Scheduling to Minimize Average Stretch Online

J.E. Gehrke, S. Muthukrishnan, Rajmohan Rajaraman and Anthony Shaheen

FOCS 99

May 1, 2000

Why Stretch?

- Slowdown (stretch) of Job

$$S(J) = \frac{ResponseTime}{Size}$$

- Fairness, expect to wait less if small.

- Each job weighted equally.

- Problem: Minimize the Mean Slowdown.
  Equivalently, Minimize the Total Slowdown.

Getting started

- There is no optimal online algorithm.
  Scenario with just three jobs.

- Offline Complexity?
  Unknown for single processor $m = 1$,
  NP-Hard for $m \geq 2$

- All jobs arrive at time 0.

- How about shortest job first?

- Should have taken remaining times into account.
  What would be a greedy strategy?

Greedy Strategy

Is it good in the online case?
No one seems to have analysed it.
Except that it is optimal if there are only 2 job sizes.

We are able to show that it has C.R. 2 for the single processor case.
Also have a worst case example where it achieves that ratio.

No good!! Since SRPT also has C.R. 2 too, in addition to being optimal for response time.

May be able to show better under restricted settings.
Had a hard time coming up with the C.R. 2 example.

SRPT again!!

Main Result: SRPT is 2-competitive with respect to average slowdown for single processors and 14-competitive for multi-processors.

Quite surprising, since does bad even with just two jobs.

Even more surprising for multi-processor case, since SRPT was optimal for response time for single processor case,
and had $\Omega(\max \log n/m, \log P)$ C.R. for multi-processors.

Notation:

Given a job j.

$p(j)$: Size (processing time).

$r(j)$: Remaining processing time.

$S(j)$: Slowdown.

At any time t, Q_t will be the set of jobs in the system (processor + queue).

$$S(j) = \sum_{t: j \in Q_t} \frac{1}{p(j)}$$

$$\begin{aligned} \text{Total Slowdown } S &= \sum_{jobs} \sum_{t: j \in Q_t} \frac{1}{p(j)} \\ &= \sum_{t} \sum_{j \in Q_t} \frac{1}{p(j)} \end{aligned}$$

Key idea for single processor case:

Usually big jobs around. So small contribution to slowdown at each time step.

Proof

Split the total slowdown into two parts S_1 and S_2 , contribution due to active and waiting jobs in SRPT.

$$S_1 = \sum_t \frac{1}{p(q_{t,1})}$$

$$S_2 = \sum_t \sum_{j \in U_t} \frac{1}{p_j}$$

where $q_{t,1}$ is the job in service at time t and U_t is set of waiting jobs at time t.

$$S_1 = |J|$$

$S = S_2 + |J|$

Let $\tilde{S}$ be the slowdown of the optimal algorithm.

Showing $S_2 \leq \tilde{S}$ implies a C.R. 2,

Since $S \leq \tilde{S} + |J|$

and $|J| \leq \tilde{S} \Rightarrow S \leq 2\tilde{S}$

More Notation: Let α_t be the sum of reciprocals of sizes at time t in SRPT queue. β_t be the sum of reciprocals of remaining processing times at time t.

Similarly, $\tilde{\alpha}_t$ and $\tilde{\beta}_t$ be the corresponding quantities for the execution of an arbitrary algorithm.

$S = \sum_t \alpha_t$
$S_2 \leq \tilde{S}$ can be written $\alpha_t - \frac{1}{r_t(q_{t,1})} \leq \tilde{\alpha}_t$

To Show $S_2 \leq \tilde{S}$
$(\alpha_t - \frac{1}{r_t(q_{t,1})} \leq \tilde{\alpha}_t)$

Sufficient to show

$$\beta_t - \frac{1}{r_t(q_{t,1})} \leq \tilde{\alpha}_t$$

Show a weaker result first:

$$\beta_t - \frac{1}{r_t(q_{t,1})} \leq \tilde{\beta}_t$$

Greedy Lemma: $r_t(q_{t,k}) \geq \tilde{r}_t(\tilde{q}_{t,k-1})$

How to show that infact smaller than the sum of reciprocals of sizes ??

Greedy Lemma: $r_t(q_{t,k}) \geq \tilde{r}_t(\tilde{q}_{t,k-1})$

Proof: Let $V_t(i)$ be the sum of the remaining processing times of all jobs in SRPT system with remaining processing time at most i.

Claim: For any time t and any i, $V_t(i) \leq \tilde{V}_t(i) + i$.

Proof: Induction on time.

What happens if jobs are released.

What happens if $V_{t-1}(i)$ is

1) Nonempty

2) Empty

Greedy Lemma follows . Why?

For Multi-processors

Recall the approach, mapped each job in SRPT (except the first) to a job with larger remaining time in the other queue.

This approach fails for multiprocessors.
We saw an instance where there are an arbitrary large number of jobs in SRPT but 2-3 in the optimal.

How to proceed in this case?
Analysis quite involved.
1) Partial mapping.
2) Split unmapped into 3 types, independent, marked dependent and unmarked dependent.
3) Amortize the contribution of the slowdown for each type of unmapped job.

Finally show $S \leq 3\tilde{S} + 11|J|$

Instead we consider what happens to the worst case example for response time.

For response time, if we started with a job of size k, there were $\log k$ jobs left at the end. These were delayed by x, by jobs of size 1.

Mean response time

$$\approx \frac{x + x \log k + f(k)}{x + g(k)}$$

However Mean slowdown

$$\approx \frac{x + x/k + 2x/k + 4x/k + ... + kx/k + u(k)}{x + v(k)}$$

$$\approx \frac{2x + u(k)}{x + v(k)}$$