Performance Analysis of Modified SRPT in Multiple-Processor Multitask Scheduling

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

Ness Shroff
Department of ECE and CSE
The Ohio State University
shroff.11@osu.edu

June 22, 2022

Abstract

In this paper we study the multiple-processor multitask scheduling problem in both deterministic and stochastic models, where each job have 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 processes tasks in an arbitrary order. The M-SRPT algorithm is proved to achieve a competitive ratio of $\Theta(\log \alpha + \beta)$ for minimizing response time, where α denotes the ratio between maximum job workload and minimum job workload, β 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 ρ 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/N system with finite task workload. As a special case, we show that M-SRPT is asymptotic optimal in M/M/1 model, in which the task size distribution is allowed to have infinite support.

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 operation 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.

Optimizing the flow time of single-task jobs has been considered both in offline and online scenarios. If preemption is allowed, the *shortest remaining processing time* (SRPT) discipline is shown to be optimal in single machine environment. Many generalizations of this basic formulation become NP-hard, for example, minimizing the total flow time in non-preemptive single machine model and preemptive model with two machines [7]. When jobs arrive online, no information

about jobs is known to the algorithm in advance, several algorithms with logarithmic competitive ratios are proposed in various settings [1, 7]. On the other hand, while SRPT minimizes the mean response time sample-path wise, it requires the knowledge of remaining job service time. Gittins proved that the Gittins index policy minimizes the mean delay in an M/G/1 queue, which only requires the access to the information about job size distribution.

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 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. Tasks can also be map, shuffle and reduce procedures in MapReduce framework. 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.*

Related Work. There has been a large literature on single-task job scheduling, with parallel developments taking place in competitive analysis and queuing theory. However, little is known about multitask scheduling. Scully et. al [11] presented the first theoretical analysis of *single-processor multitask scheduling* problem, and gave an optimal policy that is easy to compute for batch arrival, together with the assumption that the processing time of tasks satisfies the aged Pareto distributions. Sun et al. [12] studied the multitask scheduling problem when all the tasks are of unit size, and proved that among causal and non-preemptive policies, fewest unassigned tasks first (FUT) policy, earliest due date first (EDD) policy, and first come first serve (FCFS) are near delay-optimal in distribution (stochastic ordering) for minimizing the metric of average delay, maximum lateness and maximum delay respectively. To model the scenario when the scheduler has incomplete information about the job size, Scully et. al [16] introduced the multistage job model and proposed an optimal scheduling algorithm for multistage job scheduling in M/G/1 queue. In addition, the closed-form expression of the mean response time is given for the optimal scheduler.

As an concrete example of multiple-processor multitask scheduling, there is a separate line of work focusing on the MapReduce framework. Here we only mention a few as examples. Wang et al. [13] studied the problem of scheduling map tasks with data locality, and proposed a map task scheduling algorithm consisting of the Join the Shortest Queue policy and MaxWeight policy. The algorithm asymptotically minimizes the number of backlogged tasks (which is directly related to the delay performance based on Little's law), when the arrival rate vector approaches the capacity region boundary. Zheng et al. [15] proposed an online scheduler called available shortest remaining processing time (ASRPT), which is shown to achieve an efficiency ratio no more than two.

Contributions. In this paper, we investigate how to minimize the total response time of multitask jobs in a multi-server system and answer the aforementioned question. Our contributions are summarized as follows.

- We first propose Algorithm 1, the Modified SRPT algorithm, for minimizing the total response time. Algorithm 1 is a simple modification of SRPT and achieves a competitive ratio of

$O(\log \alpha + \beta)$, where α is the maximum-to-minimum job workload ratio, β represents the ratio between maximum non-preemptive task workload and minimum job workload. It can be shown that no $o(\log \alpha + \beta)$-competitive algorithm exists when the number of machines is constant. In addition, $O(\log \alpha + \beta^{1-\varepsilon})$ is the best possible competitive ratio for the class of work-conserving algorithms.

- Besides the worst case relative ratio above, we further prove our main result, absolute performance guarantees for Algorithm 1 under certain probabilistic structure on the input instances, in which the remaining workload bound established for the adversarial inputs contributes significantly to the stochastic analysis. Assuming that jobs arrive according to a Poisson process, *i.e.*, in M/GI/N system, we prove that the average response time incurred by Algorithm 1 is asymptotic optimal when load $\rho \to 1$, as long as the job size distribution has finite support. The assumption of finite job service time can be relaxed to finite task workload for exponentially distributed job size, *i.e.*, M/M/N, together with other infinite distributions with certain properties on the tail of the distribution. Last but not least, we prove the asymptotic optimality of Algorithm 1 in M/M/1 without the bounded task size assumption.

The remainder of this paper is organized as following. We introduce the problem definition, notations and necessary background in Section 2. In Section 4 we formally present Modified SRPT algorithm, together with the analysis of its competitive ratio and lower bounds. Section 5 is devoted to the proof of the asymptotic optimality of Modified SRPT in heavy traffic regime, together with the extensions to infinite job size distributions. We conclude our work in Section 6.

2 Model and preliminaries

Deterministic Model. We are given a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ 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.*, $p_i = \sum_{\ell \in n_i} p_{i,\ell}$, where $p_{i,\ell}$ represents the processing time of the ℓ-th task of job i. Tasks can be either preemptive or non-preemptive. A task is non-preemptive if it is not allowed to interrupt the task once it starts service, *i.e.*, the task is run to completion. 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. Formally, let $C_j^{(\ell)}$ be the completion time of task ℓ in job j, then $C_j = \max_{\ell \in [n_i]} C_j^{(\ell)}$. The response time of job j is defined as $F_j = C_j - r_j$, our objective is to minimize the total response time $\sum_{j \in [n]} F_j$.

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 $\eta = \max\{p_{i,\ell} |$ task ℓ of job i is non-preemptive$\}$ be the maximum processing time of a non-preemptive task, $\beta = \eta / \min_{i \in [n]} p_i$ be the ratio between η and minimum job workload. In some sense, parameters β and η represent the degree of non-preemptivity and exhibits a trade-off between the preemptive and non-preemptive setting. More specifically, the problem approaches the preemptive case when η is small, and degenerates to the non-preemptive case if all the jobs are consisted of a single non-preemptive task, in which η reaches the maximum value of $\max_{i \in [n]} p_i$.

The definitions of work-conserving algorithms and competitive ratios are formally given as following.

Definition 1 (Work-conserving scheduling algorithm)**.** *A scheduling algorithm π is called work-conserving if it never idles machines when there exists at least one feasible job or task awaiting the execution in the system. Here a job or task is called feasible, if it satisfies all the given constraints of the system (e.g, preemptive and non-preemptive constraint, precedence constraint, etc).*

N	number of machines
n	number of jobs
r_i	arrival time of job i
p_i	total workload of job i
η	maximum workload of a single non-preemptive task
α	job size ratio: $\alpha = \max_{i\in[n]} p_i / \min_{i\in[n]} p_i$
β	relative ratio of the longest non-preemptive task: $\eta / \min_{i\in[n]} p_i$
ρ	traffic intensity $\rho = \mathbb{E}[p_i]/(N \cdot \mathbb{E}[\Delta r_i])$
$\rho(y)$	load composed of jobs with size 0 to y: $\rho(y) = \lambda \cdot \int_0^y t f(t) dt$
F_ρ^π	job average response time under algorithm π and load ρ

Table 1: Notation Table

Definition 2 (Competitive ratio)**.** *The competitive ratio of online algorithm $\mathcal{A}$ refers to the worst ratio of the cost incurred by $\mathcal{A}$ and that of optimal offline algorithm $\mathcal{A}^*$ over all input instances ω in Ω,* i.e.*,*

$$\mathcal{CR}_{\mathcal{A}} = \max_{\omega\in\Omega} \frac{\mathrm{Cost}_{\mathcal{A}}(\omega)}{\mathrm{Cost}_{\mathcal{A}^*}(\omega)}.$$

In the multiple-processor multitask scheduling problem, the cost is the total response time under instance $\omega = \{(r_i, \{p_{i,\ell}\}_{\ell\in[n_i]})\}_{i\in[n]}$.

Stochastic Model. In the stochastic setting, we assume that jobs arrive into the system according to a Poisson process with rate λ. Job processing times are i.i.d distributed with probability density function $f(\cdot)$. Formally, we consider a sequence of M/GI/N queues indexed by n, the traffic intensity of the n-th system is equal to $\rho^{(n)} = \lambda^{(n)} \cdot \mathbb{E}[p_i^{(n)}]$, where $\lambda^{(n)}$ denotes the arrival rate of the n-th Poisson arrival process, job workload distribution has a density function of $f^{(n)}(\cdot)$. Stability of the queuing systems requires that $\rho^{(n)} < 1$ for $\forall n$. As standard in the literature, we assume that $\rho^{(n)} \to 1$ when $n \to \infty$. In this paper, we further assume that the probability density function $f^{(n)}(\cdot)$ is continuous. For notational convenience, we will suppress index n whenever it is clear from the context.

The stochastic analysis in this paper relies heavily on the concept of busy period, which is defined as following.

Definition 3 (Busy Period [6])**.** *Busy period is defined to be the longest time interval in which no machines are idle.*

We use $\mathsf{B}(w)$ to denote the length of a busy period with started by a workload of w. It can be seen that $\mathsf{B}(\cdot)$ is an additive function [6], *i.e.*, $\mathsf{B}(w_1 + w_2) = \mathsf{B}(w_1) + \mathsf{B}(w_2)$ for $\forall w_1, w_2$, since a busy period with initial workload of $w_1 + w_2$ can be regarded as a busy period started by initial workload w_2, following a busy period started by initial workload w_1. Moreover, for M/GI/1 queue, the length of a busy period with initial workload of w and load ρ is shown to be equal to $\mathsf{B}(w) = \mathbb{E}[w]/(1-\rho)$ [6].

2.1 Notations

Notations of this paper are summarized in Table 1. Most of our analysis are presented using asymptotic notations. We say $f(n) = o(g(n))$, $f(n) = O(g(n))$, $f(n) = \Theta(g(n))$, $f(n) = \Omega(g(n))$ if and only if $\limsup_{n\to\infty} f(n)/g(n) = 0$, $\limsup_{n\to\infty} f(n)/g(n) < \infty$, $0 < \liminf_{n\to\infty} f(n)/g(n) \leq \limsup_{n\to\infty} f(n)/g(n) < \infty$ and $\liminf_{n\to\infty} f(n)/g(n) > 0$ respectively. All these notations only hide quantities that do not scale with n (or $\rho^{(n)}$).

3 Challenges with multi-task scheduling

SRPT and its analysis do not easily generalize to multi-task scenario, due to the non-preemptivity of tasks. Firstly, in the analysis of single-task scheduling, the server only processes relevant work during the waiting time of the tagged job, under SRPT discipline. This holds for both single-server and multi-server settings. When jobs have multiple tasks, a challenge is raised: the system might be dealing with non-preemptive task of irrelevant jobs upon the arrival of the tagged job. It is unknown how the algorithm should be designed and how the amount of irrelevant workload involved can be bounded. Secondly, [5] bound the relevant work by comparing the multi-server SRPT system with single server system using SRPT. The analysis relies on the following fact: the workload difference is bounded in few-jobs interval and is non-decreasing in many-jobs interval, since the two systems are experiencing identical arrival sequence, while multi-server SRPT processes relevant workload at a maximum rate in such interval. However, in multi-task scheduling, the workload difference might be decreasing in many-jobs interval as resources might be used to process irrelevant jobs.

4 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 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 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`
- $\mathcal{J}_1(t) \leftarrow \{i \in [n] | w_i(t) = 0\}$`// Jobs with tasks that are finished at time` t
- $\mathcal{J}_2(t) \leftarrow \{i \in [n] |$task of job i is being processed at time t and is preemptive$\}$

and assign alive jobs to the $|\mathcal{J}_1(t) \cup \mathcal{J}_2(t)|$ machines, where jobs with smaller value of $W_i(t)$ have a higher priority.

4.1 Performance Analysis

Our main result is stated in the following theorem.

Theorem 4. *Algorithm 1 achieves a competitive ratio that is no more than*

$$\mathcal{CR}_{\mathrm{M-SRPT}} \leq 4 \log \alpha + 2\beta + 8.$$

To show the competitive ratio above, we divide the jobs into different classes and compare the remaining number of jobs under Algorithm 1 with that under optimal algorithm π^*. For any algorithm π, at time slot t, we divide the unfinished jobs into $\Theta(\log \alpha)$ classes $\{\mathcal{C}_k(\pi, t)\}_{k \in [\log \alpha + 1]}$, based on their remaining workload. 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,

$$\mathcal{C}_k(\pi, t) = \left\{ i \in [n] \;\middle|\; W_i(\pi, t) \in (2^{k-1}, 2^k] \right\},$$

where $W_i(\pi, t)$ represents the unfinished workload of job i at time t. In the following analysis, we use $\mathcal{C}^{[k]}(\pi, t) = \cup_{i=1}^{k} \mathcal{C}_i(\pi, t)$ to denote the collection of jobs in the first k classes, and let $W_\pi^{[k]}(t) = \sum_{i=1}^{k} W_\pi^{(i)}(t)$ represent the total remaining workload of jobs in the first k classes, where $W_\pi^{(k)}(\pi, t)$ denotes the amount of remaining workload of jobs in class $\mathcal{C}_k(\pi, t)$. $W_{\pi^*}^{(k)}(t)$ and $W_{\pi^*}^{[k]}(t)$ are defined in a similar way for the optimal scheduling algorithm π^*.

Similar to the proof in [8], we first show the following lemma, which relates the remaining workload under M-SRPT with that under optimal algorithm π^*, then complete the proof of Theorem 4 in Appendix A.

Lemma 5. *For $\forall k, t \geq 0$, the unfinished workload under Algorithm 1 can be upper bounded as*

$$W_{\mathrm{M-SRPT}}^{[k]}(t) \leq W_{\pi^*}^{[k]}(t) + N \cdot (2^{k+1} + \eta). \tag{1}$$

Proof: In the following of the proof, we always divide jobs into different classes according to the remaining workload under M-SRPT, we suppress reference to M-SRPT in the notation of $\mathcal{C}_k$. Without loss of generality we can assume that $W_{\mathrm{M-SRPT}}^{[k]}(t) > W_{\pi^*}^{[k]}(t)$, otherwise Lemma 5 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 in $(0, t]$, at which either

- Idle machines exist under M-SRPT;
- Jobs with remaining workload (under M-SRPT) larger than 2^k are processed.

Otherwise, all the machines will be processing jobs belonging to set $\mathcal{C}_{[k]}(t)$ before time t, while no jobs in higher classes, *i.e.*, $\cup_{i>k}\mathcal{C}_i(t)$, will be switched into class $\mathcal{C}_{[k]}(t)$. Combining with the fact that the initial workload under Algorithm 1 and optimal algorithm are identical, *i.e.*, $W^{[k]}_{\text{M-SRPT}}(0) = W^{[k]}_{\pi^*}(0)$, we can see that $W^{[k]}_{\text{M-SRPT}}(t)$ should be no more than $W^{[k]}_{\pi^*}(t)$ and the contradiction appears.

Now consider the following two collections of time:

$$\mathcal{T}_k^{(1)} = \Big\{\bar{t} \in [0,t] \Big| \text{ At time } \bar{t}, \text{ at least one machine is idle under Algorithm 1}\Big\},$$
$$\mathcal{T}_k^{(2)} = \Big\{\bar{t} \in [0,t] \Big| \text{ At time } \bar{t}, \text{ there exists } i > k \text{ such that at least one machine is processing jobs in } \mathcal{C}_i \text{ under Algorithm 1}\Big\}.$$

Let $\bar{t}_k^{(i)} = \max\{t | t \in \mathcal{T}_k^{(i)}\}$ $(i \in \{1,2\})$ be the last time slot in $\mathcal{T}_k^{(i)}$, based on which we divide our proof into the following two cases.

Case 1: $\bar{t}_k^{(1)} \geq \bar{t}_k^{(2)}$**.** From the definition of $\bar{t}_k^{(1)}$, it can be seen that during $(\bar{t}_k^{(1)}, t]$, no machines are idle or process jobs with remaining workload larger than 2^k under Algorithm 1, while the increment in remaining workload incurred by newly arriving jobs are identical for Algorithm 1 and π^*. In addition, it is important to point out that $([n] \setminus \mathcal{C}_{[k]}(\bar{t}_k^{(1)})) \cap \mathcal{C}_{[k]}(\tilde{t}) = \emptyset$ for $\forall \tilde{t} \in (\bar{t}_k^{(1)}, t]$, *i.e.*, no job will switch from a higher class to $\mathcal{C}_{[k]}$ during $(\bar{t}_k^{(1)}, t]$. Hence

$$W^{[k]}_{\text{M-SRPT}}(t) - W^{[k]}_{\pi^*}(t) \leq W^{[k]}_{\text{M-SRPT}}(\bar{t}_k^{(1)}) - W^{[k]}_{\pi^*}(\bar{t}_k^{(1)}).$$

It suffices to prove the workload difference inequality (1) for $t = \bar{t}_k^{(1)}$, *i.e.*,

$$W^{[k]}_{\text{M-SRPT}}(\bar{t}_k^{(1)}) \leq W^{[k]}_{\pi^*}(\bar{t}_k^{(1)}) + N \cdot (2^{k+1} + \eta). \tag{2}$$

Note that there exists some idle machines at time $t = \bar{t}_k^{(1)}$, which implies that under Algorithm 1, the number of jobs alive must be less than N. Hence $W^{[k]}_{\text{M-SRPT}}(\bar{t}_k^{(1)}) \leq (N-1) \cdot 2^k$ and (2) holds.

Case 2 : $\bar{t}_k^{(1)} < \bar{t}_k^{(2)}$**.** According to the definition of $\bar{t}_k^{(2)}$, there exist jobs with remaining workload larger than 2^k being processed at $\bar{t}_k^{(2)}$, we use $\hat{\mathcal{J}}(\bar{t}_k^{(2)}) \subseteq [n] \setminus \mathcal{C}_{[k]}(\bar{t}_k^{(2)})$ to denote the collection of such jobs.

When all the tasks are processed preemptively, we can obtain (1) directly, as we are able to conclude that there are at most $N-1$ jobs in $\mathcal{C}_{[k]}(\bar{t}_k^{(2)})$. This is because that tasks are allowed to be preempted, and Algorithm 1 selects a job with remaining workload larger than 2^k at time $\bar{t}_k^{(2)}$. Consequently $W^{[k]}_{\text{M-SRPT}}(\bar{t}_k^{(2)}) \leq n^{[k]}_{\text{M-SRPT}}(\bar{t}_k^{(2)}) \cdot 2^k$, and for $\forall t > \bar{t}_k^{(2)}$,

$$W^{[k]}_{\text{M-SRPT}}(t) - W^{[k]}_{\pi^*}(t) \leq W^{[k]}_{\text{M-SRPT}}(\bar{t}_k^{(2)}) - W^{[k]}_{\pi^*}(\bar{t}_k^{(2)}) + [N - n^{[k]}_{\text{M-SRPT}}(\bar{t}_k^{(2)})] \cdot 2^k \leq N \cdot 2^k,$$

where the first inequality follows from the fact that no more than $N - n^{[k]}_{\text{M-SRPT}}(\bar{t}^{(2)}_k)$ jobs switches from higher classes to $\mathcal{C}_{[k]}(t)$, as there are at most $N - n^{[k]}_{\text{M-SRPT}}(\bar{t}^{(2)}_k)$ jobs with remaining workload larger than 2^k are being processed at time $\bar{t}^{(2)}_k$. Hence Lemma 5 holds.

Now for the case when there exist non-preemptive tasks, arguments above does not work, because machines may be processing tasks with remaining workload larger than 2^k and hence $n^{[k]}_{\text{M-SRPT}}(\bar{t}^{(2)}_k)$ may be larger than N. Let $r \in [N]$ be the number of tasks that are being processed at time $\bar{t}^{(2)}_k$ and belongs to $[n] \setminus \mathcal{C}_{[k]}(\bar{t}^{(2)}_k)$, and $t_s \le \bar{t}^{(2)}_k$ be the latest starting processing time of these tasks. We divide our analysis into the following two subcases:

- **Case** 2.1**:** No jobs switch from set $[n]\setminus\mathcal{C}_{[k]}(t_s)$ to $\mathcal{C}_{[k]}(\bar{t}^{(2)}_k)$ under Algorithm 1. We use Δ_k to represent the increment of $W^{[k]}_{\text{M-SRPT}}$, incurred by the newly arriving jobs during time period $[t_s, \bar{t}^{(2)}_k]$. Then we have:

$$W^{[k]}_{\text{M-SRPT}}(\bar{t}^{(2)}_k) - W^{[k]}_{\text{M-SRPT}}(t_s) = -(N-r)(\bar{t}^{(2)}_k - t_s) + \Delta_k. \tag{3}$$

  On the other hand, $W^{[k]}_{\pi^*}$, the remaining workload of jobs in class $\mathcal{C}_{[k]}$ under the optimal algorithm π^*, decreases at a speed that is no more than N units of workload per time slot, hence

$$W^{[k]}_{\pi^*}(\bar{t}^{(2)}_k) - W^{[k]}_{\pi^*}(t_s) \ge -N \cdot (\bar{t}^{(2)}_k - t_s) + \Delta_k. \tag{4}$$

  According to the definition of $\bar{t}^{(2)}_k$, no jobs with remaining workload larger than 2^k are processed in $(\bar{t}^{(2)}_k, t]$. Compared with time $\bar{t}^{(2)}_k$, there are at most r jobs switch from $[n]\setminus\mathcal{C}_{[k]}(\bar{t}^{(2)}_k)$ to set $\mathcal{C}_{[k]}(\bar{t}^{(2)+}_k)$. Therefore

$$W^{[k]}_{\text{M-SRPT}}(\bar{t}^{(2)+}_k) - W^{[k]}_{\pi^*}(\bar{t}^{(2)+}_k) \le W^{[k]}_{\text{M-SRPT}}(\bar{t}^{(2)}_k) - W^{[k]}_{\pi^*}(\bar{t}^{(2)}_k) + r \cdot 2^k. \tag{5}$$

  Combining inequalities (3)—(5), we can obtain

$$\begin{aligned} W^{[k]}_{\text{M-SRPT}}(t) - W^{[k]}_{\pi^*}(t) &\le W^{[k]}_{\text{M-SRPT}}(\bar{t}^{(2)+}_k) - W^{[k]}_{\pi^*}(\bar{t}^{(2)+}_k) \\ &\le W^{[k]}_{\text{M-SRPT}}(t_s) - W^{[k]}_{\pi^*}(t_s) + r \cdot [2^k + (\bar{t}^{(2)}_k - t_s)] \\ &\le (N-1) \cdot 2^k + r \cdot (\bar{t}^{(2)}_k - t_s) \\ &\le N \cdot (2^k + \eta). \end{aligned}$$

  The third inequality above holds since at time t_s, Algorithm 1 is required to do job selection and a job with remaining workload larger than 2^k is selected. The last inequality follows from the fact that $\bar{t}^{(2)}_k - t_s \le \eta$, as t_s is the starting time of a non-preemptive task that is still alive at time $\bar{t}^{(2)}_k$.

- **Case** 2.2**:** There exist jobs switching from set $[n]\setminus\mathcal{C}_{[k]}(t_s)$ to $\mathcal{C}_{[k]}(\bar{t}^{(2)}_k)$ under Algorithm 1. We use $\mathcal{J}_s$ to denote the collection of such switching jobs. It is essential to bound the number of switching jobs, which will incur an increment of $|\mathcal{J}_s| \cdot 2^k$ in the remaining workload of class $\mathcal{C}_{[k]}$. A straightforward bound is $|\mathcal{J}_s| \le N \cdot (\bar{t}^{(2)}_k - t_s) \le N \cdot \eta$, since at most N jobs receive service

at each time slot, and hence the number of switching jobs is no more than N. However, this bound is indeed loose, we argue that

$$|\mathcal{J}_s| \leq N - r. \tag{6}$$

Notice that after a job switches to class $\mathcal{C}_{[k]}$ during $[t_s, \bar{t}_k^{(2)}]$, it will only be preempted by jobs that are also in class $\mathcal{C}_{[k]}$, which is due to the SRPT rule. According to the precondition of this case, there are r jobs in set $[n]\setminus\mathcal{C}_{[k]}$ that are continuously being processed during $[t_s, \bar{t}_k^{(2)}]$, hence at most $N-r$ units of resources per time slot are available for the remaining jobs. Note that resources that are allocated to jobs in $\mathcal{C}_{[k]}$ will not be utilized for switching a job from a higher class to $\mathcal{C}_{[k]}$. In addition, finished jobs will have no contribution to the total remaining workload $W^{[k]}_{\mathrm{M-SRPT}}(t)$. Hence $|\mathcal{J}_s|$ is no more than $N - r$.

Furthermore, we can derive the following conclusion:

$$\begin{aligned}
&W^{[k]}_{\mathrm{M-SRPT}}(t) - W^{[k]}_{\pi^*}(t) \\
\leq &W^{[k]}_{\mathrm{M-SRPT}}(\bar{t}_k^{(2)+}) - W^{[k]}_{\pi^*}(\bar{t}_k^{(2)+}) \\
\leq &W^{[k]}_{\mathrm{M-SRPT}}(\bar{t}_k^{(2)}) - W^{[k]}_{\pi^*}(\bar{t}_k^{(2)}) + r\cdot 2^k + N && \text{(job switching at } t^{(2)}\text{)} \\
\leq &[W^{[k]}_{\mathrm{M-SRPT}}(t_s) - W^{[k]}_{\pi^*}(t_s) + (N-r)\cdot 2^k \\
&+ N\cdot(\bar{t}_k^{(2)} - t_s)] + r\cdot 2^k + N && \text{(job switching during } [t_s, \bar{t}_k^{(2)}]\text{)} \\
\leq &N\cdot(2^{k+1} + \eta + 1). && (\bar{t}_k^{(2)} - t_s \leq \eta)
\end{aligned}$$

The proof is complete. □

4.2 Competitive ratio lower bound

The following lower bounds mainly follow from the observation that, multiple-processor multitask scheduling problem generalizes the single-task job scheduling problem in both preemptive and non-preemptive settings.

Proposition 6. *For multiple-processor multitask scheduling problem with constant number of machines, there exists no algorithm that achieves a competitive ratio of* $o(\log\alpha + \beta)$.

Proof: When $p_{\min} = \eta = 1$, the problem degenerates to preemptive setting and no algorithm can achieve a competitive ratio of $o(\log\alpha)$ [8]. When $\eta = p_{\max}$, the problem degenerates to the non-preemptive setting and $O(\beta)$ is the best possible competitive ratio if the number of machines is constant [4]. The proof is complete. □

Proposition 7. *For multiple-processor multitask scheduling problem, the competitive ratio of any work-conserving algorithms have an competitive ratio of* $\Omega(\log\alpha + \beta^{1-\varepsilon})$ *for* $\forall\varepsilon > 0$.

Proof: The reasoning is similar as the proof of Proposition 6, since work-conserving algorithms cannot achieve a competitive ratio of $o(\beta^{1-\varepsilon})$ in the non-preemptive single-task job scheduling [4]. □

5 Asymptotic Optimality of Modified SRPT with Poisson Arrival

In this section we show that under mild probabilistic assumptions, Algorithm 1 is asymptotic optimal for minimizing the total response time in the heavy traffic regime. The result is formally stated as following.

Theorem 8. *Let $F_\rho^{\text{M}-\text{SRPT}}$ and $F_\rho^{\pi^*}$ be the response time incurred by Algorithm 1 and optimal algorithm respectively, when the traffic intensity is equal to ρ. In an M/GI/N with finite job size distribution, Algorithm 1 is heavy traffic optimal,* i.e.*,*

$$\lim_{\rho\to 1} \frac{\mathbb{E}[F_\rho^{\text{M}-\text{SRPT}}]}{\mathbb{E}[F_\rho^{\pi^*}]} = 1. \tag{7}$$

The probabilistic assumptions here are with respect to the distribution of job size, *i.e.*, the total workload of tasks. For the processing time of a single task, the only assumption we have is the upper bound η, which is finite since the job size distribution has finite support. It can be seen that the optimality result in [5] corresponds to a special case of Theorem 8.

5.1 Average response time bound

We first remark that Lemma 5 can be extended to any non-negative number $y \geq 0$.

Lemma 9. *The difference of the amount of remaining workload under Algorithm 1 and that under* SRPT *algorithm in a single server system with speed N, is upper bounded by*

$$\mathsf{W}_{\leq y}^{\text{M}-\text{SRPT}}(t) - \mathsf{W}_{\leq y}^{\text{SRPT}_{1,N}}(t) \leq N \cdot (2y + \eta), \forall y, t \geq 0,$$

where $\text{SRPT}_{k,\ell}$ *denotes the* SRPT *algorithm in a system with k servers, and each server has a speed of ℓ.*

Proof: The proof is identical to that of Lemma 5. □

Our main goal is to derive the following analytical upper bound on $\mathbb{E}[F_\rho^{\text{M}-\text{SRPT}}]$.

Theorem 10. *The average response time under Algorithm 1 satisfies that*

$$\mathbb{E}[F_\rho^{\text{M}-\text{SRPT}}] \leq \mathbb{E}[F_\rho^{\text{SRPT}_{1,N}}] + O\Big(\log \frac{1}{1-\rho}\Big). \tag{8}$$

Proof: Similar as the techniques in [5, 10], we relate the response time of the tagged job with an appropriate busy period.

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 $\mathsf{W}_{\text{waste}}(r_x)$ to represent the amount of such resources, then

   $$\mathsf{W}_{\text{waste}}(r_x) \leq (N-1) \cdot x, \tag{9}$$

   which is indeed the same as Lemma 5.1 in [5]. The reason is straightforward—as the tagged job must be in service, hence the number of such time slots should not exceed x, and thus (9) holds.

2. The system may be dealing with *jobs with remaining workload no more than x at time r_x*, the amount of resources spent on this class is no more than $\mathsf{W}_{\leq x}^{\mathrm{M-SRPT}}(r_x)$. Here for any algorithm π, we use $\mathsf{W}_{\leq x}^{\pi}(t)$ to denote the total workload of jobs with remaining workload no more than x at time t.

3. 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 $\mathsf{W}_{\mathrm{non-pm}}(r_x)$ 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,

   $$\mathsf{W}_{\mathrm{non-pm}}(r_x) \leq (N^2 + N) \cdot \eta + N \cdot x. \tag{10}$$

   To see the correctness of inequality (10), 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:
     - Jobs that have a remaining workload larger than x at time $t = r_x + \eta$. Note that jobs of this class 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 $\mathsf{W}_{\mathrm{waste}}(r_x)$, and we can ignore this subclass.
     - For the collection of jobs with remaining workload no more than x at time $t = r_x + \eta$, it is clear to see that the remaining workload of such jobs at time $t = r_x$ must be no more than $x + N \cdot \eta$ (different tasks within the same job might be processed in parallel). 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 $N \cdot (x + N \cdot \eta) = N \cdot x + N^2 \cdot \eta$, which implies that $\mathsf{W}_{\mathrm{non-pm}}(r_x) \leq N \cdot x + N^2 \cdot \eta + N\eta$ and (10) holds.

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

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

Hence $\mathsf{T}_x^{\mathrm{M-SRPT}}$, 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 a initial workload of

$$\mathsf{W}_{\mathrm{waste}}(r_x) + \mathsf{W}_{\mathrm{non-pm}}(r_x) + \mathsf{W}_{\leq x}^{\mathrm{M-SRPT}}(r_x) + x.$$

Combining with the aforementioned analysis, formally we have

$$\begin{aligned}
\mathsf{T}_x^{\mathrm{M-SRPT}} &\leq_{st} \mathsf{B}^{(\rho_x)}\Big(\mathsf{W}_{\mathrm{waste}}(r_x)+\mathsf{W}_{\mathrm{non-pm}}(r_x)+\mathsf{W}_{\leq x}^{\mathrm{M-SRPT}}(r_x)+x\Big)\\
&\overset{(a)}{=} \mathsf{B}^{(\rho_x)}\Big(\mathsf{W}_{\mathrm{waste}}(r_x)+\mathsf{W}_{\mathrm{non-pm}}(r_x)+x\Big)+\mathsf{B}^{(\rho_x)}\Big(\mathsf{W}_{\leq x}^{\mathrm{M-SRPT}}(r_x)\Big)\\
&\overset{(b)}{\leq} \mathsf{B}^{(\rho_x)}\Big(N^2\cdot\eta+(2x+\eta)\Big)+\mathsf{B}^{(\rho_x)}\Big(\mathsf{W}_{\leq x}^{\mathrm{M-SRPT}}(r_x)\Big)\\
&\overset{(c)}{\leq} \underbrace{\mathsf{B}^{(\rho_x)}\Big(3N^2\cdot(\eta+x)\Big)}_{\Sigma_1}+\underbrace{\mathsf{B}^{(\rho_x)}\Big(\mathsf{W}_{\leq x}^{\mathrm{SRPT}_{1,N}}(r_x)\Big)}_{\Sigma_2},
\end{aligned}$$

where (a) follows from the additivity of busy period; In (b) we utilize the upper bounds established in (9) and (10) and (c) follows from Lemma 9.

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

$$\mathbb{E}[F_\rho^{\mathrm{SRPT}_{1,N}}]\geq \mathbb{E}_{x,r_x}[\mathsf{B}^{(\rho(x))}(W_{\leq x}^{\mathrm{SRPT}_{1,N}}(r_x))]=\mathbb{E}_{x,r_x}[\Sigma_2], \tag{11}$$

where the first equality holds due to the Poission Arrivals See Time Average (PASTA) property [14]. Note that

$$\begin{aligned}
\mathbb{E}[\Sigma_1] &= O\Big(\mathbb{E}\Big(\mathsf{B}^{(\rho(x))}(\eta+x)\Big)\Big)=O\Big(\mathbb{E}\Big[\frac{\eta+x}{1-\rho(x)}\Big]\Big)\\
&= O\Big(\log\frac{1}{1-\rho}\Big)+\mathbb{E}[\eta]\cdot O\Big(\int_0^\infty\frac{f(x)}{1-\rho(x)}dx\Big).
\end{aligned}\tag{12}$$

In addition,

$$\int_0^\infty\frac{f(x)}{1-\rho(x)}dx=\int_0^\xi\frac{f(x)}{1-\rho(x)}dx+\int_\xi^\infty\frac{f(x)}{1-\rho(x)}dx\leq\frac{1}{1-\rho(\xi)}+\frac{1}{\xi}\cdot\int_\xi^\infty\frac{xf(x)}{1-\rho(x)}dx, \tag{13}$$

where ξ satisfies that $\rho(\xi)=\rho/2$. Note that

$$\rho(\xi)=\lambda\cdot\int_0^\xi tf(t)dt\leq\lambda\cdot\xi,$$

hence we have $\xi\geq\mathbb{E}[p_i]/2$. Then the right hand side of (12) can be further bounded as

$$\int_0^\infty\frac{f(x)}{1-\rho(x)}dx\leq 2+\frac{2}{\mathbb{E}[p_i]}\cdot\int_0^\infty\frac{xf(x)}{1-\rho(x)}dx=2+\frac{2}{\mathbb{E}[p_i]}\cdot\log\frac{1}{1-\rho}.$$

Therefore for any input instance, the average response time under Modified-SRPT, is no more than,

$$\begin{aligned}
\mathbb{E}[F_\rho^{\mathrm{M-SRPT}}] &= \mathbb{E}_{x,r_x}[\mathsf{T}_x^{\mathrm{M-SRPT}}]=\mathbb{E}_{x,r_x}[\Sigma_1]+\mathbb{E}_{x,r_x}[\Sigma_2]\\
&\leq \mathbb{E}[F_\rho^{\mathrm{SRPT}_{1,N}}]+O\Big(\log\frac{1}{1-\rho}\Big).
\end{aligned}\tag{14}$$

The proof is complete. □

5.2 Existing lower bound for M/GI/1

To start with, we consider the benchmark system consisting of a single machine with speed N, where all the tasks can be allowed to be served in preemptive fashion, *i.e.*, the concept of task is indeed unnecessary in this setting. It is clear to see that the mean response time under optimal algorithm for this single machine system can be performed as a valid lower bound for the multitask problem, *i.e.*,

$$\mathbb{E}[F_\rho^{\pi^*}] \geq \mathbb{E}[F_\rho^{\mathrm{SRPT}_{1,N}}]. \tag{15}$$

It is well-known that SRPT minimizes the average response time in single server system. For the case when job size distribution has finite support, Lin et al. [9] derived the heavy traffic growth rate of the average response time under SRPT [9].

Lemma 11 ([9])**.** *In an $M/GI/1$ with finite job size distribution, the average response time under SRPT is in the order of*

$$\mathbb{E}[F_\rho^{\mathrm{SRPT}_{1,1}}] = \Theta\Big(\frac{1}{1-\rho}\Big).$$

5.3 Proof of optimality

To achieve heavy traffic optimality, it suffices to show that the difference between the average response time under Algorithm 1 and the optimal algorithm is a lower order term, *i.e.*,

$$\lim_{n\to\infty} \frac{\mathbb{E}[F_\rho^{\mathrm{M-SRPT}}] - \mathbb{E}[F_\rho^{\mathrm{SRPT}_{1,N}}]}{\mathbb{E}[F_\rho^{\mathrm{SRPT}_{1,N}}]} = 0, \tag{16}$$

which holds according to Lemma 11 and inequality (14)-(15).

5.4 Beyond Job Size Distribution with Finite Support

Up to this point, we have focused on job size distributions with finite support, which is rather restrictive. It is natural to consider various relaxations of this assumption. In this section, we turn to other classes of job size distributions and the scenario when there are random number of tasks. These results provide complement to our developments about the theory of the asymptotic optimality of Modified SRPT.

5.4.1 Exponential distribution and beyond

M/M/N model. 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 12. *The average response time under Algorithm 1 is asymptotic optimal in M/M/N, if task workload is finite.*

Proof: The conclusion follows from the fact that in M/M/1 [2],

$$\frac{1/(18e)}{\mu(1-\rho)\log(1/(1-\rho))} \leq \mathbb{E}[F_\rho^{\mathrm{SRPT-1}}] \leq \frac{7}{\mu(1-\rho)\log(1/(1-\rho))}.$$

□

M/M/1 model. For single server with Poisson arrival and exponentially distributed workload, we show that Modified SRPT is asymptotic optimal without any finite workload assumptions.

Theorem 13. *Algorithm 1 is asymptotic optimal in M/M/1.*

We first introduce the following propositions that will be used in our proof.

Proposition 14. *The expected value of the maximum of n i.i.d exponentially distributed random variables with mean $1/\mu$ is*

$$(1/\mu) \cdot \sum_{k=1}^{n} (1/k) = \Theta(\log n).$$

Proposition 15 ([3])**.** *For M/M/1 model and any work-conserving algorithm, let n_{busy} be the number of arrivals in a busy period, then*

$$\mathbb{E}[n_{\text{busy}}] = O\Big(\frac{1}{1-\rho}\Big).$$

Proof of Theorem 13: From the proof of Theorem 10, it can be verified that the average response time under Algorithm 1 is no more than

$$\mathbb{E}[F_\rho^{\text{M-SRPT}}] \le \mathbb{E}[F_\rho^{\text{SRPT}_{1,N}}] + \Big(1 + \mathbb{E}[\eta]\Big) \cdot O\Big(\log \frac{1}{1-\rho}\Big). \tag{17}$$

Based on Proposition 14, Proposition 15 and Jenson's inequality, we have

$$\mathbb{E}[\eta] \le (1/\mu) \cdot \mathbb{E}\Big[\sum_{k=1}^{n_{\text{busy}}} (1/k)\Big] \le (1/\mu) \cdot \sum_{k=1}^{\mathbb{E}[n_{\text{busy}}]} (1/k) = O\Big(\log \frac{1}{1-\rho}\Big).$$

This implies that

$$\mathbb{E}[F_\rho^{\text{M-SRPT}}] - \mathbb{E}[F_\rho^{\text{SRPT}_{1,N}}] \le O\Big(\log^2 \frac{1}{1-\rho}\Big),$$

which is a lower order term. The proof is complete. □

M/GI/N model. In addition to exponential distribution, Lin et al. [9] also gave a characterization of the heavy-traffic behavior of SRPT with general job size distribution. We first introduce the background on Matuszewska index.

Definition 16 (Upper Matuszewska Index [9])**.** *Let f be a positive function defined in $[0, \infty)$, the upper Matuszewska index is defined as the infimum of α for which there exists a constant $C = C(\alpha)$ such that for each $\bar{\lambda} > 1$,*

$$\lim_{x \to \infty} \frac{f(\gamma x)}{f(x)} \le C\gamma^\alpha,$$

holds uniformly for $\lambda \in [1, \bar{\lambda}]$.

Proposition 17 ([9])**.** *In an $M/GI/1$ queue, if the upper Matuszewska index of the job size distribution is less than -2, then*

$$\mathbb{E}[F_\rho^{\mathrm{SRPT}-1}] = \Theta\Big(\frac{1}{(1-\rho)\cdot G^{-1}(\rho)}\Big),$$

where $G^{-1}(\cdot)$ denotes the inverse of $G(x) = \rho_{\le x}/\rho = \int_0^x tf(t)dt/\mathbb{E}[p_i]$.

For example, exponential distribution has an upper Matuszewska index $M_f = -\infty$ and $G^{-1}(\rho) = \Theta(\log(1/(1-\rho)))$, hence Theorem 12 is also implied by Proposition 17. In addition, from Proposition 17, we can see that the following theorrem holds.

Theorem 18. *The average response time under Algorithm 1 is asymptotic optimal in M/GI/N, if task workload is finite,*

$$G^{-1}(\rho) = o\Big(\frac{1}{(1-\rho)\cdot\log(1/(1-\rho))}\Big),$$

and upper Matuszewska index of job size distribution is less than -2.

Examples include but not limited to *Weibull distribution*, *Pareto distribution* and *regularly varying distributions*. Details are deferred in Appendix B.

5.4.2 Random number of tasks

In the following proposition, we prove that the expected value of the maximum task size is finite, if the moment generating function of the task size distribution is finite.

Proposition 19. *If the number of jobs and the number of tasks in each job are independently distributed with finite mean value, then the mean value of the maximum task size is no more than,*

$$\mathbb{E}[\eta] \le \min_{s\in D(s)} \frac{\log(\mathbb{E}[n_t]) + \log m(s)}{s} < \infty,$$

where $m(s) = \mathbb{E}[e^{sp_i^\ell}]$ denotes the moment generating function of the task size distribution and $D(s) = \{s | m(s) < \infty\}$.

Proof: We first note that the expected value of the total number of tasks $\mathbb{E}[n_t] = \mathbb{E}[n]\cdot[n_i] < \infty$, given that $\mathbb{E}[n], \mathbb{E}[n_i] < \infty$. For any $s > 0$, we have

$$e^{s\mathbb{E}[\eta]} = e^{\sum_{k\ge 0} s\mathbb{E}[\eta|n_t=k]\cdot\mathbb{P}(n_t=k)} = \prod_{k\ge 0} e^{s\mathbb{E}[\eta|n_t=k]\cdot\mathbb{P}(n_t=k)},$$

where $e^{s\mathbb{E}[\eta|n_t=k]} \le \sum_{i=1}^{n_t}\sum_{j=1}^{\ell}\mathbb{E}[e^{sp_i^\ell}] = k\cdot m(s)$, which implies that

$$\begin{aligned} e^{s\mathbb{E}[\eta]} &\le \prod_{k\ge 0}(k\cdot m(s))^{\mathbb{P}(n_t=k)} = m(s)\cdot e^{\sum_{k\ge 0}\log k\cdot\mathbb{P}(n_t=k)} \\ &= m(s)\cdot e^{\mathbb{E}[\log n_t]} \le \mathbb{E}[n_t]\cdot m(s), \end{aligned}$$

where the last inequality follows from the fact that $\mathbb{E}[\log(n_t)] \le \log(\mathbb{E}[n_t])$. Hence the expected maximum task size

$$\mathbb{E}[\eta] \le \min_{s\in D(s)} \frac{\log(\mathbb{E}[n_t]) + \log m(s)}{s} < \infty.$$

□

Lemma 20. *In M/GI/N queue, Algorithm 1 is heavy traffic optimal with random number of jobs and tasks, if the upper Matuszewska index of the job size distribution is less than* -2 *and* $G^{-1}(\rho) = o(\frac{1}{(1-\rho)\cdot\log(1/(1-\rho))})$.

Proof: The Lemma mainly follows from Proposition 17, Proposition 19 and inequality (17). □

6 Conclusion

In this work, we study the multitask scheduling problem, for which the optimal algorithms and tight analyses remain widely open for almost all settings. We propose Modified-SRPT algorithm, which achieves a competitive ratio that is order optimal when the number of machines is constant. Another appealing and more important property of Modified-SRPT is that, the average response time incurred under Poisson arrival is asymptotic optimal when the traffic intensity goes to 1, if job service times are finite or exponentially distributed with finite task workload. We also show that this bounded workload assumption can be removed in M/M/1.

References

[1] Yossi Azar and Noam Touitou. Improved online algorithm for weighted flow time. In *FOCS*, pages 427–437, 2018.

[2] Nikhil Bansal. On the average sojourn time under M/M/1/SRPT. *Operation Research Letters*, 33(2):195–200, 2005.

[3] Nikhil Bansal, Bart Kamphorst, and Bert Zwart. Achievable performance of blind policies in heavy traffic. *Mathematics of Operations Research*, 43(3):949–964, 2018.

[4] David Pattison Bunde. Approximating total flow time. *Master Thesis*, 2002.

[5] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. Srpt for multiserver systems. *Performance Evaluation*, 127:154–175, 2018.

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

[7] Stefano Leonardi and Danny Raz. Approximating total flow time on parallel machines. In *STOC*, pages 110–119, 1997.

[8] Stefano Leonardi and Danny Raz. Approximating total flow time on parallel machines. *Journal of Computer and System Sciences*, 73(6):875–891, 2007.

[9] Minghong Lin, Adam Wierman, and Bert Zwart. Heavy-traffic analysis of mean response time under shortest remaining processing time. *Performance Evaluation*, 68(10):955–966, 2011.

[10] Linus E Schrage and Louis W Miller. The queue m/g/1 with the shortest remaining processing time discipline. *Operations Research*, 14(4):670–684, 1966.

[11] Ziv Scully, Guy Blelloch, Mor Harchol-Balter, and Alan Scheller-Wolf. Optimally scheduling jobs with multiple tasks. *ACM SIGMETRICS Performance Evaluation Review*, 45(2):36–38, 2017.

[12] Yin Sun, C Emre Koksal, and Ness B. Shroff. Near delay-optimal scheduling of batch jobs in multi-server systems. *Ohio State Univ., Tech. Rep*, 2017.

[13] Weina Wang, Kai Zhu, Lei Ying, Jian Tan, and Li Zhang. Maptask scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality. *IEEE/ACM Transactions on Networking*, 24(1):190–203, 2016.

[14] Ronald W Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2):223–231, 1982.

[15] Yousi Zheng, Ness B. Shroff, and Prasun Sinha. A new analytical technique for designing provably efficient mapreduce schedulers. In *INFOCOM*, pages 1600–1608, 2013.

[16] Alan Scheller-Wolf Ziv Scully, Mor Harchol-Balter. Optimal scheduling and exact response time analysis for multistage jobs. 2018.

A Proof of Theorem 4

Proof: Let $n_{\text{M-SRPT}}(t)$ and $n_{\pi^*}(t)$ represent the number of jobs alive at time t under Modified SRPT and optimal scheduler respectively. Without loss of generality, in the following of the proof we assume $\log p_{\max}$ and $\log p_{\min}$ are integers. For $\forall t \geq 0$, the number of unfinished jobs under the optimal algorithm is no less than,

$$
\begin{aligned}
n_{\pi^*}(t) \geq & \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{W_{\pi^*}^{(k)}(t)}{2^k} && (18)\\
= & \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{\left[W_{\pi^*}^{[k]}(t) - W_{\pi^*}^{[k-1]}(t)\right]}{2^k} && \text{(definition of } W_{\pi^*}^{[k]}(t)\text{)}\\
= & \frac{W_{\pi^*}^{[\log p_{\max}+1]}(t)}{2^{\log p_{\max}+1}} + \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{W_{\pi^*}^{[k]}(t)}{2^{k+1}} && (19)\\
\geq & \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{W_{\pi^*}^{[k]}(t)}{2^{k+1}}. && (20)
\end{aligned}
$$

On the other hand, the number of jobs alive under Algorithm 1 can be upper bounded in a similar fashion,

$$
\begin{aligned}
n_{\text{M-SRPT}}(t) \leq & \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{W_{\text{M-SRPT}}^{(k)}(t)}{2^{k-1}}\\
= & \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{\left[W_{\text{M-SRPT}}^{[k]}(t) - W_{\text{M-SRPT}}^{[k-1]}(t)\right]}{2^{k-1}} && \text{(definition of } W_{\text{M-SRPT}}^{[k]}(t)\text{)}\\
= & \sum_{k=\log p_{\min}}^{\log p_{\max}} \frac{W_{\text{M-SRPT}}^{[k]}(t)}{2^k} + \frac{W_{\text{M-SRPT}}^{[\log p_{\max}+1]}(t)}{2^{\log p_{\max}}}\\
\leq & \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{W_{\text{M-SRPT}}^{[k]}(t)}{2^{k-1}}
\end{aligned}
$$

Using Lemma 5, we are able to relate the number of unfinished jobs under two algorithms,

$$\begin{aligned} n_{\text{M-SRPT}}(t) \leq & \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{W^{[k]}_{\text{M-SRPT}}(t)}{2^{k-1}} \\ \leq & \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{W^{[k]}_{\pi^*}(t)}{2^{k-1}} + \sum_{k=\log p_{\min}}^{\log p_{\max}+1} \frac{N \cdot (2^k + \eta)}{2^{k-1}} \\ \leq & 4 n_{\pi^*}(t) + N \cdot \Big(4 \log \alpha + 4 \frac{\eta}{p_{\min}} + 4\Big), \end{aligned}$$

where the last inequality follows from inequality (20). To summarize, the competitive ratio of Algorithm 1 satisfies that

$$\begin{aligned} \mathcal{CR}_{\text{M-SRPT}} = & \frac{\sum_{t: n_{\text{M-SRPT}}(t) < N} n_{\text{M-SRPT}}(t) + \sum_{t: n_{\text{M-SRPT}}(t) \geq N} n_{\text{M-SRPT}}(t)}{F^{\pi^*}} \\ \leq & \frac{\sum_{t: n_{\text{M-SRPT}}(t) < N} n_{\text{M-SRPT}}(t)}{F^{\pi^*}} + \frac{\sum_{t: n_{\text{M-SRPT}}(t) \geq N} 4 n_{\pi^*}(t)}{F^{\pi^*}} \\ & + \Big(4 \log \alpha + 4 \frac{\eta}{p_{\min}} + 4\Big) \cdot \frac{\sum_{t: n_{\text{M-SRPT}}(t) \geq N} N}{F^{\pi^*}} \\ \leq & 4 \log \alpha + 4 \frac{\eta}{p_{\min}} + 8, \end{aligned}$$

where the second inequality is due to Lemma 5. The proof is complete. □

B List of Distributions [9]

- *Weibull distribution.* Weibull distribution has a cumulative distribution function of $F(x) = 1 - e^{-\mu x^\alpha}$, upper Matuszewska index $M_f = -\infty$ and $G^{-1}(\rho) = \Theta((\log(1/(1-\rho))^{1/\alpha})$. Indeed exponential distribution is a special case of the Weilbull distribution with $\alpha = 1$.

- *Pareto distribution.* A power-law job size distribution is often modeled with Pareto distribution, which has a cumulative distribution function of $F(x) = 1 - (x_{\min}/x)^\alpha$ ($\alpha \geq 4$) for $x \geq x_{\min}$. The upper Matuszewska index $M_f = \alpha$ and $\mathbb{E}[F_\rho^{\text{SRPT}-1}] = \Theta(1/(1-\rho)^{\frac{\alpha+2}{\alpha+1}})$.

- *Regularly varying distributions.* More generally, the optimality condition also holds for regularly varying job size distribution RV_α ($\alpha \in (-\infty, -4)$) with cumulative distribution function $F(x) = 1 - L(x) \cdot x^\alpha$, where $L(\cdot)$ is a slowly varying function, *i.e.*, $\lim_{x\to\infty} \frac{L(cx)}{L(x)} = 1$ for any fixed $c > 0$. The upper Matuszewska index of RV_α is equal to α and there exists a slowly varying function $L'(\cdot)$ such that $G^{-1}(\rho) = L'(1/(1-\rho)) \cdot (1-\rho)^{1/(\alpha+1)}$, which implies that $\mathbb{E}[F_\rho^{\text{SRPT}-1}] = \Omega(1/(1-\rho)^{\frac{\alpha+2}{\alpha+1}-\epsilon})$.