

# Universal Scheduling

Julián Mestre (University of Sydney)

Joint work with:

L. Epstein (Haifa University)

A. Levin (Technion)

N. Megow (MPI-INF)

A. Marchetti-Spaccamela (La Sapienza)

M. Skutella (TU Berlin)

L. Stougie (TU Eindhoven)



THE UNIVERSITY OF  
SYDNEY



# Universal Scheduling

Traditional scheduling assumes **ideal machines**.

Traditional scheduling assumes **ideal machines**.



Traditional scheduling assumes **ideal machines**.



Traditional scheduling assumes **ideal machines**.

In practice though machines may **vary speed** unpredictably or may **break down** all together.

Traditional scheduling assumes **ideal machines**.

In practice though machines may **vary speed** unpredictably or may **break down** all together.

We want scheduling policies **robust against uncertainty**:

Traditional scheduling assumes **ideal machines**.

In practice though machines may **vary speed** unpredictably or may **break down** all together.

We want scheduling policies **robust against uncertainty**:

- Machine behavior is unpredictable

Traditional scheduling assumes **ideal machines**.

In practice though machines may **vary speed** unpredictably or may **break down** all together.

We want scheduling policies **robust against uncertainty**:

- Machine behavior is unpredictable
- We must commit to a schedule in advance

Traditional scheduling assumes **ideal machines**.

In practice though machines may **vary speed** unpredictably or may **break down** all together.

We want scheduling policies **robust against uncertainty**:

- Machine behavior is unpredictable
- We must commit to a schedule in advance
- Compare against best schedule for observed machine behavior

# Scheduling on a single machine

Jobs have **weight** and **processing time**

Objective: minimize **weighted average completion time**

# Scheduling on a single machine

Jobs have **weight** and **processing time**

Objective: minimize **weighted average completion time**

Smith's rule: sort by **processing time/weight**

# Scheduling on a single machine

Jobs have **weight** and **processing time**

Objective: minimize **weighted average completion time**

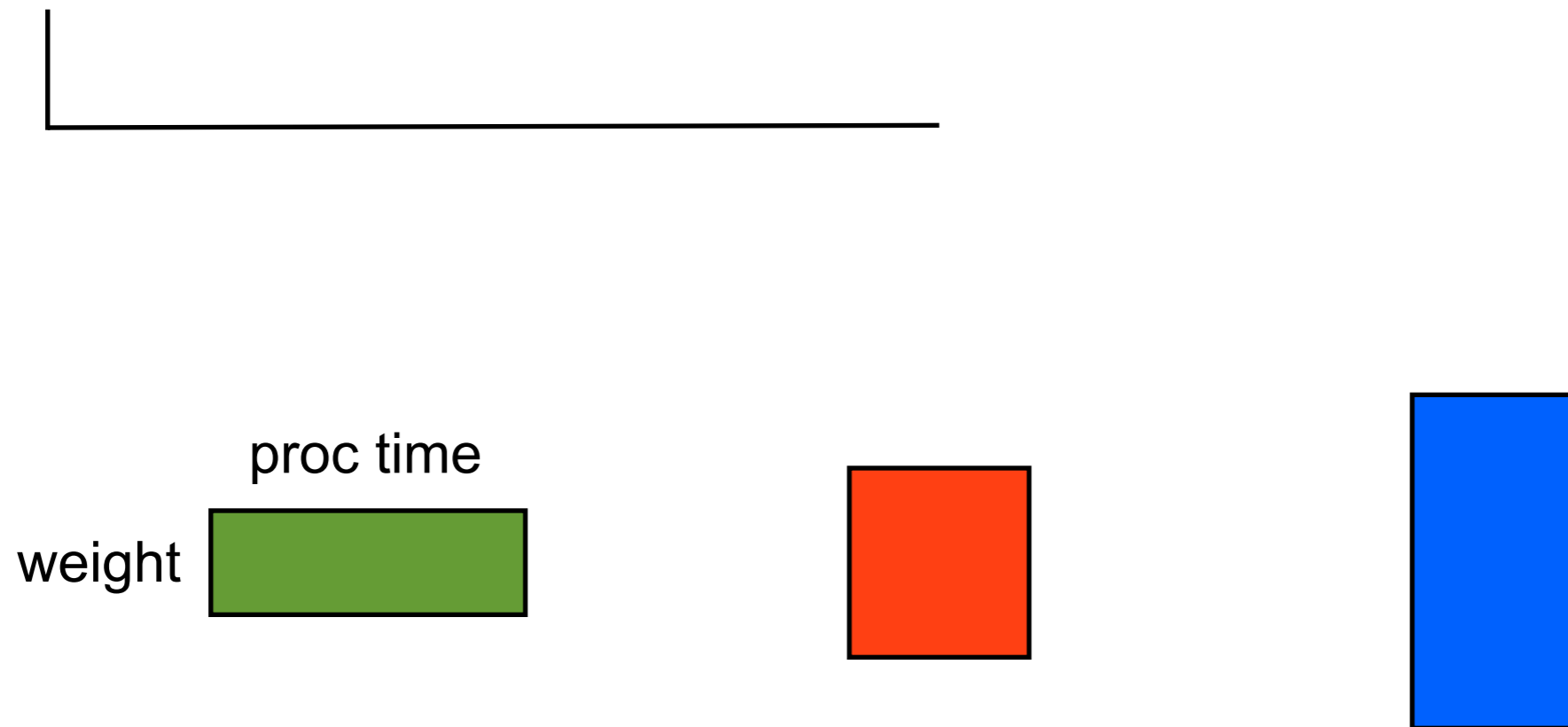
Smith's rule: sort by **processing time/weight**

However, the “optimal solution” for an ideal machine can be **arbitrarily bad** on an unreliable machine

# A pictorial view of the objective



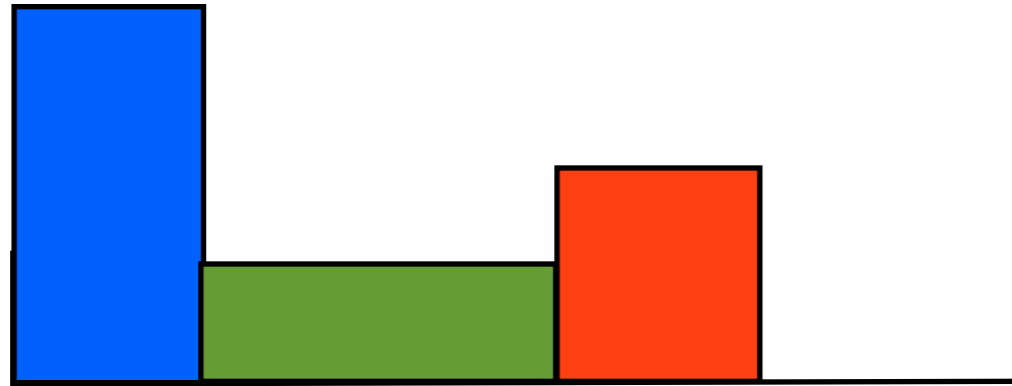
# A pictorial view of the objective



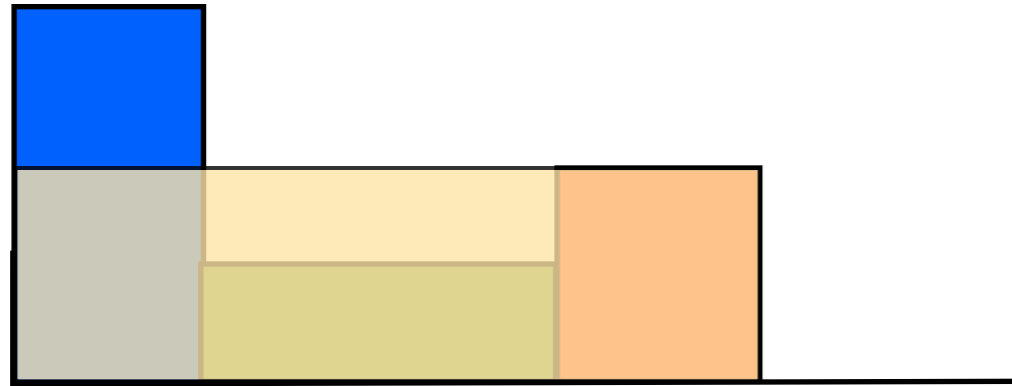
# A pictorial view of the objective



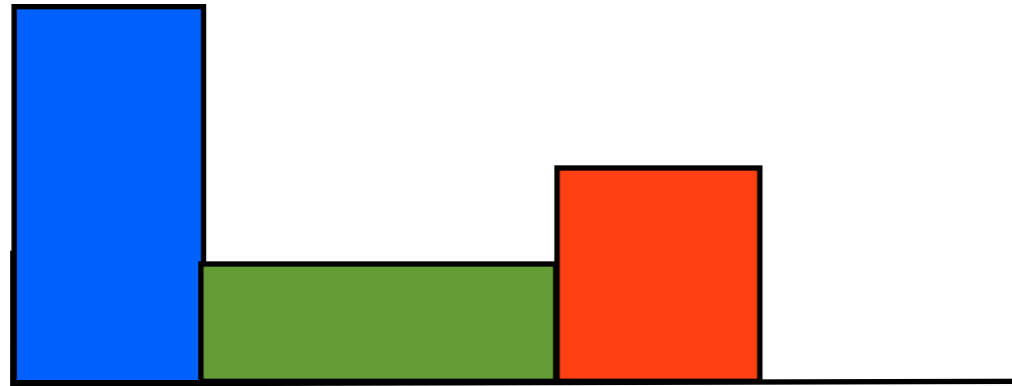
# A pictorial view of the objective



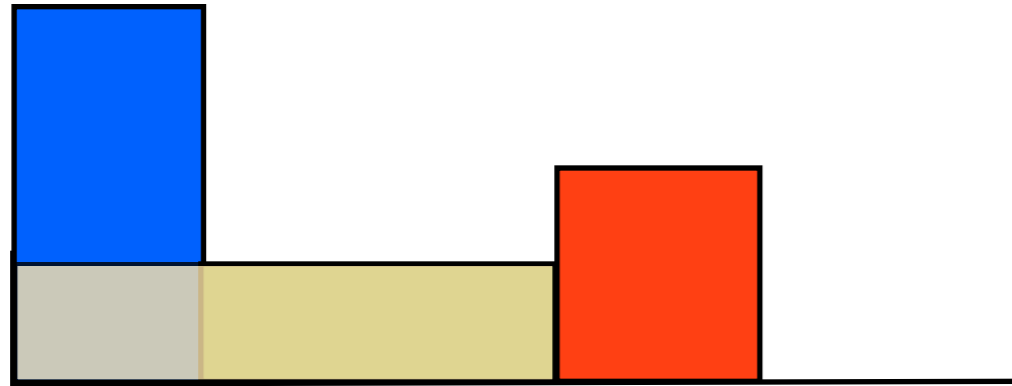
# A pictorial view of the objective



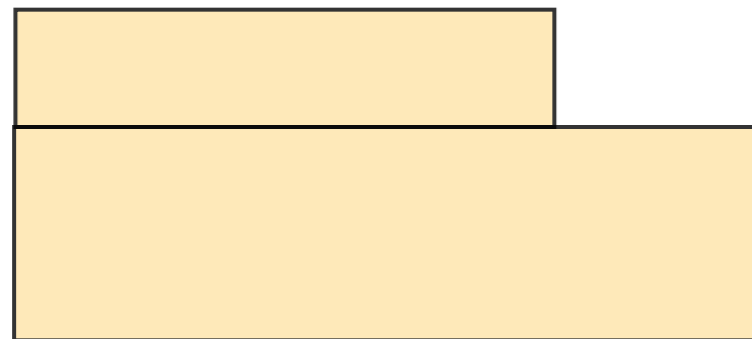
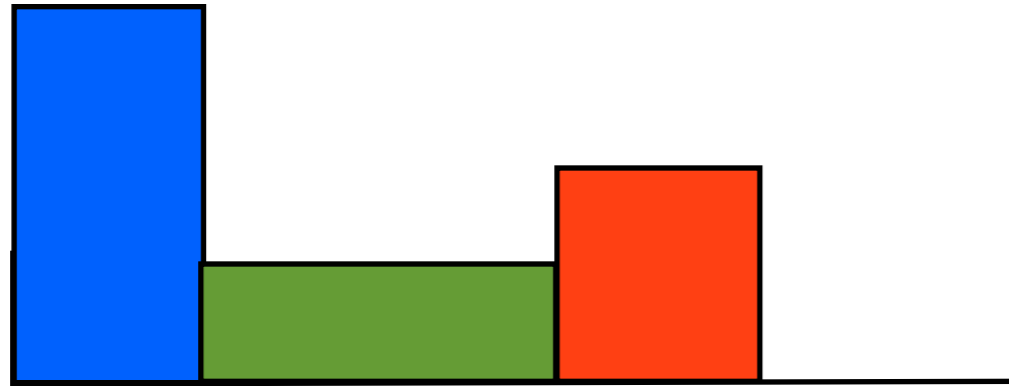
# A pictorial view of the objective



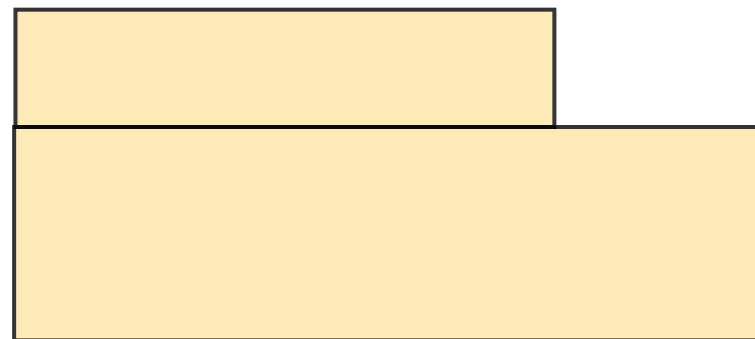
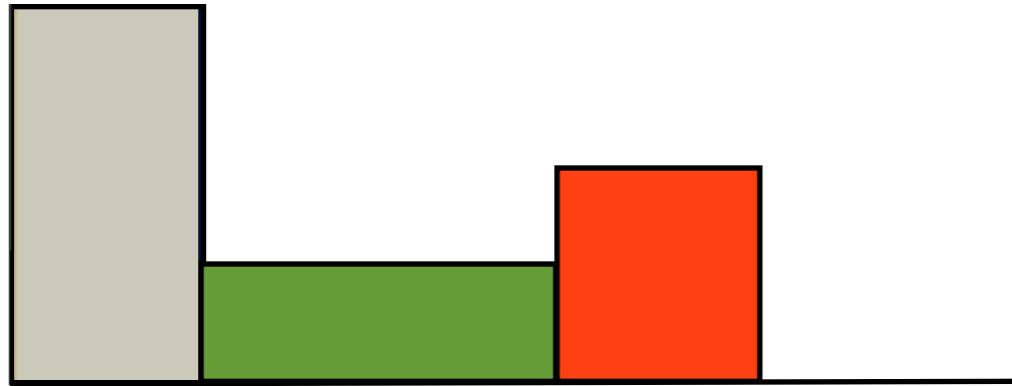
# A pictorial view of the objective



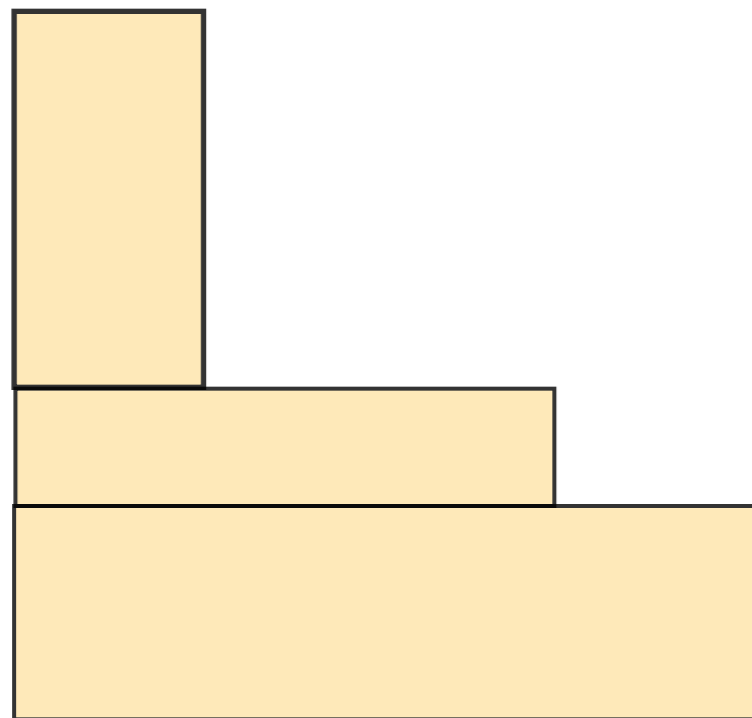
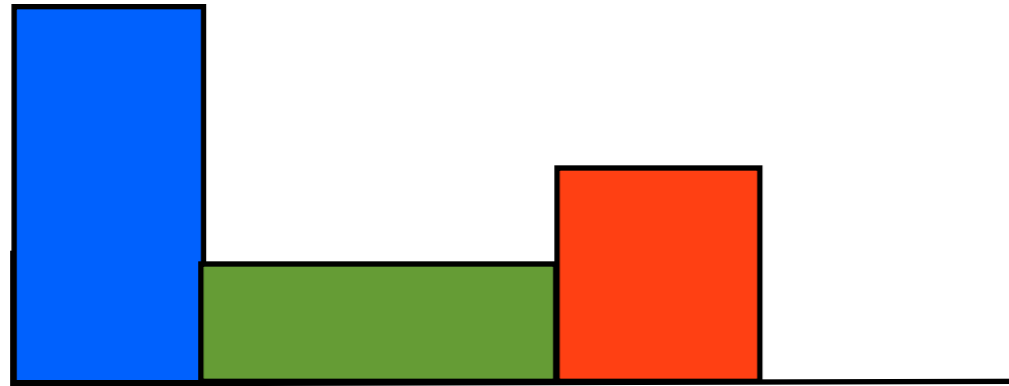
# A pictorial view of the objective



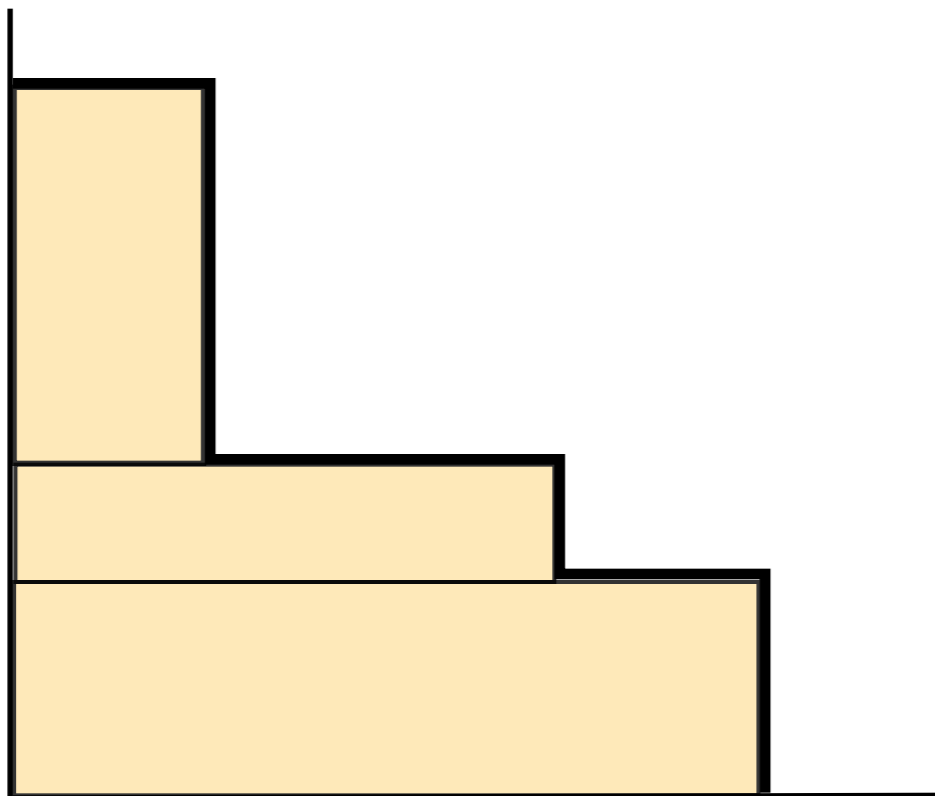
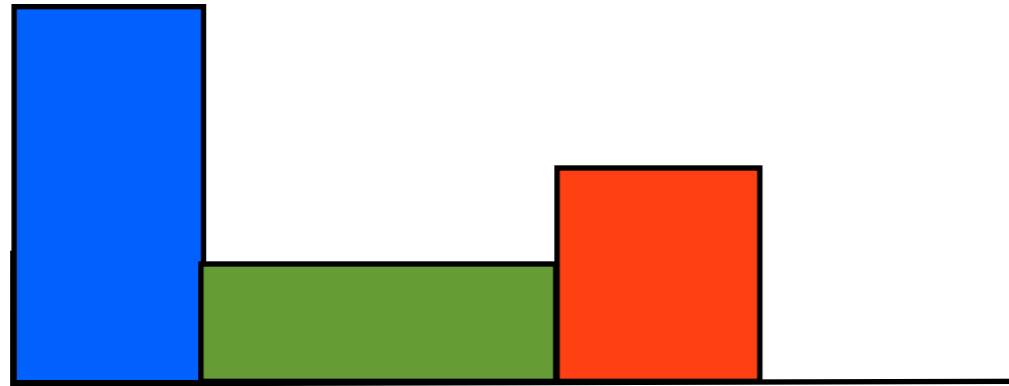
# A pictorial view of the objective



# A pictorial view of the objective

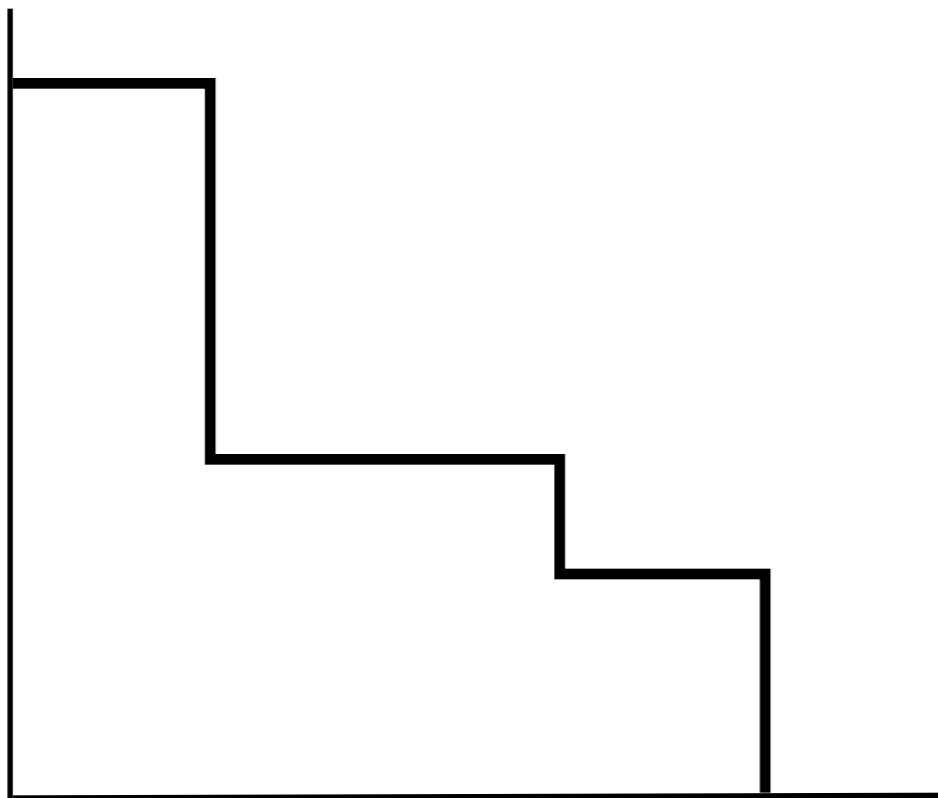
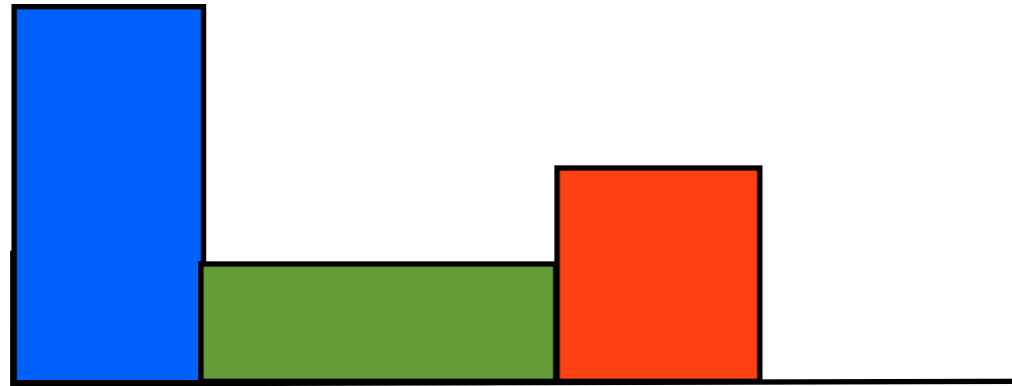


# A pictorial view of the objective

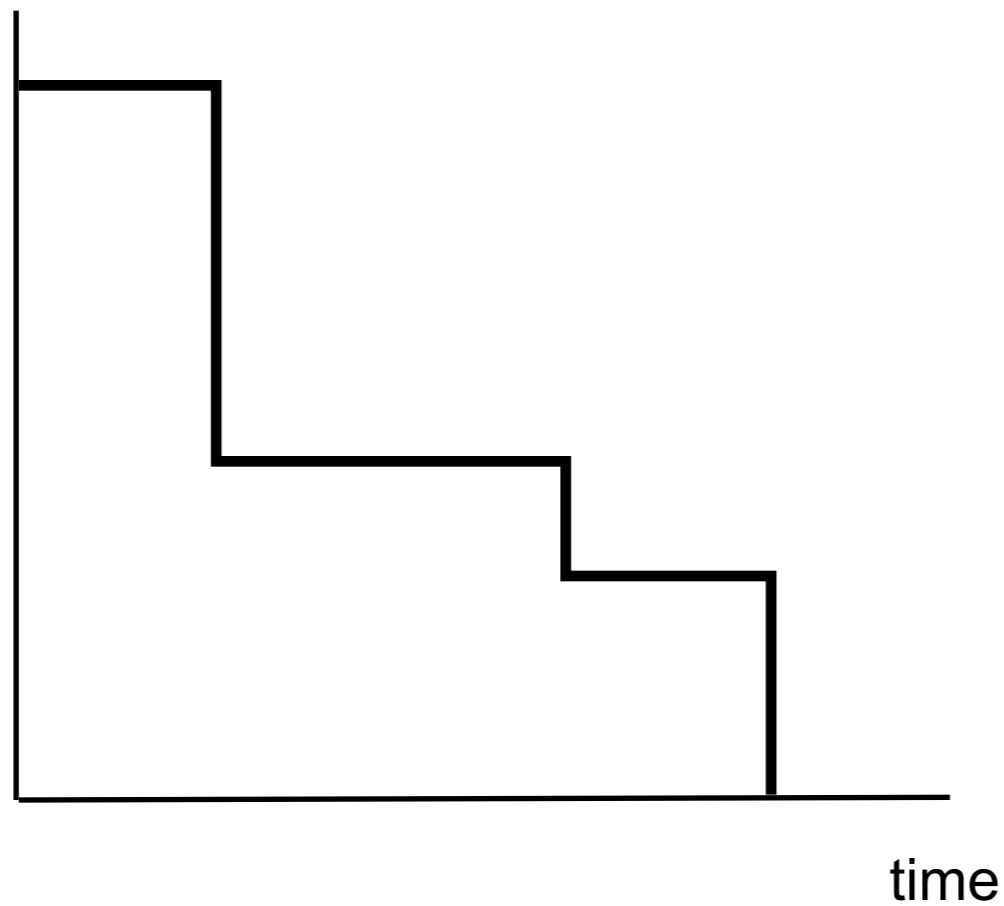
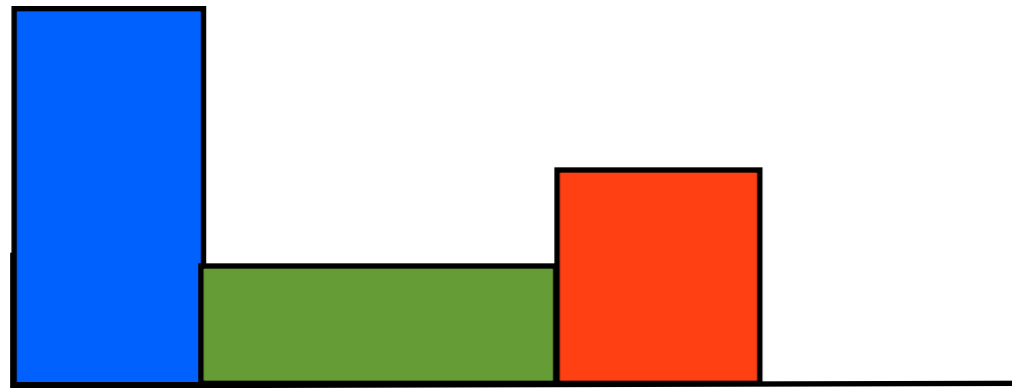




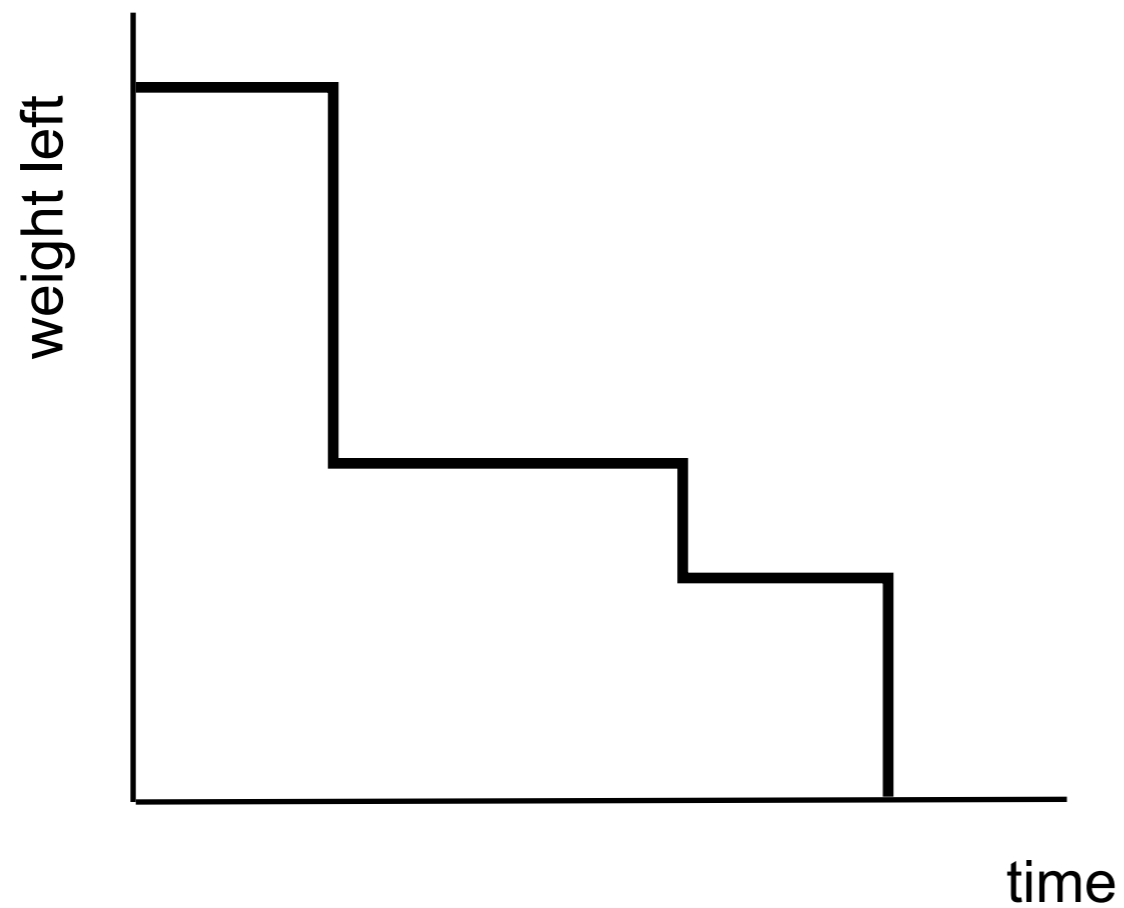
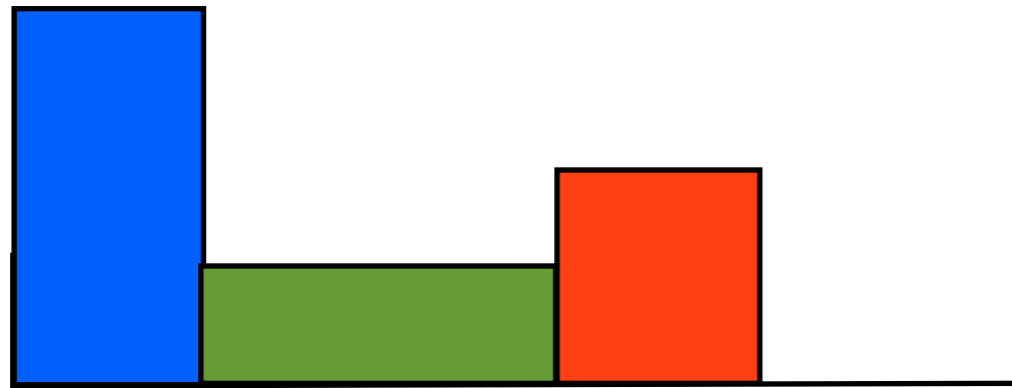
# A pictorial view of the objective



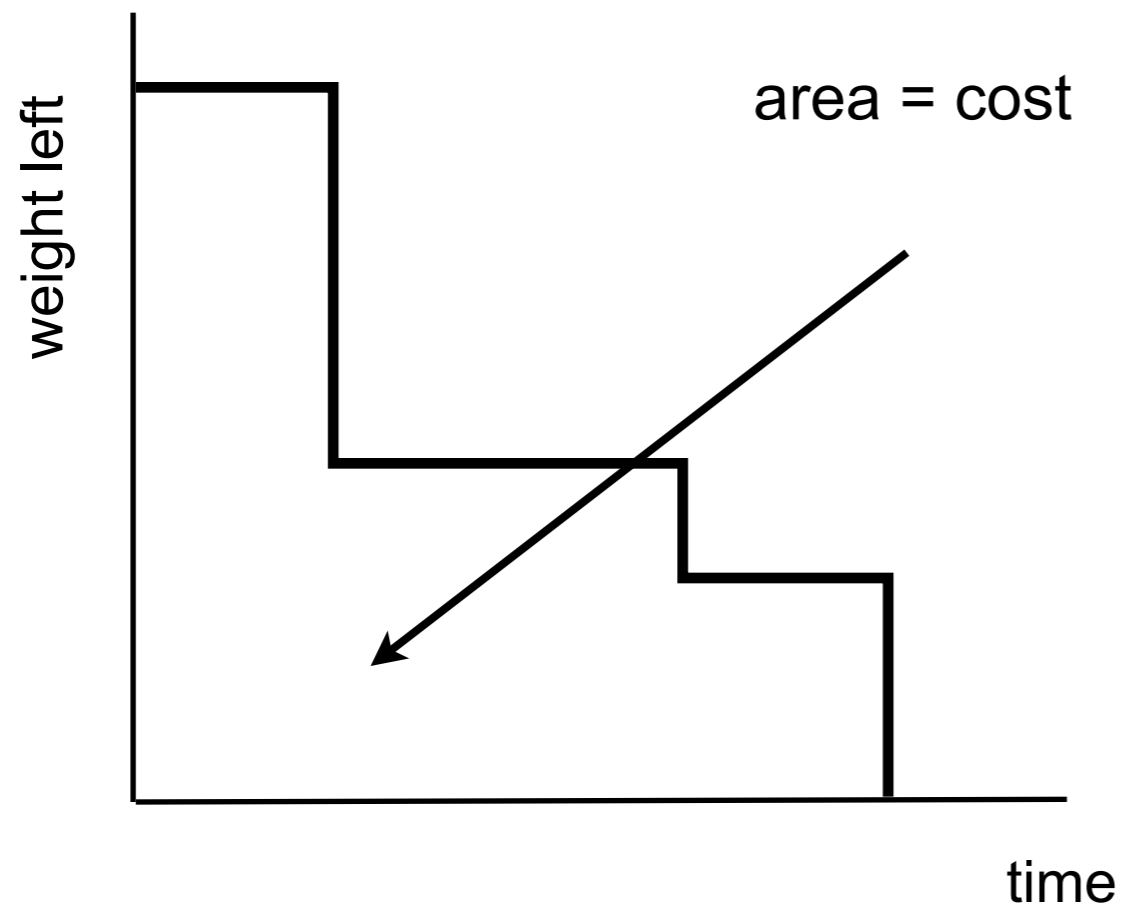
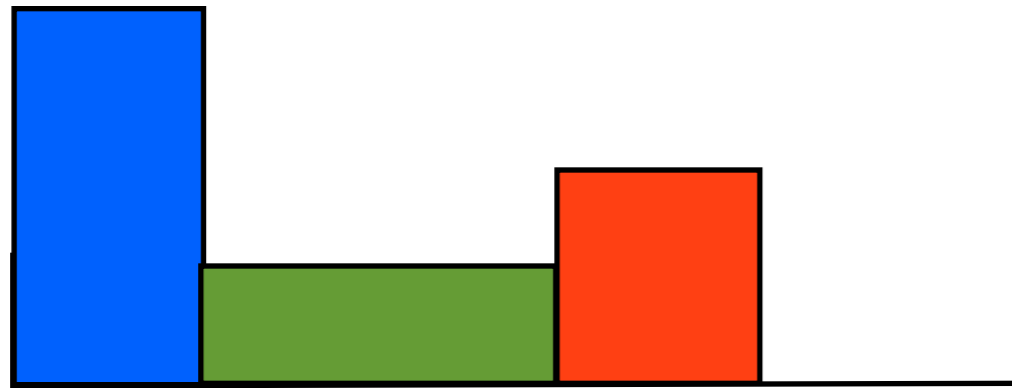
# A pictorial view of the objective



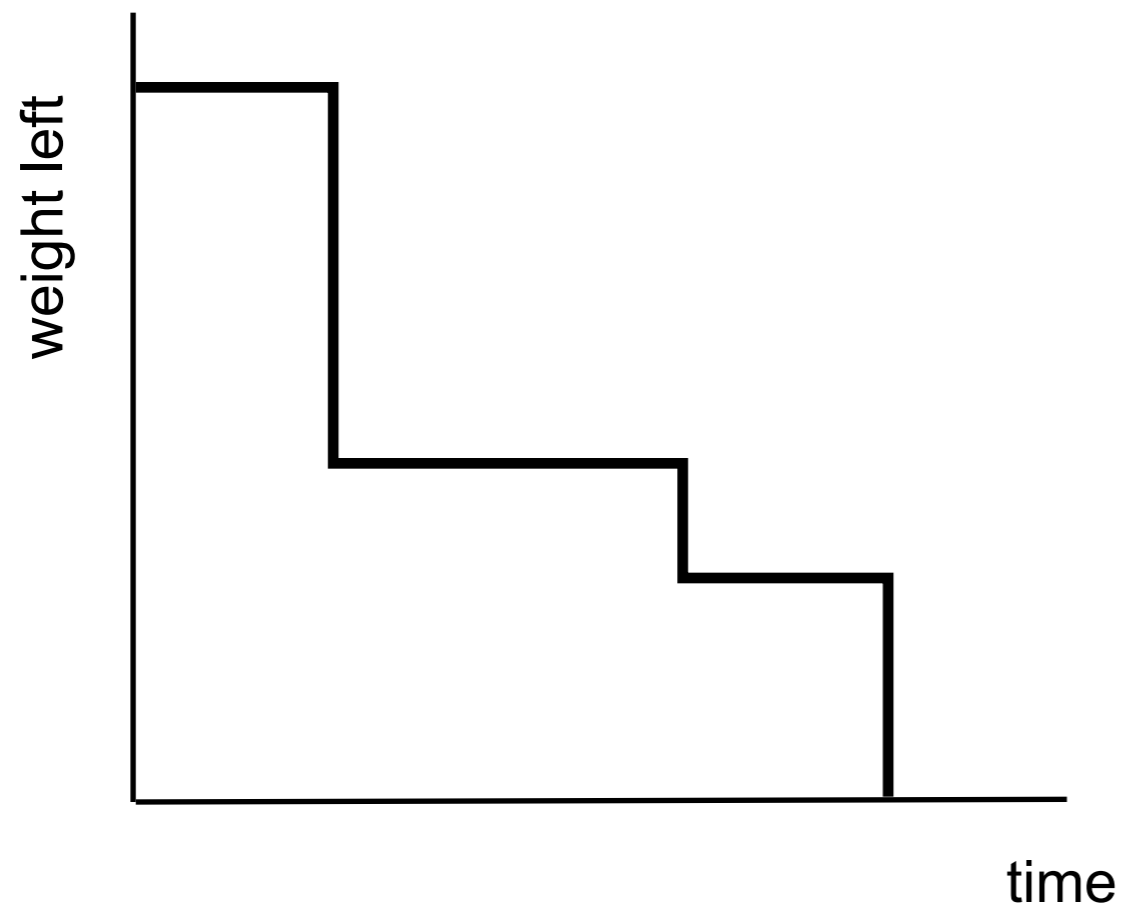
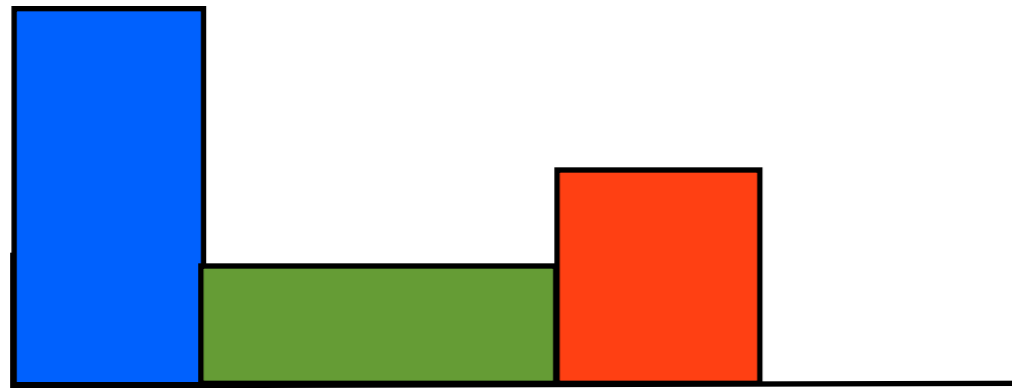
# A pictorial view of the objective



# A pictorial view of the objective

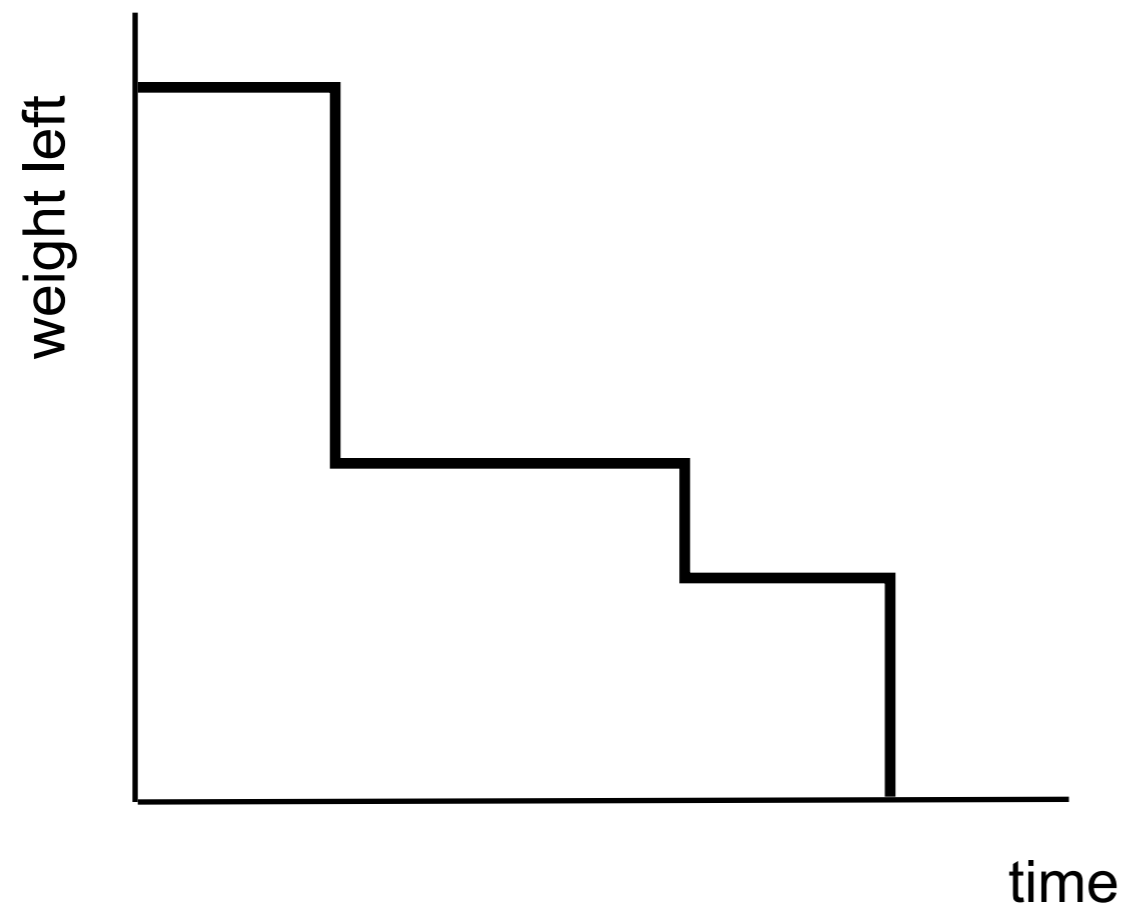
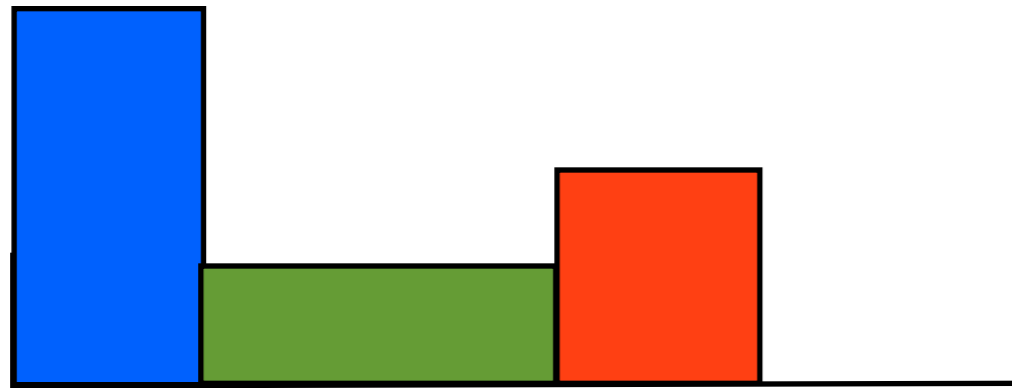


# A pictorial view of the objective

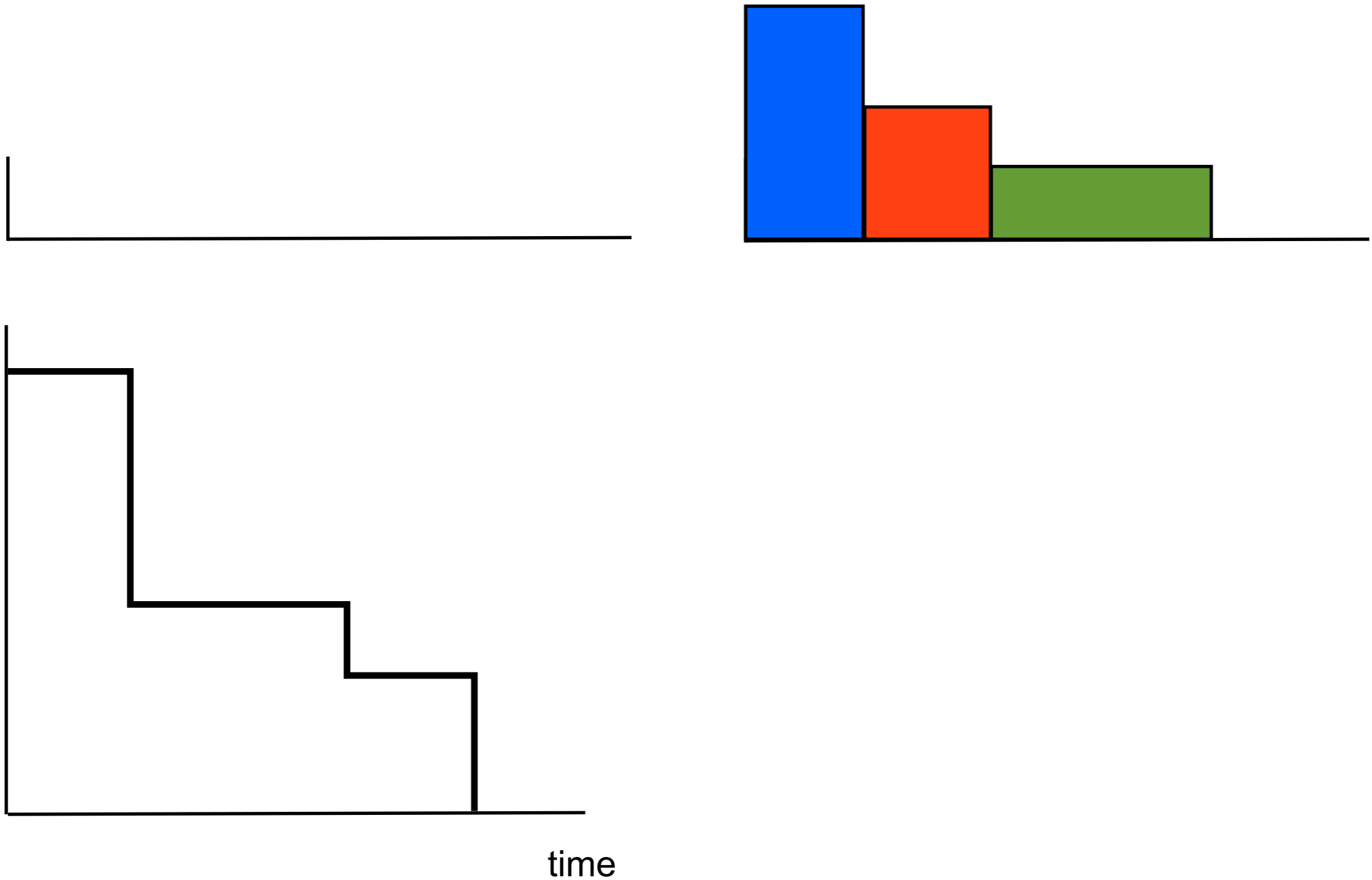




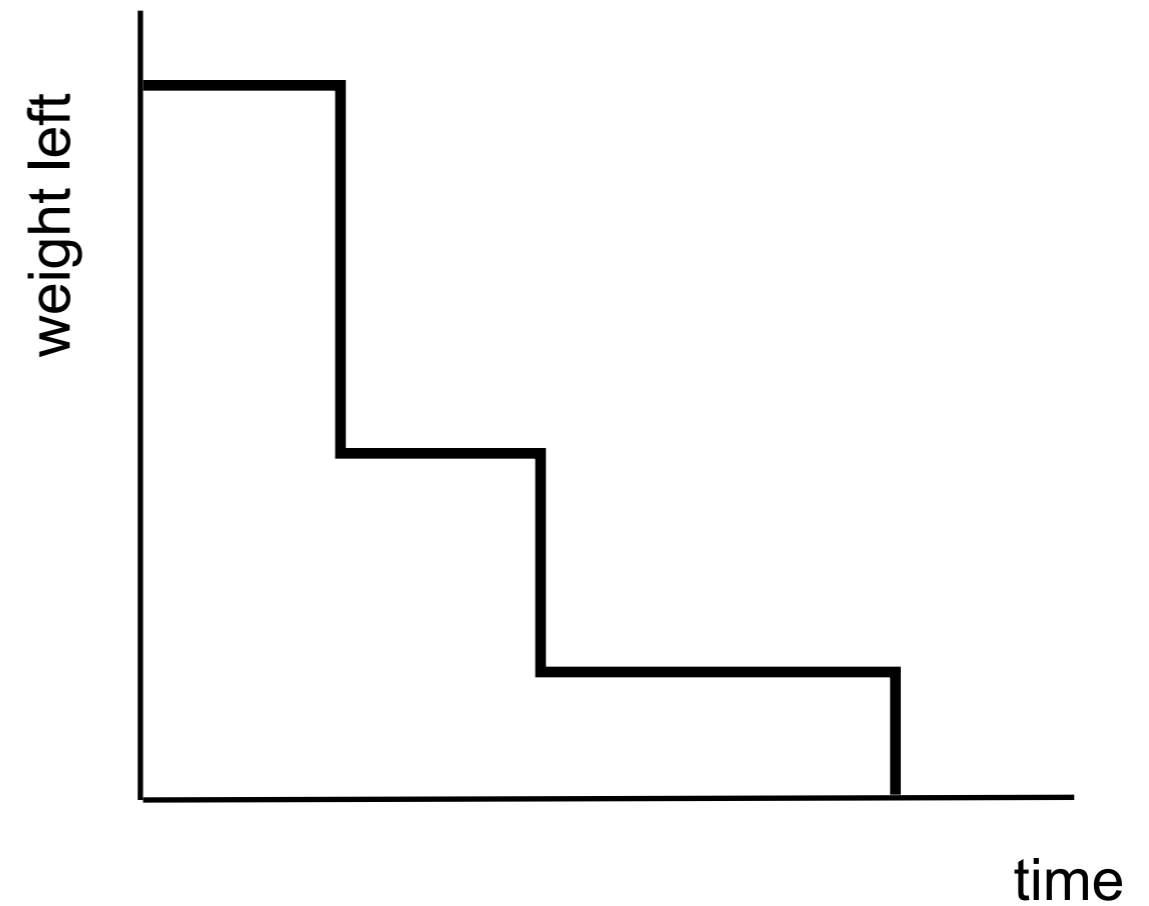
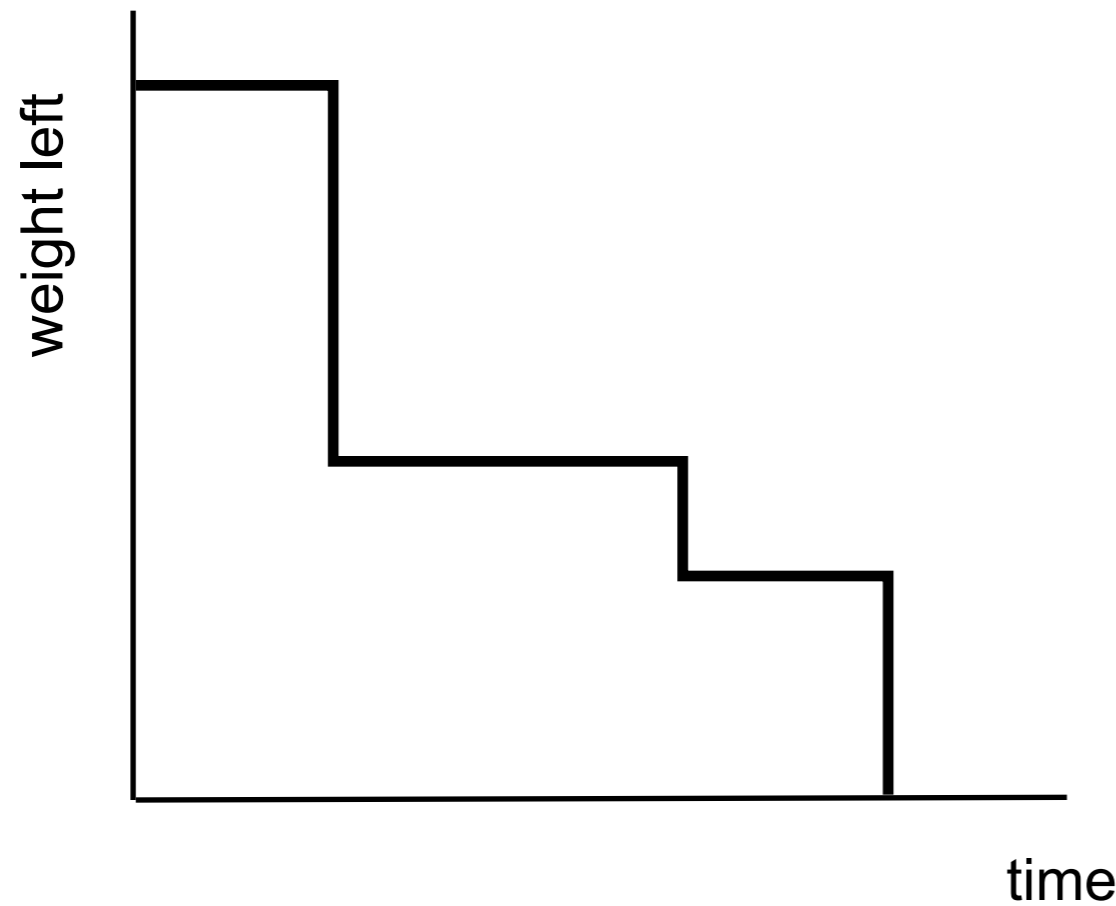
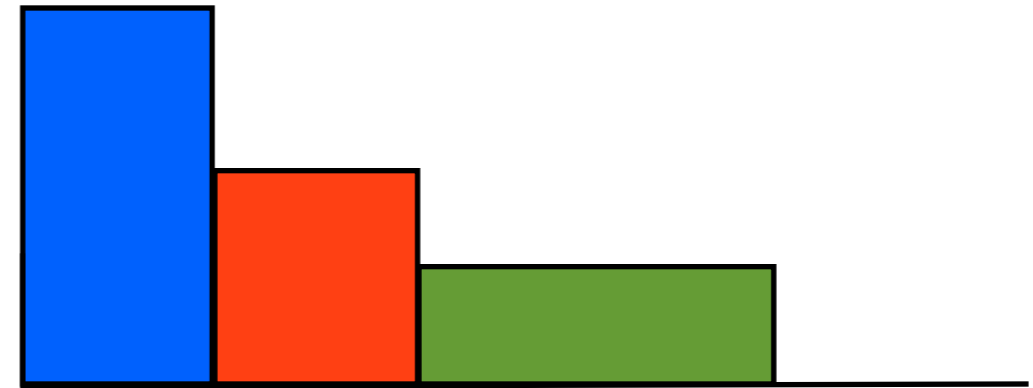
# A pictorial view of the objective



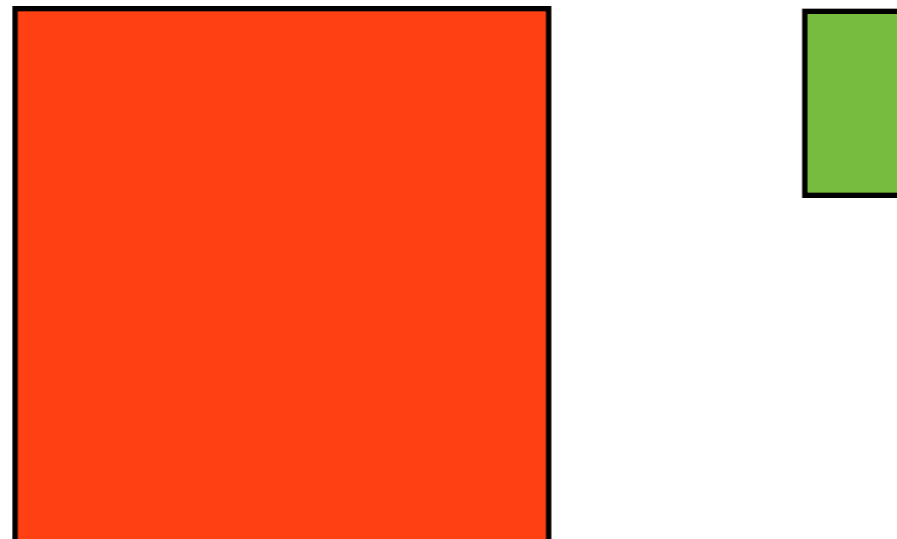
# A pictorial view of the objective



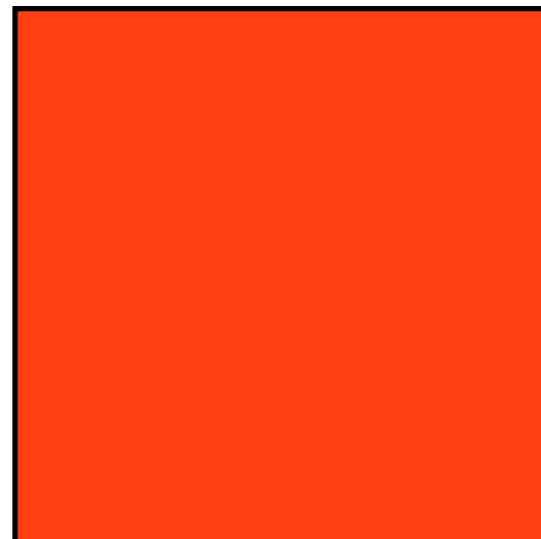
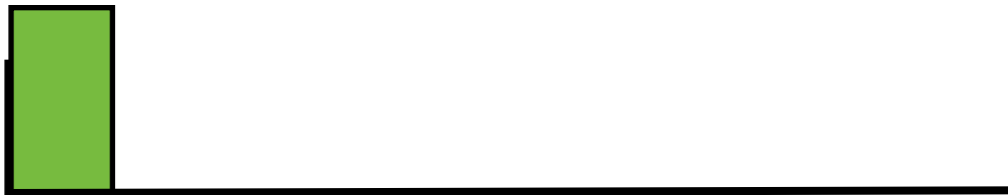
# A pictorial view of the objective



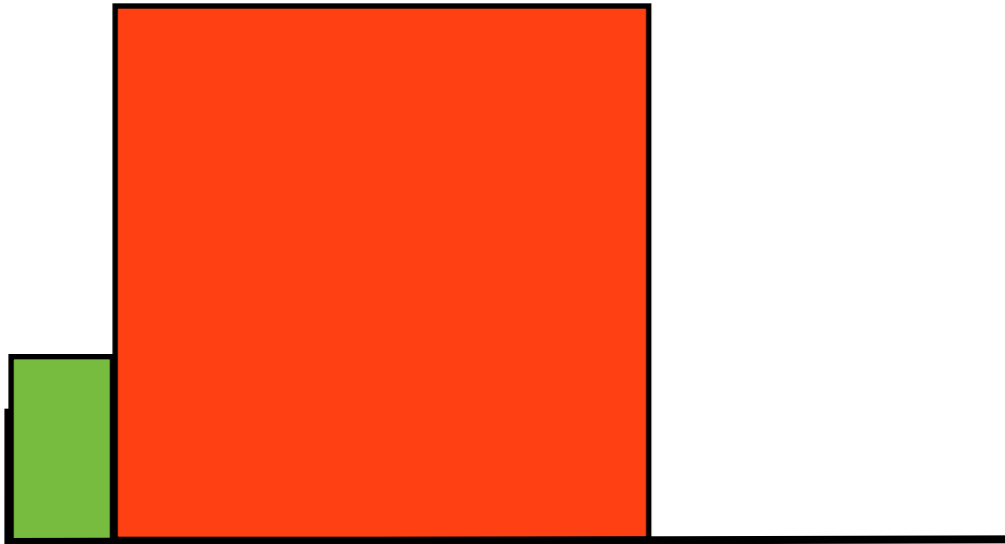
# Smith's rule is not competitive



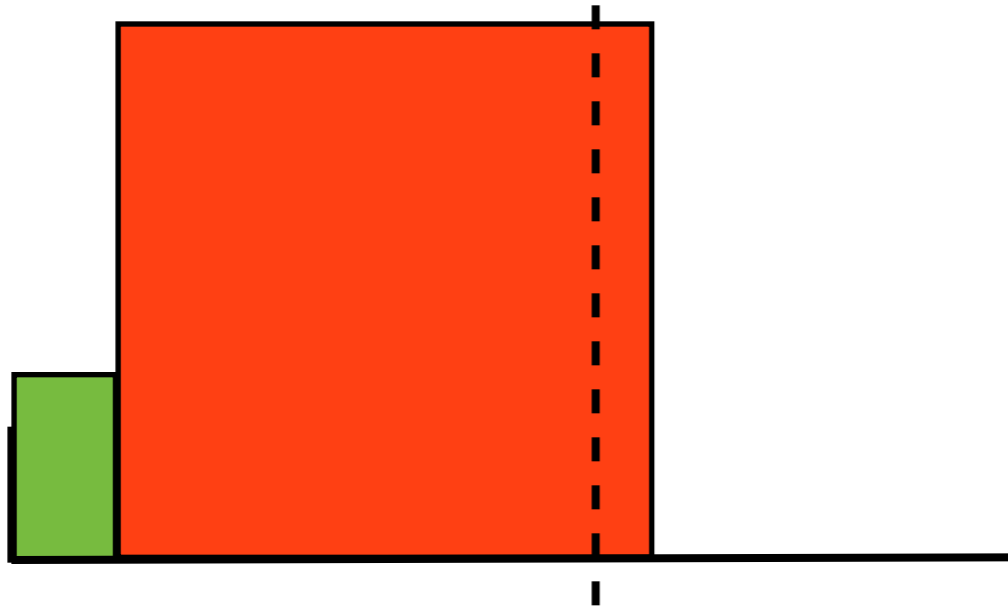
# Smith's rule is not competitive



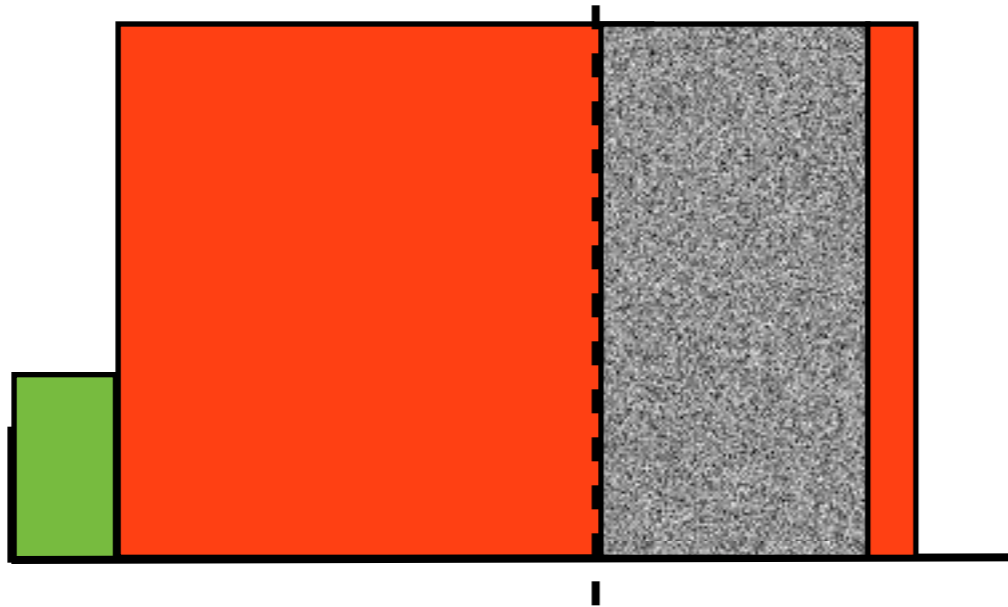
# Smith's rule is not competitive



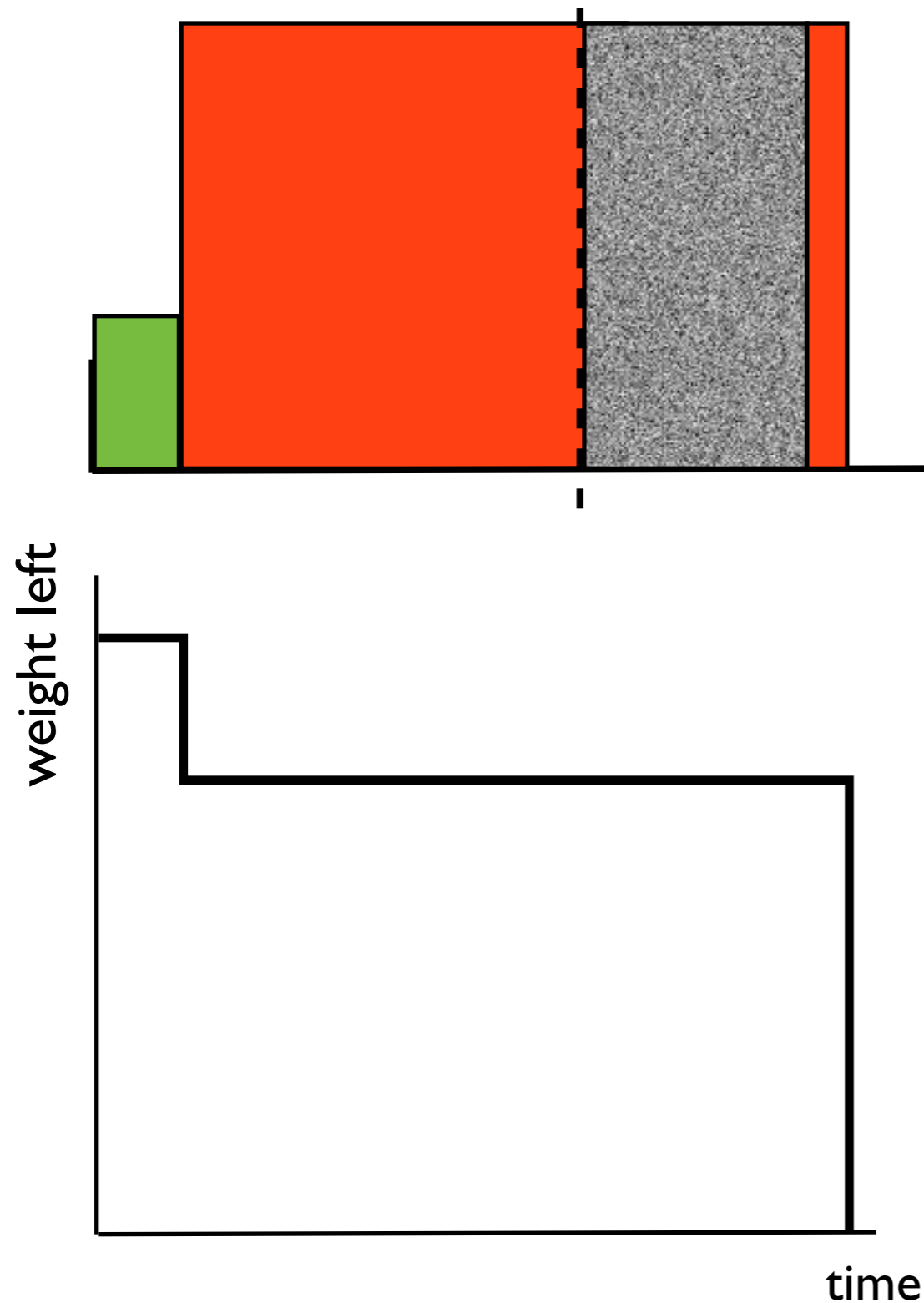
# Smith's rule is not competitive



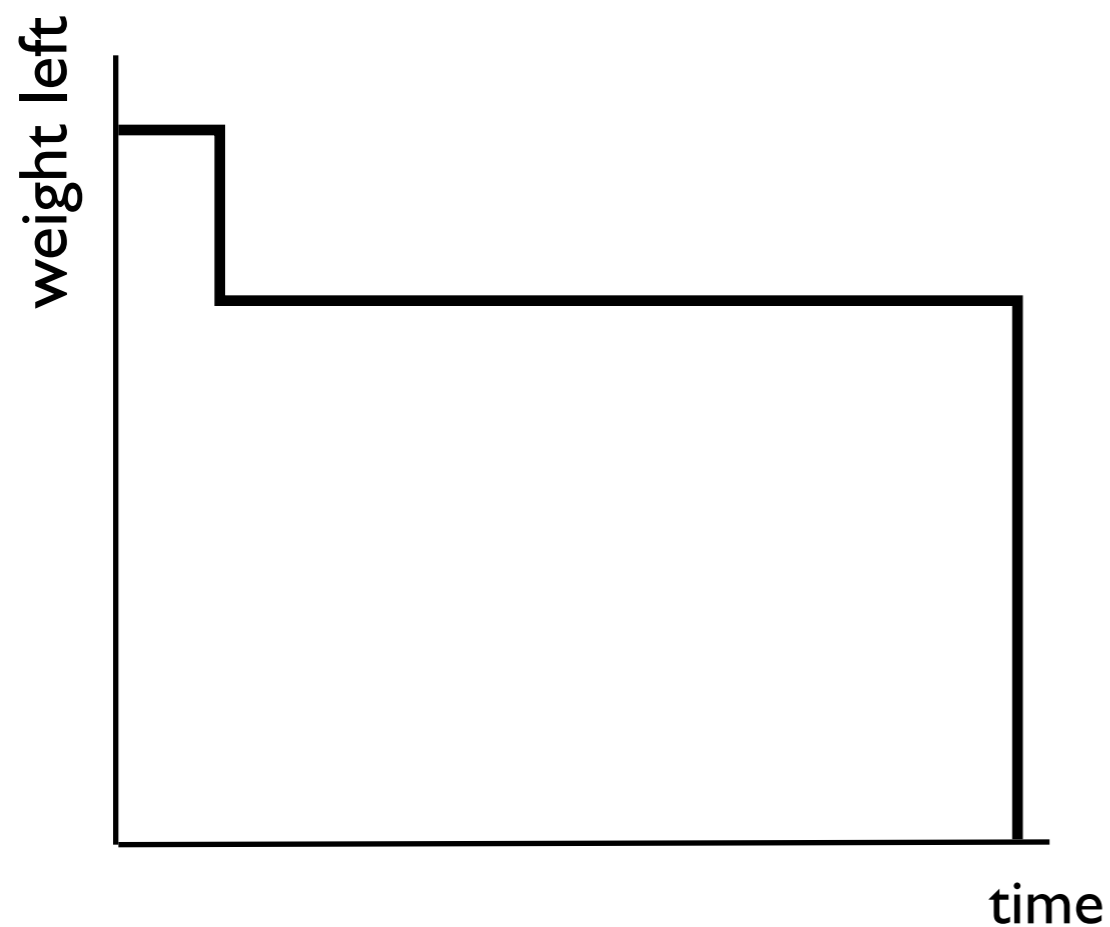
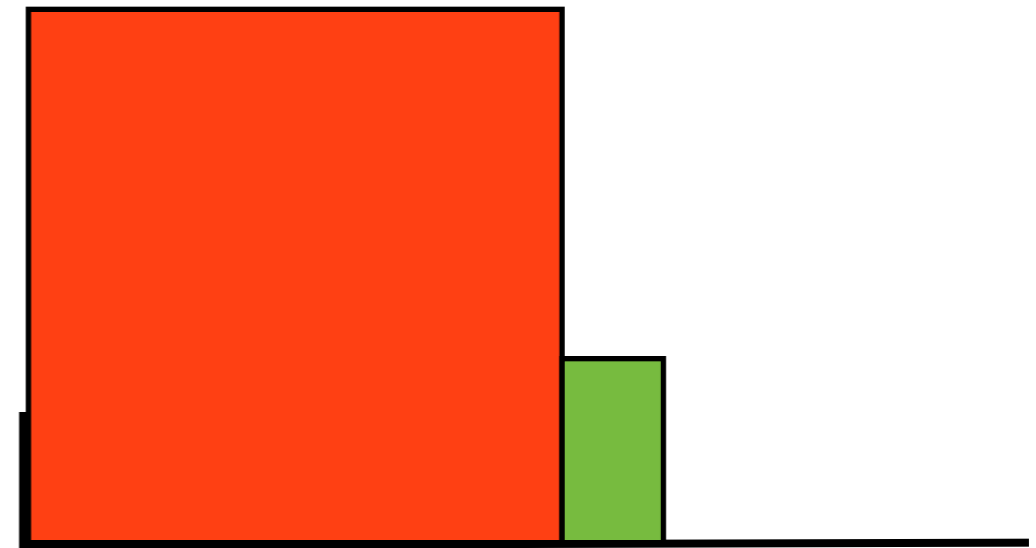
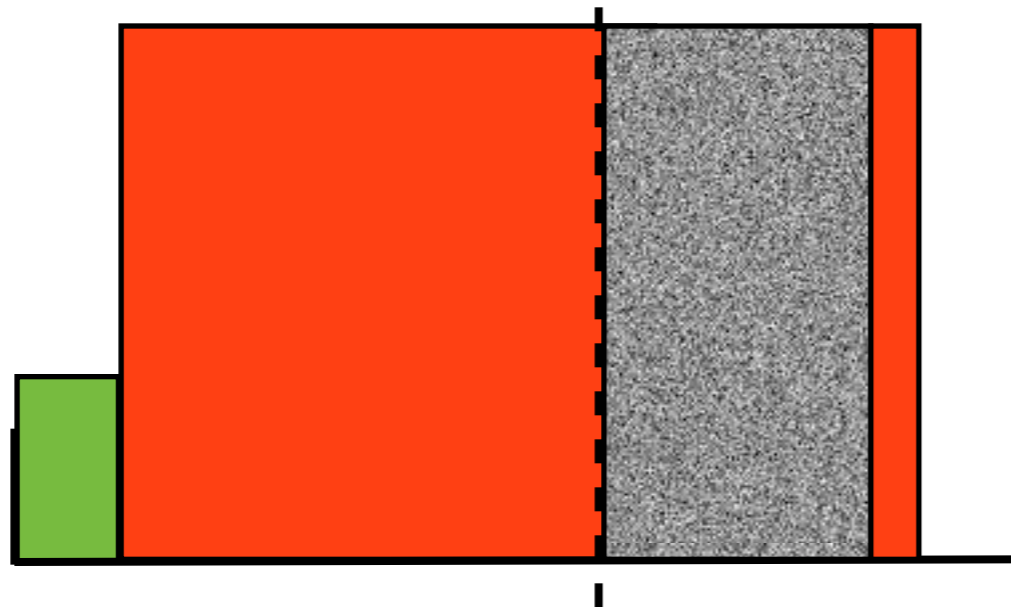
# Smith's rule is not competitive



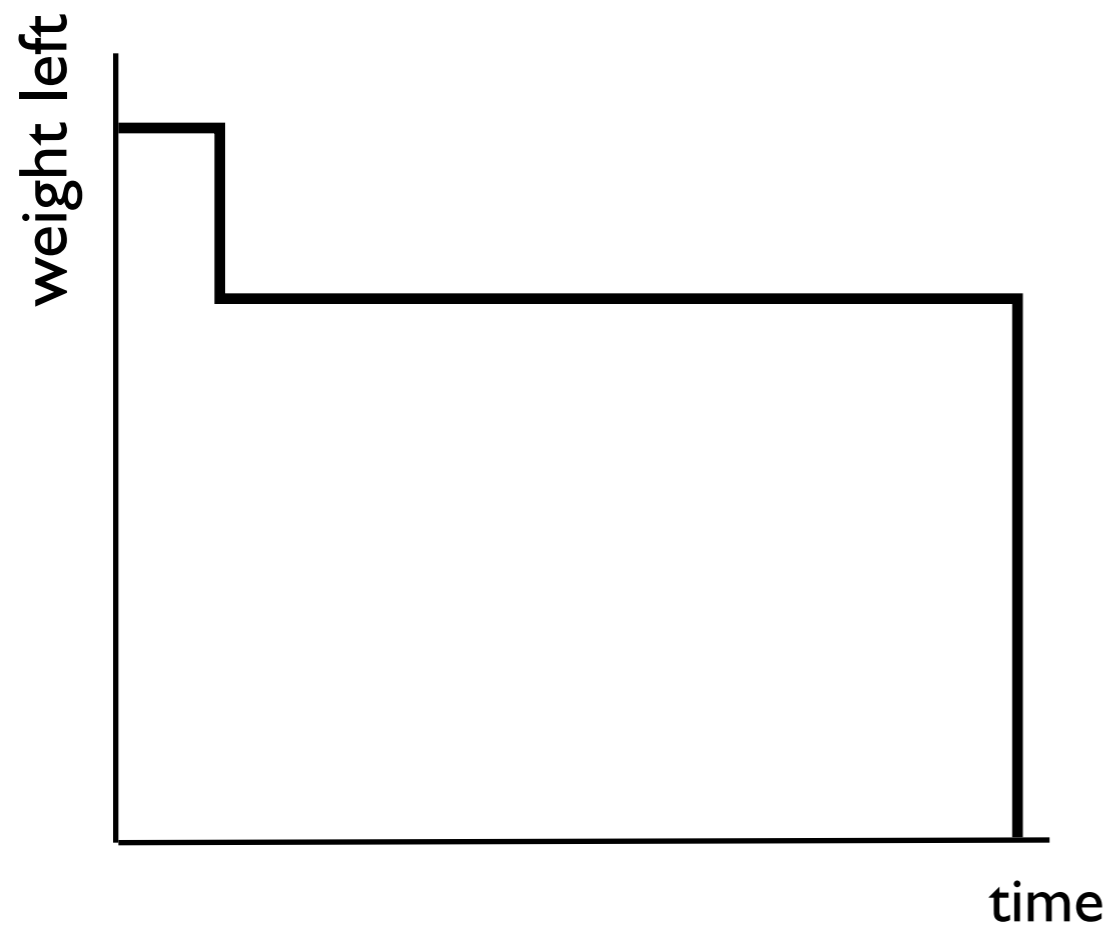
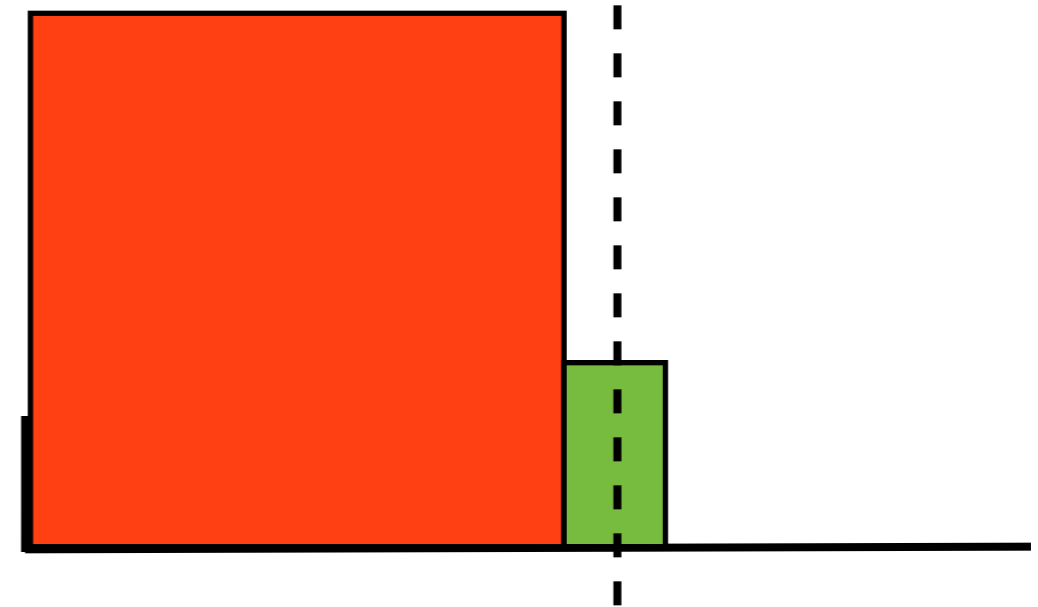
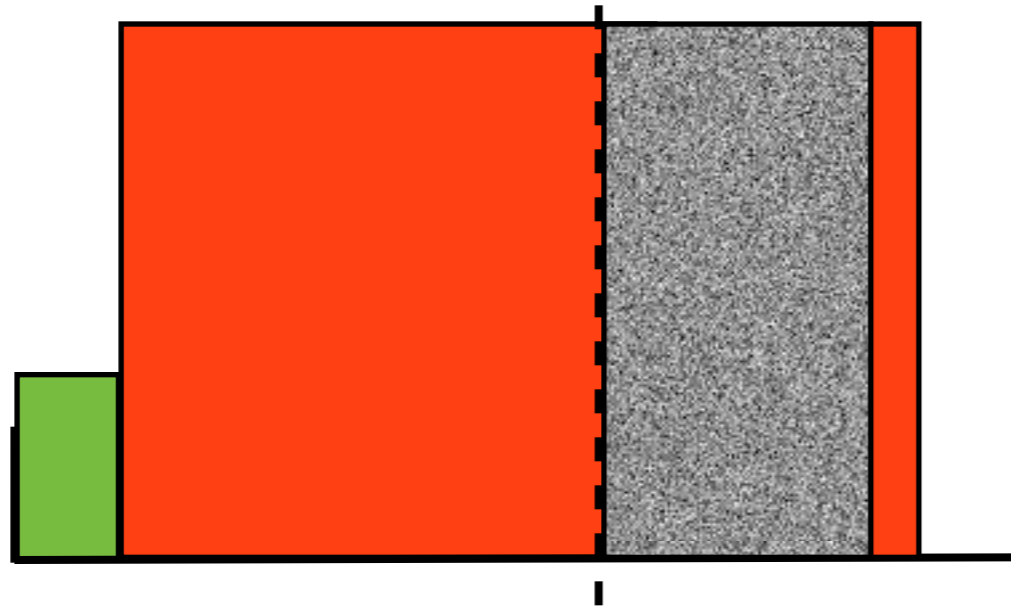
# Smith's rule is not competitive



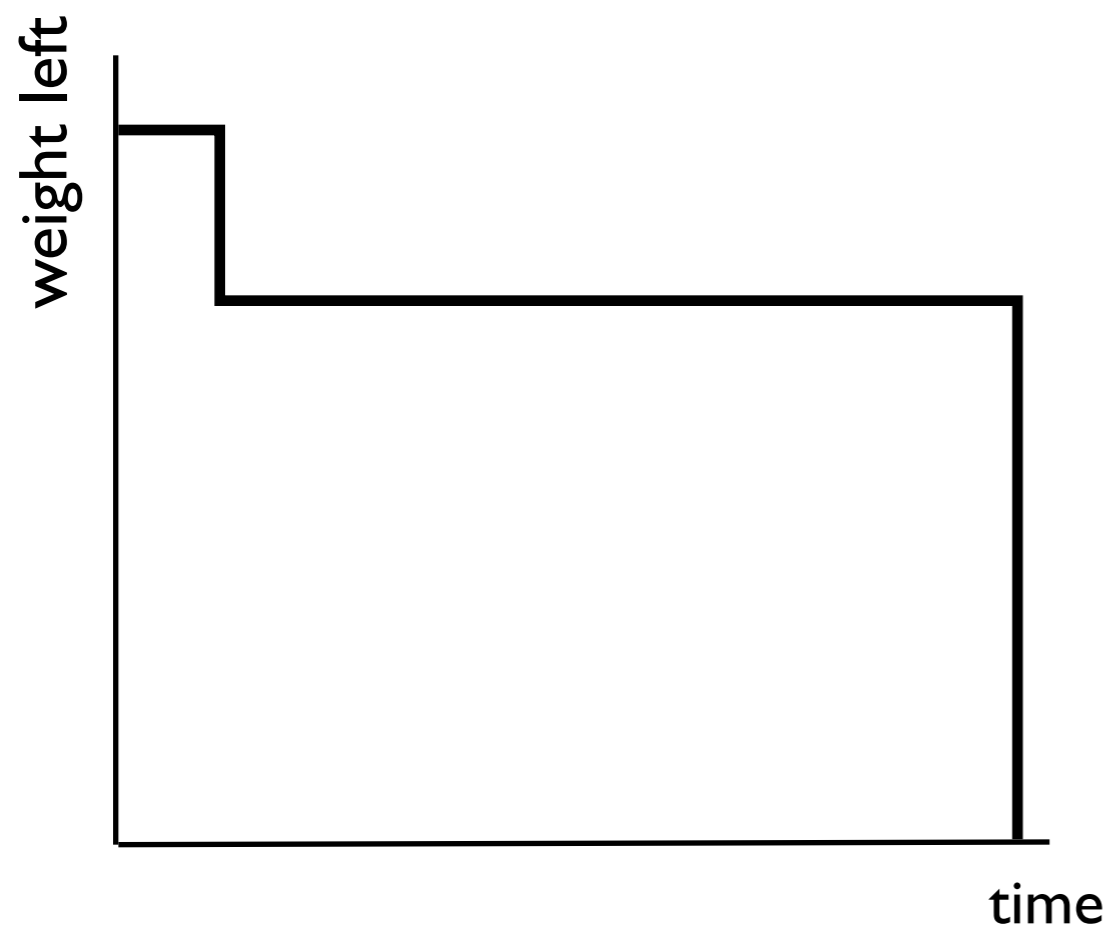
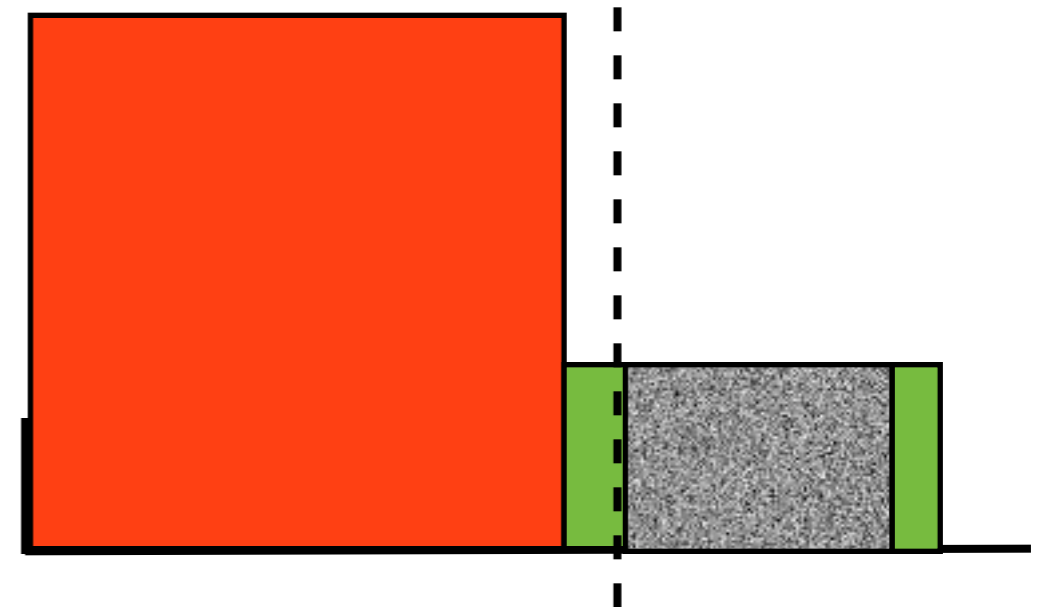
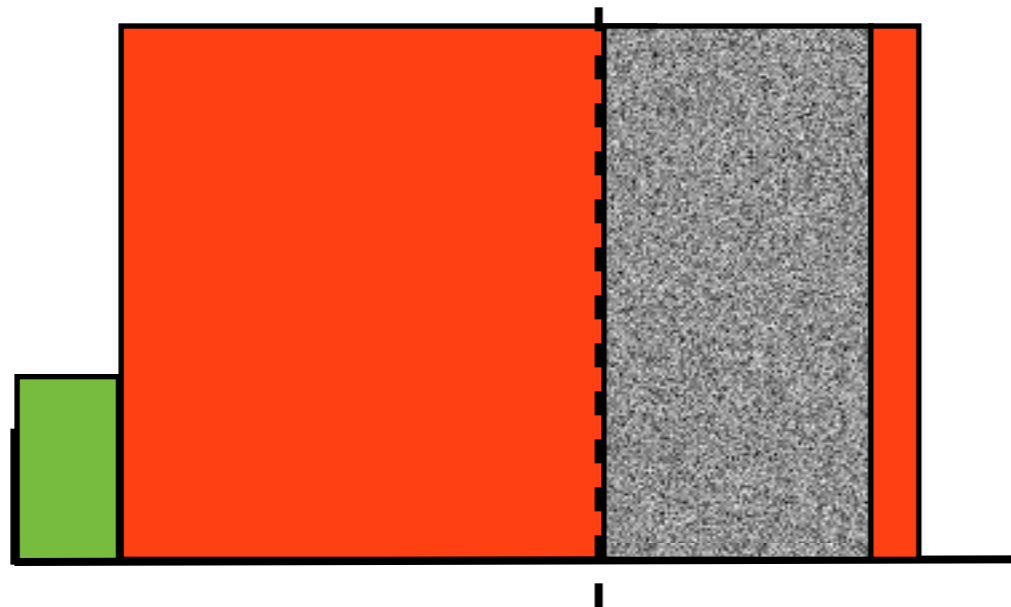
# Smith's rule is not competitive



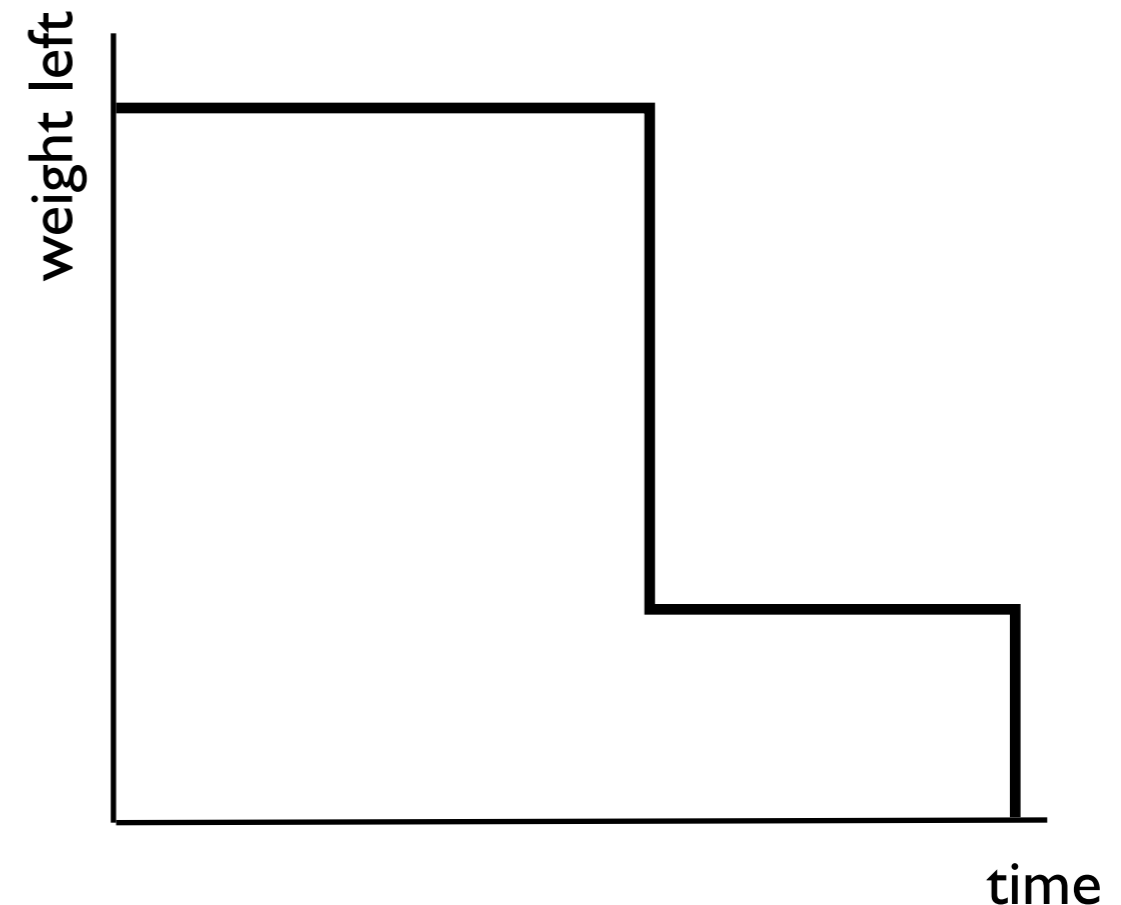
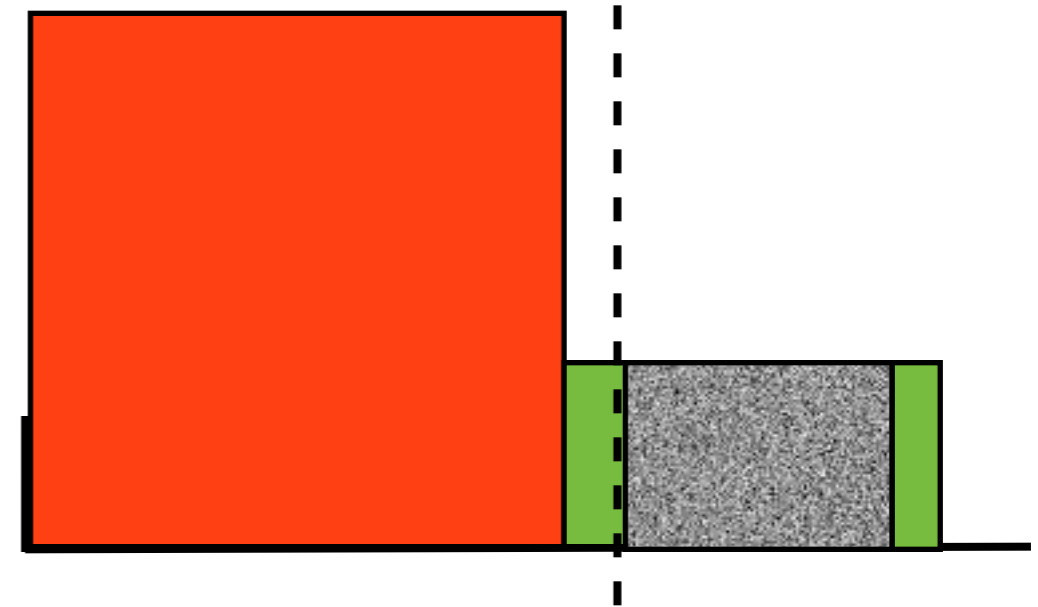
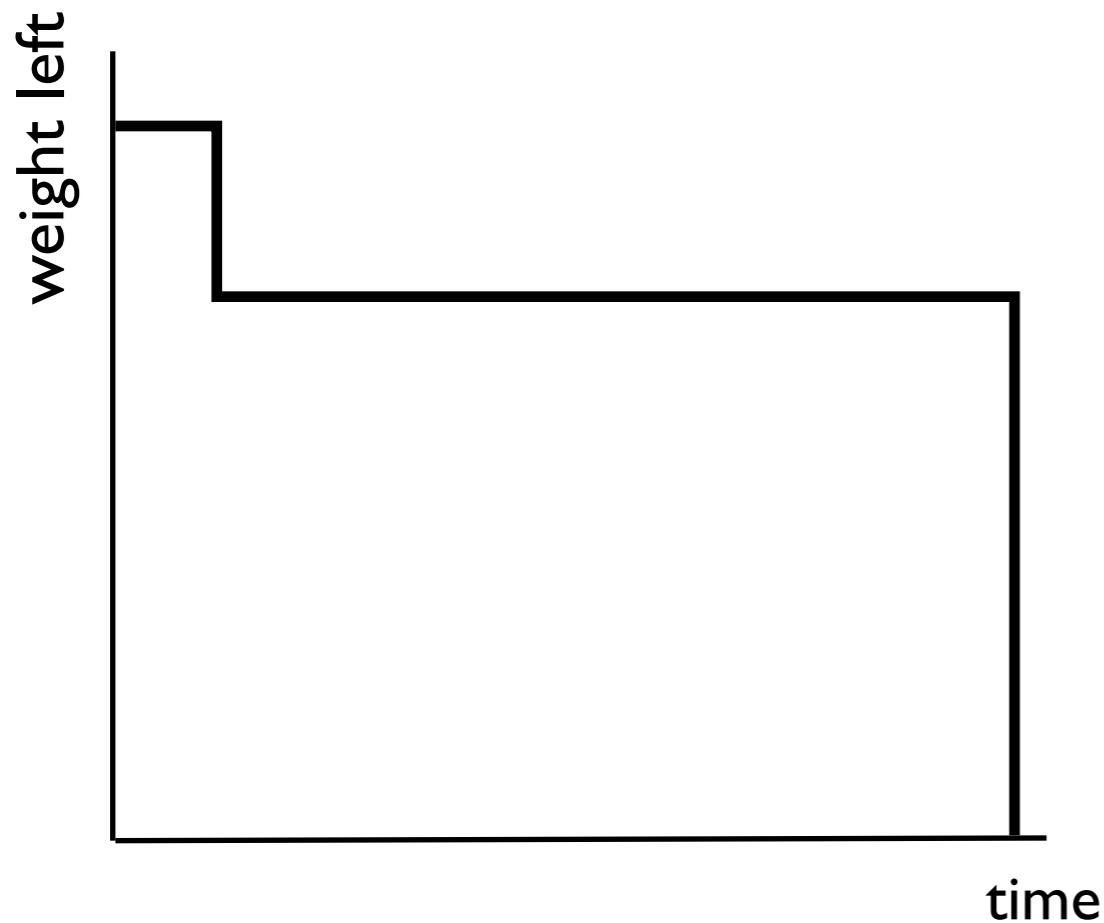
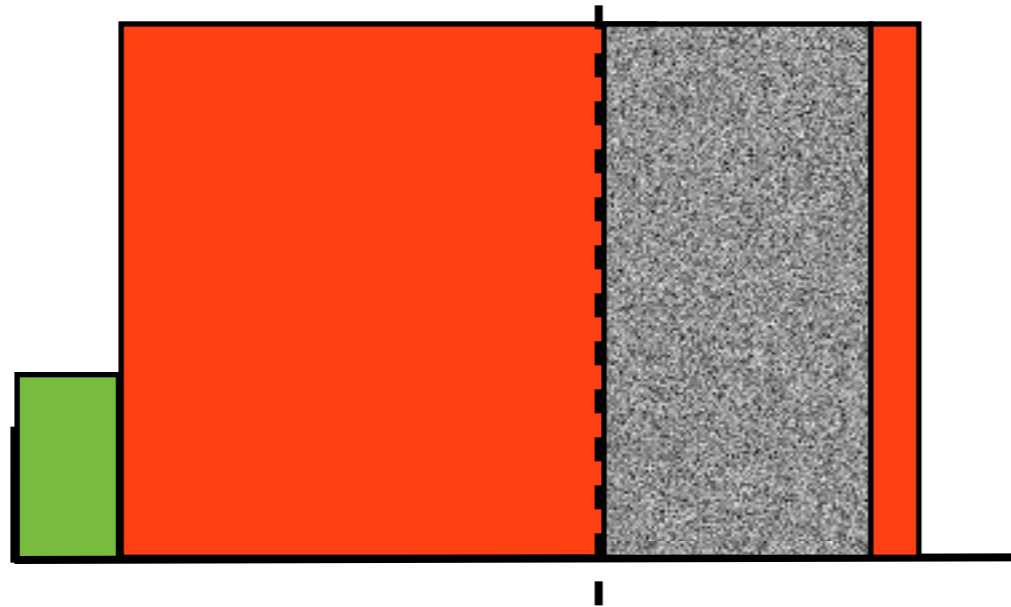
# Smith's rule is not competitive



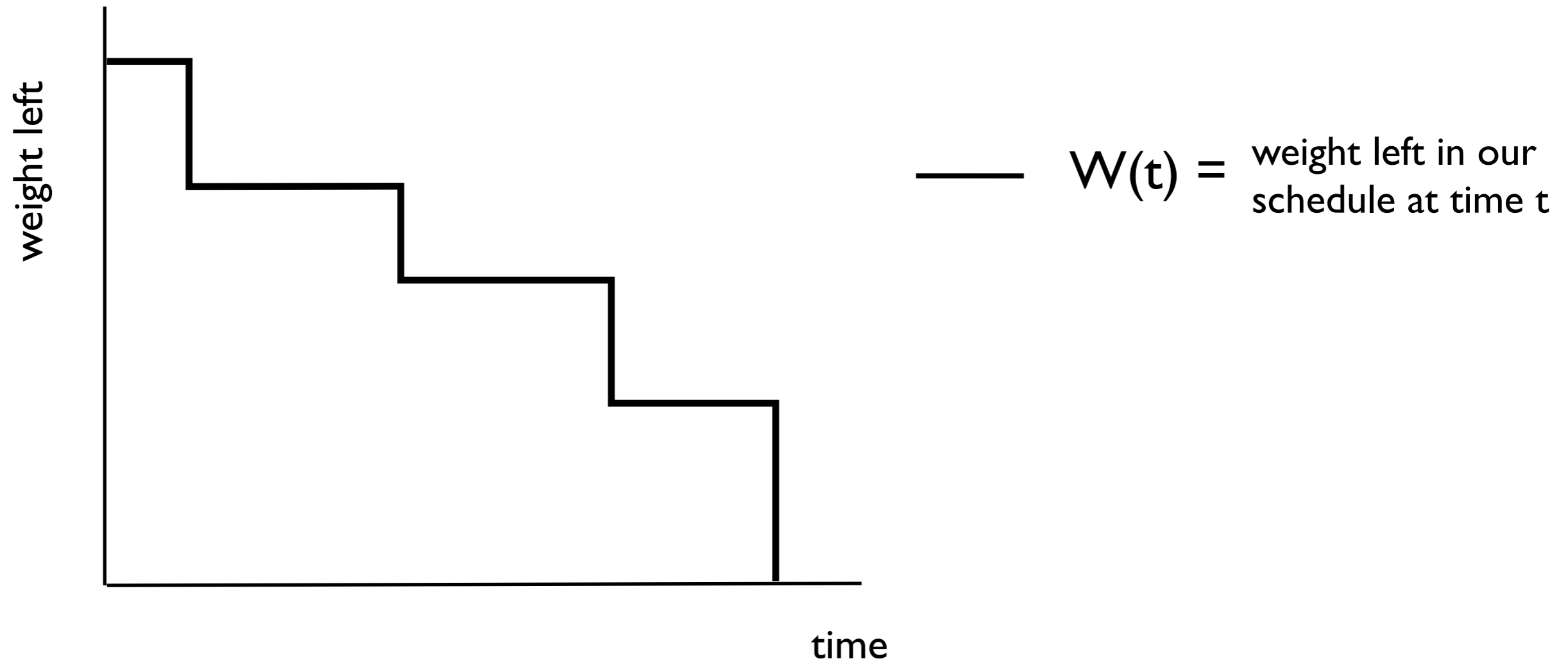
# Smith's rule is not competitive



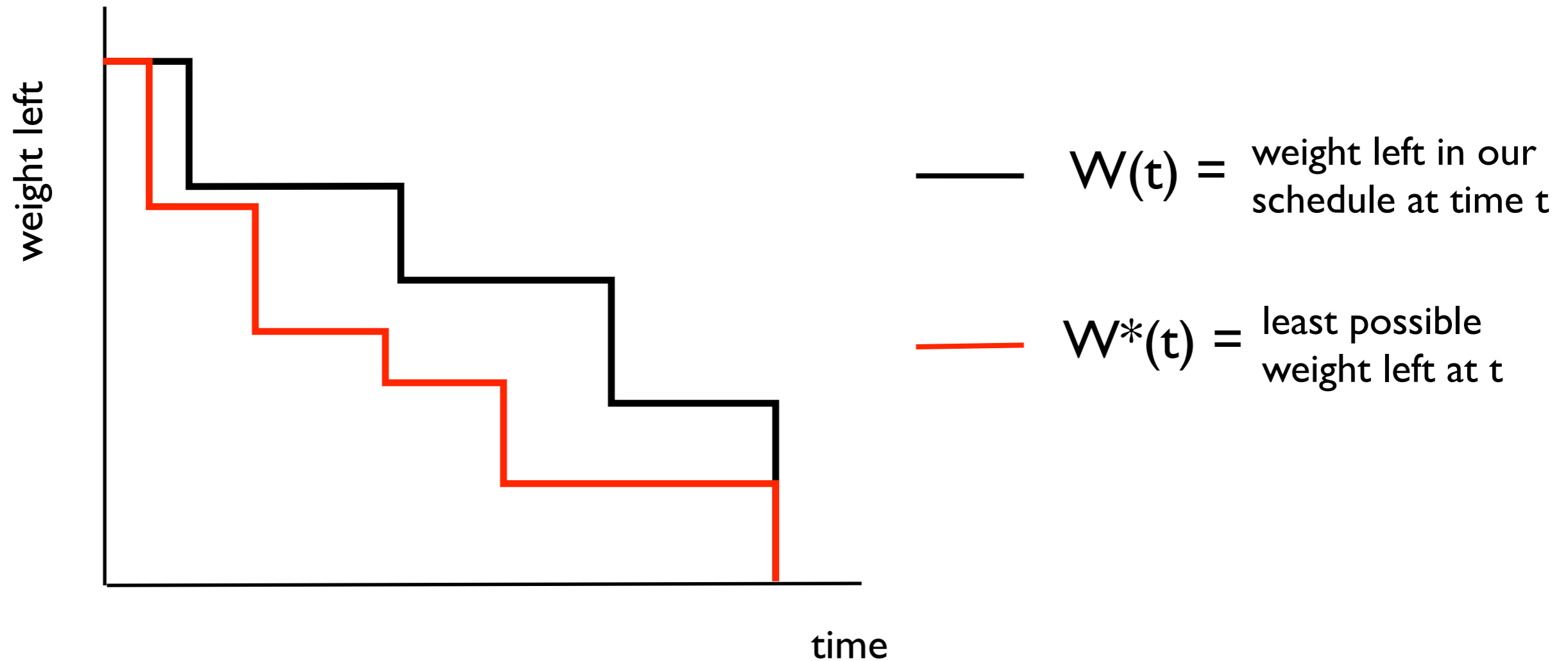
# Smith's rule is not competitive



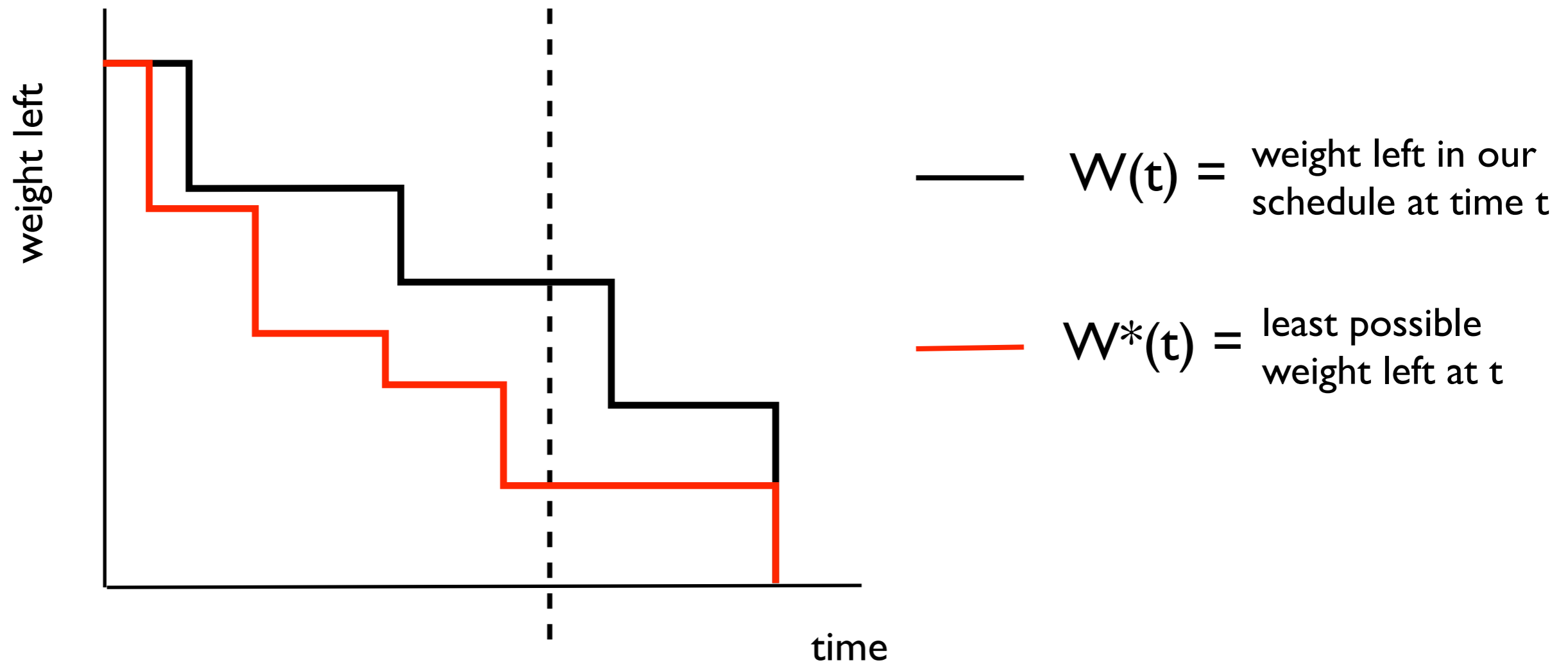
# What can go wrong?



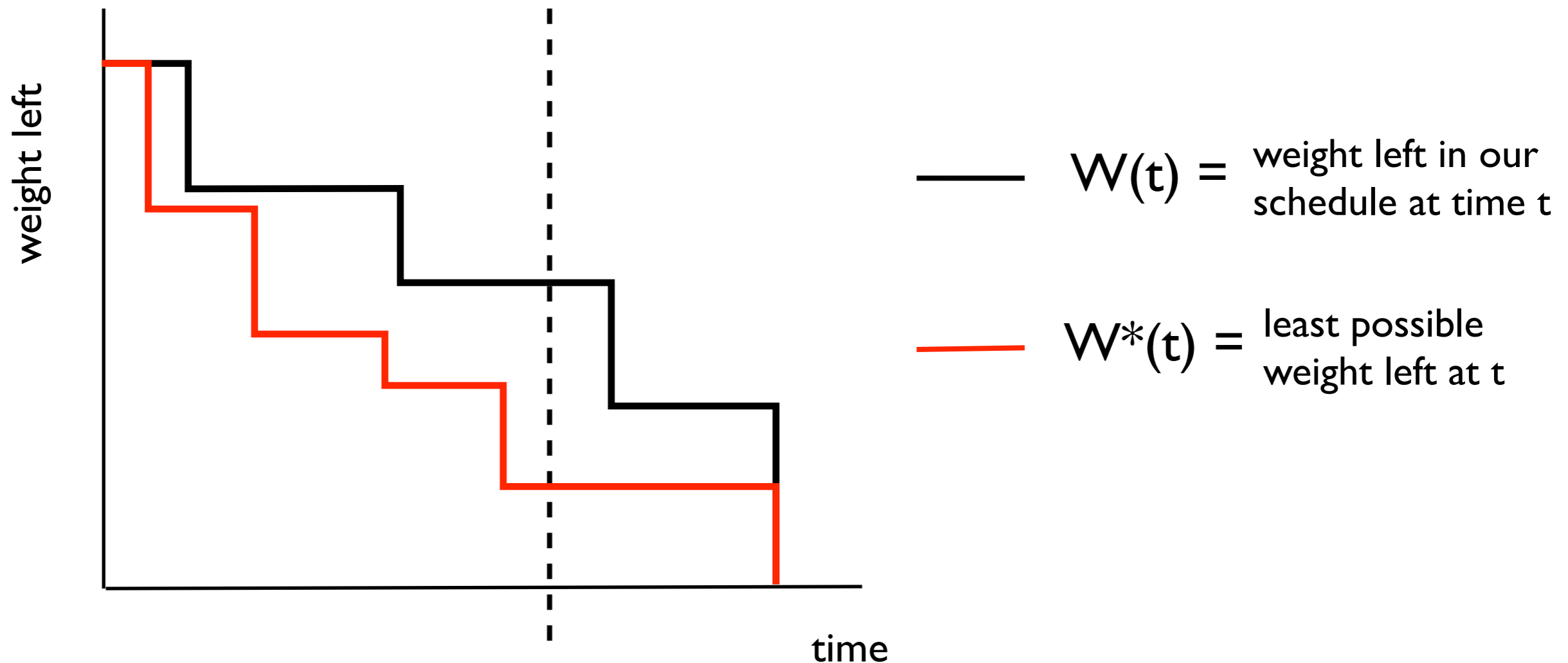
# What can go wrong?



# What can go wrong?



# What can go wrong?



Thm.

The competitive ratio of schedule  $W$  is  
 $\max W(t) / W^*(t)$  over all  $t$

# Algorithm for universal scheduling

```
 $\pi$  = empty list  
for  $i = 0$  to  $k$   
    find set of jobs  $J$  maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$   
    prepend to  $\pi$  every job in  $J \setminus \pi$   
return  $\pi$ 
```

# Algorithm for universal scheduling

$\pi$  = empty list  
for  $i = 0$  to  $k$   
    find set of jobs  $J$  maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$   
    prepend to  $\pi$  every job in  $J \setminus \pi$   
return  $\pi$

Thm.

The algorithm produces a schedule  
such that  $W(t) < 4 W^*(t)$  for all  $t$

# Algorithm for universal scheduling

$\pi$  = empty list  
for  $i = 0$  to  $k$   
    find set of jobs  $J$  maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$   
    prepend to  $\pi$  every job in  $J \setminus \pi$   
return  $\pi$

Thm.

The algorithm produces a schedule such that  $W(t) < 4 W^*(t)$  for all  $t$

Coro.

The universal schedules produced by the algorithm are **4 competitive**

```
 $\pi$  = empty list
for i = 0 to k
  find set of jobs J maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$ 
  prepend to  $\pi$  every job in  $J \setminus \pi$ 
return  $\pi$ 
```

$\pi$  = empty list  
for  $i = 0$  to  $k$   
    find set of jobs  $J$  maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$   
    prepend to  $\pi$  every job in  $J \setminus \pi$   
return  $\pi$

$J_0$

$\pi$  = empty list

for  $i = 0$  to  $k$

find set of jobs  $J$  maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$

prepend to  $\pi$  every job in  $J \setminus \pi$

return  $\pi$

$J_0$

$w(J_0) \leq 1$

$\pi$  = empty list  
for  $i = 0$  to  $k$   
    find set of jobs  $J$  maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$   
    prepend to  $\pi$  every job in  $J \setminus \pi$   
return  $\pi$

$J_1 \setminus J_0$	$J_0$
$w(J_1) \leq 2$	$w(J_0) \leq 1$

$\pi$  = empty list

for  $i = 0$  to  $k$

find set of jobs  $J$  maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$

prepend to  $\pi$  every job in  $J \setminus \pi$

return  $\pi$

$$J_2 \setminus J_1 \cup J_0$$

$$w(J_2) \leq 4$$

$$J_1 \setminus J_0$$

$$w(J_1) \leq 2$$

$$J_0$$

$$w(J_0) \leq 1$$

$\pi$  = empty list

for  $i = 0$  to  $k$

find set of jobs  $J$  maximizing  $p(J)$  s.t.  $w(J) \leq 2^i$

prepend to  $\pi$  every job in  $J \setminus \pi$

return  $\pi$

$$J_i \setminus J_{i-1} \cup \dots \cup J_0$$

$$w(J_i) \leq 2^i$$

...

$$J_2 \setminus J_1 \cup J_0$$

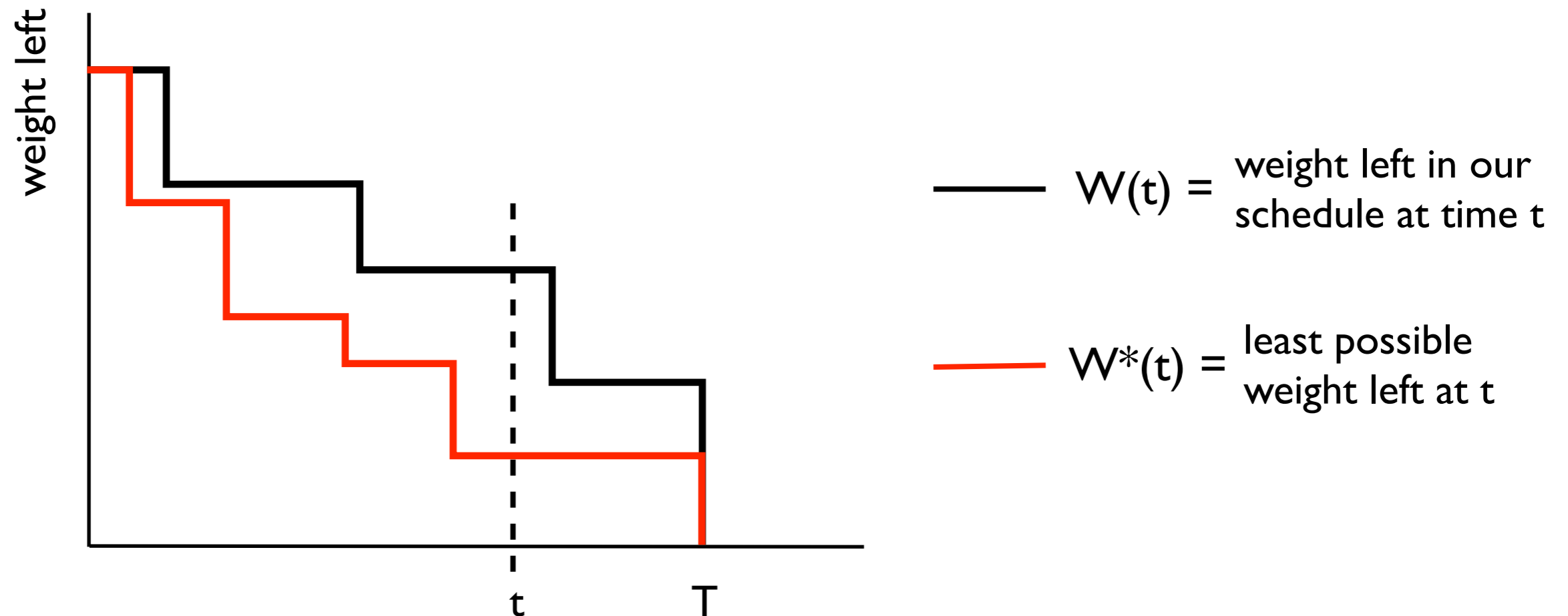
$$w(J_2) \leq 4$$

$$J_1 \setminus J_0$$

$$w(J_1) \leq 2$$

$$J_0$$

$$w(J_0) \leq 1$$



$$J_i \setminus J_{i-1} \cup \dots \cup J_0$$

...

$$J_2 \setminus J_1 \cup J_0$$

$$J_1 \setminus J_0$$

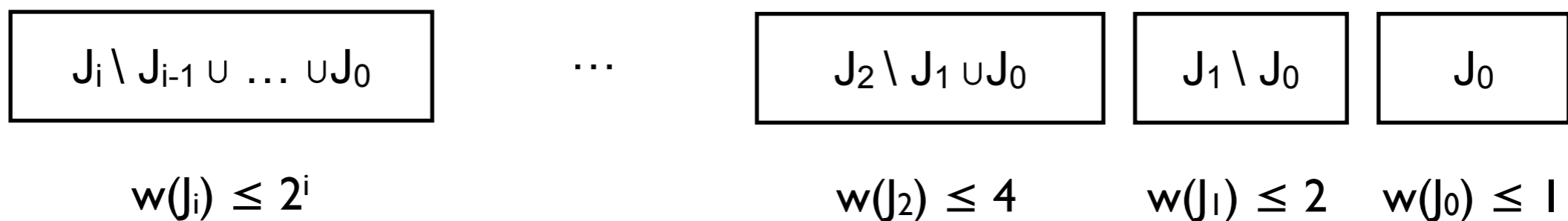
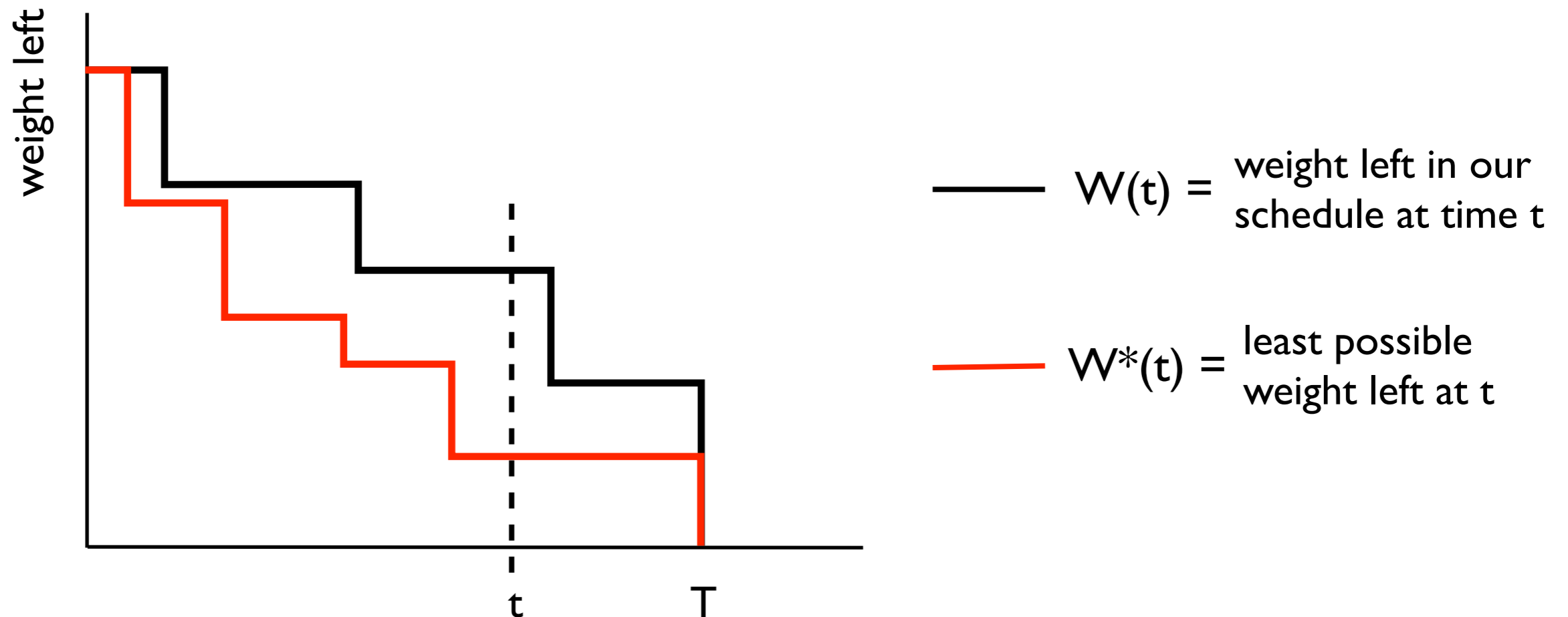
$$J_0$$

$$w(J_i) \leq 2^i$$

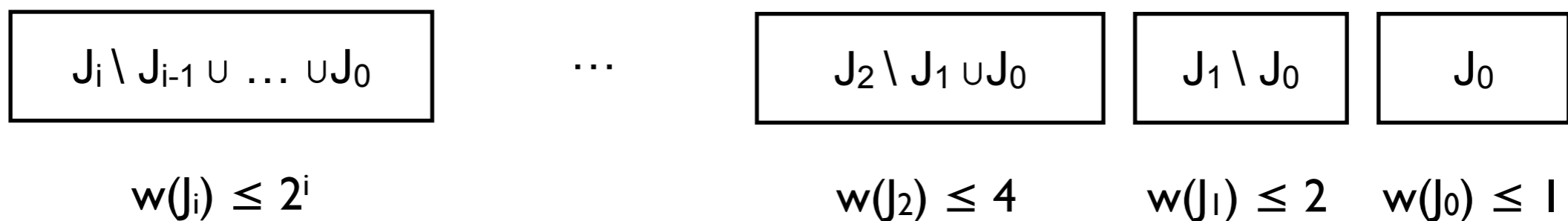
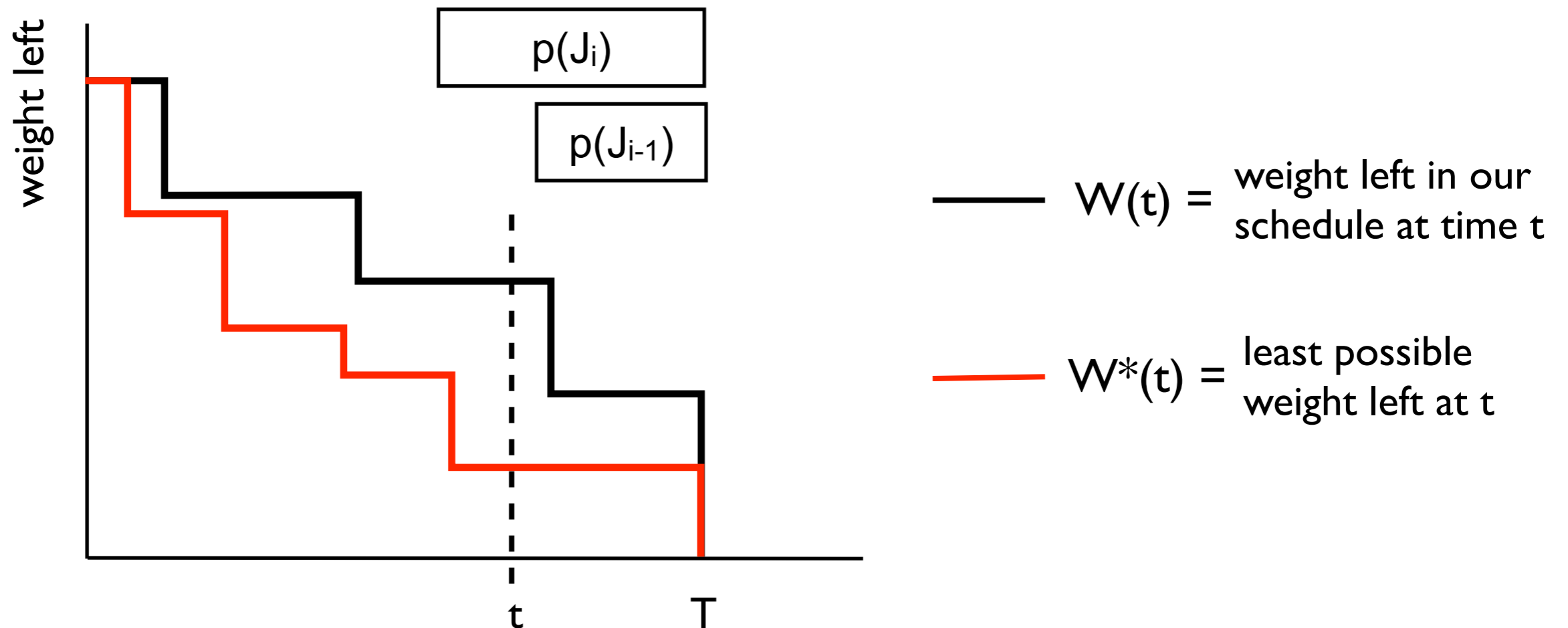
$$w(J_2) \leq 4$$

$$w(J_1) \leq 2$$

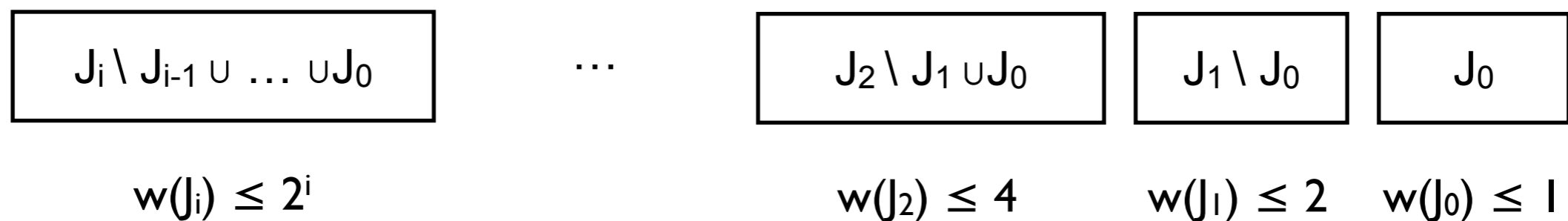
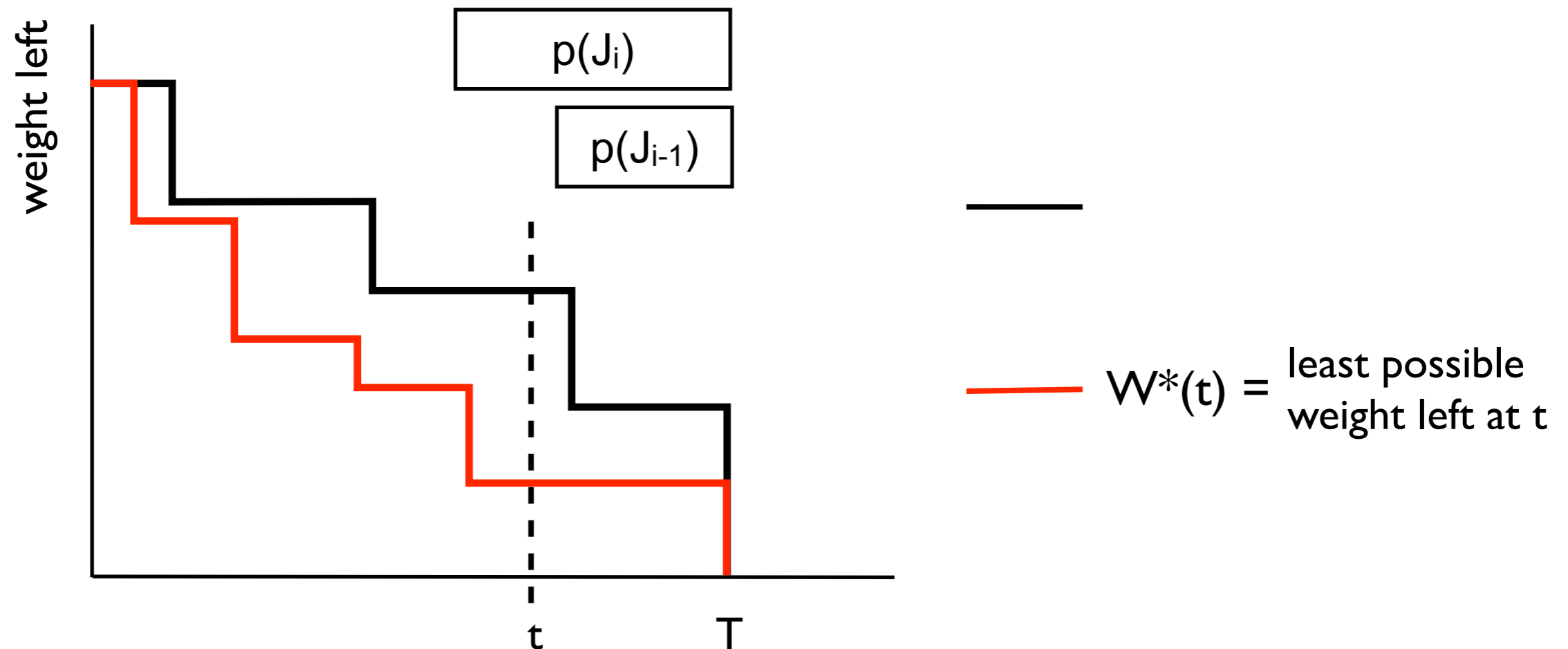
$$w(J_0) \leq 1$$



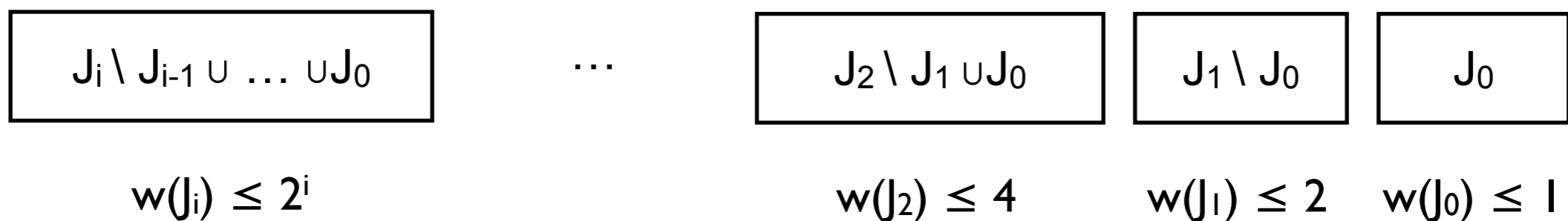
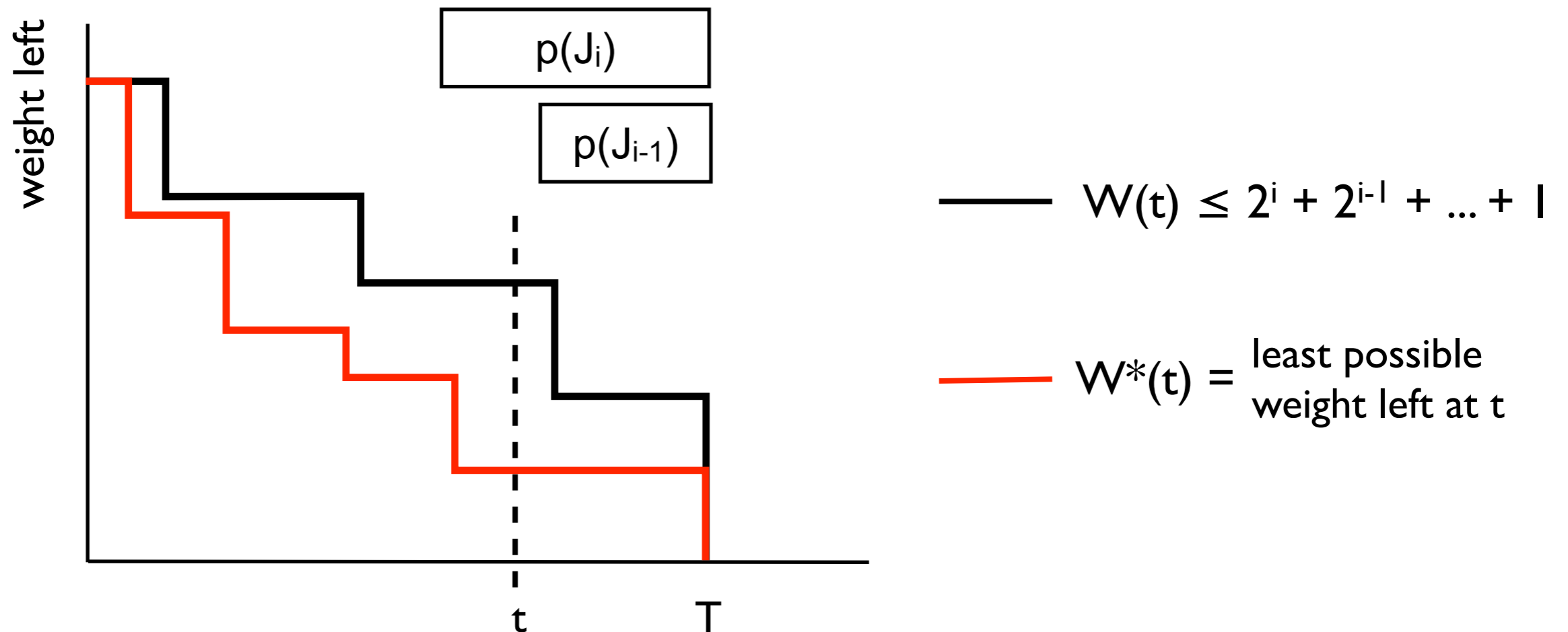
Let  $i$  be such that  $p(J_i) \geq T - t$  and  $p(J_{i-1}) < T - t$



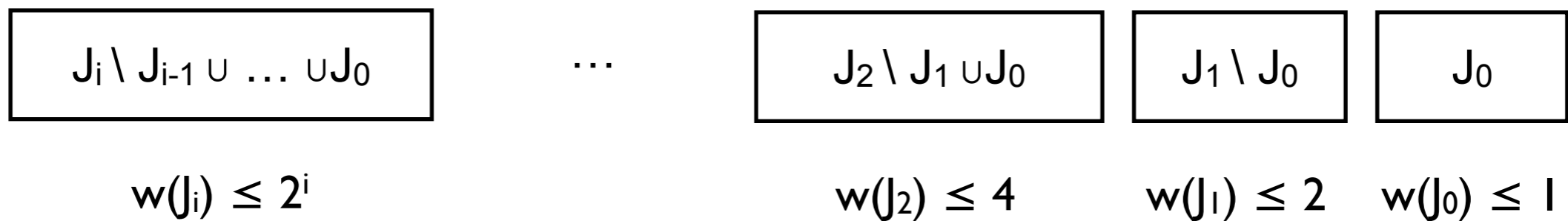
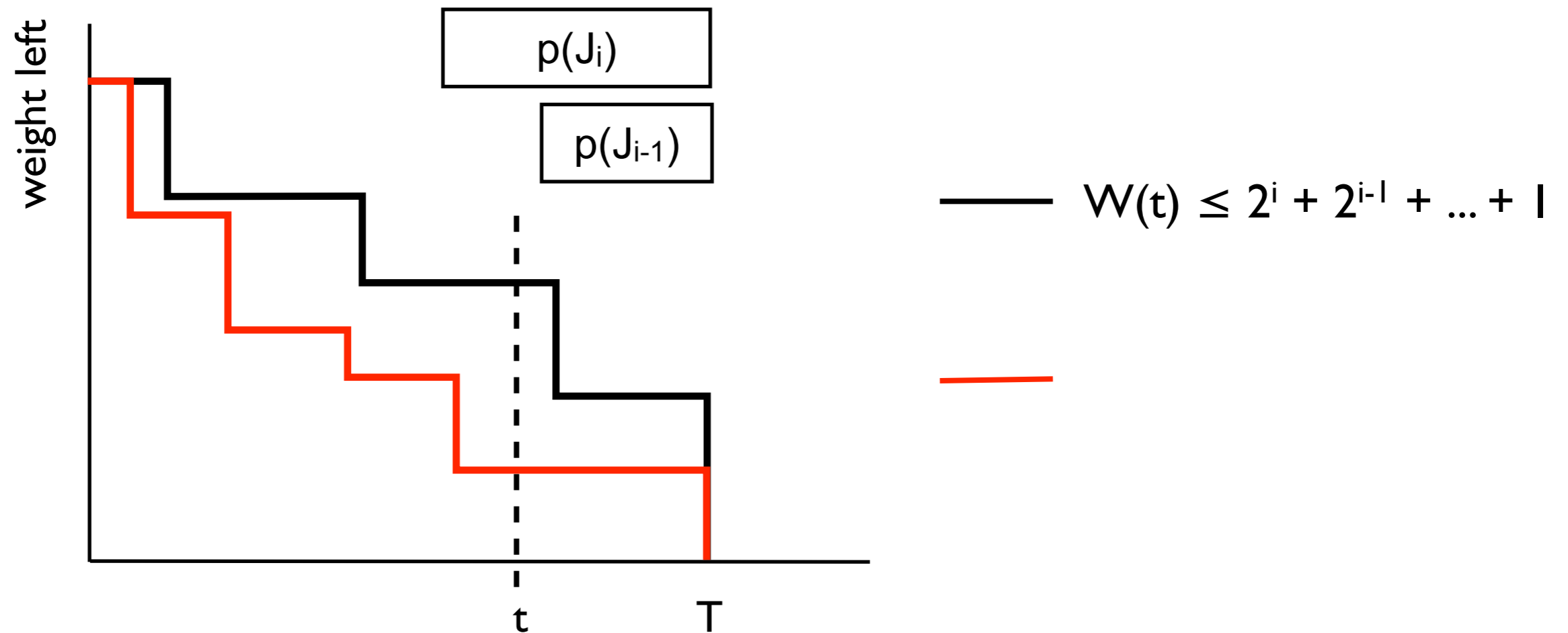
Let  $i$  be such that  $p(J_i) \geq T - t$  and  $p(J_{i-1}) < T - t$



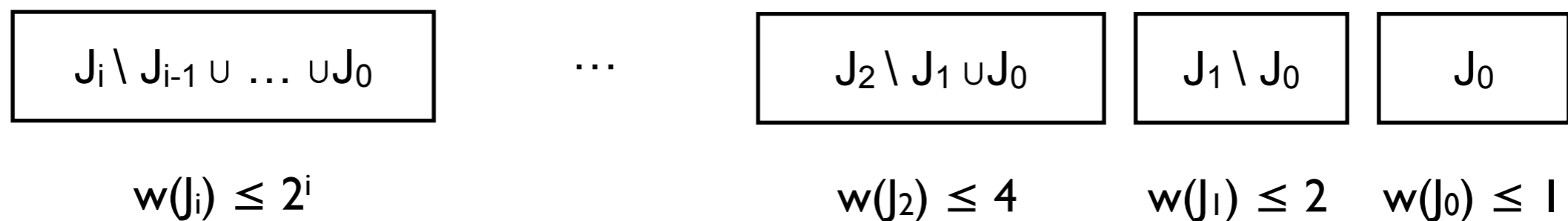
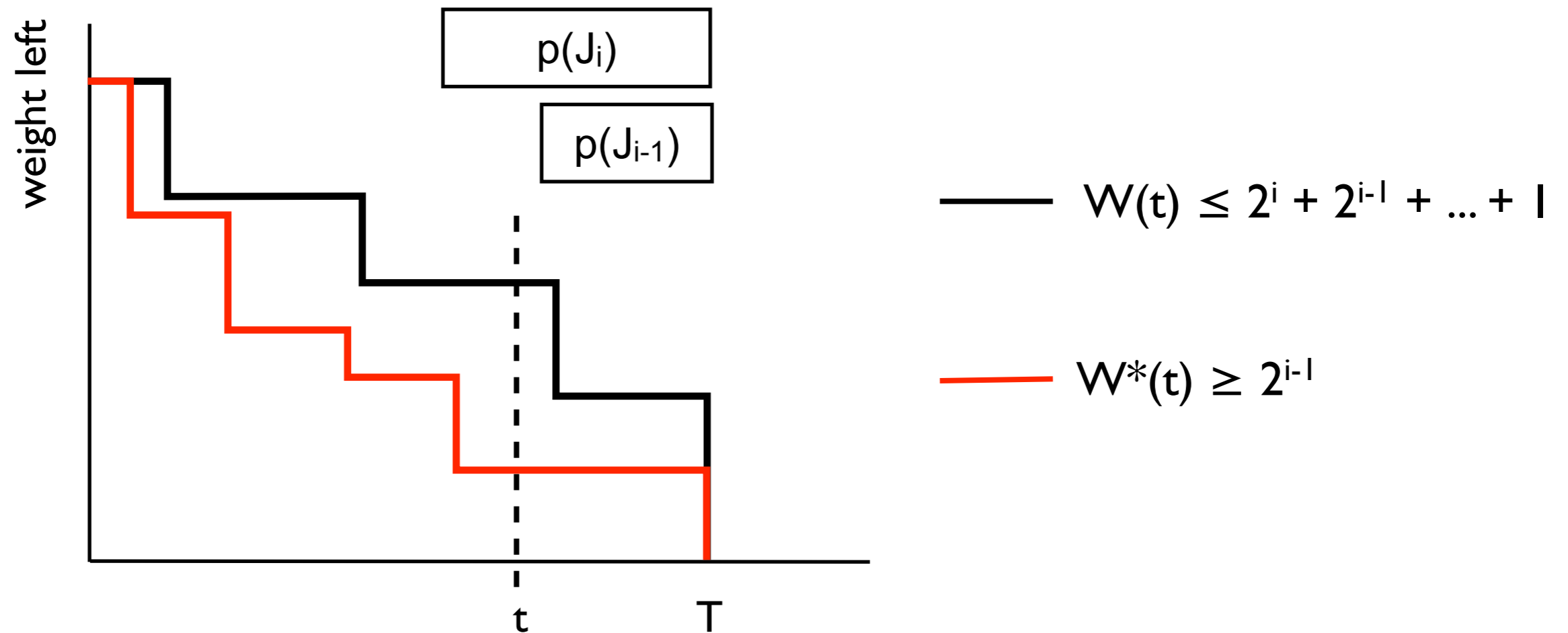
Let  $i$  be such that  $p(J_i) \geq T - t$  and  $p(J_{i-1}) < T - t$



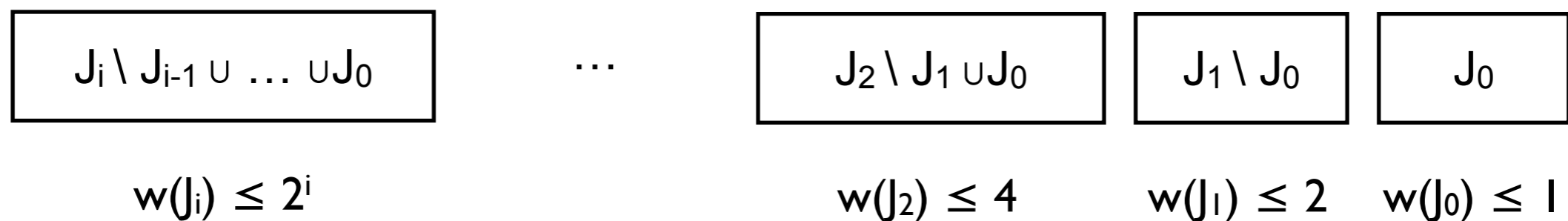
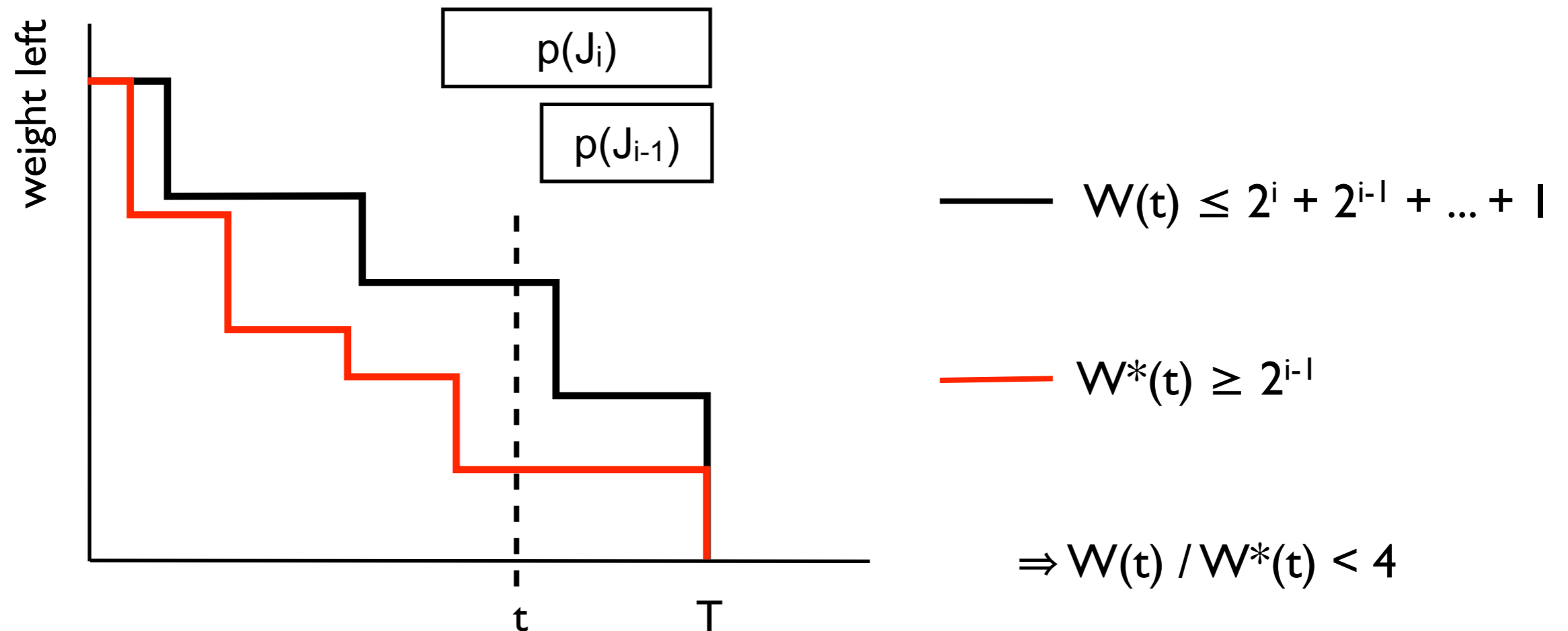
Let  $i$  be such that  $p(J_i) \geq T - t$  and  $p(J_{i-1}) < T - t$



Let  $i$  be such that  $p(J_i) \geq T - t$  and  $p(J_{i-1}) < T - t$



Let  $i$  be such that  $p(J_i) \geq T - t$  and  $p(J_{i-1}) < T - t$



Let  $i$  be such that  $p(J_i) \geq T - t$  and  $p(J_{i-1}) < T - t$



# Lowerbounds

There are instances where for every universal schedule, there is a machine behavior under which the schedule is  $4-\epsilon$  competitive

There are instances where for every universal schedule, there is a machine behavior under which the schedule is  $4-\epsilon$  competitive

If we introduce release date then there are instances where we must be  $\Omega(\log n / \log \log n)$  competitive



## Randomized schedules

Randomized schedules

Precedence constraints

Randomized schedules

Precedence constraints

Release dates

Randomized schedules

Precedence constraints

Release dates

Offline version



# Open problems

## Parallel machines

Parallel machines

Stochastic information about breakdown pattern

Parallel machines

Stochastic information about breakdown pattern

Better performance through adaptive algorithms

Parallel machines

Stochastic information about breakdown pattern

Better performance through adaptive algorithms

Optimal universal algorithms for a given instance

Parallel machines

Stochastic information about breakdown pattern

Better performance through adaptive algorithms

~~Optimal universal algorithms for a given instance~~



Thanks for  
your attention!