ONLINE SCHEDULING ON IDENTICAL MACHINES USING SRPT

BY

KYLE J. FOX

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Professor Jeff Erickson

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, we did not know the performance of SRPT when given machines of speed $1 + \epsilon$ for any $0 < \epsilon < 1 - \frac{1}{m}$.

We answer the question in this thesis. 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 any $\epsilon > 0$. We also show 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.

To my parents, Jan and Lonnie Fox.

Acknowledgements

The author would like to extend thanks to the following people: Benjamin Moseley for his advice and assistance as a coauthor on the preliminary version of this work; the anonymous reviewers of the preliminary version for their writing tips and suggestions; and Chandra Chekuri and Jeff Erickson for their advice in preparing and disseminating the results in this work. The preliminary version of this work appears in [FM11].

This research was supported in part by an award from the Department of Energy (DOE) Office of Science Graduate Fellowship Program (DOE SCGF). The DOE SCGF Program was made possible in part by the American Recovery and Reinvestment Act of 2009. The DOE SCGF program is administered by the Oak Ridge Institute for Science and Education for the DOE. ORISE is managed by Oak Ridge Associated Universities (ORAU) under DOE contract number DE-AC05-06OR23100. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DOE, ORAU, or ORISE. This work was performed while the author was at Google Inc. in Mountain View, CA.

Table of Contents

Chapter 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. Jobs are identified by positive integers between 1 and n. Each job i is released at time r_i and requires processing time p_i. 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. Online schedules are desirable in practice since most systems are not aware of the entire job sequence in advance.

We call a scheduling algorithm *optimal* for some metric if it optimizes the metric for every given set of jobs. While it may seem natural to seek an optimal algorithm for any given metric, it is not always possible. In the online setting, a lack of knowledge about the future may cause any given algorithm to make decisions that hurt its performance on certain inputs. Even if we do not require online algorithms, it may still be computationally intractable to compute optimal schedules.

For these reasons, we often seek algorithms with a good *competitive ratio*. Fix some performance metric $\mathcal{M}$. For some scheduling algorithm $\mathcal{A}$ and set of jobs σ, let $\mathcal{A}(\sigma)$ be the value of the metric $\mathcal{M}$ on $\mathcal{A}$'s schedule for σ. Let $\text{OPT}(\sigma)$ be the optimal value of $\mathcal{M}$ on any schedule for σ. Assuming we are attempting to minimize $\mathcal{M}$, we say $\mathcal{A}$ is *c-competitive* or has competitive ratio c if $\mathcal{A}(\sigma) \leq c \cdot \text{OPT}(\sigma)$ for every input σ. Ideally, c should be a constant, but we may make it a function of the input sequence if necessary. Algorithms with small competitive ratio should perform reasonably well for any given input.

The most popular 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 i is $C_i - r_i$ where C_i is the completion time of job i. The completion time of job i is defined to be the earliest time t such that the scheduler has devoted p_i units of time to job i during $(r_i, t]$. The total flow time of the schedule is $\sum_{i \in [n]} C_i - r_i$. By minimizing the total flow time, the scheduler also minimizes the total time jobs must wait to be satisfied.

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

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]. In this setting, 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 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 the competitive ratio of any online randomized algorithm in the *oblivious adversary* model [LR07], where P is the ratio of maximum processing time to minimum processing time. In the oblivious adversary model, we are allowed to consider worst-case input for randomized algorithms without knowing the results of the randomization in advance. 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 online algorithm up to a constant factor in the competitive ratio [LR07, Leo03].

The strong lower bound on online algorithms has led previous work to use resource augmentation analysis. In *resource augmentation* analysis the adversary is given m unit-speed processors and the algorithm is given m processors of speed $s > 1$ [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 offline solution's objective when the algorithm is given s resource augmentation. In the multiple machine setting, the best possible 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 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 scalable 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, and 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 is $(2-\frac{1}{m})$-speed 1-competitive [PSTW02]. In fact, this result has been extended to show that SRPT is s-speed $\frac{1}{s}$-competitive when $s \geq 2 - \frac{1}{m}$ [TM08]. These results show that SRPT 'efficiently' uses the faster processors it is given. The fairly recent online scheduling survey of Pruhs, Sgall, and Torng posed the scalability of SRPT as an important open problem [PST04, Open Problem 2.9]. In this thesis 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 any* $\epsilon > 0$*.*

Unfortunately, algorithms which are competitive 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 1 and 2 arrive at time 0 and at every unit time step thereafter, one more 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 job 1 and then schedule jobs as they arrive, scheduling job 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 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 is $\left(\sum_{i\in[n]}(C_i - r_i)^k\right)^{1/k}$; in particular, the ℓ_1-norm is total flow time. For the ℓ_k-norm objective for any $k > 1$, the previous example has one optimal solution: schedule jobs in the order they arrive. This schedule can be seen as 'fair' to each job.

For the ℓ_k-norm objective, every online deterministic algorithm is $n^{\Omega(1)}$-competitive, even on a single machine, when $1 < k < \infty$ [BP03]. This situation is quite different from the ℓ_1-norm, where SRPT is an optimal algorithm. In the single machine setting, SRPT is a scalable algorithm for the ℓ_k-norm objective for all $k > 1$ [BP03]. The competitiveness of SRPT in the multiple machine setting was not known for ℓ_k-norms for any constant speed. The algorithm of Chekuri et al. is scalable for the problem of minimizing the ℓ_k-norm of flow time on identical machines for all $k > 1$ [CGKK04]. Pruhs et al. suggested that determining whether or not SRPT is scalable for the ℓ_k-norms of flow time on identical machines is another interesting open question [PST04]. In this thesis 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*-norm of flow time on* m *identical parallel machines for any* $k \geq 1$ *and* $1/2 \geq \epsilon > 0$*.*

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.

1.1 Related Work

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 the property can be shown for SRPT on every input. In this thesis, we avoid a local analysis by using a potential function argument which we discuss further in the following chapter.

Chapter 2

Preliminaries

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

Throughout this thesis, we will concentrate on bounding SRPT's *kth power flow time*, $\sum_{i\in[n]} \left(C_i^S - r_i\right)^k$, as it is the ℓ_k-norm of flow time without the outer root. For each released job i, we let $\text{SRPT}(i,t)$ and $\text{OPT}(i,t)$ denote the respective algorithm's accumulated kth power flow time for job i at time t. In other words,

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

Also, we let

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

We define $\text{OPT}(i,t)$ and $\text{OPT}(t)$ similarly. When used as values, $\text{SRPT} = \text{SRPT}(\infty)$ and $\text{OPT} = \text{OPT}(\infty)$.

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

$$R^S(i,t) = \sum_{j:r_j\leq t, C_j^S\leq C_i^S} p_j^S(t).$$

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, and we only include those jobs with original processing time at most p_i. Precisely,

$$V^O(i,t) = \sum_{j:r_j\leq t, C_j^S\leq C_i^S, p_j\leq p_i} p_j^O(t).$$

We will assume without loss of generality that all arrival and completion times are distinct by breaking ties arbitrarily but consistently so that jobs have total orderings by their arrival and completion times in each schedule. We also

assume jobs are numberd in increasing order by their arrival time.

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 each job j contributing to $R^S(i,t)$ with $p_j^S(t) \leq p_i$. In other words,

$$X(i,t) = \sum_{j: r_j \leq t, C_j^S \leq C_i^S, p_j^S(t) \leq p_i} p_j^S(t).$$

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 i, so we see $X(i,t) = R^S(i,t)$ whenever $t \geq r_i$. Thus it suffices to show that $X(i,t) - V^O(i,t) \leq mp_i$ for all $t > 0$. 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 job 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 i. SRPT always works on the m available jobs with earliest completion time, and as these jobs also have small remaining processing time, this work causes X to decrease by at least $m(t' - t)$ (this argument even assumes SRPT is not given resource augmentation). Combining these facts we have the following:

$$\begin{aligned} X(i,t) - V^O(i,t) &\leq (X(i,t') + T - m(t'-t)) - \left(V^O(i,t') + T - m(t'-t)\right) \\ &\leq mp_i \end{aligned}$$

□

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 SRPT + Φ. These changes may come from the following sources:

Job Arrival: Arriving jobs will not affect SRPT but they will change 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 change Φ. We will bound these increases by γOPT where γ is a non-negative constant which may depend on k and ϵ.

Running Condition: The running condition essentially captures everything else. We will show a bound on the continuous changes in SRPT + Φ over time as well as the changes to each job's remaining processing time. Surprisingly, we find $\frac{\mathrm{d}}{\mathrm{dt}}\text{SRPT} + \frac{\mathrm{d}}{\mathrm{dt}}\Phi \leq 0$, 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 value is at most 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.

Chapter 3

Total Flow Time

Assume SRPT is given speed $(1+\epsilon)$ for some fixed $\epsilon > 0$. We focus on upper bounding SRPT's total flow time by some multiple of OPT's. To accomplish this task, we define a potential function with one term for each job being processed. Define $\Phi(i,t) : [n] \times [0,\infty) \to \mathbb{R}$ as follows:

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

Our potential function is

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

We base this potential function on the following intuition:

- Job arrivals and completions increase the potential function at most some multiple of OPT.
- Each term decreases at a sufficient rate to counteract the gradual increase in SRPT's flow time.

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 arrival of a job makes no change to the accumulated flow time, but it can change Φ. Consider the arrival of job i at time $t = r_i$. For any $j \neq i$ such that $j \in \mathcal{Q}^S(t)$, consider the term $\Phi(j,t)$. The arrival of job i changes both $R^S(j,t)$ and $V^O(j,t)$ equally (either by p_i or 0 depending on if $p_i \leq p_j^S(t)$) creating no net change in the potential function. We do gain a new term in the summation, but we can upper bound the term as follows:

$$\begin{aligned}
\Phi(i,t) &= \frac{1}{m\epsilon}\left(R^S(i,t) + mp_i - V^O(i,t)\right) \\
&\leq \frac{1}{m\epsilon}(2mp_i) && \text{by Lemma 2.1} \\
&= \frac{2}{\epsilon}p_i \leq \frac{2}{\epsilon}\text{OPT}(i,\infty)
\end{aligned}$$

In total, job arrivals increase Φ by

$$\sum_{i\in[n]} \frac{2}{\epsilon}\text{OPT}(i,\infty) = \frac{2}{\epsilon}\text{OPT}.$$

Job Completion: Job completions also do not change the accumulated flow time. Consider the completion of a job i by OPT at time $t = C_i^O$. For any job $j \in \mathcal{Q}^{\mathcal{S}}(t)$, the term $\Phi(j,t)$ does not change as job i already contributes nothing to $V^O(j,t)$.

Likewise, consider the completion of job i by SRPT at time $t = C_i^S$. For any $j \neq i$ such that $j \in \mathcal{Q}^{\mathcal{S}}(t)$, the term $\Phi(j,t)$ does not change as job i already contributes nothing to $R^S(j,t)$. Unfortunately, we need a more sophisticated argument to derive an upper bound on the increase in Φ from removing the term $\Phi(i,t)$.

We know $\Phi(i,t) = -\frac{1}{m\epsilon}V^O(i,t)$, because SRPT has completed all jobs contributing to $R^S(i,t)$ and $p_i^S(t) = 0$. We use the following scheme to charge the removal of $\Phi(i,t)$ and similar increases of Φ to OPT's total flow time. Consider any job j contributing volume to $V^O(i,t)$. If $r_j < r_i$, the definition of $V^O(i,t)$ implies $p_j \leq p_i$. Further, if $r_j \geq r_i$, the definition of $R^S(i,t)$ implies $p_j \leq p_i^S(r_j)$. In either case, SRPT performs at least p_j units of work on job i while job 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 job j's contribution to $\frac{1}{m\epsilon}V^O(i,t)$, we charge at least $\frac{1+\epsilon}{m\epsilon}\frac{p_j}{1+\epsilon} = \frac{p_j}{m\epsilon}$ to job j's increase in flow time during this period by charging at a rate of $\frac{1+\epsilon}{m\epsilon}$. Letting $\mathcal{S}_i$ be the set of jobs contributing to $V^O(i,t)$, we see the total charge accrued during this period is

$$\sum_{j\in\mathcal{S}_i} \frac{p_j}{m\epsilon} \geq \frac{1}{m\epsilon}V^O(i,t).$$

Now we need to bound the total charge over all job completions. Observe that any charge to some job j accrues at $\frac{1+\epsilon}{m\epsilon}$ times the rate that job j is accumulating flow time. Further, SRPT is working on at most m jobs at any time, so our combined charges are accruing at $\frac{1+\epsilon}{\epsilon}$ times the rate that job j is accumulating flow time. By summing over all time and jobs, we conclude that we charge at most $\frac{1+\epsilon}{\epsilon}$OPT. Thus,

$$\sum_{i\in[n]} \Phi(i, C_i^S) \geq -\frac{1+\epsilon}{\epsilon}\text{OPT},$$

and job completions increase Φ by at most $\frac{1+\epsilon}{\epsilon}$OPT.

Running Condition: We now proceed to upper bound $\frac{\mathrm{d}}{\mathrm{dt}}\text{SRPT}(t) + \frac{\mathrm{d}}{\mathrm{dt}}\Phi(t)$ at any time t when no job arrives or finishes. Observe:

- $\frac{\mathrm{d}}{\mathrm{dt}}V^O(i,t) \geq -m$ because OPT can only process m jobs at a time using unit speed.
- $\frac{\mathrm{d}}{\mathrm{dt}}\left(R^S(i,t) + mp_i^S(t)\right) \leq -m(1+\epsilon)$. Neither term can increase, so we upper bound the change of one or the other to prove this fact. If SRPT is processing job i (using $(1+\epsilon)$ speed) at time t, then $\frac{\mathrm{d}}{\mathrm{dt}}mp_i^S(t) \leq -m(1+\epsilon)$.

Otherwise, m other jobs are being processed. By definition, these jobs are contributing their volume to $R^S(i,t)$, so $\frac{\mathrm{d}}{\mathrm{dt}} R^S(i,t) \leq -m(1+\epsilon)$.

We infer the following:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{SRPT}(t) + \frac{\mathrm{d}}{\mathrm{dt}}\Phi(t) &= \sum_{i \in \mathcal{Q}^S(t)} \left[\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{SRPT}(i,t) + \frac{\mathrm{d}}{\mathrm{dt}}\Phi(i,t) \right] \\
&\leq \sum_{i \in \mathcal{Q}^S(t)} \left[1 + \frac{1}{m\epsilon}\left(-m(1+\epsilon) + m\right) \right] \\
&= 0
\end{aligned}$$

Final Analysis: Using the framework described in Chapter 2 and the above analysis, we conclude $\mathrm{SRPT} \leq \frac{4}{\epsilon}\mathrm{OPT}$ for any $\epsilon < 1$. Combining this fact with the earlier speed augmentation results of [PSTW02, TM08], we conclude the proof of Theorem 1.1. $\square$

Chapter 4

ℓ_k-Norms of Flow Time

In this chapter we focus on the ℓ_k-norms of flow time for any fixed $k \geq 1$. Assume that SRPT is given speed $(1+\epsilon)$ where $1/2 \geq \epsilon > 0$. We require 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 let

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

and

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

Consider any job sequence σ.

Job Arrival: Consider the arrival of job i at time $t = r_i$. Again, the objective function does not change. Also, as in the case for standard flow time, $\Phi(j,t)$ does not change for any $j \neq i$. However, a new term $\Phi(i,t)$ is added to the summation in the potential function, which we can upper bound as follows:

$$\begin{aligned}
\Phi(i,t) &= \frac{1}{(1-\epsilon)^k}\left(\max\left\{\frac{1}{m\epsilon}\left(R^S(i,t) + mp_i - V^O(i,t)\right), 0\right\}\right)^k \\
&\leq \frac{1}{(1-\epsilon)^k}\left(\frac{1}{m\epsilon}(2mp_i)\right)^k && \text{by Lemma 2.1} \\
&= \left(\frac{2}{\epsilon(1-\epsilon)}\right)^k (p_i)^k \leq \left(\frac{2}{\epsilon(1-\epsilon)}\right)^k \text{OPT}(i,\infty)
\end{aligned}$$

In total, job arrivals increase Φ by

$$\sum_{i \in [n]} \left(\frac{2}{\epsilon(1-\epsilon)}\right)^k \text{OPT}(i,\infty) = \left(\frac{2}{\epsilon(1-\epsilon)}\right)^k \text{OPT}.$$

Job Completion: Completing job i at time $t = C_i^S$ decreases Φ by $\Phi(i,t)$. We will use the following lemmas.

Lemma 4.1. *For any job $i \in \mathcal{Q}^S(t)$, if $V^O(i,t) \leq m\epsilon^2(t-r_i)$ then $\Phi(i,t) \geq 0$.*

Proof. The hypothesis implies $t - r_i + \frac{1}{m\epsilon}\left(R^S(i,t) + mp_i^S(t) - V^O(i,t)\right) \geq 0$, because

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

for all $t \geq r_i$ and $\epsilon \leq 1/2$. Because $V^O(i,t) \leq m\epsilon^2(t-r_i)$, we have

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

□

Definition-chasing implies:

Lemma 4.2. *For any job $i \in \mathcal{Q}^{\mathcal{S}}(t)$, if $V^O(i,t) > m\epsilon^2(t-r_i)$ then $\Phi(i,t) \geq -\left(\frac{1}{\epsilon^2 m}V^O(i,t)\right)^k$.*

We conclude that the total increase to Φ from all job completions is at most

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

In Section 4.1, we prove that this increase is at most $\left(\frac{1+\epsilon}{\epsilon^2}\right)^k$ OPT.

Running Condition: We now upper bound $\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{SRPT}(t) + \frac{\mathrm{d}}{\mathrm{dt}}\Phi(t)$ at any time t when no job arrives or finishes. Let

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

Again, $\frac{\mathrm{d}}{\mathrm{dt}}V^O(i,t) \geq -m$ and $\frac{\mathrm{d}}{\mathrm{dt}}\left(R^S(i,t) + mp_i^S(t)\right) \leq -m(1+\epsilon)$. Therefore,

$$\frac{\mathrm{d}}{\mathrm{dt}}\tau(i,t) \leq 1 + \frac{1}{m\epsilon}(-m(1+\epsilon) + m) = 0.$$

We infer the following:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\text{SRPT}(t) + \frac{\mathrm{d}}{\mathrm{dt}}\Phi(t) &= \sum_{i \in \mathcal{Q}^{\mathcal{S}}(t)} \left[\frac{\mathrm{d}}{\mathrm{dt}}\text{SRPT}(i,t) + \frac{\mathrm{d}}{\mathrm{dt}}\Phi(i,t)\right] \\ &\leq \sum_{i \in \mathcal{Q}^{\mathcal{S}}(t)} \left[k \cdot (t - r_i)^{k-1} - k \cdot (t - r_i)^{k-1}\right] \\ &= 0\end{aligned}$$

Final Analysis: The framework discussed in Chapter 2 and the arrival, completion, and running conditions shown in this chapter imply that

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

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

4.1 Final Steps

In this section, we prove that if SRPT is running m machines at speed $(1+\epsilon)$ while OPT is running m machines at unit speed, then

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

for the metric of kth power flow time. We use the following charging scheme to prove this bound. For concision, let $V_i = V^O(i, C_i^S)$.

Let $\mathcal{S}_i$ denote the set of jobs that contribute to V_i and let $\mathcal{T}_j = \{i : j \in \mathcal{S}_i\}$.

Lemma 4.3. *For any job j and job $i \in \mathcal{T}_j$,* SRPT *performs at least $p_j^O(C_i^O)$ volume of work on job i during the time interval $[r_j, C_j^O]$.*

Proof. SRPT gives higher priority to job j than job i, because job j contributes to V_i. As seen in the completion condition arguments for total flow time, SRPT does p_j volume of work on job i during $[r_j, C_i^S]$. Namely, if $r_j < r_i$ then $p_j \leq p_i$, and if $r_j \geq r_i$ then $p_j \leq p_i^S(r_j)$. Finally, we note that $p_j \geq p_j^O(C_i^O)$ and $C_i^S < C_j^O$. □

Recall that jobs are numberd in order of increasing arrival time. In order to upper bound $\left(\frac{1}{\epsilon^2 m} V_i\right)^k$, we charge the following to each job $j \in \mathcal{S}_i$:

$$\left(\frac{1}{\epsilon^2 m}\right)^k \left(\left(V_i - \sum_{a \in \mathcal{S}_i : a < j} p_a^O(C_i^S)\right)^k - \left(V_i - \sum_{a \in \mathcal{S}_i : a \leq j} p_a^O(C_i^S)\right)^k\right)$$

Note that $p_j^O(C_i^S)$ is not included in the left summation, but it is included in the right one. In total, we charge the following to pay for $\left(\frac{1}{\epsilon^2 m}V_i\right)^k$ (note the telescoping sum in the topmost expression):

$$\begin{aligned}
&\sum_{j\in\mathcal{S}_i}\left[\left(\frac{1}{\epsilon^2 m}\right)^k\left(\left(V_i-\sum_{a\in\mathcal{S}_i:a<j}p_a^O(C_i^S)\right)^k-\left(V_i-\sum_{a\in\mathcal{S}_i:a\le j}p_a^O(C_i^S)\right)^k\right)\right]\\
&\quad=\left(\frac{1}{\epsilon^2 m}\right)^k\left((V_i)^k-\left(V_i-\sum_{a\in\mathcal{S}_i}p_a^O(C_i^S)\right)^k\right)\\
&\quad=\left(\frac{1}{\epsilon^2 m}V_i\right)^k
\end{aligned}$$

Now fix some job j. We show that we charge at most $\left(\frac{1+\epsilon}{\epsilon^2}\right)^k(C_j^O-r_j)^k$ in total to job j.

Lemma 4.4. *For any job $i\in\mathcal{T}_j$,*

$$\frac{1}{\epsilon^2 m}\left(V_i-\sum_{a\in\mathcal{S}_i:a<j}p_a^O(C_i^S)\right)\le\frac{1}{\epsilon^2}\left((1+\epsilon)(C_j^O-r_j)-\frac{1}{m}\sum_{a\in\mathcal{T}_j:C_a^S>C_i^S}p_j^O(C_a^O)\right).$$

Proof. We account for work done by SRPT during $[r_j,C_j^O]$ in two stages and use the result to derive the inequality. Let $\mathcal{T}_j'=\left\{a\in\mathcal{T}_j:C_a^S>C_i^S\right\}$. By Lemma 4.3, SRPT must do at least $\sum_{a\in\mathcal{T}_j'}p_j^O(C_a^O)$ volume of work on the jobs in $\mathcal{T}_j'$ during $[r_j,C_j^O]$.

Next, let $\mathcal{S}_i'=\{a\in\mathcal{S}_i:a\ge j\}$. These jobs arrive after time r_j and must be completed by time $C_i^S\le C_j^O$ since they count toward V_i. This means SRPT completes at least $\sum_{a\in\mathcal{S}_i'}p_a\ge\sum_{a\in\mathcal{S}_i'}p_a^O(C_i^S)$ volume of work on the jobs in $\mathcal{S}_i'$ during $[r_j,C_j^O]$. Observe that $\mathcal{T}_j'$ and $\mathcal{S}_i'$ are disjoint as no job a with $C_a^S>C_i^S$ counts toward V_i by definition.

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

$$\begin{aligned}
&r_j+\frac{1}{(1+\epsilon)m}\left(\sum_{a\in\mathcal{T}_j'}p_j^O(C_a^O)+\sum_{a\in\mathcal{S}_i'}p_a^O(C_i^S)\right)\\
&\quad=r_j+\frac{1}{(1+\epsilon)m}\left(\sum_{a\in\mathcal{T}_j:C_a^S>C_i^S}p_j^O(C_a^O)+V_i-\sum_{a\in\mathcal{S}_i:a<j}p_a^O(C_i^S)\right).
\end{aligned}$$

This expression is at most C_j^O, so the lemma follows by basic algebra. □

The total amount charged to job j is

$$\sum_{i\in\mathcal{T}_j}\left[\left(\frac{1}{\epsilon^2 m}\right)^k\left(\left(V_i-\sum_{a\in\mathcal{S}_i:a<j}p_a^O(C_i^S)\right)^k-\left(V_i-\sum_{a\in\mathcal{S}_i:a\le j}p_a^O(C_i^S)\right)^k\right)\right]$$
$$\le\sum_{i\in\mathcal{T}_j}\left[\frac{1}{\epsilon^{2k}}\left(\left((1+\epsilon)(C_j^O-r_j)-\frac{1}{m}\sum_{a\in\mathcal{T}_j:C_a^S>C_i^S}p_j^O(C_a^O)\right)^k\right.\right.$$
$$\left.\left.-\left((1+\epsilon)(C_j^O-r_j)-\frac{1}{m}\sum_{a\in\mathcal{T}_j:C_a^S\ge C_i^S}p_j^O(C_a^O)\right)^k\right)\right].$$

where the inequality follows from Lemma 4.4 and the convexity of x^k for all $k\ge 1$.

Once again, the above expression is a telescoping sum and is equal to

$$\frac{1}{\epsilon^{2k}}\left(\left((1+\epsilon)(C_j^O-r_j)\right)^k-\left((1+\epsilon)(C_j^O-r_j)-\frac{1}{m}\sum_{a\in\mathcal{T}_j}p_j^O(C_a^O)\right)^k\right).$$

By Lemma 4.3, SRPT completes at least $\sum_{a\in\mathcal{T}_j}p_j^O(C_a^O)$ work during $[r_j,C_j^O]$. Therefore,

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

and

$$(1+\epsilon)(C_j^O-r_j)-\frac{1}{m}\sum_{a\in\mathcal{T}_j}p_j^O(C_a^O)\ge 0.$$

Thus, the total charged to job j is at most $\left(\frac{1+\epsilon}{\epsilon^2}\right)^k(C_j^O-r_j)^k$. Summing, we see the total amount charged to all jobs is at most

$$\sum_{j\in[n]}\left(\frac{1+\epsilon}{\epsilon^2}\right)^k(C_j^O-r_j)^k=\left(\frac{1+\epsilon}{\epsilon^2}\right)^k\text{OPT},$$

concluding the lemma. □

Chapter 5

Conclusion

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

[FM11] Kyle Fox and Benjamin Moseley. Online scheduling on identical machines using SRPT. In *SODA '11: Proceedings of the Twenty-first Annual ACM -SIAM Symposium on Discrete Algorithms*, 2011.

[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.