Scheduling to Minimize Total Flow-Time

Stefano Leonardi
Università di Roma "La Sapienza"

Parallel Machine Scheduling

- m parallel machines;
- J : set of n jobs
- p_j: processing time of job j;
- r_j : release time of job j
- Job j must be processed for p_j units of time after release time r_j
- C_j: completion time of job j
- $P = \dfrac{\max_j p_j}{\min_j p_j}$

Several Issues

1. *On-line scheduling:* the existence of a job is only known at time of release
2. *Preemption:* the execution of a job may be interrupted and resumed later
3. *No-migration:* preempted jobs must be resumed on the same machine
4. *Non-clairvoyant scheduling*: processing time is only known at time of completion.

   Subject of the second lecture.

Measure Quality of Service

- Average Response Time or Flow Time:

$$\frac{1}{n}\sum_{j\in J} F_j = \frac{1}{n}\sum_{j\in J} C_j - r_j$$

- Measure average waiting time of the users
- Widely accepted as a good measure of the QoS provided to the users

Measure Algorithm's Performance

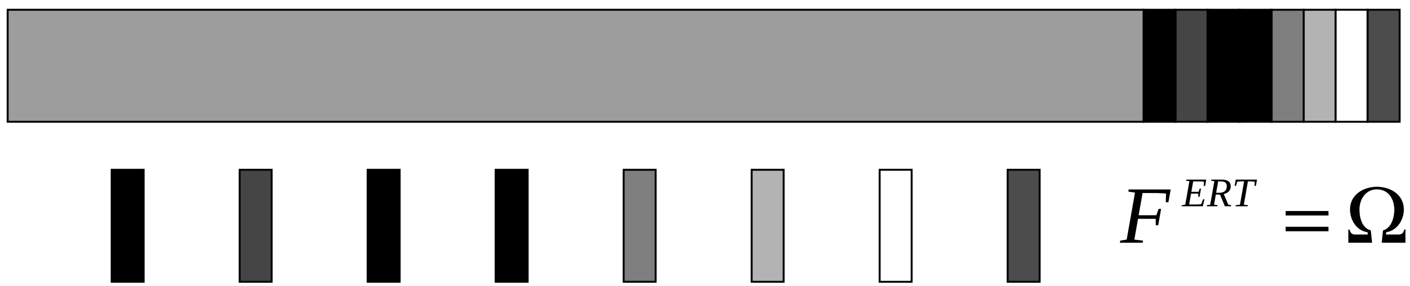
- Jobs are scheduled on-line, i.e. without knowledge of jobs released in the future.
- Algorithm *A* is *c*-competitive if for any input instance *J*:

$$A\lg(J) \leq c\,Opt(J)$$

- Opt is the optimal scheduler that knows the whole input sequence in advance
- Also interested in off-line polynomial time approximation algorithms for NP-hard versions of the problem.

Simple strategies fail

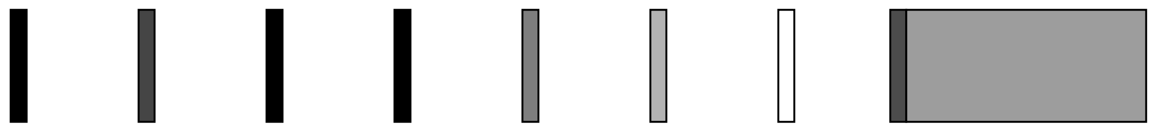
- Earliest Release Time:

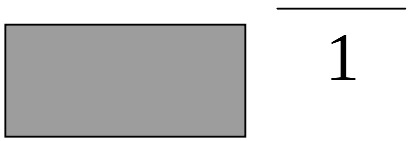


$$F^{ERT} = \Omega(nP)$$

$$F^{OPT} = O(P)$$

- Shortest Processing Time:

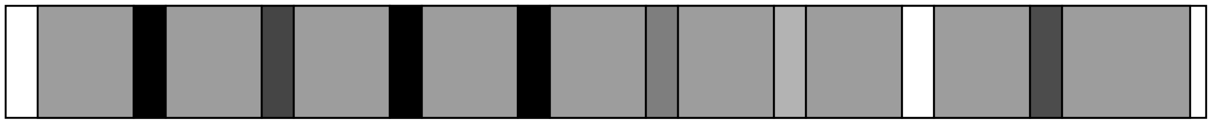




$$F^{SPT} = \Omega(n) \quad F^{OPT} = O(1)$$

Preemption Improves System Responsiveness

- Job preemption improves average flow time.
- E.g.: preempt long jobs to give precedence to short jobs.
- Preemption is not very expensive, context switching at the local processor.



$$F^{SRPT} = O(P)$$

Approximating Total Flow Time with Preemption

- Shortest Remaining Processing Time (SRPT) optimal for m=1[Baker 74]
- NP-hard for m=1 machines if preemption is not allowed [J.K. Lenstra 77]. We discuss later non-preemptive approximations.
- NP-hard for m>1 machines even if preemption is allowed [Du, Leung, Young 90]

Approximating total flow time on parallel machines with preemption

- SRPT is $\Theta\left(\log\frac{n}{m}, \log P\right)$-competitive for m machines [Leonardi, Raz, 97]
- SRPT uses migration
- No better approximation is known
- Migration of preempted jobs may be expensive. We discuss later non-migrative algorithms.

SRPT is $\Theta(\log P)$ - competitive

- $x_j(t)$: remaining processing time of job j at time t

- Job j of class k at time t if $x_j \in [2^k, 2^{k+1})$

- At most *O(log P)* classes: a job is preempted only when a shorter job is released

Analysis of SRPT

- $\delta^A(t)$: # of jobs in *A*'s schedule at time *t*
- $V^A(t)$: Volume = total remaining processing time in in *A*'s schedule at time *t*
- $\gamma^A(t)$: # of non-idle machines in *A*'s schedule at time *t*
- T : set of time instants with $\gamma^A(t) = m$

More notation

- $\Delta V^{A}(t) = V^{A}(t) - V^{OPT}(t)$

- $f_{\leq k, \geq h}(t)$: value of function f when restricted to job of class between h and k

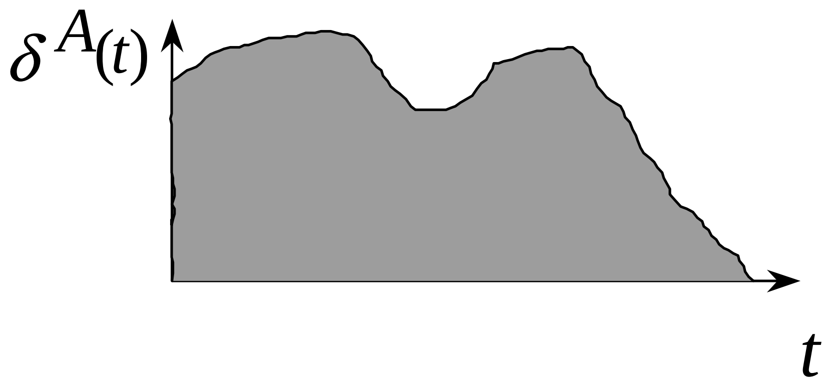
- $f = \delta, V, \Delta\delta, \Delta V$

- $\Delta V_{\leq k}(t)$: volume difference of jobs of class at most k between the algorithm and the optimum

An alternative definition of Total Flow Time

$$F^A(t) = \sum_{j \in J} C_j^A - r_j = \int_{t \geq 0} \delta^A(t)\,dt$$

$$\int_{t \geq 0} \gamma^A(t)\,dt = \sum_{j \in J} p_j \leq F^{OPT}$$



SRPT is *O(log P)*-comp. I

- $$F^A = \int_{t \geq 0} \delta^A(t)dt = \int_{t \notin T} \delta^A(t)dt + \int_{t \in T} \delta^A(t)dt$$

$$\leq \int_{t \notin T} \gamma^A(t)dt + \sum_k \int_{t \in T} \delta^A_{=k}(t)dt$$

- If $\forall t \in T,\ \delta^A_{=k}(t) \leq 2m + 2\delta^{OPT}(t)$ then

$$F^A \leq \int_{t \notin T} \gamma^A(t)dt + 2\sum_k \int_{t \in T} m\,dt + 2\sum_k \int_{t \in T} \delta^{OPT}_{=k}(t)dt$$

$$\leq O(\log P)\int_{t \geq 0} \gamma^A(t)dt + O(\log P)\int_{t \in T} \delta^{OPT}(t)dt$$

$$= O(\log P)F^{OPT}$$

SRPT is *O(log P)*-comp. II

- t_k : last time $\leq t \in T$ when jobs of class >k have been processed

- *Lemma*: $\Delta V_{\leq k}(t) \leq m2^{k+1}$

Proof. Since *SRPT* has processed jobs of class $\leq k$ between t_k and t, $\Delta V_{\leq k}(t) \leq \Delta V_{\leq k}(t_k)$

At time t_k, at most *m* jobs of class $\leq k$ in *SRPT's* schedule: $\Delta V_{\leq k}(t_k) \leq V^A_{\leq k}(t_k) \leq m2^{k+1}$

SRPT is *O(log P)*-comp. III

- Lemma: $\forall t \in T,\ \ \delta^{A}_{=k}(t) \le 2m + 2\delta^{OPT}(t)$

Proof:

$$\delta^{A}_{=k}(t) \le \frac{V^{A}_{=k}}{2^k} = \frac{\Delta V_{=k} + V^{OPT}_{=k}}{2^k} = \frac{\Delta V_{\le k} - \Delta V_{\le k-1}}{2^k} + \frac{V^{OPT}_{=k}}{2^k}$$

$$\le \frac{m2^{k+1}}{2^k} + \frac{V^{OPT}{}_{\le k-1}}{2^k} + \frac{V^{OPT}_{=k}}{2^k} \le 2m + \frac{V^{OPT}{}_{\le k}}{2^k}$$

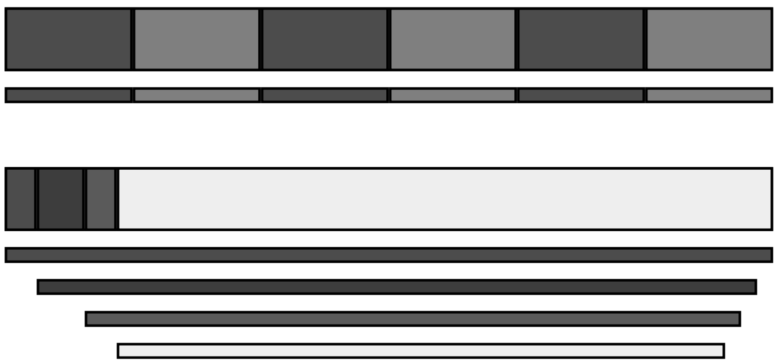
$$\le 2m + 2\delta^{OPT}_{\le k}$$

Approximating Total Flow Time without Migration

- A preemptive algorithm for parallel machine scheduling without migration [Awerbuch, Azar, Leonardi, Regev,99]
- AALR is $O(\log n, \log P)$- competitive
- Non- migrative algorithms are still very effective for Flow Time optimization

Why the ‘‘most” straightforward approach fails

- Assign jobs to machines using the SPT rule among the jobs never processed
- Process a job up to completion on the machine of assignment
- Some machines get overloaded. Ex:



The Algorithm for Preemptive Scheduling Without Migration

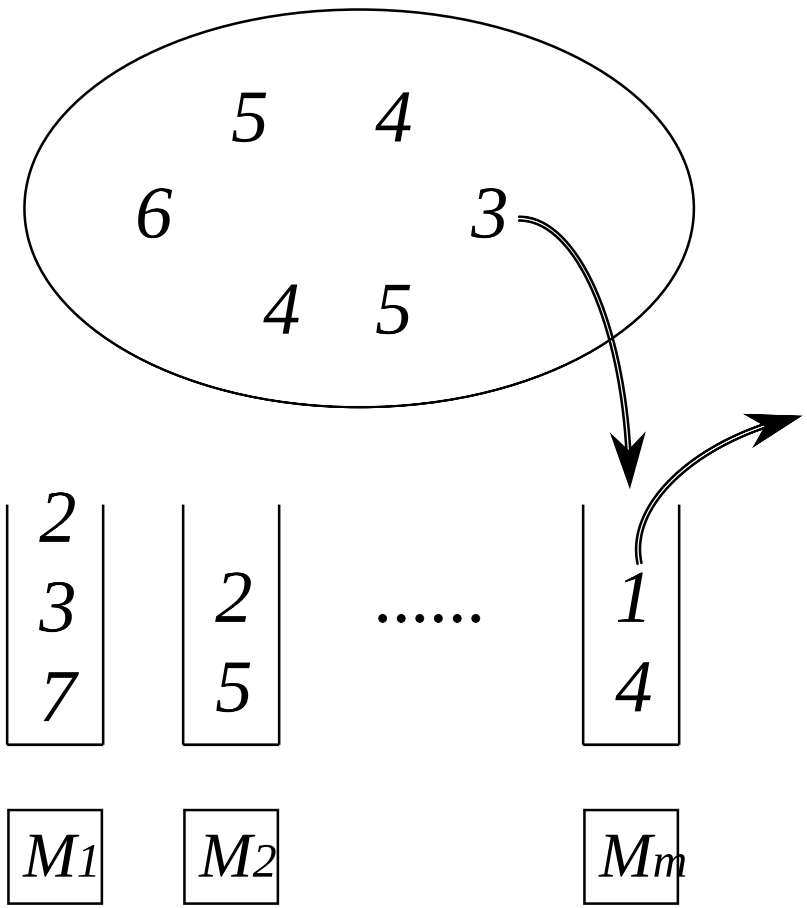
- Jobs never processed are in a ***Pool***
- Jobs assigned to a machine are in a ***Stack***
- Job j of class k at time t if remaining processing time at time t is $x_j(t) \in [2^k, 2^{k+1})$
- Jobs are inserted in the pool at release time

Observe: Class of a job changes over time

Assign Jobs to Machines

- On machine *i* schedule the job at the top of machine *i*'s stack
- If the stack of a machine is empty, push the job of lowest class in the pool
- If a job in the pool has class lower than a job at the top of a stack, then push this job into that stack.
- Pop a job from the stack when completed

A picture of the algorithm



AALR is *O(logP)*-competitive

- $\delta^{A,S}(t)$: total # of jobs in the *m stacks* at time *t*
- $\delta^{A,P}(t)$: # of jobs in the *pool* at time *t*

$$F^{A}(t) = \int_{t\geq 0} \delta^{A,S}(t)dt + \int_{t\geq 0} \delta^{A,P}(t)dt$$

- Prove the bound separately for jobs in the stacks and in the pool.
- For jobs in the pool: $\int_{t\geq 0} \delta^{A,P}(t)dt = O(\log P)\, F^{OPT}$ proved similarly to SRPT

Jobs in the Machine's Stacks:

$$\int_{t\geq 0} \delta^{Alg,S}(t)\,dt = O(\log P)\, F^{OPT}$$

- Jobs in a stack are in a strictly decreasing class order
- At most O(log P) jobs in a stack at any time t

$$\int_{t\geq 0} \delta^{Alg,S}(t)\,dt \leq \int_{t\geq 0} \gamma(t) \log P\,dt =$$

$$\log P \int_{t\geq 0} \gamma(t)\,dt = \log P \sum_{j\in J} p_j \leq$$

$$\log P\, F^{OPT}$$

Non-preemptive Algorithms

- $\Omega(n)$ -competitive lower bound
- NP-hard for *m*=1 [J.K. Lenstra 77]
- $\Theta(\sqrt{n})$ approximation for *m=1*
  [Kellerer,Tauthenam, Woeginger, 96]
- $\tilde{\tilde{O}}\left(\sqrt{n/m}\right)$ approximation for *m* machines
  [Leonardi, Raz, 97]
- $\Omega(n^{1/3})$ approximation lower bound for *m* machines if $P \neq NP$ [Leonardi, Raz, 97]

Non-preemptive approximation

Theorem 1 [LR97] From preemptive c approximation to a $cO\left(\sqrt{n/m}\right)$ non-preemptive approximation

Corollary 2 There exists an $O\left(\sqrt{n}\right)$ approximation for a single machine.

Corollary 3 There exists an $O\left(\sqrt{n/m}\log n/m\right)$ approximation for parallel machines

How to remove preemptions

- Notation for preemptive solution:
  - S_j^p : starting time
  - $C_j^p \geq S_j^p + p_j$: completion time
  - $F^p = \sum_{j \in J} F_j^p = \sum_{j \in J} C_j^p - r_j$: total flow time
- Non-preemptive solution:

$$C_j = S_j + p_j$$

- Remove preemption in big jobs expensive

The Algorithm for m=1

Big and Small jobs:

- Big jobs: $B = \left\{ j : F_j^p \geq \frac{F^p}{\sqrt{n}} \right\}$
- Small jobs $S = \left\{ j : F_j^p < \frac{F^p}{\sqrt{n}} \right\}$
- At most $\sqrt{n}$ big jobs

Algorithm for Small jobs

- Schedule small jobs in order of S_j^p
- $S_j \leq S_j^p + \frac{F^p}{\sqrt{n}}$ since

$$S_j \leq \max_{j':S_{j'}^p \leq S_j^p} \left\{ S_{j'}^p + F_{j'}^p \right\} \leq S_j^p + \frac{F^p}{\sqrt{n}}$$

- $|F - F^p|_S \leq n\left(\frac{F^p}{\sqrt{n}}\right) = \sqrt{n} F^p$

Algorithm for Big jobs

- For a big job j, partition time $t \geq r_j$ into intervals with p_j units of idle time.
- Between the first $\sqrt{n}$ such intervals there exists an interval I with at most $\sqrt{n}$ jobs.
- Schedule job j at the beginning of interval I
- Shift ahead the at most $\sqrt{n}$ jobs scheduled in I

Big jobs: Analysis.

1. Big job *j* delayed by p_j idle time units for each interval before I

$$| F - F^{p} |_{B}^{'} \leq \sqrt{n} \sum_{j \in B} p_j$$

2. Big job j delayed by at most $\sum_j p_j$

$$| F - F^{p} |_{B}^{''} \leq \sqrt{n} \sum_{j \in B} p_j$$

3. At most $\sqrt{n}$ jobs in interval I delayed by p_j

.

$$| F - F^{p} |_{B}^{'''} \leq \sqrt{n} \sum_{j \in B} p_j$$

Weighted Flow Time

$$\min \sum_{j\in J} w_j F_j^p = \sum_{j\in J} w_j (C_j^p - r_j)$$

- $O(\log^2 P)$ - competitive algorithm for m=1.
- Ω (min $\left\{ \sqrt{W}, \sqrt{P}, \left(\frac{w_j n}{m}\right)^{1/4} \right\}$ - rand. comp. lower bound for m>1

  [Chekuri, Khanna, Zhu, 2001]
- O(k) – comp for k weight classes

  [Bansal, Dhamdere, 2003]

Measure Quality of Service *and* System's Load

- Average Stretch:

$$\frac{1}{n}\sum_{j\in J} S_j = \frac{1}{n}\sum_{j\in J}\frac{C_j - r_j}{p_j}$$

- Measure of the load of the system
- *It is not affected from long jobs*
- Measure user's service degradation with respect to an unloaded system

Results for minimizing average stretch

- SRPT is O(1) competitive for m=1 and m>1. [Muthukrishnan, Rajaraman, Shaheen, Gehrke, 1999]

- AALR is O(1) on m>1 competitive without job migration [Becchetti, Leonardi, Muthukrishnan, 2000]

Open Problems

- Two major open problems:
  - O(1) approximation for preemptive flow time on parallel machines
  - O(1) approximation for weighted flow time on a single machine