Performance Analysis of Modified SRPT in Multiple-Processor Multitask Scheduling

Wenxin Li
The Ohio State University
li.7328@osu.edu

ABSTRACT
In this paper we study the multiple-processor multitask scheduling problem in both deterministic and stochastic models, where each job has several tasks and is complete only when all its tasks are finished. We consider and analyze Modified Shortest Remaining Processing Time (M-SRPT) scheduling algorithm, a simple modification of SRPT, which always schedules jobs according to SRPT whenever possible, while satisfying the non-preemptive constraint. The M-SRPT algorithm is proved to achieve a competitive ratio of $\Theta(\log \alpha + \beta)$ for minimizing response time, where $\alpha$ denotes the ratio between maximum job workload and minimum job workload, $\beta$ represents the ratio between maximum non-preemptive task workload and minimum job workload. In addition, the competitive ratio achieved is shown to be optimal (up to a constant factor), when there are constant number of machines. We further consider the problem under Poisson arrival and general workload distribution (i.e., M/GI/N system), and show that M-SRPT achieves asymptotic optimal mean response time when the traffic intensity $\rho$ approaches 1, if job size distribution has finite support. Beyond finite job workload, the asymptotic optimality of M-SRPT also holds for infinite job size distributions with certain probabilistic assumptions, for example, M/M/1 system with finite task workload.

1. INTRODUCTION
With widespread applications in various manufacturing industries, scheduling jobs to minimize the total flow time (also known as response time, sojourn time and delay) is a fundamental problem in operations research that has been extensively studied. As an important metric measuring the quality of a scheduler, flow time, is formally defined as the difference between job completion time and releasing date, and characterizes the amount of time that the job spends in the system [1].

Optimizing the flow time of single-task jobs has been considered both in offline and online scenarios. However, jobs with multiple tasks are more common and relevant in practice, which can take many different forms in modern computing environments. For example, for the objective of computing the matrix-vector product, we can divide matrix elements and vector elements into groups of columns and rows respectively, then the tasks correspond to the block-wise multiplication operations. With the tremendous increasing in data size and job complexity, we cannot emphasize too much the importance of designing scheduling algorithms for jobs with multiple tasks. Though much progresses have been made in single-task job scheduling, there is a lack of theoretical understanding regarding multiple-processor multitask scheduling (MPMS), where a job is considered to be completed only when all the tasks within the job are finished. A natural question that arises is, how to design an efficient scheduling algorithm to minimize the total amount time that the multitask jobs spend in the system.

2. MODEL AND PRELIMINARIES
We are given a set $J = \{J_1, J_2, \ldots, J_n\}$ of $n$ jobs arriving online over time, together with a set of $N$ identical machines. Job $i$ consists of $n_i$ tasks and its workload $p_i$ is equal to the total summation of the processing time of tasks, i.e., $p_i = \sum_{t \in N} p_i(t)$, where $p_i(t)$ represents the processing time of the $t$-th task of job $i$. Tasks can be either preemptive or non-preemptive. All the information of job $i$ is unknown to the algorithm until its releasing date $r_i$. Under any given scheduling algorithm, the completion time of job $j$ under the algorithm, denoted by $C_j$, is equal to the maximum completion time of individual tasks within the job. The flow time of job $j$ is defined as $F_j = C_j - r_j$, our objective is to minimize the total flow time $\sum_{j \in J} F_j$.

Throughout the paper we use $\alpha = \max_{i \in [n]} p_i / \min_{i \in [n]} p_i$ to denote the ratio of the maximum to the minimum job workload. Let $\eta$ be the maximum processing time of a non-preemptive task, $\beta = \eta / \min_{i \in [n]} p_i$ be the ratio between $\eta$ and minimum job workload. In some sense, parameters $\beta$ and $\eta$ represent the degree of non-preemptivity and exhibit a trade-off between the preemptive and non-preemptive setting. More specifically, the problem approaches the preemptive case when $\eta$ is small, and degenerates to the non-preemptive case if all the jobs consist of a single non-preemptive task, in which $\eta$ reaches the maximum value of $\max_{i \in [n]} p_i$.

In the stochastic setting, we assume that jobs arrive into the system according to a Poisson process with rate $\lambda$. Job processing times are i.i.d distributed with probability density function $f(\cdot)$.

3. MODIFIED SRPT ALGORITHM AND COMPETITIVE RATIO ANALYSIS
The details of the Modified SRPT algorithm are specified in Algorithm 1. At each time slot $t$, jobs with non-preemptive task are kept processing on the machines, while the remaining machines are used to process jobs with the
smallest remaining workload. The main idea of Algorithm 1 is similar to SRPT, i.e., we utilize as many resources as possible on the job with the smallest remaining workload, to reduce the number of alive jobs in a greedy manner, while satisfying the non-preemptive constraint.

Algorithm 1: Modified SRPT (M-SRPT)

1. At time $t$, maintain the following quantities:
   - For each job $i \in [n]$, maintain $W_i(t)$ // remaining workload
   - $w_i(t)$ // remaining workload of the shortest single task being processed (if exists) or alive
   - $J_1(t)$ ← jobs which have tasks that are finished at time $t$
   - $J_2(t)$ ← jobs which have preemptive task that is being processed at time $t$

and assign alive jobs to the $|J_1(t)\cup J_2(t)|$ machines, where jobs with smaller value of $W_i(t)$ have a higher priority.

Theorem 1. Algorithm 1 achieves a competitive ratio that is no more than $C_{\text{RM-SRPT}} \leq 4 \log \alpha + 2\beta + 8$.

We in addition have the following lower bound.

Proposition 2. For multiple-processor multitask scheduling problem with constant number of machines, there exists no algorithm that achieves a competitive ratio of $o(\log \alpha + \beta)$.

The main ingredient of the proof is the following lemma. To show the competitive ratio upper bound, we divide the jobs into different classes and compare the remaining number of jobs under Algorithm 1 with that under optimal algorithm $\pi^\ast$. At time slot $t$, we divide the unfinished jobs into $\Theta(\log \alpha)$ classes $\{C_k(\pi, t)\}_{k \in [\log \alpha + 1]}$, based on their remaining workload under Algorithm 1. Jobs with remaining workload that is no more than $2^k$ and larger than $2^k - 1$ are assigned to the $k$-th class. Formally, $C_k(\pi, t) = \{i \in [n] | W_{i, \text{M-SRPT}}(t) \in (2^k - 1, 2^k]\}$, where $W_{i, \text{M-SRPT}}(t)$ represents the unfinished workload of job $i$ at time $t$ under Algorithm 1. We use $C_k^{(2)}(t) = \cup_{l=1}^{k} C_l(t)$ to denote the collection of jobs in the first $k$ classes, and for any algorithm $\pi$, let $W_{\pi}^{(2)}(t)$ represent the total remaining workload of jobs in the first $k$ classes.

Lemma 3. For all $\epsilon > 0$, the unfinished workload under Algorithm 1 can be upper bounded as

$$W_{\text{M-SRPT}}^{(2)}(t) \leq W_{\pi}^{(2)}(t) + N \cdot (2^{k+1} + \eta).$$

Proof. Without loss of generality we can assume that $W_{\pi}^{(2)}(t) > W_{\pi}^{(2)}(t)$, otherwise Lemma 3 already holds.

Since the remaining workload under M-SRPT is strictly larger than that under the optimal algorithm, we claim that there must exist time slots in $[0, t]$, at which either idle machines exist under M-SRPT, or jobs with remaining workload (under M-SRPT) larger than $2^k$ are processed. Let $t_k^{(1)}$ be the largest time slot at which at least one machine is idle under Algorithm 1, and $t_k^{(2)}$ be the largest time slot at which there exists $i > k$ such that at least one machine is processing jobs in $C_i$ under Algorithm 1. Here we only discuss the case when $t_k^{(1)} < t_k^{(2)}$ and the whole proof is deferred to [2].

According to the definition of $t_k^{(2)}$, there exist jobs with remaining workload larger than $2^k$ being processed at $t_k^{(2)}$. Let $r \in [N]$ be the number of tasks that are being processed at time $t_k^{(2)}$ and belongs to $[n] \setminus C_k^{(2)}(t_k^{(2)})$, and $t_s \leq t_k^{(2)}$ be the latest starting processing time of these tasks. We divide our analysis into the following two cases:

- Case 1: No jobs switch from set $[n] \setminus C_k^{(2)}(t_s)$ to $C_k^{(2)}(t_k^{(2)})$ under Algorithm 1. We use $\Delta_s$ to represent the increment of $W_{\pi}^{k}$, incurred by the newly arriving jobs during time period $[t_s, t_k^{(2)}]$. Then we have $W_{\pi}^{k}(t_k^{(2)}) - W_{\pi}^{k}(t_s) = (N - r)(t_k^{(2)} - t_s) + \Delta_s$. On the other hand, $W_{\pi}^{k}$ the remaining workload of jobs in class $C_k^{(2)}$ under the optimal algorithm $\pi^\ast$, decreases at a speed that is no more than $N$ units of workload per time slot, hence $W_{\pi}^{k}(t_k^{(2)}) - W_{\pi}^{k}(t_s) \geq -N \cdot (t_k^{(2)} - t_s) + \Delta_s$.

According to the definition of $t_k^{(2)}$, no jobs with remaining workload larger than $2^k$ are processed in $(t_k^{(2)}, t]$.

Combining with analysis above, we can obtain

$$W_{\pi}^{k}(t_k^{(2)}) - W_{\pi}^{k}(t_s) \leq W_{\pi}^{k}(t_k^{(2)}) - W_{\pi}^{k}(t_s) + r \cdot 2^k.$$

Case 2: There exist jobs switching from set $[n] \setminus C_k^{(2)}(t_s)$ to $C_k^{(2)}(t_k^{(2)})$ under Algorithm 1. We use $\mathcal{J}_s$ to denote the collection of such switching jobs, we argue that $|\mathcal{J}_s| \leq N - r$, and we can derive the following conclusion:

$$W_{\pi}^{k}(t_k^{(2)}) - W_{\pi}^{k}(t_s) \leq W_{\pi}^{k}(t_k^{(2)}) - W_{\pi}^{k}(t_s) + r \cdot 2^k + N \cdot (t_k^{(2)} - t_s).$$

The proof is complete. $\square$

4. ASYMPTOTIC OPTIMALITY OF MODIFIED SRPT WITH POISSON ARRIVAL

In this section we show that under mild probabilistic assumptions, Algorithm 1 is asymptotically optimal for minimizing the total response time in the heavy traffic regime. The result is formally stated as following. It can be seen that the optimality result in [3] corresponds to a special case of our Theorem 4.
Theorem 4. Let $F_{\rho}^{M\text{-SRPT}}$ and $F_{\rho}^{1\text{-SRPT}}$ be the response time incurred by Algorithm 1 and optimal algorithm respectively, when the traffic intensity is equal to $\rho$. In an $M/GI/N$ with finite job size distribution, Algorithm 1 is heavy traffic optimal, i.e., $\lim_{\rho \to 1} \mathbb{E}[F_{\rho}^{M\text{-SRPT}}] = 1$.

For any algorithm $\pi$, we use $W_{\pi}^{x}(t)$ to denote the total workload of jobs with remaining workload more than $x$ at time $t$. We first remark that Lemma 3 can be extended to any non-negative number $y \geq 0$.

Lemma 5. Let $SRPT_{x,t}$ be the SRPT algorithm in a system with $k$ servers, where each server has a speed of $t$, then

$$W_{\pi}^{x}(t) - W_{SRPT}^{x,N}(t) \leq N \cdot (2y + \eta), \forall y, t \geq 0.$$ 

The main conclusion used to prove the optimality condition is stated as following.

Theorem 6. 

$$\mathbb{E}[F_{\rho}^{M\text{-SRPT}}] \leq \mathbb{E}[F_{\rho}^{SRPT\cdot1,N}] + O(\log \frac{1}{1 - \rho}).$$

Proof. Consider a tagged job with workload $x$, arriving time $r_x$ and completion time $C_x$. The computing resources of $N$ servers must be spent on the following types of job during $[r_x, C_x]$:

1. The system may be processing jobs with remaining workload larger than $x$, or some machines are idle, while the tagged job is in service, because the number of jobs alive is smaller than $N$. We use $W_{\text{waste}}(r_x)$ to represent the amount of such resources, then $W_{\text{waste}}(r_x) \leq (N - 1) \cdot x$.

2. The system may be dealing with jobs that have a remaining workload larger than $x$ at time $t = r_x$, while the tagged job is not in service. This is possible and happens only if the system is processing non-preemptive tasks, which belong to a job with total remaining workload larger than $x$. The tasks are in service before the arrival of the tagged job, and the non-preemptive rule allows the task to be served from time $r_x$ onwards.

Let $W_{\text{non-pm}}(r_x)$ denote the total units of computing resources spent on this class of jobs during $[r_x, C_x]$. Our main argument for this class of jobs is, $W_{\text{non-pm}}(r_x) \leq (N^2 + N) \cdot \eta + N \cdot x$. To see the correctness, we consider time intervals $[r_x, r_x + \eta]$ and $(r_x + \eta, C_x]$ separately.

- Note that there are $N \cdot \eta$ computing resources during time $[r_x, r_x + \eta]$ in total, hence it is obvious to see that the amount of resources spent on this collection of jobs during $[r_x, r_x + \eta]$ cannot exceed $N \cdot \eta$.
- We next show that in time interval $(r_x + \eta, C_x]$, the total amount of computing resources spent on such jobs is no more than $N^2 \cdot \eta + N \cdot x$. Consider the following two types of jobs:

  - Note that jobs of this class that have a remaining workload larger than $x$ at time $t = r_x + \eta$ will be processed after time $t = r_x + \eta$ only if the tagged job is in service, hence the amount of resources spending on such jobs are already taken into account in the first class above, i.e., the quantity $W_{\text{waste}}(r_x)$, and we can ignore this subclass.
  
  - For the collection of jobs with remaining workload no more than $x$ at time $t = r_x + \eta$, we first consider the setting when different tasks within the same job can be processed in parallel. It is clear to see that the remaining workload of such jobs at time $t = r_x$ must be no more than $x + N \cdot \eta$. Since there are at most $N$ such jobs in total, we can conclude that the remaining workload of jobs in this subclass must be no more than $N \cdot (x + N \cdot \eta) = N \cdot x + N^2 \cdot \eta$.

3. Tagged job itself. The amount of resources is equal to $x$, the size of the tagged job.

4. Newly arriving jobs during $[r_x, C_x]$ with size no more than $x$.

Hence $T_{\pi}^{M\text{-SRPT}}$, the response time of the tagged job, is no more than the length of a busy period of a single server system with speed $N$, which starts at time $r_x$ and has an initial workload of $W_{\text{waste}}(r_x) + W_{\text{non-pm}}(r_x) + W_{\text{SRPT}}^{x,N}(r_x) + x$. Combining with the aforementioned analysis, formally we have

$$T_{\pi}^{M\text{-SRPT}} \leq \mathbb{E}[\rho(x)](W_{\text{waste}}(r_x)) + \mathbb{E}[\rho(x)](W_{\text{SRPT}}^{x,N}(r_x) + x).$$

$$\leq \mathbb{E}[\rho(x)](3N^2 \cdot \eta + x) + \mathbb{E}[\rho(x)](W_{\text{SRPT}}^{x,N}(r_x)).$$

Note that the average response time under SRPT in a single server system is lower bounded as

$$\mathbb{E}[F_{\rho}^{SRPT\cdot1,N}] \geq \mathbb{E}[\rho(x)](W_{\text{SRPT}}^{x,N}(r_x)) = \mathbb{E}[\rho(x)] = O(\log \frac{1}{1 - \rho}).$$

The proof is complete. \Box

Beyond Job Size Distribution with Finite Support. For the most elementary model of $M/M/N$, i.e., when the job service times are exponentially distributed, we have the following theorem, which only requires one additional assumption on task workload.

Theorem 7. The average response time under Algorithm 1 is asymptotically optimal in $M/M/N$, if task workload has finite support.

5. REFERENCES

[1] Mor Harchol-Balter. Performance modeling and design of computer systems: queueing theory in action. Cambridge University Press, 2013.

[2] Wenxin Li and Ness Shroff. Performance analysis of modified srpt in multiple-processor multitask scheduling. arXiv:2006.06632, 2020.

[3] Isaac Grosso, Ziv Scully, and Mor Harchol-Balter. Srpt for multi-server systems. Performance Evaluation, 127:154–175, 2018.