Online Scheduling on Identical Machines using SRPT

Kyle Fox[*]   Benjamin Moseley[†]

Abstract

Due to its optimality on a single machine for the problem of minimizing average flow time, Shortest-Remaining-Processing-Time (SRPT) appears to be the most natural algorithm to consider for the problem of minimizing average flow time on multiple identical machines. It is known that SRPT achieves the best possible competitive ratio on multiple machines up to a constant factor. Using resource augmentation, SRPT is known to achieve total flow time at most that of the optimal solution when given machines of speed $2 - \frac{1}{m}$. Further, it is known that SRPT's competitive ratio improves as the speed increases; SRPT is s-speed $\frac{1}{s}$-competitive when $s \geq 2 - \frac{1}{m}$.

However, a gap has persisted in our understanding of SRPT. Before this work, the performance of SRPT was not known when SRPT is given $(1 + \epsilon)$-speed when $0 < \epsilon < 1 - \frac{1}{m}$, even though it has been thought that SRPT is $(1+\epsilon)$-speed $O(1)$-competitive for over a decade. Resolving this question was suggested in Open Problem 2.9 from the survey "Online Scheduling" by Pruhs, Sgall, and Torng [PST04], and we answer the question in this paper. We show that SRPT is *scalable* on m identical machines. That is, we show SRPT is $(1 + \epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive for $\epsilon > 0$. We complement this by showing that SRPT is $(1 + \epsilon)$-speed $O(\frac{1}{\epsilon^2})$-competitive for the objective of minimizing the ℓ_k-norms of flow time on m identical machines. Both of our results rely on new potential functions that capture the structure of SRPT. Our results, combined with previous work, show that SRPT is the best possible online algorithm in essentially every aspect when migration is permissible.

---
[*]Department of Computer Science, University of Illinois, 201 N. Goodwin Ave., Urbana, IL 61801. `kylefox2@illinois.edu`. This work was done while the author was at Google Inc. (Mountain View).

[†]Department of Computer Science, University of Illinois, 201 N. Goodwin Ave., Urbana, IL 61801. `bmosele2@illinois.edu`. This work was done while the author was at Yahoo! Labs (Santa Clara).

1 Introduction

Scheduling jobs that arrive over time is a fundamental problem faced by a variety of systems. In the simplest setting there is a single machine and n jobs that arrive online. A job J_i is released at time r_i where $i \in [n]$. The job has some processing time p_i. This is the amount of time the scheduler must devote to job J_i to complete the job. The goal of the scheduler is to determine which job should be processed at any given time while optimizing a quality of service metric. In the *online* setting the scheduler is not aware of a job until it is released. Thus, an online scheduler must make scheduling decisions without access to the entire problem instance. Having the scheduler be online is desirable in practice since most systems are not aware of the entire jobs sequence in advance.

The most popular quality of service metric considered in online scheduling theory is total flow time, or equivalently, average flow time [PST04]. The flow time[1] of a job is the amount of time it takes the scheduler to satisfy the job. Formally, the flow time of job J_i is $C_i - r_i$ where C_i is the completion time of job J_i. The completion time of a job J_i is defined to be the earliest time t such that the scheduler has devoted p_i units of time to job J_i during $(r_i, t]$. The total flow time of the schedule is $\sum_{i \in [n]} C_i - r_i$. By focusing on minimizing the total flow time, the scheduler minimizes the total time jobs must wait to be satisfied.

On a single machine, the algorithm Shortest-Remaining-Processing-Time (SRPT) always schedules the job whose remaining processing time is the smallest, breaking ties arbitrarily. It is well known that SRPT is optimal for total flow time in this setting. A more complicated scheduling model is where there are m identical machines. Minimizing the flow time in this model has been studied extensively in scheduling theory [LR07, AALR02, CGKK04, AA07, BL04, CKZ01, TM08]. When there is more than one machine the scheduler must not only chose which subset of jobs to schedule, but it must also decide how to distribute jobs across the machines. Naturally, it is assumed that a job can only be processed by one

---
[1]Flow time is also referred to as response time or waiting time.

machine at a time. For this scheduling setting, it is known that there there is a $\Omega(\min\{\log P, \log n/m\})$ lower bound on any online randomized algorithm in the oblivious adversary model [LR07]. Here P is the ratio of maximum processing time to minimum processing time. The algorithm SRPT in the m identical machine setting always schedules the m jobs with least remaining processing time. SRPT has competitive ratio $O(\min\{\log P, \log n/m\})$ for average flow time, making SRPT the best possible algorithm up to a constant factor in the competitive ratio.

The strong lower bound on online algorithms has led previous work to use a resource augmentation analysis. In a *resource augmentation* analysis the adversary is given m unit-speed processors and the algorithm is given m processors of speed s [KP00]. We say that an algorithm is s-speed c-competitive if the algorithm's objective is within a factor of c of the optimal solution's objective when the algorithm is given s resource augmentation. An ideal resource augmentation analysis shows that an algorithm is $(1 + \epsilon)$-speed $O(1)$-competitive for any fixed $\epsilon > 0$. Such an algorithm is called *scalable*. A scalable algorithm is $O(1)$-competitive when given the minimum amount of extra resources over the adversary. Given the strong lower bound on flow time in the identical machines model, finding a scalable algorithm is essentially the best positive result that can be shown using worst case analysis.

Given that SRPT is an optimal algorithm on a single machine and achieves the best possible competitive ratio on multiple machines without resource augmentation, it was widely thought that SRPT would be a scalable algorithm in the multiple machine case. However, the competitive ratio of SRPT when given $1+\epsilon$ speed had been unresolved for about a decade when $0 < \epsilon < 1 - \frac{1}{m}$. Instead, another algorithm was shown to be scalable [CGKK04]. This algorithm geometrically groups jobs according to their size. It uses these groupings to assign each job to exactly one machine. The algorithm then runs the single machine version of SRPT separately on each machine.

Although the competitiveness of SRPT was not known when given speed less than $2 - \frac{1}{m}$, it was known that SRPT achieves total flow time at most that of the optimal solution's flow time when given machines of speed at least $2 - \frac{1}{m}$ [PSTW02]. In fact, this has been extended to show that SRPT is s-speed $\frac{1}{s}$-competitive when $s \geq 2 - \frac{1}{m}$ [TM08]. This result shows that SRPT 'efficiently' uses the faster processors it is given. In the fairly recent online scheduling survey of Pruhs, Sgall, and Torng it was suggested in Open Problem 2.9 that an important question is to resolve whether or not SRPT is a scalable algorithm [PST04]. In this paper we answer this question in the affirmative by showing the following theorem.

THEOREM 1.1. *The algorithm* SRPT *is* $(1+\epsilon)$-*speed* $\frac{4}{\epsilon}$-*competitive for average flow time on m identical parallel machines for $\epsilon > 0$.*

Unfortunately, algorithms which are optimal for average flow time can starve individual jobs of processing power for an arbitrary finite amount of time. For example, suppose we are given a single machine. Jobs J_1 and J_2 arrive at time 0 and at every unit time step another job arrives. All jobs have unit processing time. Using average flow time as the objective, an optimal algorithm for this problem instance is to schedule J_1 and then schedule jobs as they arrive, scheduling J_2 after the last of the other jobs is completed. Although this algorithm is optimal, it can be seen that the algorithm is not 'fair' to job J_2.

Algorithms which are fair at the individual job level are desirable in practice [Tan07, SG94]. In fact, algorithms that are competitive for total flow time are sometimes not implemented due to the possibility of unfairness [BP03]. To overcome the disadvantage of algorithms that merely optimize the average flow time, the objective of minimizing the ℓ_k-norms of flow time for small k was suggested by Bansal and Pruhs [BP03, BP04]. This objective tries to balance overall performance and fairness. Specifically, the ℓ_k-norm objective minimizes $\left(\sum_{i \in [n]} (C_i - r_i)^k \right)^{1/k}$. Notice that optimizing the ℓ_1-norm is equivalent to optimizing the average flow time. For the ℓ_k-norm objective when $k > 1$, the previous example has one optimal solution. This solution schedules jobs in the order they arrive, which can be seen to be 'fair' to each job.

For the ℓ_k-norm objective it is known that every online deterministic algorithm is $n^{\Omega(1)}$-competitive even on a single machine when $1 < k < \infty$ [BP03]. This is quite different from the ℓ_1-norm where SRPT is an optimal algorithm. In the single machine setting, it was shown that SRPT is a scalable algorithm for the ℓ_k-norm objective for all k [BP03]. The competitiveness of SRPT in the multiple machine setting was not known for the ℓ_k-norms even when SRPT is given any constant amount of resource augmentation. The previously discussed algorithm that was analyzed in [CGKK04] was shown to be scalable for the problem of minimizing the ℓ_k-norms of flow time on identical machines for all $k > 1$. It was suggested in [PST04] that determining whether or not SRPT is scalable for the ℓ_k norms of flow time on identical machines is another interesting open question. In this

paper we analyze SRPT and show that it is a scalable algorithm for the ℓ_k-norm objective on multiple machines. This shows that not only is SRPT essentially the best possible algorithm for the objective of average flow time in almost all aspects in the worst case model, SRPT will also balance the fairness of the schedule when given a small amount of resource augmentation.

THEOREM 1.2. *The algorithm* SRPT *is* $(1+\epsilon)$-*speed* $\frac{4}{\epsilon^2}$-*competitive for the* ℓ_k-*norms of flow time on* m *identical parallel machines for* $k \geq 1, 1/2 \geq \epsilon > 0$.

To prove both of these results, we introduce novel potential functions that we feel capture the structure of SRPT. SRPT is a natural algorithm to consider in many other scheduling models where potential function analysis is commonly found. We believe that the potential functions introduced here will be useful for analyzing SRPT and similar algorithms in these other settings.

Related Work: As mentioned, SRPT is an optimal algorithm for minimizing average flow time on a single machine. SRPT was the first algorithm to be analyzed in the worst case model when there are m identical machines. It was shown by Leonadi and Raz that SRPT is $O(\min\{\log P, \log n/m\})$-competitive and there is a matching lower bound on any randomized algorithm [LR07]. A simpler analysis of SRPT in the multiple machine setting can be found in [Leo03]. SRPT is $\left(2 - \frac{1}{m}\right)$-speed 1-competitive and SRPT is the only algorithm known to be 1-competitive with any resource augmentation in the multiple machine setting. Notice that SRPT in the multiple machine setting could schedule a job on one machine and then later schedule the job on another machine. That is, SRPT *migrates* jobs between the machines. To eliminate migration Awerbuch et al. introduced an algorithm that processes each job on exactly one machine and showed that this algorithm is $O(\min\{\log P, \log n\})$-competitive [AALR02]. A related algorithm was developed by Chekuri, Khanna, and Zhu that does not migrate jobs and it was shown to be $O(\min\{\log P, \log n/m\})$-competitive [CKZ01]. Each of the previously discussed algorithms hold the jobs in a central pool until they are scheduled. Avrahami and Azar introduced an algorithm which does not hold jobs in a central pool, but rather assigns a job to a unique machine as soon as the job arrives [AA07]. They showed that their algorithm is $O(\min\{\log P, \log n\})$-competitive. Chekuri et al. showed that the algorithm of Avrahami and Azar is a scalable algorithm [AA07, CGKK04]. For the ℓ_k-norms of flow time Chekuri et al. also showed

that the algorithm of Avrahami and Azar is scalable [CGKK04].

The analysis in [CGKK04], which shows a scalable algorithm for average flow time on multiple machines, uses a local competitiveness argument. In a local argument, it is shown that at any time, the increase in the algorithm's objective function is bounded by a constant factor of the optimal solution's objective. From the lower bound given above, we know this property does not hold when SRPT is not given resource augmentation. With resource augmentation, it is unclear whether or not this can be shown for SRPT on every input. In this paper, we avoid a local analysis by using a potential function argument which we discuss further in the following section.

2 Preliminaries

Before giving our analysis, we introduce a fair bit of notation. Let $\mathcal{Q}^S(t)$ and $\mathcal{Q}^O(t)$ be the set of jobs alive (released but unsatisfied) at time t in SRPT's and OPT's schedules, respectively. Let $\mathcal{W}^S(t)$ be the set of jobs scheduled for processing at time t in SRPT's schedule. Let $p_i^S(t)$ and $p_i^O(t)$ be the remaining processing times at time t for job J_i in SRPT's and OPT's schedules, respectively. Finally, let C_i^S and C_i^O be the completion time of job J_i in SRPT's and OPT's schedules, respectively.

Throughout this paper, we will concentrate on bounding SRPT's *kth power flow time*, $\sum_{i \in [n]} \left(C_i^S - r_i\right)^k$, as this is the ℓ_k-norm of flow time without the outer root. We will proceed to use SRPT and OPT as functions of t that return their respective algorithm's accumulated kth power flow time. In other words, $\mathrm{SRPT}(t) = \sum_{i \in [n], t \geq r_i} (\min\{C_i^S, t\} - r_i)^k$, and $\mathrm{OPT}(t)$ is defined similarly. When SRPT or OPT is used as a value without a time specified, it is assumed we mean their final objective value.

For any job J_i and time t, we let $R^S(i, t)$ be the total volume of work remaining at time t for every released job with completion time at most C_i^S in SRPT's schedule. Precisely,

$$R^S(i, t) = \sum_{J_j \in \mathcal{Q}^S(t), C_j^S \leq C_i^S} p_j^S(t).$$

We also define $V^O(i, t)$ to be the volume of work in OPT's schedule at time t for a subset of those same jobs, except we only include those jobs with original processing time at most p_i. Precisely,

$$V^O(i, t) = \sum_{J_j \in \mathcal{Q}^O(t), C_j^S \leq C_i^S, p_j \leq p_i} p_j^O(t).$$

We will assume without loss of generality that all arrival and completion times are distinct by breaking ties arbitrarily but consistently.

The following lemma will help us to characterize the current status of SRPT compared to OPT at any point in time. This is a modification of a lemma given in [MRSG04, PST04].

LEMMA 2.1. *At any time $t \geq r_i$, for any sequence of requests σ, and for any $i \in [n]$, it is the case that $R^S(i,t) - V^O(i,t) \leq mp_i$.*

Proof. Define $X(i,t)$ to be the sum of the remaining processing times in SRPT's schedule at time t for jobs with remaining processing time at most p_i while also contributing to $R^S(i,t)$. In other words,

$$X(i,t) = \sum_{J_j \in \mathcal{Q}^S(t), C_j^S \leq C_i^S, p_j^S(t) \leq p_i} p_j^S(t).$$

Every job contributing to $R^S(i,t)$ must have remaining processing time at most p_i in order for SRPT to schedule it ahead of J_i, so we see $X(i,t) = R^S(i,t)$ whenever $t \geq r_i$. Thus is suffices to show that $X(i,t) - V^O(i,t) \leq mp_i$. If there are m or fewer jobs contributing to $X(i,t)$ at time t in $\mathcal{Q}^S(t)$ then the lemma follows easily. Now consider the case where there are more than m jobs contributing to $X(i,t)$.

Let $t' \geq 0$ be the earliest time before time t such that SRPT always had at least m jobs contributing to $X(i,t)$ during $(t',t]$. We will show $X(i,t) - V^O(i,t) \leq mp_i$. Let $T = \sum_{r_j \in (t',t], C_j^S \leq C_i^S, p_j \leq p_i} p_j$ be the total processing time of jobs that arrive during $(t',t]$ that are completed by SRPT before J_i and have original processing time at most p_i. It can be seen that X will increase by T during $(t',t]$ due to the arrival of jobs. However, V^O will also increase by T during $(t',t]$ by definition of V^O.

The only other change that occurs to X and V^O during $(t',t]$ is due to the processing of jobs by the algorithm SRPT and OPT. Knowing that OPT has m machines of unit speed, V^O can decrease by at most $m(t-t')$ during (t,t'). We also know that during $(t',t]$, there always exists at least m jobs with remaining processing time at most p_i unsatisfied by SRPT that will be completed by SRPT before job J_i. SRPT always works on the m available jobs with earliest completion time, so this causes X to decrease by at least $m(t'-t)$ (this even assumes SRPT is not given resource augmentation). Combining these facts we have the following:

$$X(i,t) - V^O(i,t) \leq (X(i,t') + T - m(t'-t))$$
$$- \left(V^O(i,t') + T - m(t'-t) \right)$$
$$= X(i,t') - V^O(i,t')$$
$$\leq mp_i$$

2.1 Potential Function Analysis For our proofs of the theorems, we will use a potential function argument [Edm00]. In each proof we will define a potential function $\Phi : [0,\infty) \to \mathbb{R}$ such that $\Phi(0) = \Phi(\infty) = 0$. We will proceed to bound discrete and continuous local changes to $\text{SRPT} + \Phi$. These changes may come from the following sources:

Job Arrival: Arriving jobs will not affect SRPT but they will make a change to Φ. The total increase in Φ over all jobs arrivals will be bounded by δOPT where δ is a non-negative constant which may depend on k and ϵ.

Job Completion: Again, job completions will not affect SRPT, but they will make a change to Φ. We will bound these increases by γOPT where γ is a non-negative constant which may depend on k and ϵ.

Running Condition: This essentially captures everything else. We will show a bound on the continuous changes in $\text{SRPT} + \Phi$ due to the change in time as well as the changes to each job's remaining processing time. Surprisingly, we find $\frac{d}{dt}\text{SRPT} + \frac{d}{dt}\Phi \leq 0$, meaning we can ignore the running condition in our final calculations.

Knowing that $\Phi(\infty) = \Phi(0) = 0$, we have that $\text{SRPT} = \text{SRPT}(\infty) + \Phi(\infty)$. This is bounded by the total increase in the arrival and completion conditions, thus we will have $\text{SRPT} \leq (\delta + \gamma)\text{OPT}$, which will complete our analysis.

3 Total Flow Time

We assume SRPT is given $(1+\epsilon)$ speed where $\epsilon > 0$. We proceed by placing our focus on minimizing the total flow time. To accomplish this, we will define a potential function with one term for each job being processed such that the following conditions are met:

- Job arrivals and completions do not increase the potential function beyond a strong lower bound on OPT.

- Each term has a decreasing component that counteracts the gradual increases in SRPT's flow time.

- There may be components of each term that increase, but we can easily bound these increases by the decreases from other components.

We use the following potential function based on the intuition given above:

$$\Phi(t) = \frac{1}{m\epsilon} \sum_{J_i \in \mathcal{Q}^S(t)} \left(R^S(i,t) + mp_i^S(t) - V^O(i,t) \right)$$

Now, consider the different changes that occur to SRPT's accumulated flow time as well as Φ for any job sequence σ.

Job Arrival: The event of a job's arrival makes no change to the accumulated flow time, but it can change Φ. Consider the arrival of job J_i at time $t = r_i$. For any $j \neq i$ such that $J_j \in \mathcal{Q}^S(t)$, consider the term

$$\frac{1}{m\epsilon} \left(R^S(j,t) + mp_j^S(t) - V^O(j,t) \right)$$

in the potential function. The arrival of job J_i will change both $R^S(j,t)$ and $V^O(j,t)$ equally (either by p_i or 0 depending on if $p_j \leq p_i^S(t)$) creating no net change in the potential function. We do gain a new term in the summation, but this can be bounded as follows:

$$\frac{1}{m\epsilon} \left(R^S(i,t) + mp_i - V^O(i,t) \right)$$
$$\leq \frac{1}{m\epsilon}(2mp_i) \quad \text{By Lemma 2.1}$$
$$= \frac{2}{\epsilon}p_i$$

We use the trivial lower bound of p_i on J_i's total flow time to see that the total increase in Φ from job arrivals is at most $\frac{2}{\epsilon}$OPT.

Job Completion: Same as above, job completions make no change to the accumulated flow time. Consider the completion of a job J_i by OPT at time $t = C_i^O$. For any job $J_j \in \mathcal{Q}^S(t)$, the term

$$\frac{1}{m\epsilon} \left(R^S(j,t) + mp_j^S(t) - V^O(j,t) \right)$$

sees no change as J_i is already contributing nothing to $V^O(j,t)$.

Likewise, consider the completion of job J_i by SRPT at time $t = C_i^S$. For any $j \neq i$ such that $J_j \in \mathcal{Q}^S(t)$, the term

$$\frac{1}{m\epsilon} \left(R^S(j,t) + mp_j^S(t) - V^O(j,t) \right)$$

sees no change as J_i is already contributing nothing to $R^S(j,t)$. Unfortunately, we need a more sophisticated argument to bound the increase in Φ from removing the term

$$\frac{1}{m\epsilon} \left(R^S(i,t) + mp_i^S(t) - V^O(i,t) \right).$$

The increase from removing this term is precisely $\frac{1}{m\epsilon}V^O(i,t)$, because SRPT has completed all jobs contributing to $R^S(i,t)$ and $p_i^S(t) = 0$. We can use the following scheme to charge this and similar increases to OPT's total flow time. Consider any job J_j contributing volume to $V^O(i,t)$. We know that if $r_j < r_i$, we have $p_j \leq p_i$ by definition of V^O. Further, if $r_j \geq r_i$, we have $p_j \leq p_i^S(r_j)$ by definition of R^S. In either case, SRPT performs at least p_j units of work on job J_i while J_j is sitting in OPT's queue, and this work occurs over a period of at least $p_j/(1+\epsilon)$ time units. To pay for J_j's contribution to $\frac{1}{m\epsilon}V^O(i,t)$, we charge to J_j's increase in flow time during this period at a rate of $\frac{1+\epsilon}{m\epsilon}$.

The total charge accrued during this period due to J_j is at least $\frac{1+\epsilon}{m\epsilon}\frac{p_j}{1+\epsilon} = \frac{p_j}{m\epsilon}$. Summing over all jobs contributing to $V^O(i,t)$, we see that we charge enough. Now we need to bound our total charge. Observe that any one of these charges to a job J_j accrues at $\frac{1+\epsilon}{m\epsilon}$ times the rate that J_j is accumulating flow time. Further, SRPT is working on at most m jobs at any point in time, so our combined charges are accruing at $\frac{1+\epsilon}{\epsilon}$ times the rate that J_j is accumulating flow time. By summing over all time and jobs, we conclude that we charge at most $\frac{1+\epsilon}{\epsilon}$OPT, giving us an upper bound on $\Phi's$ increase due to SRPT's job completions.

Running Condition: We now proceed to show a bound on $\frac{\mathrm{d}}{\mathrm{dt}}$SRPT $+ \frac{\mathrm{d}}{\mathrm{dt}}\Phi$ at an arbitrary time t ignoring the arrival and completion of jobs. First, note that

$$\frac{\mathrm{d}}{\mathrm{dt}}\text{SRPT} = \sum_{J_i \in \mathcal{Q}^S(t)} 1.$$

To bound $\frac{\mathrm{d}}{\mathrm{dt}}\Phi$, we fix some i such that $J_i \in \mathcal{Q}^S(t)$ and consider J_i's term in Φ's summation.

We begin by considering the change due to $V^O(i,t)$. OPT can only process m jobs at a time, so the ith term of Φ changes at a rate of at most

$$\frac{1}{m\epsilon}m = \frac{1}{\epsilon}.$$

Finally, we consider the change due to both $R^S(i,t)$ and $mp_i^S(t)$ together and derive a lower bound on their combined decrease. Neither term can increase, so we accomplish this by finding a lower bound on the decrease of one or the other.

Suppose SRPT is processing job J_i (using $(1 + \epsilon)$ speed) at time t. If this is the case, $mp_i^S(t)$ decreases at a rate of $m(1+\epsilon)$. If job J_i is not being processed, then there are m other jobs in $\mathcal{W}^S(t)$ being processed instead. By definition, these jobs are contributing their volume to $R^S(i,t)$, and we see it decreases at a rate of $m(1 + \epsilon)$. Considering both terms together, we find an upper bound for their contribution to Φ's rate of change which is

$$\frac{1}{m\epsilon}\left(-m(1+\epsilon)\right) = -\frac{1}{\epsilon} - 1.$$

By summing over the above rates of change, we see everything cancels out to 0. Summing over all jobs gives us $\frac{d}{dt}\text{SRPT}(t) + \frac{d}{dt}\Phi(t) \leq 0$. Integrating the left hand side over all time, we see SRPT and Φ together *do not increase* if we only consider events other than the arrival and completion of jobs.

Final Analysis: Using the framework described in Section 2 and the above analysis, we see SRPT $\leq \frac{4}{\epsilon}$OPT. This concludes the proof of Theorem 1.1.

4 ℓ_k-Norms of Flow Time

In this section we focus on minimizing the ℓ_k-norms of flow time. Consider any job sequence σ and assume that SRPT is given $(1+\epsilon)$-speed where $1/2 \geq \epsilon > 0$. We use a somewhat different potential function that includes extra components meant to reflect the increasing speed at which alive jobs contribute to kth power flow time. We use the following potential function to directly bound SRPT's kth power flow time:

$$\Phi(t) = \frac{1}{(1-\epsilon)^k}\sum_{i\in\mathcal{Q}^S(t)}\left(\max\left\{t - r_i + \right.\right.$$
$$\left.\left.\frac{1}{m\epsilon}\left(R^S(i,t) + mp_i^S(t) - V^O(i,t)\right), 0\right\}\right)^k$$
$$-\sum_{i\in\mathcal{Q}^S(t)}(t-r_i)^k$$

Job Arrival: Consider the arrival of job J_i at time $t = r_i$. Again, no change occurs to the objective function. Also, as in the case for standard flow time, no change will occur to the J_jth term of the potential function for any $j \neq i$. However, a new term is added to the summation in the potential function. The increase in Φ due to this new term is at most

$$\frac{1}{(1-\epsilon)^k}\left(\max\left\{\frac{1}{m\epsilon}\left(R^S(i,t) + mp_i - V^O(i,t)\right), 0\right\}\right)^k$$
$$\leq \frac{1}{(1-\epsilon)^k}\left(\frac{1}{m\epsilon}(2mp_i)\right)^k \quad \text{By Lemma 2.1}$$
$$\leq \left(\frac{2}{\epsilon(1-\epsilon)}\right)^k(p_i)^k.$$

The value $(p_i)^k$ is a trivial lower bound on J_i's kth power flow time, so we can bound the total increase in Φ due to job arrivals by $\left(\frac{2}{\epsilon(1-\epsilon)}\right)^k$ OPT.

Job Completion: Again, the only effect of job completion we are concerned with is the increase of each job J_i's term in Φ when SRPT completes J_i at time $t = C_i^S$. The increase from this occurrence is

$$(t-r_i)^k - \frac{1}{(1-\epsilon)^k}\left(\max\left\{t - r_i + \frac{1}{m\epsilon}V^O(i,t), 0\right\}\right)^k$$

We will use the following lemmas.

LEMMA 4.1. *For any job $J_i \in \mathcal{Q}^S(t)$, if $V^O(i,t) \leq m\epsilon^2(t - r_i)$ then*

$$(t-r_i)^k - \frac{1}{(1-\epsilon)^k}\left(\max\left\{t - r_i + \frac{1}{m\epsilon}V^O(i,t), 0\right\}\right)^k \leq 0.$$

Proof. Note that hypothesis cannot apply when $t - r_i + \frac{1}{m\epsilon}\left(R^S(i,t) + mp_i^S(t) - V^O(i,t)\right) < 0$. This is because

$$t - r_i + \frac{1}{m\epsilon}\left(R^S(i,t) + mp_i^S(t) - V^O(i,t)\right) \geq (1-\epsilon)(t-r_i)$$

which is non-negative for all $t \geq r_i, \epsilon \leq 1$. Given the assumption that $V^O(i,t) \leq m\epsilon^2(t - r_i)$, we have

$$(t - r_i)^k - \frac{1}{(1-\epsilon)^k}\left(t - r_i + \frac{1}{m\epsilon}V^O(i,t)\right)^k$$
$$\leq (t - r_i)^k - \frac{1}{(1-\epsilon)^k}\left((1-\epsilon)(t-r_i)\right)^k$$
$$= 0.$$

LEMMA 4.2. *For any job $J_i \in \mathcal{Q}^S(t)$, if $V^O(i,t) > m\epsilon^2(t - r_i)$ then*

$$(t - r_i)^k - \frac{1}{(1-\epsilon)^k}\left(\max\left\{t - r_i + \frac{1}{m\epsilon}V^O(i,t), 0\right\}\right)^k$$
$$\leq \left(\frac{1}{\epsilon^2}\right)^k\left(\frac{1}{m}V^O(i,t)\right)^k.$$

Proof. We will ignore the negative term from the expression. Given the assumption that $V^O(i,t) > m\epsilon^2(t-r_i)$, we have

$$(t-r_i)^k \leq \left(\frac{1}{m\epsilon^2}V^O(i,t)\right)^k$$
$$= \left(\frac{1}{\epsilon^2}\right)^k \left(\frac{1}{m}V^O(i,t)\right)^k.$$

Based on these lemmas, we see the total increase to Φ from job completions is bounded by

$$\sum_{i\in[n]} \left(\frac{1}{\epsilon^2}\right)^k \left(\frac{1}{m}V^O(i,C_i^S)\right)^k.$$

The following lemma, which we will prove later, implies that this bound is at most $\left(\frac{1+\epsilon}{\epsilon^2}\right)^k\mathrm{OPT}$.

LEMMA 4.3. *We have*

$$\sum_{i\in[n]} \left(\frac{1}{m}V^O(i,C_i^S)\right)^k \leq (1+\epsilon)^k\mathrm{OPT}.$$

Running Condition: We now ignore the arrival and completion of jobs and consider the change in the kth power flow time as well as Φ due to other events. Consider any time t. Note that

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{SRPT}(t) = \sum_{J_i\in\mathcal{Q}^S(t)} k\cdot(t-r_i)^{k-1}.$$

Now, fix some i such that $J_i \in \mathcal{Q}^S(t)$. We will examine the contribution of the J_ith term to $\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{SRPT}$. We will begin by assuming $t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right) > 0$ and consider the other case later.

First, consider how the change in t affects this term while keeping the dependent variables fixed. The rate of change is at most

$$\frac{k}{(1-\epsilon)^k}\left(t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right)\right)^{k-1}$$
$$-k(t-r_i)^{k-1}.$$

Next we consider the change due to $V^O(i,t)$. In the worst case, OPT works on m jobs at time t so the rate of increase in Φ due to the change in $V^O(i,t)$ is at most

$$\frac{km}{m\epsilon}\frac{1}{(1-\epsilon)^k}\left(t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right)\right)^{k-1}$$
$$= \frac{k}{\epsilon(1-\epsilon)^k}\left(t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right)\right)^{k-1}$$

Now consider the change in Φ due to $R^S(i,t)+mp_i^S(t)$. As in the average flow time argument, this sum decreases at a rate of at least $(1+\epsilon)m$, so these terms cause Φ to change at a rate of at most

$$-\frac{k(1+\epsilon)m}{m\epsilon(1-\epsilon)^k}\left(t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right)\right)^{k-1}$$
$$= -\frac{k}{\epsilon(1-\epsilon)^k}\left(t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right)\right)^{k-1}$$
$$-\frac{k}{(1-\epsilon)^k}\left(t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right)\right)^{k-1}.$$

Summing over the above terms shows that J_i contributes at most 0 to $\mathrm{SRPT}+\Phi$'s rate of change.

We have yet to consider the case when $t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right) \leq 0$. The above arguments concerning the running condition and job arrivals show this term to be non-increasing. Further, we see that once the expression

$$\max\left\{t-r_i + \frac{1}{m\epsilon}\left(R^S(i,t)+mp_i^S(t)-V^O(i,t)\right), 0\right\}$$

hits 0, it will never leave that value.

We now consider the various sources of change in Φ's ith term when the above expression equals 0 by simply plugging 0 into the above inequalities. Changes in t contribute at most $-k(t-r_i)^{k-1}$. Also, changes in $R^S(i,t)$, $mp_i^S(t)$, and $V^O(i,t)$ have no effect. Summing, we still get 0 as an upper bound on J_i's contribution to $\mathrm{SRPT}+\Phi$'s rate of change. Summing over all jobs and integrating over time, we see this bound holds for the running condition's overall contribution to $\mathrm{SRPT}+\Phi$.

Final Analysis: Using the framework discussed in Section 2 and the arrival, completion, and running conditions shown in this section, we have that

$$\mathrm{SRPT} \leq \left(\left(\frac{2}{\epsilon(1-\epsilon)}\right)^k + \left(\frac{1+\epsilon}{\epsilon^2}\right)^k\right)\mathrm{OPT}.$$

By taking the outer kth root of the ℓ_k-norm flow time and assuming $\epsilon \leq 1/2$, we derive Theorem 1.2.

5 Proof of Lemma 4.3

In this section, we prove Lemma 4.3. Namely, if SRPT is running m machines of speed $(1+\epsilon)$ while OPT is running m machines of unit speed, we have

$$\sum_{i\in[n]} \left(\frac{1}{m}V^O(i,C_i^S)\right)^k \leq (1+\epsilon)^k\mathrm{OPT}$$

for the metric of kth power flow time. We will use a charging scheme to prove the lemma.

Fix some job J_i and let $\mathcal{S}_i$ denote the set of jobs that contribute to $V^O(i, C_i^S)$. We charge the following to each $J_j \in \mathcal{S}_i$:

$$\left(\frac{1}{m} \left(V^O(i, C_i^S) - \sum_{J_a \in \mathcal{S}_i, r_a < r_j} p_a^O(C_i^S) \right) \right)^k$$

$$- \left(\frac{1}{m} \left(V^O(i, C_i^S) - p_j^O(C_i^S) - \sum_{J_a \in \mathcal{S}_i, r_a < r_j} p_a^O(C_i^S) \right) \right)^k$$

By considering the jobs in $\mathcal{S}_i$ in order of increasing arrival time, we see the charges form a telescoping sum that evaluates to

$$\left(\frac{1}{m} V^O(i, C_i^S) \right)^k - \left(\frac{1}{m} \left(V^O(i, C_i^S) - \sum_{J_j \in \mathcal{S}_i} p_j^O(C_i^S) \right) \right)^k$$

$$= \left(\frac{1}{m} V^O(i, C_i^S) \right)^k.$$

Now our goal is to show that we charge at most $(1 + \epsilon)^k (C_j^O - r_j)^k$ in total to any job J_j. Let $\mathcal{T}_j = \{J_i \mid J_j \in \mathcal{S}_i\}$, the set of jobs whose completion causes us to charge some amount to J_j. Consider the charge on J_j due to the completion of $J_i \in \mathcal{T}_j$.

LEMMA 5.1. *We have*

$$\frac{1}{(1+\epsilon)m} \left(V^O(i, C_i^S) - \sum_{J_a \in \mathcal{S}_i, r_a < r_j} p_a^O(C_i^S) \right)$$

$$\leq \quad C_j^O - r_j - \frac{1}{(1+\epsilon)m} \sum_{J_a \in \mathcal{T}_j, C_a^S > C_i^S} p_j^O(C_a^O).$$

Proof. We will account for work done by SRPT during $[r_j, C_j^O]$ in two stages and use the result to derive the inequality. First, consider any job $J_a \in \mathcal{T}_j$ with $C_a^S > C_i^S$. We know SRPT gave higher priority to J_j than J_a, because J_j is included in $V^O(a, C_a^S)$. As seen in the completion condition arguments for total flow time, we know SRPT did p_j volume of work on job J_a during $[r_j, C_a^S]$. Namely, we have $p_j \leq p_a$ when $r_j < r_a$ and $p_j \leq p_a^S(r_j)$ when $r_j \geq r_a$ by definition of $V^O(i, C_i^S)$. Therefore, we have at least $\sum_{J_a \in \mathcal{T}_j, C_a^S > C_i^S} p_j \geq \sum_{J_a \in \mathcal{T}, C_a^S > C_i^S} p_j^O(C_a^O)$ volume of work done by SRPT during $[r_j, C_j^O]$.

Next, we note that an additional $V^O(i, C_i^S) - \sum_{J_a \in \mathcal{S}_i, r_a < r_j} p_a^O(C_i^S)$ volume of work must be completed by SRPT during $[r_j, C_j^O]$. This is because SRPT completed the jobs being counted in the above expression by time $C_i^S \leq C_j^O$ and these jobs arrived after time r_j. Further, we are not counting the work in the above paragraph a second time, because no job J_a with $C_a^S > C_i^S$ can count toward $R^S(i, C_i^S)$ or $V^O(i, C_i^S)$ by definition of R^S and V^O.

We know SRPT has m machines of speed $1+\epsilon$, so the soonest SRPT can complete the above mentioned work is

$$r_j + \frac{1}{(1+\epsilon)m} \left(\sum_{J_a \in \mathcal{T}, C_a^S > C_i^S} p_j^O(C_a^O) + V^O(i, C_i^S) \right.$$

$$\left. - \sum_{J_a \in \mathcal{S}_i, r_a < r_j} p_a^O(C_i^S) \right).$$

This expression is at most C_j^O. The lemma follows by simple algebra.

Now we are ready to prove a bound on the amount charged to J_j. The total amount charged is

$$\sum_{J_i \in \mathcal{T}_j} \left\{ \left(\frac{1}{m} \left(V^O(i, C_i^S) - \sum_{J_a \in \mathcal{S}_i, r_a < r_j} p_a^O(C_i^S) \right) \right)^k \right.$$

$$\left. - \left(\frac{1}{m} \left(V^O(i, C_i^S) - p_j^O(C_i^S) - \sum_{J_a \in \mathcal{S}_i, r_a < r_j} p_a^O(C_i^S) \right) \right)^k \right\}.$$

Using Lemma 5.1 and the convexity of x^k for $k \geq 1$ (where x is any positive number), we can upper bound this by

$$\sum_{J_i \in \mathcal{T}_j} \left[\left((1+\epsilon)(C_j^O - r_j) - \frac{1}{m} \sum_{J_a \in \mathcal{T}_j, C_a^S > C_i^S} p_j^O(C_a^O) \right)^k \right.$$

$$\left. - \left((1+\epsilon)(C_j^O - r_j) - \frac{1}{m} \left(-p_j^O(C_i^S) - \sum_{J_a \in \mathcal{T}_j, C_a^S > C_i^S} p_j^O(C_a^O) \right) \right)^k \right].$$

Again, it can be seen that this is a telescoping sum by considering terms in order of decreasing completion time. By the arguments given in the proof of Lemma 5.1, we see

$$\frac{1}{(1+\epsilon)m} \sum_{J_a \in \mathcal{T}_j} p_j^O(C_a^O) \leq C_j^O - r_j,$$

giving us a lower bound of 0 for the last negative term in the telescoping sum. Therefore, the total charged to J_j is at most $\left((1+\epsilon)(C_j^O - r_j) \right)^k$. Summing over all jobs, we see the total amount charged is at most

$$\sum_{j \in [n]} (1+\epsilon)^k (C_j^O - r_j)^k = (1+\epsilon)^k \text{OPT},$$

which implies the lemma.

6 Conclusion

We have shown SRPT to be $(1 + \epsilon)$-speed $O(1)$-competitive for both average flow time and further

127

for the ℓ_k-norms of flow time on m identical machines. This combined with previous work shows that SRPT is the best possible algorithm in many aspects for scheduling on m identical machines. It is known that SRPT is $(2 - \frac{1}{m})$-speed 1-competitive on multiple machines . Further, it is known that no $(\frac{22}{21} - \epsilon)$-speed online algorithm is 1-competitive [PSTW02]. It remains an interesting open question to determine the minimum speed needed for an algorithm for be 1-competitive on m identical machines.

References

[AA07] Nir Avrahami and Yossi Azar. Minimizing total flow time and total completion time with immediate dispatching. *Algorithmica*, 47(3):253–268, 2007.

[AALR02] Baruch Awerbuch, Yossi Azar, Stefano Leonardi, and Oded Regev. Minimizing the flow time without migration. *SIAM J. Comput.*, 31(5):1370–1382, 2002.

[BL04] Luca Becchetti and Stefano Leonardi. Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *J. ACM*, 51(4):517–539, 2004.

[BP03] Nikhil Bansal and Kirk Pruhs. Server scheduling in the l$_p$ norm: a rising tide lifts all boat. In *STOC*, pages 242–250, 2003.

[BP04] Nikhil Bansal and Kirk Pruhs. Server scheduling in the weighted l$_p$ norm. In Martin Farach-Colton, editor, *LATIN*, volume 2976 of *Lecture Notes in Computer Science*, pages 434–443, 2004.

[CGKK04] Chandra Chekuri, Ashish Goel, Sanjeev Khanna, and Amit Kumar. Multi-processor scheduling to minimize flow time with epsilon resource augmentation. In László Babai, editor, *STOC*, pages 363–372, 2004.

[CKZ01] Chandra Chekuri, Sanjeev Khanna, and An Zhu. Algorithms for minimizing weighted flow time. In *STOC*, pages 84–93, 2001.

[Edm00] Jeff Edmonds. Scheduling in the dark. *Theor. Comput. Sci.*, 235(1):109–141, 2000.

[KP00] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

[Leo03] Stefano Leonardi. A simpler proof of preemptive total flow time approximation on parallel machines. *Lecture Notes in Computer Science*, pages 71–77, 2003.

[LR07] Stefano Leonardi and Danny Raz. Approximating total flow time on parallel machines. *J. Comput. Syst. Sci.*, 73(6):875–891, 2007.

[MRSG04] S. Muthukrishnan, Rajmohan Rajaraman, Anthony Shaheen, and Johannes Gehrke. Online scheduling to minimize average stretch. *SIAM J. Comput.*, 34(2):433–452, 2004.

[PST04] Kirk Pruhs, Jiri Sgall, and Eric Torng. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter Online Scheduling. 2004.

[PSTW02] Cynthia A. Phillips, Clifford Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.

[SG94] Abraham Silberschatz and Peter Galvin. *Operating System Concepts, 4th edition*. Addison-Wesley, 1994.

[Tan07] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2007.

[TM08] Eric Torng and Jason McCullough. Srpt optimally utilizes faster machines to minimize flow time. *ACM Transactions on Algorithms*, 5(1), 2008.