



max planck institut
informatik

Julián Mestre

Universal Scheduling

L. Epstein (Haifa University)

A. Levin (Technion)

N. Megow (MPI-INF)

A. Marchetti-Spaccamela (La Sapienza)

M. Skutella (TU Berlin)

L. Stougie (TU Eindhoven)

Universal Scheduling



Universal Scheduling

- Traditional scheduling assumes **ideal machines**



Universal Scheduling

- Traditional scheduling assumes **ideal machines**



Universal Scheduling

- Traditional scheduling assumes **ideal machines**



Universal Scheduling

- Traditional scheduling assumes **ideal machines**
- In practice though machines may **vary speed** unpredictably or may **break down** all together



Universal Scheduling

- 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**:



Universal Scheduling

- 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



Universal Scheduling

- 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 fix a job sequence in advance



Universal Scheduling

- 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 fix a job sequence in advance
 - Compare again best offline schedule



Universal Scheduling

- 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 fix a job sequence in advance
 - Compare again best offline schedule

“Competitive ratio”



Scheduling on a single machine



Scheduling on a single machine

- Jobs have **weight** and **processing time**.



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 **proc. time/weight**



Scheduling on a single machine

- Jobs have **weight** and **processing time**.
- Objective: minimize **weighted average completion time**
- Smith's rule: sort by **proc. time/weight**
- Optimal solution with an ideal machine can be arbitrarily bad when using unreliable machine



Scheduling on a single machine

- Jobs have **weight** and **processing time**.
- Objective: minimize **weighted average completion time**
- Smith's rule: sort by **proc. time/weight**
- Optimal solution with an ideal machine can be arbitrarily bad when using unreliable machine

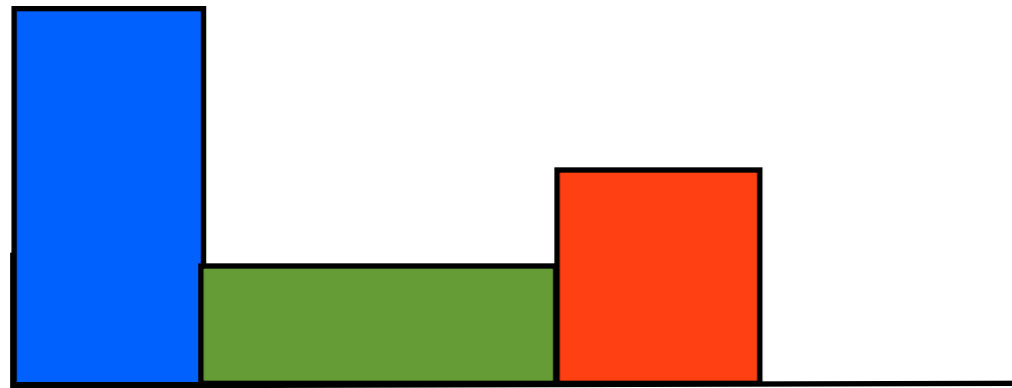
Today: Tight bounds for the competitive ratio
of universal schedules



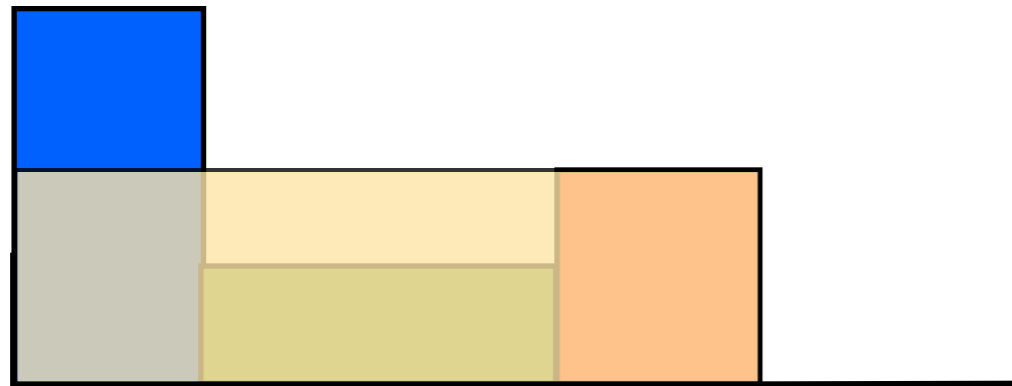
A pictorial view of the objective function



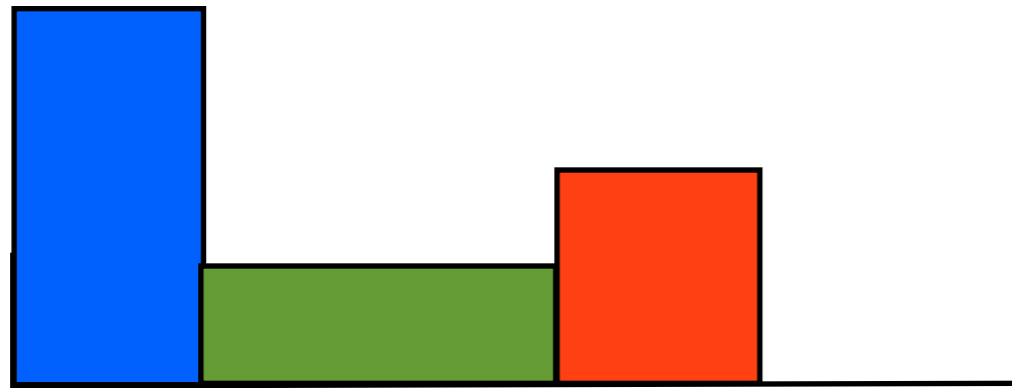
A pictorial view of the objective function



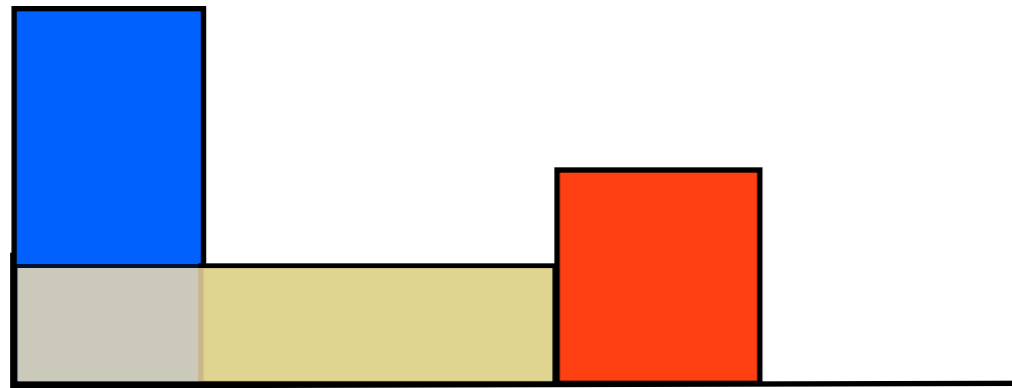
A pictorial view of the objective function



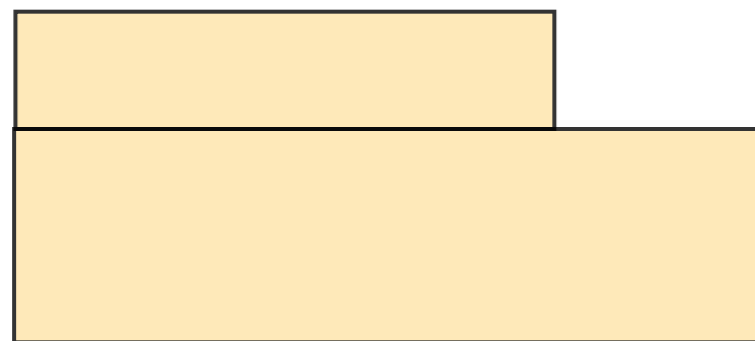
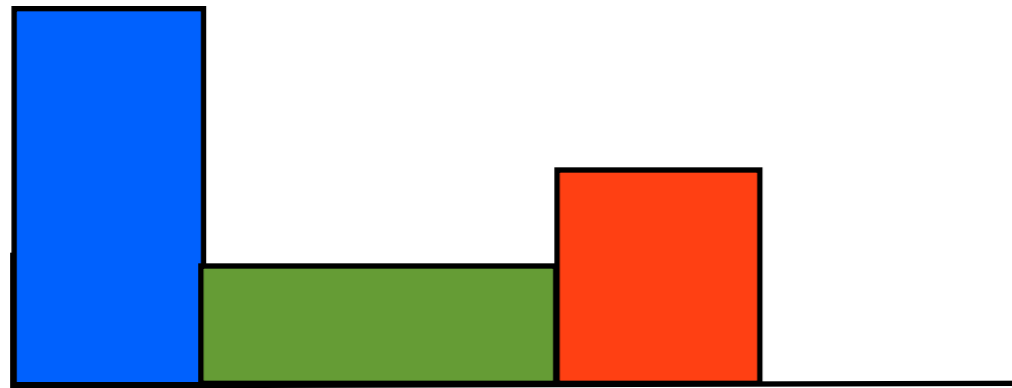
A pictorial view of the objective function



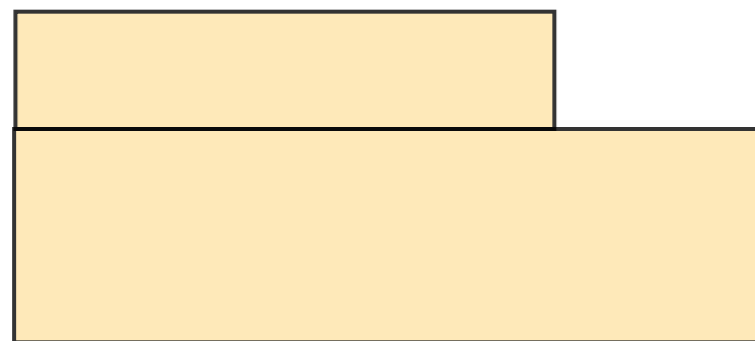
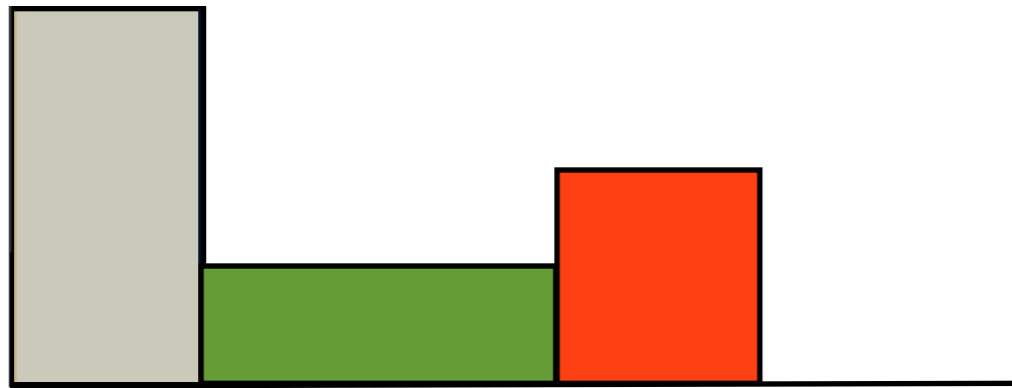
A pictorial view of the objective function



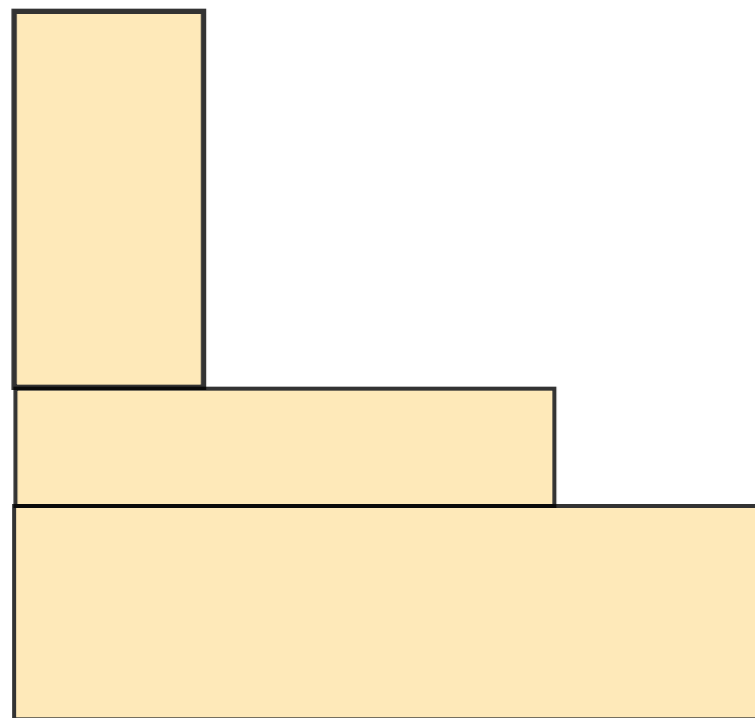
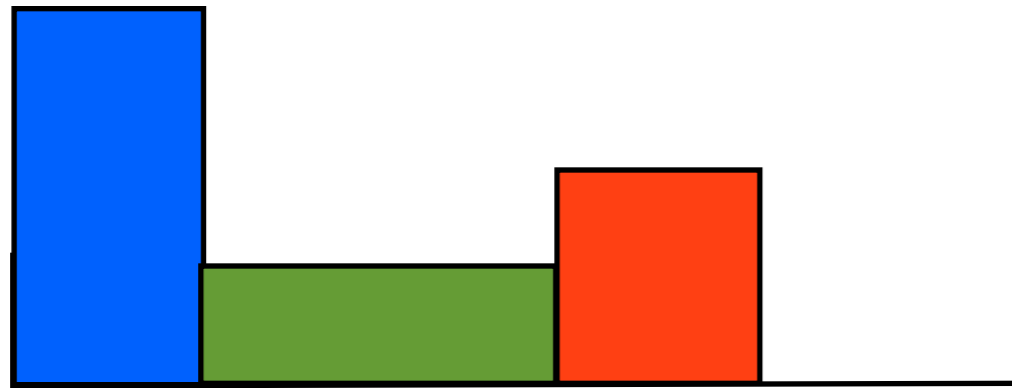
A pictorial view of the objective function



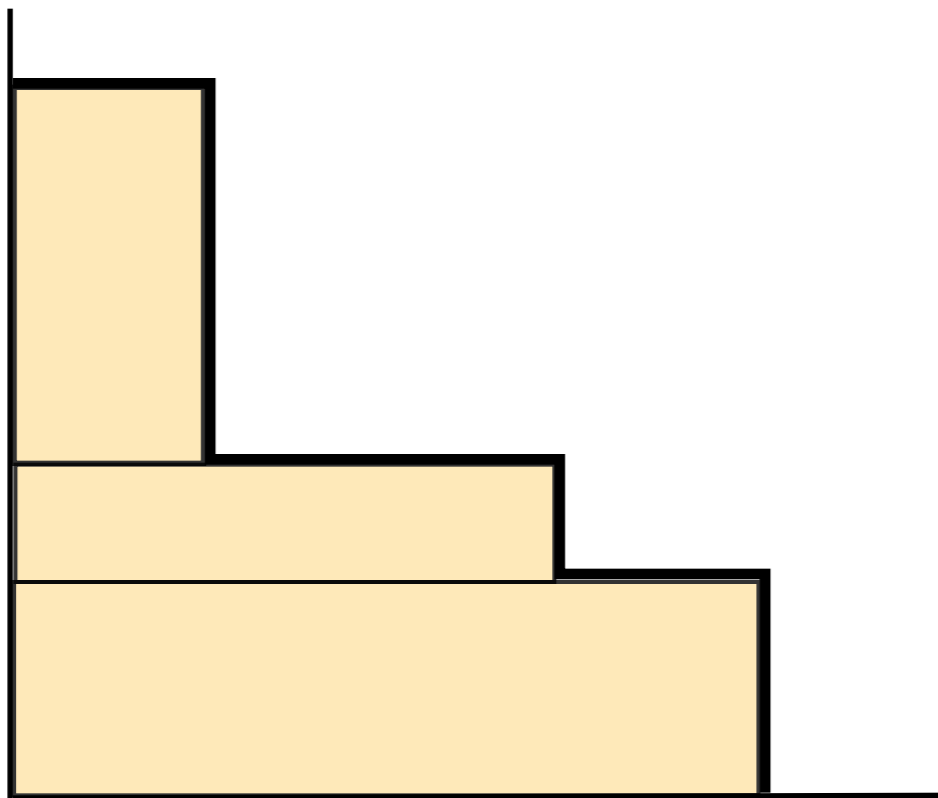
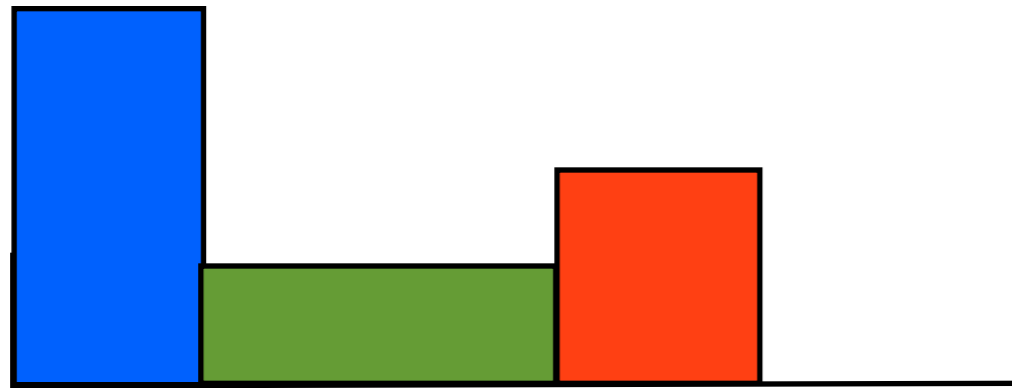
A pictorial view of the objective function



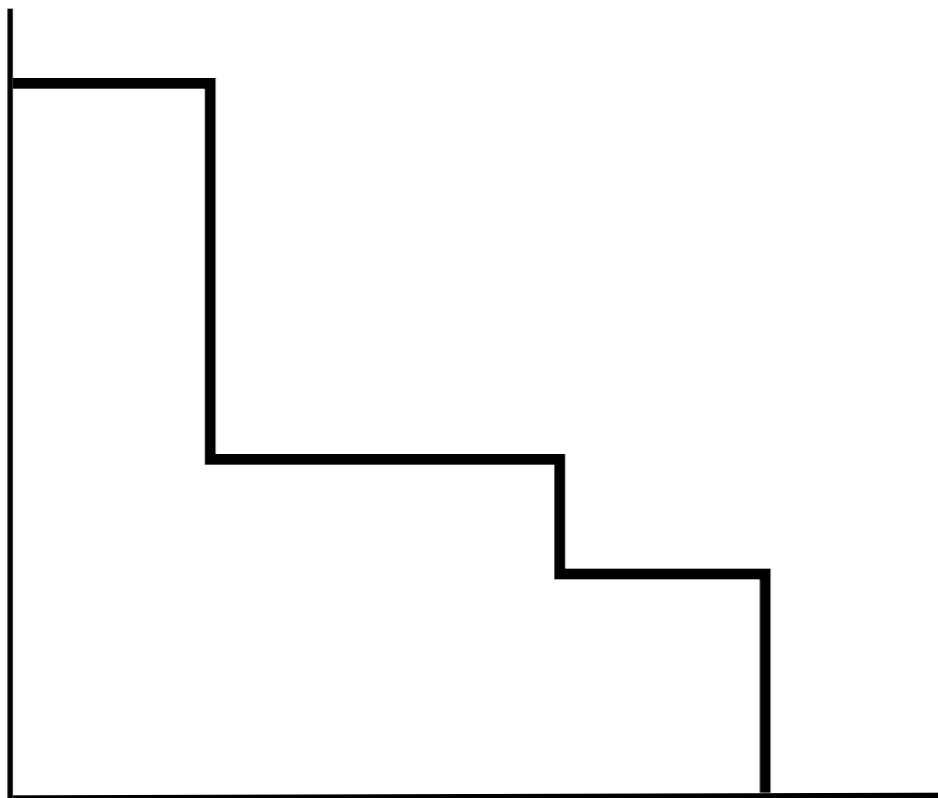
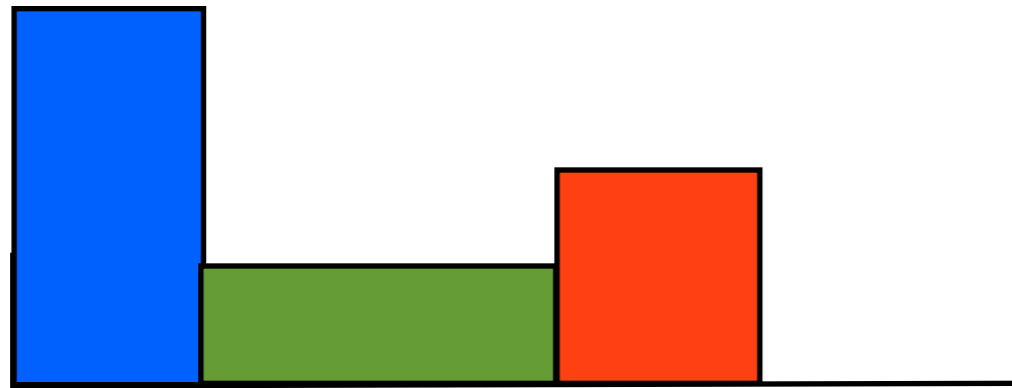
A pictorial view of the objective function



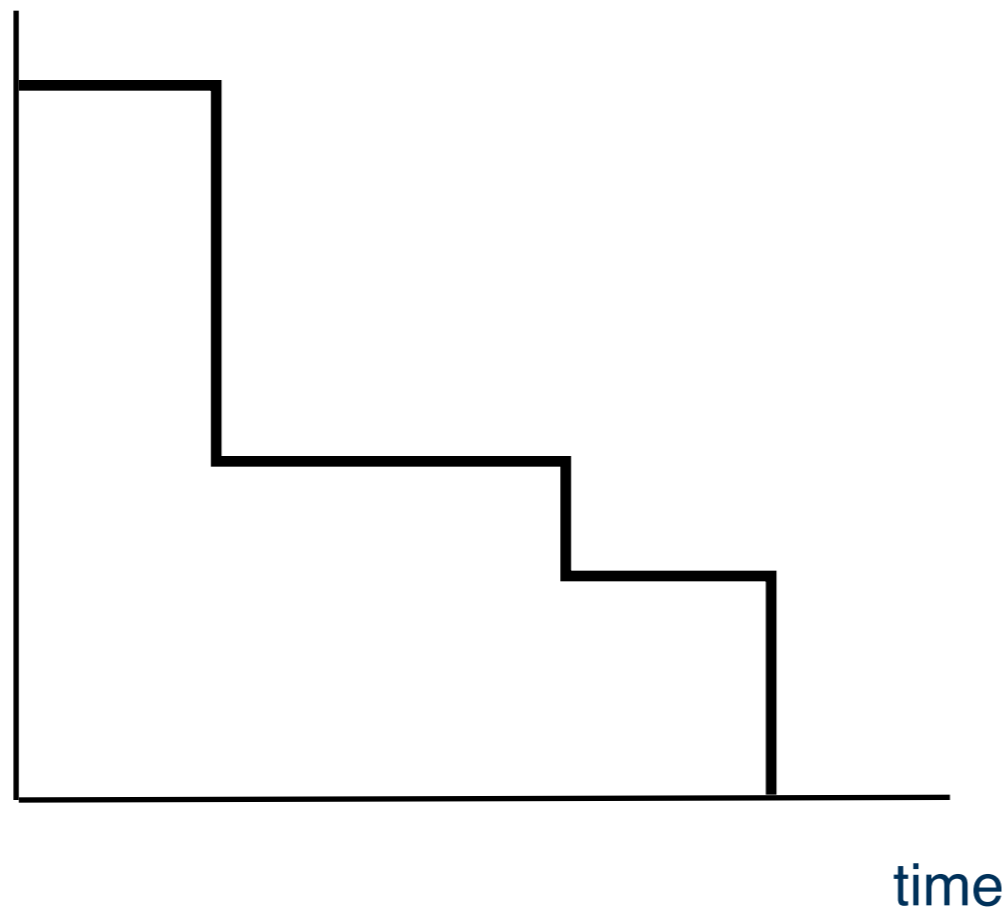
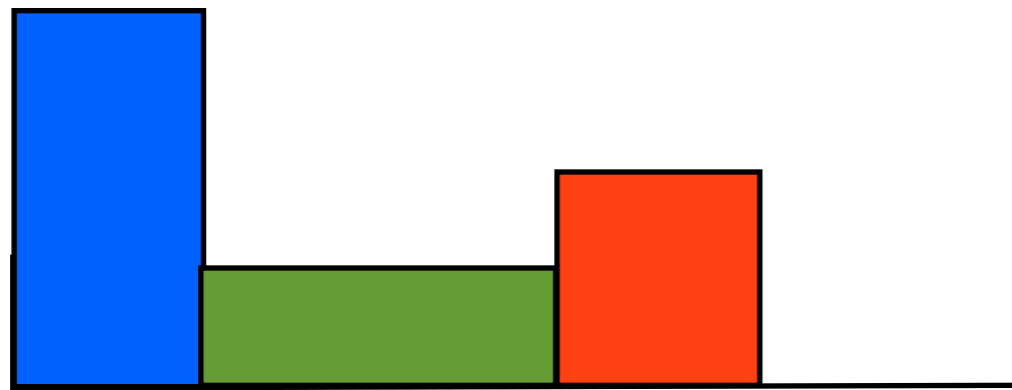
A pictorial view of the objective function



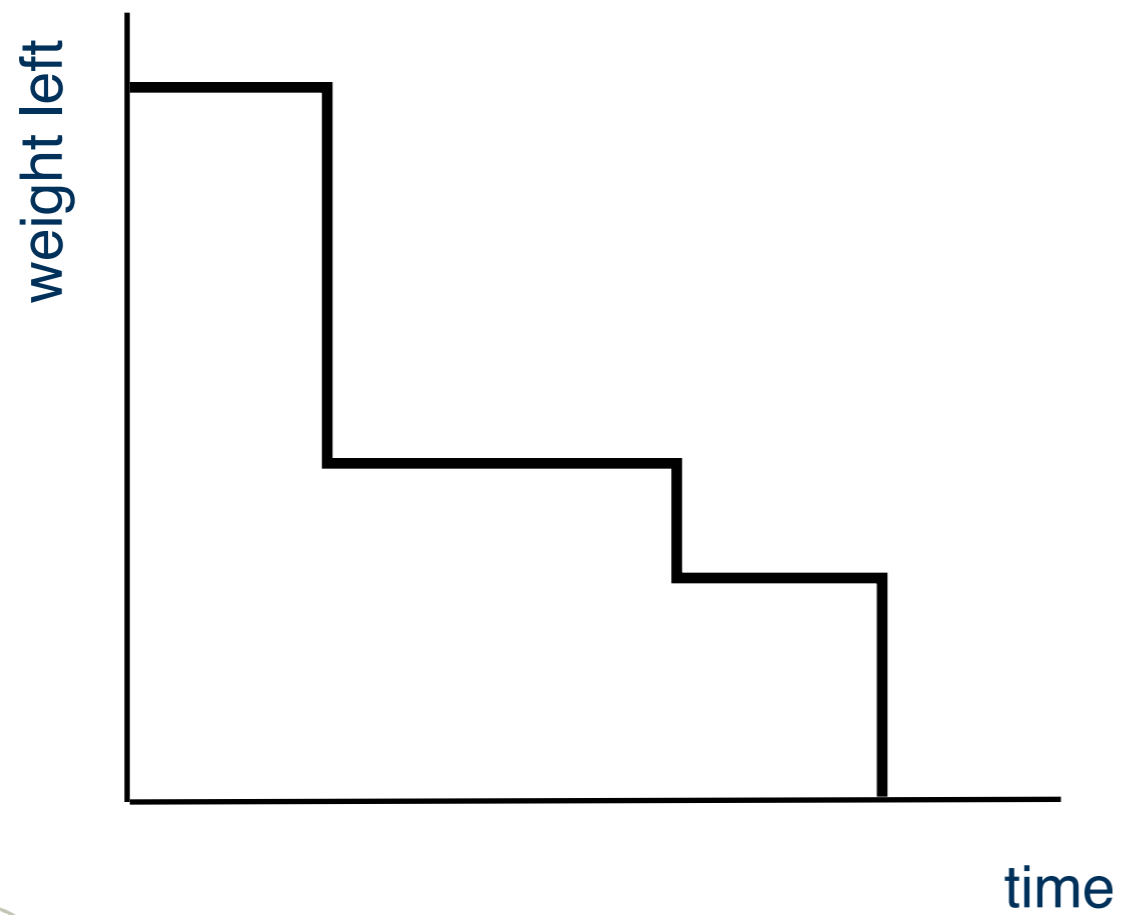
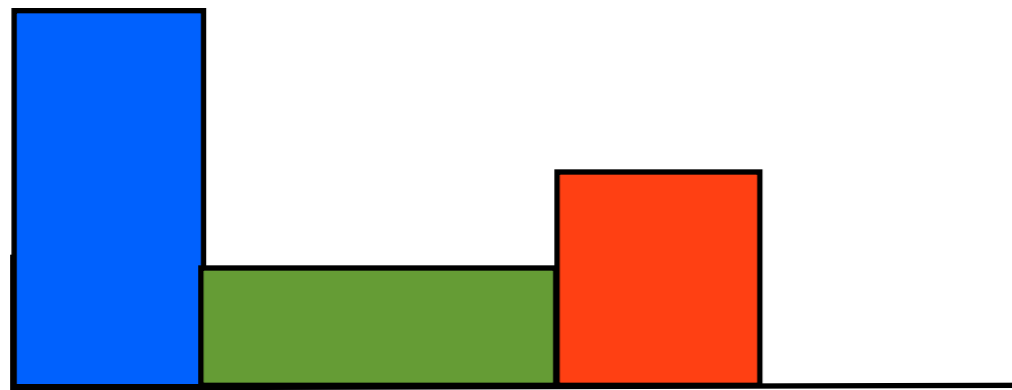
A pictorial view of the objective function



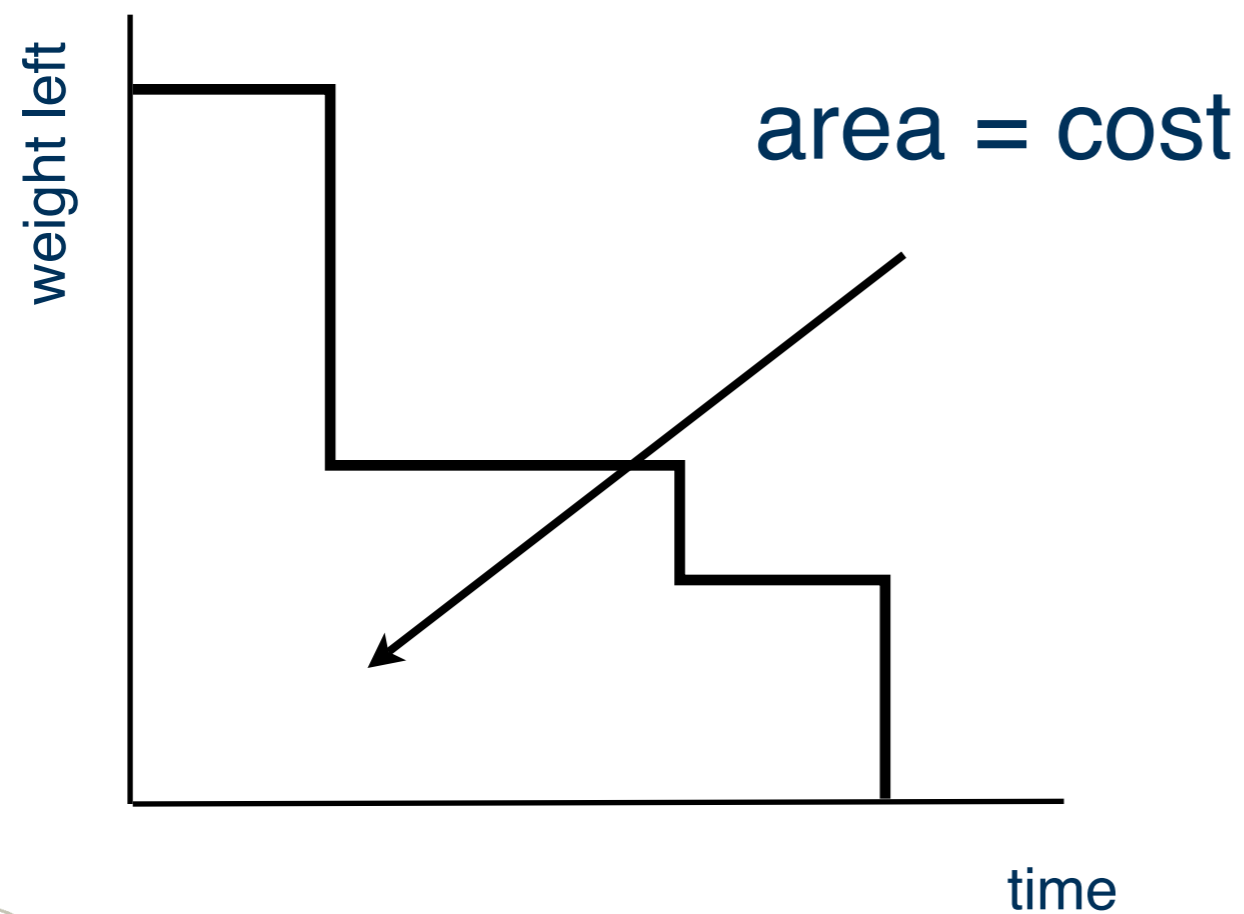
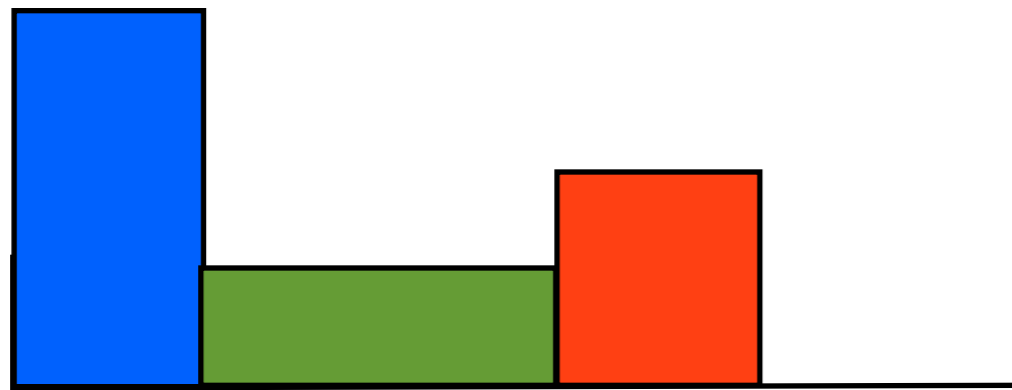
A pictorial view of the objective function



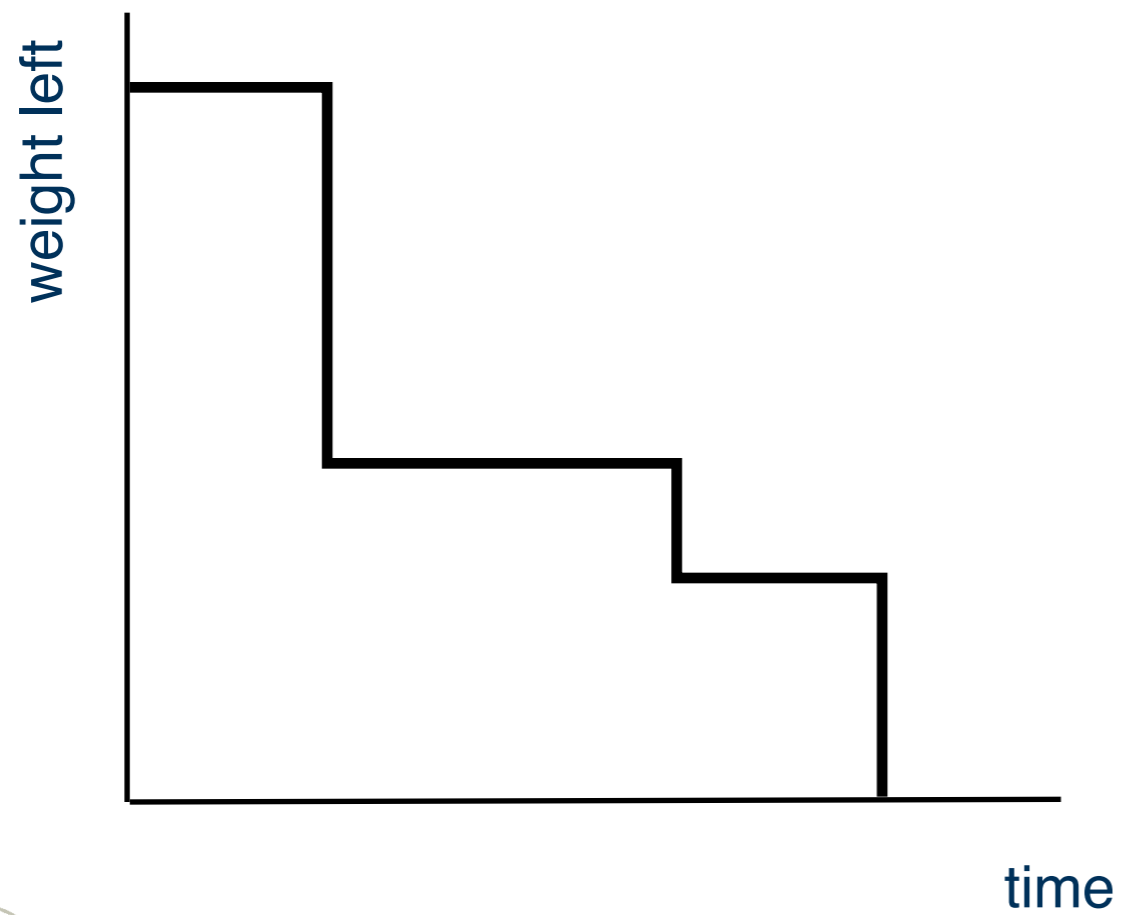
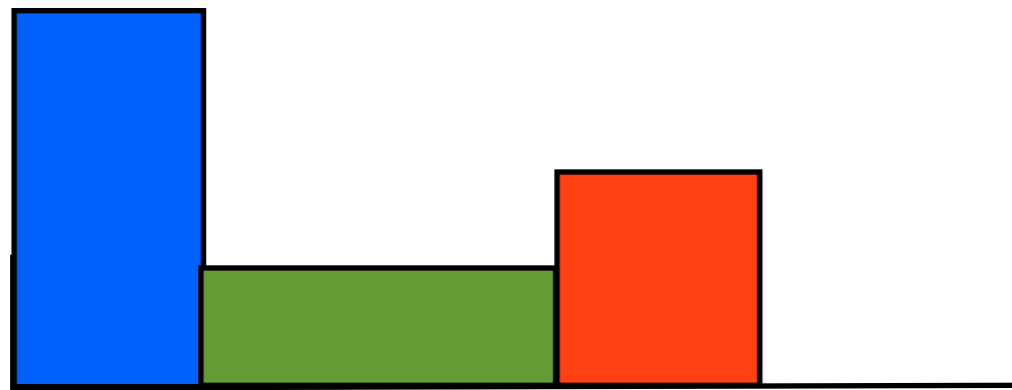
A pictorial view of the objective function



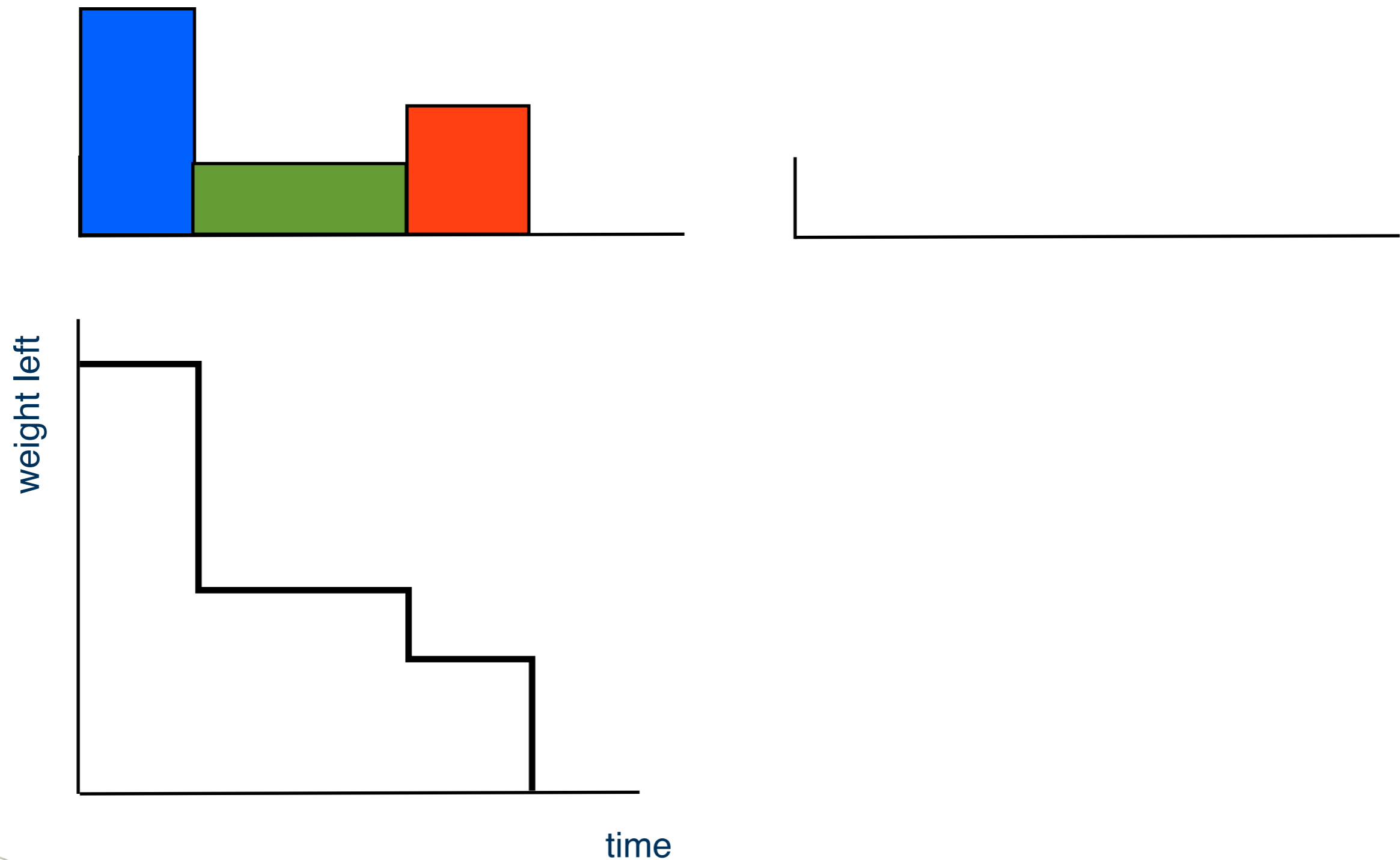
A pictorial view of the objective function



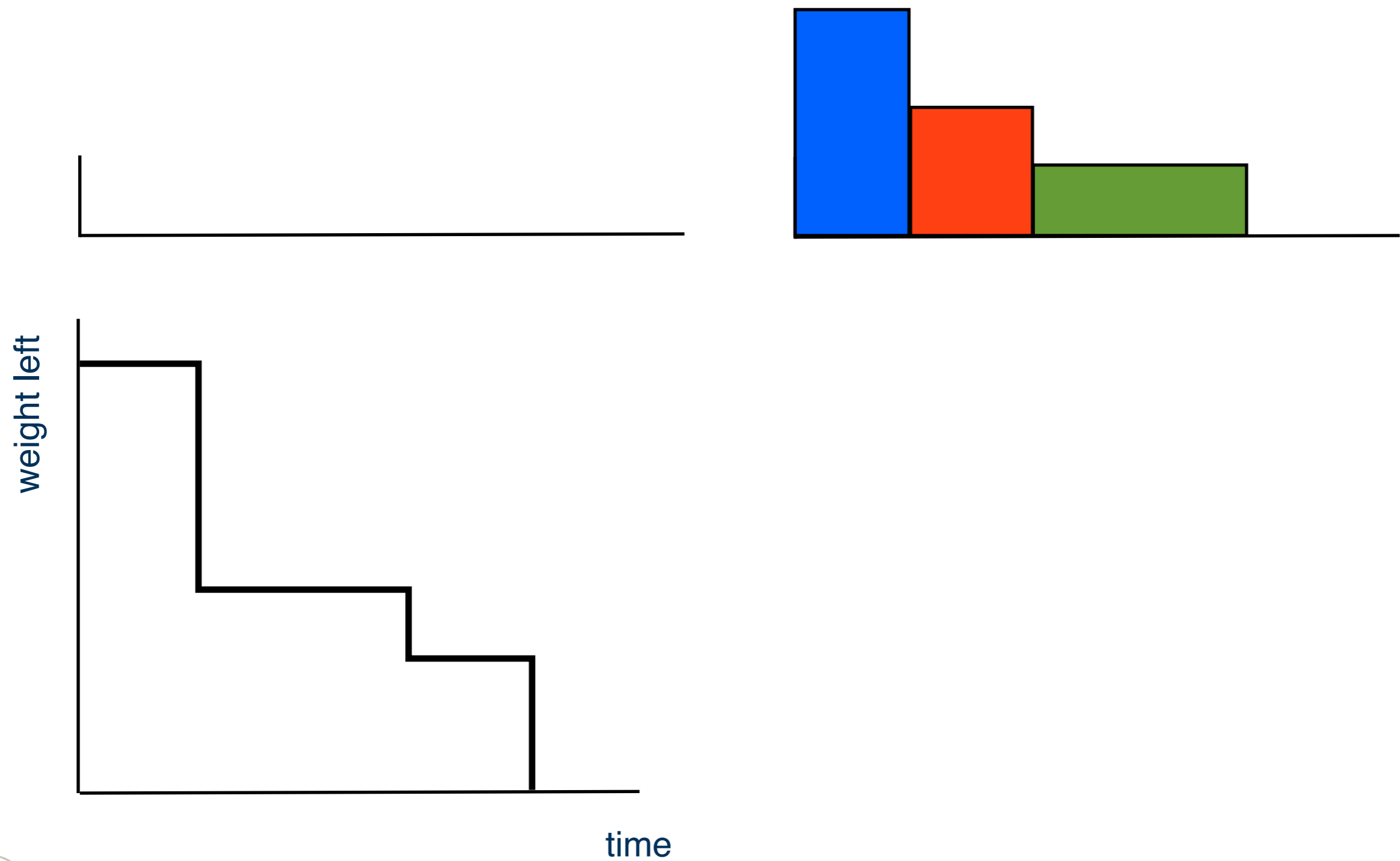
A pictorial view of the objective function



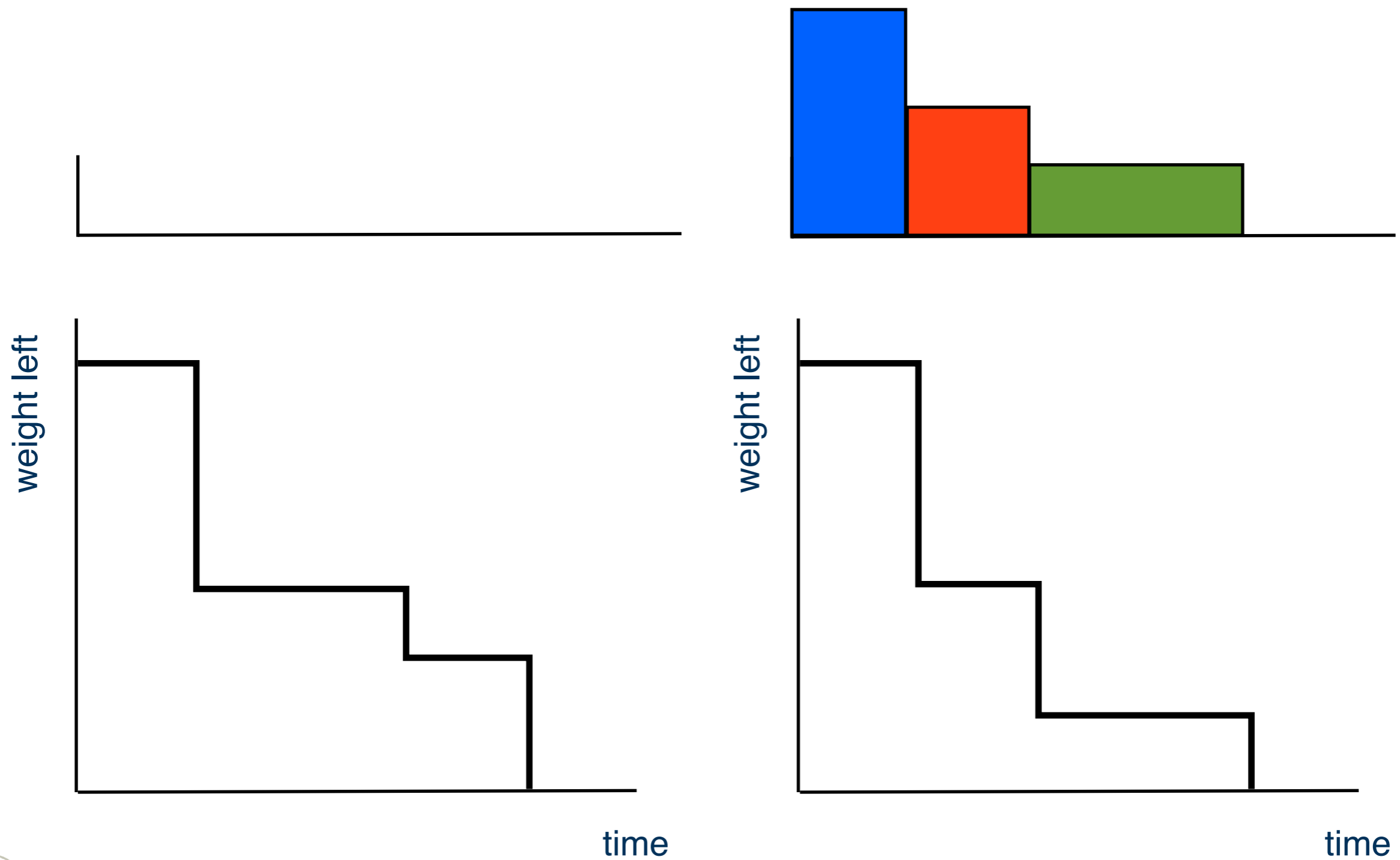
A pictorial view of the objective function



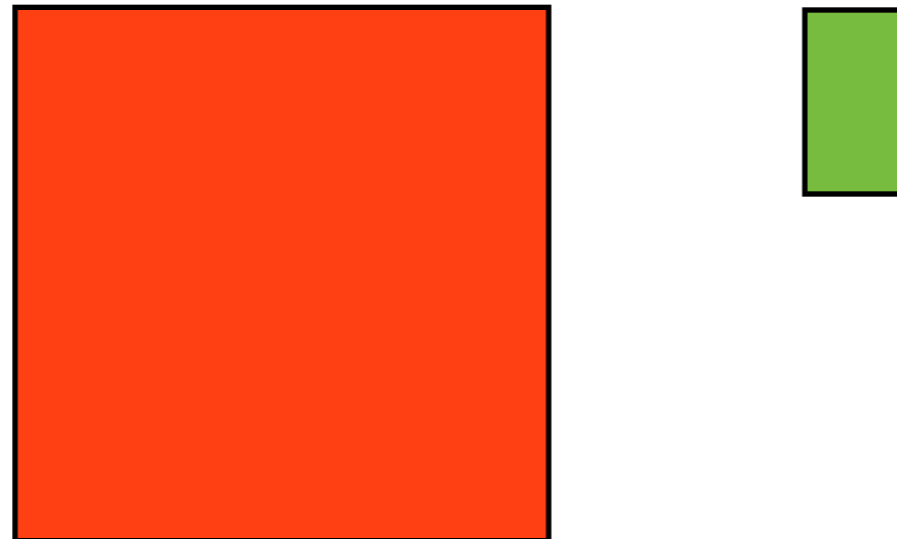
A pictorial view of the objective function



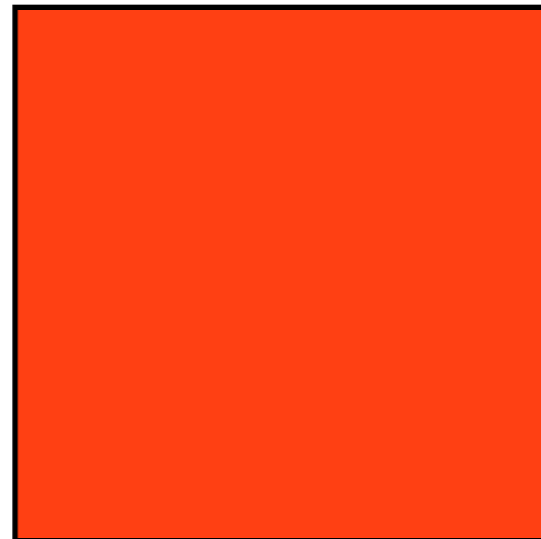
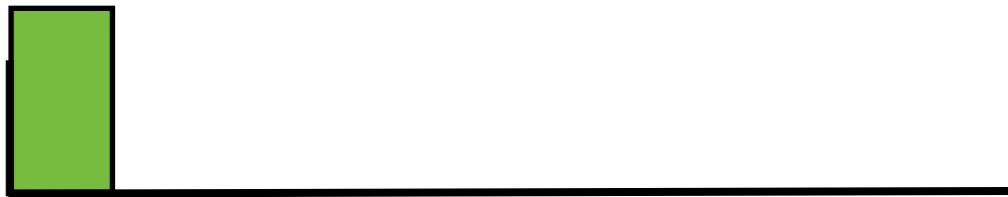
A pictorial view of the objective function



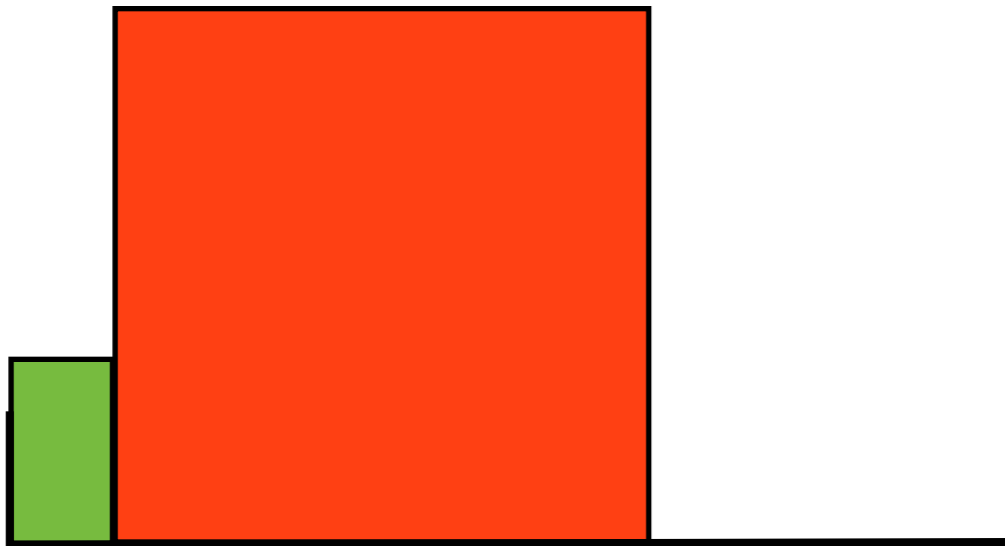
Smith's rule is not competitive



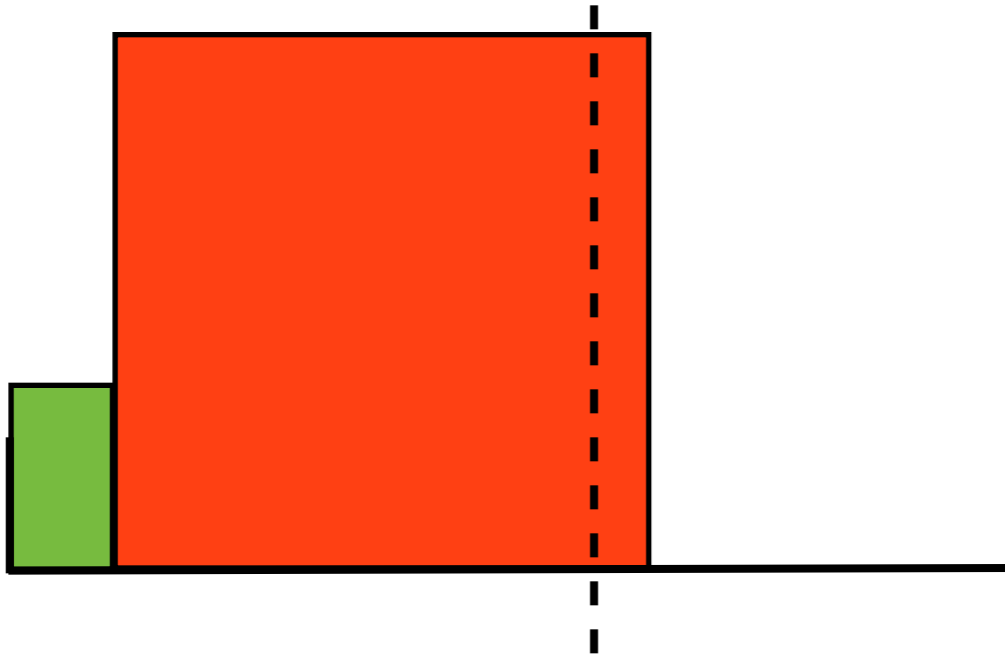
Smith's rule is not competitive



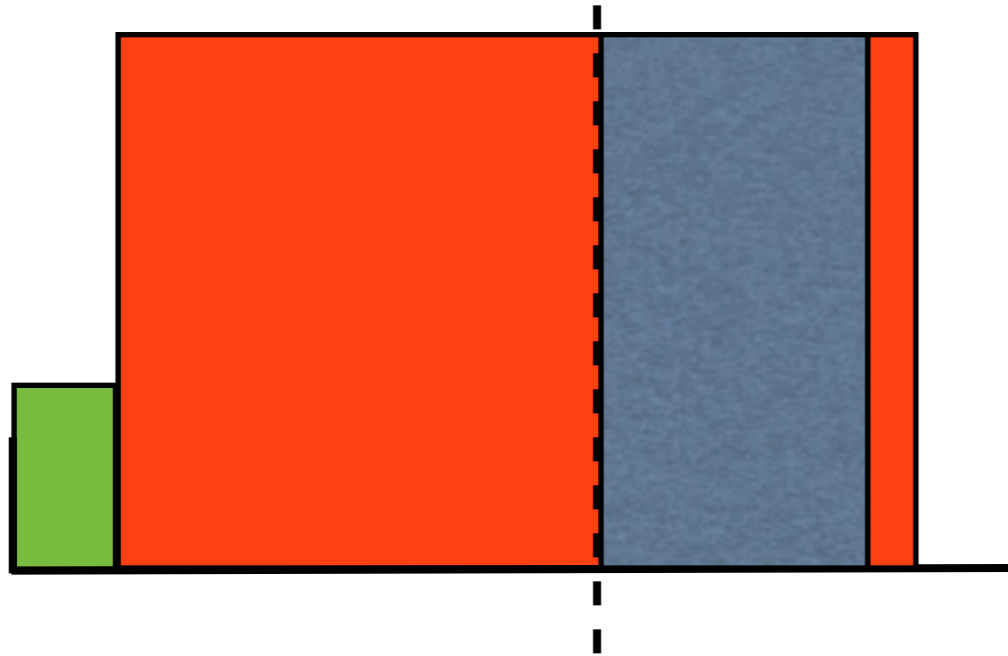
Smith's rule is not competitive



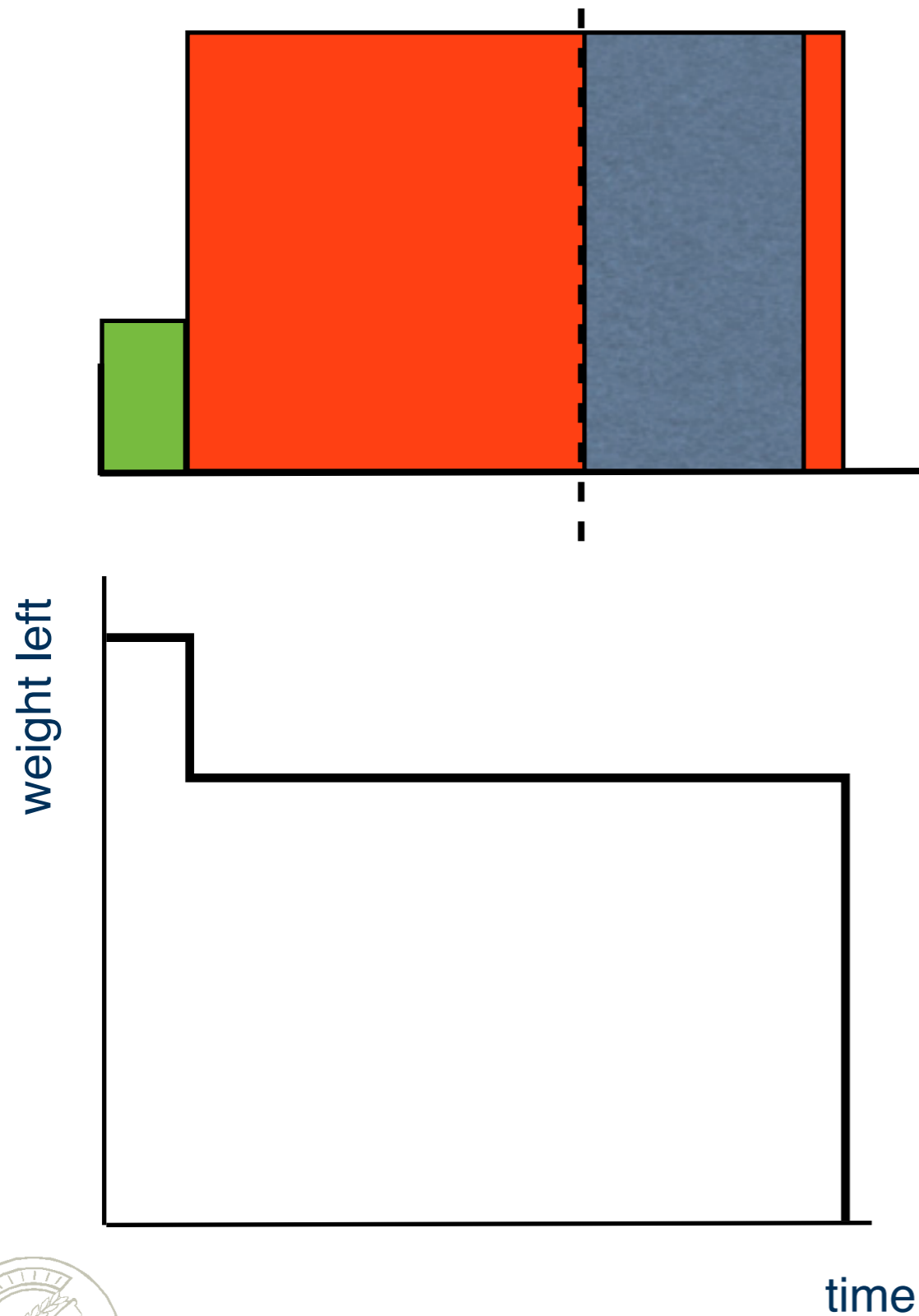
Smith's rule is not competitive



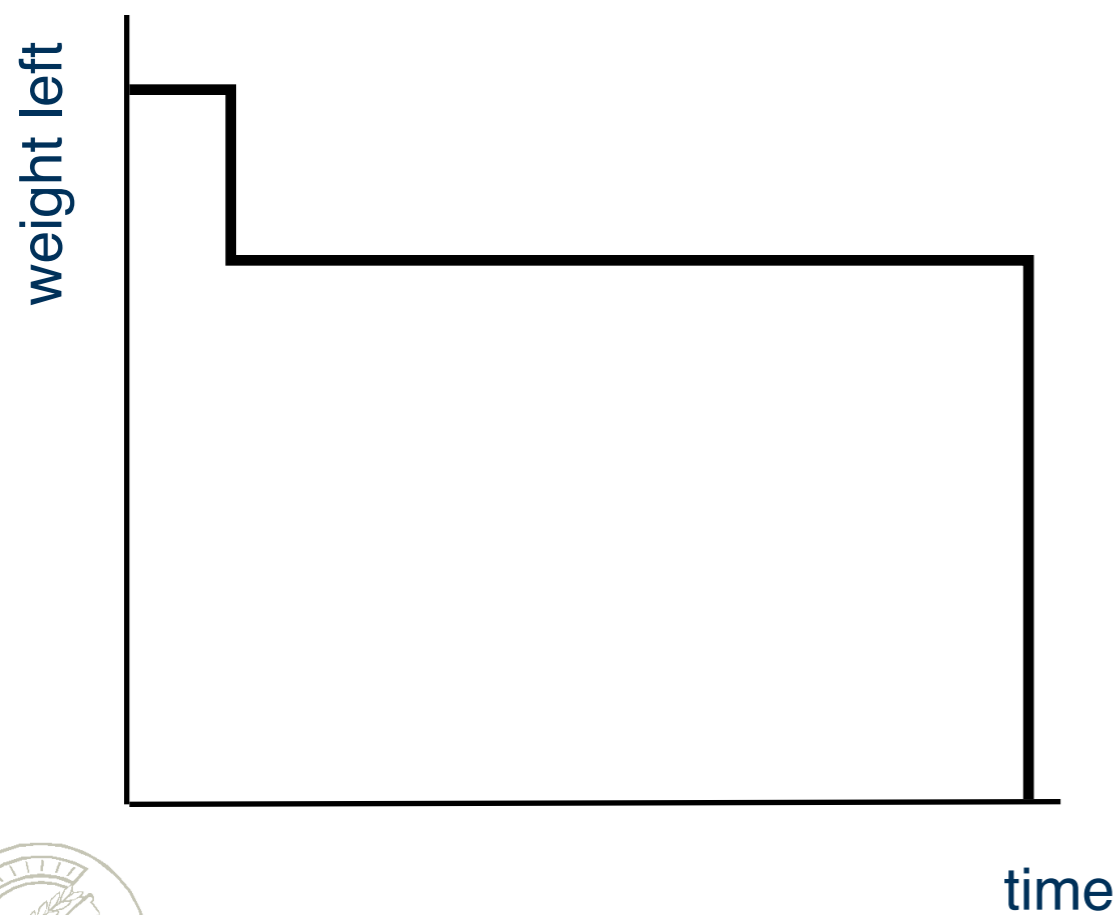
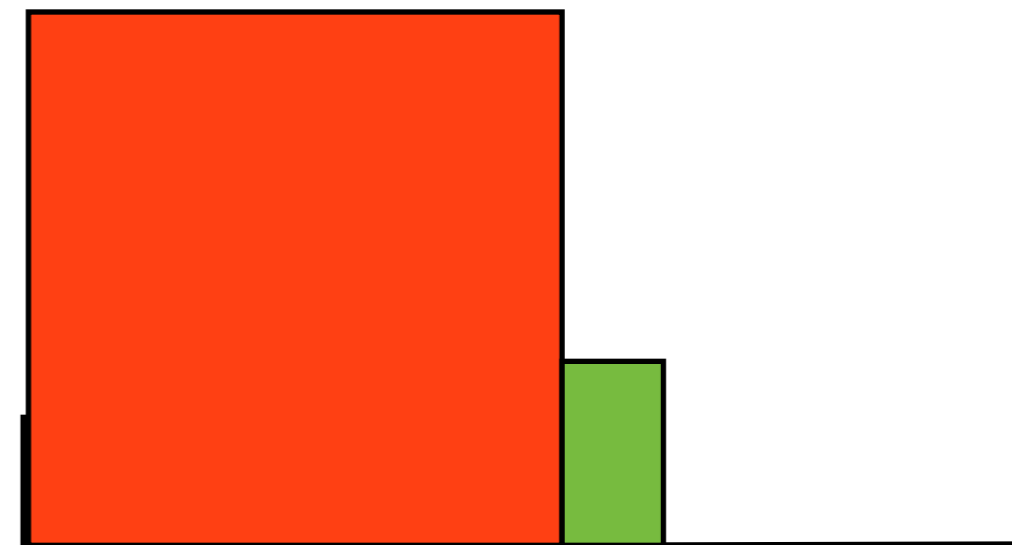
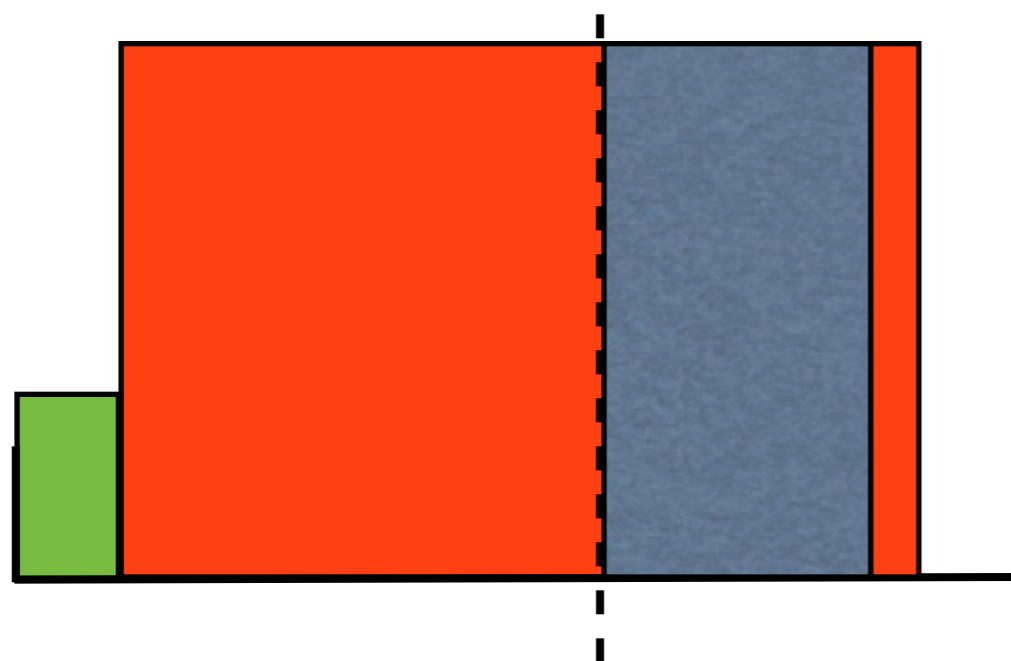
Smith's rule is not competitive



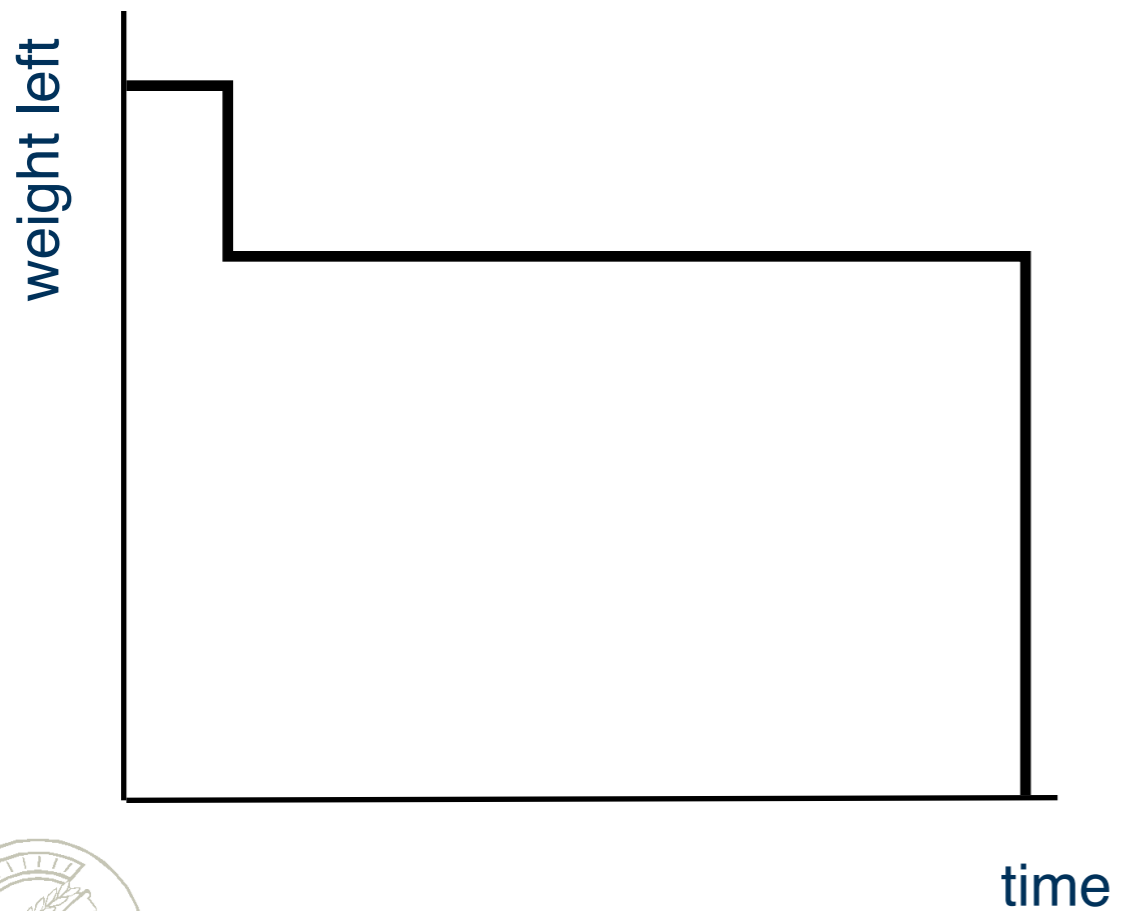
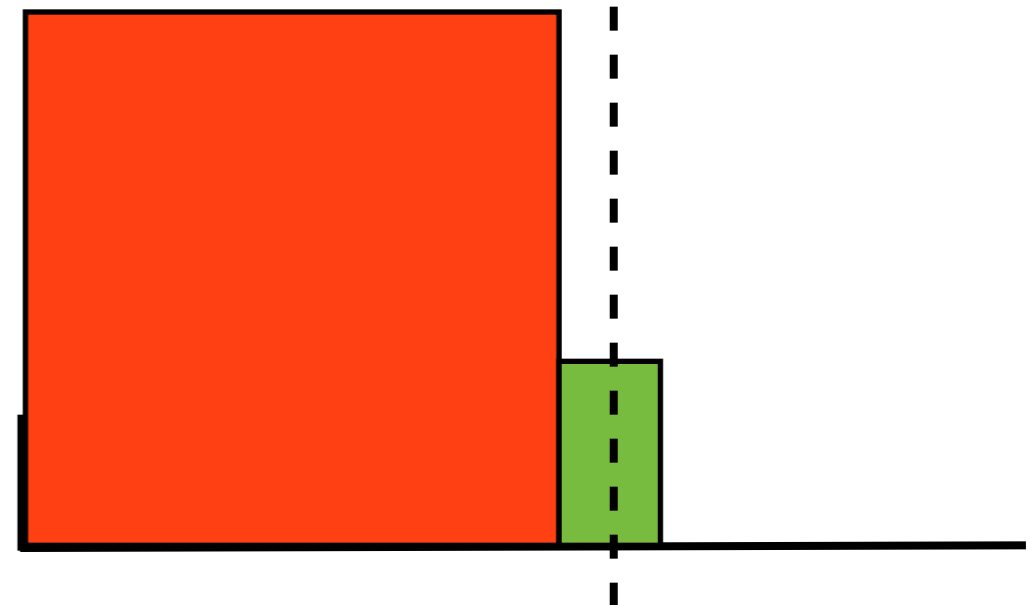
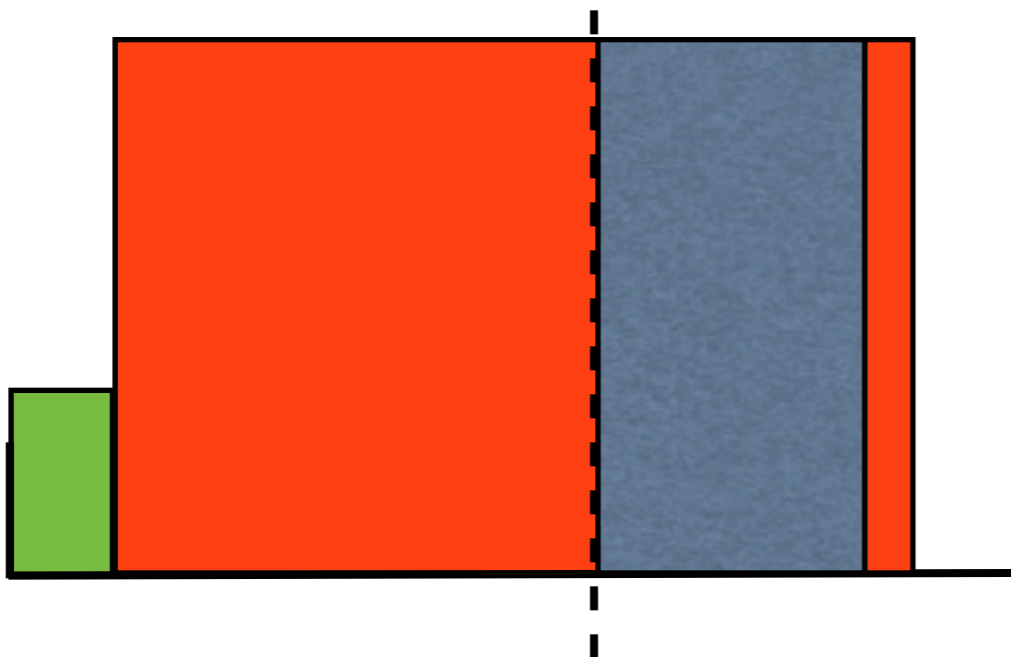
Smith's rule is not competitive



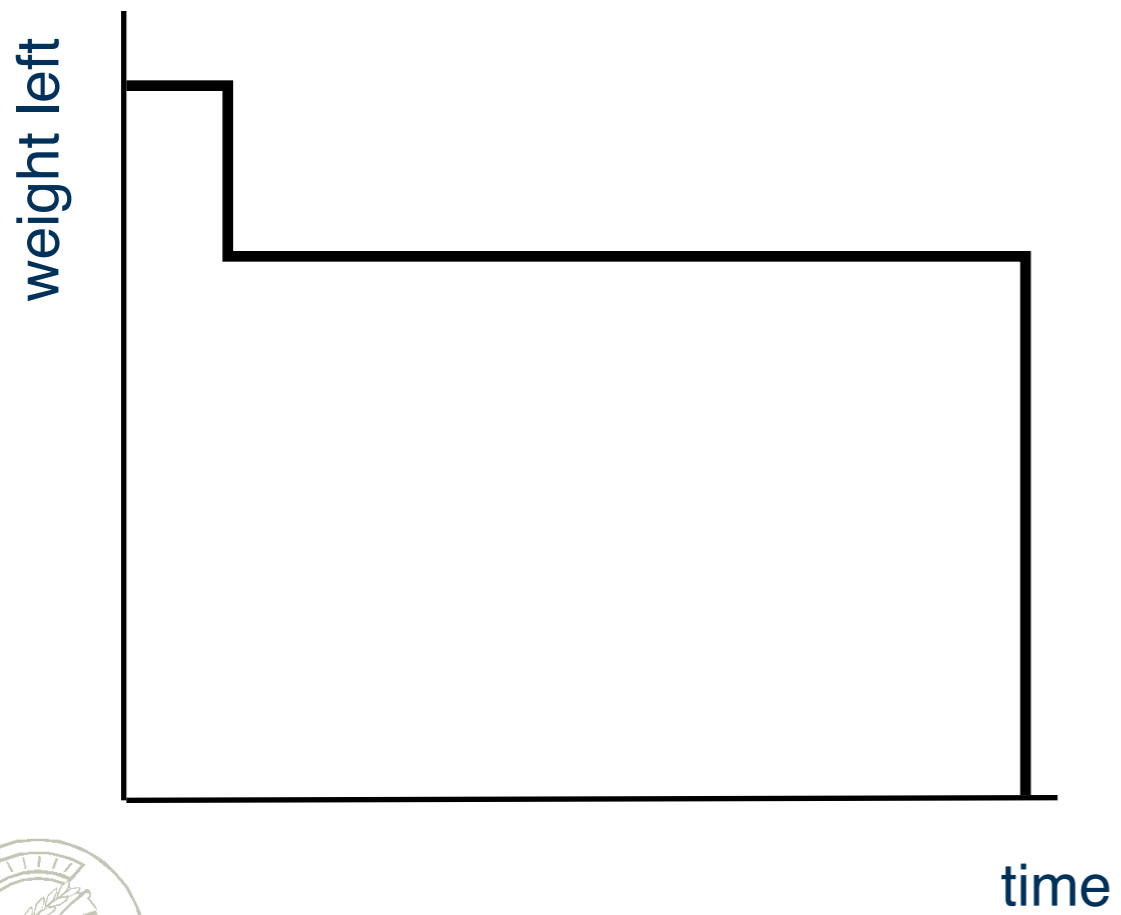
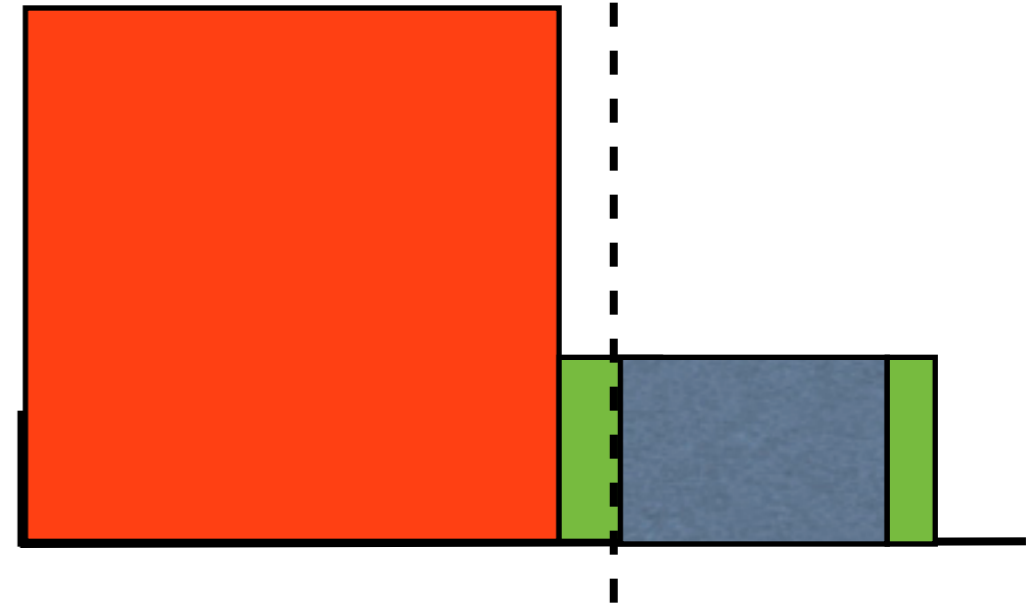
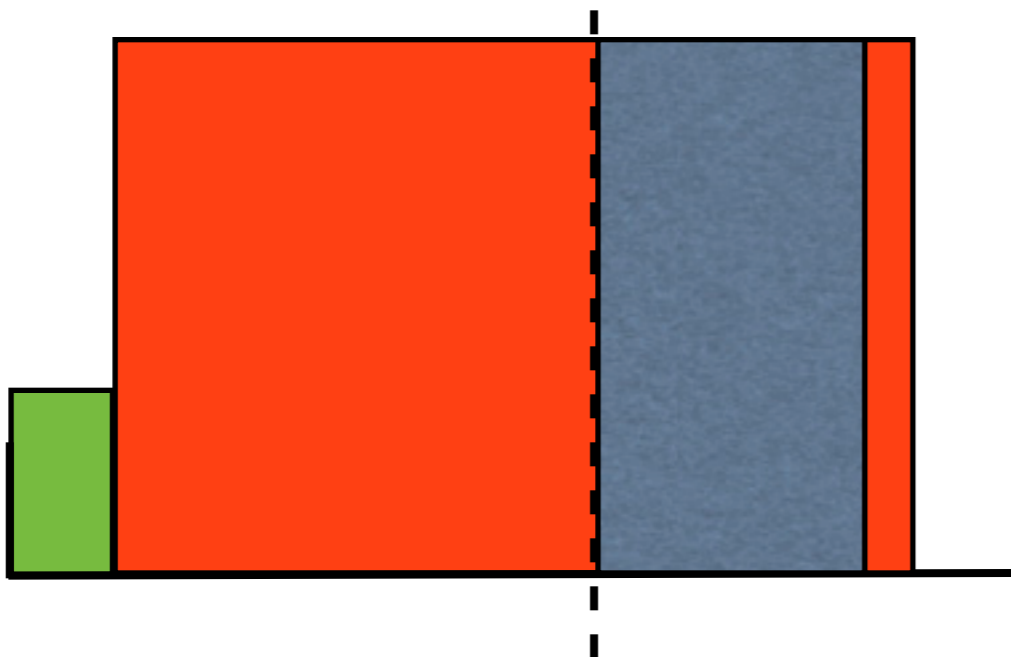
Smith's rule is not competitive



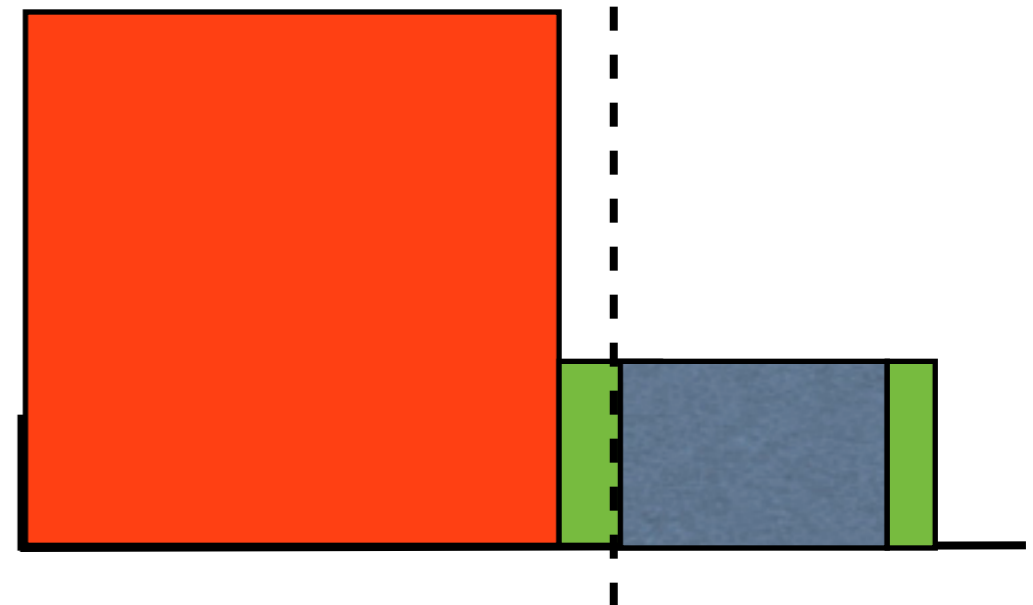
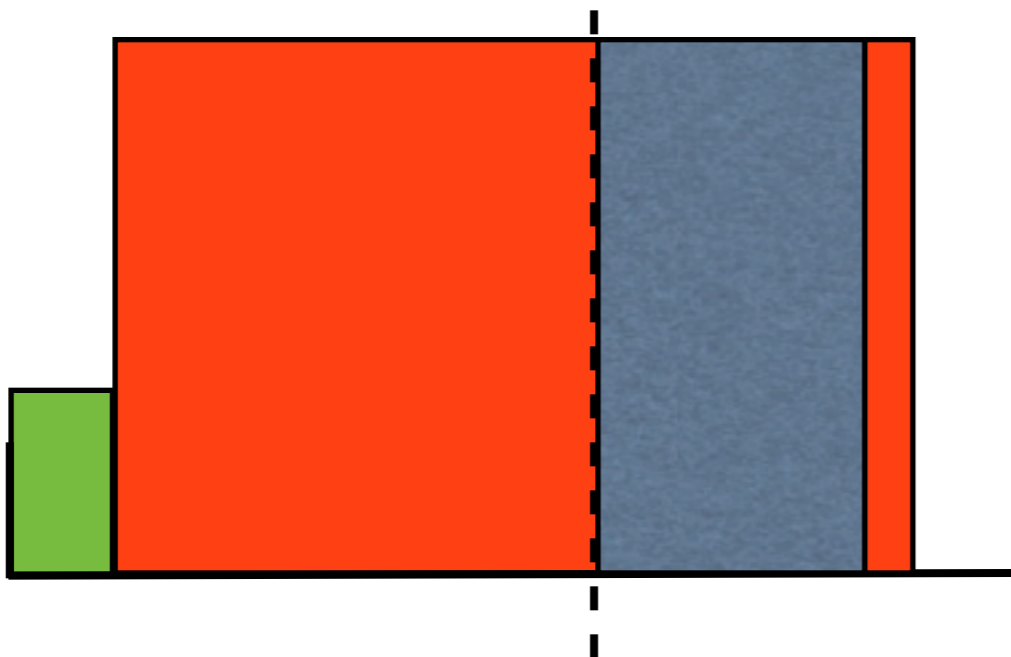
Smith's rule is not competitive



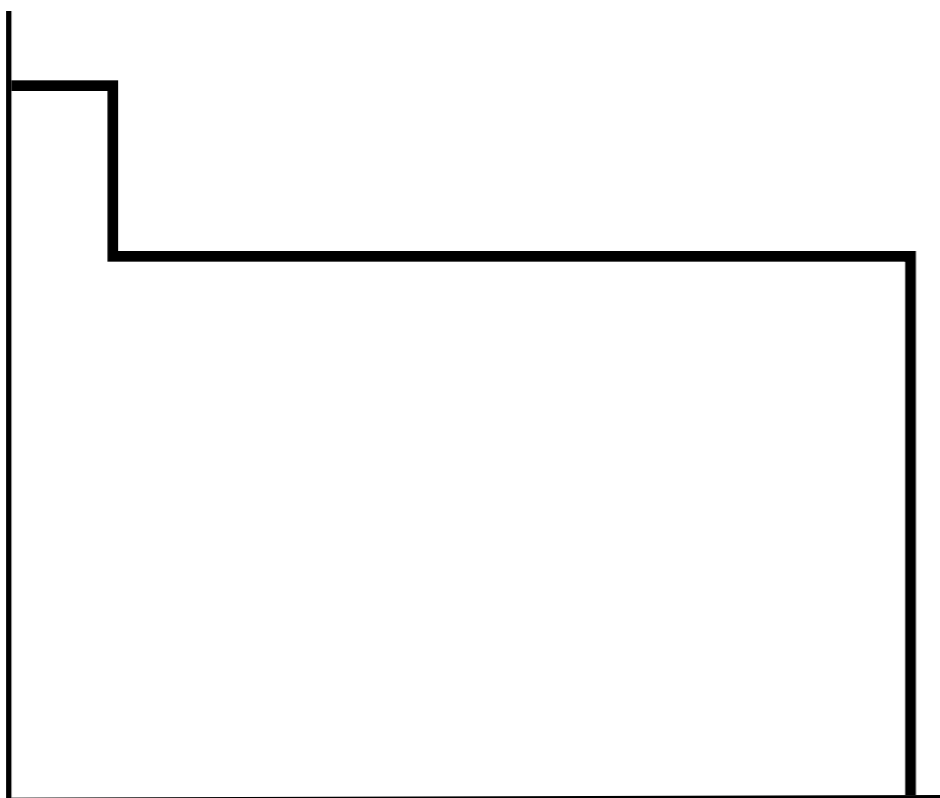
Smith's rule is not competitive



Smith's rule is not competitive

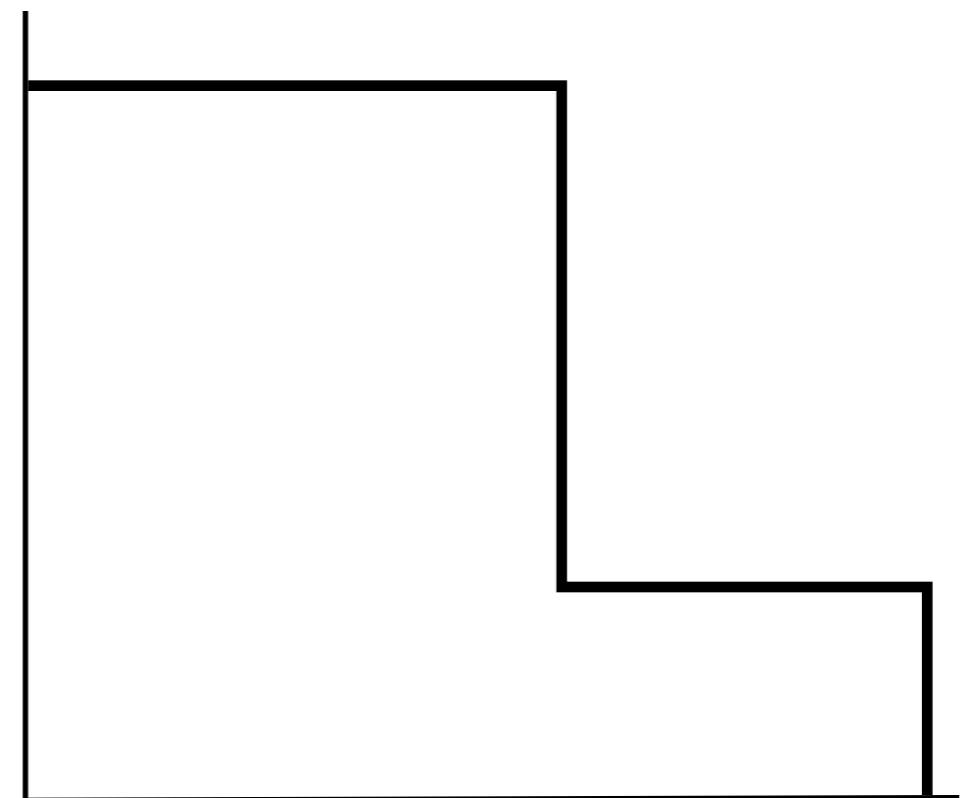


weight left



time

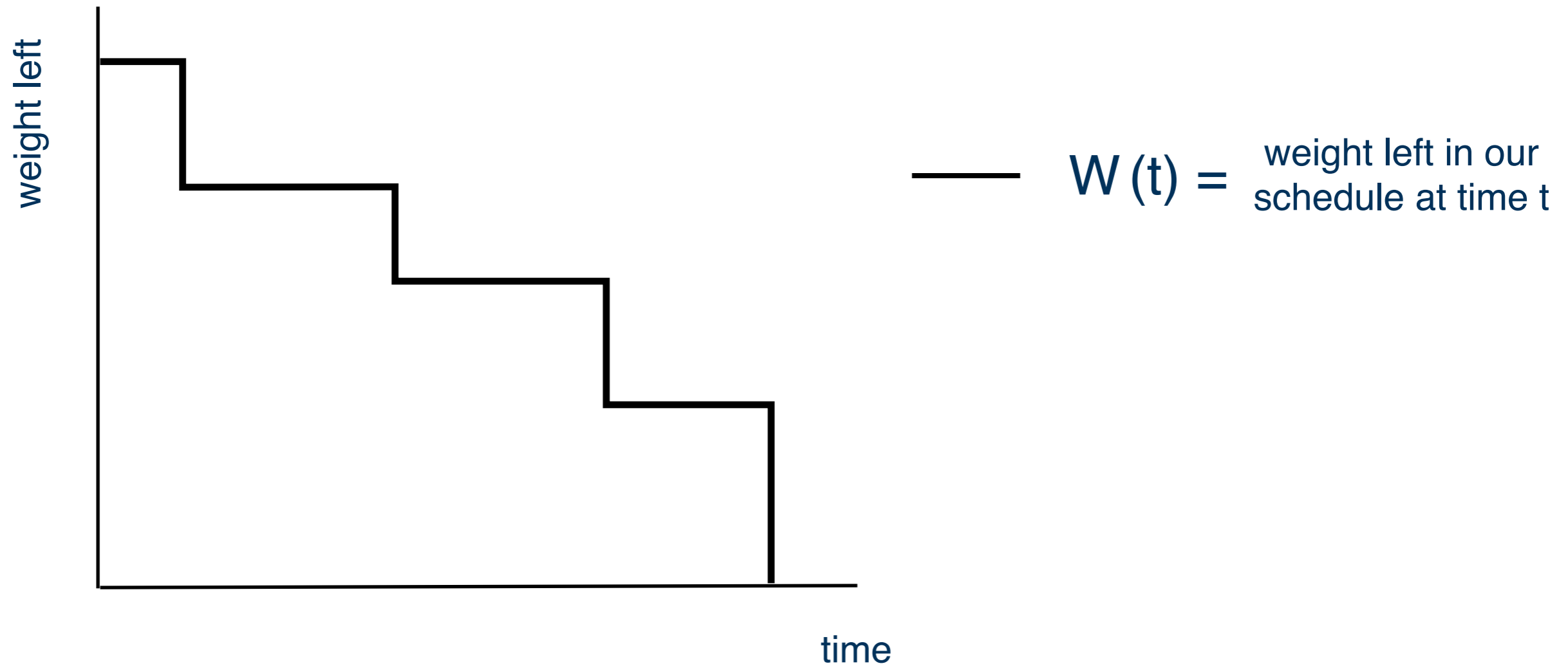
weight left



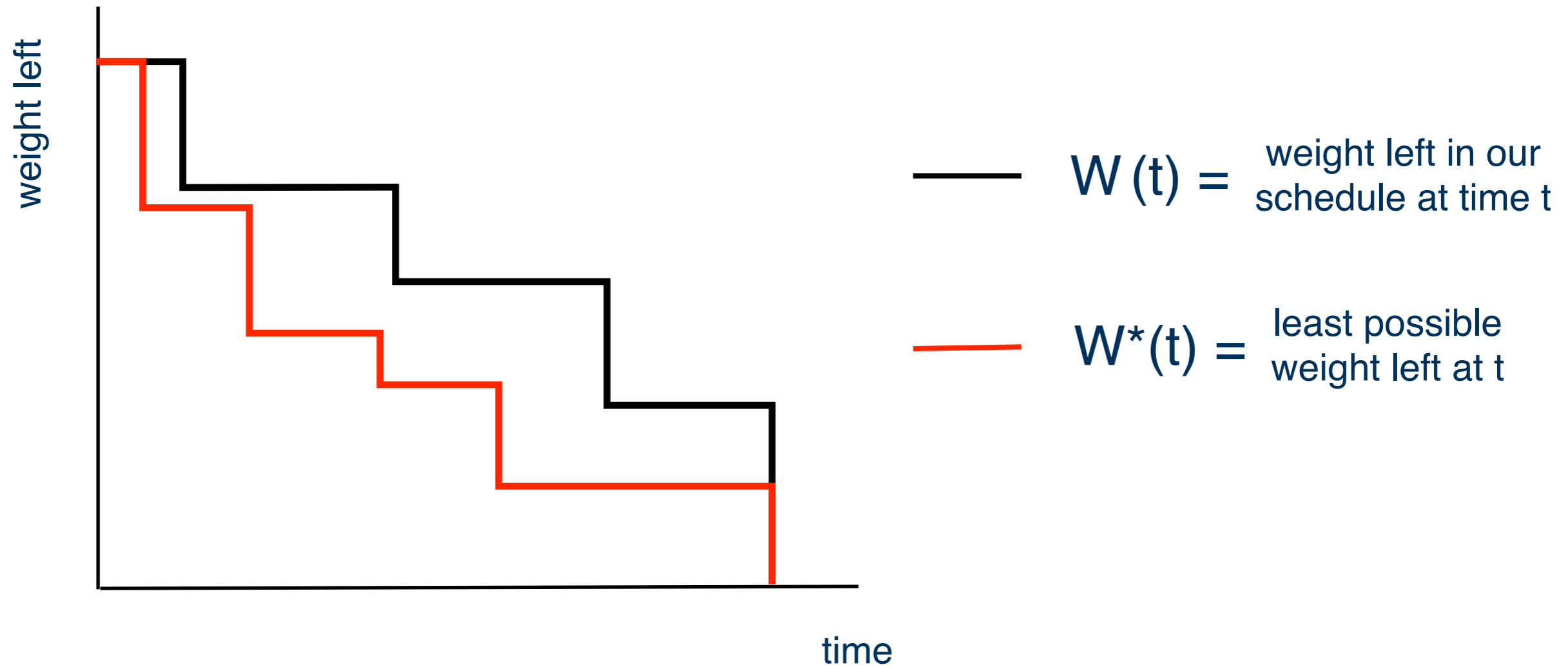
time



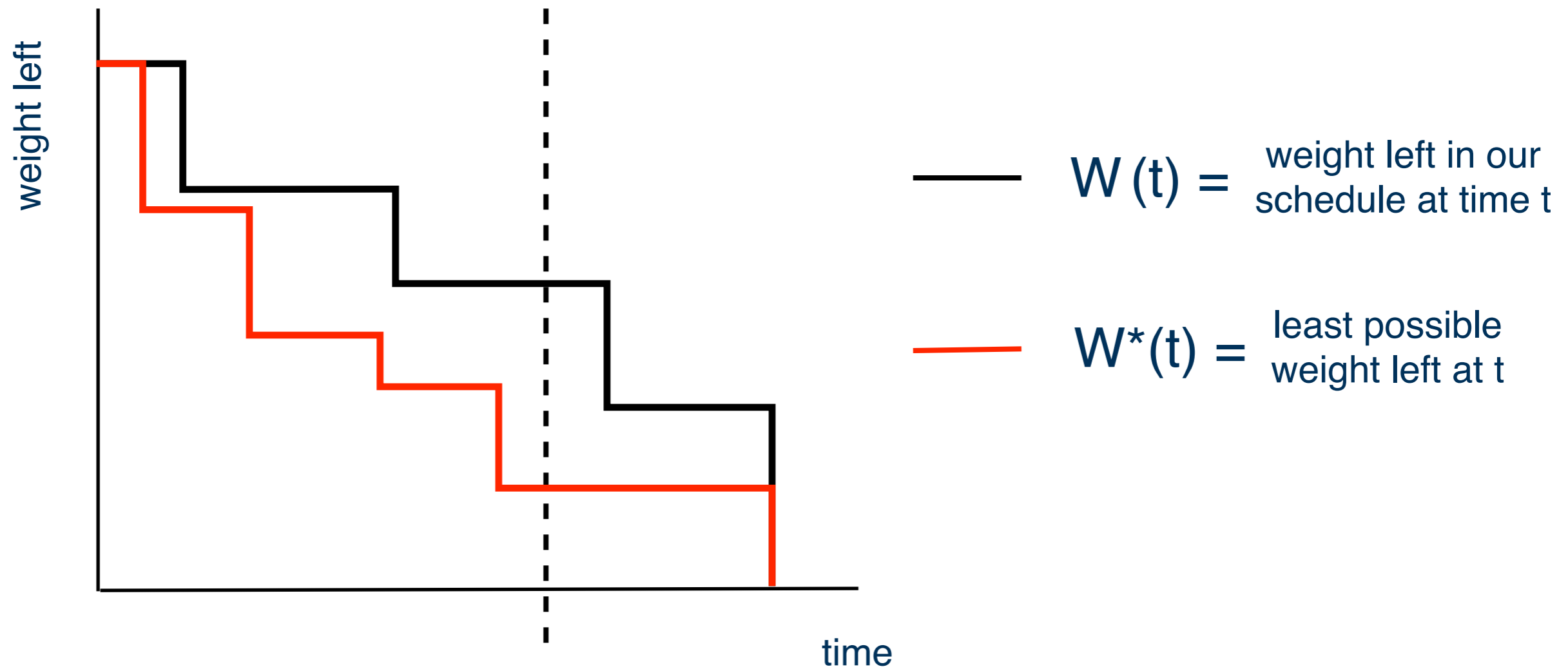
What can go wrong?



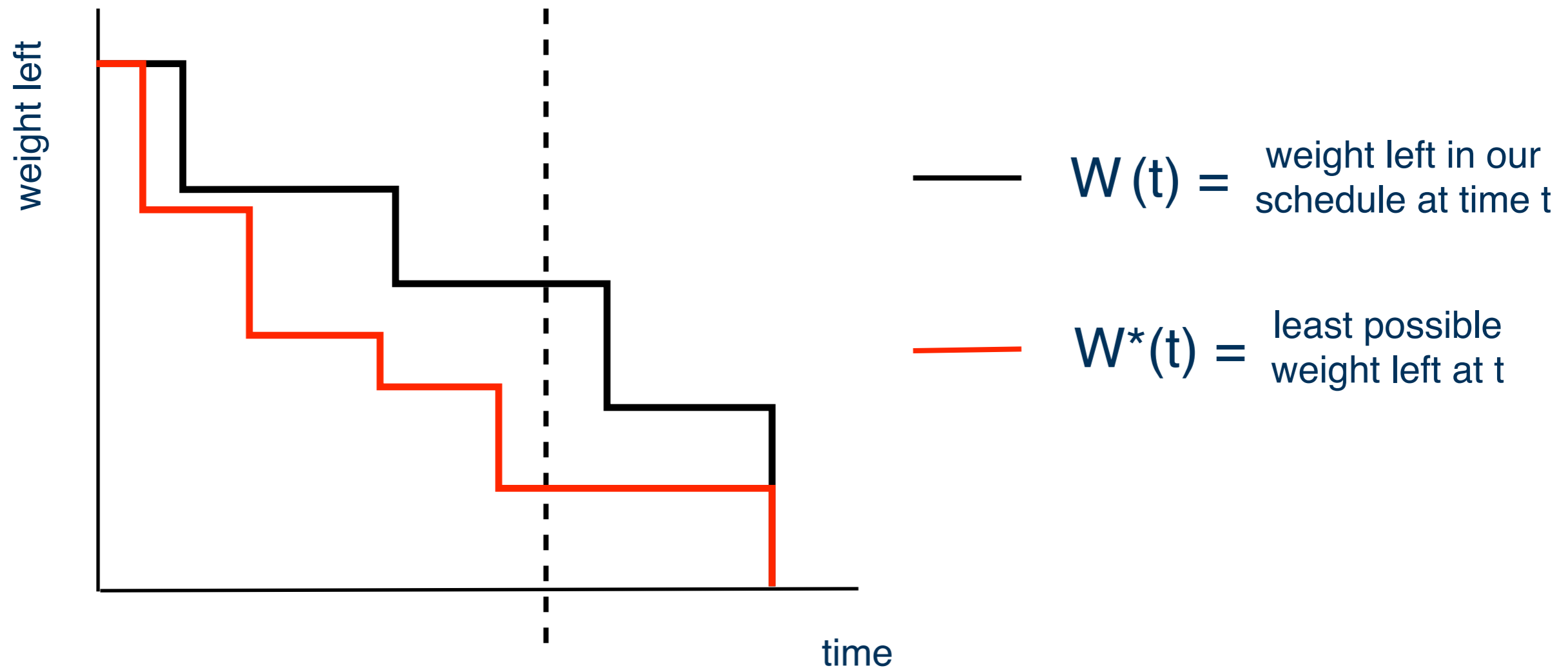
What can go wrong?



What can go wrong?



What can go wrong?



Lem:

The competitive ratio equals $\max W(t) / W^*(t)$ over all t for all t



Algorithm for universal scheduling



π = empty list

for $i = 0$ to k

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

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

return π



Algorithm for universal scheduling

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

Lem:

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  $\pi$ 
    
```

Lem:

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

Thm:

The universal schedules produced are at most 4 competitive



Bidding game



Bidding game

- Want to buy an object with unknown value $1 \leq x \leq T$



Bidding game

- Want to buy an object with unknown value $1 \leq x \leq T$
- Specify a bidding strategy $b_1 < b_2, \dots < b_k = T$



Bidding game

- Want to buy an object with unknown value $1 \leq x \leq T$
- Specify a bidding strategy $b_1 < b_2, \dots < b_k = T$
- Pay $b_1 + b_2 + \dots + b_i$ where $b_{i-1} < x \leq b_i$



Bidding game

- Want to buy an object with unknown value $1 \leq x \leq T$
- Specify a bidding strategy $b_1 < b_2, \dots < b_k = T$
- Pay $b_1 + b_2 + \dots + b_i$ where $b_{i-1} < x \leq b_i$
- We are α -competitive if $b_1 + b_2 + \dots + b_i \leq \alpha x$ for all x



Bidding game

- Want to buy an object with unknown value $1 \leq x \leq T$
- Specify a bidding strategy $b_1 < b_2, \dots < b_k = T$
- Pay $b_1 + b_2 + \dots + b_i$ where $b_{i-1} < x \leq b_i$
- We are α -competitive if $b_1 + b_2 + \dots + b_i \leq \alpha x$ for all x

Folk:

“Doubling” is 4-competitive
and this best possible



Lowerbound



Lowerbound

- Create a scheduling instance with **T** jobs



Lowerbound

- Create a scheduling instance with T jobs
- Given a specific schedule π we can read off a bidding strategy b_1, b_2, \dots, b_k



Lowerbound

- Create a scheduling instance with T jobs
- Given a specific schedule π we can read off a bidding strategy b_1, b_2, \dots, b_k

Lem:

If a schedule is α -competitive then bidding strategy is α -competitive



Lowerbound

- Create a scheduling instance with T jobs
- Given a specific schedule π we can read off a bidding strategy b_1, b_2, \dots, b_k

Lem:

If a schedule is α -competitive then bidding strategy is α -competitive

Thm:

Some time universal schedules must be at least 4 competitive



Extensions



Extensions

- Randomized schedules



Extensions

- Randomized schedules
- Precedence constraints



Extensions

- Randomized schedules
- Precedence constraints
- Release dates



Extensions

- Randomized schedules
- Precedence constraints
- Release dates
- Offline version



Open problems



Open problems

- Parallel machines



Open problems

- Parallel machines
- Stochastic information about breakdown pattern



Open problems

- Parallel machines
- Stochastic information about breakdown pattern
- Better performance through adaptive algorithms



Open problems

- Parallel machines
- Stochastic information about breakdown pattern
- Better performance through adaptive algorithms
- Optimal universal algorithms for a given instance

