Online Scheduling on Identical Machines using SRPT

Kyle Fox and Benjamin Moseley

University of Illinois at Urbana-Champaign
Client-Server Scheduling
Client-Server Scheduling
Client-Server Scheduling
Client-Server Scheduling
Client-Server Scheduling
Single Machine Scheduling

- $n$ jobs arrive over time
- Job $i$ arrives at time $r_i$
- Job $i$ has processing time $p_i$
- Preemption is allowed
Single Machine Scheduling

- $n$ jobs arrive over time
- Job $i$ arrives at time $r_i$
- Job $i$ has processing time $p_i$
- Preemption is allowed
Single Machine Scheduling

- $n$ jobs arrive over time
- Job $i$ arrives at time $r_i$
- Job $i$ has processing time $p_i$
- Preemption is allowed
Single Machine Scheduling

- $n$ jobs arrive over time
- Job $i$ arrives at time $r_i$
- Job $i$ has processing time $p_i$
- Preemption is allowed
Multiple Identical Machines

- Given $m$ machines of same speed
- Must chose $m$ jobs to schedule at any time
Minimizing Total Flow Time

- $C_i$: completion time of job $i$
- Flow time: $C_i - r_i$
- Total flow time: $\sum_{i \in [n]} (C_i - r_i)$
- Most popular metric in scheduling theory
- A measure of total social welfare
Competitive Analysis

• We want *online* algorithms

• No optimal online algorithm for multiple machines

• Want to find small competitive ratio $c$ such that $A_{\text{online}}(I) \leq c \cdot \text{OPT}_{\text{offline}}(I)$ for all input $I$

• A *worst case* analysis framework
Speed Augmentation

- Strong lower bound \( \Omega(\min\{\log P, \log n/m\}) \) on competitive ratio for multiple machines [Leonardi, Raz ’07]

  - \( P \) is ratio of max. processing time to min.

- Give algorithm \( s \) speed and compare to unit speed OPT

- \( s \)-speed \( c \)-competitive

- Algorithm \textit{scalable} if \((1 + \epsilon)\)-speed \( f(\epsilon) \)-competitive
Shortest-Remaining-Processing-Time

• Conceptually simple and used in practice

• Optimal algorithm for one machine

• $O(\min\{\log P, \log n/m\})$ competitive for multiple machines [Leonardi, Raz ’07]
  
  • Asymptotically best possible competitive ratio
With Extra Speed

• $(2 - \frac{1}{m})$-speed 1-competitive [Phillips, et al. ’02]

• $s$-speed $\frac{1}{s}$-competitive when $s \geq 2 - \frac{1}{m}$ [Torng, McCullough ’08]

• Was unknown if SRPT is scalable ($(1 + \epsilon)$-speed $f(\epsilon)$-competitive) for a decade (ahem)

  • First scalable algorithm found by Chekuri et al. [’04]

  • Geometrically group jobs into classes and spread work in each class evenly between machines
Main Theorem

• **Theorem:** SRPT is \((1 + \epsilon)\)-speed \(\frac{4}{\epsilon}\)-competitive for total flow time

• Found by Buseemma and Torng ['06]

• Proof compares an **augmented** SRPT to a unit speed OPT

• Some older proofs show algorithm is *locally competitive*

• We use a potential function
Accumulated Flow Time

- $C_i^S$ and $C_i^O$: SRPT and OPT’s completion times for job $i$

- $SRPT(i, t) = \min \left\{ C_i^S, t \right\} - r_i$

- $SRPT(t) = \sum_{i \in [n]: t \geq r_i} SRPT(i, t)$

- OPT functions defined similarly
Potential Function Analysis

• Difficult to compare growth of SRPT with OPT directly

• Find some $\Phi : [0, \infty) \rightarrow \mathbb{R}$ such that $\Phi(0) = \Phi(\infty) = 0$

• Upper bound total increase on $SRPT + \Phi$ to upper bound $SRPT$
Changes to $SRPT + \Phi$

- Continuous changes (running condition)
- Job arrivals
- Job completions
Finding the Potential Function

- $Q^S(t)$: alive jobs that SRPT needs to schedule at time $t$

- Use one term $\Phi(i, t)$ for each alive job

- $\Phi(t) = \sum_{i \in Q^S(t)} \Phi(i, t)$

\[
\Phi(i, t) = \frac{1}{m \epsilon} \left( \right)
\]
Counteracting Increasing Flow Time

- $\Phi(i, t)$ must decrease continuously to counteract increasing flow time of job $i$

- $p^S_i(t)$: remaining processing time of job $i$ for SRPT

- $R^S(i, t)$: remaining processing time for all released jobs completed by SRPT before job $i$ $(\sum_{j : r_j \leq t, c^S_j \leq c^S_i} p^S_j(t))$

- Either $\frac{d}{dt} mp^S_i(t) \leq -m(1 + \epsilon)$ or $\frac{d}{dt} R^S(i, t) \leq -m(1 + \epsilon)$

$$\Phi(i, t) = \frac{1}{m\epsilon}$$
Counteracting Increasing Flow Time

1. $\Phi(i, t)$ must decrease continuously to counteract increasing flow time of job $i$

2. $p^S_i(t)$: remaining processing time of job $i$ for SRPT

3. $R^S(i, t)$: remaining processing time for all released jobs completed by SRPT before job $i$ ($\sum_{j: r_j \leq t, c^S_j \leq c^S_i} p^S_j(t)$)

4. Either $\frac{d}{dt} mp^S_i(t) \leq -m(1 + \epsilon)$ or $\frac{d}{dt} R^S(i, t) \leq -m(1 + \epsilon)$

\[
\Phi(i, t) = \frac{1}{m\epsilon} \left( R^S(i, t) + mp^S_i(t) \right)
\]
Counteracting Job Arrivals

- Need to limit weight of $\Phi(i, t)$ on arrival

- $V^O(i, t)$: similar to $R^S(i, t)$ but measures OPT’s remaining processing times

- Lemma: $R^S(i, t) - V^O(i, t) \leq mp_i$ [Muthukrishnan, et al. ’04; Pruhs, et al. ’04]

\[
\Phi(i, t) = \frac{1}{m\epsilon} \left( R^S(i, t) + mp^S_i(t) \right)
\]
Counteracting Job Arrivals

- Need to limit weight of \( \Phi(i, t) \) on arrival

- \( V^O(i, t) \): similar to \( R^S(i, t) \) but measures OPT’s remaining processing times
  \( (\sum_{j: r_j \leq t, C^S_j \leq C^M_i, p_j \leq p_i} p^O_j(t)) \)

- Lemma: \( R^S(i, t) - V^O(i, t) \leq mp_i \) [Muthukrishnan, et al. ’04; Pruhs, et al. ’04]

\[
\Phi(i, t) = \frac{1}{m\epsilon} \left( R^S(i, t) + mp^S_i(t) - V^O(i, t) \right)
\]
Counteracting Job Arrivals

• Upon job arrival:

\[ \Phi(i, t) = \frac{1}{m\epsilon} \left( R^S(i, t) + mp_i - V^O(i, t) \right) \]

\[ \leq \frac{1}{m\epsilon} (2mp_i) \]

\[ = \frac{2}{\epsilon} p_i \leq \frac{2}{\epsilon} \text{OPT}(i, \infty) \]

• So job arrivals contribute at most \( \frac{2}{\epsilon} \text{OPT} \) to SRPT + \( \Phi \)

\[ \Phi(i, t) = \frac{1}{m\epsilon} \left( R^S(i, t) + mp_i^S(t) - V^O(i, t) \right) \]
Counteracting Job Arrivals

• \( \frac{d}{dt} V^O(i, t) \geq -m \)

• \( \frac{d}{dt} \text{SRPT}(t) + \frac{d}{dt} \Phi(t) = \sum_{i \in Q^S(t)} \left[ \frac{d}{dt} \text{SRPT}(i, t) + \frac{d}{dt} \Phi(i, t) \right] \)

\[ \leq \sum_{i \in Q^S(t)} \left[ 1 + \frac{1}{m \epsilon} (-m(1 + \epsilon) + m) \right] \]

\[ = 0 \]

• So the running condition does not contribute to \( \text{SRPT} + \Phi \)

\[ \Phi(i, t) = \frac{1}{m \epsilon} \left( R^S(i, t) + mp^S_i(t) - V^O(i, t) \right) \]
Dealing With Job Completions

- Leave the potential function alone
- Use a subtle charging argument
- Jobs contributing to $V^O(i, t)$ were alive for a while while SRPT scheduled job $i$

$$
\Phi(i, t) = \frac{1}{m^e} \left( R^S(i, t) + mp^S_i(t) - V^O(i, t) \right)
$$
Dealing With Job Completions

• Need to charge \( \frac{1}{m\epsilon} V^O(i, t) \)

• For any job \( j \) contributing to \( V^O(i, t) \), SRPT did \( p_j \) units of work on \( i \) while \( j \) was alive in OPT’s queue

\[
(\frac{r_i}{1+\epsilon}, \frac{r_j}{1+\epsilon}) = (\frac{C_i^S}{1+\epsilon})
\]

• Why? \( p_j \leq p_i^S(r_j) \) since SRPT completes jobs in \( V^O(i, t) \) before job \( i \)

• Work done over \( \frac{p_j}{1+\epsilon} \) time units so charge \( j \) at a rate of \( \frac{1+\epsilon}{m\epsilon} \) during these times

\[
\Phi(i, t) = \frac{1}{m\epsilon} \left( R^S(i, t) + mp_i^S(t) - V^O(i, t) \right)
\]
Dealing With Job Completions

• SRPT schedules at most $m$ jobs at a time

• Job $j$ receives charge at a rate of $m \cdot \frac{1+\epsilon}{m\epsilon} = \frac{1+\epsilon}{\epsilon}$ at times when it is alive in OPT’s queue

• So job completions contribute at most $\frac{1+\epsilon}{\epsilon}OPT$ to $SRPT + \Phi$

• So $SRPT(\infty) + \Phi(\infty) \leq \frac{3+\epsilon}{\epsilon}OPT$

$$\Phi(i, t) = \frac{1}{m\epsilon} \left( R^S(i, t) + mp_i^S(t) - V^O(i, t) \right)$$
Also: Seeking Fairness

- Total flow time metric not ‘fair’ to all jobs
- Minimize $l_k$ norms of flow time: $\left( \sum_{i \in [n]} (C_i - r_i)^k \right)^{1/k}$ [Bansal, Pruhs ’04]
- **Theorem:** SRPT is $(1 + \varepsilon)$-speed $\frac{4}{\varepsilon^2}$-competitive for $l_k$ norm of flow time
- ([Buseema, Torng ’06])
Thank you!