



max planck institut
informatik

Speed Scaling to Manage Flow Time and Energy

Lap-Kei Lee

Max Planck Institute for Informatics

Bristol Algorithms Days 2010 Feasibility Workshop

February 16

Outline

- Models and problems
 - Speed scaling
 - Minimize flow time plus energy
- Amortized local competitiveness
- Analysis techniques
 - Fractional & Integral flow time
 - From Unbounded speed to Bounded speed
 - Application of Young's Inequality




Outline

- Models and problems
 - Speed scaling
 - Minimize flow time plus energy
- Amortized local competitiveness
- Analysis techniques
 - Fractional & Integral flow time
 - From Unbounded speed to Bounded speed
 - Application of Young's Inequality



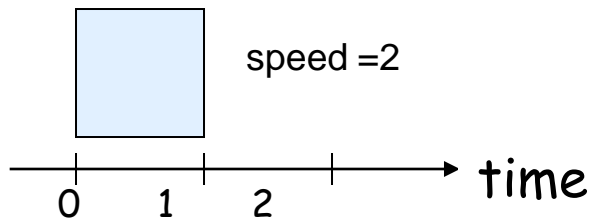
Energy efficiency

- For design of mobile devices, energy efficiency is a major concern. 
- How to save energy? **Dynamic speed (voltage) scaling**
 - Slow down processor whenever possible.

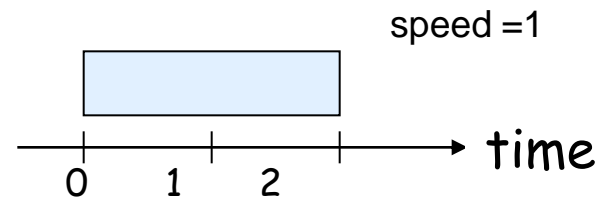
Model

- At any time, processor speed **s** is adjustable.
- power = **s^α** , where $\alpha > 1$ ($\alpha \sim 3$).

E.g., $\alpha = 3$ & run a job with **2 units of work**.



$$\text{energy: } 2^3 \times 1 = 8$$



$$\text{energy: } 1^3 \times 2 = 2$$

Online job scheduling

- Schedule a set of jobs on a processor.
- Jobs arrive online, i.e., no future information.
- Preemption is allowed.
 - job execution can be stopped and resumed later.

Quality of Service: **Total flow time** of jobs

- **Flow time** of J = completion time $c(J)$ - release time $r(J)$
 - measure how long to wait before a job completes
- Jobs may have weights to indicate their importance.
 - QoS measure: total **weighted flow time**.



Minimize flow time plus energy

- We want
 - small total flow time **F** : run job **faster**
 - small energy consumption **E** : run job **slower**
- Combined objective **F+E** [Albers and Fujiwara; STACS06]
 - From economic viewpoint, users are willing to pay one unit of energy to reduce **ρ** units of flow time
 - Minimize **F + ρ E**
 - WLOG, we can assume **$\rho=1$**



Algorithm & Performance

Scheduling algorithm needs to decide at any time:

1. which job to run (**job selection**)
2. at what speed the job is run (**speed scaling**)

Performance of online algorithm **ALG**:

Competitive analysis

- Compare cost of **ALG** with **optimal offline** algorithm **OPT**.
 - offline: **OPT** has complete information in advance.
- **ALG** is **c**-competitive if for **any** job sequence I ,

$$\mathbf{ALG}(I) \leq \mathbf{c} \mathbf{OPT}(I)$$



Clairvoyant & Non-clairvoyant settings

Clairvoyant setting:

- When a job arrives, the **job size is known**.
 - E.g., web-server serving static documents

Non-clairvoyant setting:

- When a job arrives, **job size is not known** until it completes.
 - E.g., operating systems
- Non-clairvoyant algorithm is applicable to the clairvoyant setting.



Clairvoyant algorithms

For minimizing $F+E$ for **unit-size jobs**,

Speed scaling: [Albers & Fujiwara; STACS 06]

speed $s(t)$ proportional to **$n(t)^{1/\alpha}$** , where **$n(t)$** is the number of unfinished jobs

- $8.3e \left(\frac{3+\sqrt{5}}{2}\right)^\alpha$ -competitive [Albers & Fujiwara; STACS 06]
 - Analyze using some global arguments.
 - The ratio is exponential to α .
- 4-competitive [Bansal, Pruhs, Stein; SODA 07]
 - Using “**Amortized local competitiveness**” for analysis [Bansal, Kimbrel, Pruhs; FOCS 04]



Outline

- Models and problems
 - Speed scaling
 - Minimize flow time plus energy
- **Amortized local competitiveness**
- Analysis techniques
 - Fractional & Integral flow time
 - From Unbounded speed to Bounded speed
 - Application of Young's Inequality



Local competitiveness



$n(t)$ = number of unfinished jobs at time t

Back to the classic setting where processor speed is fixed:

- Objective: minimizing total flow time
- Total flow time: $F = \int n(t) dt$
 - When speed $s(t) = n(t)^{1/\alpha}$, $E = \int s(t)^\alpha dt = \int n(t) dt = F$
- Total flow time incurred up to time t : $F(t) = \int_0^t n(x) dx$

Local competitiveness (cont')

To show that an algorithm **ALG** is **c**-competitive,

- $F_a(t)$: total flow time incurred up to **t** in **ALG**
- $F_o(t)$: total flow time incurred up to **t** in **OPT**
- $n_a(t)$: number of unfinished jobs at **t** in **ALG**
- $n_o(t)$: number of unfinished jobs at **t** in **OPT**

We show that at any time **t**,

$$n_a(t) \leq c \cdot n_o(t), \text{ or equivalently, } \frac{dF_a(t)}{dt} \leq c \cdot \frac{dF_o(t)}{dt}$$

Therefore, at any time **t**,

$$F_a(t) \leq c F_o(t) .$$

That is, $F_a \leq c F_o$.



Local competitiveness (cont')

Consider a processor with **dynamic speed scaling**.

For minimizing total flow time plus energy,

- **G(t): F + E** incurred up to **t**
- **s(t):** speed at **t**

$$\frac{dG(t)}{dt} = n(t) + s(t)^\alpha$$

- At some time **t**, **OPT** may run much slower than **ALG**, so we cannot bound $\frac{dG_a(t)}{dt}$ in terms of $\frac{dG_o(t)}{dt}$.

- It is **impossible** to show that at any time **t**,

$$\frac{dG_a(t)}{dt} \leq c \cdot \frac{dG_o(t)}{dt}$$

- Local competitiveness does not work!



Amortized local competitiveness

- Enhance local competitiveness using a potential function Φ .
- At any time t , we want to show

$$G_a(t) + \Phi(t) \leq c G_o(t)$$

- To achieve this, we want Φ to satisfy three conditions:

1. Boundary condition:

$\Phi(t) = 0$ at $t = 0$ & after ALG and OPT completes all jobs.

2. Job arrival and completion:

$\Phi(t)$ does not increase at such time.

3. Running condition: $\frac{dG_a(t)}{dt} + \frac{d\Phi(t)}{dt} \leq c \cdot \frac{dG_o(t)}{dt}$ for some $c \geq 1$

By induction, at any time t , $G_a(t) + \Phi(t) \leq c \cdot G_o(t)$

Therefore, $G_a \leq c \cdot G_o$



Amortized local competitiveness (cont')

- At any time t , we want to show

$$\mathbf{G}_a(t) + \Phi(t) \leq \mathbf{c} \mathbf{G}_o(t)$$

- No general way to design the potential function Φ .
- Intuitively, if $\mathbf{G}_a(t) \gg \mathbf{G}_o(t)$, then
ALG's remaining workload \ll OPT's remaining workload
- $\mathbf{G}_a(t) + (\text{ALG's workload}) \leq \mathbf{c} \mathbf{G}_o(t) + (\text{OPT's workload})$
- Φ should relate to $(\text{ALG's workload}) - (\text{OPT's workload})$
– capture difference in **progress** of the online algorithm
ALG and **OPT**



Outline

- Models and problems
 - Speed scaling
 - Minimize flow time plus energy
- Amortized local competitiveness
- **Analysis techniques**
 - **Fractional & Integral flow time**
 - From Unbounded speed to Bounded speed
 - Application of Young's Inequality



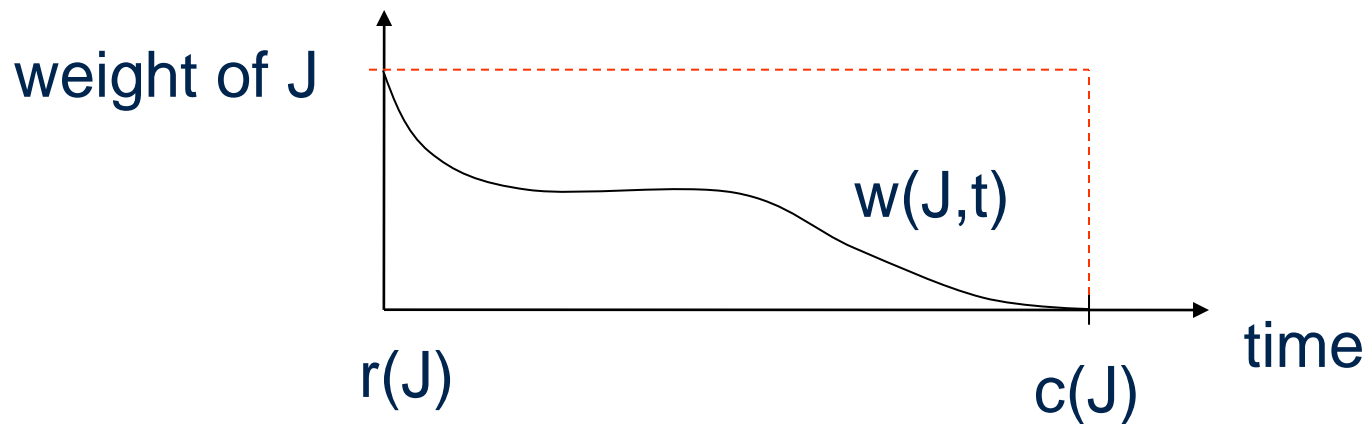
Fractional flow time

[Bansal, Pruhs, Stein; SODA 07]

First, consider minimizing **fractional** flow time plus energy.

- **fractional weight** of **J** at **t**

$$w(J, t) = (\text{weight of } J) \times (\text{remaining fraction of } J \text{ at } t)$$



- **fractional flow time** of **J** = $\int_{r(J)}^{c(J)} w(J, t) dt$
- Total flow time **is lower bound by** fractional flow time



Fractional flow time (cont')

Fractional flow time

- Total fractional weight of all unfinished jobs decreases continuously with job execution.
- Highest Density First is optimal for minimizing fractional F.
 - density of a job = weight / size

Total (integral) flow time

- Number of unfinished jobs decreases discretely at job completions.
- For weighted jobs, there is no known optimal job selection algorithm for minimizing weighted flow time.

Easier to design potential function for fractional F+E.



Minimize fractional $F + E$

[Bansal, Pruhs, Stein; SODA 07]

Consider **unit-size** (unit-weight) jobs.

Algorithm:

- Job selection: Run each job one after another (=HDF)
- Speed scaling: $s(t) = \mathbf{w}(t)^{1/\alpha}$,

where $\mathbf{w}(t)$ is the total fractional weight at t .

- $G(t)$: total **fractional $F+E$** incurred up to t .

- $$\frac{dG_a(t)}{dt} = w_a(t) + s_a(t)^\alpha = 2w_a(t)$$

We can use amortized local competitiveness to show that this algorithm is **2-competitive** for **fractional $F + E$** .



Minimize fractional F + E (cont')

Consider unit-size jobs. [Bansal, Pruhs, Stein; SODA 07]

- Potential function:

$$\Phi(t) = \frac{2\alpha}{\beta+1} \max(0, w_a(t) - w_o(t))^{\beta+1} \text{ where } \beta = 1 - \frac{1}{\alpha}.$$

- Boundary condition:

$\Phi(t) = 0$ at $t = 0$ & after ALG and OPT completes all jobs.

- Job arrival:

$w_a(t)$ and $w_o(t)$ increase by 1, so $\Phi(t)$ is not changed.

- Job completion:

$\Phi(t)$ remains the same.

- Running condition (remain to prove):

$$\frac{dG_a(t)}{dt} + \frac{d\Phi(t)}{dt} \leq 2 \cdot \frac{dG_o(t)}{dt}$$



Minimize fractional F + E (cont')

Consider unit-size jobs. [Bansal, Pruhs, Stein; SODA 07]

- **Potential function:**

$$\Phi(t) = \frac{2\alpha}{\beta+1} \max(0, w_a(t) - w_o(t))^{\beta+1} \text{ where } \beta = 1 - \frac{1}{\alpha}.$$

- **Running condition:** $\frac{dG_a(t)}{dt} + \frac{d\Phi(t)}{dt} \leq 2 \cdot \frac{dG_o(t)}{dt}$

Case 1: $w_a(t) < w_o(t)$

- $\frac{d\Phi(t)}{dt} = 0$

- $\frac{dG_a(t)}{dt} = 2w_a(t) < 2w_o(t) \leq 2 \frac{dG_o(t)}{dt}$



Minimize fractional F + E (cont')

Consider unit-size jobs. [Bansal, Pruhs, Stein; SODA 07]

- **Potential function:**

$$\Phi(t) = \frac{2\alpha}{\beta+1} \max(0, w_a(t) - w_o(t))^{\beta+1} \text{ where } \beta = 1 - \frac{1}{\alpha}.$$

- **Running condition:** $\frac{dG_a(t)}{dt} + \frac{d\Phi(t)}{dt} \leq 2 \cdot \frac{dG_o(t)}{dt}$
 $\Rightarrow 2w_a(t) + \frac{d\Phi(t)}{dt} \leq 2 \cdot (w_o(t) + s_o(t)^\alpha)$

Case 2: $w_a(t) \geq w_o(t)$

$$\begin{aligned} \bullet \frac{d\Phi}{dt} &= \frac{2\alpha}{\beta+1} \frac{d(w_a - w_o)^{\beta+1}}{dt} = 2\alpha(w_a - w_o)^\beta \frac{d(w_a - w_o)}{dt} \\ &= 2\alpha(w_a - w_o)^\beta (-s_a + s_o) \end{aligned}$$

- Since $s_a = w_a^{1/\alpha}$,

$$\frac{d\Phi}{dt} \leq 2\alpha(-(w_a - w_o) + (w_a - w_o)^{1 - \frac{1}{\alpha}} s_o)$$



Minimize fractional F + E (cont')

Consider unit-size jobs. [Bansal, Pruhs, Stein; SODA 07]

- Running condition:

$$2w_a(t) + \frac{d\Phi(t)}{dt} \leq 2 \cdot (w_o(t) + s_o(t)^\alpha)$$

Case 2: $w_a(t) \geq w_o(t)$

- $\frac{d\Phi}{dt} \leq 2\alpha(-(w_a - w_o) + (w_a - w_o)^{1-\frac{1}{\alpha}} s_o)$

Young's Inequality:

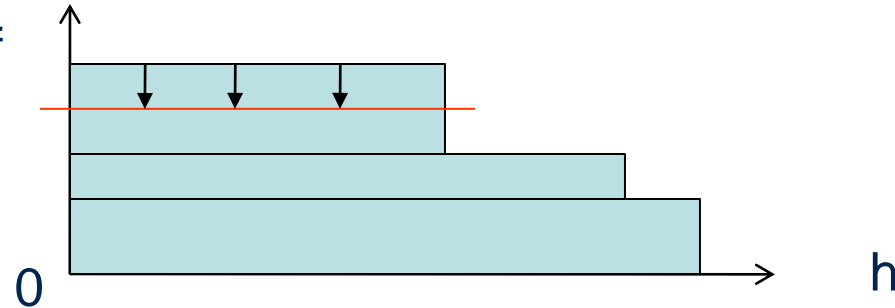
For any $x, y, p, q > 0$ & $p + q = 1$, $x^p y^q \leq px + qy$

- $\frac{d\Phi}{dt} \leq 2\alpha(-(w_a - w_o) + (1 - \frac{1}{\alpha})(w_a - w_o) + \frac{1}{\alpha}s_o^\alpha)$
 $= 2 \cdot (w_o + s_o^\alpha) - 2w_a$
- So, the algorithm is **2-competitive** for **fractional F+E**.
- For **integral F+E**, the competitive ratio is increased to **4**.



Minimize fractional $F + E$ (cont')

total fractional weight of jobs with $1/\text{density} \geq h$



[Bansal, Pruhs, Stein; SODA 07]

For general jobs (arbitrary size and weight),

- Algorithm A: HDF & $s(t) = w(t)^{1/\alpha}$
- Consider the execution of HDF.
- Design $\Phi(t)$ to account for the change of ALG and OPT.
- $O\left(\frac{\alpha}{\log \alpha}\right)$ - competitive for fractional $F + E$.



Minimize (integral) F + E

[Bansal, Pruhs, Stein; SODA 07]

Algorithm A: HDF & $s(t) = w(t)^{1/\alpha}$

- $O\left(\frac{\alpha}{\log \alpha}\right)$ - competitive for fractional F+ E.

Algorithm B: Simulate A, then run HDF & $(1 + \varepsilon) s(t)$, $\varepsilon > 0$

If J is unfinished in B, then J has at least $\frac{\varepsilon}{1 + \varepsilon}$ fraction of work unfinished in A.

- Integral flow time of B $\leq \frac{1 + \varepsilon}{\varepsilon}$ x (fractional flow time of A)
- Energy of B $\leq (1 + \varepsilon)^\alpha$ x (energy of A)
- For integral F+ E, the competitive ratio of B is

$$\max\left\{1 + \frac{1}{\varepsilon}, (1 + \varepsilon)^\alpha\right\} \cdot O\left(\frac{\alpha}{\log \alpha}\right) = O\left(\left(\frac{\alpha}{\log \alpha}\right)^2\right)$$



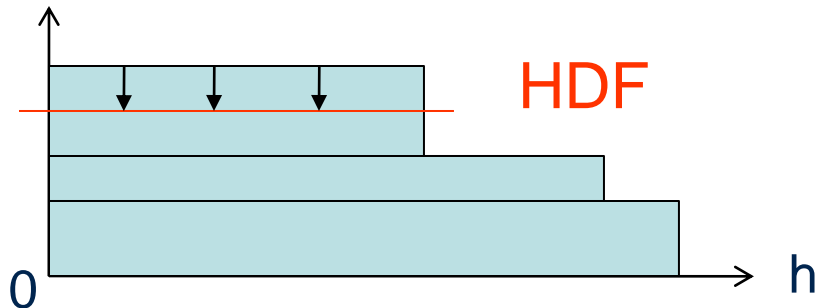
Minimize (integral) F + E (cont')

For **unweighted** jobs, [Lam, Lee, To, Wong; ESA 08] reduces the competitive ratio from $O\left(\left(\frac{\alpha}{\log \alpha}\right)^2\right)$ to $O\left(\frac{\alpha}{\log \alpha}\right)$.

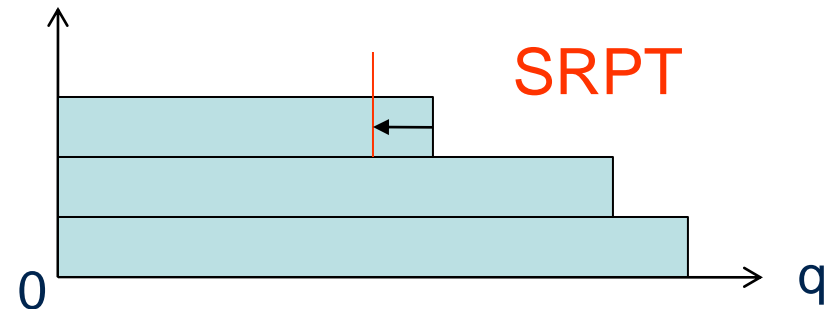
Idea: Analyzing (integral) flow time directly.

- Algorithm: **SRPT** (optimal job selection) & $s(t) = n(t)^{1/\alpha}$

fractional weight of jobs
with $1/\text{density} \geq h$



number of jobs
with remaining work $\geq q$



- Question:** For **weighted** jobs, can we analyze (integral) weighted flow time directly to achieve better ratio?

List of results

Unbounded speed model

- Unweighted jobs:

$O\left(\left(\frac{\alpha}{\log \alpha}\right)^2\right)$ -competitive [SODA 07]

$O\left(\frac{\alpha}{\log \alpha}\right)$ -competitive [ESA 08]

3-competitive [SODA 09]

2-competitive [Perf. Eval. Review 09]

- Weighted jobs:

$O\left(\left(\frac{\alpha}{\log \alpha}\right)^2\right)$ -competitive [SODA 07]

$O\left(\frac{\alpha}{\log \alpha}\right)$ -competitive [SODA 09]



Outline

- Models and problems
 - Speed scaling
 - Minimize flow time plus energy
- Amortized local competitiveness
- Analysis techniques
 - Fractional & Integral flow time
 - **From Unbounded speed to Bounded speed**
 - Application of Young's Inequality



Bounded speed model

- Upper limit **T** on processor speed.
- Speed scaling: cap the speed at **T**

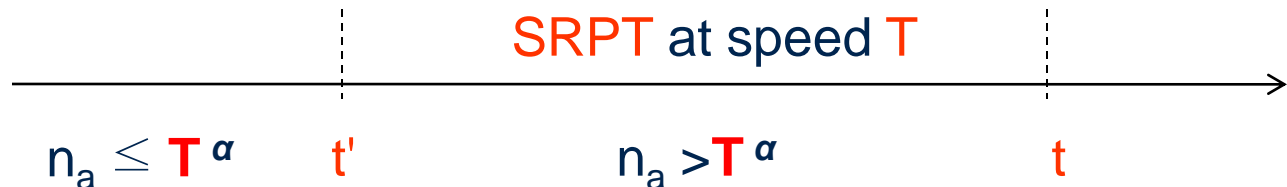
$$s(t) = \min\{n(t)^{1/\alpha}, T\}$$

- $\Phi(t)$ decreases due to execution of ALG.

– $\frac{d\Phi(t)}{dt}$ decreases with speed s_a .

- With max speed **T**, the decrease is not enough to make the running condition hold.
- **Property:** At any time t , $n_a(t) - n_o(t) \leq T^\alpha$.

- Proof:



SRPT is optimal job selection for flow time

So $n_a(t) - n_o(t)$ does not increase during $[t', t]$.



Bounded speed model (cont')

- **Property:** At any time t , $n_a(t) - n_o(t) \leq T^\alpha$.
- The difference in progress between ALG and OPT is bounded.
- We can still prove the running condition.

For weighted jobs,

- **Property:** At any time t , $w_a(t) - w_o(t) \leq T^\alpha$.
 - HDF is optimal for **fractional** flow time.
- We can still prove the running condition.



Non-clairvoyant setting

- Job size is not known when the job arrives.
- Job size is only known when the job completes.
 - No information on remaining work of jobs.

Unbounded speed model

- Algorithm **LAPS** (Latest Arrival Processor Sharing):
 - Run a collection of **latest arrived** jobs using processor sharing at $s(t) = O(n(t)^{1/\alpha})$ [C.,E.,L.,L.,M.,P.; STACS 09]
- $O\left(\frac{\alpha^2}{\log \alpha}\right)$ -competitive for $F + E$



Non-clairvoyant setting (cont')

Bounded speed model

- Since remaining work is unknown, non-clairvoyant algorithms usually run jobs by **processor sharing**.
- $n_a(t) - n_o(t)$ is unbounded at some time t .

New analysis technique: [Chan, Lam, Lee, Ting, Zhang; CATS 10]

- **Incorporate** the max speed **T** into the **potential function**.
- LAPS is $O(\alpha^2)$ -competitive for $F + E$.
- For weighted jobs, using processor of max speed $(1 + \varepsilon) T$
a variant of LAPS is $O(\max\{\alpha^2, (1 + \frac{1}{\varepsilon})^2\})$ -competitive
for weighted $F + E$.



Outline

- Models and problems
 - Speed scaling
 - Minimize flow time plus energy
- Amortized local competitiveness
- Analysis techniques
 - Fractional & Integral flow time
 - From Unbounded speed to Bounded speed
 - **Application of Young's Inequality**



Revisit Young's Inequality

In most of the existing work on minimizing F+E,

- Speed is set such that **flow time is proportional to energy**.
- Thus, $\frac{dG_a}{dt} = O(n_a)$
- We always get $\frac{d\Phi}{dt} = O(n_o - n_a) + O((n_a - n_o)^{1-\frac{1}{\alpha}} s_o)$

Young's Inequality:

For any $x, y, p, q > 0$ & $p + q = 1$, $x^p y^q \leq px + qy$

- $\frac{d\Phi}{dt} = O(n_o - n_a) + O((n_a - n_o) + s_o^\alpha)$
- Therefore,
$$\begin{aligned} \frac{dG_a}{dt} + \frac{d\Phi}{dt} &= O(n_a) + O(n_o - n_a + s_o^\alpha) \\ &= O(n_o + s_o^\alpha) = O\left(\frac{dG_o}{dt}\right) \end{aligned}$$



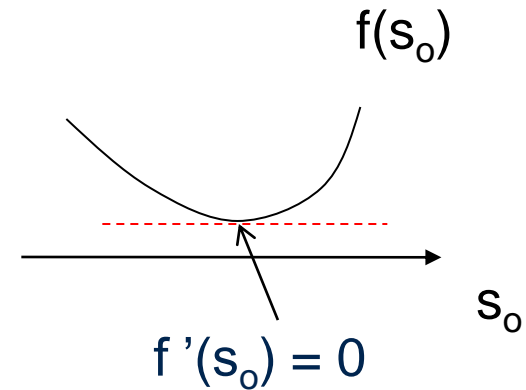
Replace Young's Inequality

We want to show that

$$c \cdot \frac{dG_o}{dt} - \left(\frac{dG_a}{dt} + \frac{d\Phi}{dt} \right) \geq 0$$

$$c \cdot (n_o + s_o^\alpha) - O(n_a + n_o + (n_a - n_o)^{1-\frac{1}{\alpha}} s_o) \geq 0$$

- L.H.S. is a convex function of s_o .
- It suffices to show that the inequality holds at the **minimum point** where $f'(s_o) = 0$.
- Easier to optimize the constants in the potential function.
 - Result is at least as good as using Young's Inequality.
- E.g., LAPS: $O\left(\frac{\alpha^2}{\log \alpha}\right) \Rightarrow O\left(\frac{\alpha^2}{\log^2 \alpha}\right)$ [B,C,E,P; 09]



Other directions

- Scheduling on **multiple processors**

Open problem in the bounded speed model:

Without extra max speed, does $O(\log P)$ -competitive clairvoyant algorithm exist? (P = max-min ratio of work)

- **Arbitrary power function**

- **Clairvoyant:**

Unweighted jobs: 2-competitive [Perf. Eval. Review 09]

Weighted jobs: $O\left(\frac{\alpha}{\log \alpha}\right)$ -competitive [SODA 09]

- **Non-clairvoyant:** Some power function does not admit $O(1)$ -competitive algorithm.

- Scheduling on processor with **sleep states**



Thank you!



max planck institut
informatik