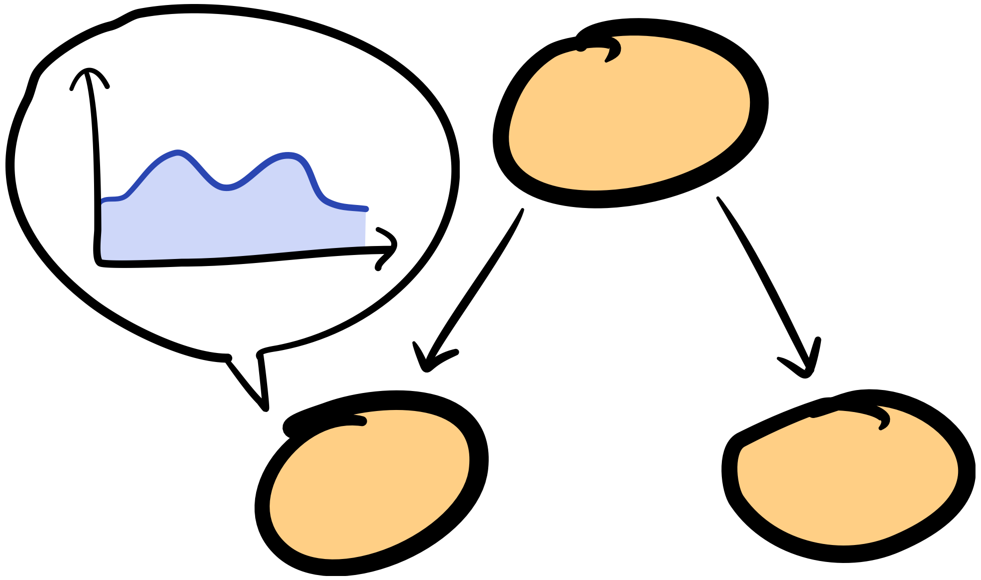
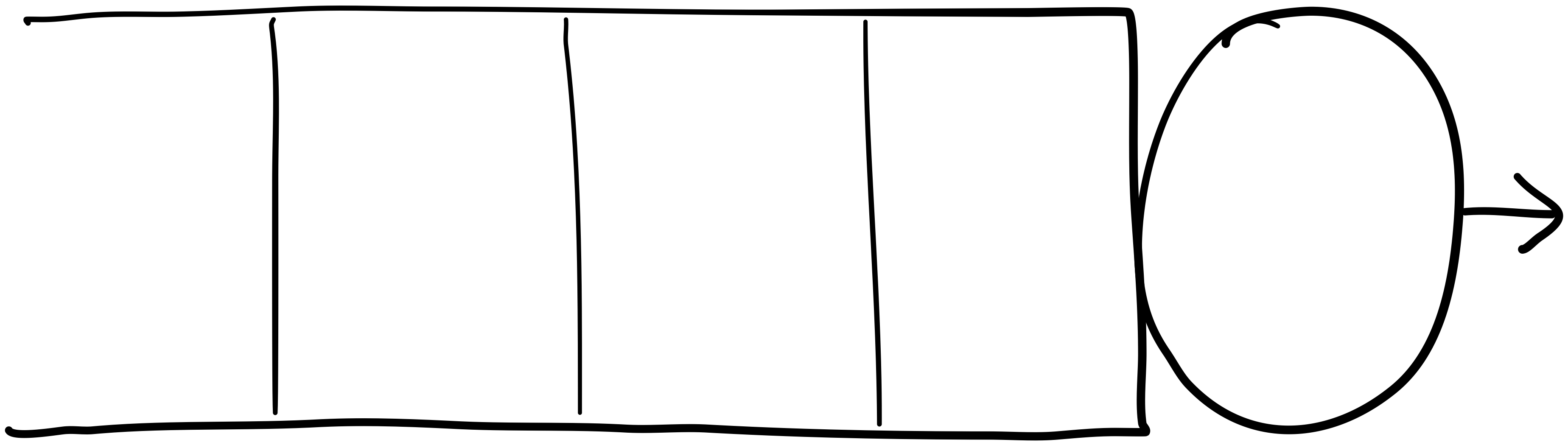
Optimal Scheduling and Exact Response Time Analysis for **Multistage Jobs**

Ziv Scully
Mor Harchol-Balter
Alan Scheller-Wolf

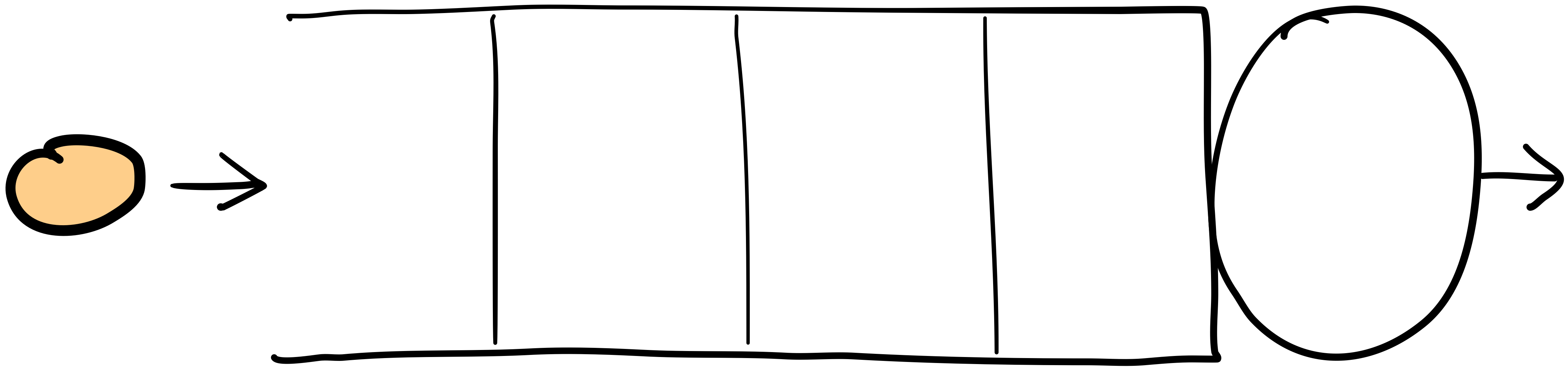
Carnegie Mellon University



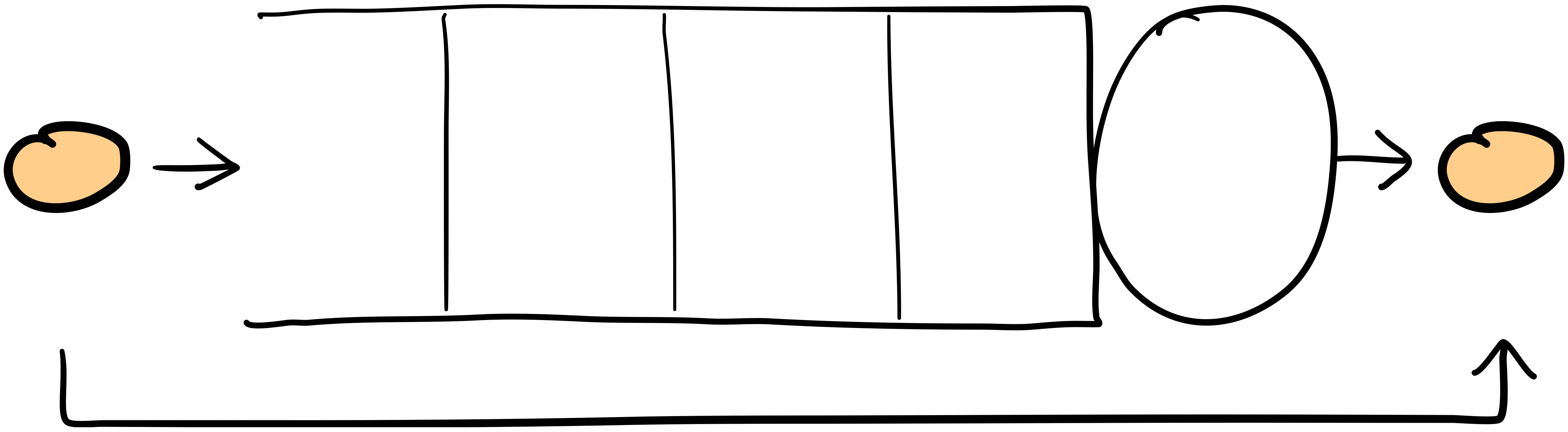
Response Time



Response Time

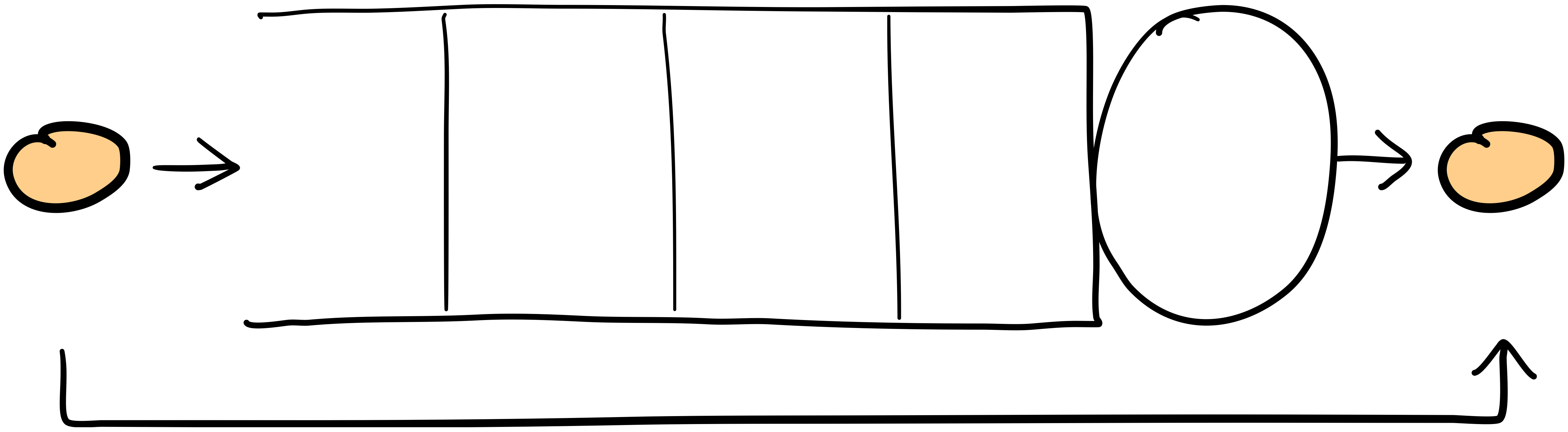


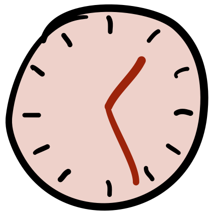
Response Time



🕒 = T = *response time*

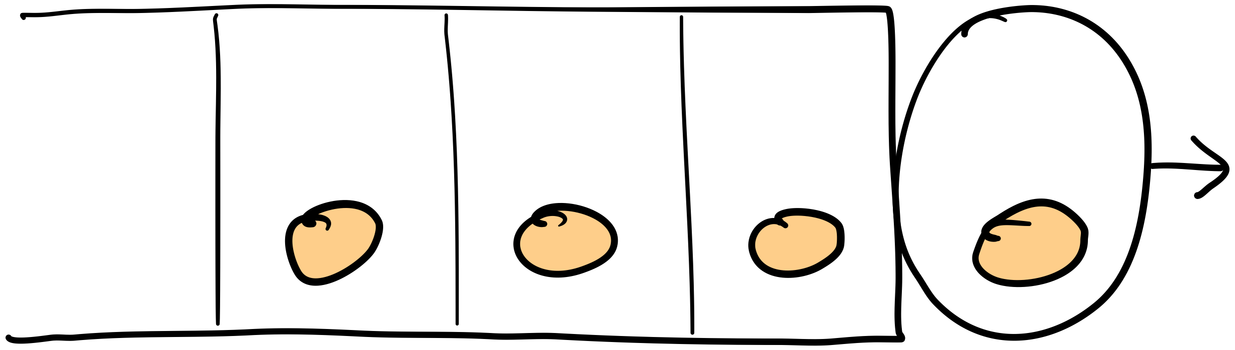
Response Time



 $= T =$ *response time*

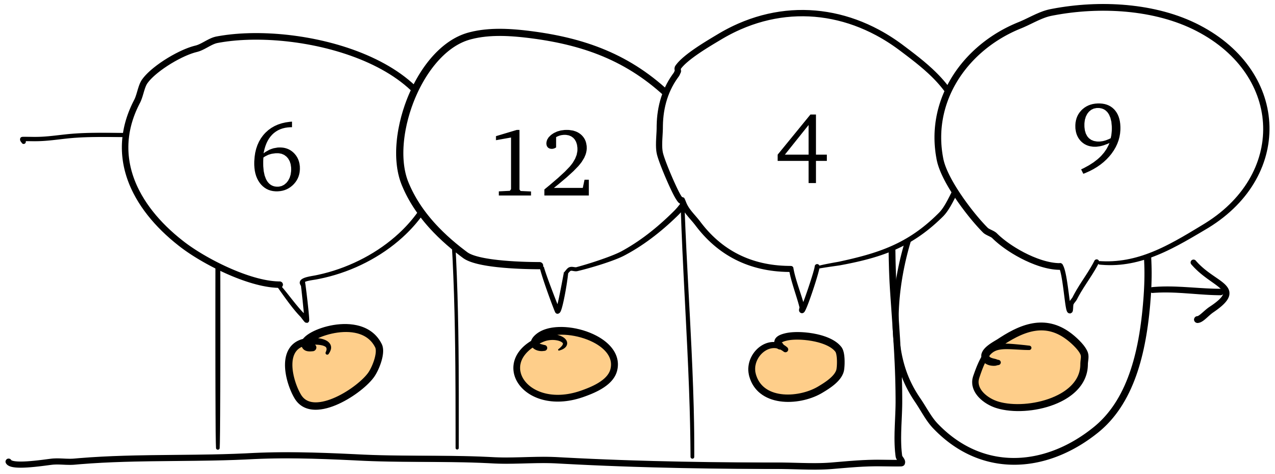
Goal: schedule to minimize *mean response time* $\mathbf{E}[T]$

Perfect Information



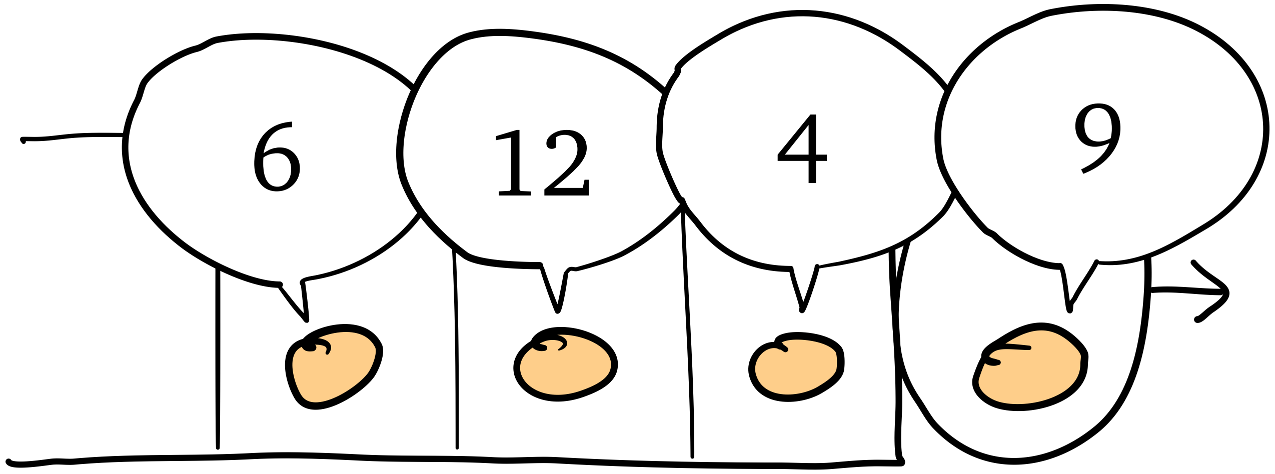
Known job sizes

Perfect Information



Known job sizes

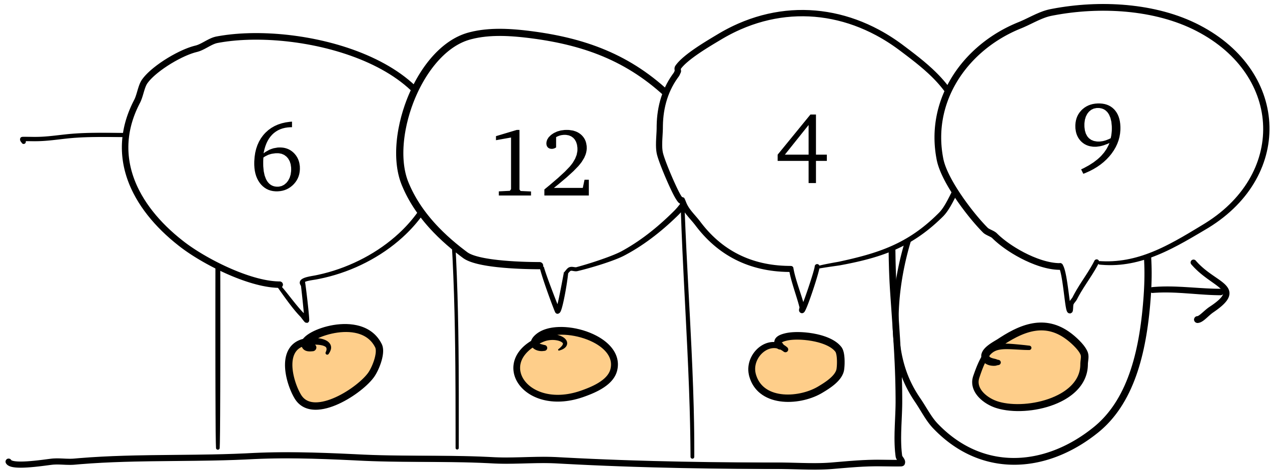
Perfect Information



Known job sizes

Optimal policy: *SRPT*
(serve job of smallest *remaining size*)

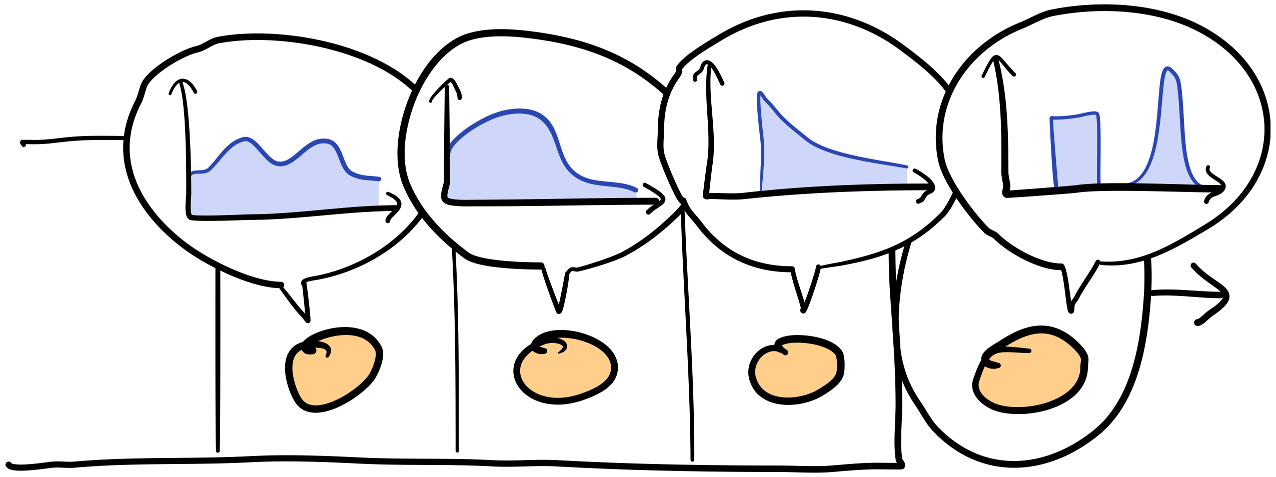
Perfect Information



Known job sizes

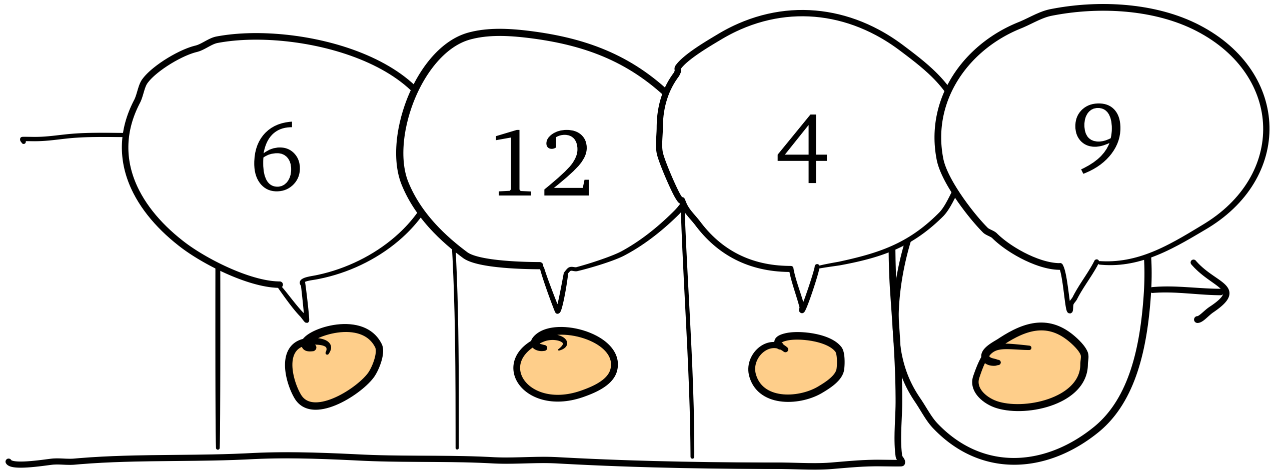
Optimal policy: *SRPT*
(serve job of smallest *remaining size*)

Zero Information



Unknown job sizes

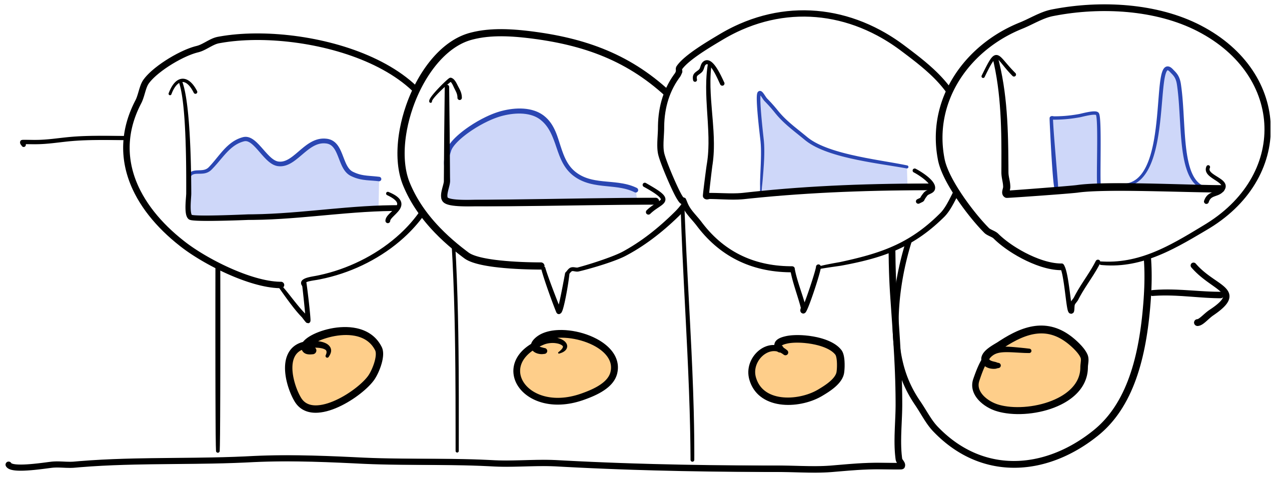
Perfect Information



Known job sizes

Optimal policy: *SRPT*
(serve job of smallest *remaining size*)

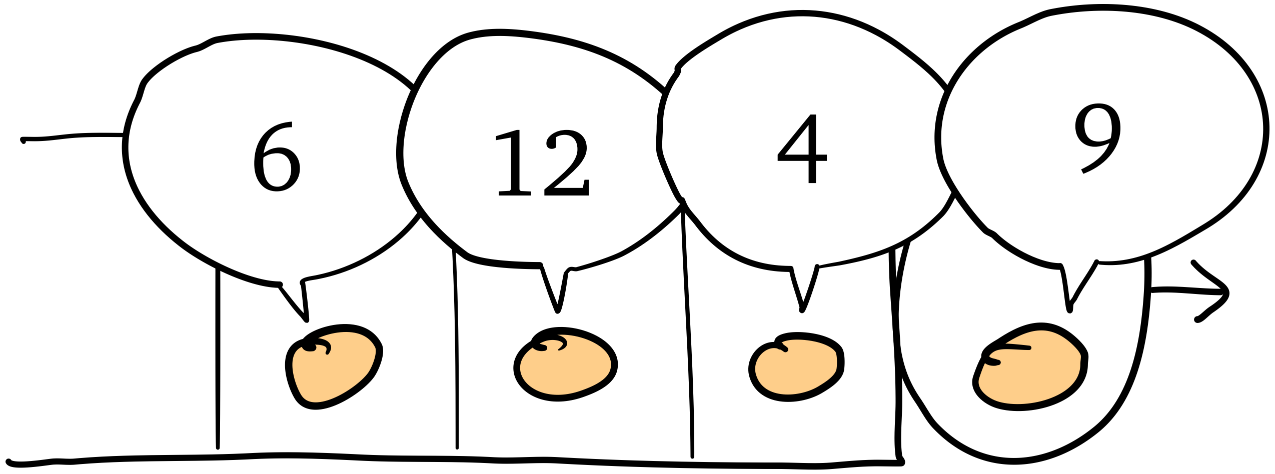
Zero Information



Unknown job sizes

Optimal policy: *Gittins policy*
(serve job of smallest *Gittins rank*)

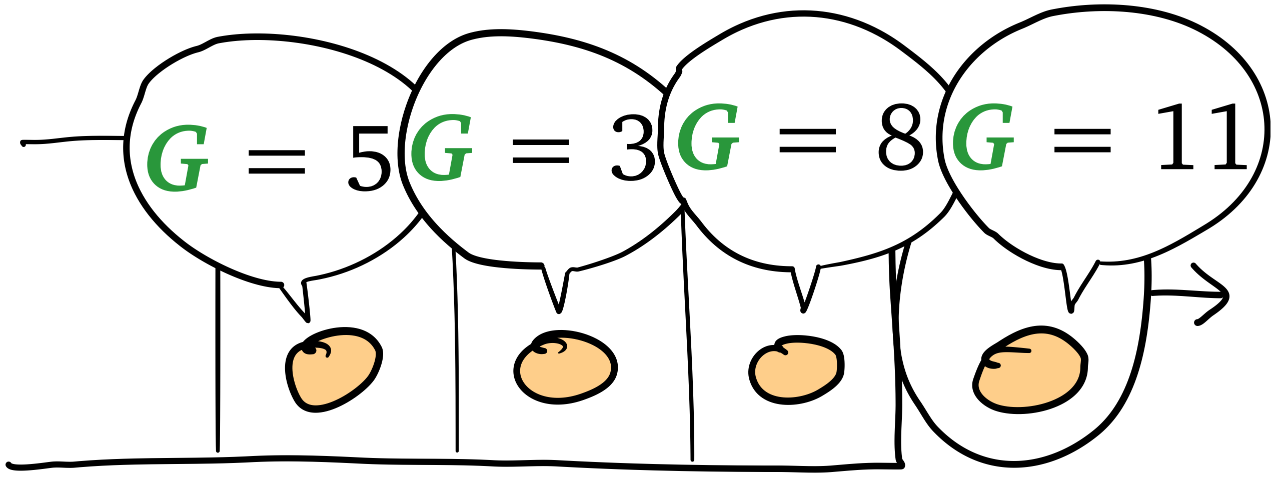
Perfect Information



Known job sizes

Optimal policy: *SRPT*
(serve job of smallest *remaining size*)

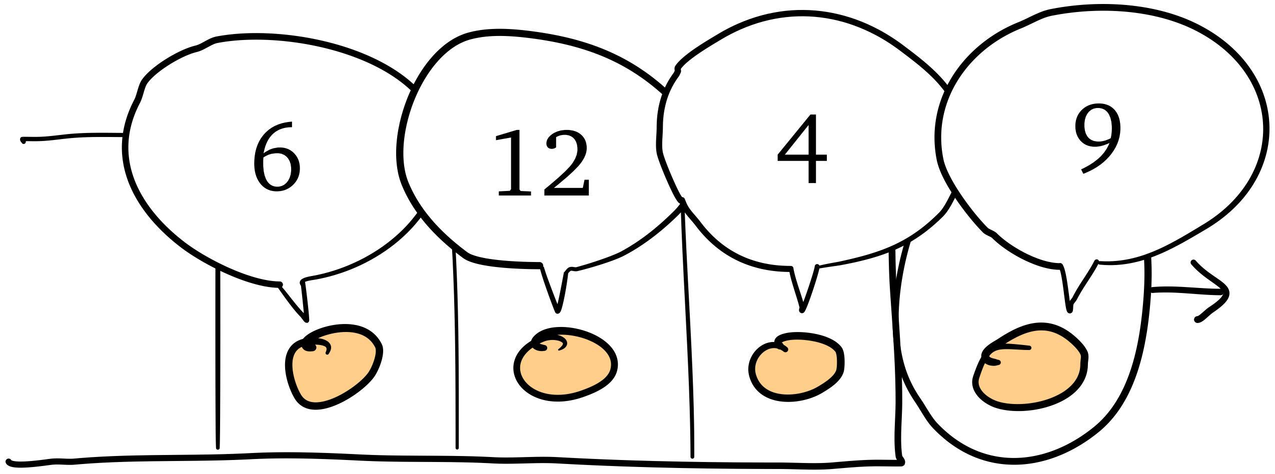
Zero Information



Unknown job sizes

Optimal policy: *Gittins policy*
(serve job of smallest *Gittins rank*)

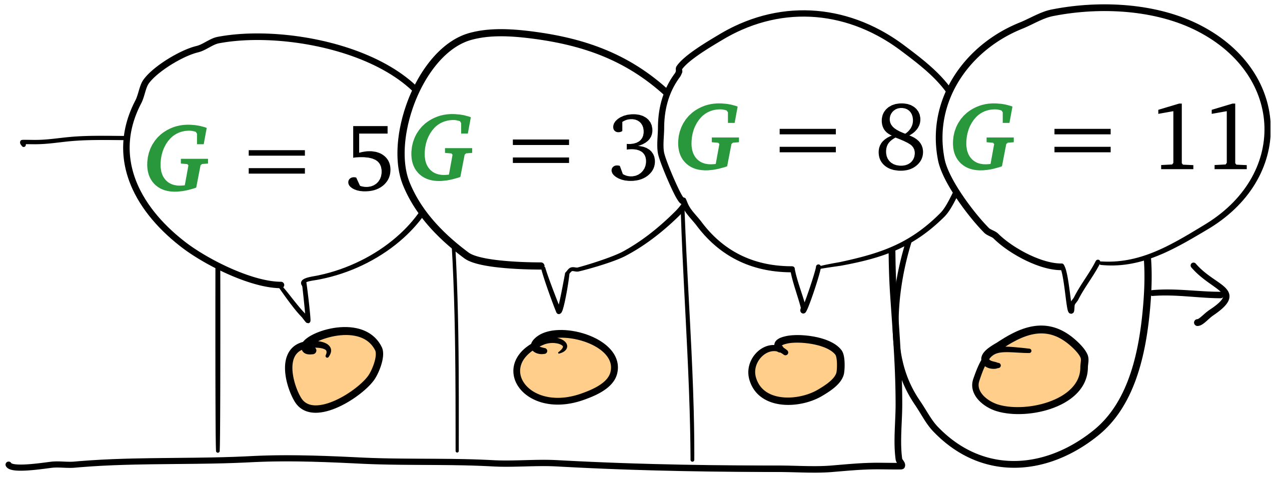
Perfect Information



Known job sizes

Optimal policy: *SRPT*
(serve job of smallest *remaining size*)

Zero Information



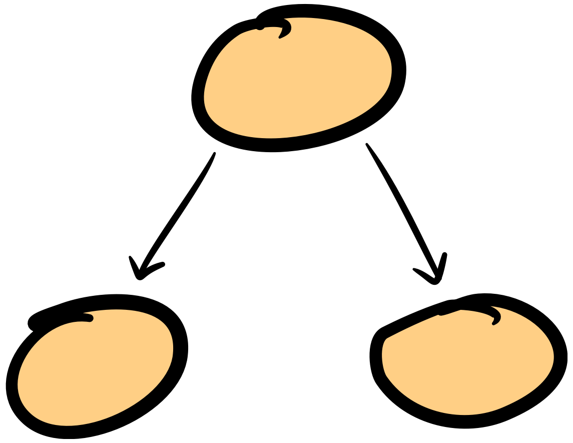
Unknown job sizes

Optimal policy: *Gittins policy*
(serve job of smallest *Gittins rank*)

Open problem: *partial information*

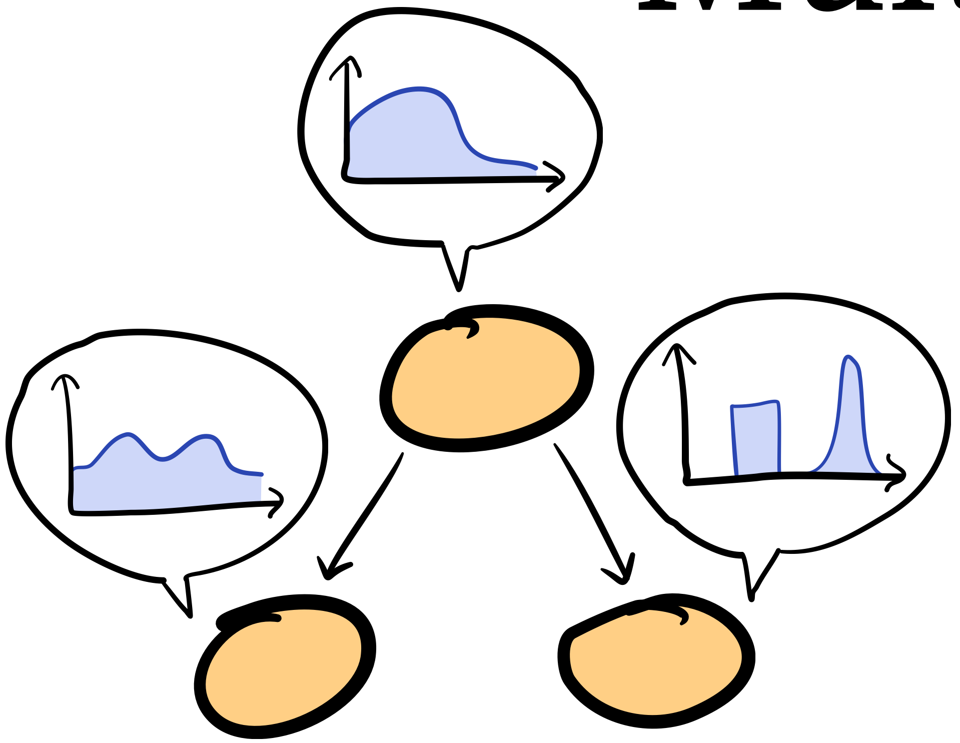
Partial Information: Multistage Jobs

Partial Information: Multistage Jobs



Jobs have *multiple stages*

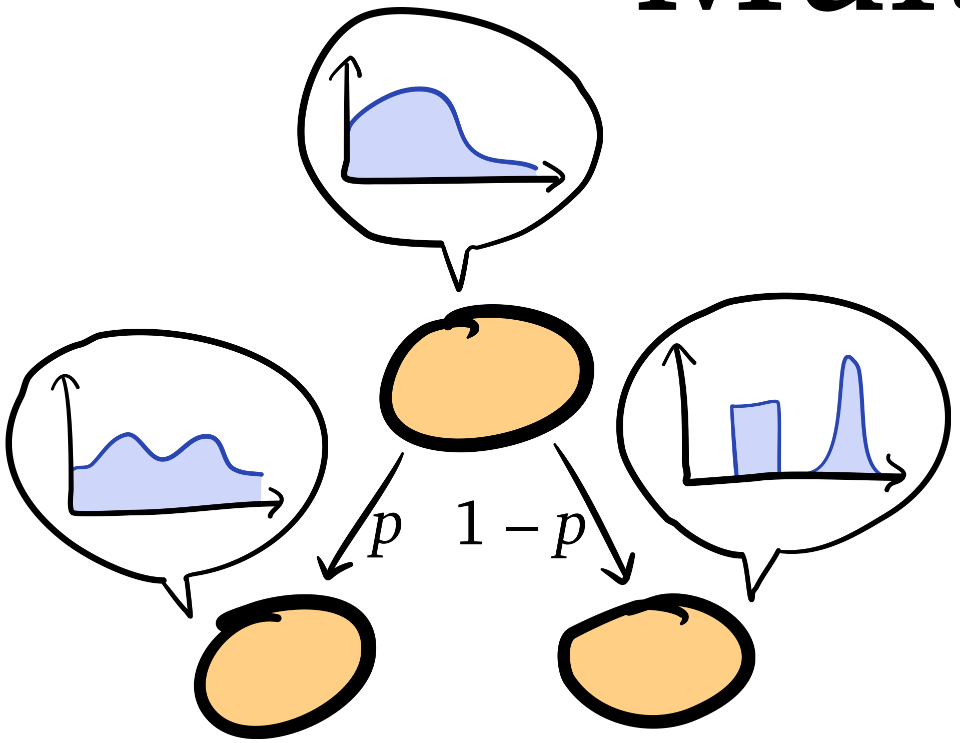
Partial Information: Multistage Jobs



Jobs have *multiple stages*

- unknown stage sizes

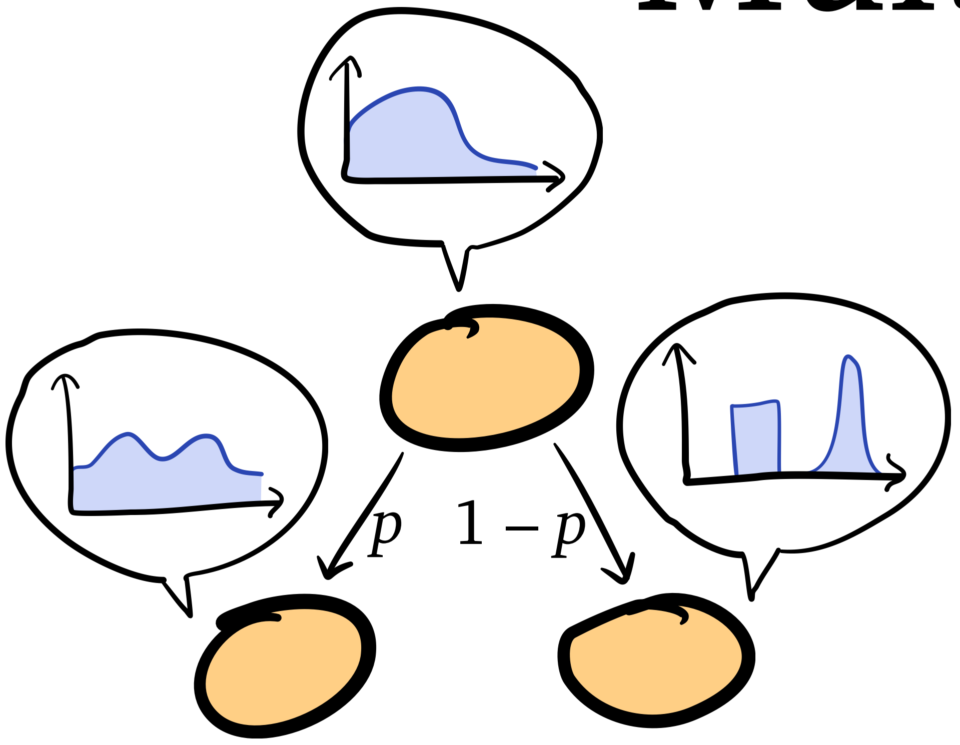
Partial Information: Multistage Jobs



Jobs have *multiple stages*

- unknown stage sizes
- unknown stage sequence

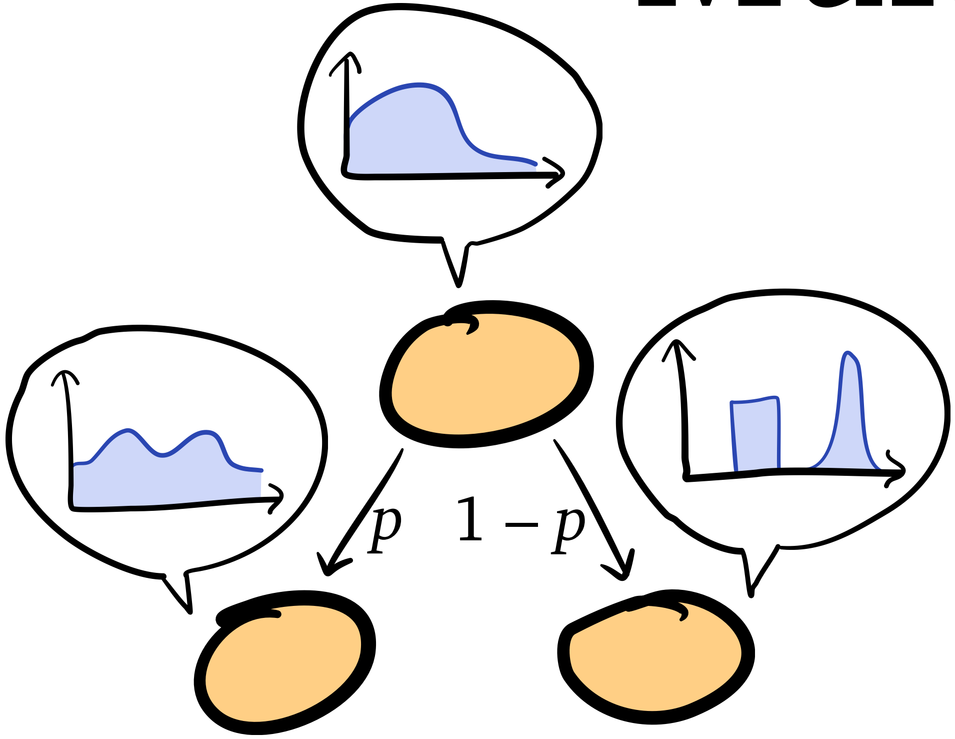
Partial Information: Multistage Jobs



Jobs have *multiple stages*

- unknown stage sizes
- unknown stage sequence
- … but know *which stage* we're on

Partial Information: Multistage Jobs

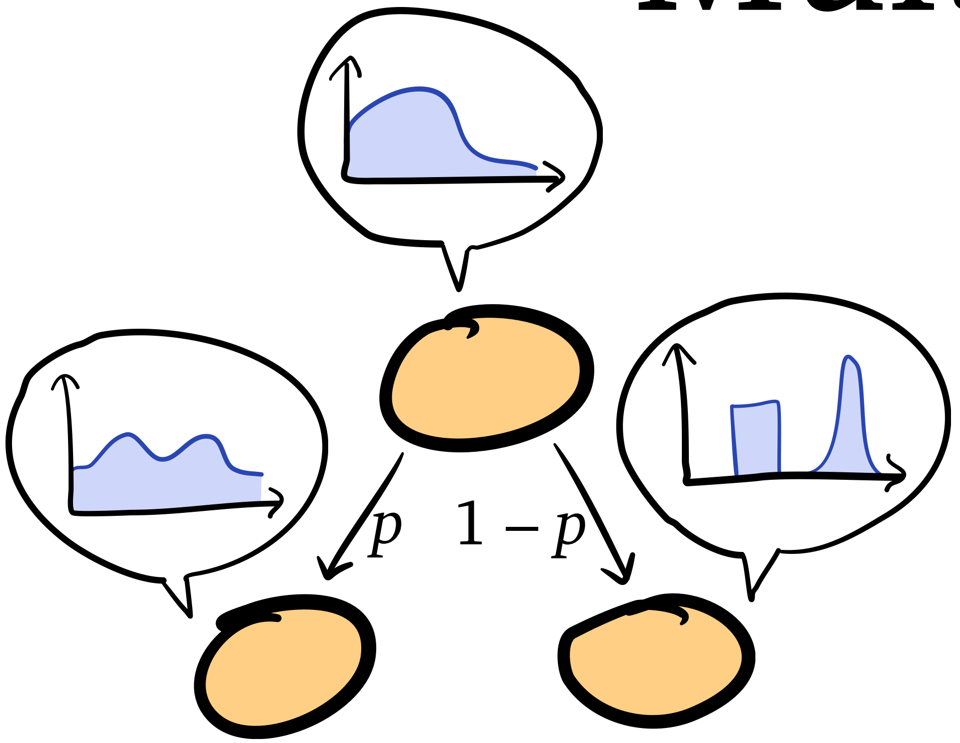


Jobs have *multiple stages*

- unknown stage sizes
- unknown stage sequence
- … but know *which stage* we're on

For each job, scheduler knows:

Partial Information: Multistage Jobs



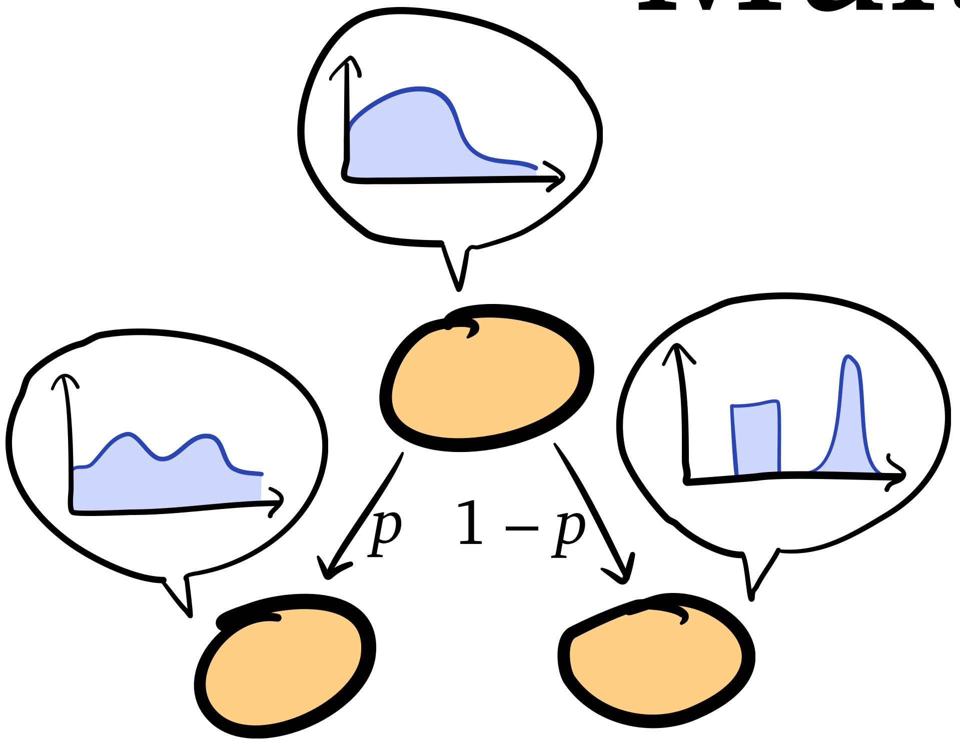
Jobs have *multiple stages*

- unknown stage sizes
- unknown stage sequence
- … but know *which stage* we're on

For each job, scheduler knows:

- *stage* in progress

Partial Information: Multistage Jobs



Jobs have *multiple stages*

- unknown stage sizes
- unknown stage sequence
- … but know *which stage* we're on

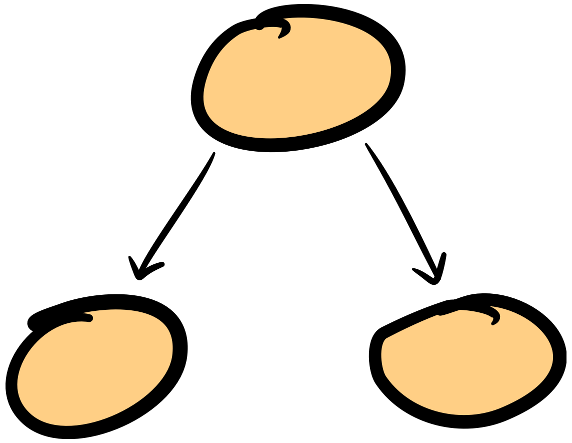
For each job, scheduler knows:

- *stage* in progress
- *age* of that stage

Multistage Job Examples

Multistage Job Examples

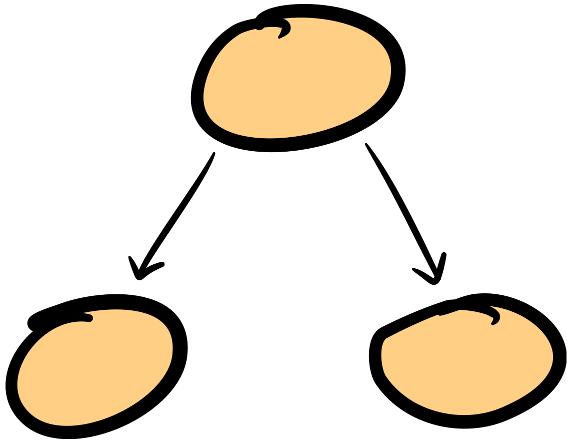
Job R: Repairing an item



Multistage Job Examples

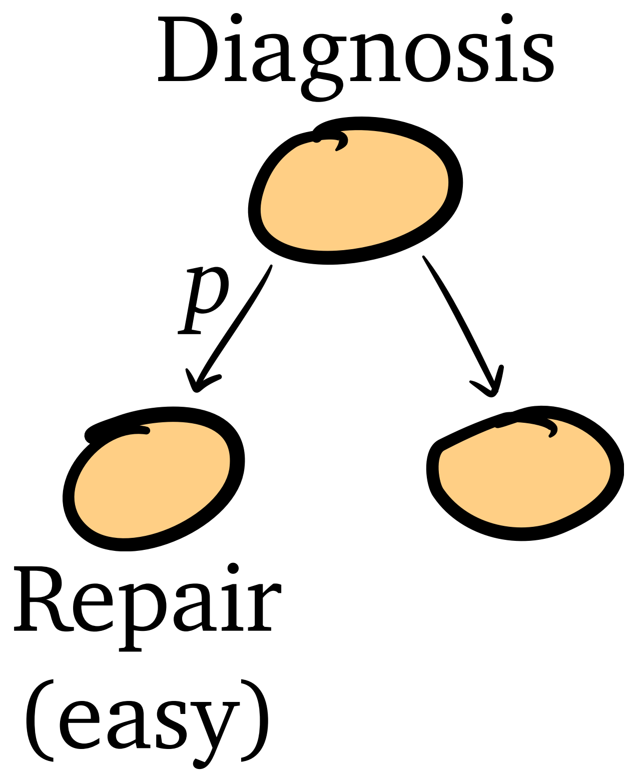
Job R: Repairing an item

Diagnosis



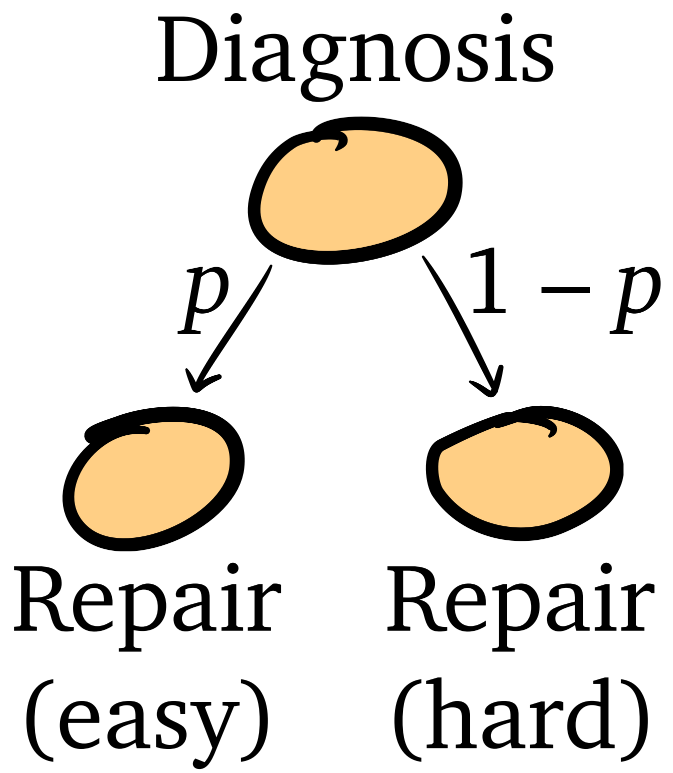
Multistage Job Examples

Job R: Repairing an item



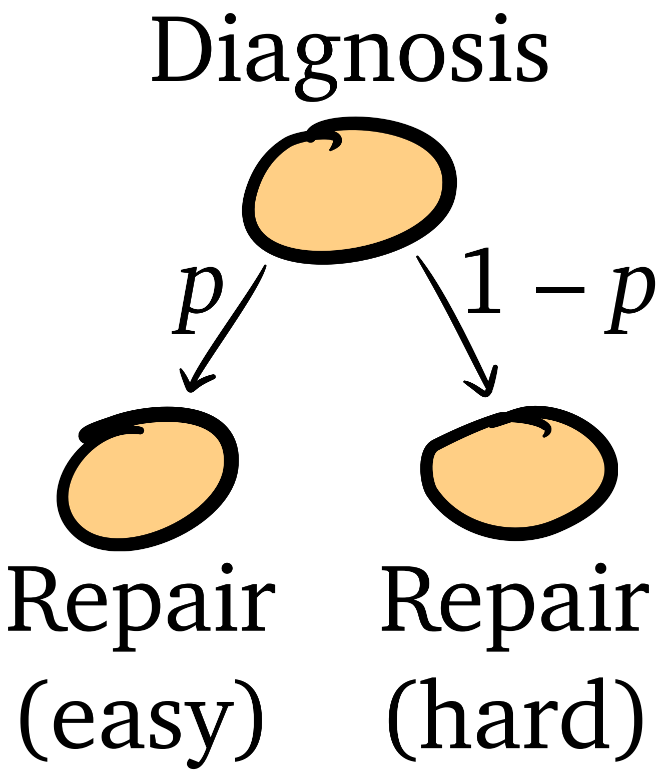
Multistage Job Examples

Job R: Repairing an item

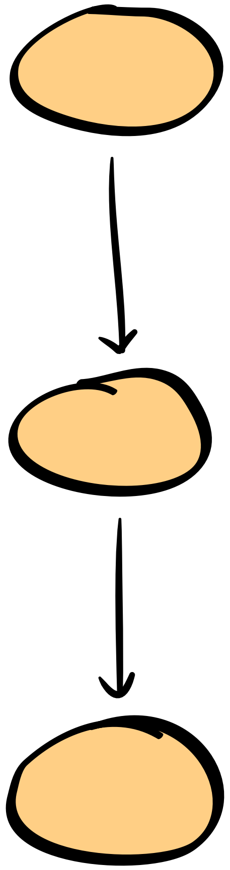


Multistage Job Examples

Job R: Repairing an item

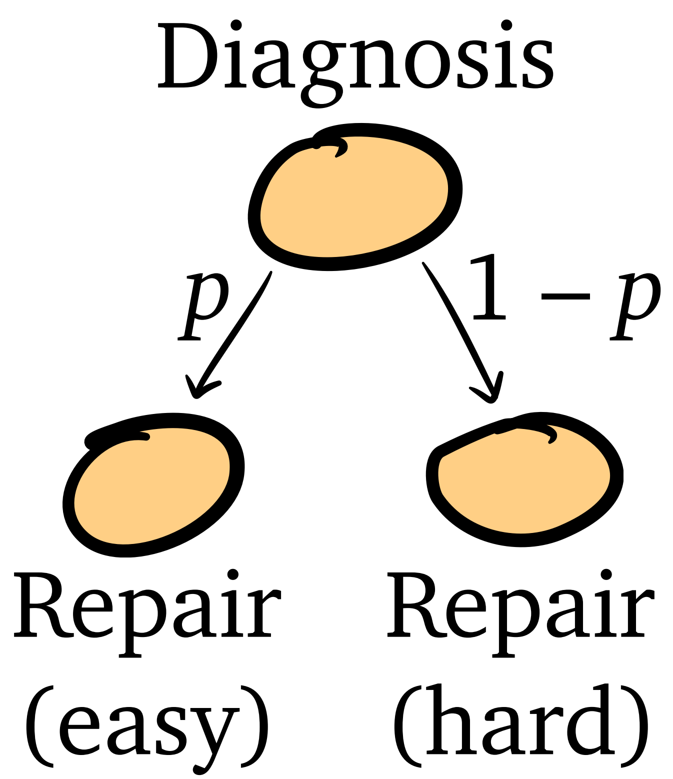


Job G: Google ad placement

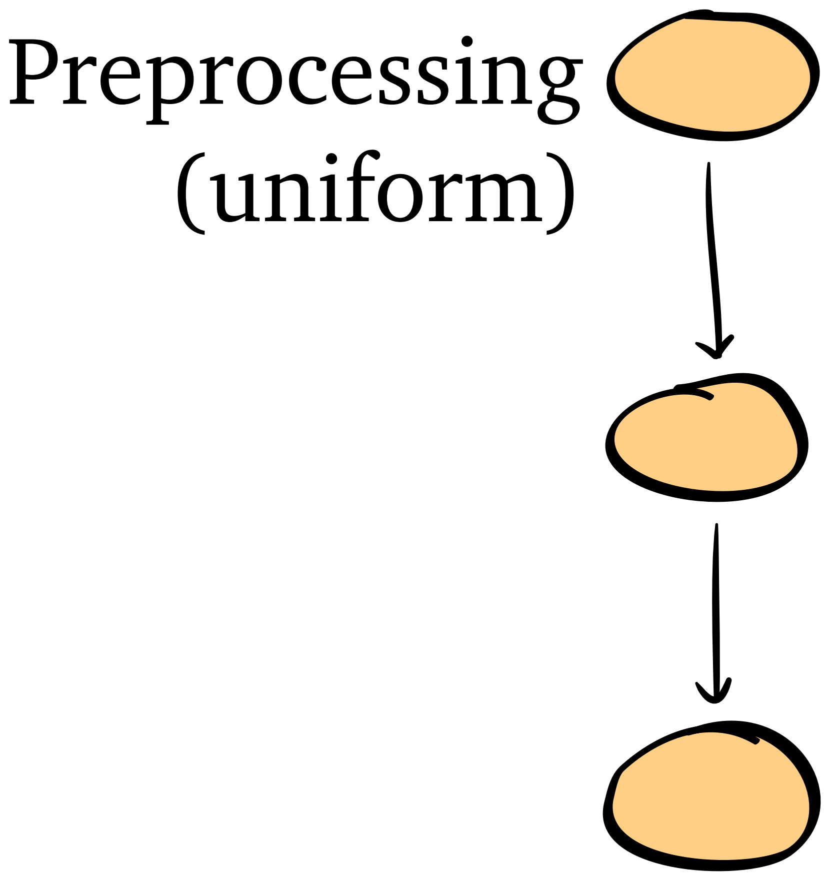


Multistage Job Examples

Job R: Repairing an item

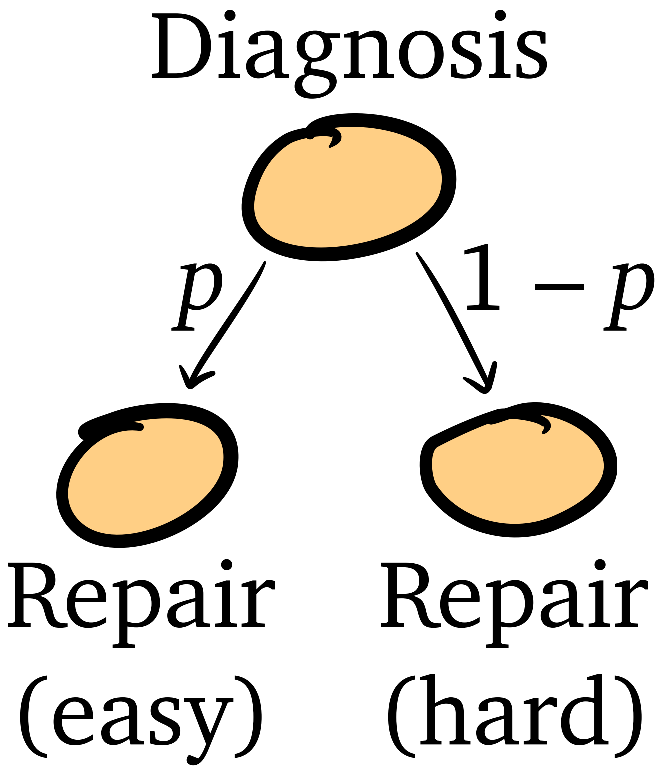


Job G: Google ad placement



Multistage Job Examples

Job R: Repairing an item



Diagnosis

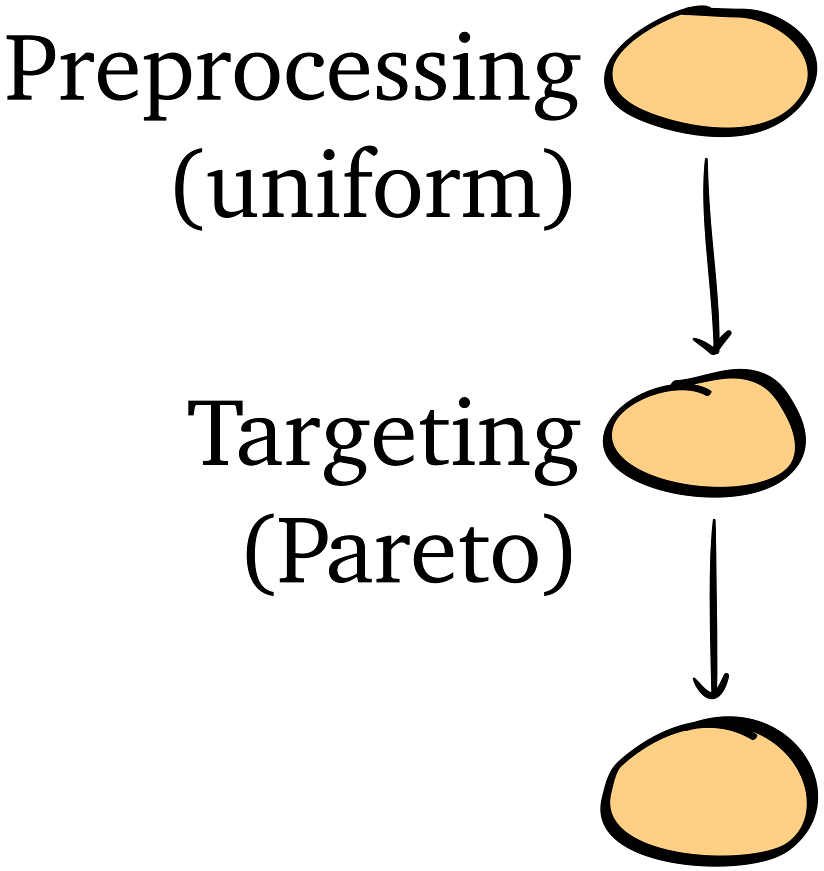
p

$1 - p$

Repair (easy)

Repair (hard)

Job G: Google ad placement

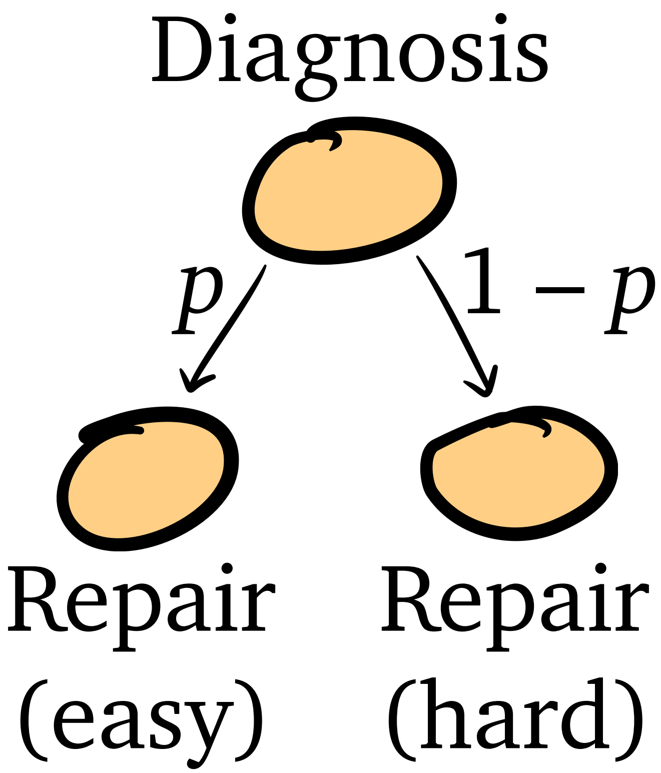


Preprocessing (uniform)

Targeting (Pareto)

Multistage Job Examples

Job R: Repairing an item



Diagnosis

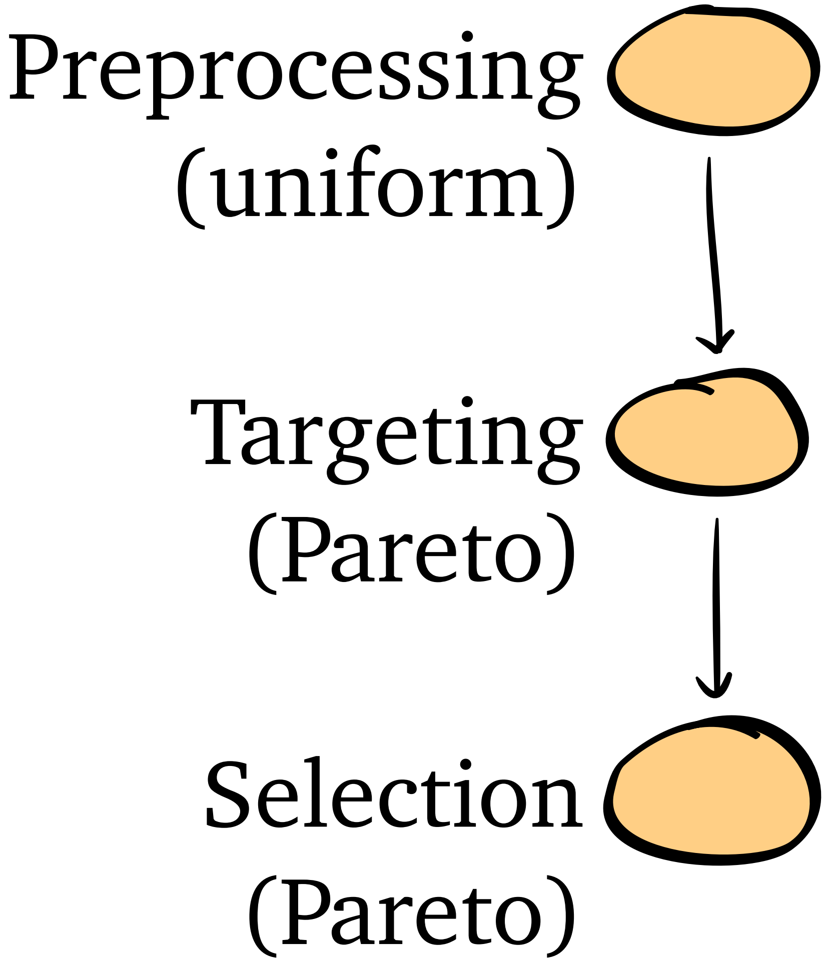
p

$1 - p$

Repair (easy)

Repair (hard)

Job G: Google ad placement



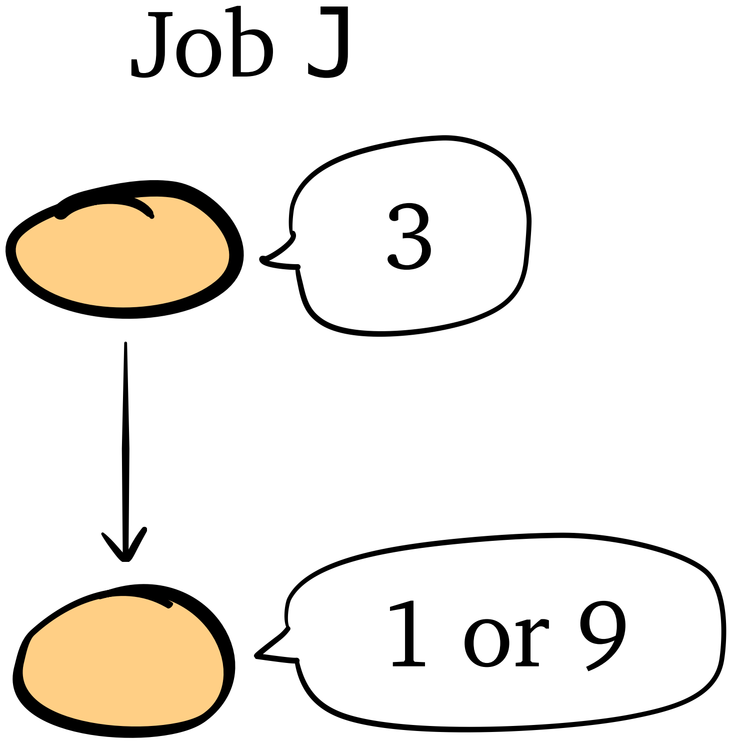
Preprocessing (uniform)

Targeting (Pareto)

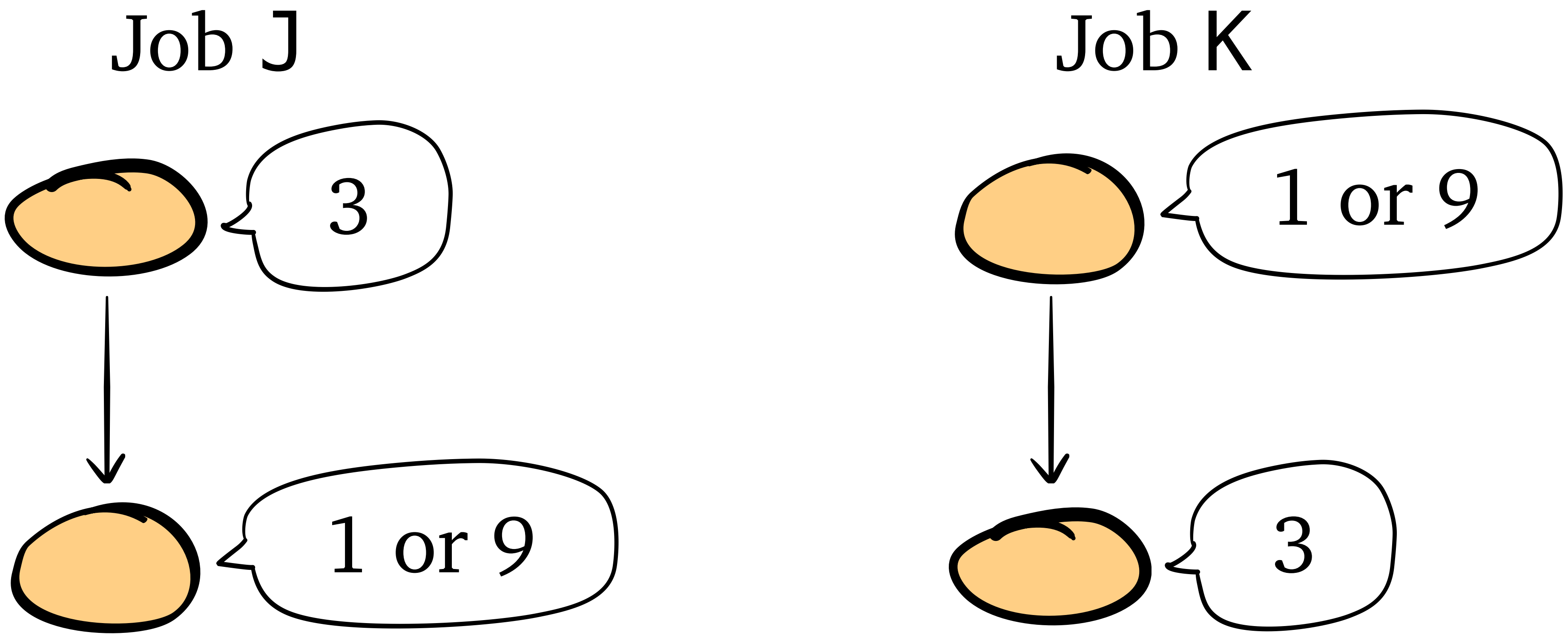
Selection (Pareto)

Scheduling Multistage Jobs

Scheduling Multistage Jobs



Scheduling Multistage Jobs



Job J

3

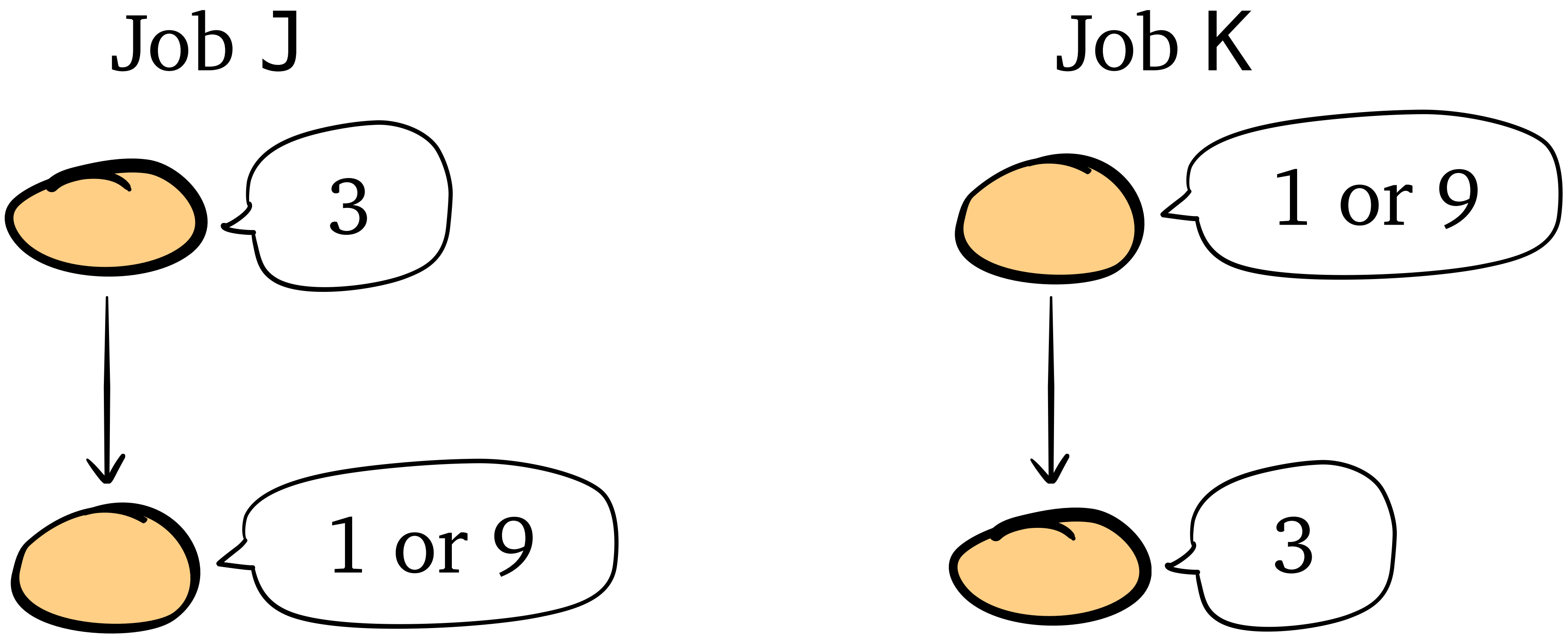
1 or 9

Job K

1 or 9

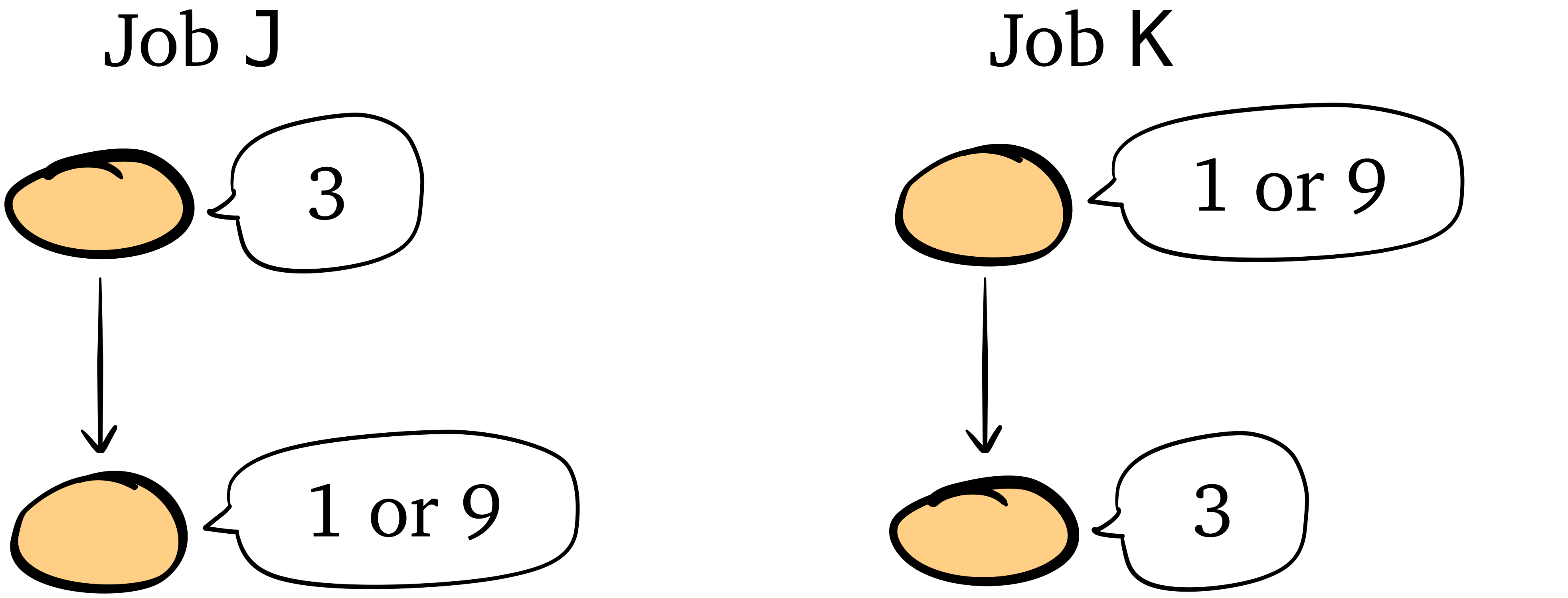
3

Scheduling Multistage Jobs



Which should we serve first?

Scheduling Multistage Jobs



Job J

3

1 or 9

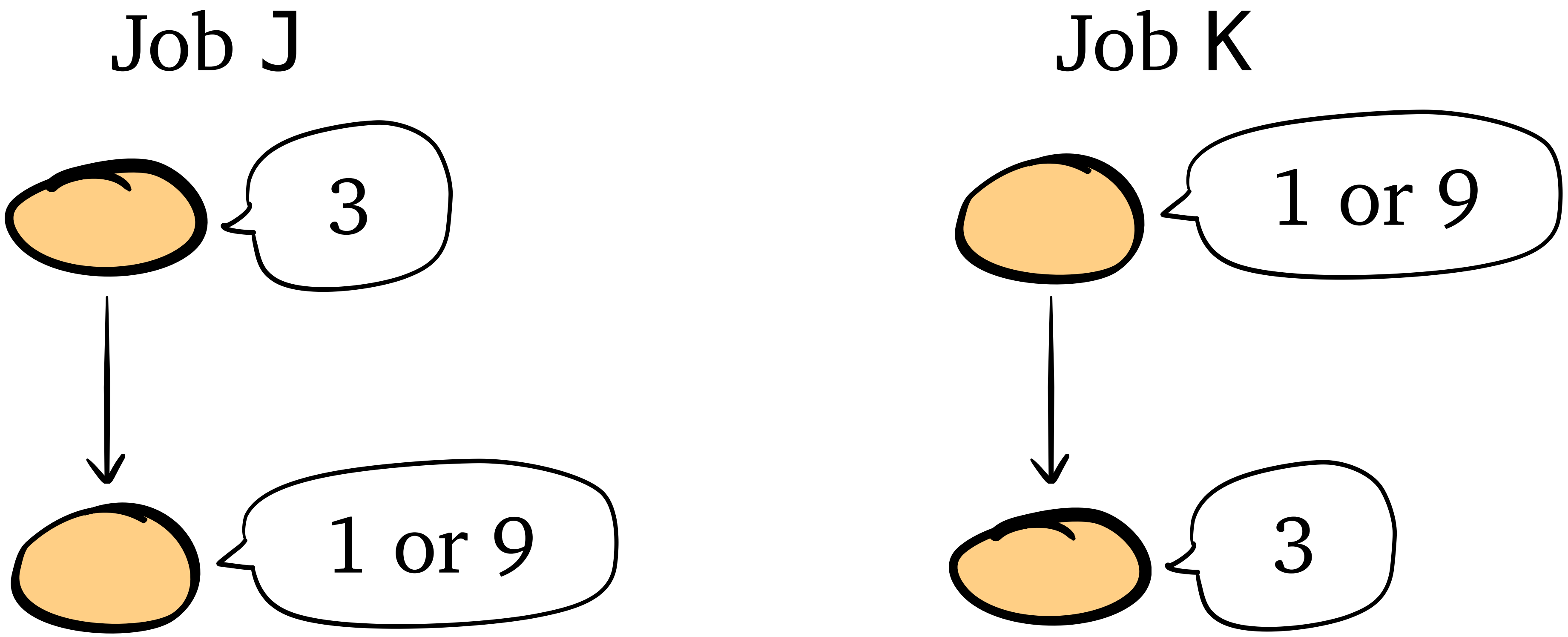
Job K

1 or 9

3

Which should we serve first? Job K first

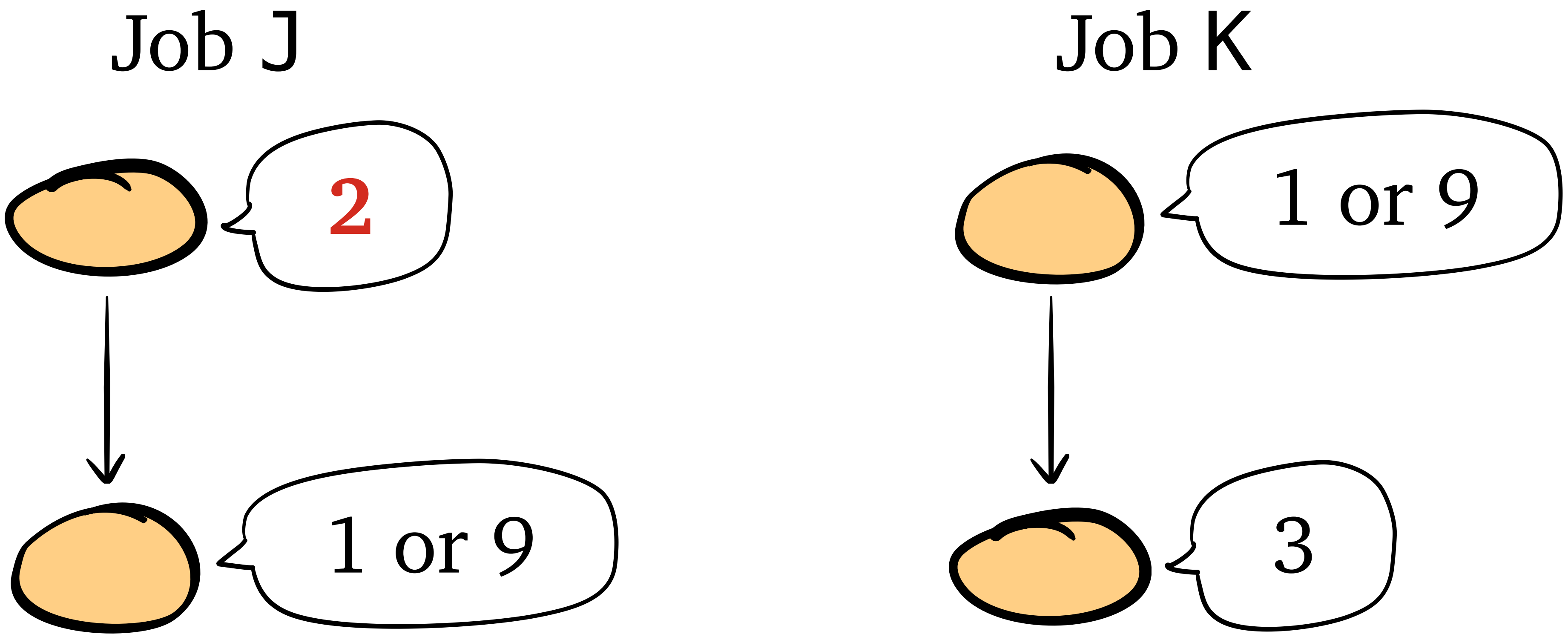
Scheduling Multistage Jobs



Which should we serve first? Job K first

What if we shorten J’s first stage?

Scheduling Multistage Jobs

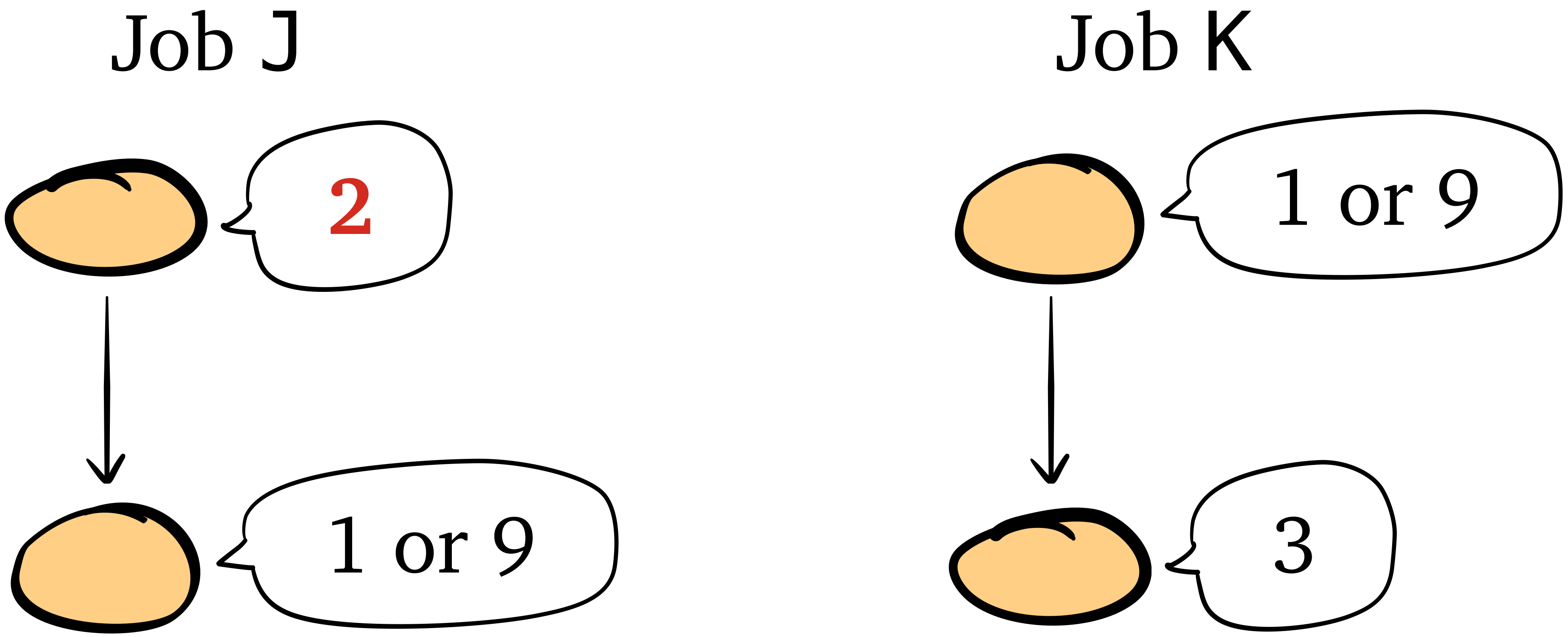


Which should we serve first? Job K first

What if we shorten J's first stage?

- Shorten to 2:

Scheduling Multistage Jobs

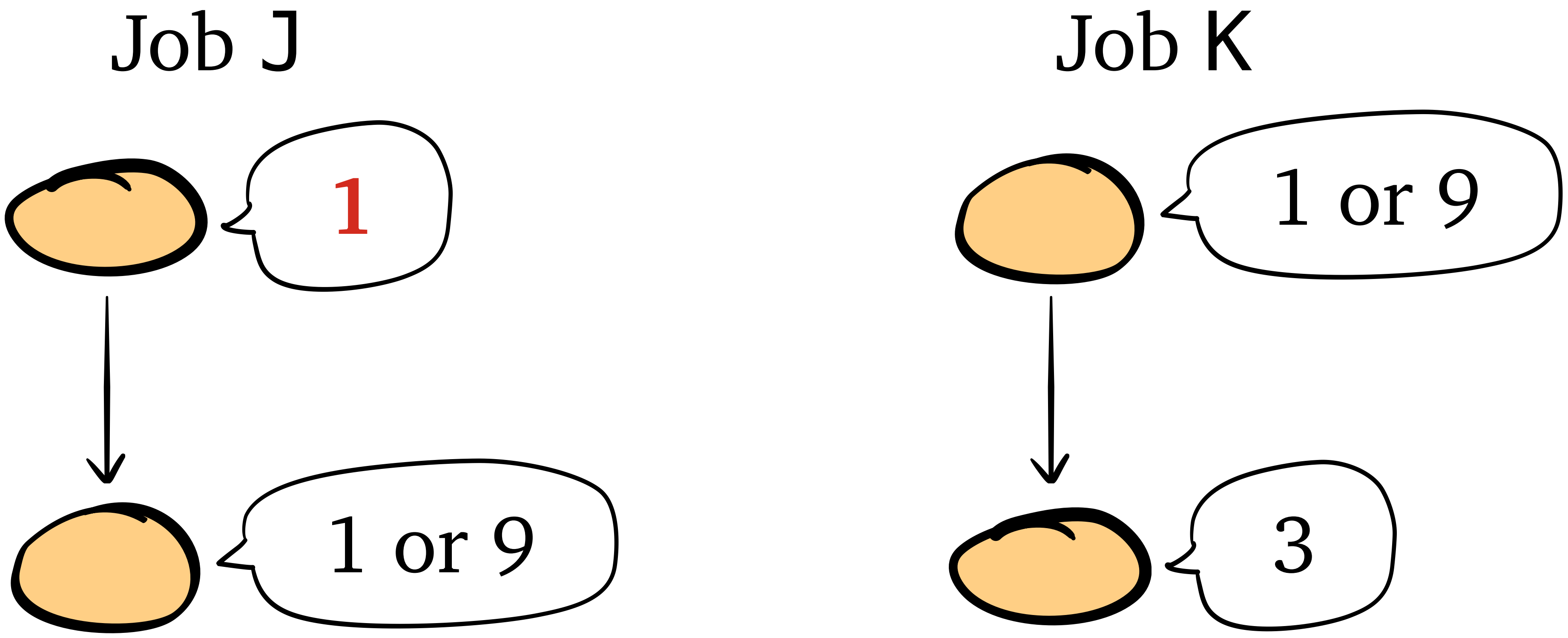


Which should we serve first? Job K first

What if we shorten J's first stage?

- Shorten to 2: still K first

Scheduling Multistage Jobs

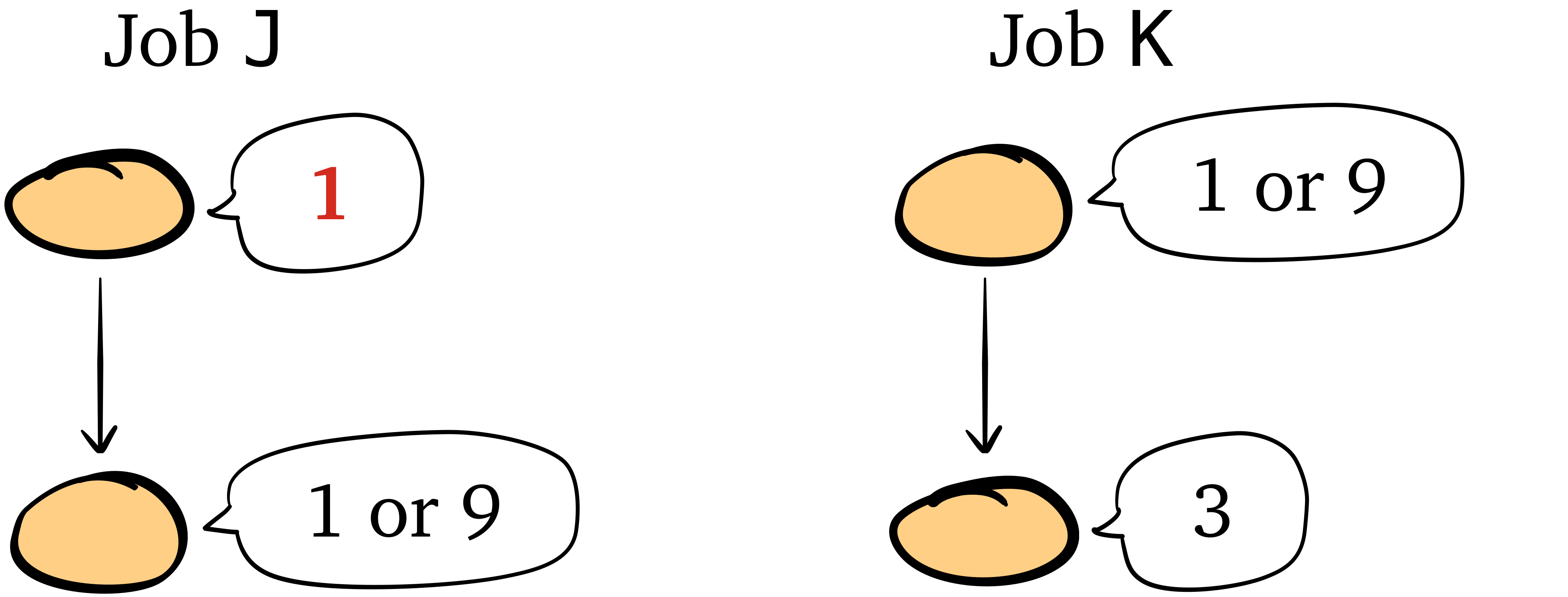


Which should we serve first? Job K first

What if we shorten J's first stage?

- Shorten to 2: still K first
- Shorten to 1:

Scheduling Multistage Jobs



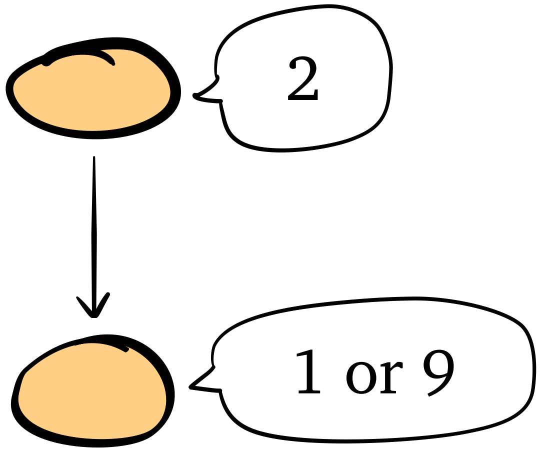
Which should we serve first? Job K first

What if we shorten J's first stage?

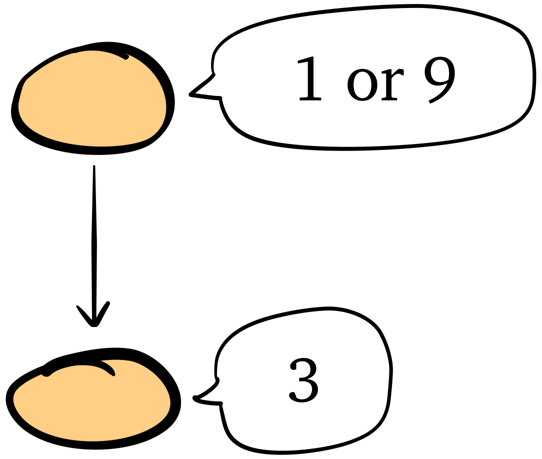
- Shorten to 2: still K first
- Shorten to 1: now J first

Tradeoff

Job J

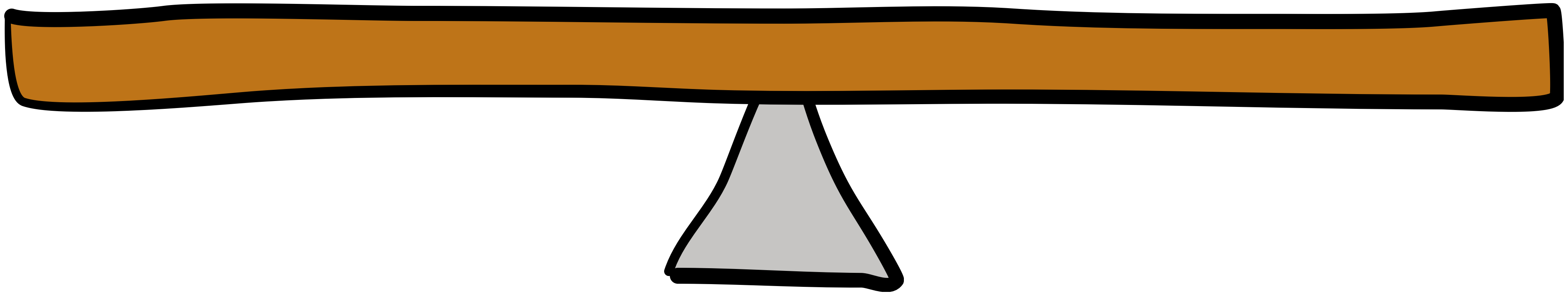


Job K

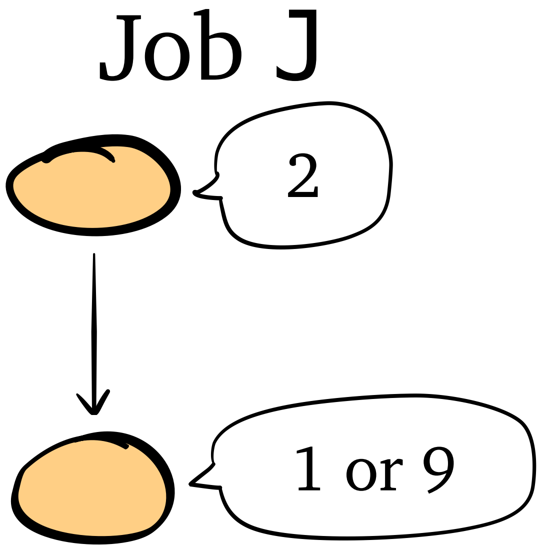


Prioritizing “short” jobs

Prioritizing “informative” jobs



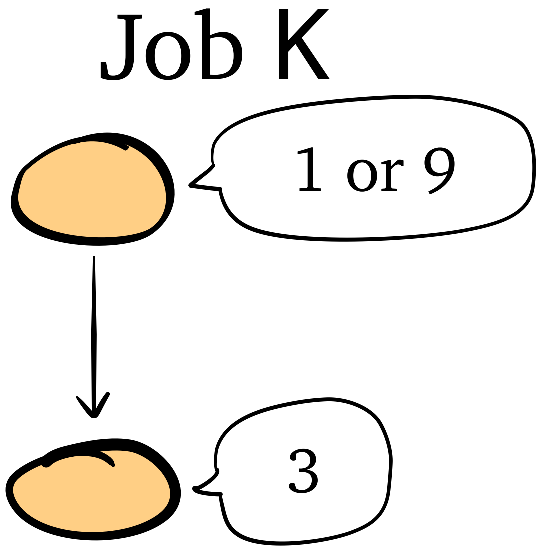
Tradeoff



Job J

2

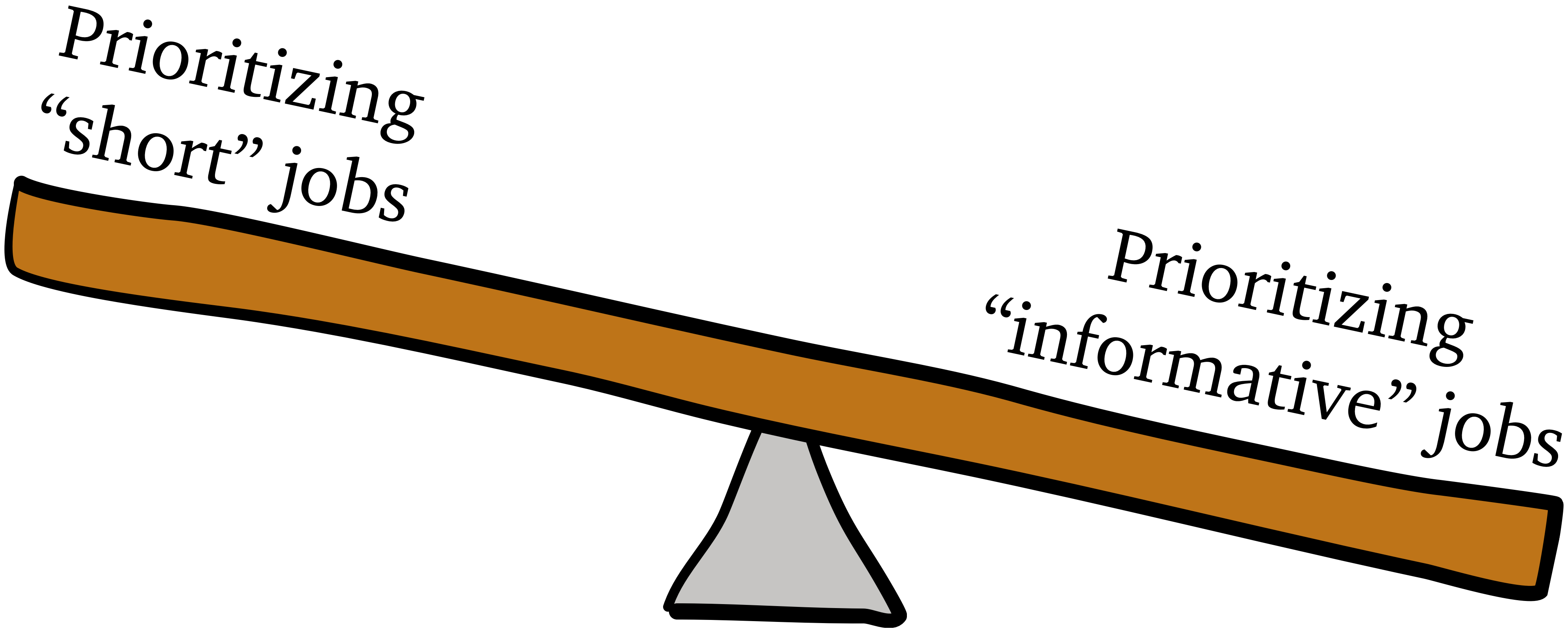
1 or 9



Job K

1 or 9

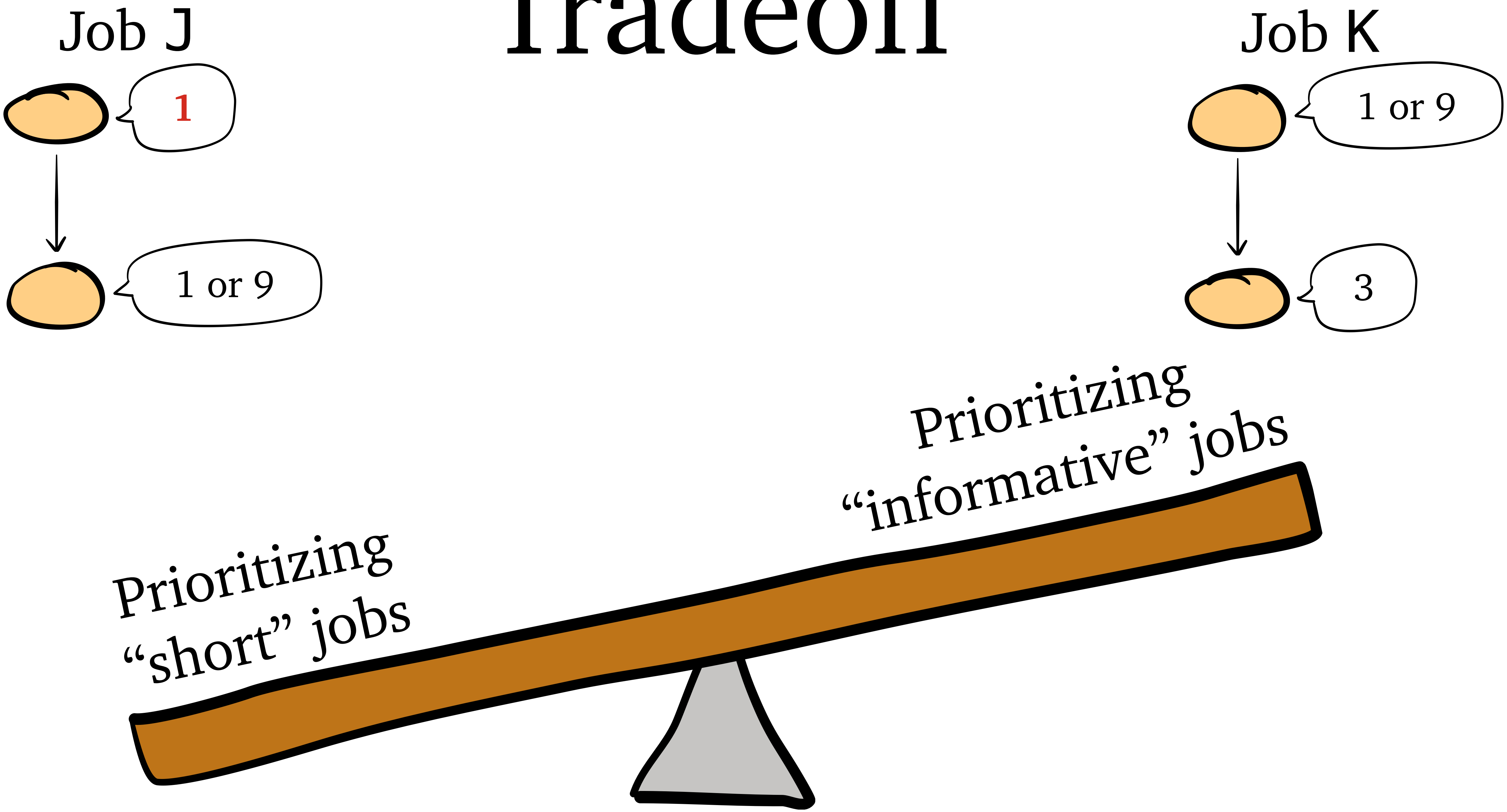
3



Prioritizing “short” jobs

Prioritizing “informative” jobs

Tradeoff



Job J

1

1 or 9

Job K

1 or 9

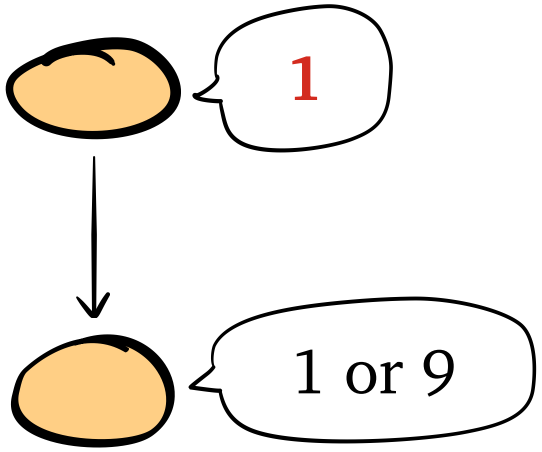
3

Prioritizing “short” jobs

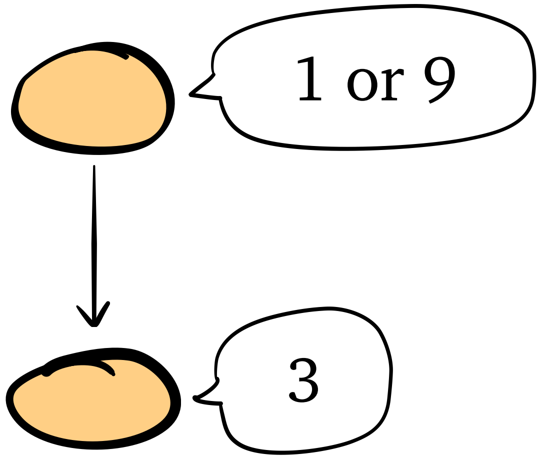
Prioritizing “informative” jobs

Tradeoff

Job J

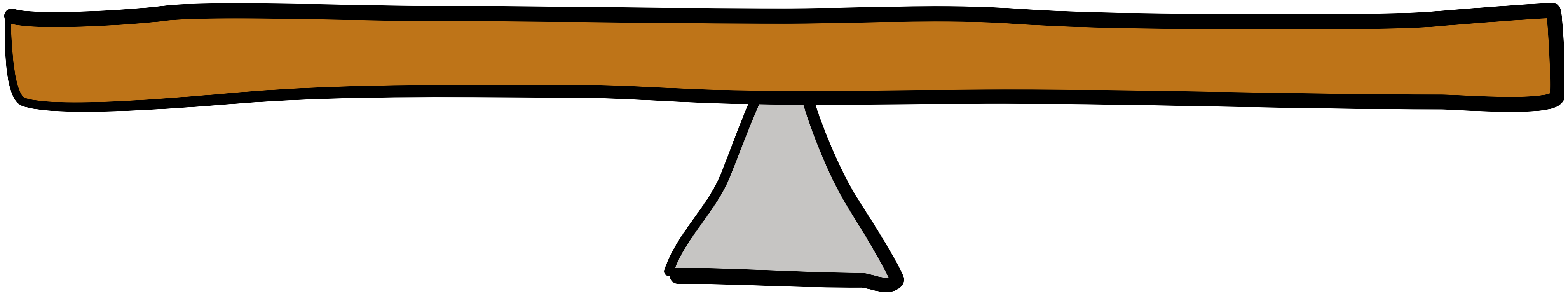


Job K



Prioritizing “short” jobs

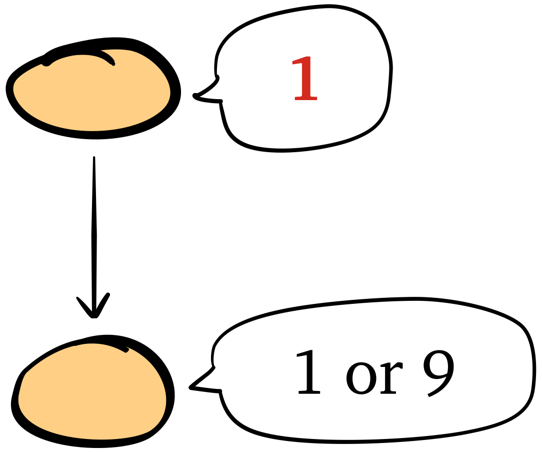
Prioritizing “informative” jobs



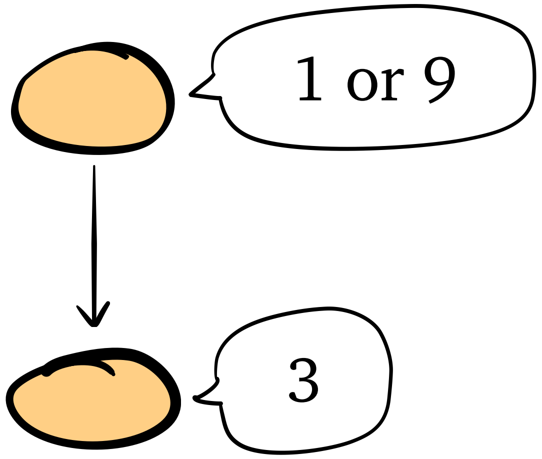
How do we balance the tradeoff?

Tradeoff

Job J

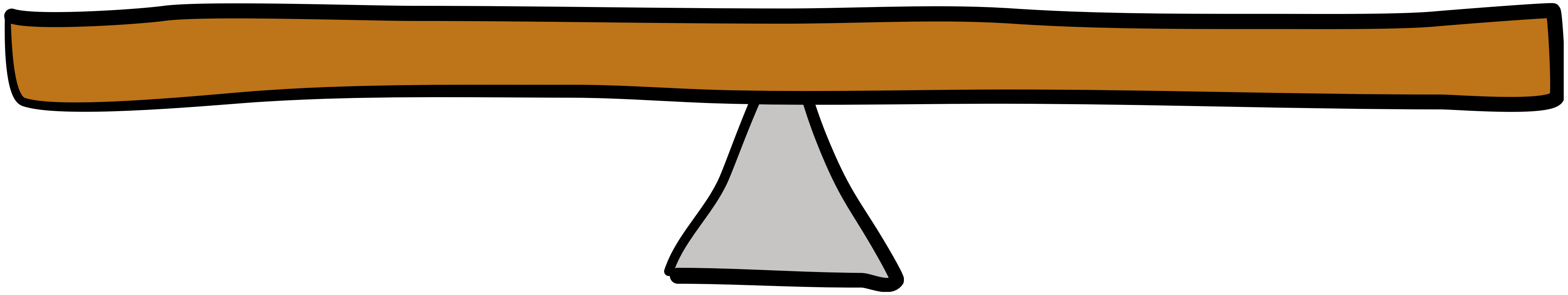


Job K



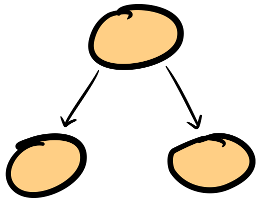
Prioritizing “short” jobs

Prioritizing “informative” jobs

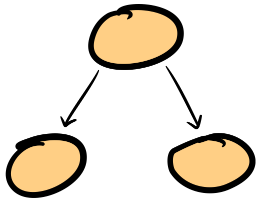


How do we balance the tradeoff?

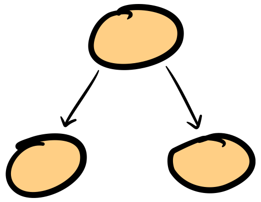
Multistage Gittins policy



For multistage jobs:

 For multistage jobs:

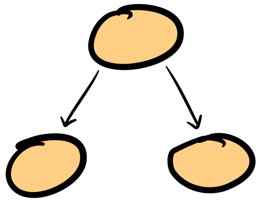
Gittins rank is *defined*…

 For multistage jobs:

Gittins rank is *defined*…

Gittins policy is *optimal*
for minimizing $\mathbf{E}[T]$…

Good News

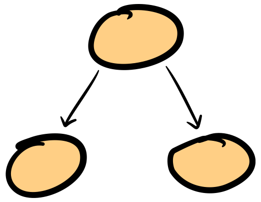

For multistage jobs:

Gittins rank is *defined*…

Gittins policy is *optimal* for minimizing $\mathbf{E}[T]$…

Good News

Bad News



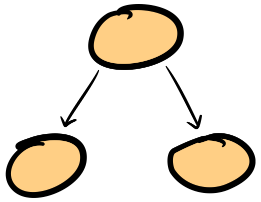
For multistage jobs:

Gittins rank is *defined*…

Gittins policy is *optimal* for minimizing $\mathbf{E}[T]$…

Good News

Bad News



For multistage jobs:

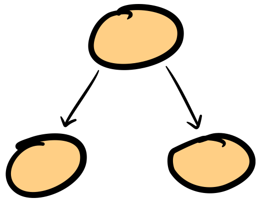
Gittins rank is *defined*…

… but is intractable to *compute*

Gittins policy is *optimal* for minimizing $\mathbf{E}[T]$…

Good News

Bad News



For multistage jobs:

Gittins rank is *defined*…

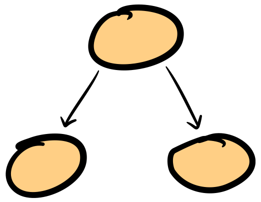
… but is intractable to *compute*

Gittins policy is *optimal* for minimizing $\mathbf{E}[T]$…

… but unknown how to *analyze* $\mathbf{E}[T]$

Good News

Bad News



For multistage jobs:

Gittins rank is *defined*…

… but is intractable to *compute*

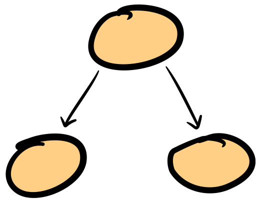
Gittins policy is *optimal* for minimizing $\mathbf{E}[T]$…

… but unknown how to *analyze* $\mathbf{E}[T]$

Need *new version* of Gittins policy

Good News

Bad News


For multistage jobs:

Gittins rank is *defined*…	… but is intractable to *compute*
Gittins policy is *optimal* for minimizing $\mathbf{E}[T]$…	… but unknown how to *analyze* $\mathbf{E}[T]$

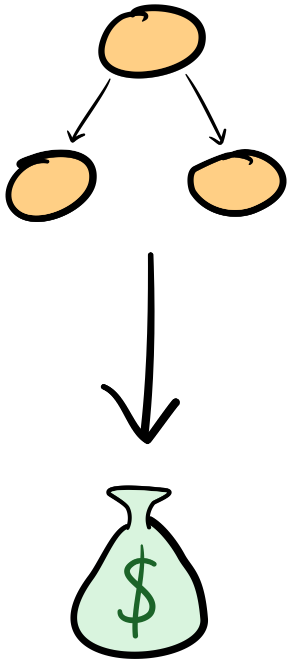
Need *new version* of Gittins policy
with **no bad news**

Our contribution:

a *new approach* to the Gittins policy that *naturally scales* to multistage jobs

Our contribution:

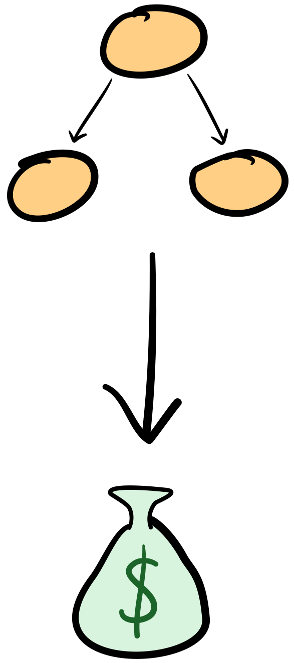
a *new approach* to the Gittins policy that *naturally scales* to multistage jobs



New approach: single-job profit (SJP)

Our contribution:

a *new approach* to the Gittins policy that *naturally scales* to multistage jobs

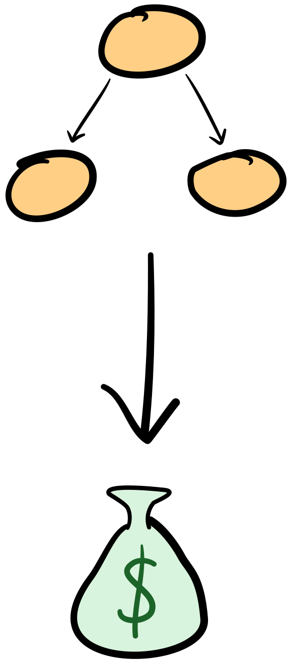


New approach: single-job profit (SJP)

- Helps *compute* Gittins rank of multistage jobs

Our contribution:

a *new approach* to the Gittins policy that *naturally scales* to multistage jobs



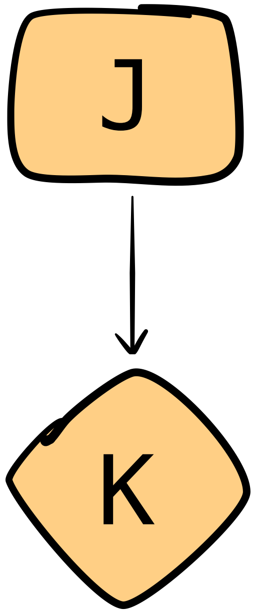
New approach: single-job profit (SJP)

- Helps *compute* Gittins rank of multistage jobs
- Yields *exact formula* for $\mathbf{E}[T]$

Wanted: Composition Law

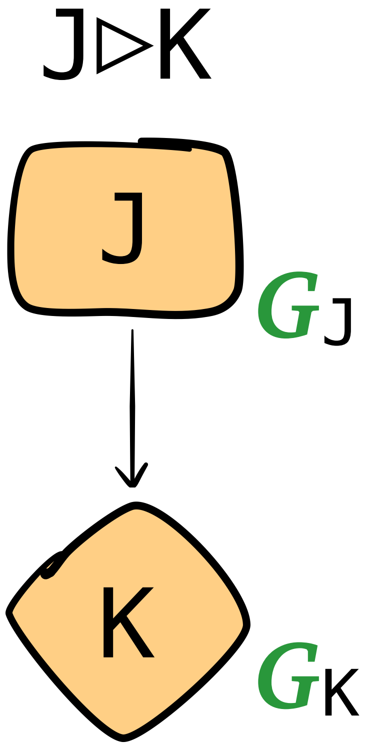
Sequential Composition

J▷K



Wanted: Composition Law

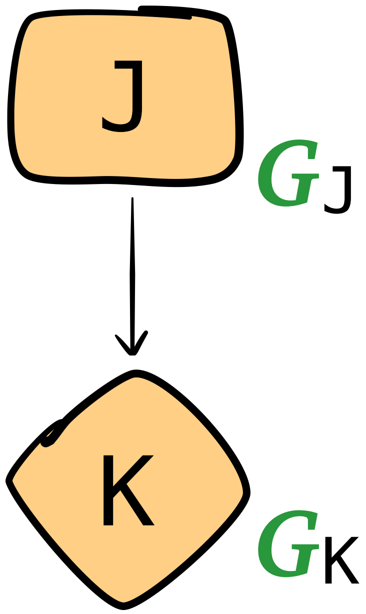
Sequential Composition



Wanted: Composition Law

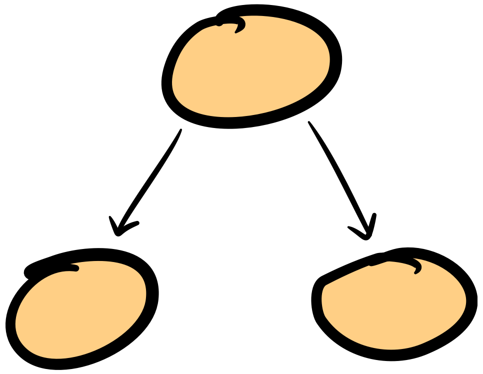
Sequential Composition

J▷K



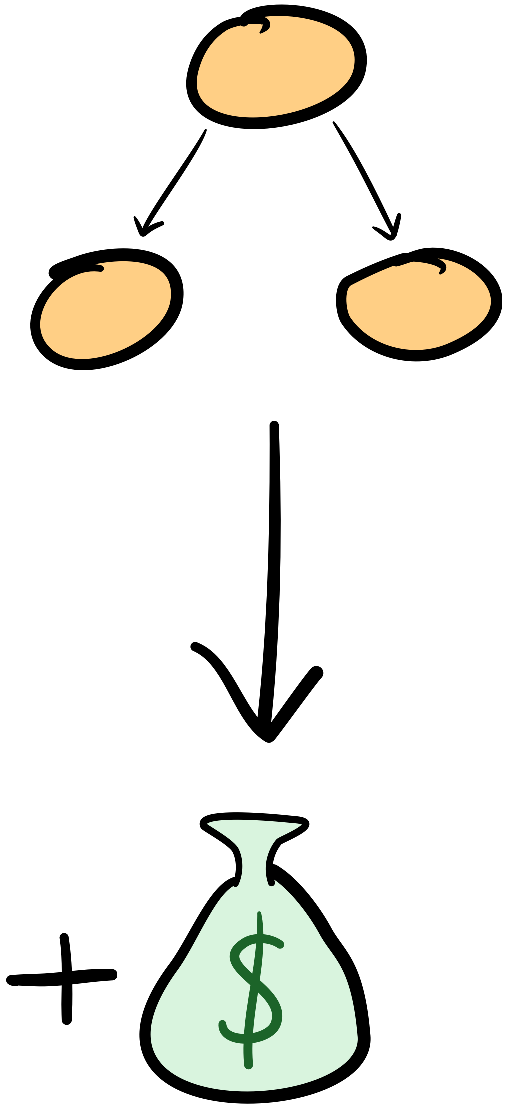
$G_{J \triangleright K} = ???$

Single-Job Profit



Game with a job and potential **reward**

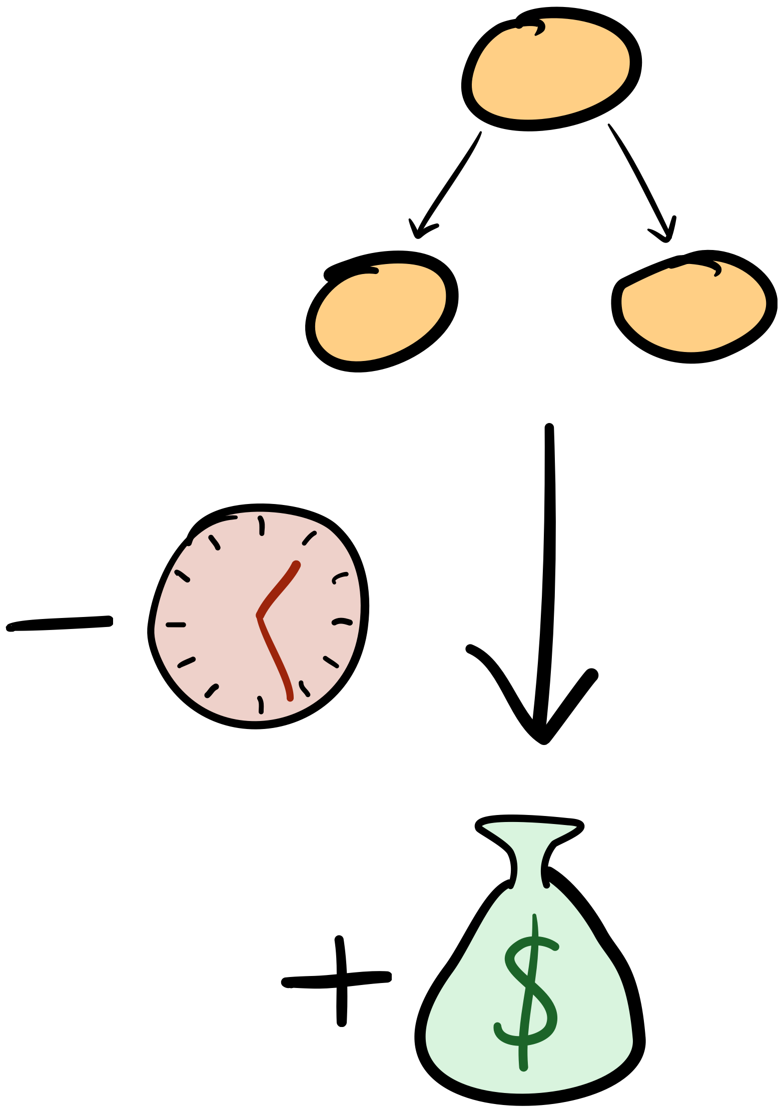
Single-Job Profit



Game with a job and potential **reward**

- Get **reward** if we complete the job

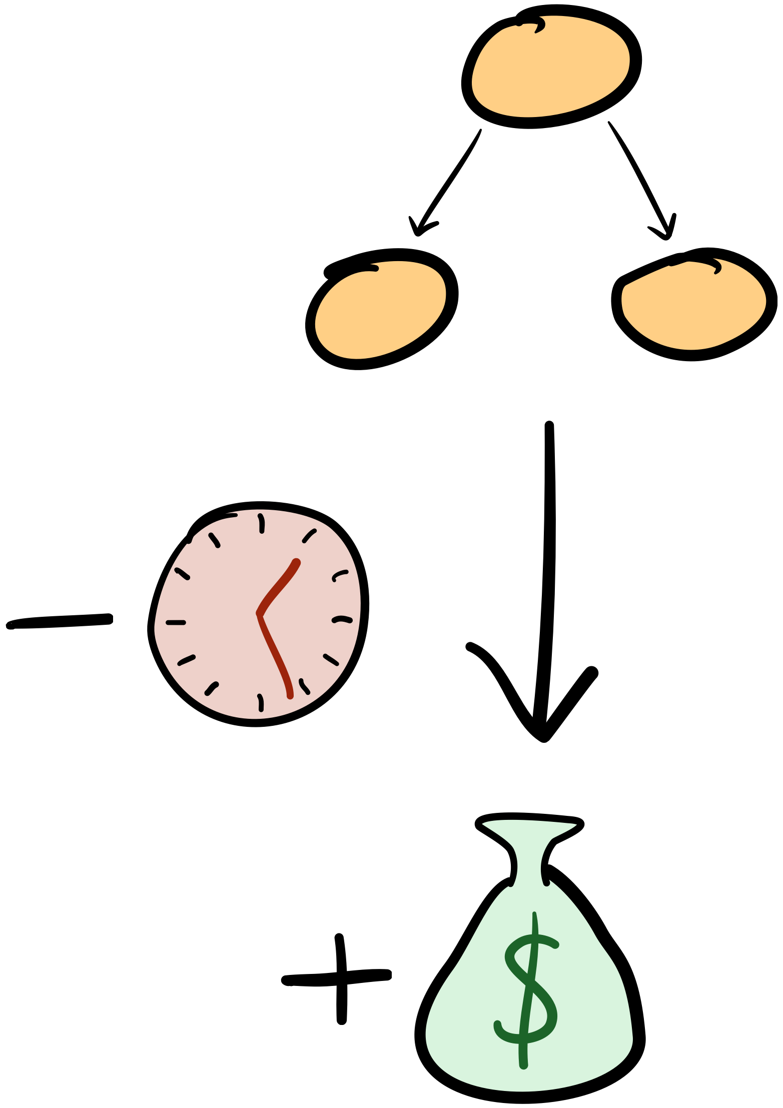
Single-Job Profit



Game with a job and potential **reward**

- Get **reward** if we complete the job
- Pay for **time** spent serving the job

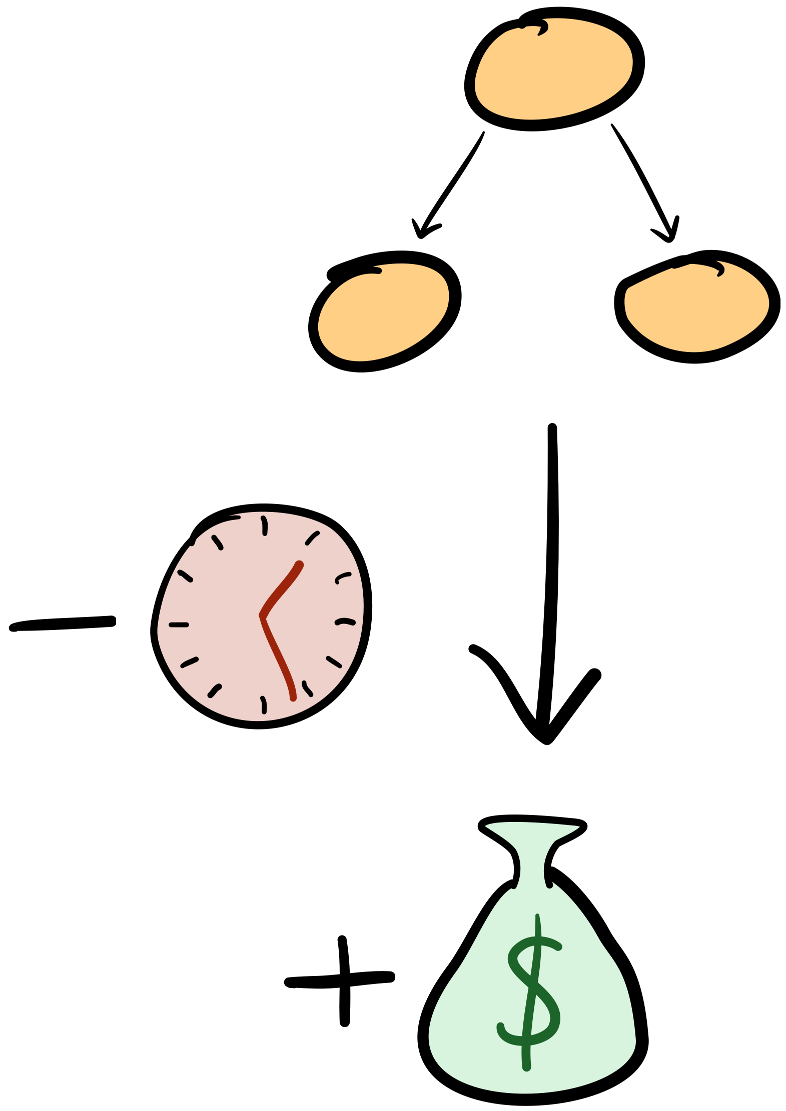
Single-Job Profit



Game with a job and potential **reward**

- Get **reward** if we complete the job
- Pay for **time** spent serving the job
- Can *give up* at any time

Single-Job Profit



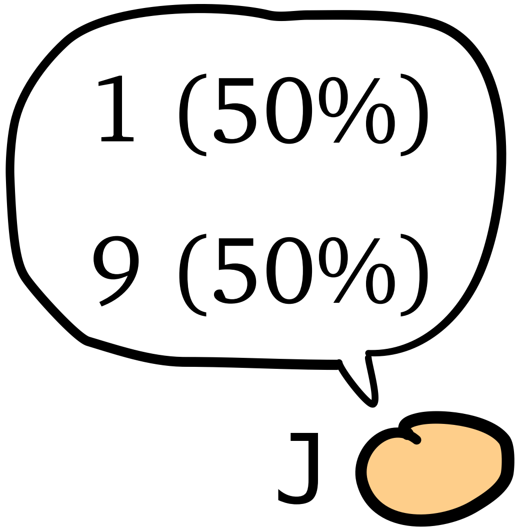
Game with a job and potential **reward**

- Get **reward** if we complete the job
- Pay for **time** spent serving the job
- Can *give up* at any time

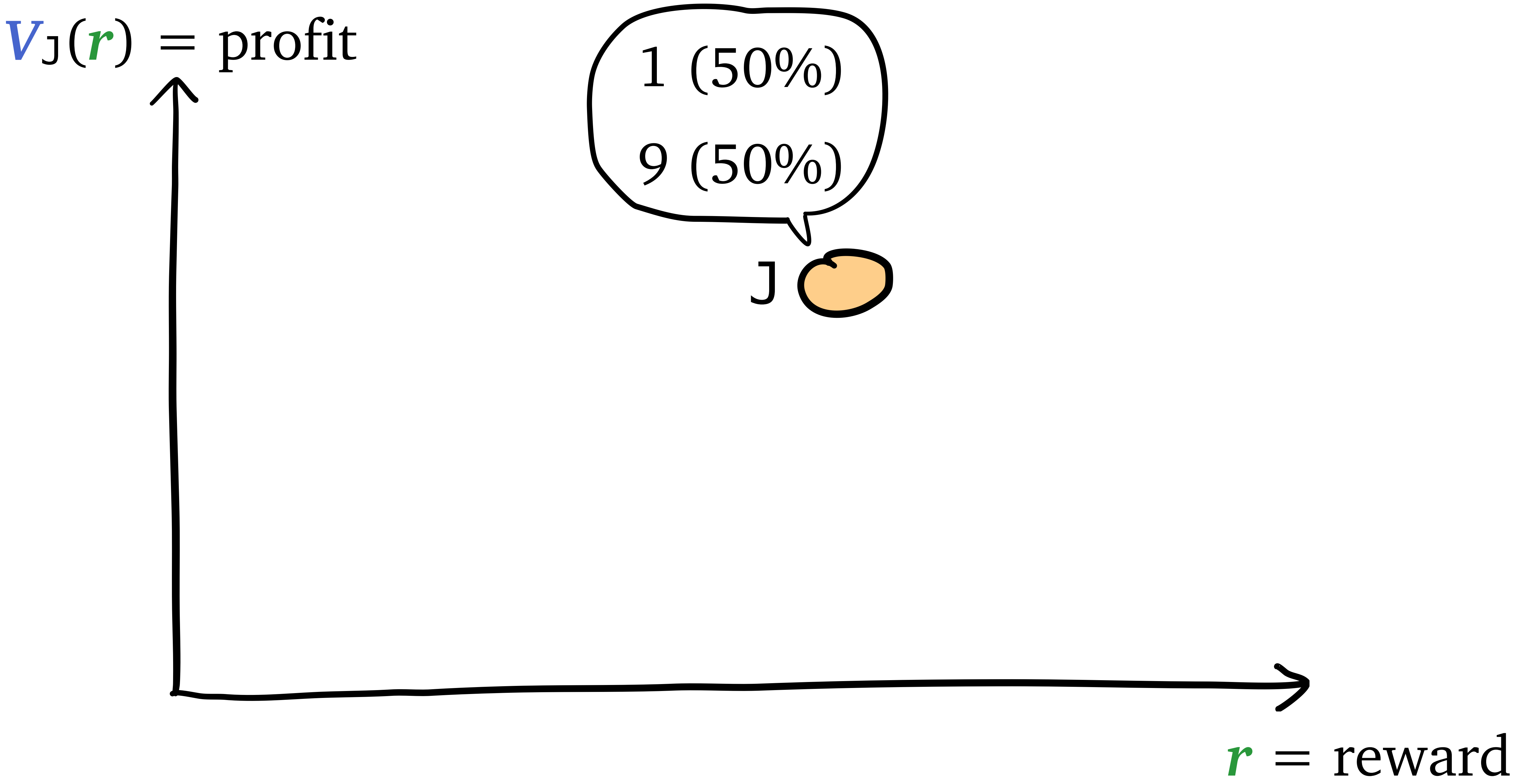
Goal: maximize **profit**:

$\mathbf{E}$[**reward** recieved – **time** spent]

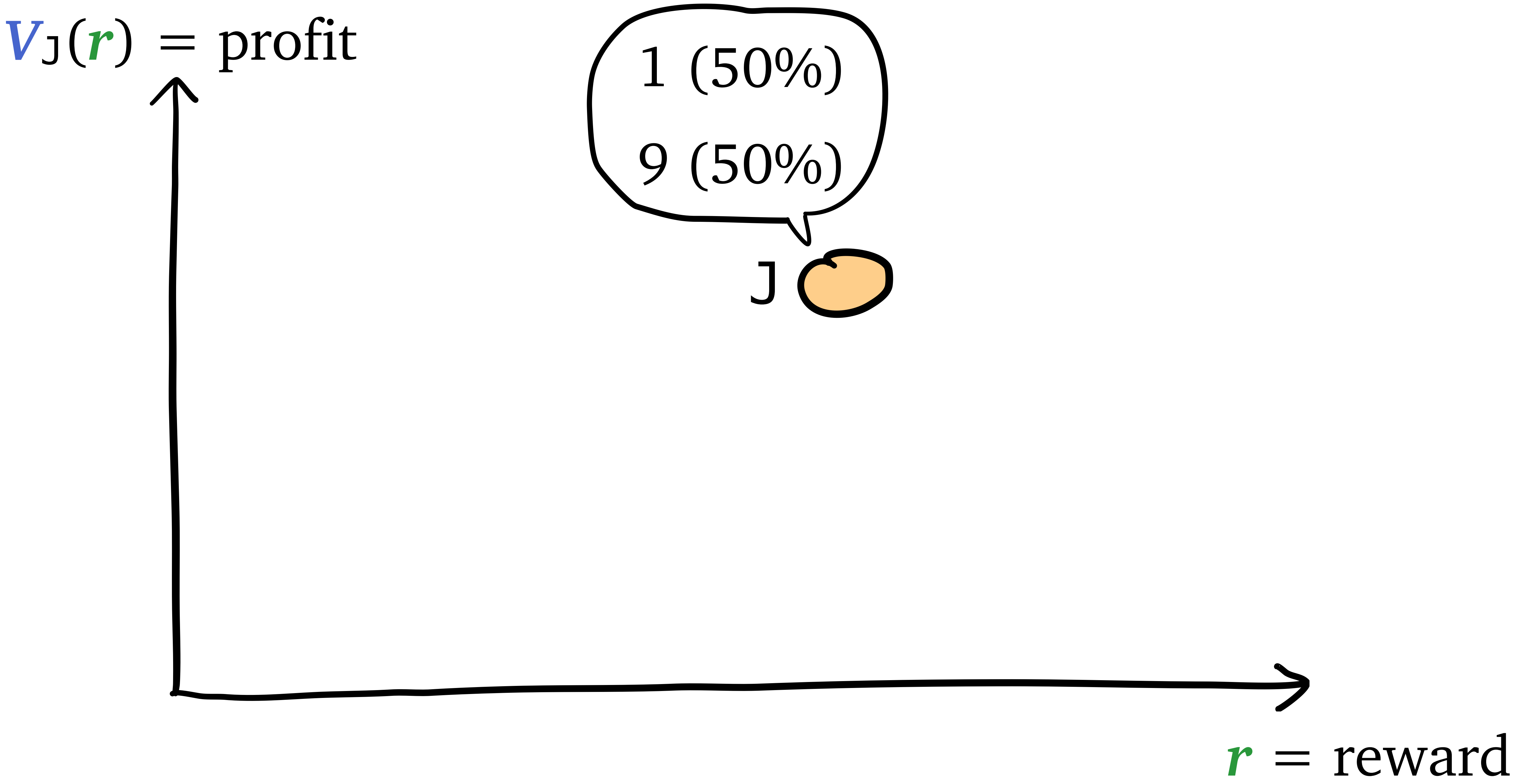
Single-Job Profit Example



Single-Job Profit Example



Single-Job Profit Example



$V_J(r)$ = profit

1 (50%)

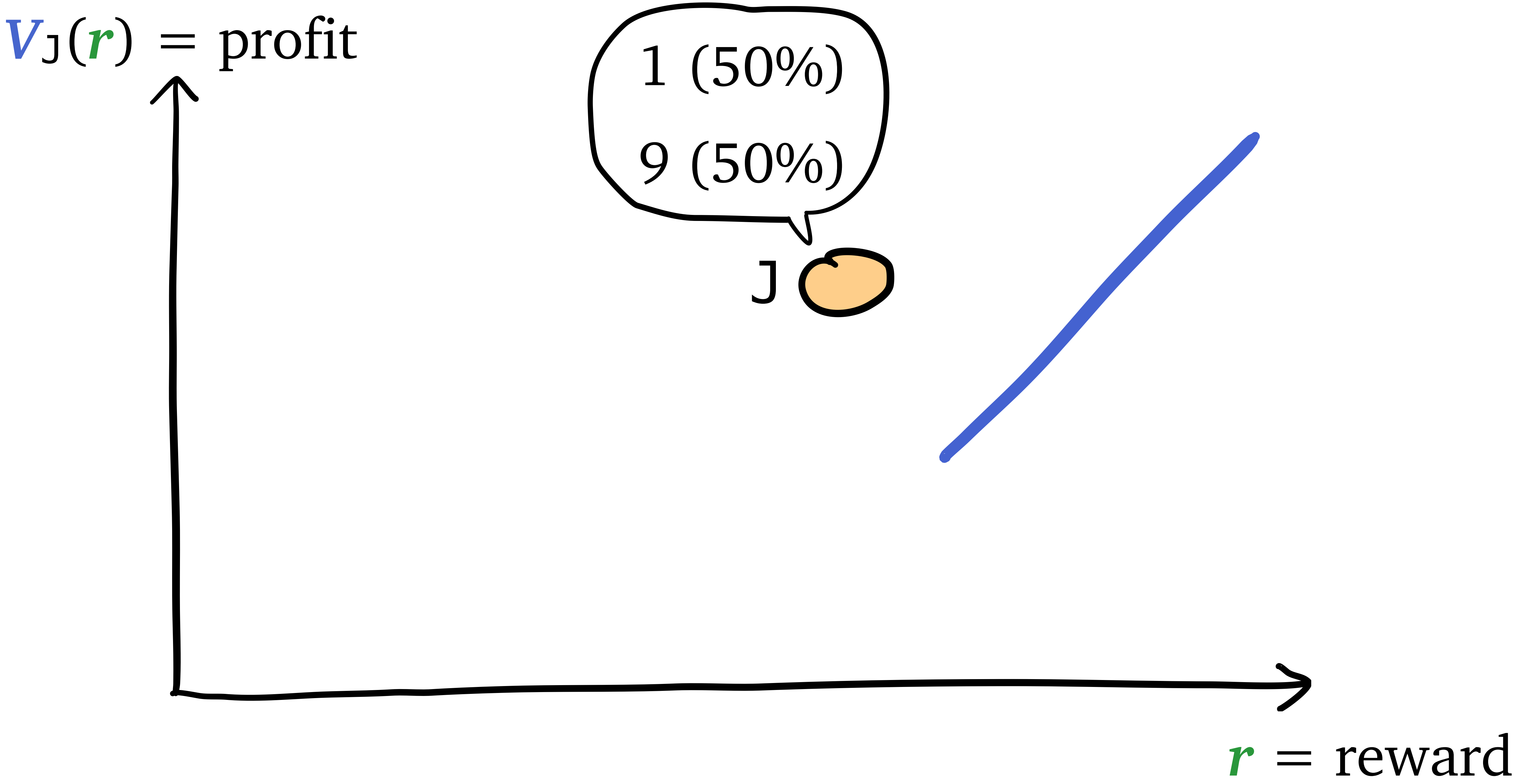
9 (50%)

J

r = reward

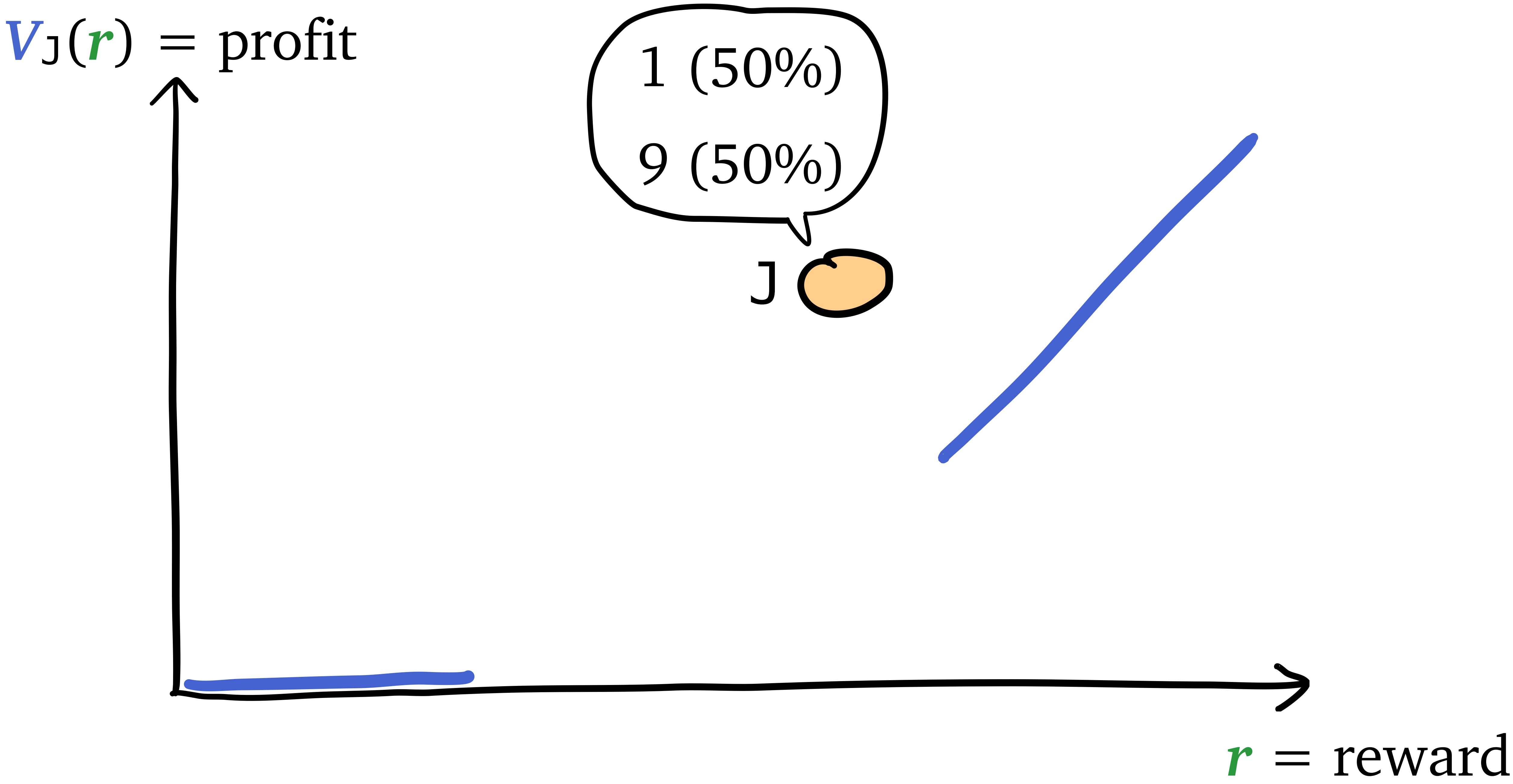
Definition: V_J is the *SJP function* of J

Single-Job Profit Example



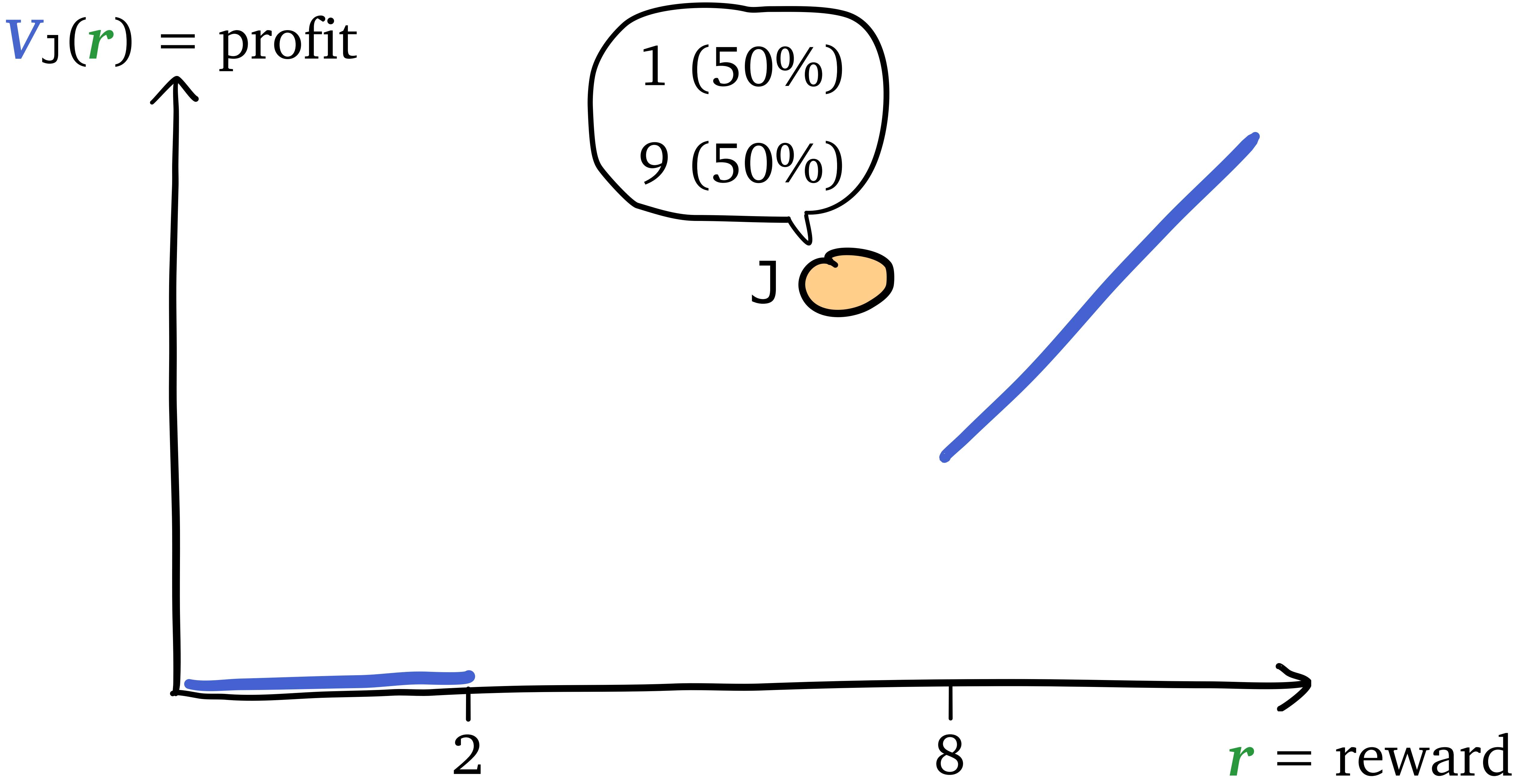
Definition: V_J is the *SJP function* of J

Single-Job Profit Example



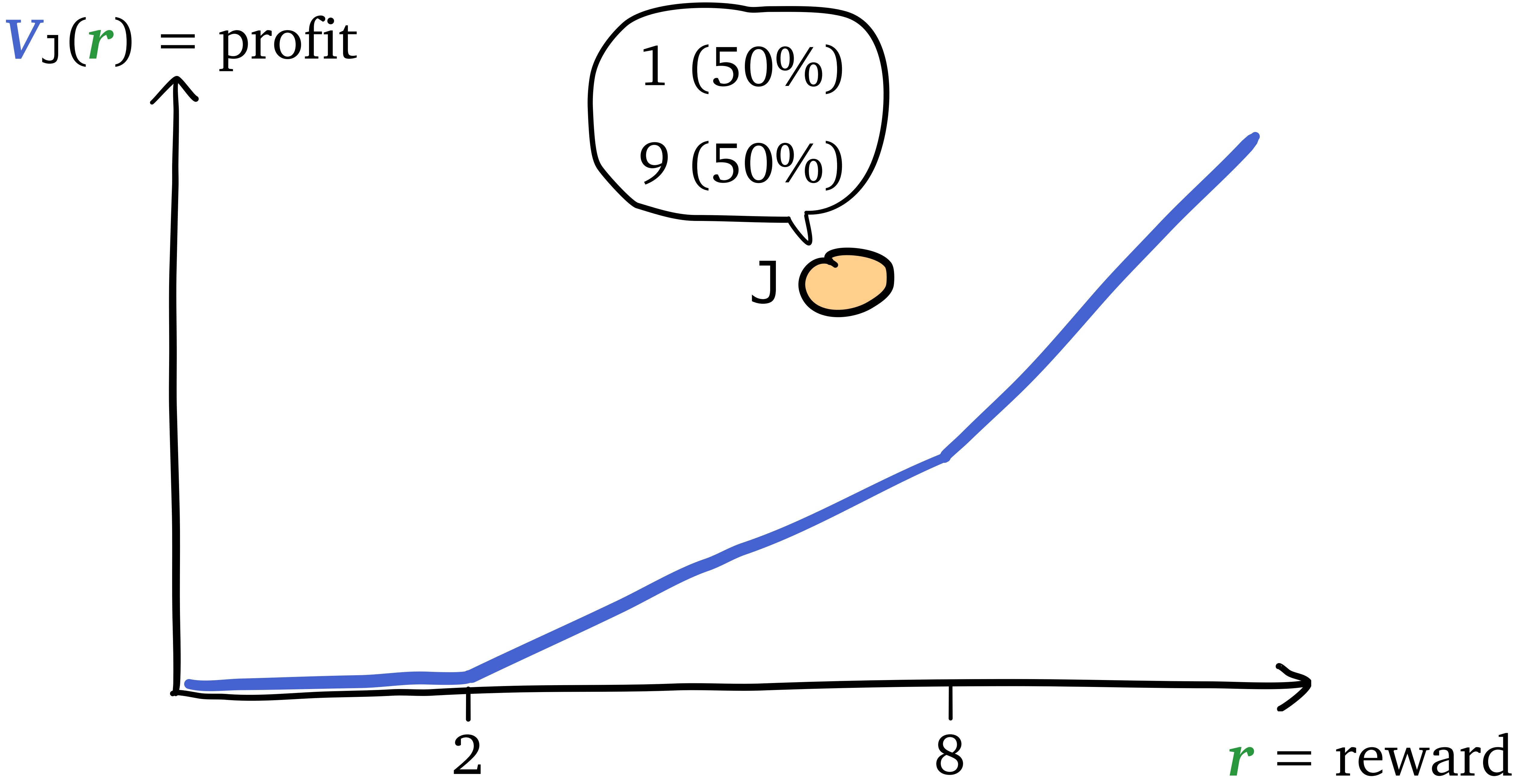
Definition: V_J is the *SJP function* of J

Single-Job Profit Example



Definition: V_J is the *SJP function* of J

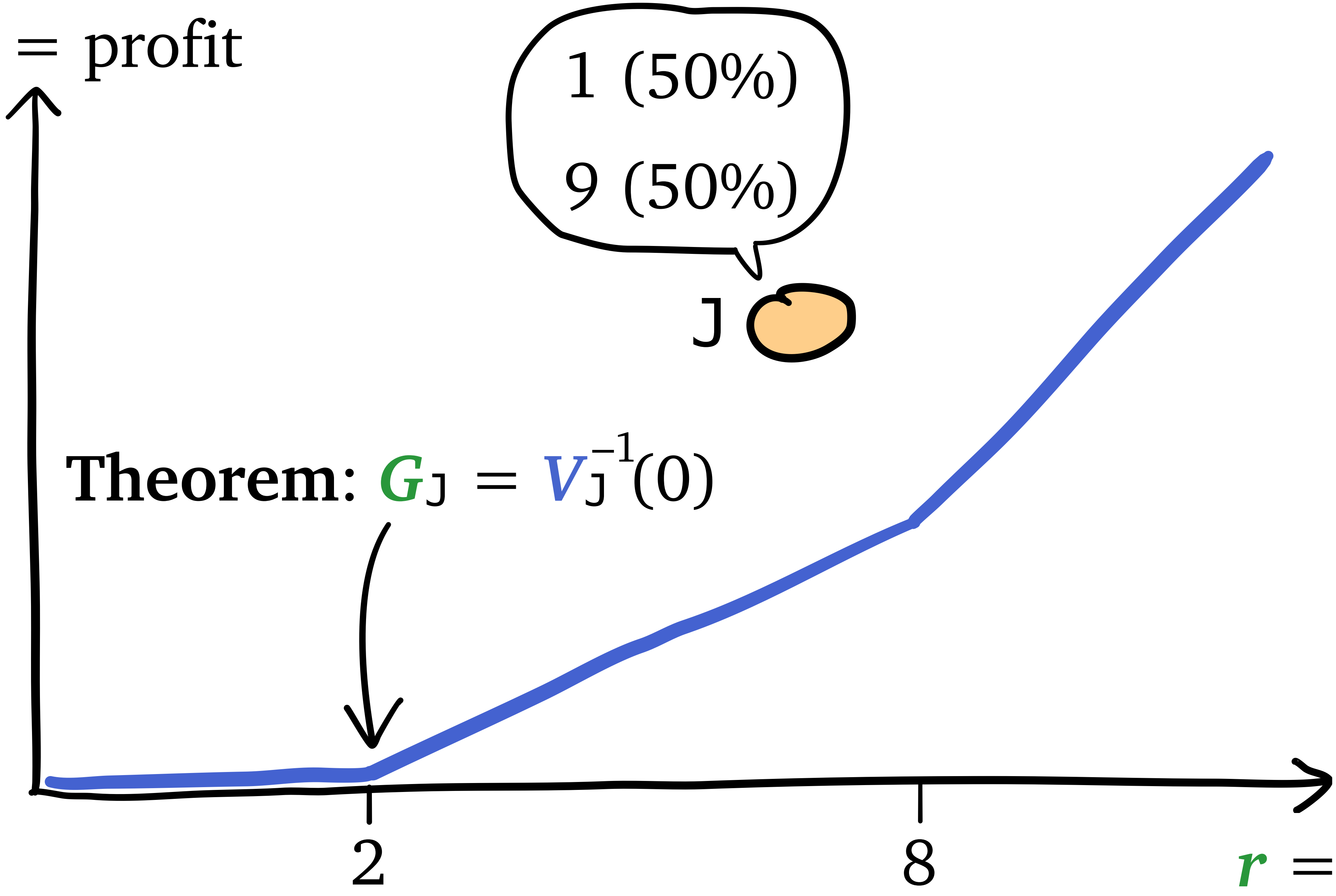
Single-Job Profit Example



Definition: V_{J} is the *SJP function* of J

Single-Job Profit Example

$V_J(r)$ = profit



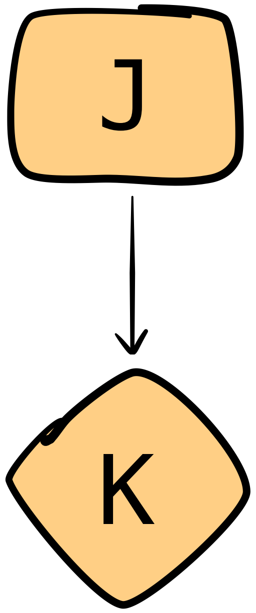
Theorem: $G_J = V_J^{-1}(0)$

r = reward

Definition: V_J is the *SJP function* of J

SJP Composition Law

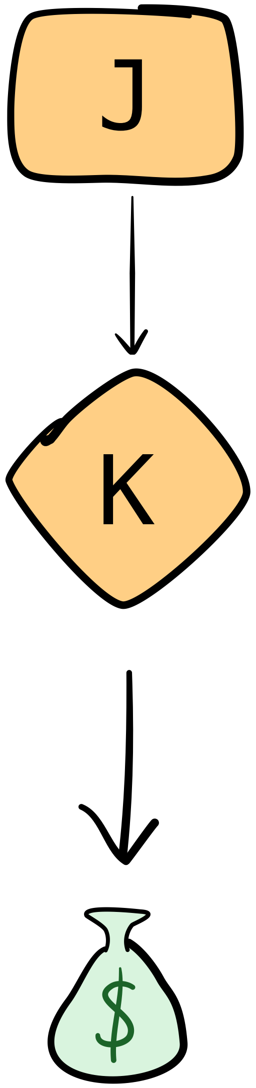
Theorem: $V_{\mathsf{J} \triangleright \mathsf{K}}(r) = V_{\mathsf{J}}(V_{\mathsf{K}}(r))$



SJP Composition Law

Theorem: $V_{\mathsf{J} \triangleright \mathsf{K}}(r) = V_{\mathsf{J}}(V_{\mathsf{K}}(r))$

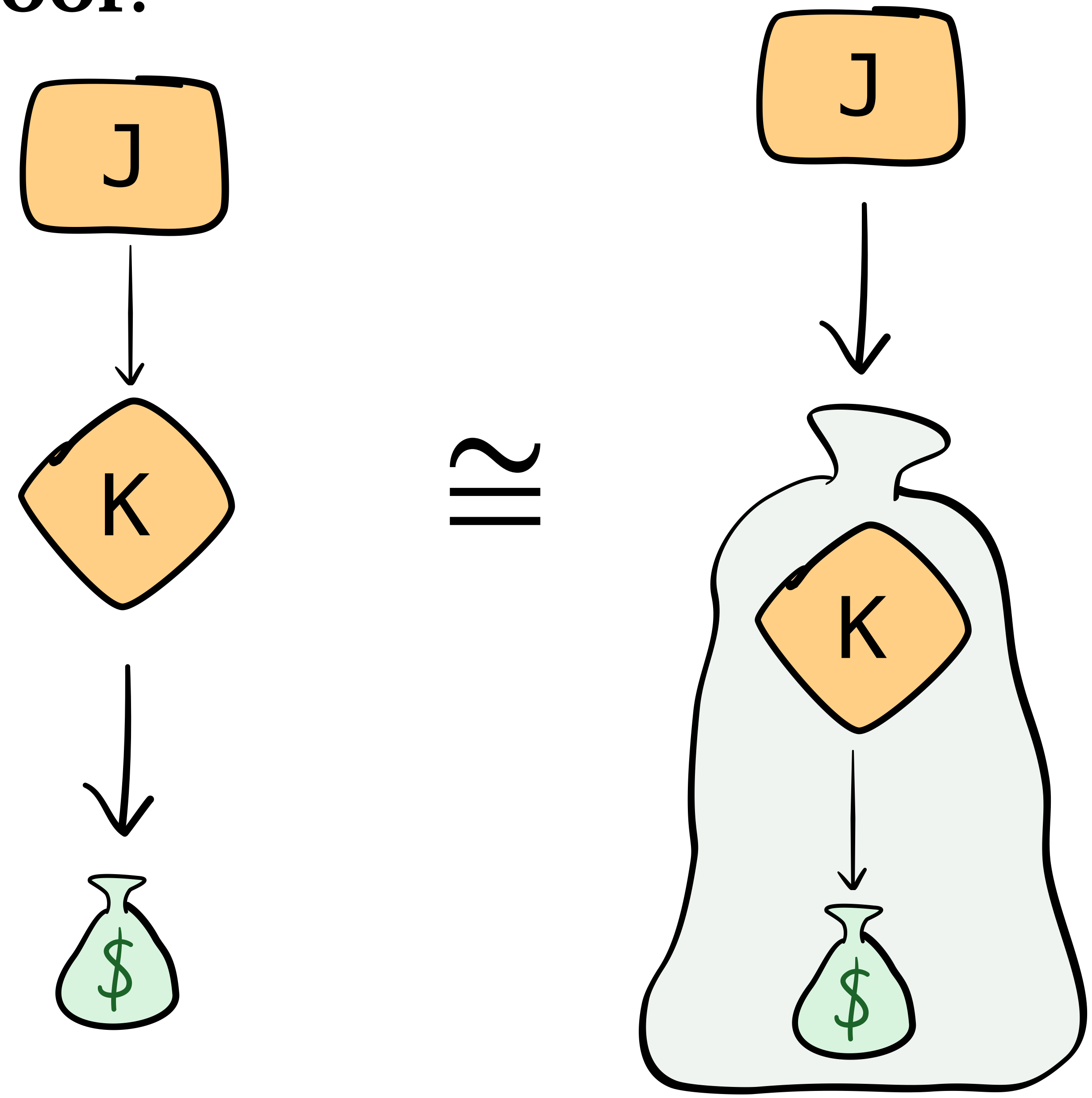
Proof:



SJP Composition Law

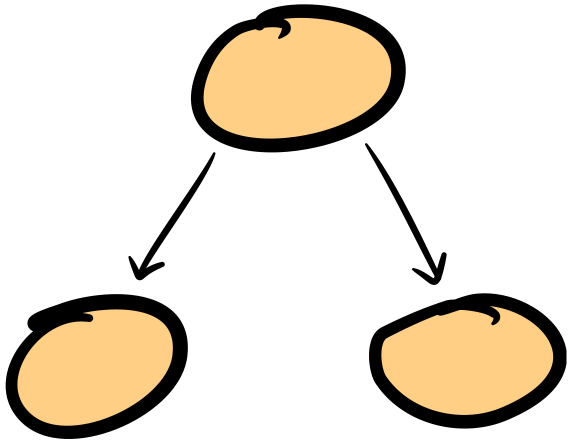
Theorem: $V_{\mathsf{J} \triangleright \mathsf{K}}(r) = V_{\mathsf{J}}(V_{\mathsf{K}}(r))$

Proof:

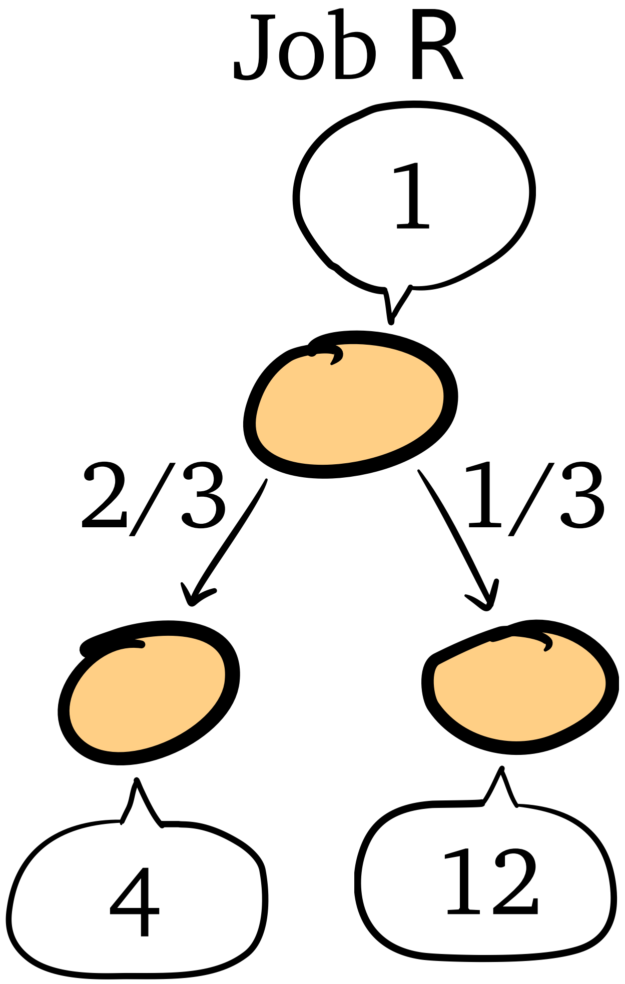


Response Time Impact

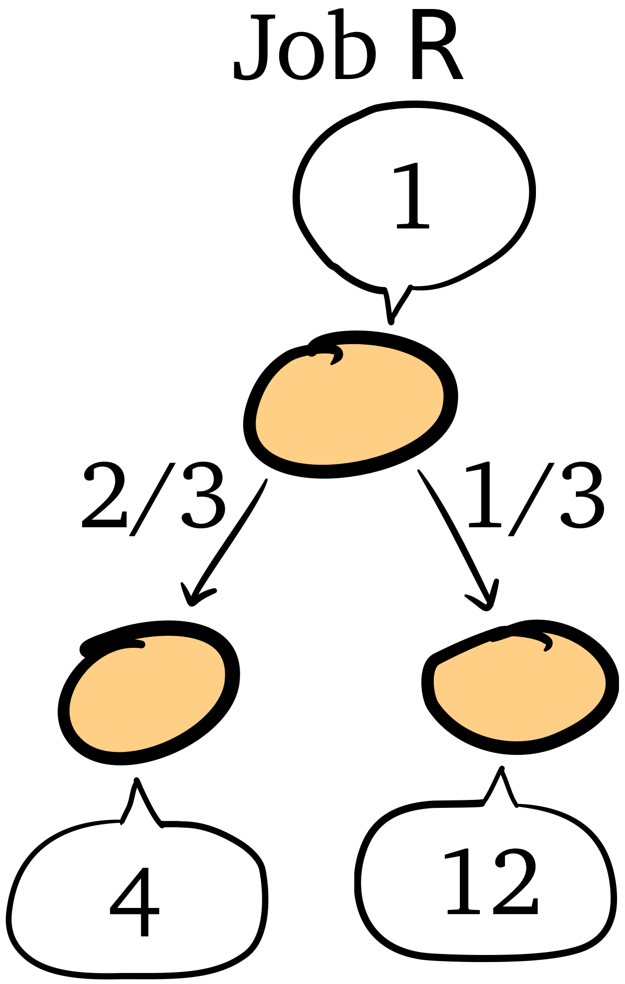
Job R



Response Time Impact

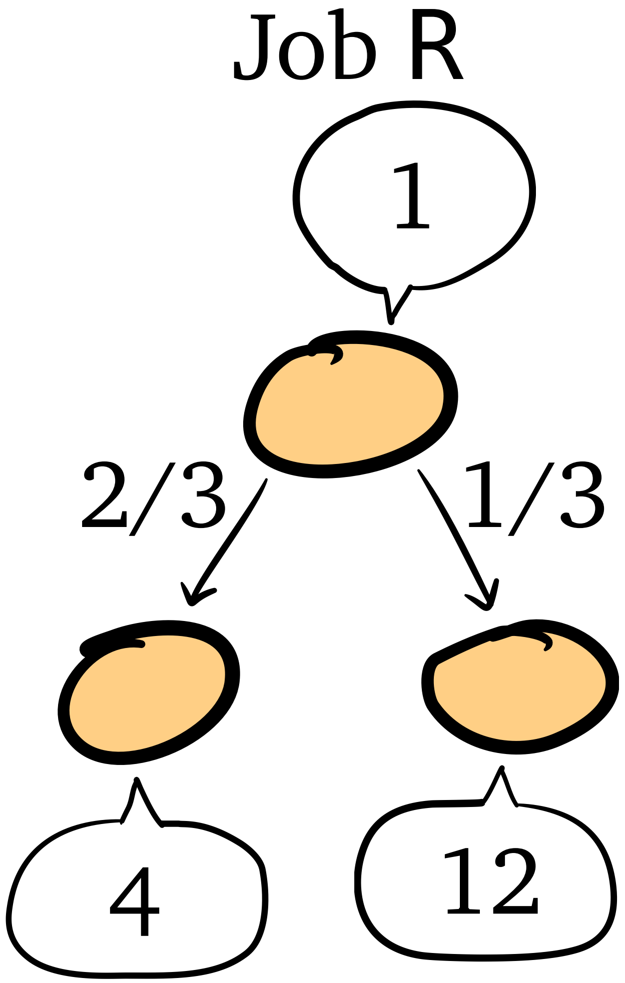


Response Time Impact



Compare three policies:

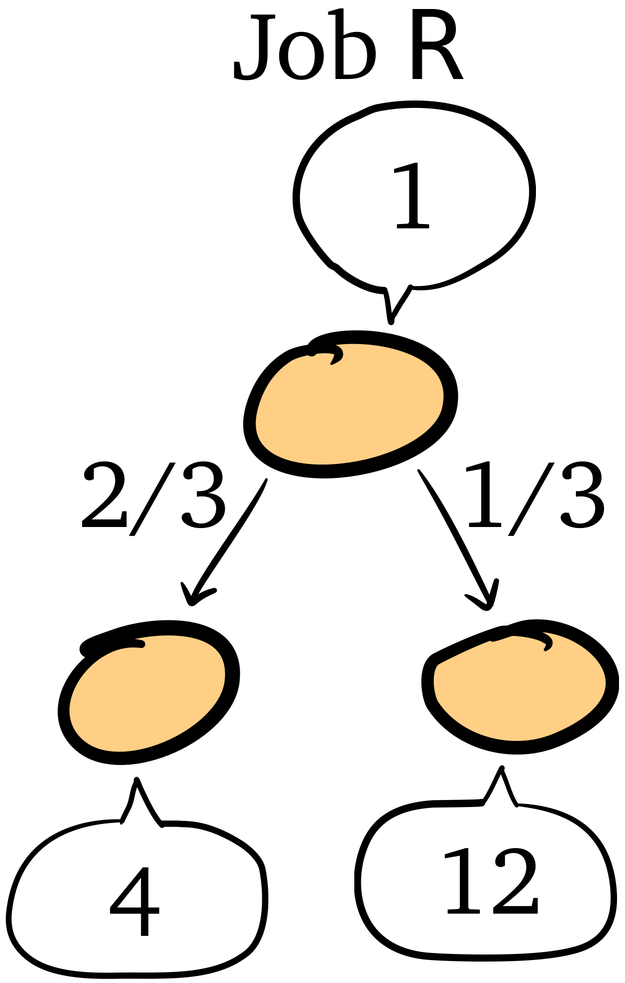
Response Time Impact



Compare three policies:

- *First-come, first-served* (FCFS)

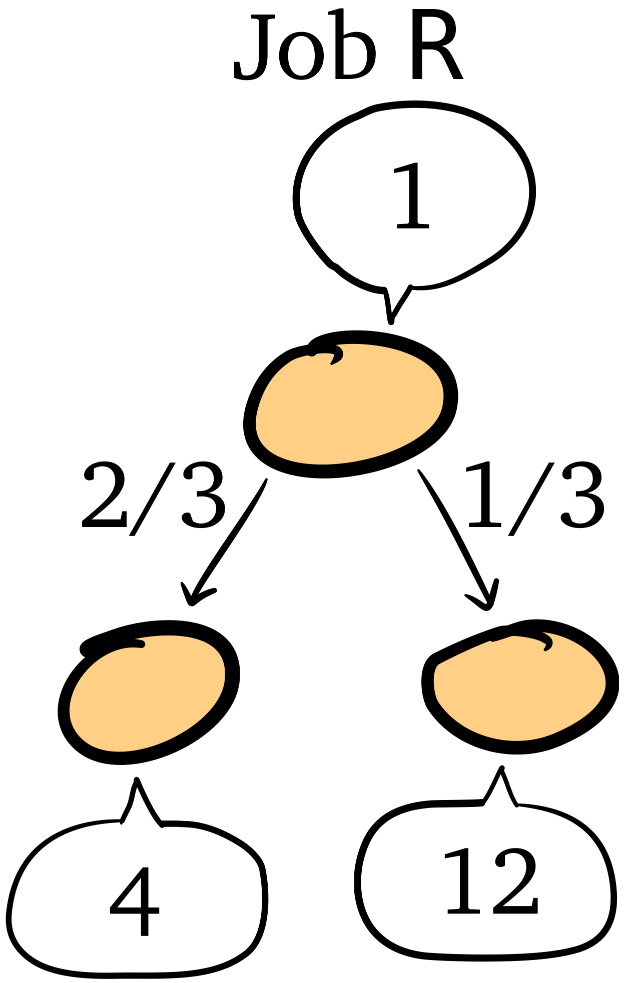
Response Time Impact



Compare three policies:

- *First-come, first-served* (FCFS)
- *Blind Gittins policy* (BGP): ignores stage information

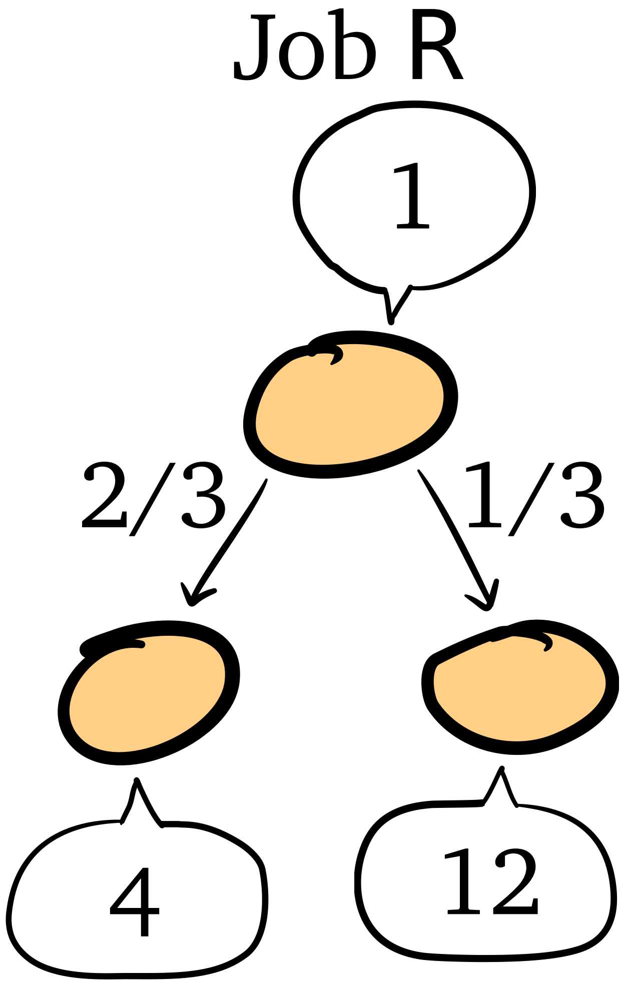
Response Time Impact

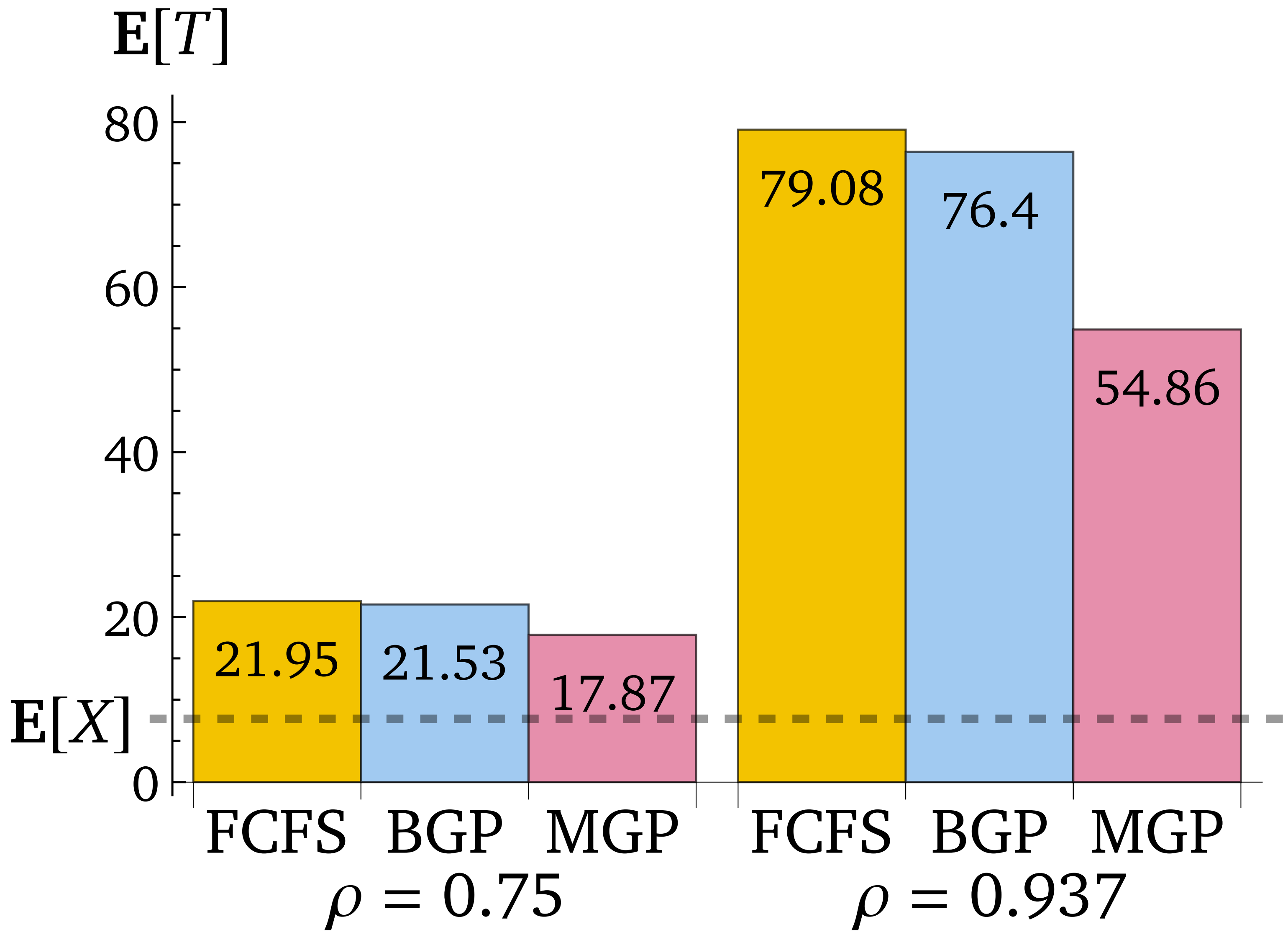


Compare three policies:

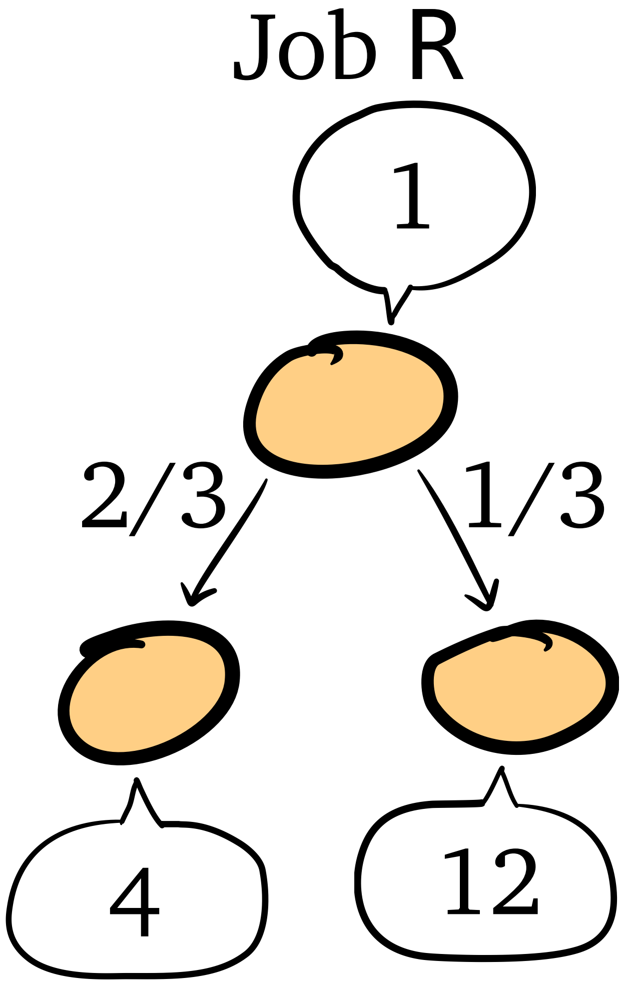
- *First-come, first-served* (FCFS)
- *Blind Gittins policy* (BGP): ignores stage information
- *Multistage Gittins policy* (MGP): exploits stage information

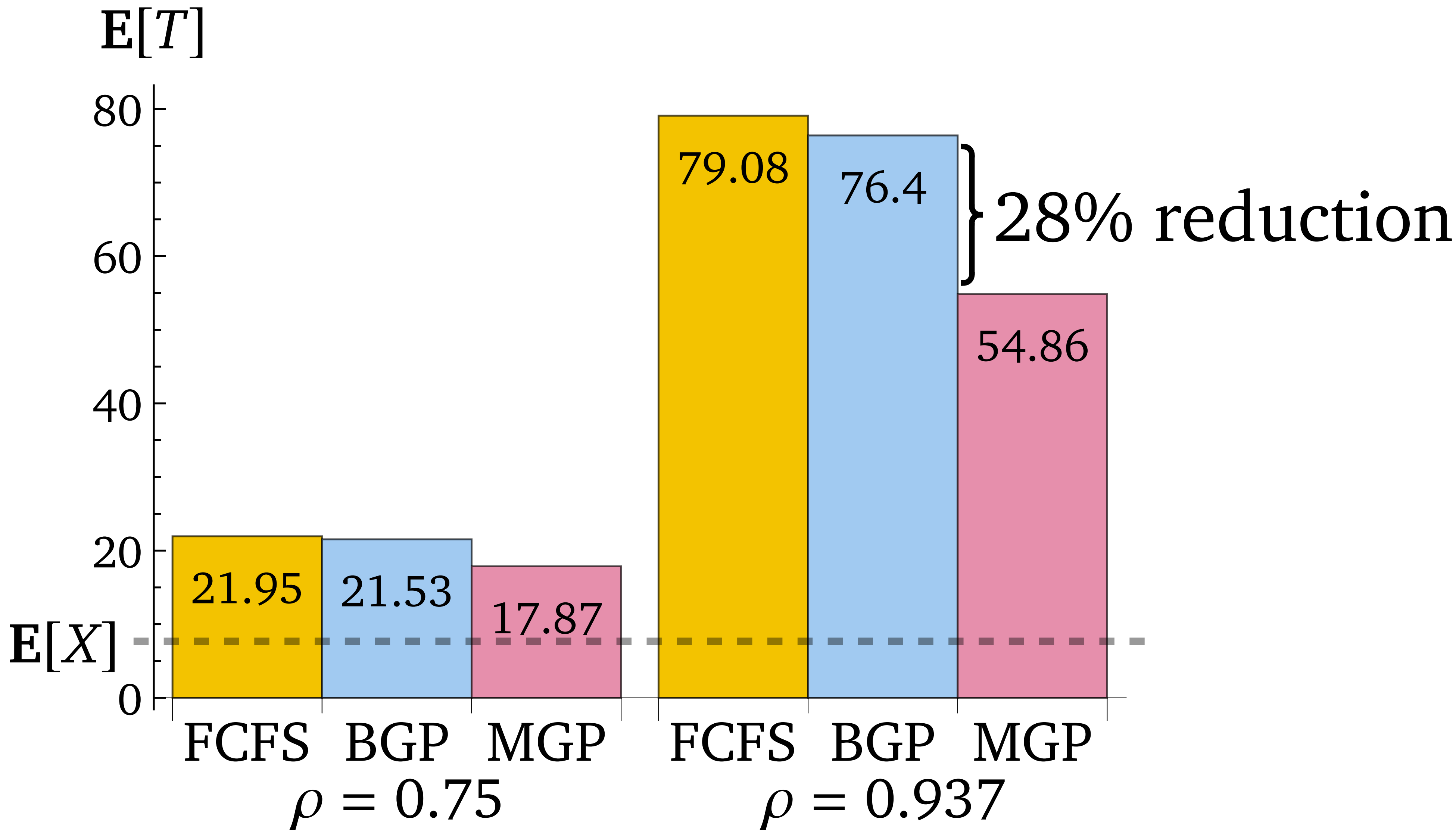
Response Time Impact

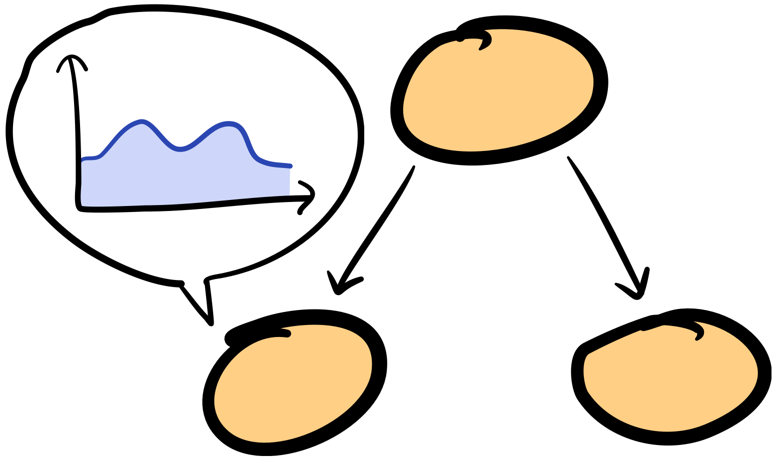




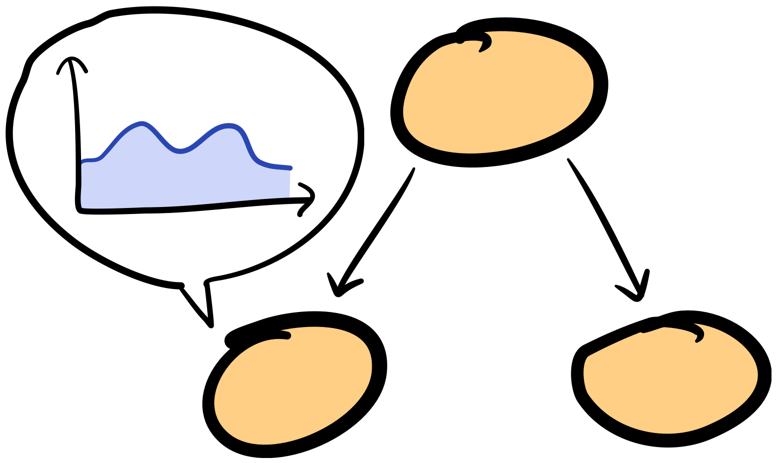
Response Time Impact



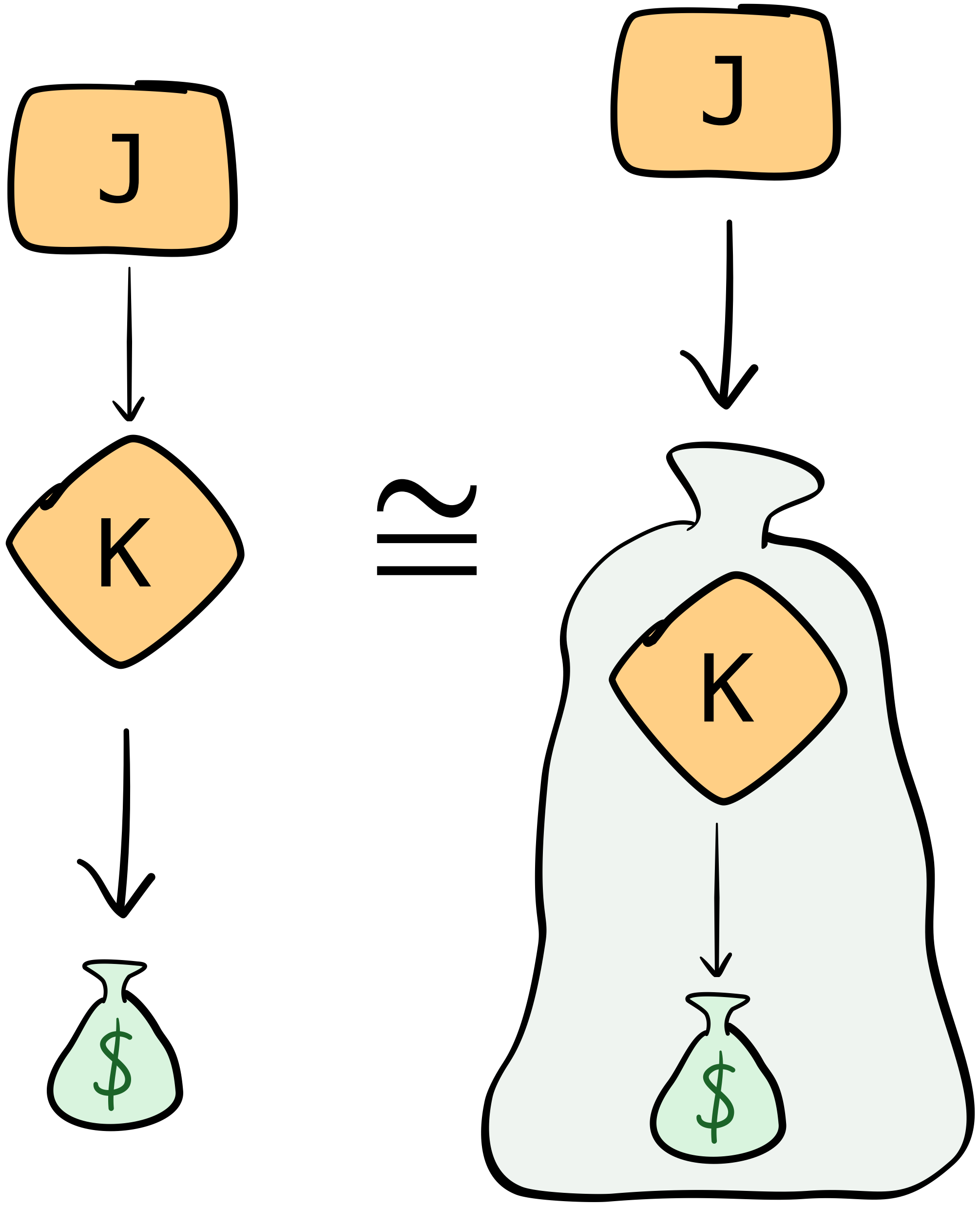




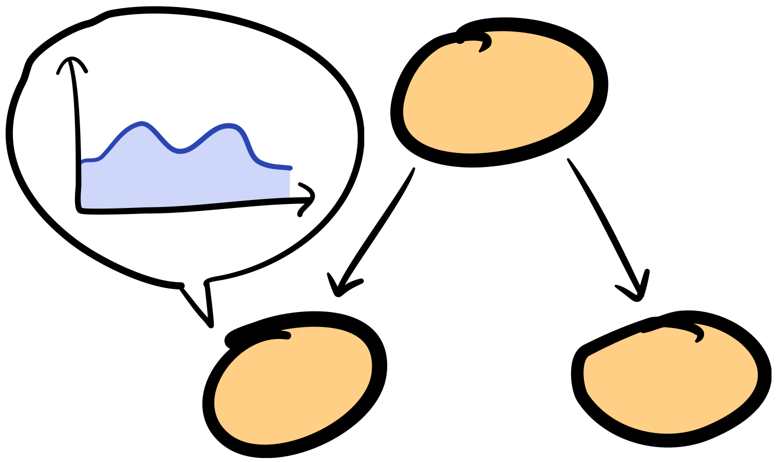
Problem: Gittins policy for *multistage jobs*



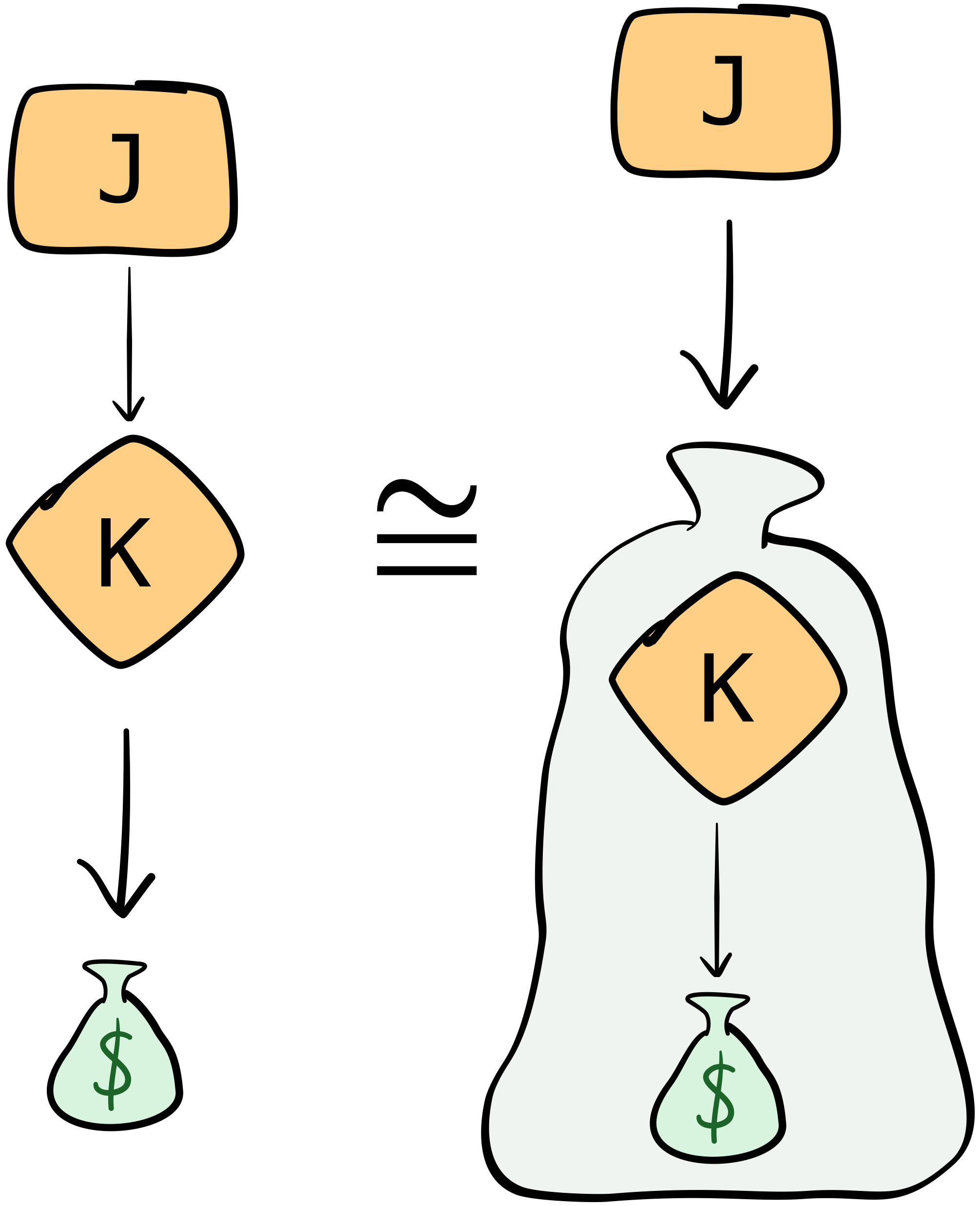
Problem: Gittins policy for *multistage jobs*



Solution: new *single-job profit* (SJP) approach

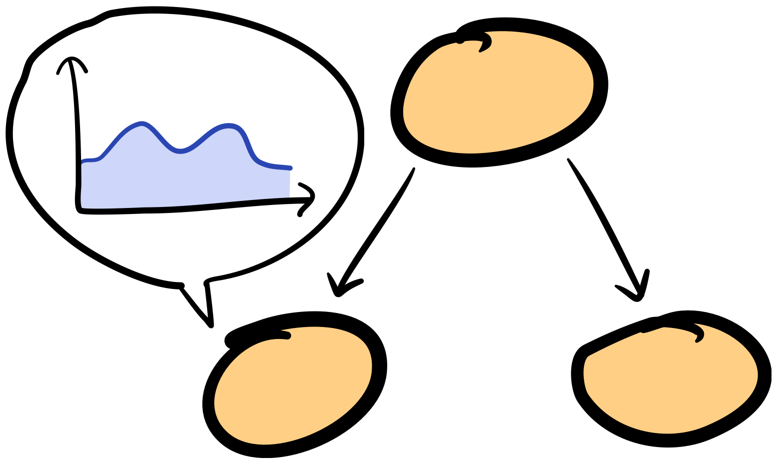


Problem: Gittins policy for *multistage jobs*

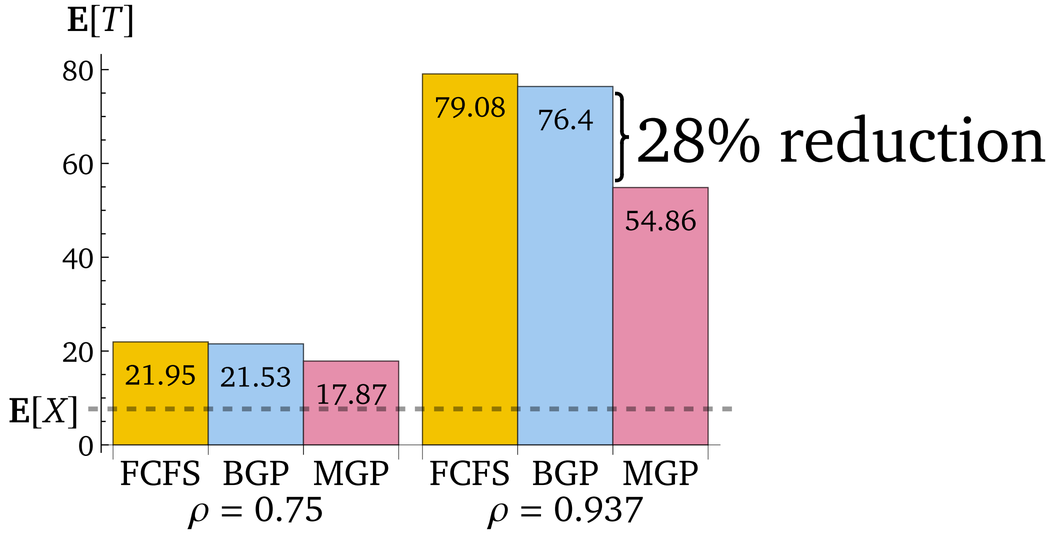


Solution: new *single-job profit* (SJP) approach

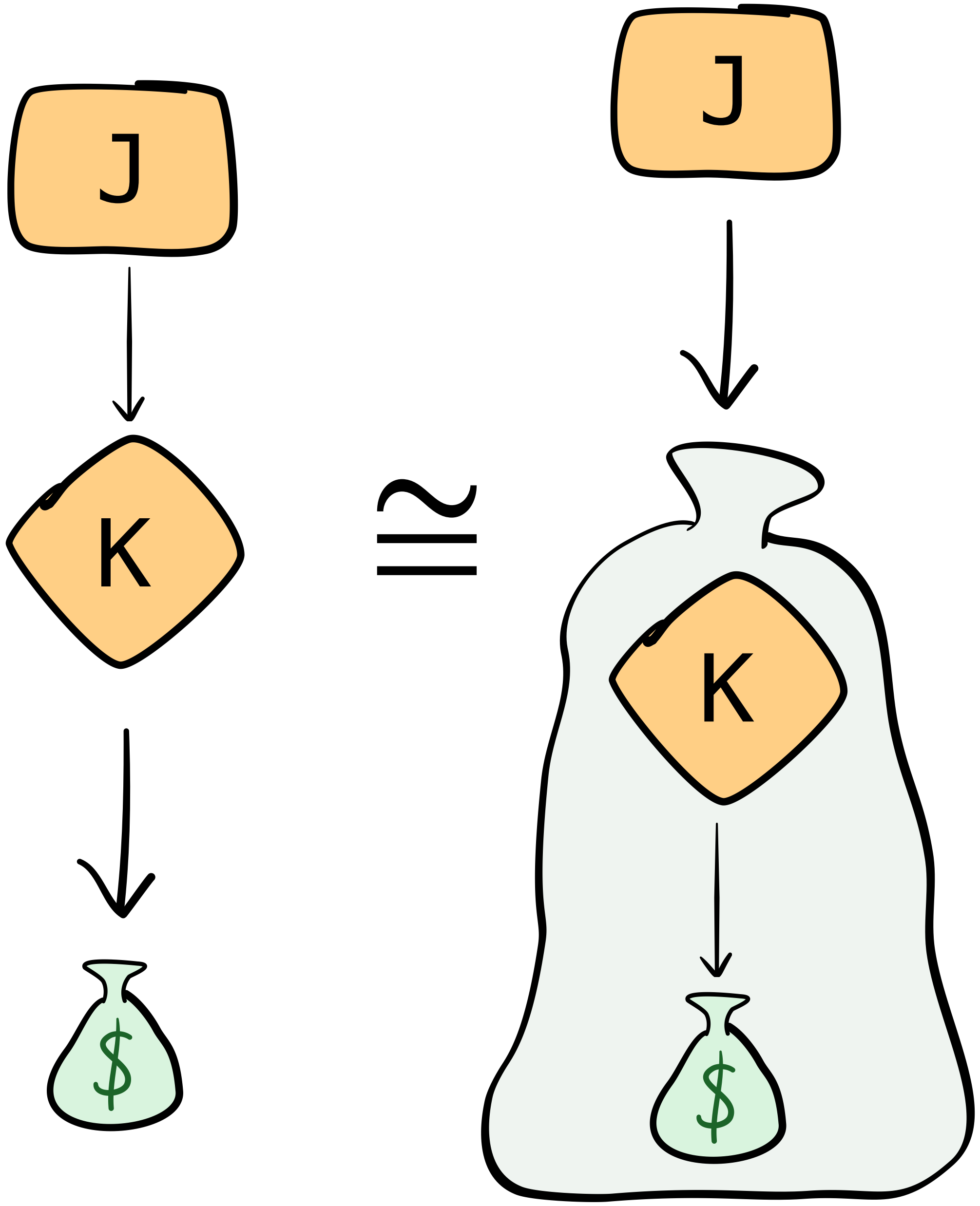
- SJP composition law



Problem: Gittins policy for *multistage jobs*



Impact: *significantly reduces* $\mathbf{E}[T]$



Solution: new *single-job profit* (SJP) approach

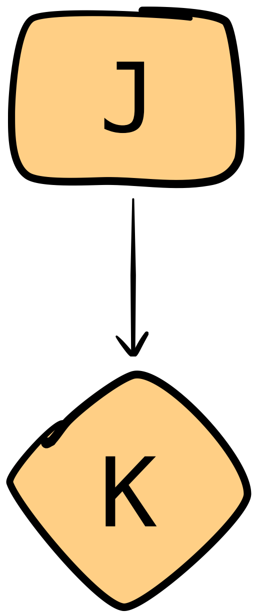
- SJP composition law

Two Building Blocks

Two Building Blocks

Sequential Composition

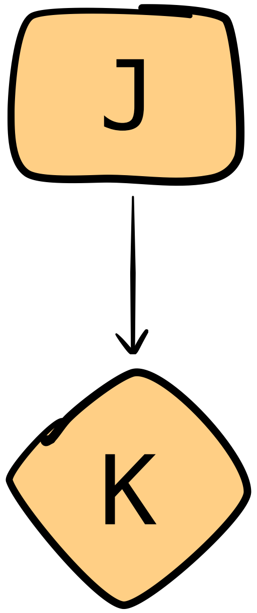
J▷K



Two Building Blocks

Sequential Composition

J▷K

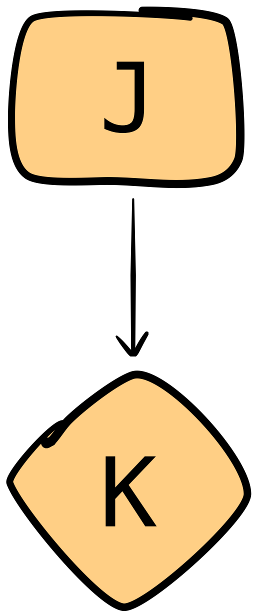


$$V_{\text{J}\triangleright\text{K}}(r) = V_{\text{J}}(V_{\text{K}}(r))$$

Two Building Blocks

Sequential Composition

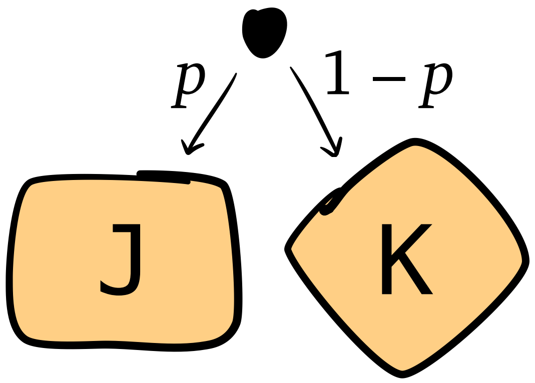
$\mathsf{J} \triangleright \mathsf{K}$



$$V_{\mathsf{J}\triangleright\mathsf{K}}(r) = V_{\mathsf{J}}(V_{\mathsf{K}}(r))$$

Mixture Composition

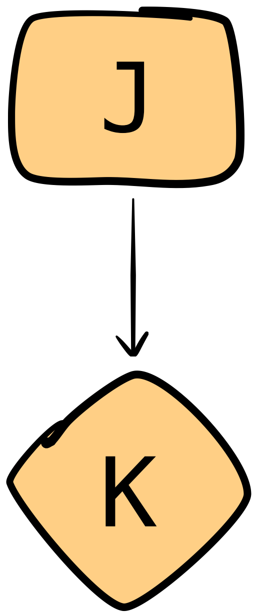
$\mathsf{J} \mid \mathsf{K}$



Two Building Blocks

Sequential Composition

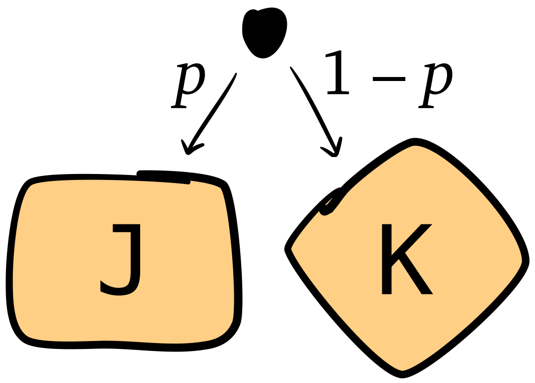
$\mathsf{J} \triangleright \mathsf{K}$



$$V_{\mathsf{J}\triangleright\mathsf{K}}(r) = V_{\mathsf{J}}(V_{\mathsf{K}}(r))$$

Mixture Composition

$\mathsf{J} \,|\, \mathsf{K}$

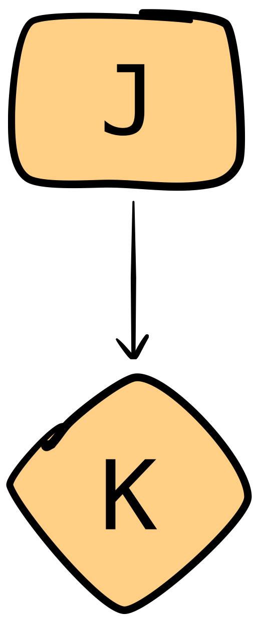


$$V_{\mathsf{J}|\mathsf{K}}(r) = pV_{\mathsf{J}}(r) + (1 - \mathrm{p})V_{\mathsf{K}}(r)$$

Two Building Blocks

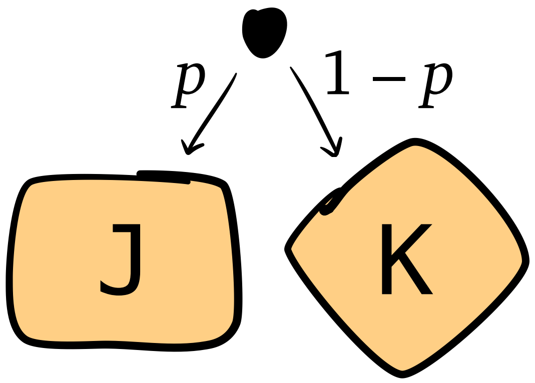
Sequential Composition

J▷K



Mixture Composition

J | K



$V_{J \triangleright K}(r) = V_J(V_K(r))$ $V_{J|K}(r) = pV_J(r) + (1 - \mathrm{p})V_K(r)$

Every multistage job can be built from these