Stackelberg Scheduling Strategies

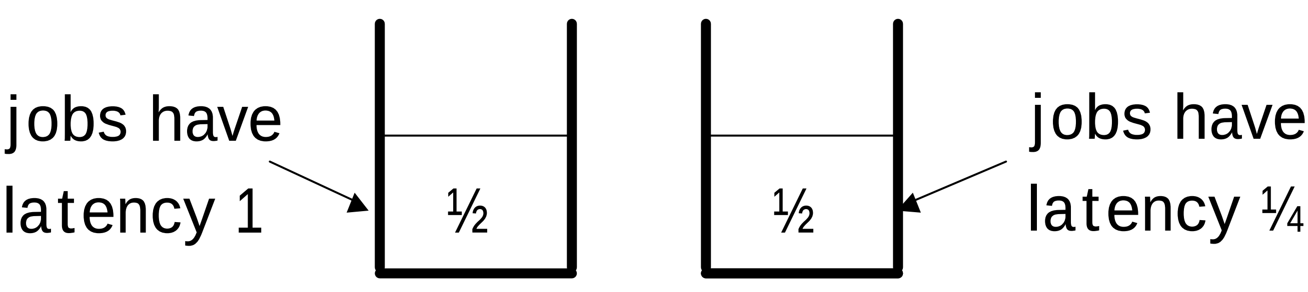
Tim Roughgarden
Cornell University

Machines and Latencies

The Model:

- m machines $1,2,\ldots,m$
- A rate r of jobs
  - jobs are small (model in a cts way)
- For each machine i, a load-dependent latency function $l_i(\bullet)$
  - assume continuous, nondecreasing

Example: $(r=1)$



$l_1(x)=1$ $\quad$ $l_2(x)=x^2$

Equilibria

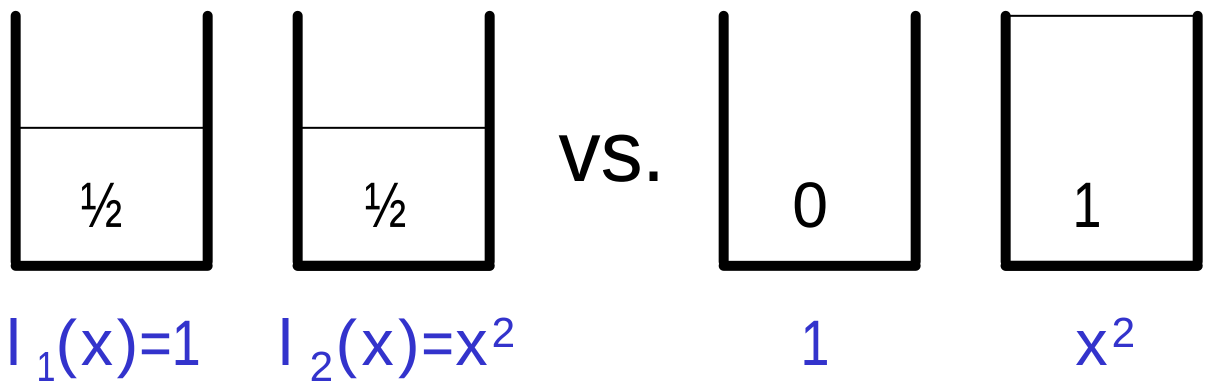
Job Assignments:

- x_i = amount of jobs assigned to machine i
- vector x ⟺ assignment of all jobs

Equilibria:

- which job assignments are "stable"?
  - jobs controlled by selfish, noncooperative agents
  - no job wants to switch machines (no job should be envious)

More on Equilibrium Assignments



Def: an assignment is at Nash equilibrium (is a Nash assignment) if:

- all used machines have equal latency
- unused machines have greater latency

Fact: always have existence, uniqueness

How Good is an Assignment?

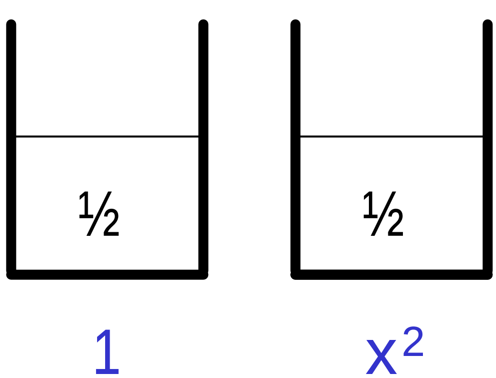
The Cost of an Assignment:

- C(x) = cost or total latency experienced by assignment x:

$$S_i\, x_i \bullet l_i(x_i)$$

Þ our notion of system performance

Example:



cost =

½•1 +½•¼ = ?

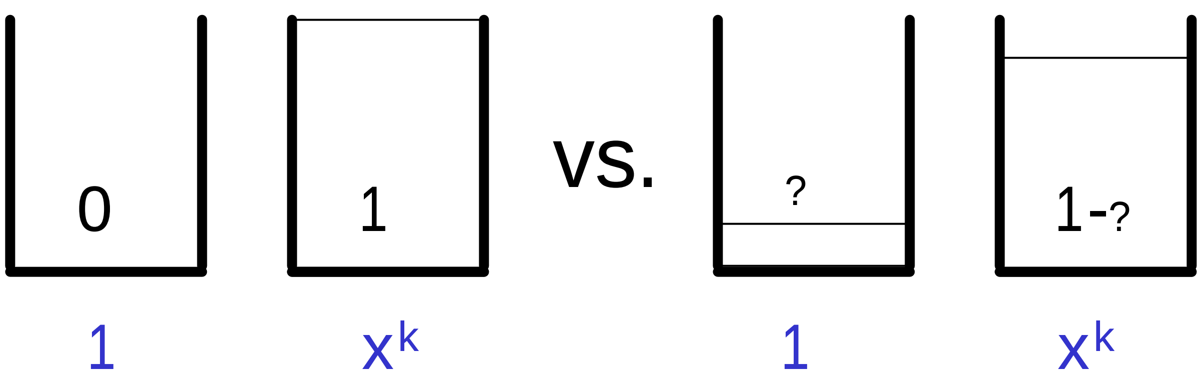
How Good are Nash Assignments?

Goal: prove that Nash assignments are near-optimal

- want a laissez faire approach to regulating users

Problem: false in general!

Example: (r=1, k large)



Nash costs 1 OPT costs ≈ 0

Near-Optimal Nash Assignments

Solution #1: relax our notion of "near-optimal"

- Theorem [RT00]: cost of Nash assignment = cost of OPT at rate $2r$

Solution #2: restrict to linear latency functions ($l_i(x)=a_i x+b_i$)

- Theorem [RT00]: cost of Nash assignment = 4/3 times cost of OPT (at same rate r)

Taming Selfishness through a Manager

Solution #3: change the model!

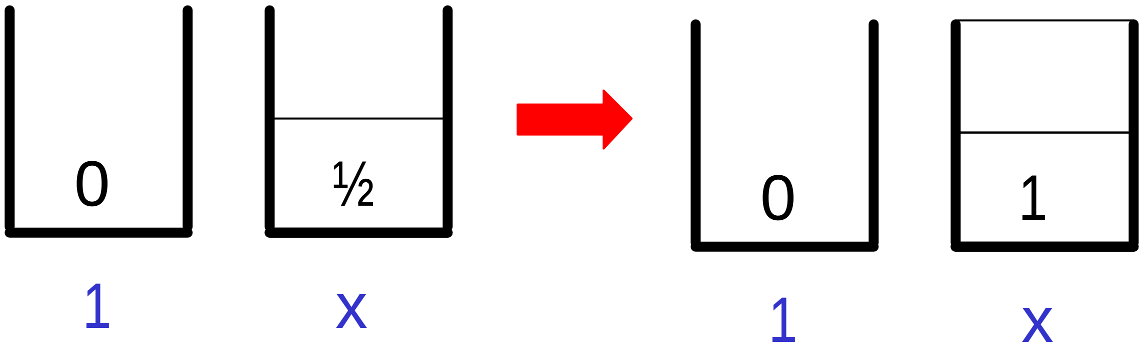
- not all jobs need be controlled by selfish users
  - "centrally controlled" vs. "selfishly controlled" jobs
  - behavior of selfish users depends on assignment of managed jobs

Goal:

- assign centrally controlled jobs to induce "good" selfish behavior
  - see also [Korlis, Lazar, Orda 97]

Examples

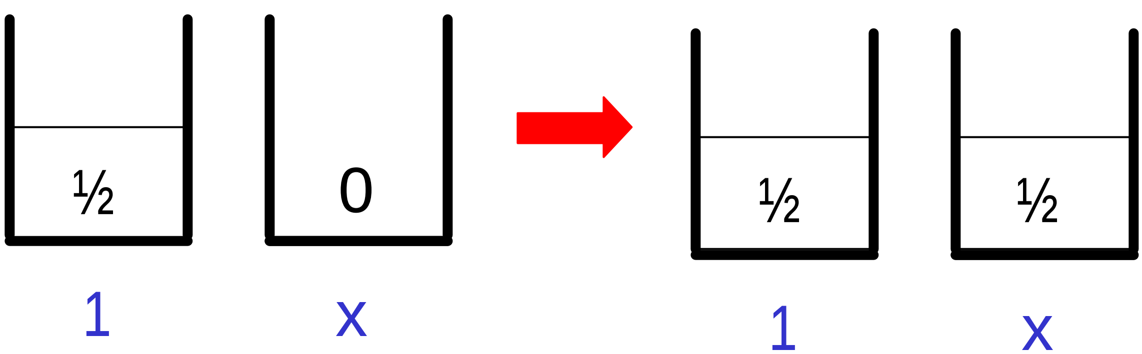
Strategy #1:



Strategy

Induced Equilibrium
(same as Nash eq)

Strategy #2:

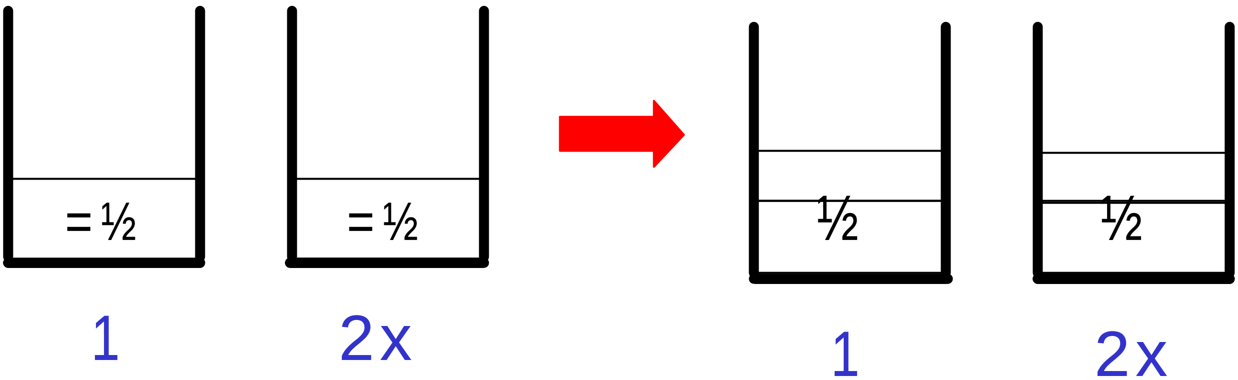


Strategy

Induced Equilibrium
(same as optimum!)

Examples

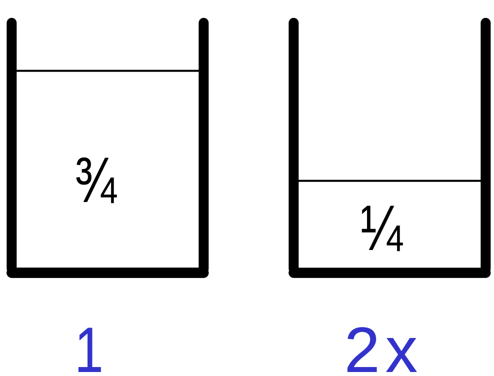
Any Strategy:



Strategy

Induced Equilibrium
(same as Nash eq)

Optimal Assignment (cost = ?):



Stackelberg Strategies

- Stackelberg strategy = assignment of centrally controlled jobs
  - Þ yields an induced equilibrium

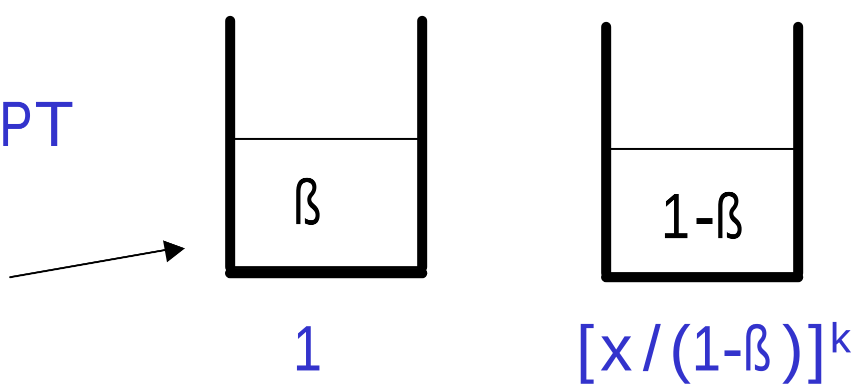
- Basic Questions:
  - what's the best strategy?
    - can we compute/characterize it?
  - how inefficient is the best induced equilibrium?
    - are we provably near-optimal?

Our Results - General Latency Functions

Theorem 1: Can efficiently compute a strategy inducing an equilibrium with cost $= (1/ß) \times$ cost of optimal assignment.

ß = fraction of centrally assigned jobs

Note:
$(1/ß) \times OPT$ is best possible



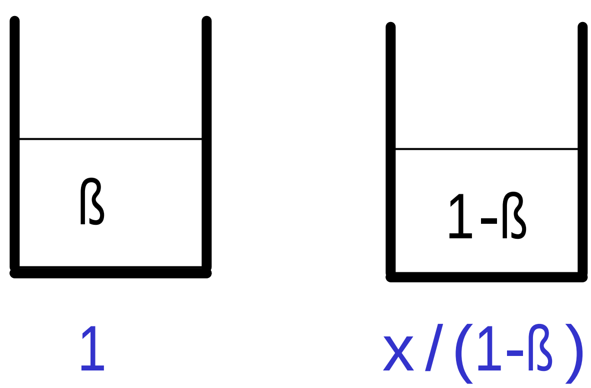
(Induced eq = Nash eq for any strategy)

Our Results - Linear Latency Functions

Theorem 2: Can efficiently compute a strategy inducing an equilibrium with cost = [4/(3+ß)] × cost of optimal assignment.

ß = fraction of centrally assigned jobs

Note:
[4/(3+ß)] × OPT is best possible

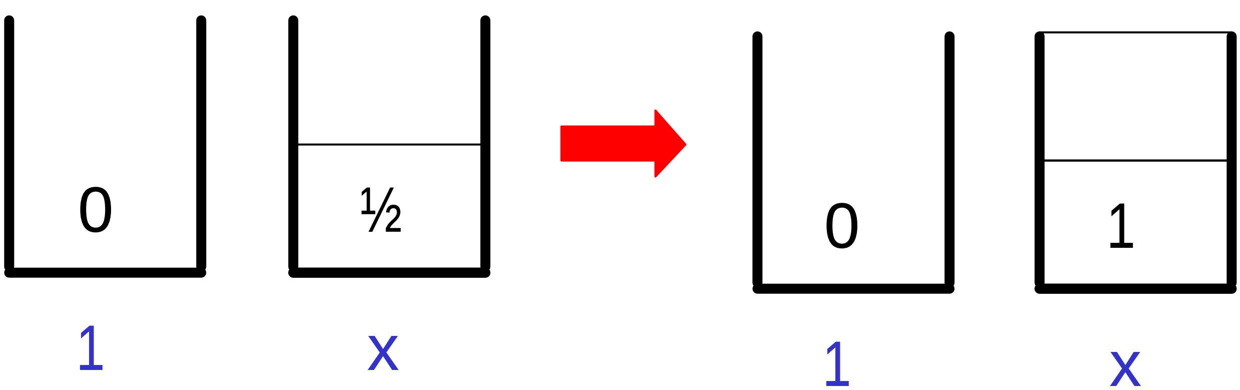


(Induced eq = Nash eq for any strategy)

What makes a strategy (in)effective?

The Aloof Strategy

- ignore selfish users, assign centrally controlled jobs in cheapest possible way



Strategy

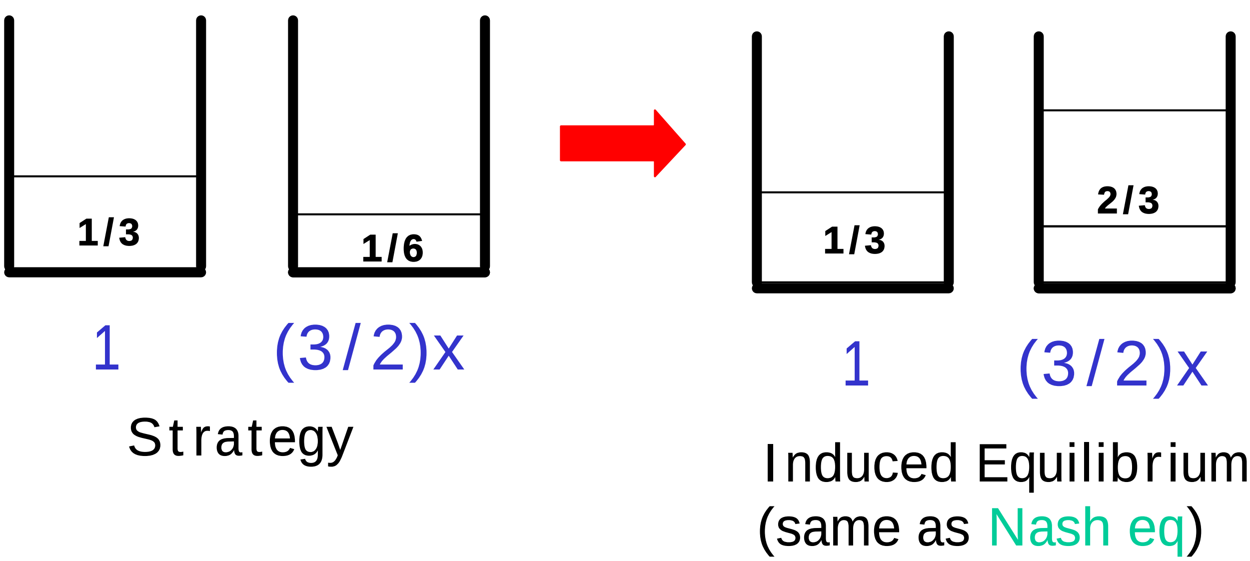
Induced Equilibrium
(same as Nash eq)

- Þ induced eq costs 1, OPT = ¾

What makes a strategy (in)effective? (con’d)

The Scale Strategy

- compute optimal assignment x of all jobs, assign centrally controlled jobs via ß • x



- Þ induced eq costs 1, OPT = 5/6

What have we learned?

Morals:

- avoid machines that selfish users will (over)use anyways
- assign centrally controlled jobs to machines least attractive to the selfish users

  (⇔ edges with large latency)
- don't oversaturate any machines

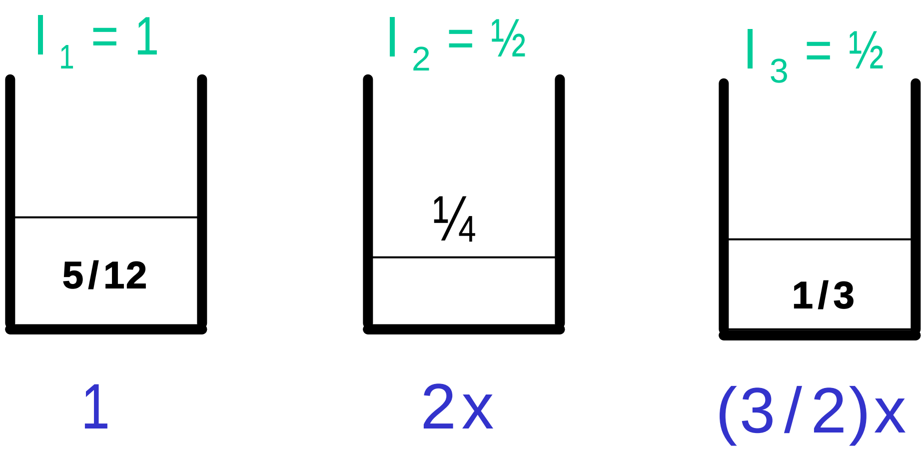
The LLF Strategy

Largest Latency First (LLF):

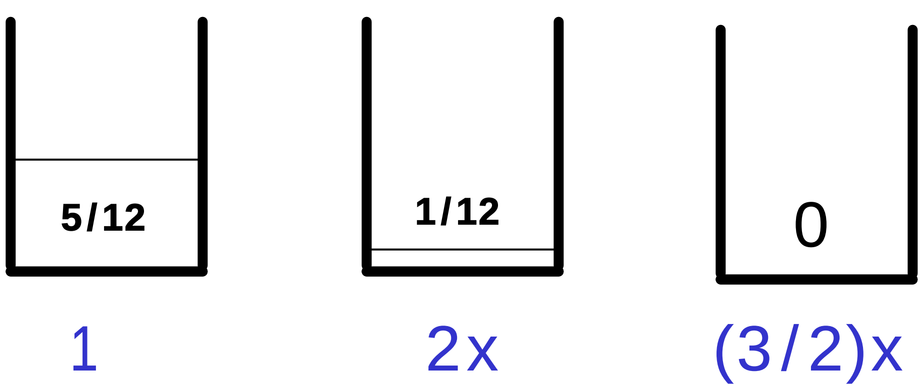
- compute opt assignment x of all jobs
- for each machine i, in order of decreasing $l_i(x_i)$ values:
  - if $\geq x_i$ centrally controlled jobs remain, assign x_i jobs to i (saturate machine i)
  - else assign remaining jobs (if any) to i

Example

Optimal Assignment: (r=1)



The LLF Strategy: (ß=½)



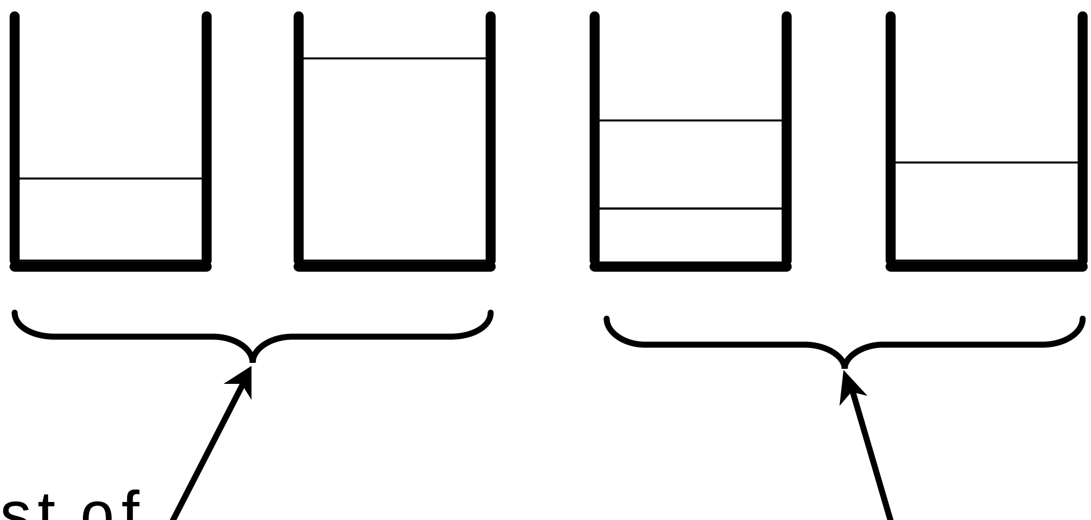
LLF with General Latency Functions

Theorem 1: The LLF strategy induces an equilibrium with cost $= (1/ß) \times$ cost of optimal assignment.

Proof idea: Use iterative structure of LLF to proceed by induction on # of machines.

Proof Idea

Induced Equilibrium (w.r.t. LLF):

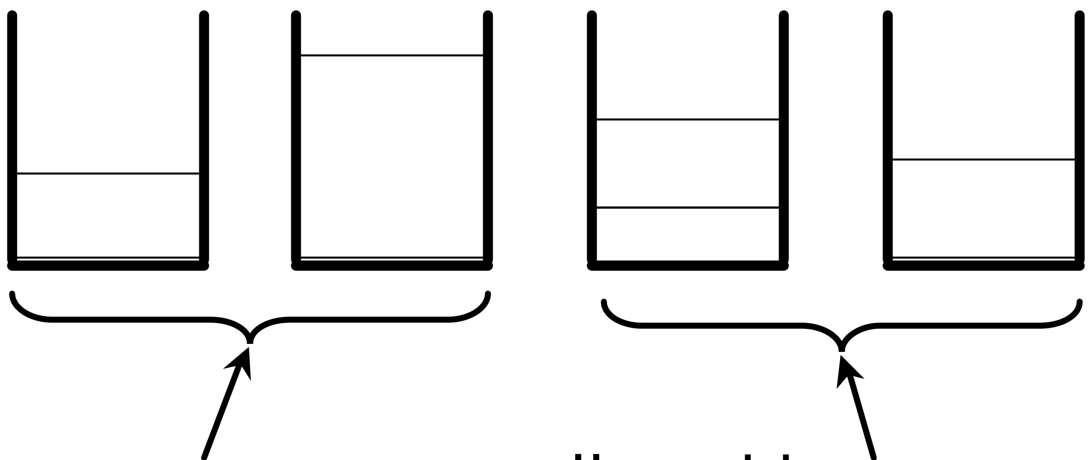


cost of induced eq and OPT are equal here

will apply IH on these machines

$64K Question: Is our inductive guarantee strong enough?

Proof of Theorem 1



machines have latency ≥ L

- μ = # of jobs
- OPT pays

  ≥ we pay

  ≥ μL

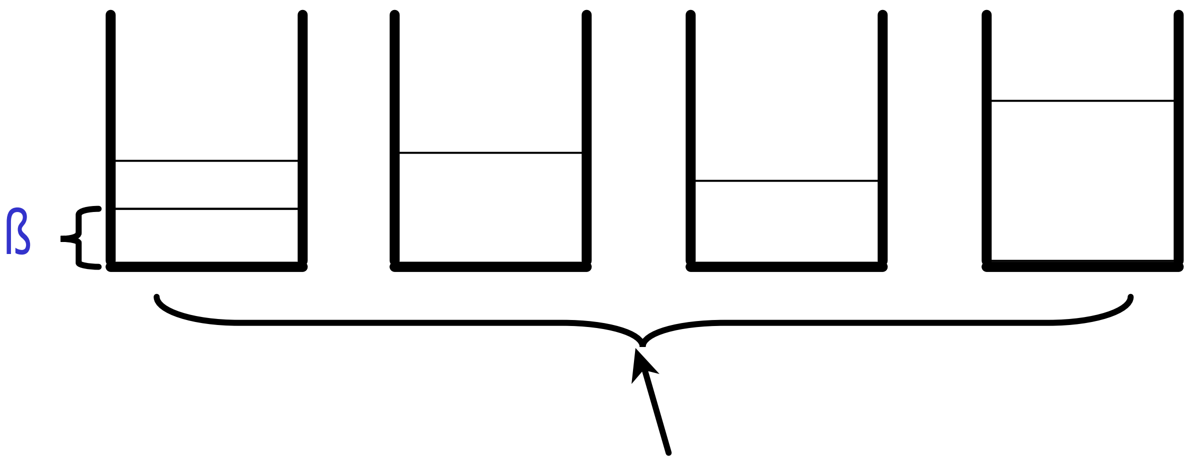
all machines same latency = L

- we pay (1-μ)L
- ß' = $\frac{(ß-μ)}{(1-μ)}$
- IH Þ OPT pays

  ≥ ß' × (1-μ)L

  = (ß-μ)L

Bug Fix



all machines same latency = L

Problem: Can’t recurse!

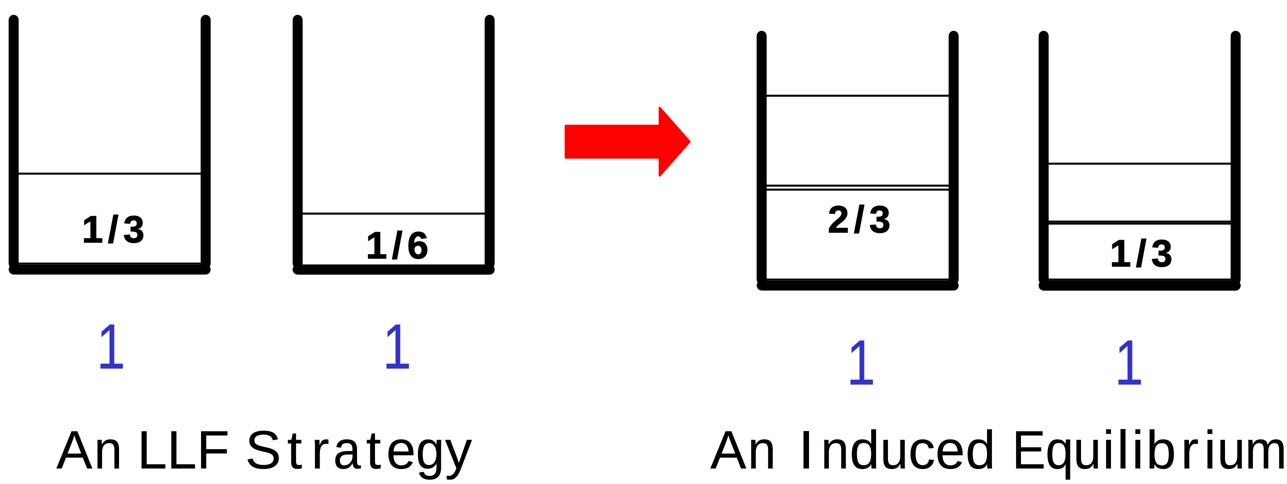
Fix: LLF failed to saturate machine 1

- Machine 1 is high-latency (≥ L)
- our solution costs L (assume r=1)
- OPT puts ≥ ß jobs on machine 1

Þ OPT pays ≥ ßL

Bug Fix (con'd)

Problem: have we really covered all the cases?



Lemma: Only a problem with locally constant latency functions.

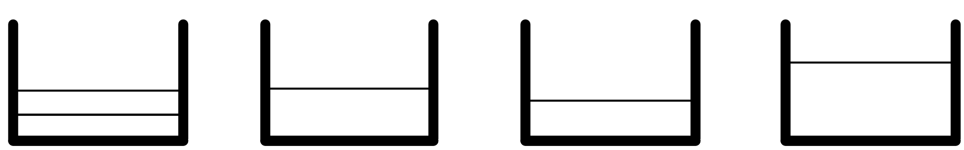
- can reassign selfish users, get a "nicer" induced equilibrium

LLF with Linear Latency Functions

Theorem 2: The LLF strategy induces an equilibrium with cost $= [4/(3+\beta)] \times$ cost of optimal assignment.

Main difficulty:

- previous argument too weak for:



- need detailed study of Nash, OPT when all latencies are linear

Computing Optimal Strategies

Question: is the LLF strategy optimal on all instances?

Bad news: no, in fact:

Theorem 3: Computing the optimal strategy is NP-hard (even for linear latency fns).

- Compare to: optimal, Nash assignments

Sketch of Reduction

Idea: view the best SS as solving the optimization problem P:

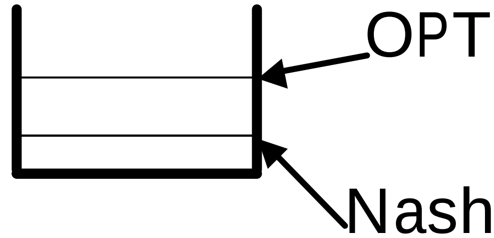
$$\max \quad S_i \max(0, x_i\text{-}Nash_i)$$

$$\text{s.t.:} \quad S_i\, x_i = ß$$

i.e., don't oversaturate

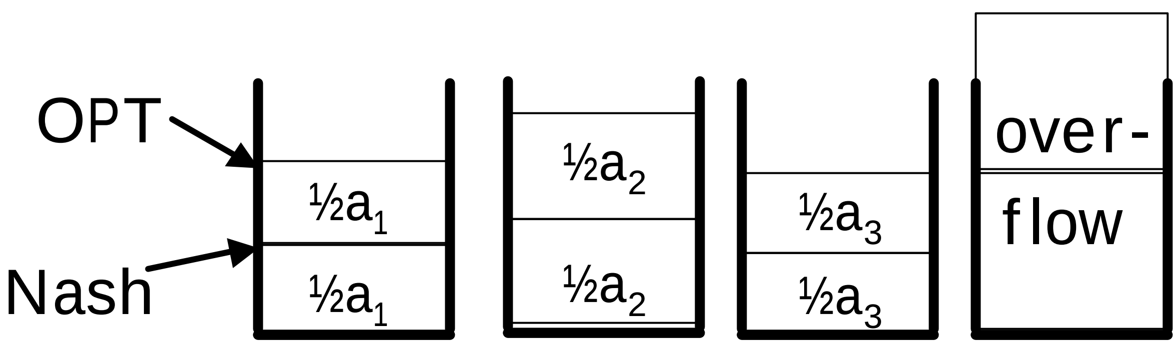
$$0 = x_i = OPT_i \qquad \text{all } i$$

– Intuition: Strategy should look more like OPT than Nash



Reduction (con'd)

Partition: given n numbers $a_1,...,a_n$ summing to A, is there a subset with sum $A/2$?



- amount of controlled jobs = $A/2$
- saturate machine i ⇔ pick a_i
- "yes" Instance Þ P has value $A/4$
- "no" Instance Þ P has value $< A/4$

Open Questions

Approximating the optimal strategy:

- LLF is best possible using OPT as a lower bound
- better guarantees for LLF via a better lower bound?
- more sophisticated algorithms?
  - existence of a (F)PTAS?

Also:

- min ß sufficient to recover OPT
  - see also [KLO 97]

General Graphs

Open:

- for general latency fns, fixed ß: a strategy inducing an equilibrium w/cost = f(ß) × opt
  - 1/ß not achievable in general graphs (!)
  - maybe 2/ß? (or O(n))
- for linear latency fns, a strategy w/cost < 4/3 × opt
  - e.g., is 8/7 achievable for ß=½?