An Introduction to Competitive Analysis in Online Scheduling

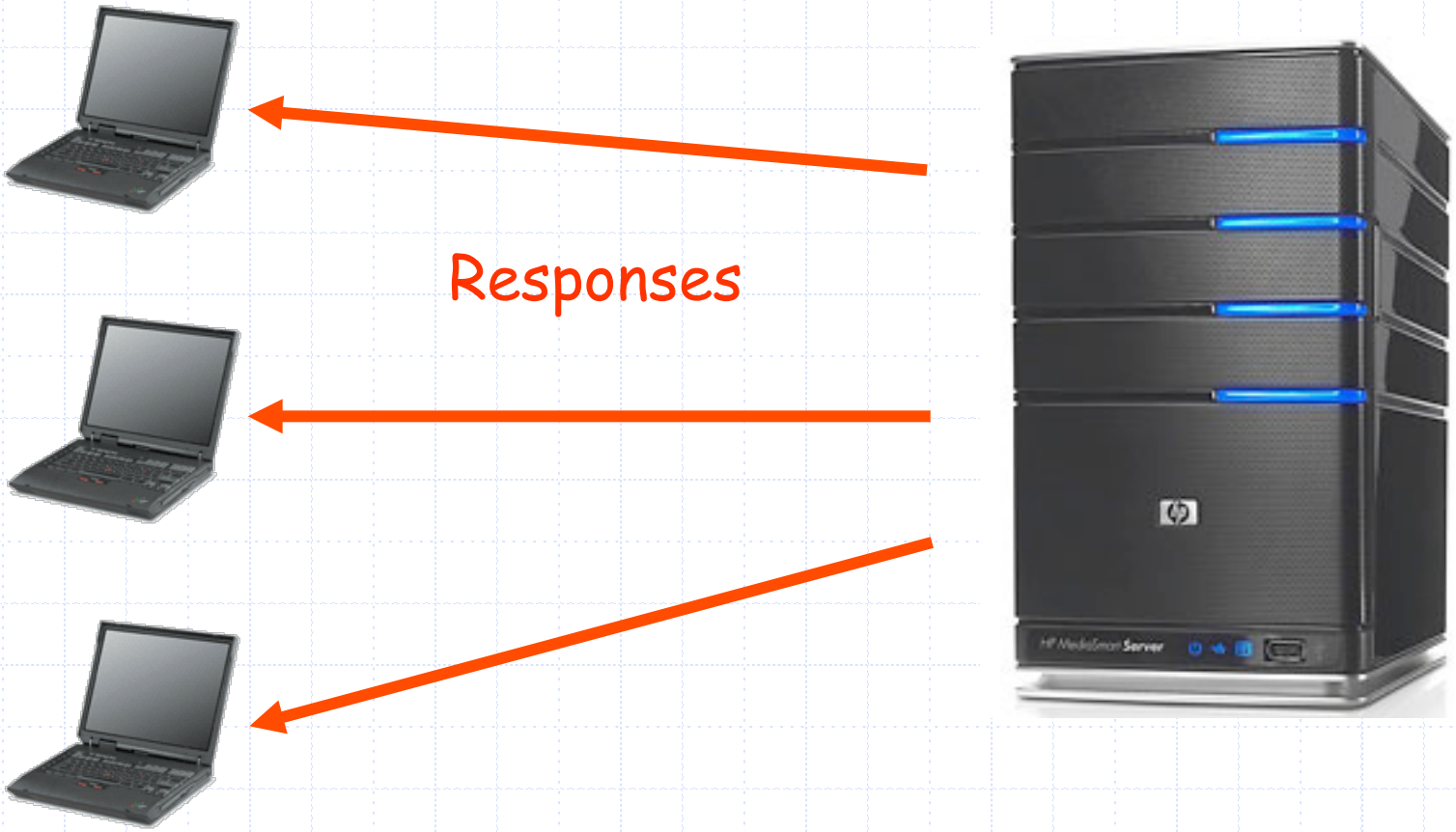
Kirk Pruhs

Scheduling Under Uncertainty Workshop

Eindhoven, June 1, 2015

Client-Server System

Client-Server System



Client-Server System

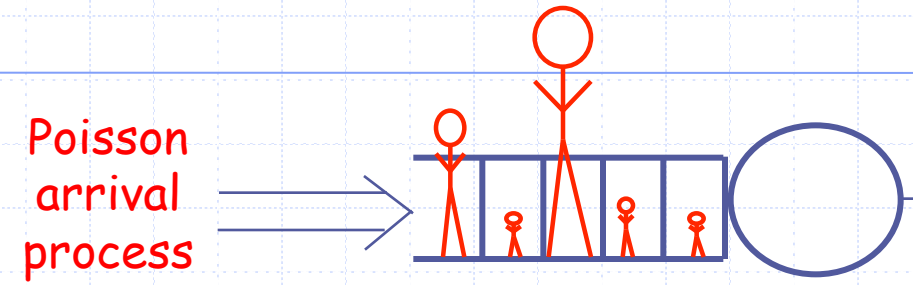
Responses

Server scheduling problem:
How does the server decide which requests to respond to first to minimize average response time?

Clients

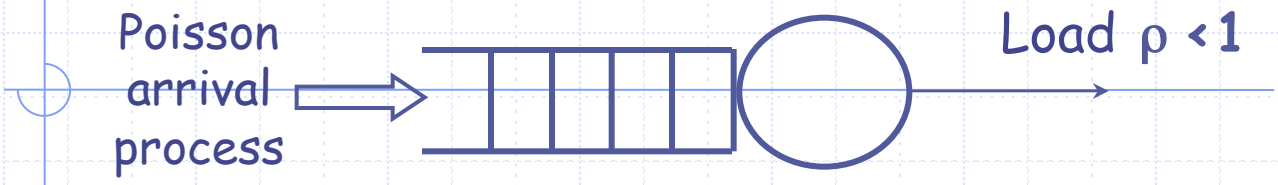
Server

Single Server Model (M/G/1)



Queuing Theoretic Approach
Slides from Mor Harchol-Balter
Lunteren Presentation

Scheduling Single Server (M/G/1)



Question: Order these scheduling policies for mean response time, E[T]:

1. FCFS (First-Come-First-Served, non-preemptive)
2. PS (Processor-Sharing, preemptive)
3. SJF (Shortest-Job-First, a.k.a., SPT, non-preemptive)
4. SRPT (Shortest-Remaining-Processing-Time, preemptive)
5. LAS (Least Attained Service, a.k.a., FB, preemptive)

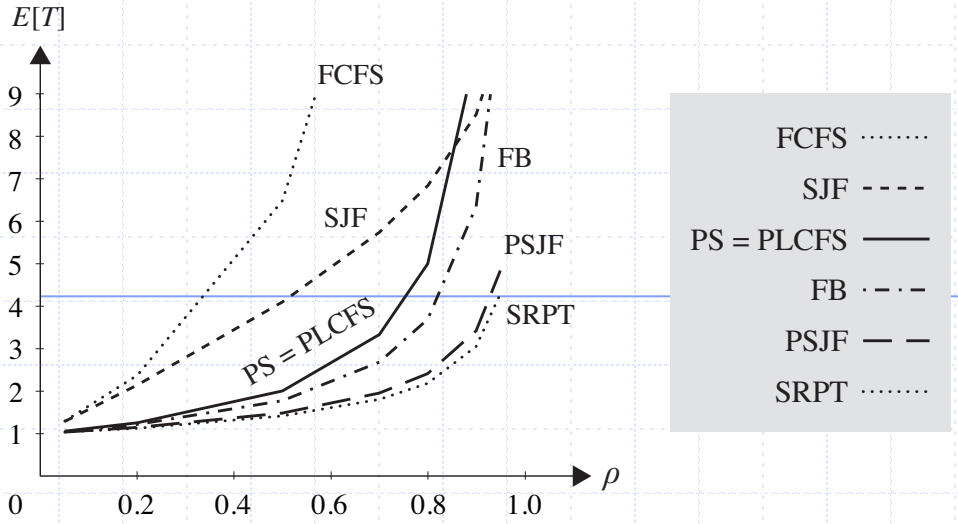
Scheduling Single Server (M/G/1)

Question: Order these scheduling policies for mean response time, E[T]:

Answer:

Low E[T] High E[T]

SRPT < PSJF < LAS < PS < FCFS



Plan for This Talk:

- Give an introduction to worst-case / competitive analysis
- Via ordering of these 5 scheduling algorithms using competitive analysis

Competitive Analysis Order:
Low total flow **High total flow**

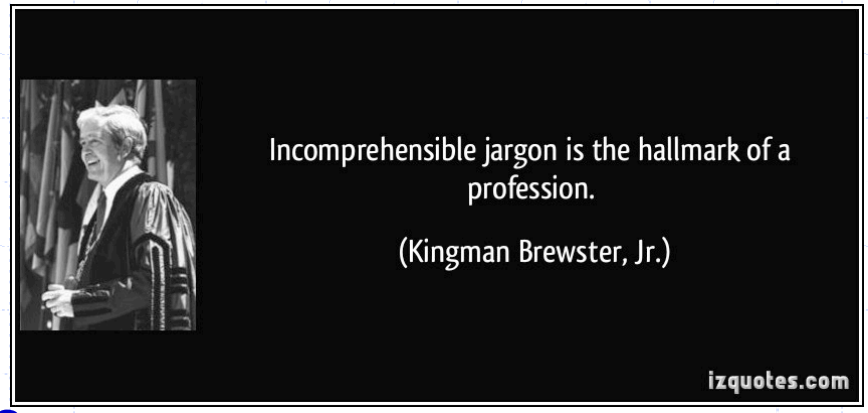
SRPT < PSJF = LAS < PS < FCFS

Recall Queuing Theory Order:
Low E[T] **High E[T]**

SRPT < PSJF < LAS < PS < FCFS

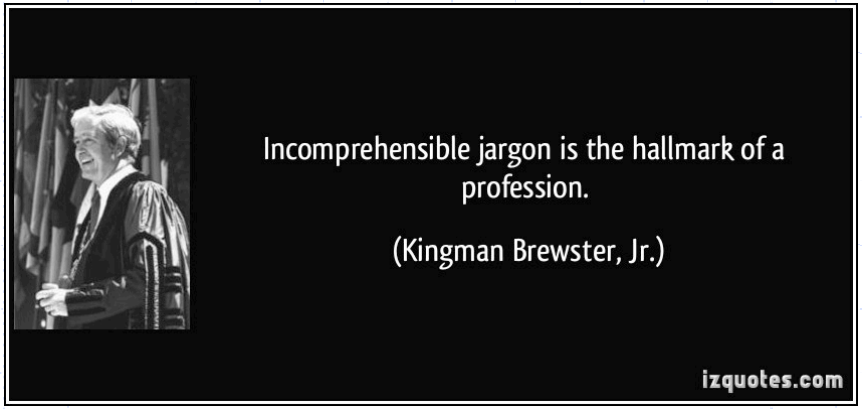
Why?

- Worst-case researchers should know enough queuing theory to
  - Understand queuing theoretic papers
  - Do basic queuing theory analyses in case of emergency



Why?

- Queuing theory researchers should know enough worst-case analysis theory to
  - Understand worst-case papers
  - Do basic worst-case analyses in case of emergency



Outline(1)

- Review of basic problem/terminology
- SRPT is optimal
- Detour: Stretch objective
  - No online optimal algorithm for average stretch
  - Definition of competitiveness
  - Local Competitiveness: SRPT is 2-competitive for average stretch
- Negative Results:
  - SJF and LAS are not competitive
  - No nonclairvoyant competitive algorithm
- Speed/Resource augmentation analysis
  - LAS and SJF are scalable

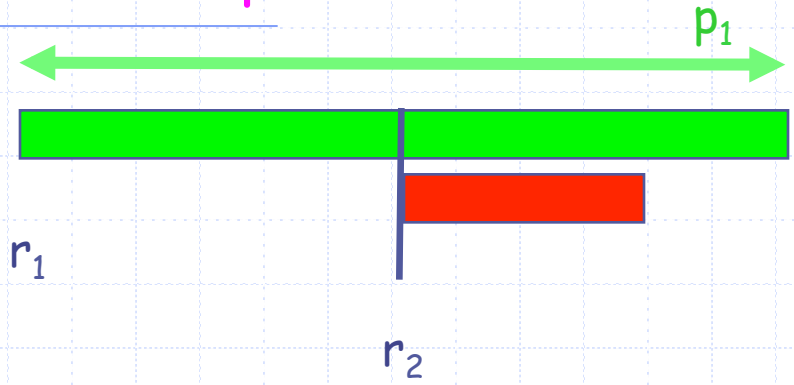
Outline(2)

- PS is 3-speed 2-competitive
  - Amortized local competitiveness analysis
  - Dual fitting analysis
- Negative Result: FCFS is not O(1)-speed O(1)-competitive
- Wrap-up

Outline(1)

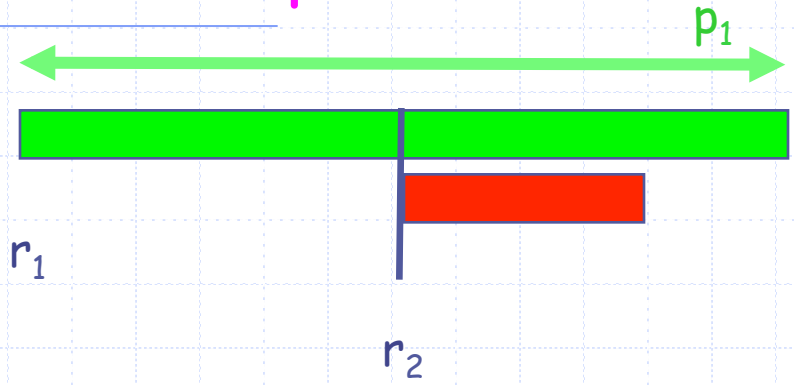
- Review of basic problem/terminology
- SRPT is optimal
- Detour: Stretch objective
  - No online optimal algorithm for average stretch
  - Definition of competitiveness
  - Local Competitiveness: SRPT is 2-competitive for average stretch
- Negative Results:
  - SJF and LAS are not competitive
  - No nonclairvoyant competitive algorithm
- Speed/Resource augmentation analysis
  - LAS and SJF are scalable

Online Input:



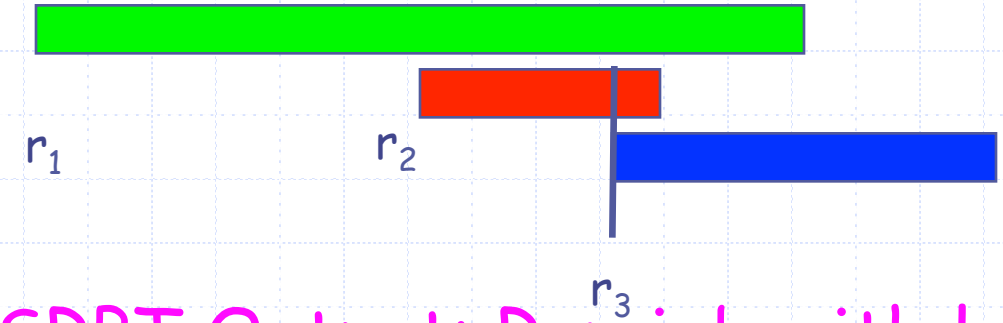
SRPT Output: Run job with least unprocessed volume

Online Input:



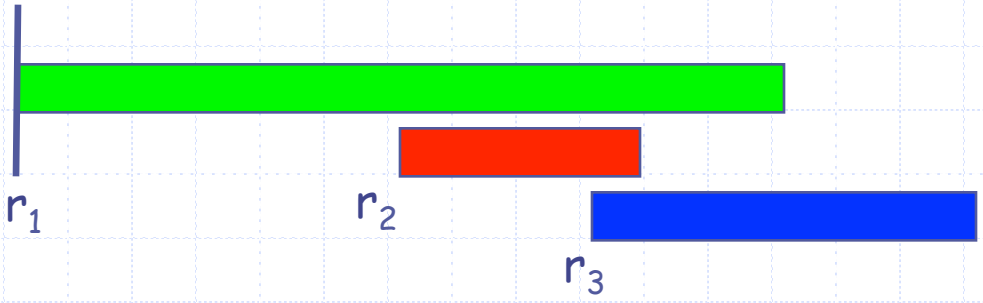
SRPT Output: Run job with least unprocessed volume

Online Input:

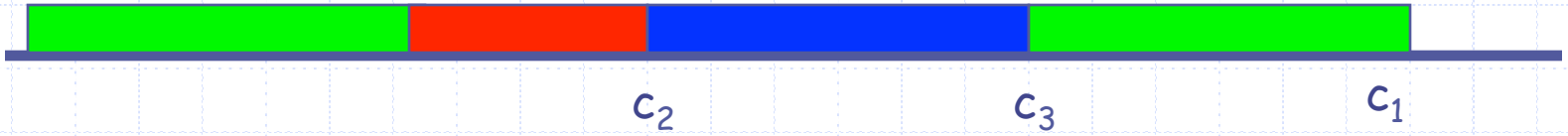


SRPT Output: Run job with least unprocessed volume

Online Input:

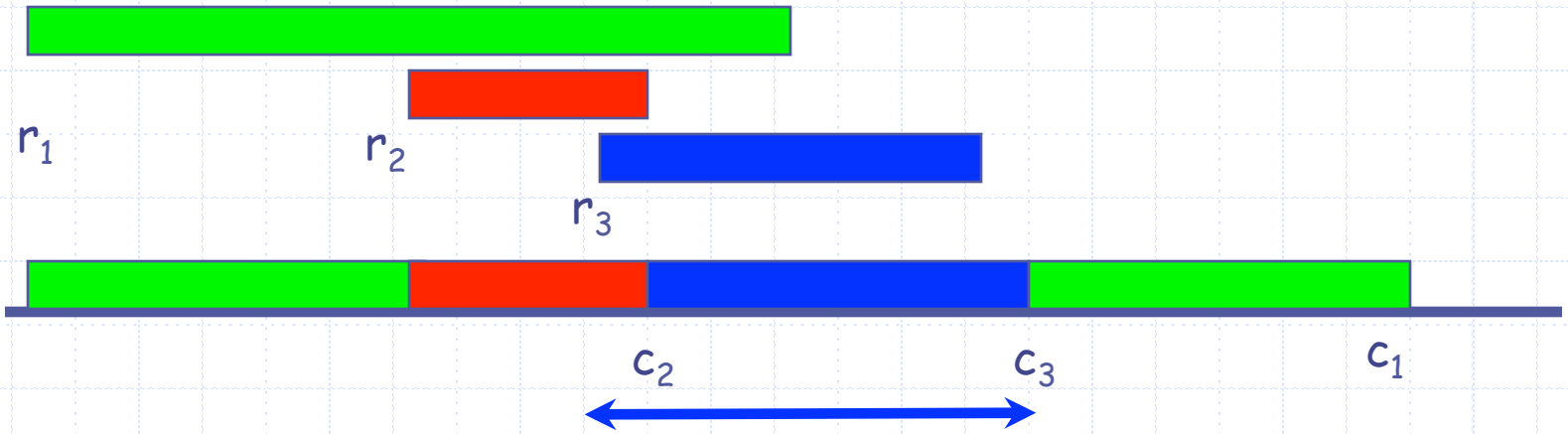


SRPT Output: Run job with least unprocessed volume



Flow Time Quality of Service (QoS) Measure of a Job

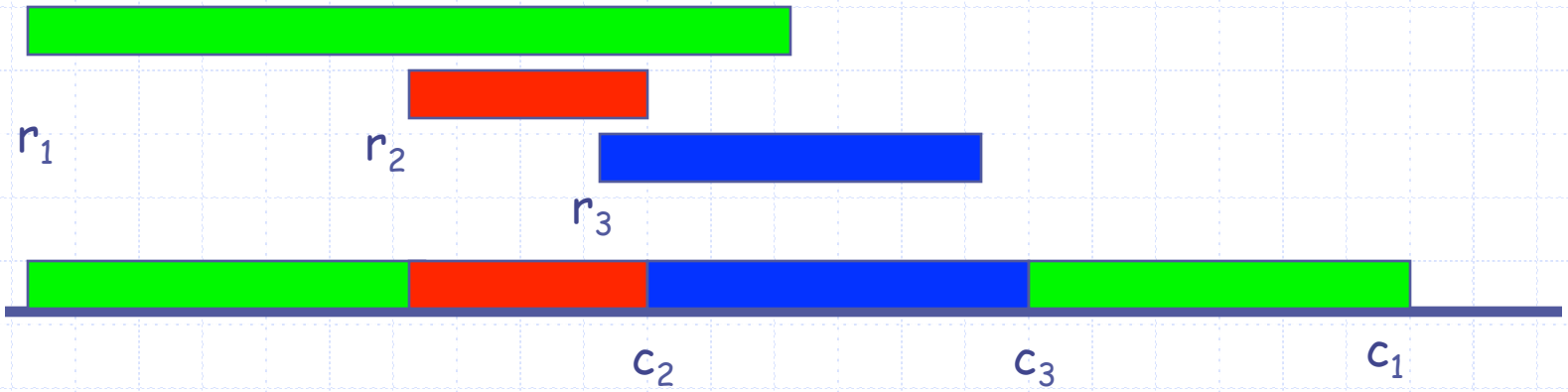
- Completion time: c_j
- Flow/response time: $f_j = c_j - r_j$



Flow Time of Job 3

Most Common Quality of Service (QoS) Measure of a Schedule

- Total flow time = $\Sigma_j f_j$
- Equivalently, average flow time = $\Sigma_j f_j / n$

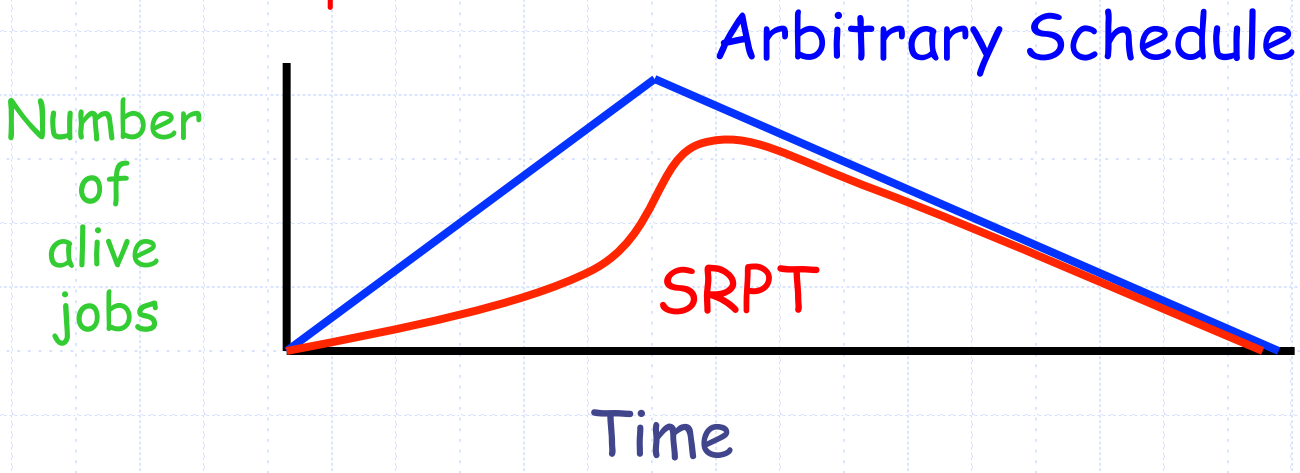


Outline(1)

- Review of basic problem/terminology
- SRPT is optimal
- Detour: Stretch objective
  - No online optimal algorithm for average stretch
  - Definition of competitiveness
  - Local Competitiveness: SRPT is 2-competitive for average stretch
- Negative Results:
  - SJF and LAS are not competitive
  - No nonclairvoyant competitive algorithm
- Speed/Resource augmentation analysis
  - LAS and SJF are scalable

Something Queuing Theorists and Worst-case Theorists can Agree on

- Theorem: SRPT is optimal for the objective of total flow
- Proof: Local Competitiveness



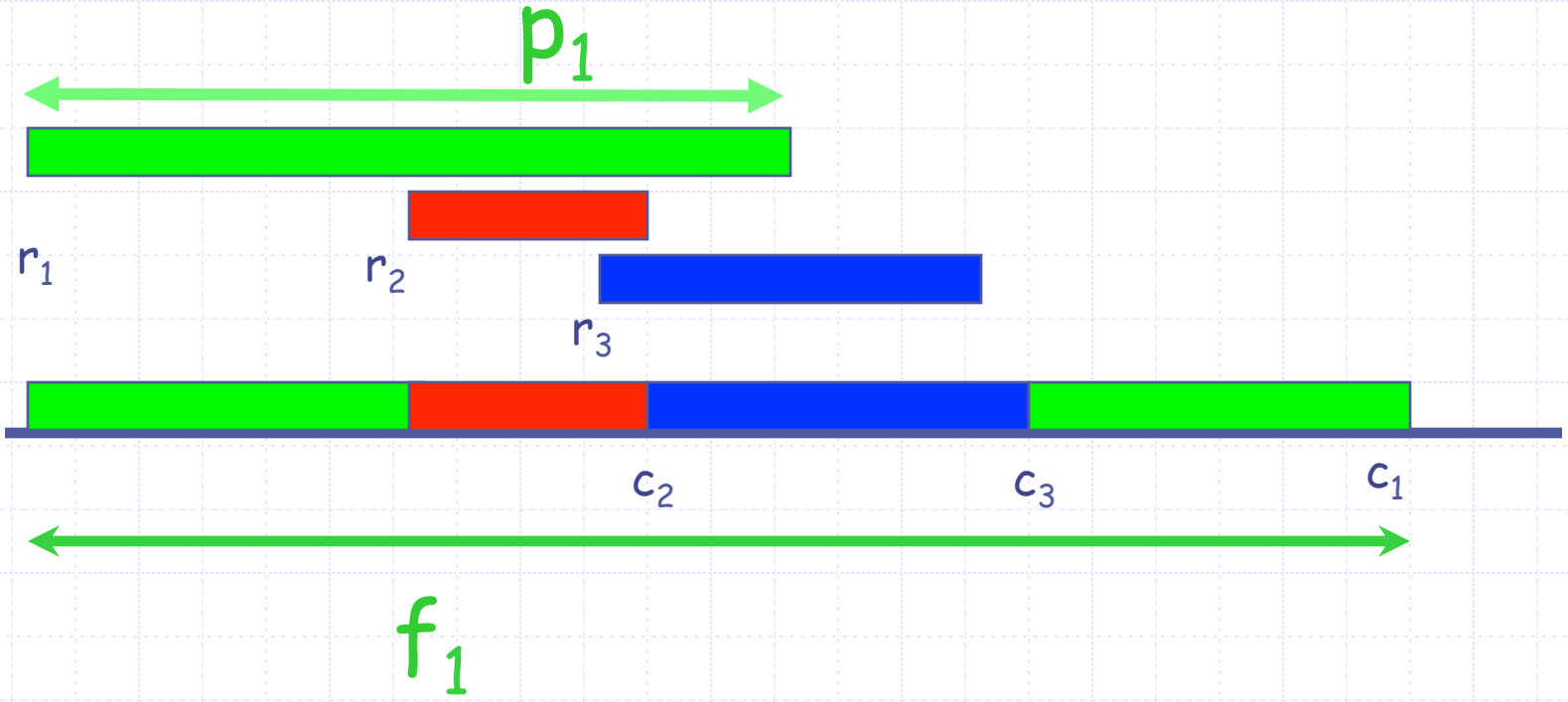
Total flow =
$\int_t$ number of alive jobs at time t dt

Outline(1)

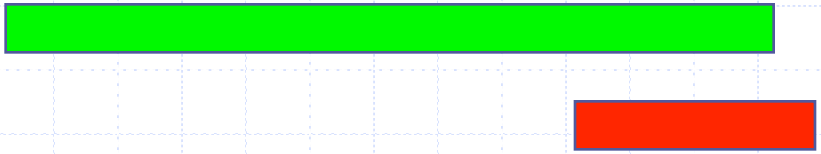
- Review of basic problem/terminology
- SRPT is optimal
- Detour: Stretch objective
  - No online optimal algorithm for average stretch
  - Definition of competitiveness
  - Local Competitiveness: SRPT is 2-competitive for average stretch
- Negative Results:
  - SJF and LAS are not competitive
  - No nonclairvoyant competitive algorithm
- Speed/Resource augmentation analysis
  - LAS and SJF are scalable

Stretch Quality of Service (QoS) Measure

- Stretch of job j: f_j / p_j
- Total stretch of a schedule : $\Sigma_j f_j/p_j$

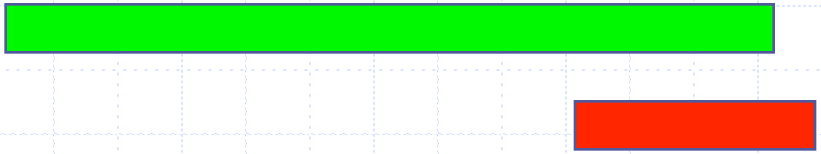


There is no online optimal algorithm for total stretch



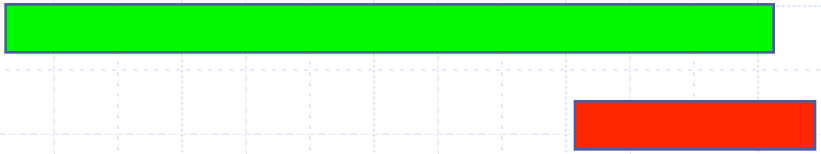
- Should the algorithm preempt for the red job or finish the green job?

There is no online optimal algorithm for total stretch

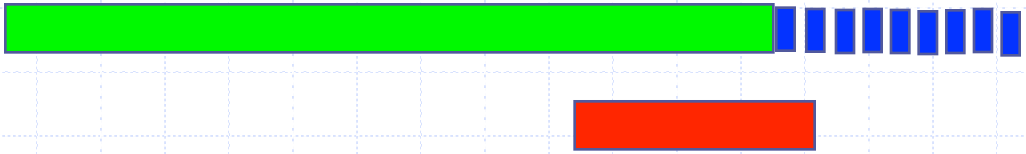


- If no more jobs arrive, then better to preempt for the red job. But this is a mistake if the future is what?

There is no online optimal algorithm for total stretch



- If no more jobs arrive, then better to preempt for the red job. But this is a mistake if the future is



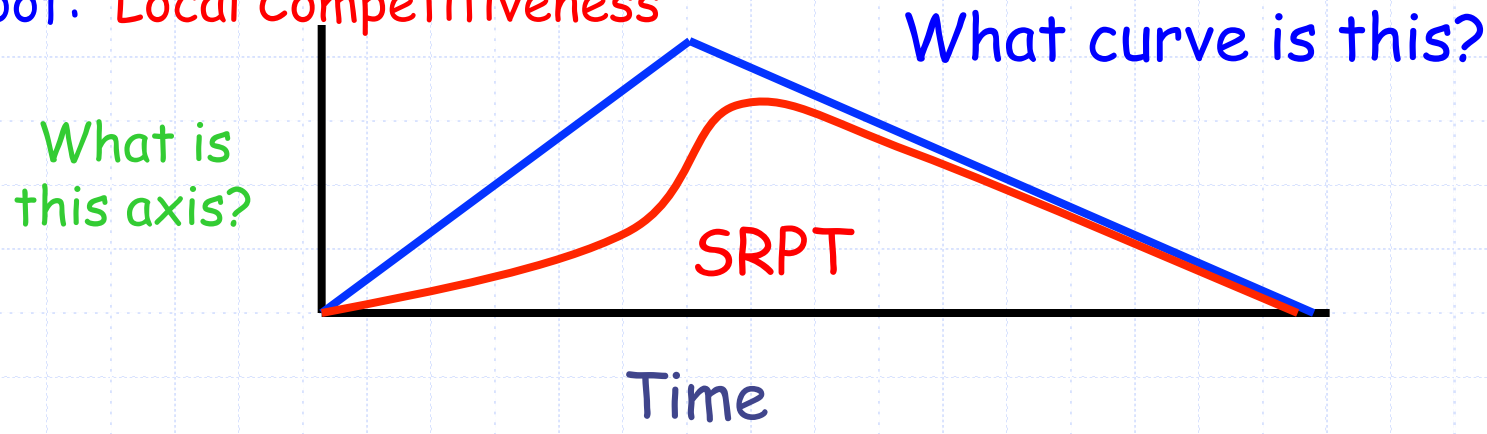
Competitive Analysis

- Philosophy: find schedule with small error
- Standard measure of error:
  - relative error = (A(I) – Opt(I))/Opt(I)
  - A(I) is approximation on input I
  - Opt(I) is optimal
- Competitive Analysis: Add 1 to relative error for convenience and take worst case over all inputs
- Competitive ratio of algorithm A is

  $\max_I A(I)/Opt(I)$
  - An algorithm is c-competitive if its competitive ratio $\leq c$

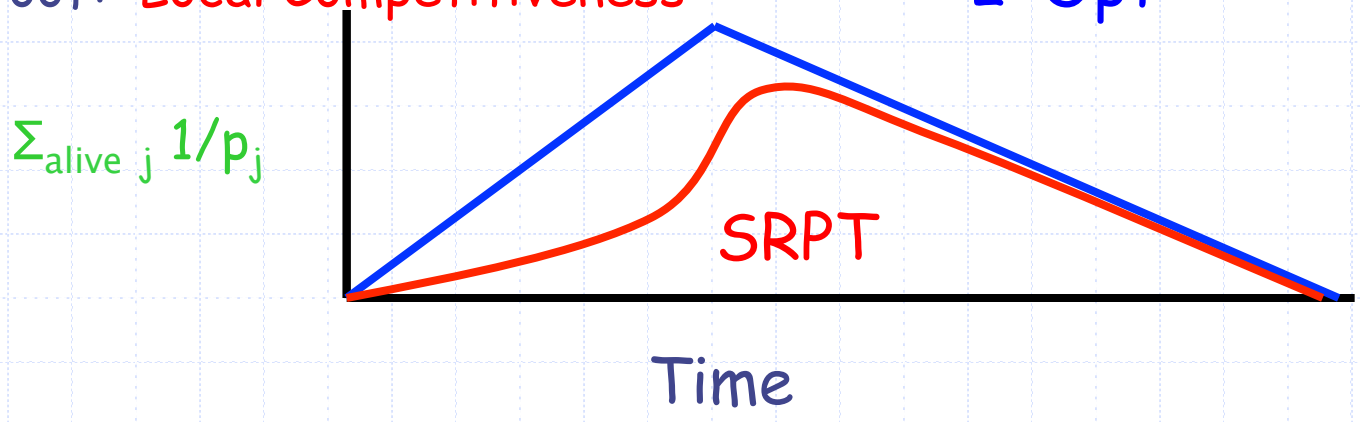
Competitiveness Result

- Theorem: SRPT is 2-competitive for the objective of total stretch
- Proof: Local Competitiveness



Competitiveness Result

- Theorem: SRPT is 2-competitive for the objective of total Stretch
- Proof: Local Competitiveness



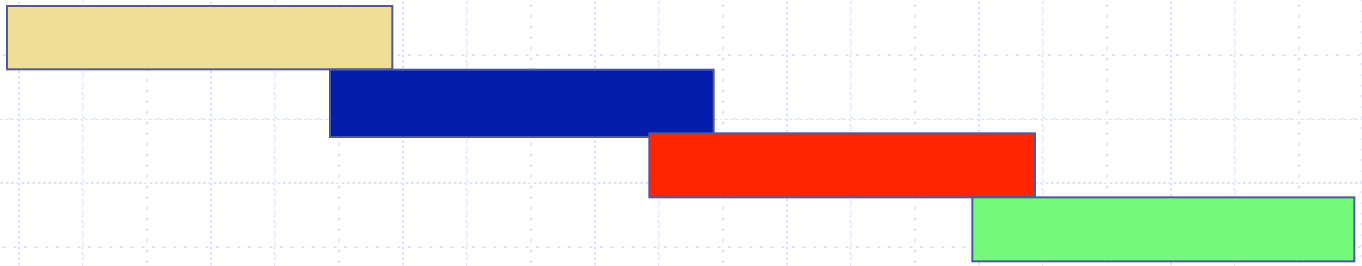
$$\text{Total stretch} = \int_t \left(\Sigma_{\text{alive jobs } j \text{ at time } t} 1/p_j \right) dt$$

Outline(1)

- Review of basic problem/terminology
- SRPT is optimal
- Detour: Stretch objective
  - No online optimal algorithm for average stretch
  - Definition of competitiveness
  - Local Competitiveness: SRPT is 2-competitive for average stretch
- Negative Results:
  - SJF and LAS are not competitive
  - No nonclairvoyant competitive algorithm
- Speed/Resource augmentation analysis
  - LAS and SJF are scalable

LAS and SJF

- SJF: Always run the job j with minimal processing time pj
- LAS/SETF: Always run the job that has been run the least so far
- Theorem: The competitiveness of both SJF and LAS is unbounded.
- Proof:



Nonclaivoyance Definition

- Nonclairvoyant algorithm: Doesn't use job sizes
  - Is SJF nonclairvoyant?
  - Is LAS nonclairvoyant?

Nonclaivoyance

- Nonclairvoyant algorithm: Doesn't use job sizes
  - Is SJF nonclairvoyant? No.
  - Is LAS nonclairvoyant? Yes.

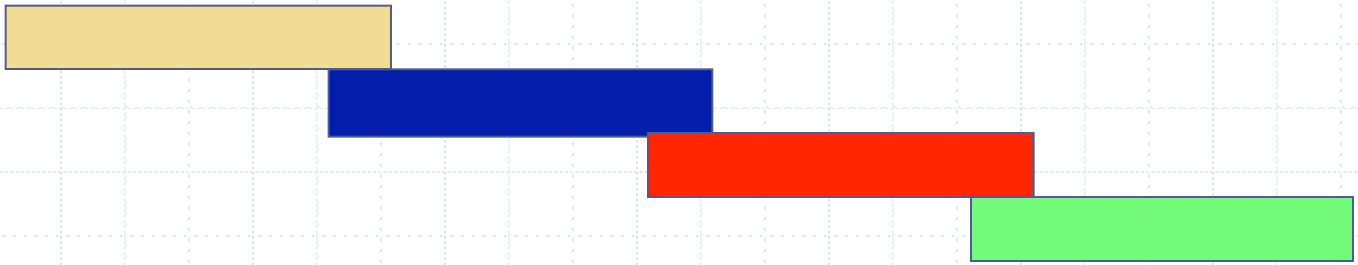
- Theorem: No (deterministic) nonclairvoyant algorithm A can be have bounded competitiveness for total flow.
- Proof:
  - Release n jobs of total size n at time 0.
  - At time n-1, A has 1/n units left on every job.
  - Optimal could have 1 unit left on 1 job
  - A stream of jobs of size 1/n arrive every 1/n time units for T units of time
  - A pays about n*T
  - Optimal pays about 2*T

Outline(1)

- Review of basic problem/terminology
- SRPT is optimal
- Detour: Stretch objective
  - No online optimal algorithm for average stretch
  - Definition of competitiveness
  - Local Competitiveness: SRPT is 2-competitive for average stretch
- Negative Results:
  - SJF and LAS are not competitive
  - No nonclairvoyant competitive algorithm
- Speed/Resource augmentation analysis
  - LAS and SJF are scalable

Observation that Motivated Resource Augmentation Analysis

- If SJF or LAS were just a bit faster, they would have done fine on this instance

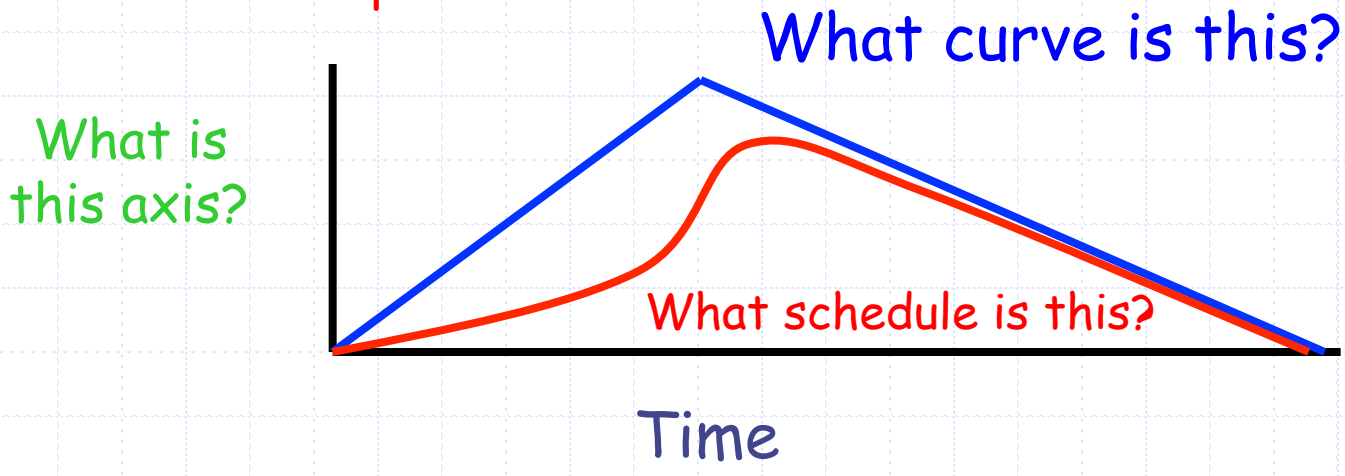


Resource/Speed Augmentation Analysis

- Online algorithm A is s-speed c-competitive if $\max_I A_s(I)/Opt_1(I) \leq c$
  - Subscript denotes processor speed
- A is scalable if it is $(1+\varepsilon)$-speed $O(1)$-competitive for all $\varepsilon > 0$.

Speed Augmentation Result

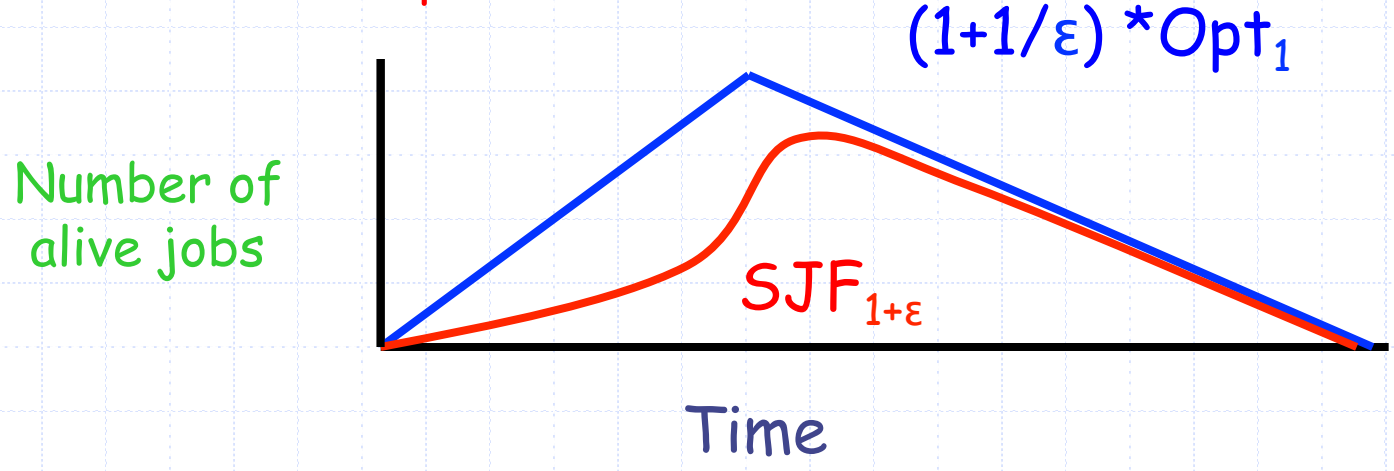
- Theorem: SJF is scalable for the objective of total flow
- Proof: Local Competitiveness



Recall: A is scalable if it is (1+ε)-speed constant-competitive for all ε > 0.

Speed Augmentation Result

- Theorem: SJF is scalable for the objective of total flow
- Proof: Local Competitiveness



Recall: A is scalable if it is (1+ε)-speed constant-competitive for all ε > 0.

Proving $SJF_{1+\varepsilon}(t) \leq (1+1/\varepsilon) * Opt_1(t)$

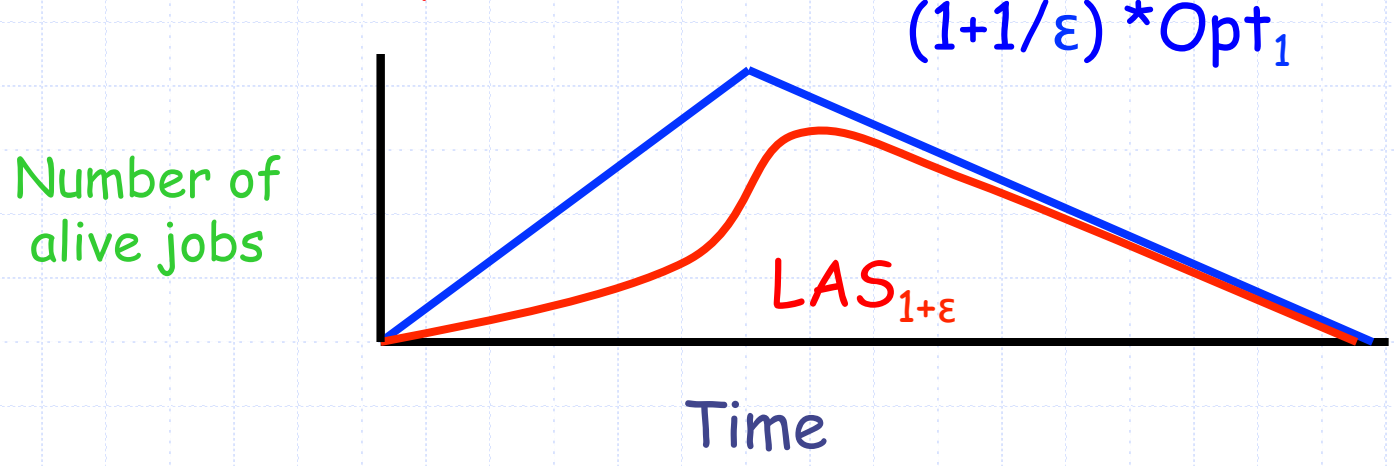
- Proof: Volume argument
- Worst-case: $SJF_{1+\varepsilon}$ has essentially no processing left on k jobs of unit size
- Opt_1 has to process ε more than Opt_1 on every job. What is the least number of jobs that Opt_1 can have unfinished?

Proving $SJF_{1+\varepsilon}(t) \leq (1+1/\varepsilon) * Opt_1(t)$

- Proof: Volume argument
- Worst-case: $SJF_{1+\varepsilon}$ has essentially no processing left on k jobs of unit size
- Opt_1 has to process ε more than Opt_1 on every job. What is the least number of jobs that Opt_1 can have unfinished?
- Opt_1 has εk jobs left that have not been processed at all, and has finished all the rest of the jobs

Speed Augmentation Result

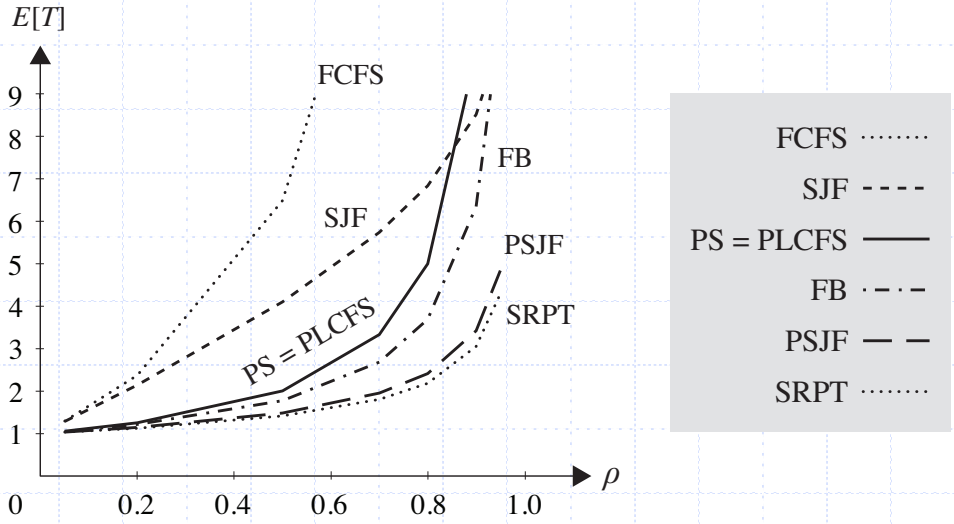
- Theorem: LAS is scalable for the objective of total flow
- Proof: Local Competitiveness



Recall: A is scalable if it is $(1+\varepsilon)$-speed constant-competitive for all $\varepsilon > 0$.

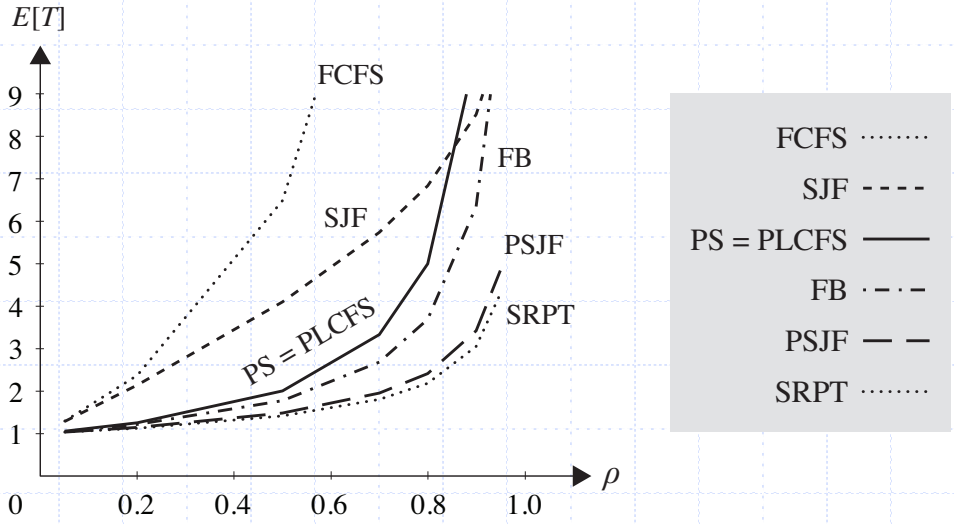
Queuing Theoretic Motivation for Speed Augmentation Analysis

- Question: For what sort of inputs below is SJF most likely not to be O(1)-competitive?

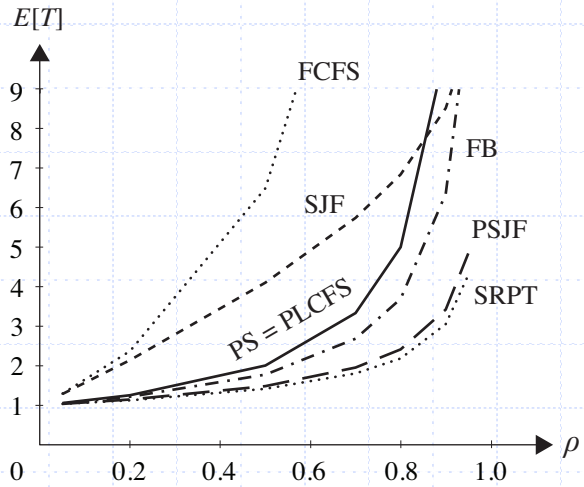


Queuing Theoretic Motivation for Speed Augmentation Analysis

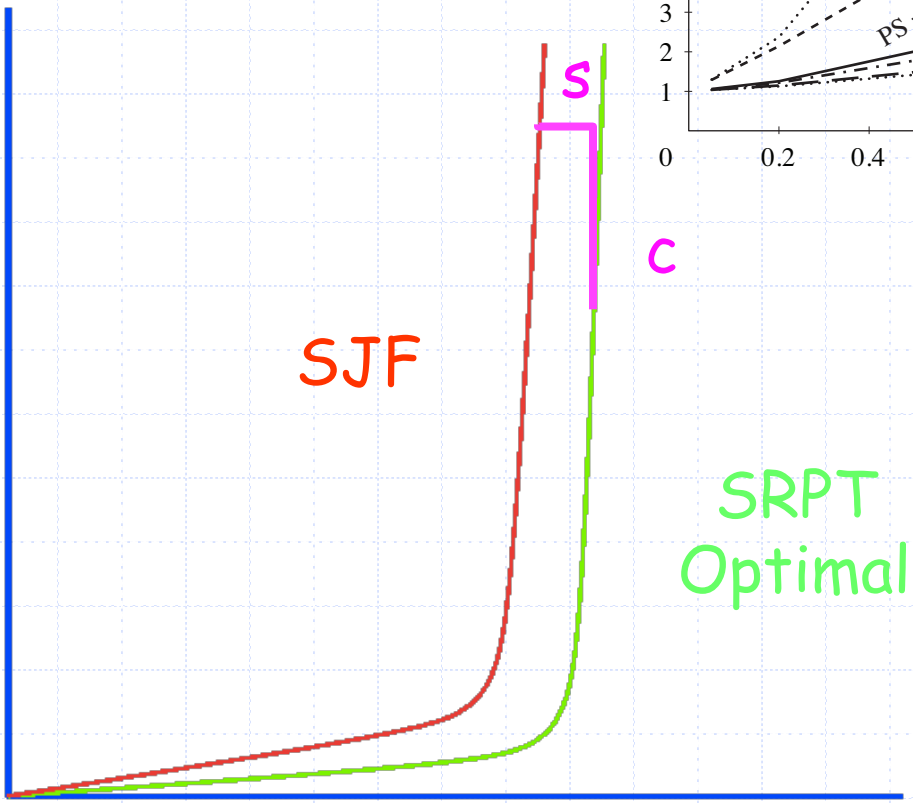
- Question: For what sort of inputs below is SJF most likely not to be O(1)-competitive?
- Answer: Load near 1
- Question: Why does this picture suggest that SJF might be scalable?



s-speed c-competitive



FCFS .....
SJF - - - -
PS = PLCFS ——
FB -·-·-
PSJF — —
SRPT .....



Average response time

Low load
Fast processor

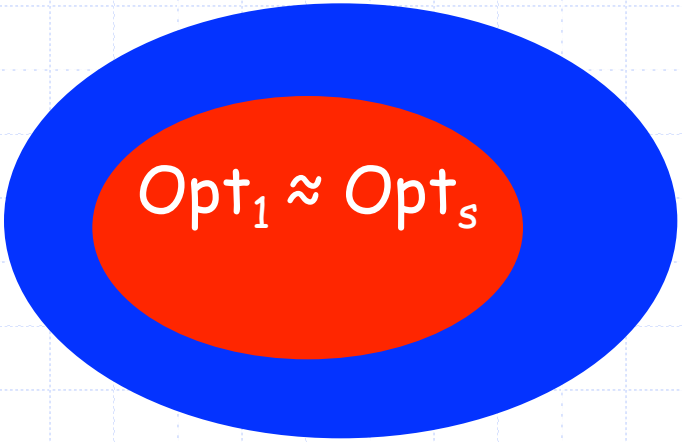
High load
Slow Processor

More Resource/Speed Augmentation Analysis Motivation

- Question: On what inputs I is an s-speed constant-competitive algorithm also (1-speed) constant-competitive?

More Resource/Speed Augmentation Analysis Motivation

- Question: On what inputs I is an s-speed constant-competitive algorithm also (1-speed) constant-competitive?
- Answer: When $Opt_1(I)$ is not too much more than $Opt_s(I)$



All Inputs

Compare to Queuing Theory

All Input distributions

Competitive Analysis in Online Scheduling using Potential Functions and Dual Fitting

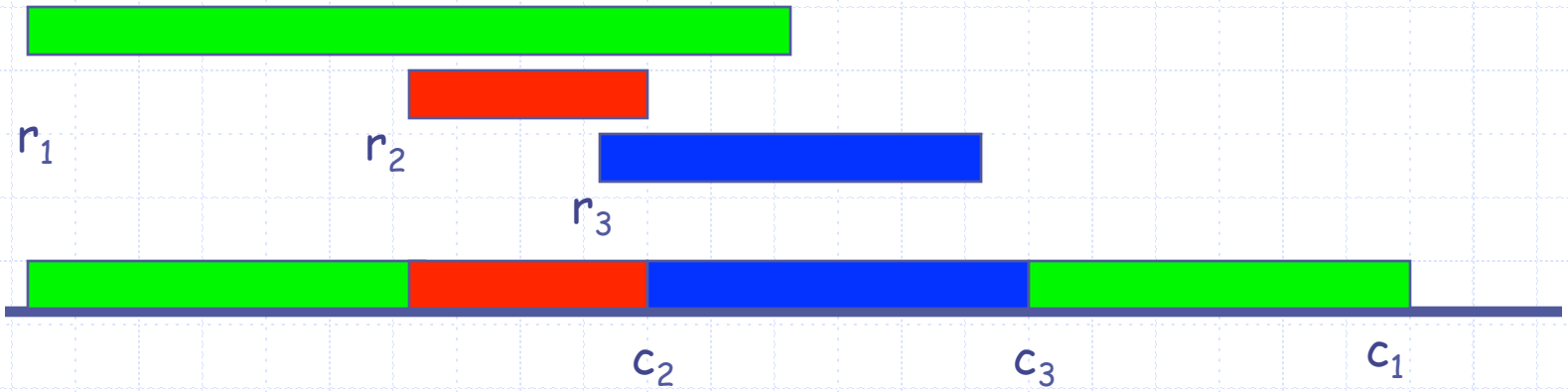
Kirk Pruhs

Scheduling Under Uncertainty Workshop

Eindhoven, June 2, 2015

Recall the Problem:

- Total flow time = $\Sigma_j f_j$
- Equivalently, average flow time = $\Sigma_j f_j / n$



Recall the Plan:

- Give an introduction to worst-case / competitive analysis via ordering of these 5 scheduling algorithms using competitive analysis

Competitive Analysis Order:
Low total flow **High total flow**

SRPT < LAS = SJF < PS < FCFS

- So far,
  - SRPT is optimal
  - Definition competitiveness, s-speed c-competitive, scalable
  - LAS and SJF are scalable, and analysis via local competitiveness via volume arguments

Outline(2)

- PS is 3-speed 2-competitive
  - Amortized local competitiveness analysis
  - Dual fitting analysis
- Negative Result: FCFS is not O(1)-speed O(1)-competitive
- Wrap-up

PS/RR Algorithm

- PS/RR Algorithm: Share the processor equally among all jobs
- Theorem: PS is not scalable. You need speed $s > 2$ to be $O(1)$-competitive
- Proof:

- For PS, when next job is released, remaining processing time of all jobs is the same
  - i^{th} job in the sequence has length $\approx 1/i^s$
- SRPT finishes jobs as they arrive

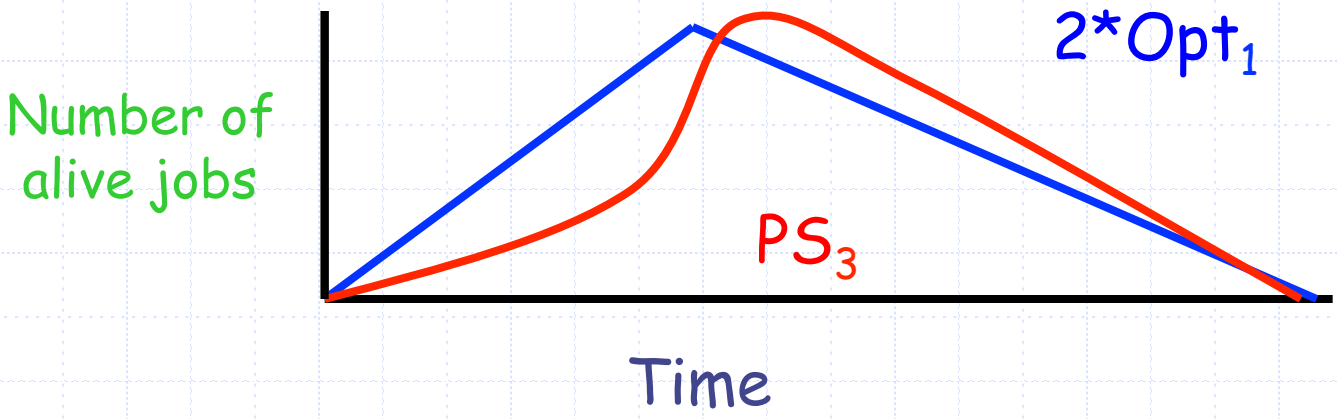
Outline(2)

- PS is 3-speed 2-competitive
  - Amortized local competitiveness analysis
  - Dual fitting analysis
- Negative Result: FCFS is not O(1)-speed O(1)-competitive
- Wrap-up

PS/RR Algorithm

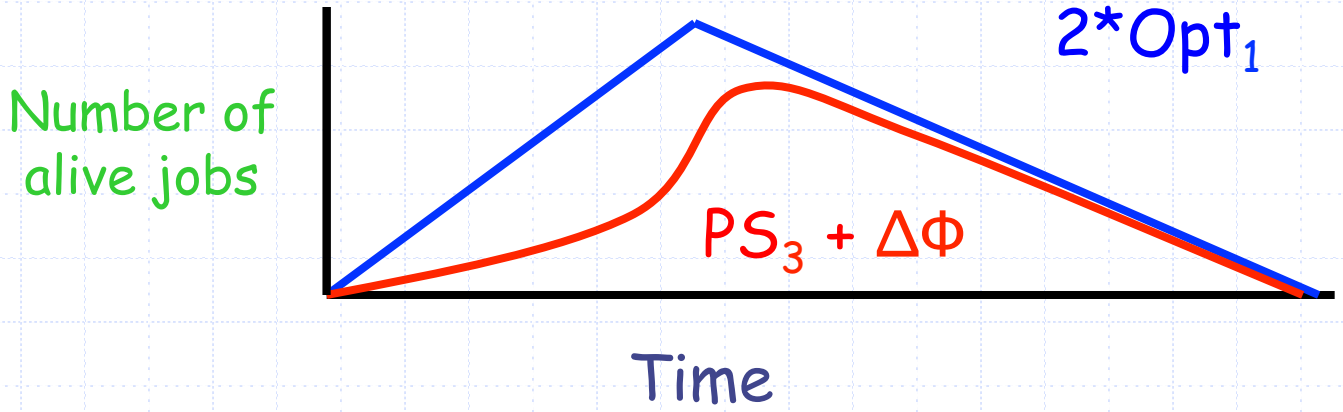
- PS/RR Algorithm: Share the processor equally among all jobs

- Theorem: PS is 3-speed 2-competitive for the objective of total flow
- Proof: We have seen local competitiveness will not work !



Amortized Local Competitiveness

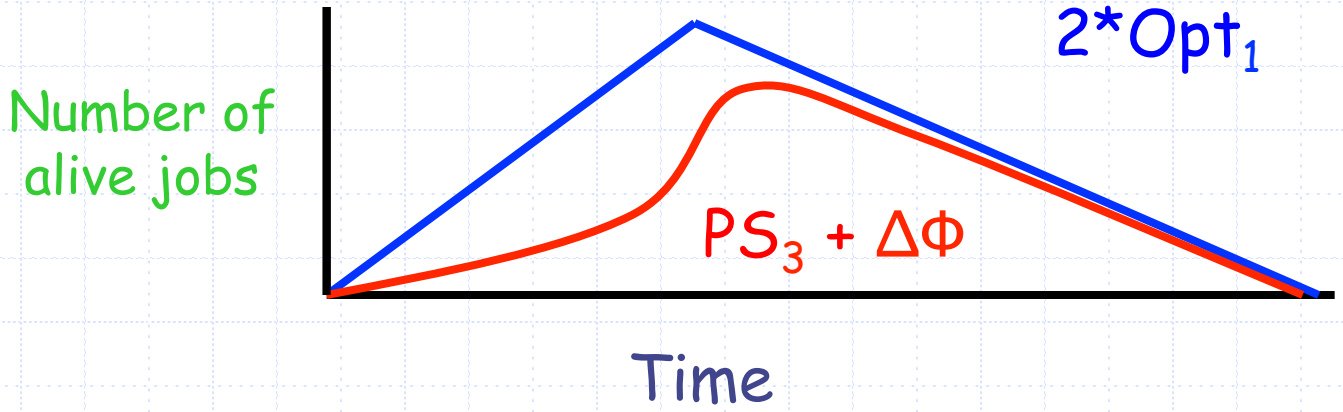
- Theorem: PS is 3-speed 2-competitive for the objective of total flow
- Proof: Amortized Local Competitiveness
- Define a potential function Φ such that



Why Amortized Local Competitiveness Arguments Work

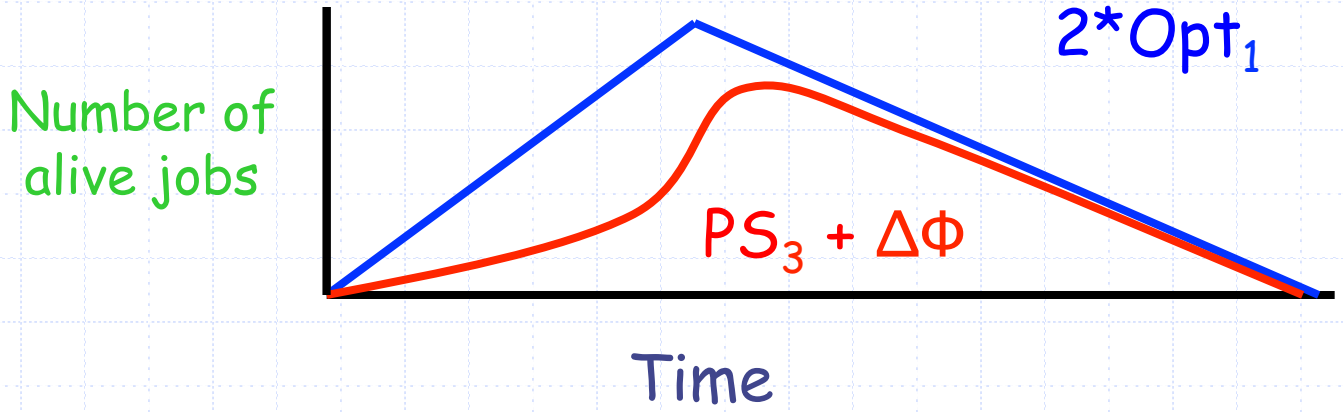
- $\Sigma_t (PS_3(t) + \Delta\Phi) =$
- $\Sigma_t (PS_3(t) + \Phi(t) - \Phi(t-1)) =$
- $PS_3 - \Phi(\text{initial}) + \Phi(\text{final}) \geq$
- PS_3

if $\Phi(\text{initial})=0$ and $\Phi(\text{final}) \geq 0$



Amortized Local Competitiveness

- Theorem: PS/RR is 3-speed 2-competitive for the objective of total flow
- Proof: Amortized Local Competitiveness
- 1. Φ initially 0,
- 2. Φ finally positive, and
- 3. For all t, $PS_3(t) + \Delta\Phi \leq 2^*Opt_1(t)$



Defining the potential function

- The potential function $\Phi(PS_3(t), Opt_1(t))$ is a function of
  - PS_3's state $PS_3(t)$ at time t
  - Opt_1's state $Opt_1(t)$ at time t

- Why should there exist a simple function such that for all possible states for PS_3 and for all possible states for Opt_1 it is the case that $PS_3(t) + \Delta\Phi \leq 2*Opt_1(t)$?

First Attempt at a Potential Function

- Question: What is lower bound on the potential if no more jobs arrive?
  - Hint: Recall key equation: $PS_3(t) + \Delta\Phi \leq 2*Opt_1(t)$

First Attempt at a Potential Function

- Question: What is lower bound on the potential if no more jobs arrive?
  - Hint: Recall key equation: $PS_3(t) + \Delta\Phi \leq 2*Opt_1(t)$
- Answer: $\Phi(t)$ = Future cost for PS_3 – 2 *future cost of Opt_1
  - Both costs under the assumption that no more jobs arrive

First Attempt at a Potential Function

- Try 1: $\Phi(t)$ = Future cost for PS_3 – 2 *future cost of Opt_1
  - Both costs under the assumption that no more jobs arrive
- Question: In what future will this potential function not work?
  - Hint: In this future the potential goes up without Opt_1 incurring any cost?

First Attempt at a Potential Function

- Try 1: $\Phi(t)$ = Future cost for PS_3 – 2 *future cost of Opt_1
  - Both costs under the assumption that no more jobs arrive
- Question: In what future will this potential function not work?
- Answer:
  - Opt_1 has few jobs alive
  - PS_3 has many, say k, jobs alive
  - A new small job of size ε arrives
  - PS_3 future cost goes up by ?
  - Opt_1 future cost goes up by ?
  - So $\Delta\Phi$ is ?

First Attempt at a Potential Function

- Try 1: $\Phi(t)$ = Future cost for PS_3 – 2 *future cost of Opt_1
  - Both costs under the assumption that no more jobs arrive
- Question: In what future will this potential function not work?
- Answer:
  - Opt_1 has few jobs alive
  - PS_3 has many, say k, jobs alive
  - A new small job of size ε arrives
  - PS_3 future cost goes up by $k\,\varepsilon/3$
  - Opt_1 future cost goes up by $O(\varepsilon)$
  - So $\Delta\Phi$ is $+ k\,\varepsilon/3$ without any justification

Now Standard Potential Function

- Standard Potential Function: Future cost for online algorithm if job sizes were how far the online algorithm is behind on these jobs

- Fact: If jobs have remaining sizes $p_1 < p_2 .. < p_k$, the future cost for PS if no more jobs arrive is $\approx \Sigma_i (k-i+1) p_i$

- Standard potential function idea applied to PS
- $\Phi = 2 \Sigma_{\text{alive jobs j for RR3}} \text{rank(j)} * \text{lag(j)}$, where
  - lag(j) is the amount that RR_3 is behind Opt_1 on job j
  - rank(j) is the number of jobs whose lag is at least lag(j)

Amortized Local Competitiveness

- Theorem: PS/RR is 3-speed 2-competitive for the objective of total flow
- Proof: Amortized Local Competitiveness
- Define $\Phi = 2 \Sigma_{\text{alive jobs j for RR3}} \text{rank}(j) * \text{lag}(j)$

- 1. Φ initially 0. Why easy?
- 2. Φ finally positive. Why easy?
- 3. For all t, $PS_3(t) + \Delta\Phi \leq 2 * Opt_1(t)$

Verifying $PS_3(t) + \Delta\Phi \leq 2^*Opt_1(t)$

- Recall $\Phi = 2 \Sigma_{\text{alive jobs j for RR3}} \text{rank(j)} * \text{lag(j)}$
- Cases:
  - New job arrives. Why $\Delta\Phi \leq 0$?
  - Rank of a job changes. Why $\Delta\Phi \leq 0$?
  - $SRPT_1$ finishes a job. Why $\Delta\Phi \leq 0$?
  - PS_3 finishes a job. Why $\Delta\Phi \leq 0$?
  - PS_3 and Opt_1 are running jobs

Verifying $PS_3(t) + \Delta\Phi \leq 2^*Opt_1(t)$

- Recall $\Phi = 2 \Sigma_{\text{alive jobs j for RR3}} \text{rank(j)} * \text{lag(j)}$
- PS_3 and Opt_1 are running jobs
- What is the worst case state for PS_3 and and Opt_1 ?

Verifying $PS_3(t) + \Delta\Phi \leq 2^*Opt_1(t)$

- Recall $\Phi = 2 \Sigma_{\text{alive jobs j for RR3}} \text{rank(j)} * \text{lag(j)}$
- PS_3 and $SRPT_1$ are running jobs
- Worst-case: PS_3 has k jobs alive and Opt_1 only has highest ranked job alive
- Change in potential due to Opt_1's running?
- Change in potential due to PS_3's running?

Verifying $PS_3(t) + \Delta\Phi \leq 2^*Opt_1(t)$

- Recall $\Phi = 2 \Sigma_{\text{alive jobs j for RR3}} \text{rank(j)} * \text{lag(j)}$
- PS_3 and $SRPT_1$ are running jobs
- Worst-case: PS_3 has k jobs alive and Opt_1 only has highest ranked job alive
- $\Delta\Phi$ due Opt_1's running is +k
- $\Delta\Phi$ due to PS_3's running is $\Sigma_{i=1}^{k}\ i\ (-3/k) \approx -3k/2$
- So evaluating $PS_3(t) + \Delta\Phi \leq 2^*Opt_1(t)$ we get:
  $k + 2\ [\ k - 3k/2] \leq\ 2^*1$

Outline(2)

- PS is 3-speed 2-competitive
  - Amortized local competitiveness analysis
  - Dual fitting analysis (Nguyễn Kim Thắng)
- Negative Result: FCFS is not O(1)-speed O(1)-competitive
- Wrap-up

Dual Fitting Technique to show PS_3 is 2-competitive

- Write a primal linear program P that is a lower bound to optimal
- Find the dual linear program D
- Find feasible setting of the dual variables with objective value = $PS_3/2$
- Appeal to weak duality to conclude that Opt $\geq$ $PS_3/2$

Constructing the Primal

- Question: What should the variables be?

Constructing the Primal

- Question: What should the variables be?
- Answer: x_{jt} = how much of job j is processed at time t

- Question: Informally, what are the constraints?

Constructing the Primal

- Question: What should the variables be?
- Answer: x_{jt} = how much of job j is processed at time t

- Question: Informally, what are the constraints?
- Answer:
  - Every job is completed
  - At most one unit of volume is processed at each time step

Constructing the Primal

- Primal P
- Min $\Sigma_{jt} (t - r_j) x_{jt} / p_j$
- $\Sigma_t x_{jt} \geq p_j$
- $\Sigma_j x_{jt} \leq 1$

- Why is the optimal solution to this primal is strictly less than the optimal objective value?

Constructing the Primal

- Primal P
- Min $\Sigma_{jt} (t - r_j) x_{jt} / p_j$
- $\Sigma_t x_{jt} \geq p_j$
- $\Sigma_j x_{jt} \leq 1$

- Why is the optimal solution to this primal is strictly less than the optimal objective value?
- Answer: The objective is fractional flow instead of integral flow

Constructing the Dual

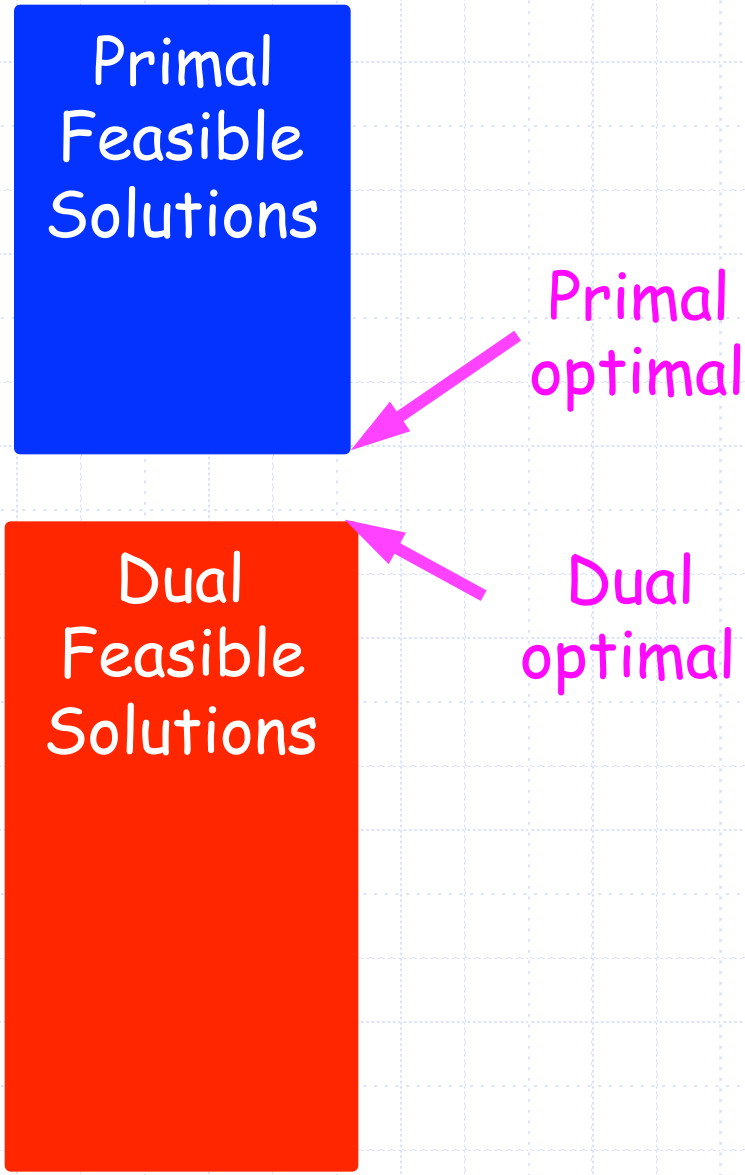
- Primal P
- Min $\Sigma_{jt} (t - r_j) x_{jt} / p_j$
- $\Sigma_t x_{jt} \geq p_j$ — dual variable α_j
- $\Sigma_j x_{jt} \leq 1$ — dual variable β_t

- Dual D
- Max $\Sigma_j p_j \alpha_j - \Sigma_t \beta_t$
- $\alpha_j - \beta_t \leq (t - r_j)/p_j$ — primal variable x_{jt}

Weak Duality

- Primal P
- Min $\Sigma_{jt} (t - r_j) x_{jt} / p_j$
- $\Sigma_t x_{jt} \geq p_j$
- $\Sigma_j x_{jt} \leq 1$

- Dual D
- Max $\Sigma_j p_j \alpha_j - \Sigma_t \beta_t$
- $\alpha_j - \beta_t \leq (t - r_j)/p_j$



Setting the Dual Variables

- Dual D
- Max $\Sigma_j p_j a_j - \Sigma_t \beta_t$
- $a_j - \beta_t \leq (t - r_j)/p_j$

primal variable x_{jt}

- a_j = 3 (response time of job j for PS_3 if no jobs were to arrive after $r_j)/p_j$
- β_t = number of jobs alive for PS_3 at time t

Showing Dual Feasiblity

- Recall
  - a_j = 3 (response time of job j for PS_3 if no jobs were to arrive after r_j)/p_j
  - β_t = number of jobs alive for PS_3 at time t
- Dual constraint rewritten: $(p_j a_j - (t - r_j))/p_j \leq \beta_t$
- Worst case: Consider $t=r_j$, and all jobs of unit remaining volume.
- When $t=r_j$ the constraint holds, and both sides decrease at the rate that t increases

Evaluating Dual Objective

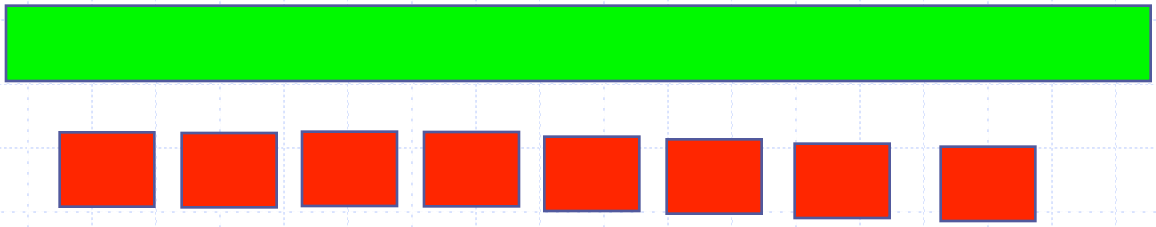
- Dual Objective: Max $\Sigma_j p_j \alpha_j - \Sigma_t \beta_t$
- Recall
  - α_j = 3 (response time of job j for PS_3 if no jobs were to arrive after rj)/pj
  - β_t = number of jobs alive for PS_3 at time t
- Obvious Fact: $\Sigma_t \beta_t = PS_3$
- Not so Obvious Fact: $\Sigma_j p_j \alpha_j = \Sigma_j$ 3 (response time of job j for PS_3 if no jobs were to arrive after r_j) = 3 PS_3 / 2
- So dual objective = $PS_3/2$

Outline(2)

- PS is 3-speed 2-competitive
  - Amortized local competitiveness analysis
  - Dual fitting analysis
- Negative Result: FCFS is not O(1)-speed O(1)-competitive
- Wrap-up

FCFS

- FCFS: Run the job that arrived the earliest
- Theorem: No nonpreemptive algorithm, including FCFS, is $O(1)$-speed $O(1)$-competitive.
- Proof:



Outline(2)

- PS is 3-speed 2-competitive
  - Amortized local competitiveness analysis
  - Dual fitting analysis
- Negative Result: FCFS is not O(1)-speed O(1)-competitive
- Wrap-up

Recall Plan for This Talk:

Competitive Analysis Order:
Low total flow **High total flow**

SRPT < LAS = SJF < PS < FCFS

- SRPT is optimal
- LAS and SJF are scalable
- PS is 3-speed 2-competitive
- Key Concepts: competitiveness, s-speed c-competitive, scalable, analysis via local competitiveness via volume arguments, analysis via amortized local competitiveness, analysis via dual fitting

Best First Read For Competitive Analysis of Scheduling Algorithms

- Sungjin Im, Benjamin Moseley, Kirk Pruhs: A tutorial on amortized local competitiveness in online scheduling. SIGACT News 42(2): 83-97 (2011)
  - Gentle introduction to everything in this talk except for dual fitting analysis

Relative Strengths?
Happy to discuss this further over a Beer

Queuing Theoretic Analysis	Competitive Analysis
+Gives absolute performance bounds instead of relative performance bounds	-Worst-case bounds are often quite large
-Value of the analysis not clear if input distribution doesn't match mathematical model	+Generally worst-case analysis is mathematically easier than average case analysis.
+Sometimes gives "interesting" results when competitive analysis gives boring negative results, e.g. FCFS.	+There are problems, particularly for for complicated settings, where it is easier to obtain "interesting" competitiveness results than it is to obtain "interesting" average case results

Neither dominates the other !!!!!!!!