Non-Clairvoyant Scheduling to Minimize Total Flow Time

Stefano Leonardi
Università di Roma “La Sapienza’’

CPU Scheduling in Time Sharing Operating Systems

- Jobs are released over time
- The time a job will be executed is unknown until its completion
- Goal: Provide fast answer to applications
- Job preemption improves responsiveness: e.g. preempt long jobs to execute short jobs
- Context switching has a reasonable cost

Parallel Machine Scheduling

- m *identical* 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 r_j
- C_j: completion time of job j
- $$P = \frac{\max_j p_j}{\min_j p_j}$$

Measure Total Flow Time

- Average Flow Time:

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

- Average time spent in the system between release and completion
- Widely accepted as a good measure of the QoS provided to jobs

Non-clairvoyant scheduling

Very little knowledge about the input instance

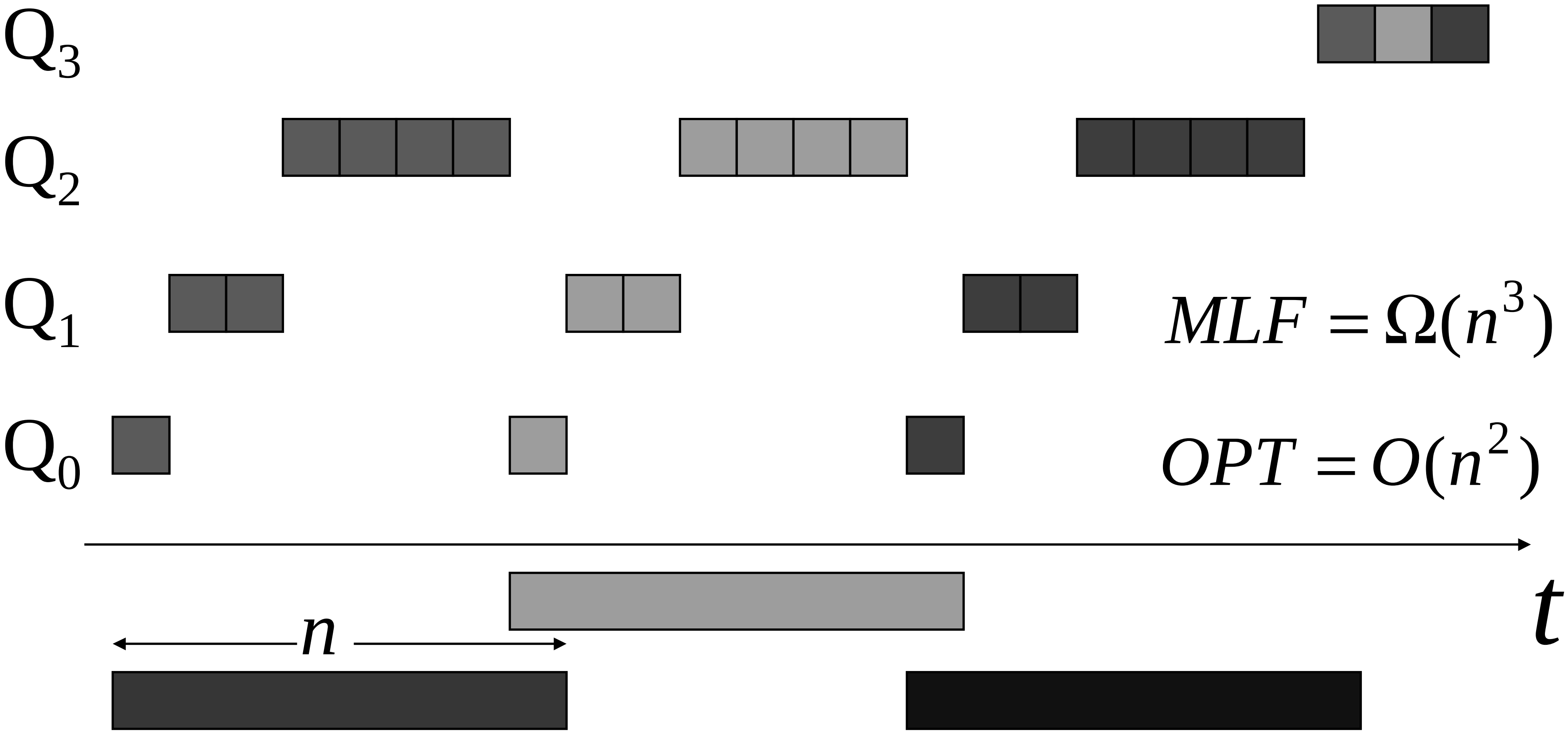
1. The existence of a job is known at the release time of the job

2. The processing time of a job is only known at its completion

Multi-level Feedback (MLF) Algorithm

- At the basis of CPU scheduling in Unix and Windows NT
  1. Jobs assigned to queue Q_0 when released
  2. Process a job for 2^i time units in queue Q_i before to promote it to queue Q_{i+1}
  3. Schedule those m jobs in the lowest queues, giving priority to jobs at the front

MLF fails ….in Theory☹



$$MLF = \Omega(n^3)$$

$$OPT = O(n^2)$$

Shortest Remaining Processing Time (SRPT)

- SRPT is a good rule of thumb for minimizing the average flow time
- Preempt a job on execution if a job with shorter remaining processing time is released
- SRPT is optimal on a single machine [Baker 74]
- Best known $O(\log P)$, $O(\log n/m)$ approximation for parallel machines [Leonardi, Raz, 97]

Why MLF fails?

- It cannot stick to SRPT since it does not known the processing time of a job
- Preempt a job with short remaining processing time in a high queue to process a long job in a lower queue
- Is it enough to follow SRPT in an approximate way?

  E.g., *r.p.t. is a large fraction of the initial processing time for a large share of the jobs*

Previous Results on Non-Clairvoyant Scheduling

- $\Omega(n^{1/3})$ deterministic lower bound [Motwani, Phillips, Torng, 95]
- $\Omega(\log n)$ randomized lower bound on a single machine against the oblivious adversary [MPT95]
- $\Omega(P)$ randomized lower bound with $n = 2^P$ jobs [Kalyanasundaram, Pruhs, 97]

Two Kinds of Analysis

1. Worst-case Competitive Analysis of Randomized Multi-level Feedback
2. *Smoothed Competitive Analysis* of Multi-level Feedback:

   A mixture of worst-case and average-case analysis introduced in [Spielman, Teng, 2001]

   "The Symplex Algorithm converges in expected polynomial time if the input instance is perturbed with a normal distribution!"

1. Worst-case Competitive Analysis of the Randomized Multi-level Feedback Algorithm

Becchetti, Leonardi, 2001

Measure Algorithm's Performance

- Competitive Analysis of On-line Algorithms
- Randomized Algorithm *A* is *c*-competitive against the *oblivious adversary* if for any input instance *J*:

$$E_\sigma[Alg_\sigma(J)] \leq c\, Opt(J)$$

where the input instance is generated by the adversary without knowledge of the random choices of the algorithm

Randomized Multi-Level Feedback (RMLF) Algorithm

[Kalyanasundaram, Pruhs, 97]

- Approximately behave like SRPT: jobs enter the queue in which they are completed with a large share of the initial processing time
- The time a job is processed in a queue is a random variable
- RMLF is $O(\log n \log \log n)$ - competitive on a *single machine* against the stronger on-line adaptive adversary[KP 97]

Our Results for RMLF$^+$

- RMLF$^+$ is Θ (log n) competitive for a single machine against the oblivious adversary [Becchetti, Leonardi, 00]
- RMLF$^+$ is $O(\log n \log n/m)$, $O(\log n \log P)$, competitive for m parallel machines against the oblivious adversary [BL00]
- First theoretical validation of the goodness of MLF in practice ☺

Clairvoyant Preemptive Results

- Shortest Remaining Processing Time First (SRPT) optimal for m=1 [Baker 74]
- SRPT is $O(\min\{\log n/m, \log P\})$-competitive for m machines [LR 97]
- $\Omega(\log n/\mathrm{m}), \Omega(\log P)$ randomized lower bounds extend to the non-clairvoyant case for parallel machines [LR97]

The RMLF$^+$ Algorithm

Randomized Multi-level Feedback ($RMLF^+$)

- Organize jobs in a set of *Priority Queues* Q_0, Q_1,….
- Order jobs in each queue by Earliest Release Time First
- Process those *m* jobs in the lowest queues, in each queue give priority to jobs released earliest

RMLF$^+$

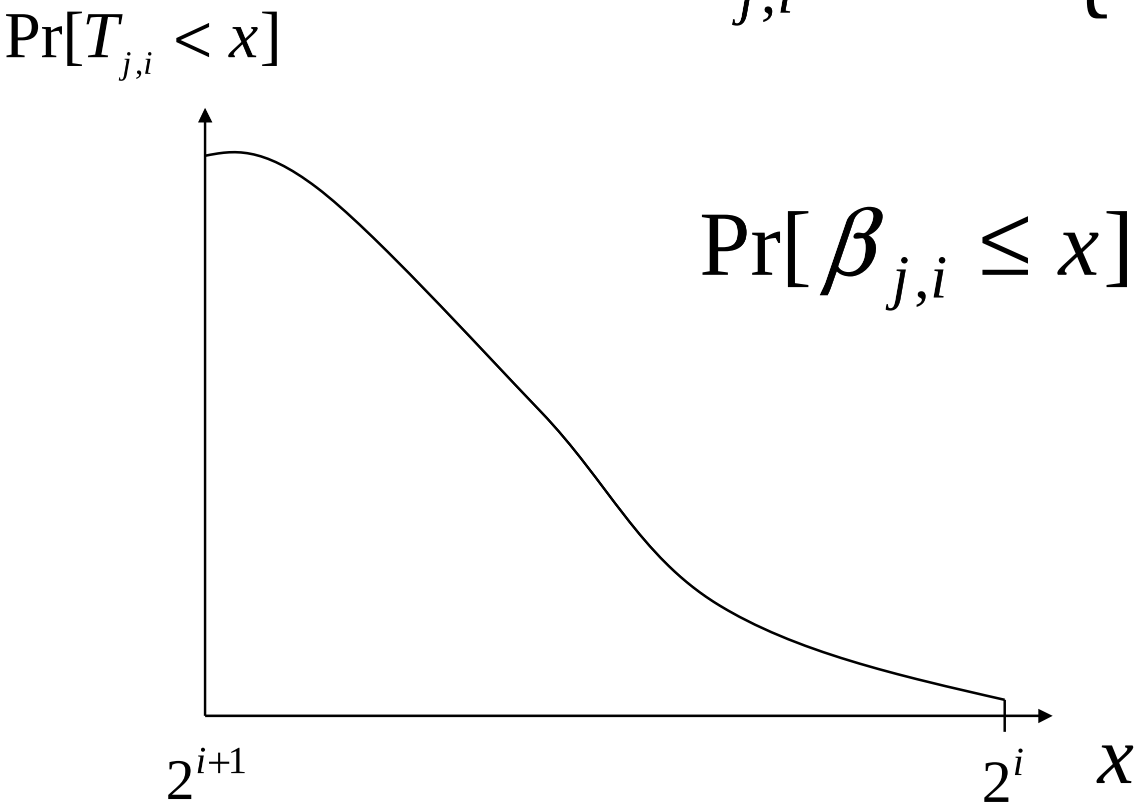
- $T_{j,i}$: Target of job j in queue Q_i
- Job j enters queue Q_0 with target $T_{j,0}$ when released

- Job j is completed in queue Q_i if $p_j \leq T_{j,i}$
- Job j is promoted to queue Q_{i+1} with target $T_{j,i+1}$ if $T_{j,i} < p_j$

RMLF$^+$

$$T_{j,i} = \max\left\{2^i, 2^{i+1} - \beta_{j,i}\right\}$$

$$\Pr[\beta_{j,i} \le x] = 1 - e^{-\gamma \frac{x}{2^i} \ln j}$$



The Analysis of RMLF^{+}

RMLF$^+$

- Job j of class i if $p_j \in [2^i, 2^{i+1})$
- Job j of class i completed in queue Q_i or Q_{i+1}:

$$T_{j,i} \in [2^i, 2^{i+1}) \text{ and } T_{j,i} \geq 2^{i+1}$$

- At most m jobs processed but not completed in every queue:

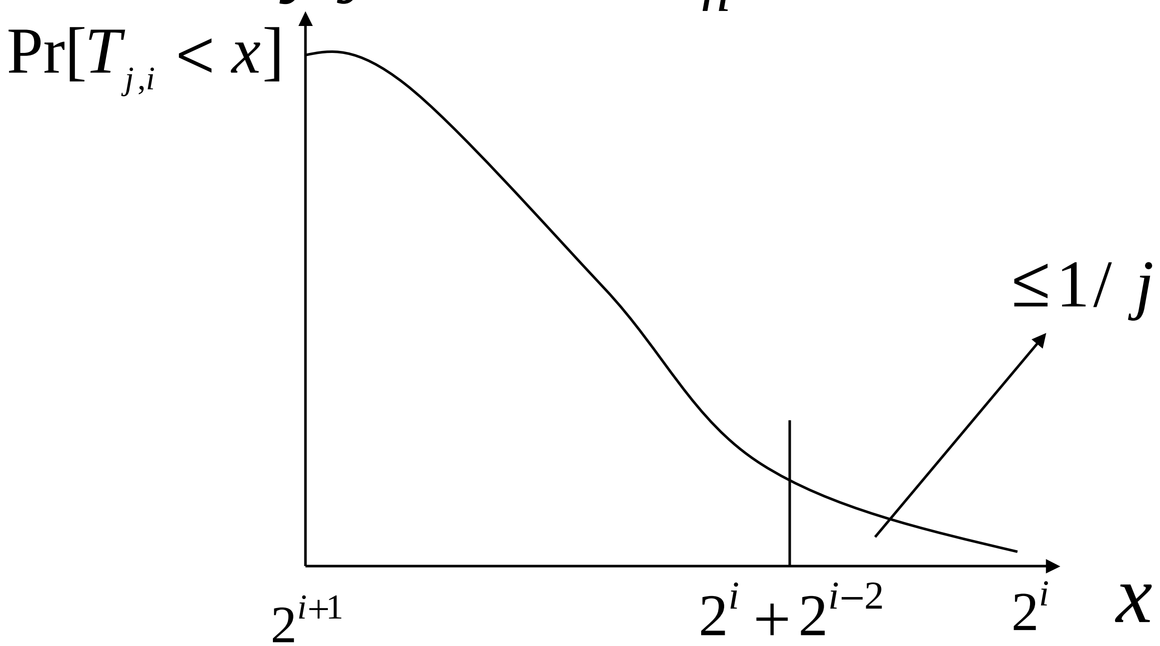
  If a job was processed, there was a time in which also all jobs with higher priority were processed.
- At most log P queues

Unlucky Jobs

- *Most jobs must have a large share of the initial processing time when they enter the queue in which they are completed.*
- A job *j* is *unlucky* if $p_j \le 2^i + 2^{i+2}$ and it ends in queue Q_{i+1}
- Otherwise a job is *lucky*.

Why the exponential distribution?

- A job is unlucky with $\Pr[j\ unlucky] \leq 1/j$
- $E[unlucky\ jobs] \leq H_n$

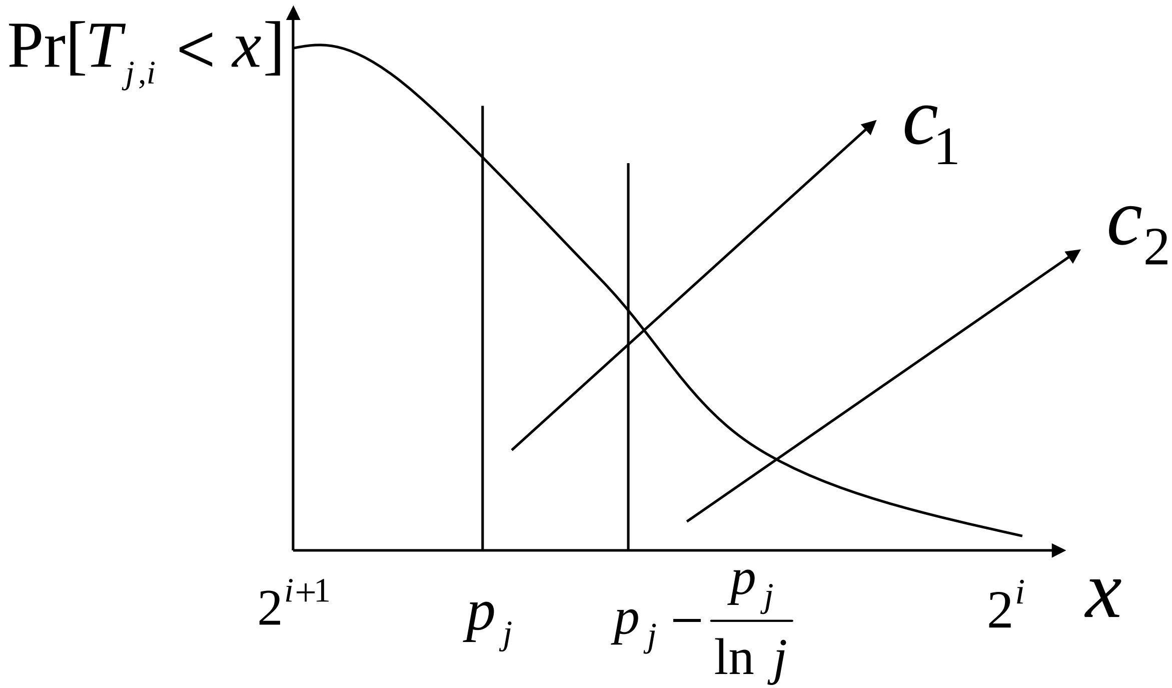


Big Jobs

- A lucky job is *big* at some time *t* if it has remaining processing time $\geq \frac{p_j}{\ln j}$
- This is always true if job *j* in queue Q_k at time *t*, $k \leq i - 1$
- It is true with constant probability also if job *j* in Q_i or Q_{i+1} at time *t*

Why the exponential distribution? II

- A lucky job alive at any time t is big with constant probability.



Why not the uniform distribution?

- Release at time 0 n jobs of size $2^i + 2\sqrt{n}$ with $n = 2^i$
- Pr[job j ends in queue Q_{i+1} with r.p.t $\approx \sqrt{n}$] $\approx 1/\sqrt{n}$
- At time $n^2 + 2n\sqrt{n} - cn$, $O(\sqrt{n})$ jobs are not completed w.h.p.
- Then release n^3 jobs of size 1
- $RMLF^u = \Omega(n^3\sqrt{n})$; OPT $= O(n^3)$

O(log n) competitiveness for *m=1*

O(log n log n/m) competitiveness for any *m*

- Outcome of a unified analysis of $RMLF^+$
- The number of jobs that are released is exponential in the size of the alive jobs difference between $RMLF^+$ and the optimum
- For parallel machines, an additional overhead is due to the idle time inserted on some machines.

Smoothed Competitive Analysis of the MLF Algorithm

Becchetti, Leonardi, Marchetti-Spaccamela, Schaefer, Vredeveld, 2002

Open Problems

- Non-clairvoyant algorithm to minimize average stretch: $\frac{1}{n}\sum_j F_j / p_j$
- A tight non-clairvoyant algorithm on *m* parallel machines
- Apply smoothed competitive analysis to other practical scheduling algorithms sucessfull in practice