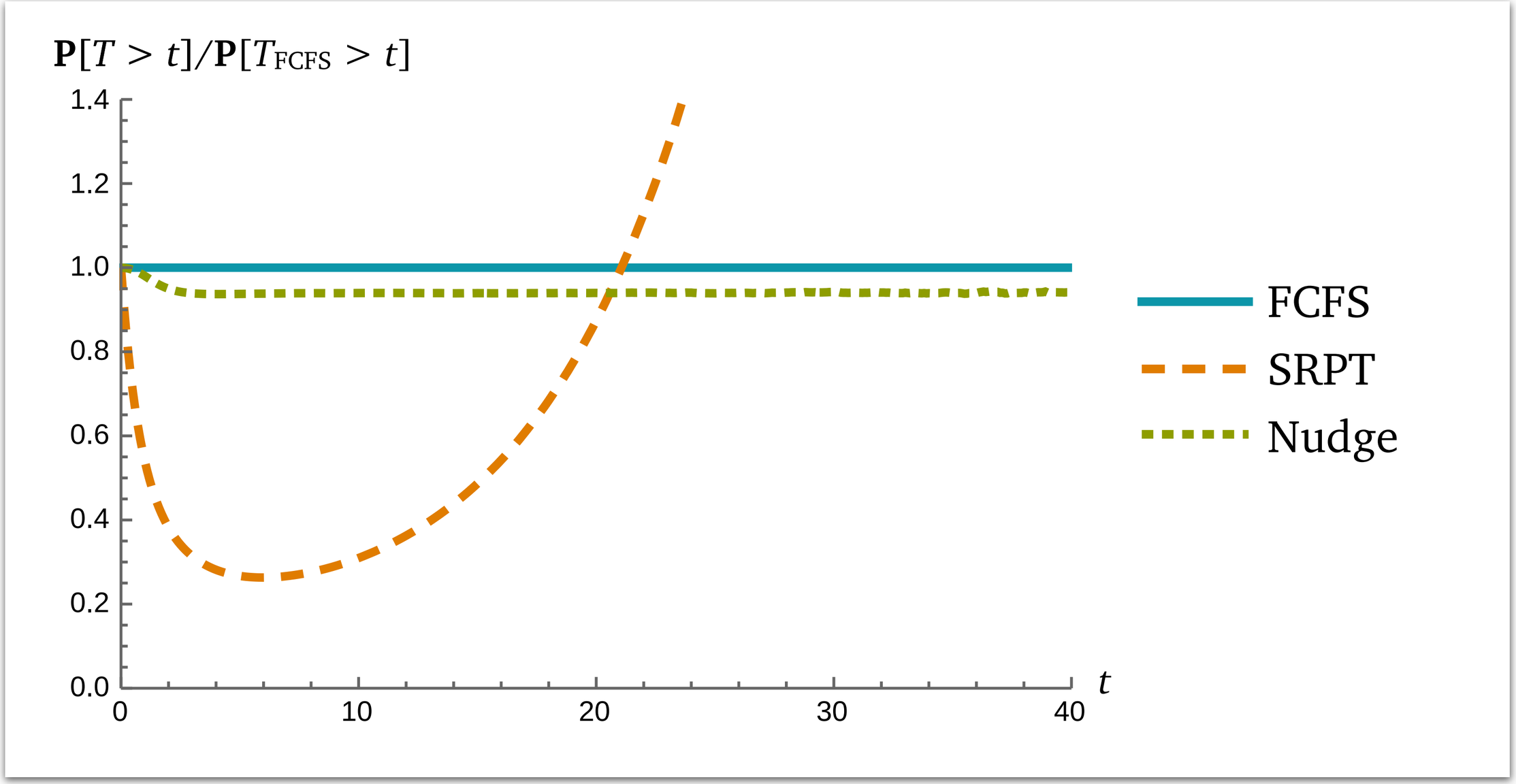
Boxma & Zwart (2007). doi:10.1145/1243401.1243406.
Wierman & Zwart (2012). doi:10.1287/opre.1120.1086.

Optimizing the tail constant C



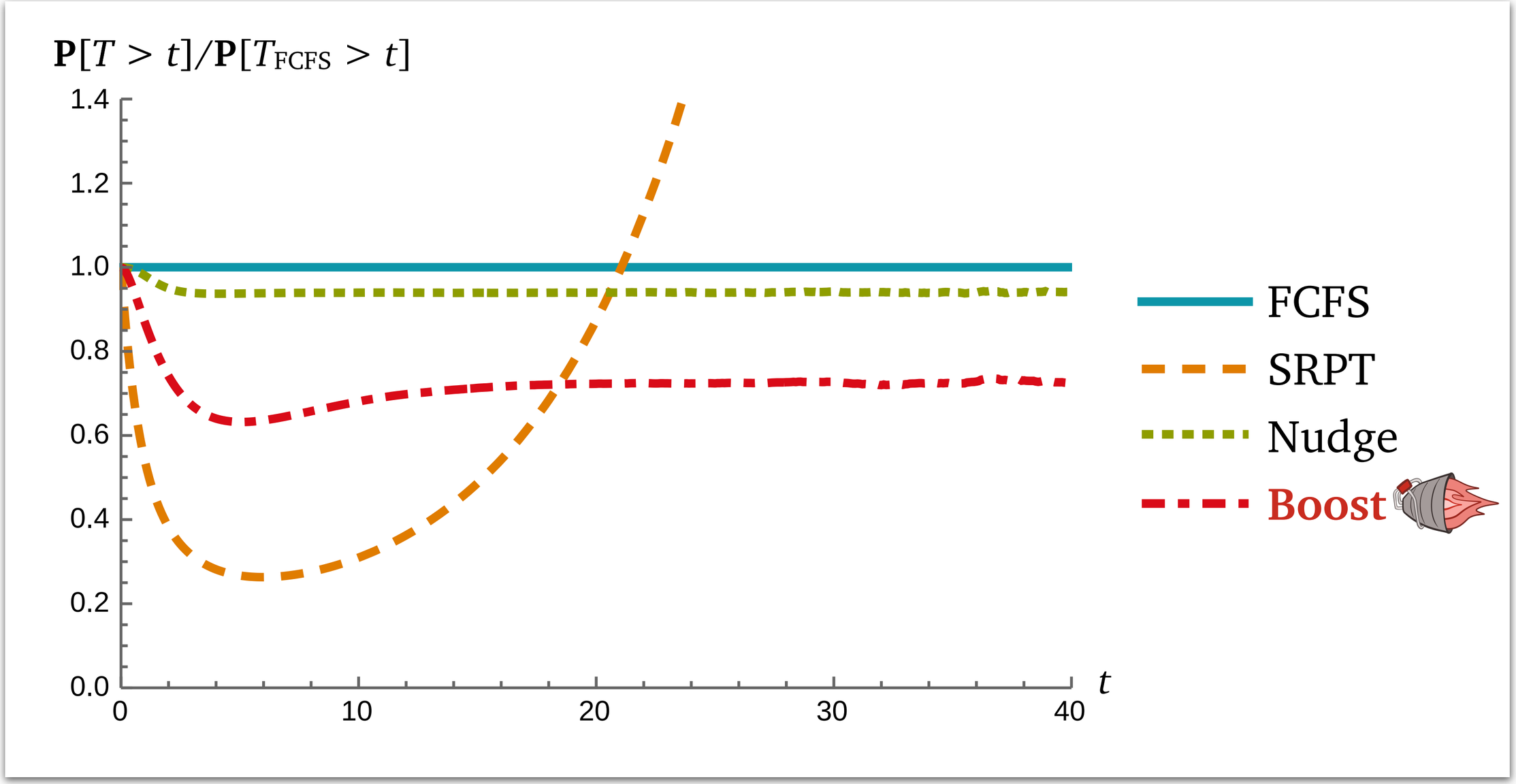
Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Optimizing the tail constant C



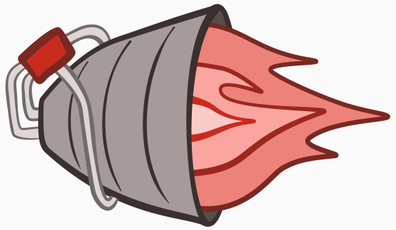
Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Optimizing the tail constant C



Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy

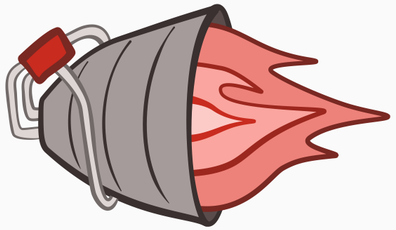


Scheduling rule: always serve job of *minimum **boosted** arrival time*

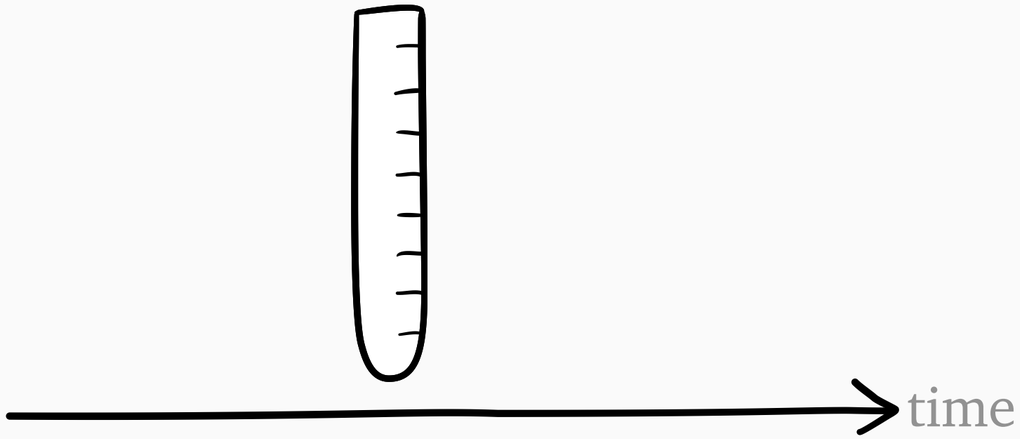
boosted arrival time
= arrival time − **boost**(size s)

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy



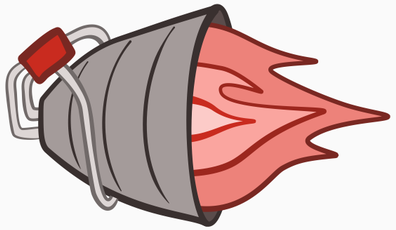
Scheduling rule: always serve job of *minimum **boosted** arrival time*



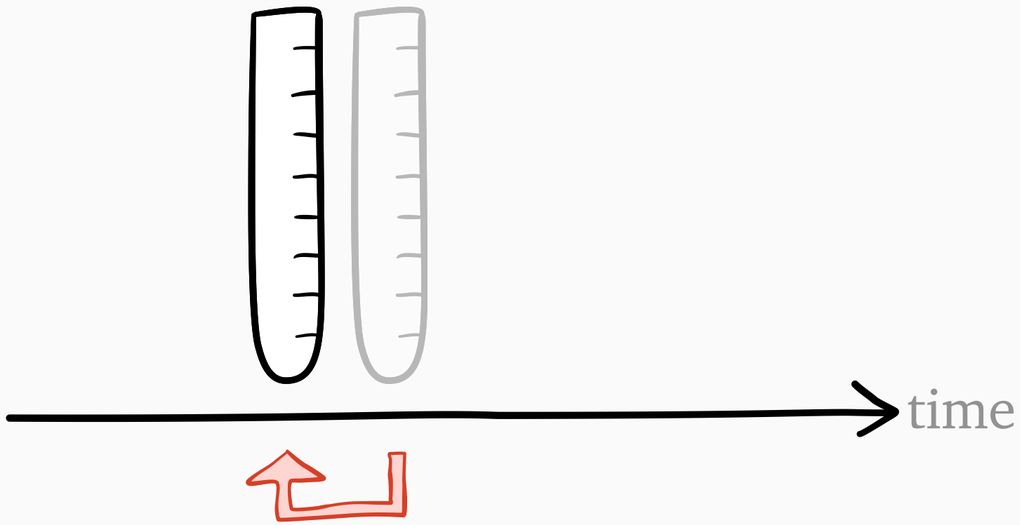
boosted arrival time
= arrival time − **boost**(size s)

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy



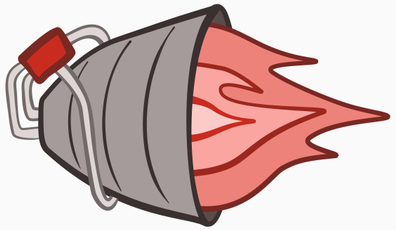
Scheduling rule: always serve job of
*minimum **boosted** arrival time*



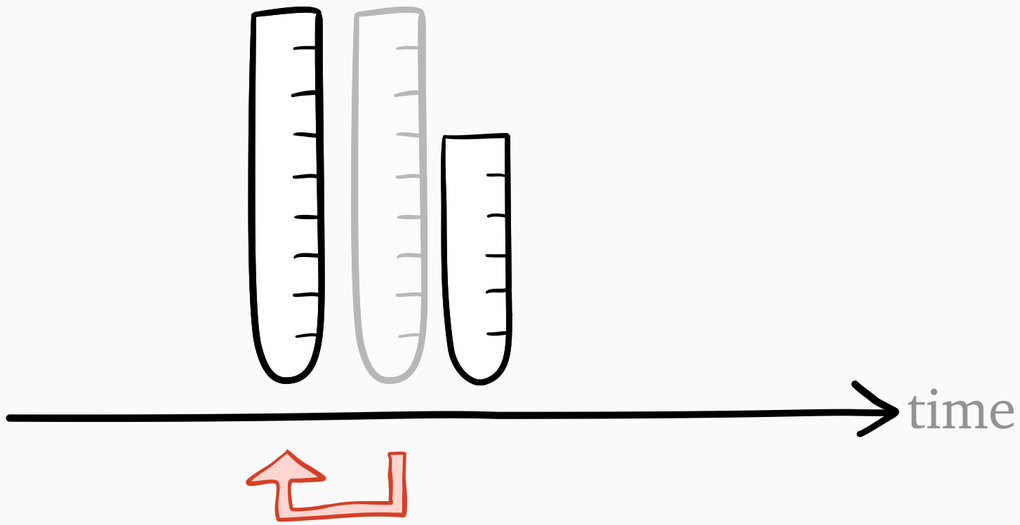
boosted arrival time
= arrival time − **boost**(size s)

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy



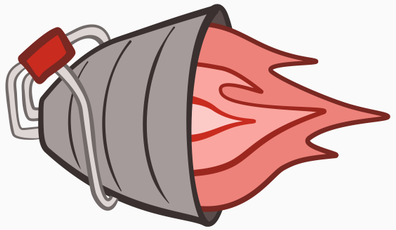
Scheduling rule: always serve job of *minimum **boosted** arrival time*



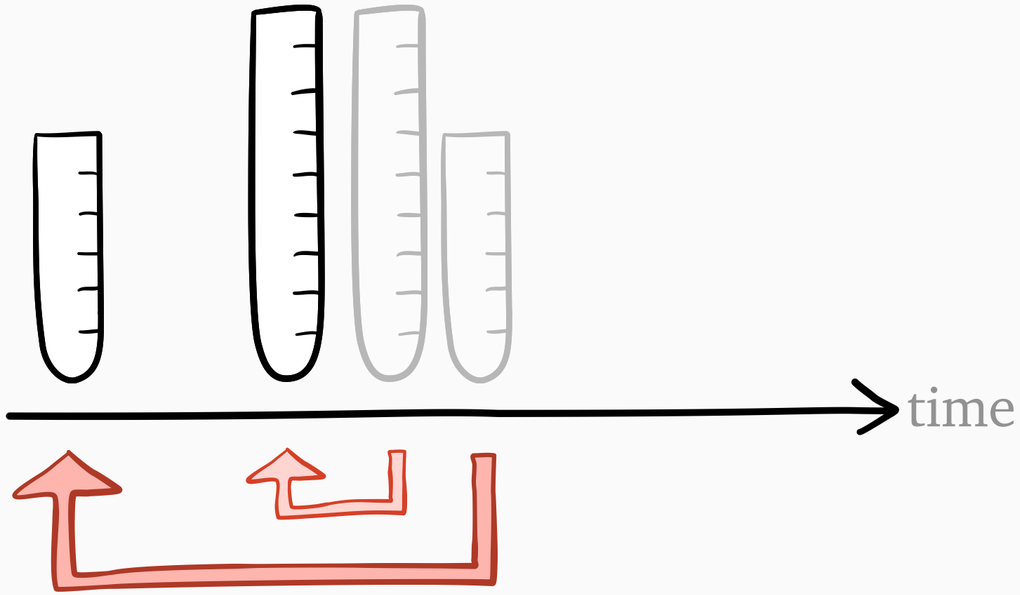
boosted arrival time
= arrival time − **boost**(size s)

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy



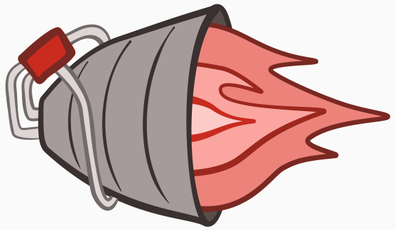
Scheduling rule: always serve job of *minimum **boosted** arrival time*



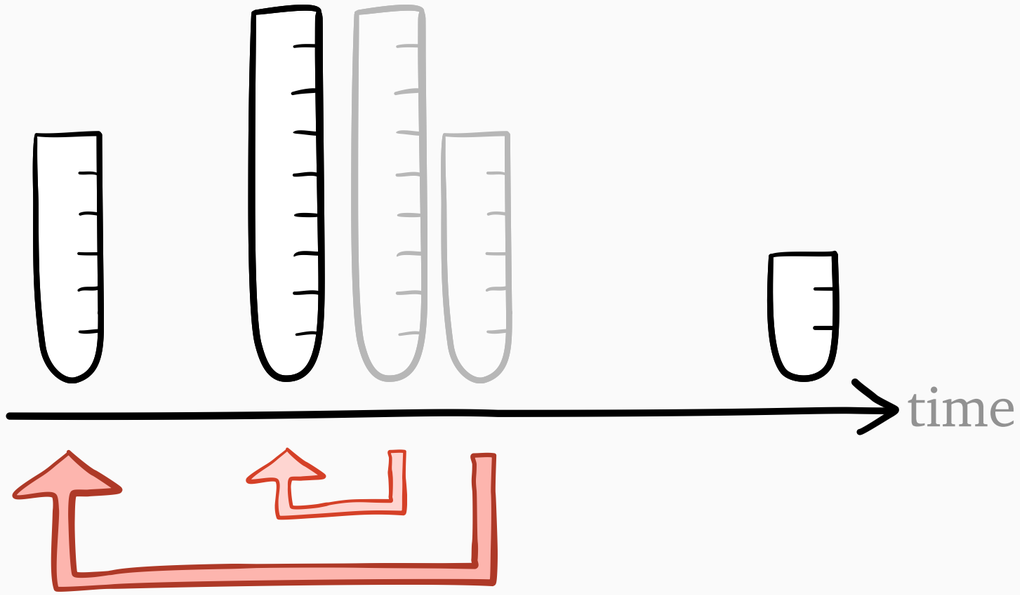
boosted arrival time
= arrival time − **boost**(size s)

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy



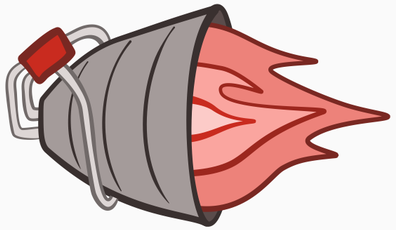
Scheduling rule: always serve job of *minimum **boosted** arrival time*



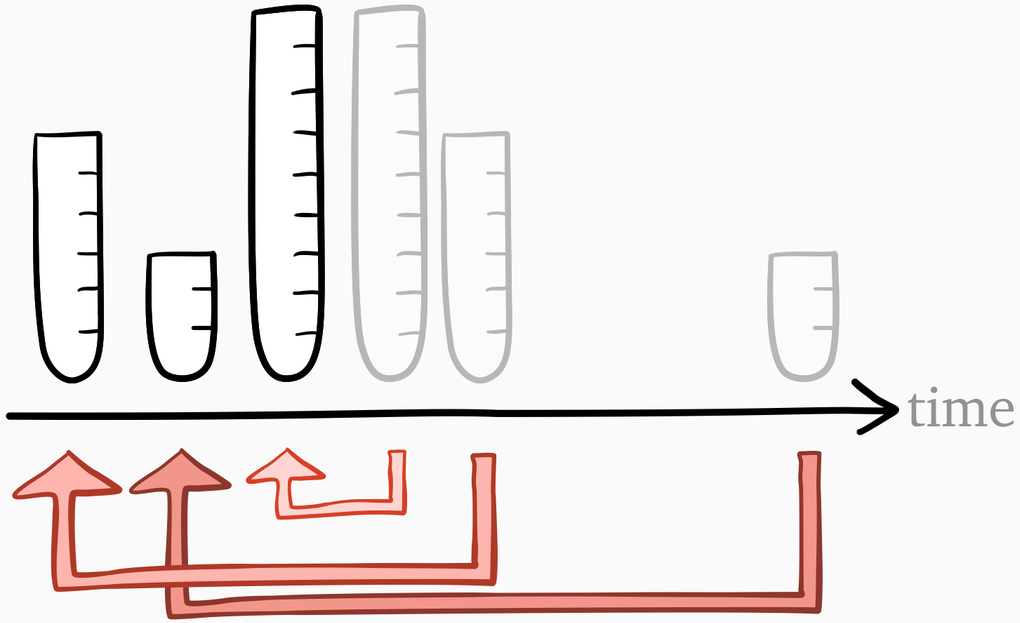
boosted arrival time
= arrival time − **boost**(size s)

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy



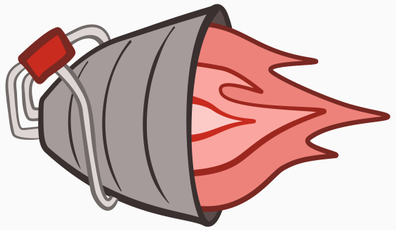
Scheduling rule: always serve job of
*minimum **boosted** arrival time*



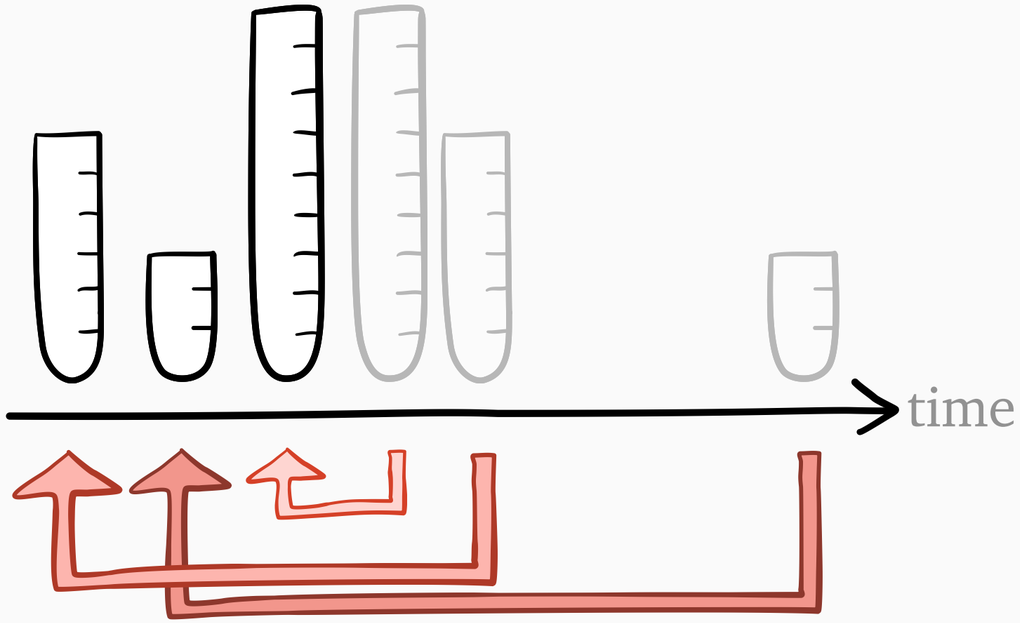
boosted arrival time
= arrival time − **boost**(size s)

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy



Scheduling rule: always serve job of
*minimum **boosted** arrival time*

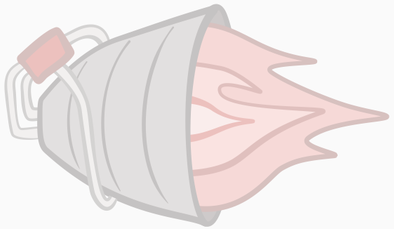


boosted arrival time
= arrival time − **boost**(size s)

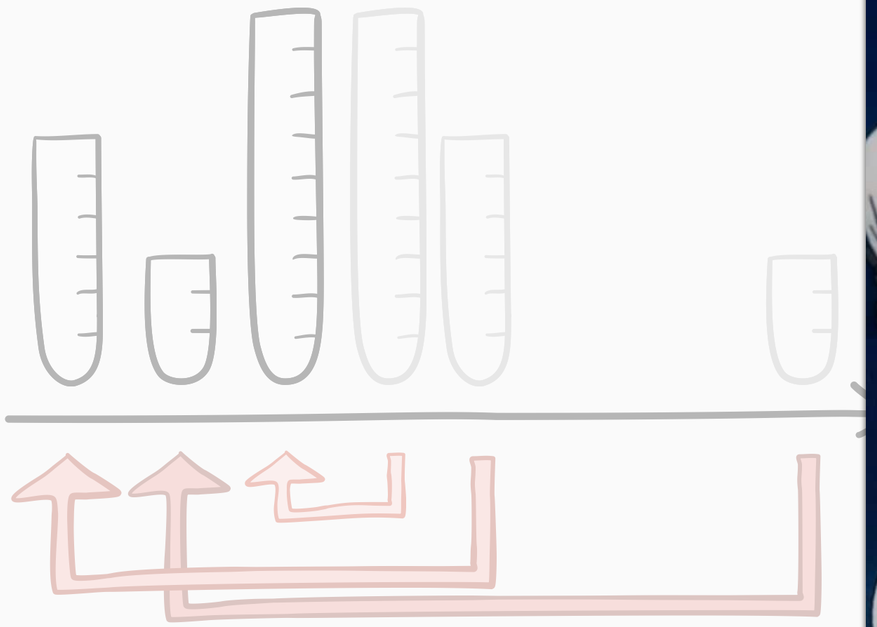
$$\frac{1}{\gamma}\log\frac{1}{1-e^{-\gamma s}}$$

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Boost policy



Scheduling rule: always serve job of
*minimum **boosted** arrival time*



rrival time
l time − **boost**(size s)

$$\frac{1}{\gamma}\log\frac{1}{1-e^{-\gamma s}}$$

Yu & Scully (2024). doi:10.1145/3656011.
Harlev, Yu, & Scully (2025). doi:10.1145/3727109.

Tail optimization questions

Tail optimization questions

From M/G/1 to M/G/k

Tail optimization questions

From M/G/1 to M/G/k

Minimizing tail of
slowdown $\mathbf{P}[T/S > x]$

Tail optimization questions

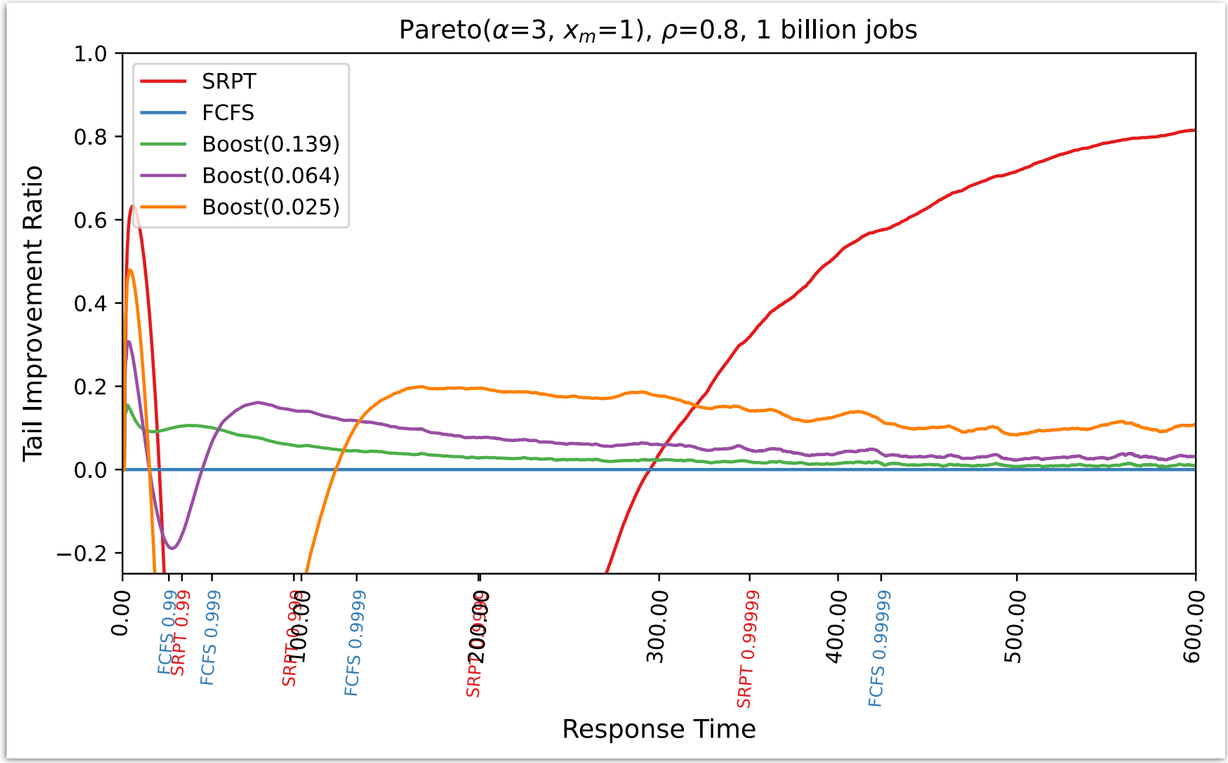
From M/G/1 to M/G/k

Minimizing tail of
slowdown $\mathbf{P}[T/S > x]$

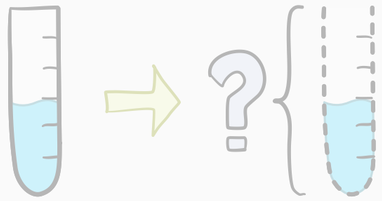
Revisiting “slightly”
heavy-tailed case ($\alpha = 4$)

Tail optimization questions

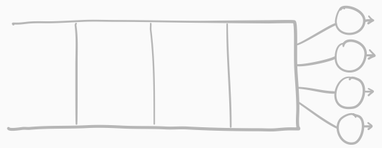
- From M/G/1 to M/G/k
- Minimizing tail of *slowdown* $\mathbf{P}[T/S > x]$
- Revisiting "slightly" heavy-tailed case ($\alpha = 4$)



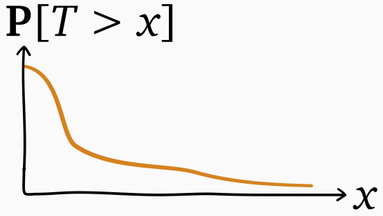
Hard scheduling questions



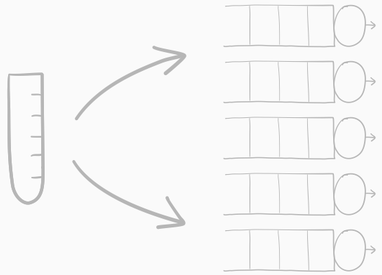
What if we have
unknown job sizes?



What if there are
multiple servers?

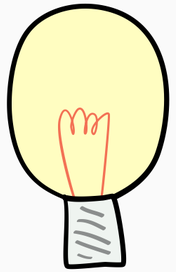


What if we want to optimize
tails instead of means?



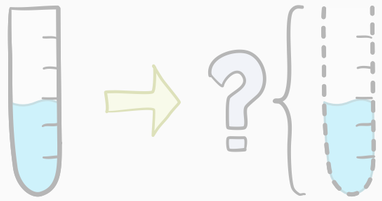
What if we can only use
dispatching without fancy scheduling?

What are the underlying
theoretical tools?

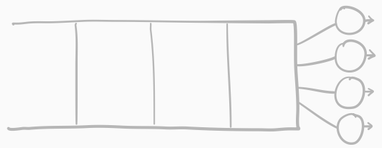


What are the resulting
practical lessons?

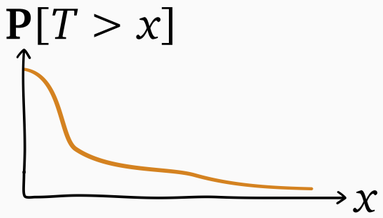
Hard scheduling questions



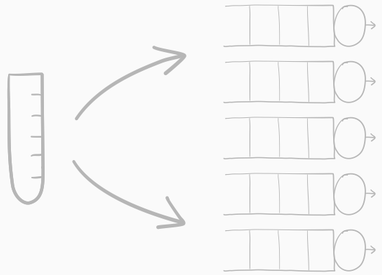
What if we have
unknown job sizes?



What if there are
multiple servers?



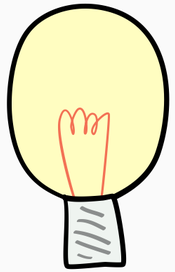
What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

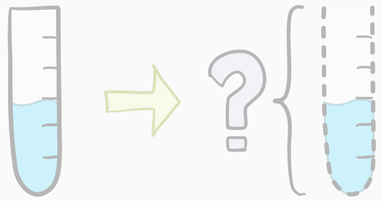
minimize C
⇳
minimize "$\mathbf{E}[e^{\gamma T}]$"

What are the underlying
theoretical tools?

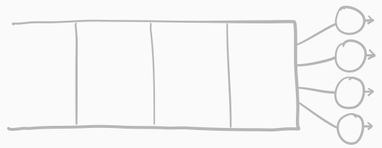


What are the resulting
practical lessons?

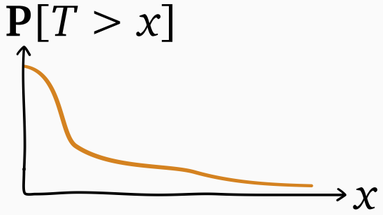
Hard scheduling questions



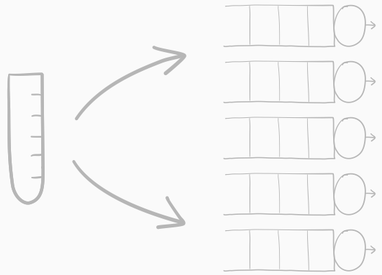
What if we have
unknown job sizes?



What if there are
multiple servers?



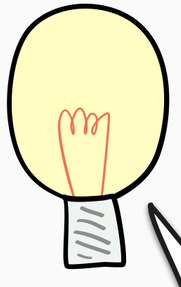
What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

minimize C
⇕
minimize "$\mathbf{E}[e^{\gamma T}]$"

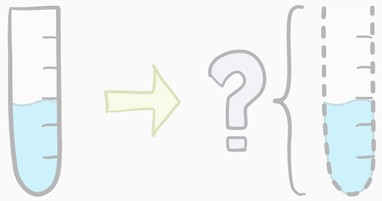
What are the underlying
theoretical tools?



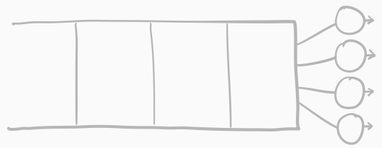
What are the resulting
practical lessons?

- Use arrival times and sizes
- Try **Boost** instead of FCFS!
- Try measuring $\mathbf{E}[e^{\gamma T}]$?

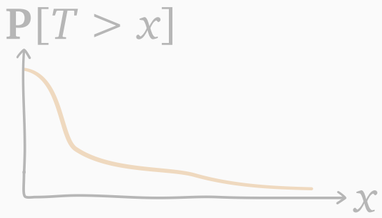
Hard scheduling questions



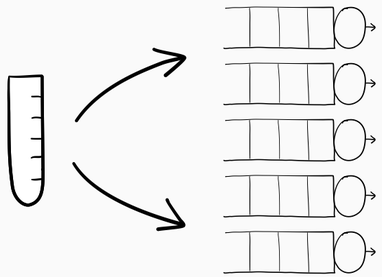
What if we have
unknown job sizes?



What if there are
multiple servers?



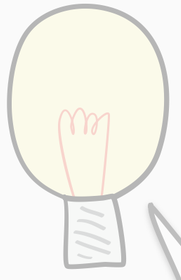
What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

minimize C
⇕
minimize "$\mathbf{E}[e^{\gamma T}]$"

What are the underlying
theoretical tools?

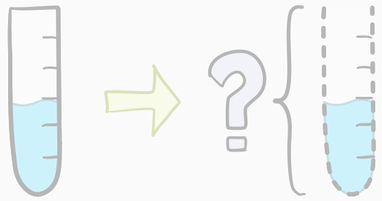


What are the resulting
practical lessons?

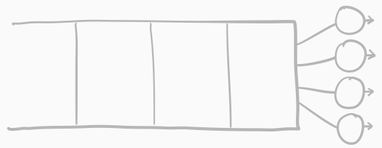
- Use arrival times and sizes
- Try **Boost** instead of FCFS!
- Try measuring $\mathbf{E}[e^{\gamma T}]$?

… stay tuned for Natalie’s talk!

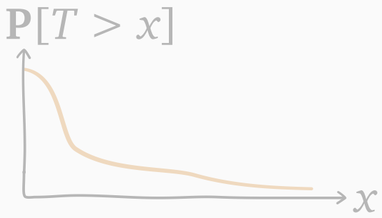
Hard scheduling questions



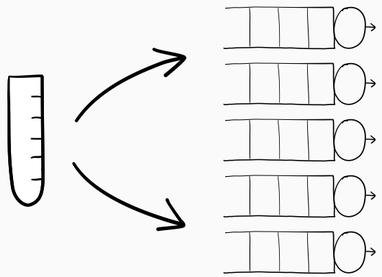
What if we have
unknown job sizes?



What if there are
multiple servers?

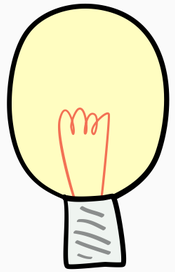


What if we want to optimize
tails instead of means?



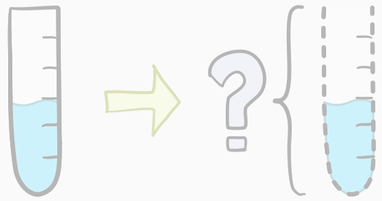
What if we can only use
dispatching without fancy scheduling?

What are the underlying
theoretical tools?

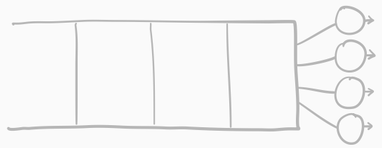


What are the resulting
practical lessons?

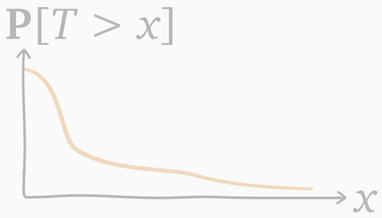
Hard scheduling questions



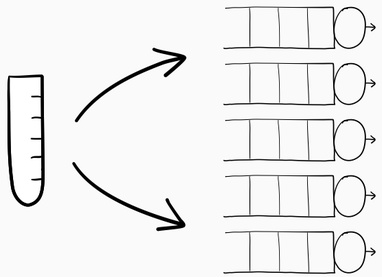
What if we have
unknown job sizes?



What if there are
multiple servers?

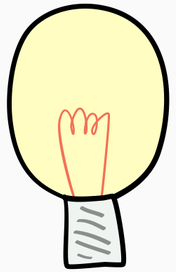


What if we want to optimize
tails instead of means?



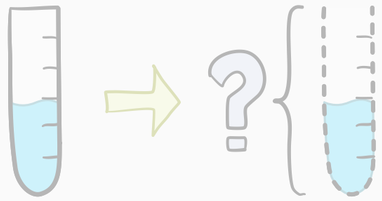
What if we can only use
dispatching without fancy scheduling?

What are the underlying
theoretical tools?

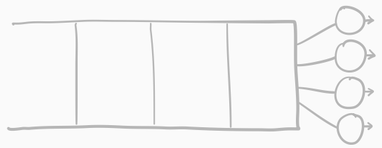


What are the resulting
practical lessons?

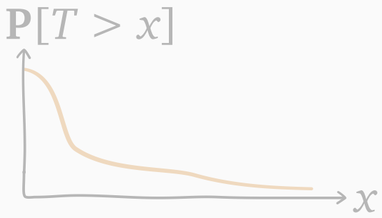
Hard scheduling questions



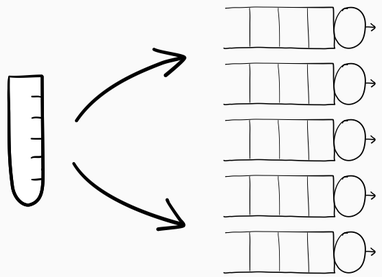
What if we have
unknown job sizes?



What if there are
multiple servers?



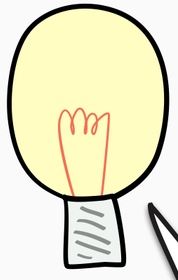
What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

newish *Continuous* **state space collapse**

What are the underlying
theoretical tools?



What are the resulting
practical lessons?

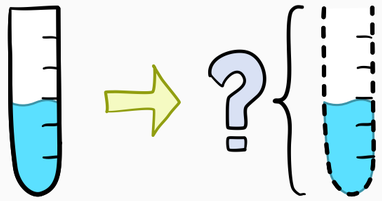
- Simulate priority classes
- Maintain work imbalance using closed-loop control

But wait, there's more!

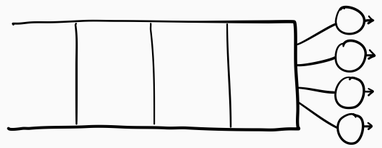
- G/G arrivals
- Generalized switch
- Parallelizable jobs and other complex service choices
- Multiserver jobs and other complex packing concerns
- Preemption overhead
- Large-system limits
- Stein's method

Hong & Scully (2024). doi:10.1016/j.peva.2023.102377.
Hurtado-Lange & Maguluri (2020). doi:10.1287/stsy.2019.0056.
Berg (2022). https://bsb20.github.io/bsberg_phd_csd_2022.pdf.
Grosof, Scully, Harchol-Balter, & Scheller-Wolf (2022). doi:10.1145/3570612.
Ramakrishna & Scully (2024). doi:10.1145/3695411.3695419.
Hong & Wang (2022). doi:10.1145/3492866.3549717.
Braverman (2023). doi:10.1287/stsy.2022.0102.

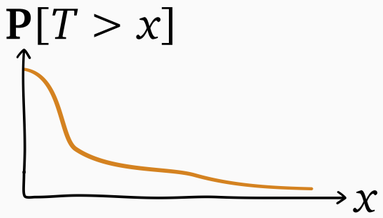
Hard scheduling questions



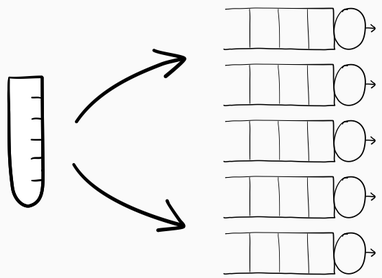
What if we have
unknown job sizes?



What if there are
multiple servers?

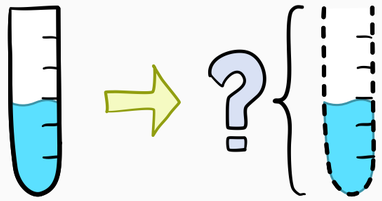


What if we want to optimize
tails instead of means?

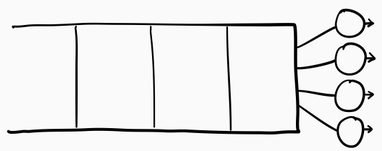


What if we can only use
dispatching without fancy scheduling?

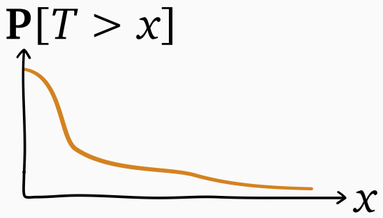
Hard scheduling questions



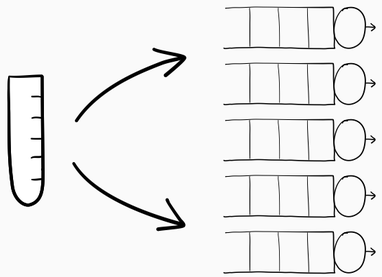
What if we have
unknown job sizes?



What if there are
multiple servers?



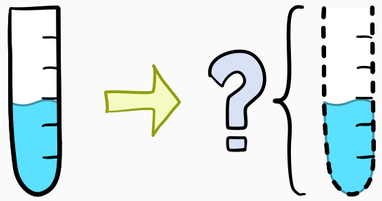
What if we want to optimize
tails instead of means?



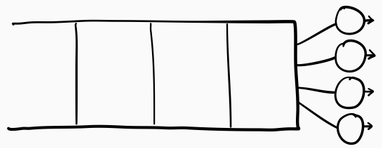
What if we can only use
dispatching without fancy scheduling?

Theoretical tools

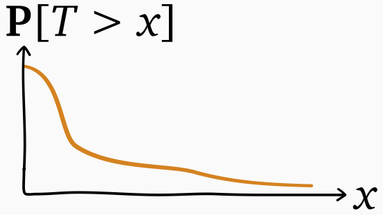
Hard scheduling questions



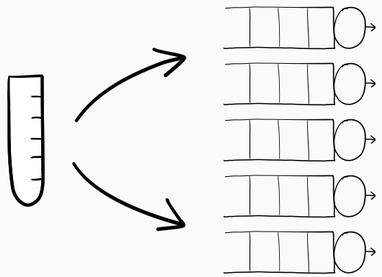
What if we have
unknown job sizes?



What if there are
multiple servers?



What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

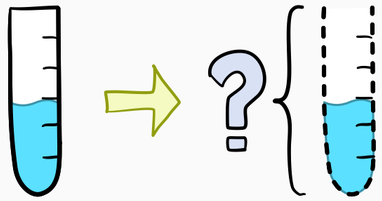
Work decomposition

State space collapse

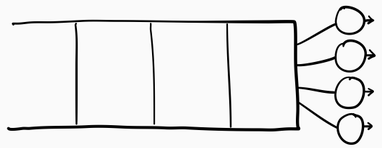
Boost 

Theoretical tools

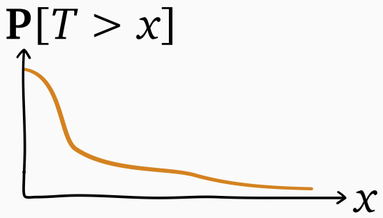
Hard scheduling questions



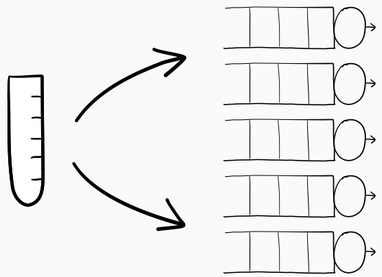
What if we have
unknown job sizes?



What if there are
multiple servers?



What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

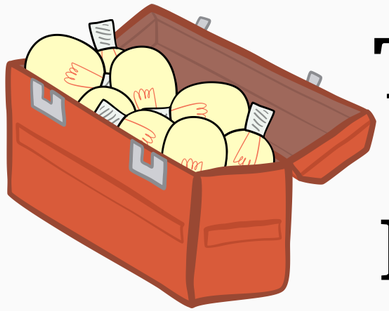

WINE

Work decomposition

State space collapse


SOAP

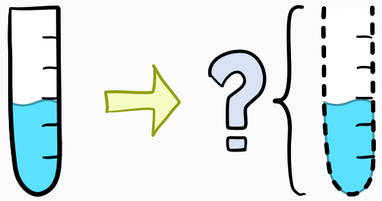
Boost 



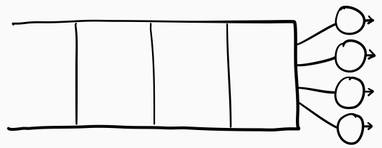
Theoretical tools

Practical lessons

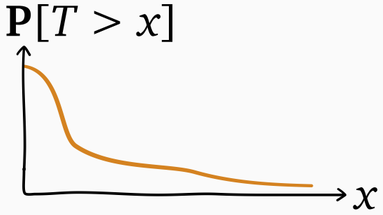
Hard scheduling questions



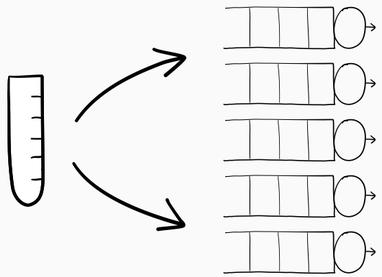
What if we have
unknown job sizes?



What if there are
multiple servers?



What if we want to optimize
tails instead of means?



What if we can only use
dispatching without fancy scheduling?

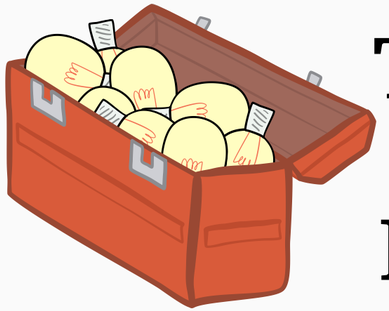

WINE

Work decomposition

State space collapse


SOAP

Boost 



Theoretical tools

Practical lessons

Empirical Gittins works

PSJF is robust

k servers ≈ 1 server

Use size and arrival time

Closed-loop dispatching control