Server Scheduling in the L_p Norm: A Rising Tide Lifts All Boat

[Extended Abstract]

Nikhil Bansal *
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
nikhil@cs.cmu.edu

Kirk Pruhs †
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
kirk@cs.pitt.edu

ABSTRACT

Often server systems do not implement the best known algorithms for optimizing average Quality of Service (QoS) out of concern of that these algorithms may be insufficiently fair to individual jobs. The standard method for balancing average QoS and fairness is optimize the L_p metric, $1 < p < \infty$. Thus we consider server scheduling strategies to optimize the L_p norms of the standard QoS measures, flow and stretch. We first show that there is no $n^{o(1)}$-competitive online algorithm for the L_p norms of either flow or stretch. We then show that the standard clairvoyant algorithms for optimizing average QoS, *SJF* and *SRPT*, are $O(1+\epsilon)$-speed $O(1/\epsilon)$-competitive for the L_p norms of flow and stretch. And that the standard nonclairvoyant algorithm for optimizing average QoS, *SETF*, is $O(1+\epsilon)$-speed $O(1/\epsilon^{(2+2/p)})$-competitive for the L_p norms of flow. These results argue that these standard algorithms will not starve jobs until the system is near peak capacity. In contrast, we show that the Round Robin, or Processor Sharing algorithm, which is sometimes adopted because of its seeming fairness properties, is not $O(1+\epsilon)$-speed $n^{o(1)}$-competitive for sufficiently small ϵ.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Sequencing and Scheduling; C.4 [**Performance of Systems**]: Performance Attributes

General Terms

Algorithms, Peformance, Theory

*Supported by IBM Research Fellowship.

†Supported in part by NSF grant CCR-0098752, NSF grant ANIR-0123705, and by a grant from the US Air Force.

STOC'03, June 9–11, 2003, San Diego, California, USA.

Keywords

scheduling, resource augmentation, flow time, shortest elapsed time first, shortest remaining processing time, multilevel feedback, shortest job first

1. INTRODUCTION

The algorithms Shortest Remaining Processing Time (*SRPT*) and Shortest Elapsed Time First (*SETF*) are generally regarded as the best *clairvoyant* and *nonclairvoyant* server scheduling policies for optimizing average Quality of service (QoS). *SRPT* is optimal for average flow time, and is 2-competitive for average stretch [25]. *SETF* is an $(1+\epsilon)$-speed $(1+1/\epsilon)$-competitive algorithm for average flow time [21] — no $O(1)$-competitive nonclairvoyant algorithm exists for average flow time [24] — and a randomized variation *RMLF* of *SETF* is known to be asymptotically strongly competitive amongst randomized algorithms for average flow time [7, 20, 24].

In spite of this, *SRPT* and *SETF* are generally not implemented in server systems. Apache, currently most widely used web server [18], uses a separate process for each request, and uses a First In First Out (*FIFO*) policy to determine the request that is allocated a free process slot [17]. The most commonly cited reason for adopting *FIFO* is a desire for some degree of fairness to individual jobs [16], i.e. it is undesirable for some jobs to be starved. In some sense *FIFO* is the optimally fair policy in that it optimizes the maximum flow time objective. Operating systems such as Unix don't implement pure *SETF*, or even pure *MLF*, the less preempting version of *SETF*. Once again this is out of fear of starving jobs [28, 29]. Unix systems adopt compromise policies that attempt to balance the competing demands of average QoS and fairness. In particular, Unix scheduling policies generally raise the priority of processes in the lower priority queues that are being starved [28].

The desire to optimize for the average and the desire to not have extreme outliers generally conflict. The most common way to compromise is to optimize the L_p norm, generally for something like $p = 2$ or $p = 3$. For example, the standard way to fit a line to collection of points is to pick the line with minimum least squares, equivalently L_2, distance to the points, and Knuth's TEXtypesetting system uses the L_3 metric to determine line breaks [22, page 97]. The L_p, $1 < p < \infty$, metric still considers the average in the sense that it takes into account all values, but because x^p is strictly a convex function of x, the L_p norm more severely penalizes outliers than the standard L_1 norm.

This leads us in this paper to consider server scheduling algorithms for optimizing the L_p norms, $1 < p < \infty$, of the two standard QoS metrics: flow and stretch. In section 4, we show that that are no $n^{o(1)}$-competitive online server scheduling algorithms for any L_p metric, $1 < p < \infty$ of either flow or stretch. Perhaps this is a bit surprising, at least for the flow metric, as there are optimal online algorithms, SRPT and FIFO, for the L_1 and L_∞ norms.

This negative result motivates us to fall back to resource augmentation analysis, which compares the online algorithm against an optimal offline algorithm with a slower processor. More formally, in the context of a scheduling minimization problem with an objective function F, an algorithm A is s-speed c-competitive if

$$\max_I \frac{F(A_s(I))}{F(Opt_1(I))} \leq c$$

where $A_s(I)$ denotes the the schedule that algorithm A with a speed s produces on input I, and similarly $Opt_1(I)$ denotes the adversarial schedule for I with a unit speed processor.

Algorithm	Speed	L_p Norm of Flow	L_p norm of Stretch
Any Clairvoyant Algorithm	1	$n^{\Omega(1)}$	$n^{\Omega(1)}$
SJF	$(1+\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
SRPT	$(1+\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
SETF	$(1+\epsilon)$	$O(1/\epsilon^{2+\frac{2}{p}})$	$O(\frac{1}{\epsilon^{3+\frac{1}{p}}} \lg^{1+\frac{1}{p}} B)$
RR	$(1+\epsilon)$	$\Omega(n^{1-2\epsilon p})$	$\Omega(n)$
Any Non-clairvoyant Algorithm	$(1+\epsilon)$		$\Omega(\min\{n, \log B\})$

Table 1: Resource augmentation results for the standard average QoS algorithms.

We first consider the standard online algorithms aimed at average QoS, that is, Shortest Job First (*SJF*), *SRPT*, and *SETF*. We show that the clairvoyant algorithms, *SJF* and *SRPT*, are $O(1+\epsilon)$-speed $O(1/\epsilon)$-competitive for the L_p norms of flow and stretch. We show that the nonclairvoyant algorithm *SETF* is $O(1+\epsilon)$-speed $O(1/\epsilon^{(2+2/p)})$-competitive for the L_p norms of flow. Further for the L_p norm of stretch, we show that given constant speed-up, *SETF* is poly-logarithmically competitive in B, the ratio of the length of the longest job to the shortest job. And we give an essentially matching lower bound. We summarize our results in table 1. Note that all of the results assume that p is constant, so that multiplicative factors, that are a function of p alone, are absorbed into the constant in the asymptotic notation.

These resource augmentation results argue that the concern, that the standard algorithms aimed at optimizing average QoS might unnecessarily starve jobs, is unfounded when the server is less than fully loaded. Average QoS curves such as those in figure 1(a) are ubiquitous in server systems [23]. That is, there is a relatively modest degradation in average

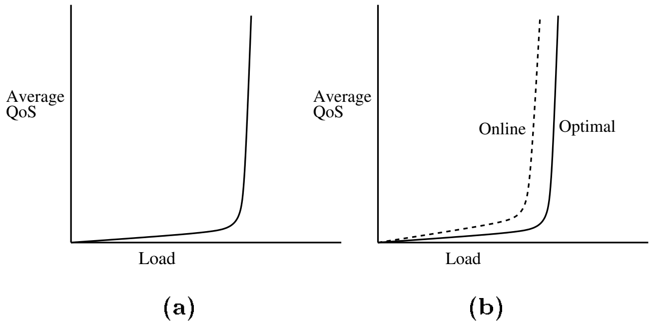

Figure 1: (a) Standard QoS curve, and (b) The worst possible QoS curve of an $(1+\epsilon)$-speed $O(1)$-competitive online algorithm.

QoS as the load increases until some threshold is reached — this threshold is essentially the capacity of the system — after which any increase in the load precipitously degrades the average QoS. The concept of load is not so easy to formally define, but generally reflects the number of users of the system. If A is an $(1+\epsilon)$-speed c-competitive server scheduling algorithm, and $Opt_1(I) \leq d \cdot Opt_{1+\epsilon}(I)$ then A is at worst $(c \cdot d)$-competitive on input I. The loads in the usual performance curve shown in figure 1(a) where $Opt_1(I)$ is not approximately $Opt_{1+\epsilon}(I)$ are those points near or above the the capacity of the system. Thus the performance curve of a $(1+\epsilon)$-speed c-competitive online algorithm A should be at no worse than shown in figure 1(b). That is, A should scale reasonably well up to quite near the capacity of the system.

It might be tempting to conclude that all reasonable algorithms should have such scaling guarantees. However, we show that this is not the case in section 9. More precisely, the standard Processor Sharing, or equivalently Round Robin, algorithm is not $(1+\epsilon)$-speed $O(1)$-competitive for any L_p norm of flow, $1 < p < \infty$ and for sufficiently small ϵ. This is perhaps surprising, since fairness is a commonly cited reason for adopting Processor Sharing [28].

2. RELATED RESULTS

The results in the literature that are closest in spirit to those here are found in a series of papers, including [6, 13, 16, 27]. These papers also argue that SRPT will not unnecessarily starve jobs any more than Processor Sharing does under "normal" situations. In these papers, "normal" is defined as there being a Poisson distribution on release times, and processing times being independent samples from a heavily tailed distribution. More precisely, these papers argue that every job should prefer SRPT to Processor Sharing under these circumstances. Experimental results supporting this hypothesis are also given. So informally our paper and these papers reach the same conclusion about the superiority of SRPT. But in a formal sense the results are incomparable.

Resource augmentation analysis was proposed as a method for analyzing scheduling algorithms in [21]. We adopt the notation and terminology from [26]. Highest Density First was shown to be $(1+\epsilon)$-speed $O(1/\epsilon)$-competitive for weighted flow time in [8]. This immediately implies a similar result for *SJF* for average flow time and for average stretch. *RR* is also known to be $O(1)$-speed $O(1)$-competitive for average flow time [14].

For minimizing average stretch, [25] show that $SRPT$ is 2-competitive. In the offline case, there is a PTAS for average stretch [9]. In the non-clairvoyant case, [5] showed that any algorithm is $\Omega(n)$ competitive (without speedup) and gave a $(1+\epsilon)$-speed $O(poly(\epsilon^{-1} \cdot \log B))$-competitive algorithm.

There have been some prior results on scheduling problems with L_p norms. Load balancing in the L_p norm is discussed in [1, 3, 2]. PTAS for L_p norms of completion times, without release times, is given in [15].

A seemingly related problem is that of minimizing the weighted flow time, studied recently by [12, 11, 4]. Observe that minimizing the sum of squares of flow time is equivalent to minimizing the weighted flow time where the weight of a job at any time is equal to its age (hence the weights are linearly increasing with time). However, the fact that the weights are changing makes our problem substantially different. For example, [4] give an $O(\log W)$ competitive algorithm for weighted flow time (which would correspond to an $O(\log n + \log B)$ competitive algorithm in our case, since the maximum weight W in our case is at most nB). However, our lower bounds in section 4 show that any randomized online algorithm is $\Omega(\max\{n^{1/5}, B^{1/5}\})$ competitive for minimizing flow time squared.

3. DEFINITIONS

We assume that all job sizes are integers in the range $[1, \ldots, B]$. We assume a collection of jobs $\mathcal{J} = J_1, \ldots, J_n$. The release time of J_i is r_i and the processing time of J_i is p_i. The completion time C_j^S of a job J_i in a schedule S is the first time after r_i where J_i has been processed for p_i time units. The flow time of J_i in S is $C_i^S - r_i$, and the stretch of J_i is $(C_i^S - r_i)/p_i$. A clairvoyant algorithm learns p_i at time r_i. A nonclairvoyant algorithm only knows a lower bound on p_i equal to the length of time that has run p_i. Finally, for an algorithm A, $F^p(A)$ (respectively $S^p(A)$) will denote the sum of the p^{th} powers of the flowtime (respectively stretch) of all jobs under A. Note that the L_p norm of flowtime (resp stretch) under algorithm A is $(F^p(A))^{1/p}$ (resp $(S^p(A))^{1/p}$).

4. GENERAL LOWER BOUNDS

THEOREM 1. *The competitive ratio of any randomized algorithm A against an oblivious adversary for F^p and S^p, $1 < p < \infty$, is $n^{\Omega(1)}$.*

PROOF. We use Yao's minimax principle for online cost minimization problems [10] and lower bound the expected value of the ratio of the objective functions on A and Opt on input distribution which we specify. The inputs are parameterized by integers L, α, and β in the following manner. A *long* job of size L arrives at $t = 0$. From 0 to time until time $L^\alpha - 1$ a job of size 1 arrives every unit of time. With probability 1/2 this is all of the input. With probability 1/2, $L^{\alpha+\beta}$ *short* jobs of length $1/L^\beta$ arrive every $1/L^\beta$ time units from time L^α until $2L^\alpha - 1/L^\beta$.

We first consider the F^p objective. In this case, $\alpha = \frac{p+1}{p-1}$, and $\beta = 2$. We now compute $F^p(A)$ and $F^p(Opt)$. Consider first the case that A doesn't finish the long job by time L^α. Then with probability 1/2 the input contains no short jobs. Then $F^p(A)$ is at least the flow of the long job, which is at least $L^{\alpha p}$. In this case the adversary could first process the long jobs and then process the unit jobs. Hence, $F^p(Opt) = O(L^p + L^\alpha \cdot L^p) = O(L^{\alpha+p})$. The competitive ratio is then $\Omega(L^{\alpha p - \alpha - p})$, which is $\Omega(L)$ by our choice of α.

Now consider the case that A finishes the long job by time L^α. Then with probability 1/2 the input contains short jobs. One strategy for the adversary is to finish all jobs, except for the big job, when they are released. Then $F^p(Opt) = O(L^\alpha \cdot 1^p + L^{\alpha+\beta} \cdot (1/L^\beta)^p + L^{\alpha p})$. It is obvious that the dominant term is $L^{\alpha p}$, and hence, $F^p(Opt) = O(L^{\alpha p})$. Now consider the subcase that A has at least $L/2$ unit jobs unfinished by time $3L^\alpha/2$. Since these unfinished unit jobs much have been delayed by at least $L^\alpha/2$, $F^p(A) = \Omega(L \cdot L^{\alpha p})$. Clearly in this subcase the competitive ratio is $\Omega(L)$. Alternatively, consider the subcase that A has finished at least $L/2$ unit jobs by time $3L^\alpha/2$. Then A has at least $L^{\alpha+\beta}/2$ released, and unfinished, small jobs at time $3L^\alpha/2$. By the convexity F^p, the optimal strategy for A from time $3L^\alpha/2$ onwards it to delay each small job by the same amount. Thus A delays $L^{\alpha+\beta}/2$ short jobs by at least $L/2$. Hence in this case, $F^p(A) = \Omega(L^{\alpha+\beta} \cdot L^p)$. This gives a competitive ratio of $\Omega(L^{\alpha+\beta+p-\alpha p})$, which by the choice of β is $\Omega(L)$.

We now consider the S^p objective. In this case, $\alpha = \frac{2p+1}{p-1}$, and $\beta = 1$. We now compute $S^p(A)$ and $S^p(Opt)$. Consider first the case that A doesn't finish the long job by time L^α. Then with probability 1/2 the input contains no short jobs. Then $S^p(A)$ is at least the stretch of the long job, which is at least $(L^\alpha/L)^p = L^{p(\alpha-1)}$. In this case the adversary could first process the long jobs and then process the unit jobs. Hence, $S^p(Opt) = O(1 + L^\alpha \cdot L^p) = O(L^{\alpha+p})$. The competitive ratio is $\Omega(L^{\alpha p - \alpha - 2p})$, which is $\Omega(L)$ by our choice of α.

Now consider the case that A finishes the long job by time L^α. Then with probability 1/2 the input contains short jobs. One strategy for the adversary is to finish all jobs, except for the big job, when they are released. Then $S^p(Opt) = O(L^\alpha \cdot 1^p + L^{\alpha+\beta} \cdot 1^p + (L^\alpha/L)^p)$. Algebraic simplification shows that $\alpha + \beta \leq \alpha p - p$ for our choice of α and β. Hence dominant term is $L^{\alpha p - p}$, and $S^p(Opt) = O(L^{\alpha p - p})$. Now consider the subcase that A has at least $L/2$ unit jobs unfinished at time $3L^\alpha/2$. Since these unfinished unit jobs much have been delayed by at least $L^\alpha/2$, $S^p(A) = \Omega(L \cdot L^{\alpha p})$. Clearly in this subcase the competitive ratio is $\Omega(L^{p+1})$. Alternatively, consider the subcase that A has finished at least $L/2$ unit jobs by time $3L^\alpha/2$. Then A has at least $L^{\alpha+\beta}/2$ released, and unfinished, small jobs at time $3L^\alpha/2$. By the convexity S^p when restricted to jobs of size $1/L^\beta$, the optimal strategy for A from time $3L^\alpha/2$ onwards always delays each small job by the same amount (we can gift A the competition of the unit jobs at time $3L^\alpha/2$). Thus A delays $L^{\alpha+\beta}/2$ short jobs by at least $L/2$. Hence in this case, $S^p(A) = \Omega(L^{\alpha+\beta} \cdot L^{(\beta+1)p})$. This gives a competitive ratio of $S^p(A) = \Omega(L^{\alpha+\beta+\beta p+2p-\alpha p})$.

By the choice of α and β, this once gives a competitive ratio of $\Omega(L^{2p+1})$. □

It is easy to see that there is no $O(1)$-speed $O(1)$-competitive nonclairvoyant algorithm for S^p, $1 < p < \infty$ using the input instance from [19]. Consider n jobs of sizes $1, 2, 4, 8, \ldots, B = 2^n$ arriving at time 0. By scheduling the jobs from shortest to longest, $S^p(Opt) = O(n)$. While for any non-clairvoyant algorithm A, $S^p(A, s) = \Omega(n^{p+1}/s^p)$.

5. ANALYSIS OF SJF

In this section we show that *SJF* is a $(1+\epsilon)$-speed $O(1/\epsilon^p)$-competitive for F^p and S^p objective functions.

We fix a time t. Let $U(SJF, t)$ and $U(Opt, t)$ denote the unfinished jobs at time t in *SJF* and *Opt* respectively, and $\mathcal{D} = U(SJF, t) - U(Opt, t)$. Let $Age^p(X, t)$ denote the sum over all jobs $J_i \in X$ of $(t - r_i)^{p-1}$. Let $SAge^p(X, t)$ denote the sum over all jobs $J_i \in X$ of $(t - r_i)^{p-1}/p_i^p$. We will demonstrate the following local competitiveness condition for all times t

$$Age^p(\mathcal{D}, t) \leq O(1/\epsilon^p) Age^p(U(Opt, t), t)$$

This will establish our desired results because $F^p(A) = p \int_t Age^p(U(A, t), t)dt$ and $S^p(A) = p \int_t SAge^p(U(A, t), t)dt$.

Before proceeding we introduce some needed notation. Let $V(t, \alpha)$ denote the aggregate unfinished work at time t among those jobs J_j that satisfy the conditions in the list α. So for example, in the next lemma we consider $V(r_i, J_j \in U(Opt, r_i), r_j \leq r_i, p_j \leq p_i)$, the amount of work that *Opt* has left on jobs J_j that arrived before J_i, are shorter than J_i and that *Opt* has not finished by time r_i. Let $P(\alpha)$ will denote the aggregate initial processing times of jobs J_j that satisfy the conditions in the list α.

We prove local competitiveness in the following manner. Let $1, \ldots, k$ denote the indices of jobs in $\mathcal{D}$ such that $p_1 \leq p_2 \ldots \leq p_k$. Consider the jobs in $\mathcal{D}$ in the order in which they are indexed. Assume that we are considering job J_i. We allocate to J_i an $\epsilon p_i/4(1+\epsilon)$ amount of work from $V(t, J_j \in U(Opt, t), r_j \leq t - \epsilon(t - r_i)/(4(1+\epsilon)), p_j \leq p_i)$ that was previously not allocated to a lower indexed job in $\mathcal{D}$. This establishes $O(1/\epsilon^p)$ local competitiveness for F^p for the following reasons. The total unfinished work in each $J_j \in U(Opt, t)$ is associated with $O(1/\epsilon)$ longer jobs in $\mathcal{D}$. Since the jobs J_j are $\Omega(\epsilon)$ as old as J_i, the contribution of to $Age^p(U(Opt, t), t)$ for J_j is $\Omega(\epsilon^{p-1})$ as large as the contribution of J_i to $Age^p(U(SJF, t), t)$. Using the same reasoning, and the fact that $p_j \leq p_i$, establishes local competitiveness for S^p.

We now turn to proving that this association scheme works, that is, the scheme never runs of work to assign to the jobs in $\mathcal{D}$. Consider a fixed job $J_i \in \mathcal{D}$. Let t' denote the time $t - \frac{\epsilon}{4(1+\epsilon)}(t - r_i)$. If this scheme is going to fail on job J_i then, informally, the amount of work on jobs of size $\leq p_i$ that *Opt* had left at time r_i, plus the work made up by jobs of size $\leq p_i$ that arrived during $(r_i, t']$, minus the work that *Opt* did during $(r_i, t]$, minus the work that is allocated to $J_1, \ldots, J_i$ should be negative. The amount of work in $V(t, J_j \in U(Opt, t))$ that is allocated to $J_1, \ldots, J_i$ is at most

$$\frac{\epsilon}{4(1+\epsilon)} P(J_j \in \mathcal{D}, r_j \leq r_i, p_j \leq p_i) + \frac{\epsilon}{4(1+\epsilon)} P(J_j \in \mathcal{D}, r_j > r_i, p_j \leq p_i)$$

Moreover, as *Opt* can finish at most $(t - r_i)$ work during time $(r_i, t]$, it is sufficient to show that

$$\begin{aligned} & V(r_i, J_j \in U(Opt, r_i), r_j \leq r_i, p_j \leq p_i) \\ & + P(r_j \in (r_i, t'], p_j \leq p_i) - (t - r_i) \\ & - \frac{\epsilon}{4(1+\epsilon)} P(J_j \in \mathcal{D}, r_j \leq r_i, p_j \leq p_i) \\ & + \frac{\epsilon}{4(1+\epsilon)} P(J_j \in \mathcal{D}, r_j > r_i, p_j \leq p_i) \geq 0 \end{aligned}$$

Or equivalently,

$$\begin{aligned} & V(r_i, J_j \in U(Opt, r_i), r_j \leq r_i, p_j \leq p_i) \qquad (1) \\ & + P(r_j \in (r_i, t'], p_j \leq p_i) \\ & - \frac{\epsilon}{4(1+\epsilon)} P(J_j \in \mathcal{D}, r_j \leq r_i, p_j \leq p_i) \\ & + \frac{\epsilon}{4(1+\epsilon)} P(J_j \in \mathcal{D}, r_j > r_i, p_j \leq p_i) \geq (t - r_i) \end{aligned}$$

The rest of this section will be devoted to establishing equation 1. We know that

$$(t - r_i) \geq P(J_j \in \mathcal{D}, r_j > r_i, p_j \leq p_i) \qquad (2)$$

since *Opt* had to finish all such jobs considered on the right hand side between time r_i and time t. Then by substitution, to prove equation 1 it suffices to prove:

$$\begin{aligned} & V(r_i, J_j \in U(Opt, r_i), r_j \leq r_i, p_j \leq p_i) \qquad (3) \\ & + P(r_j \in (r_i, t'], p_j \leq p_i) \\ & - \frac{\epsilon}{4(1+\epsilon)} P(J_j \in \mathcal{D}, r_j \leq r_i, p_j \leq p_i) \\ \geq \quad & (\frac{4+5\epsilon}{4(1+\epsilon)})(t - r_i) \qquad (4) \end{aligned}$$

We now concentrate on replacing the term $V(r_i, J_j \in U(Opt, r_i), r_j \leq r_i, p_j \leq p_i)$ in equation 3. This is where the $1+\epsilon$ speedup is crucially used.

LEMMA 2. *For all times u and for all values p_i,*

$$\begin{aligned} & V(u, J_j \in U(Opt, u), r_j \leq u, p_j \leq p_i) \geq \\ & \frac{\epsilon}{1+\epsilon}(P(J_j \in U(SJF, u), p_j \leq p_i) \\ + \quad & \frac{\epsilon}{1+\epsilon} V(u, J_j \in U(SJF, u), p_j \leq p_i)) \end{aligned}$$

PROOF. The proof is by induction on the time u. Whenever there is a job $J_j \in U(SJF, u)$ with $p_j \leq p_i$, then the right hand side of the inequality decreases at least as fast as the left hand side since *SJF* has a $(1+\epsilon)$-speed processor. If the is no such job J_j, then the right hand side of the inequality is zero. □

Using lemma 2 with $u = r_i$, then to prove equation 3 it is sufficient by substitution to prove:

$$\begin{aligned} & \frac{\epsilon}{1+\epsilon} P(J_j \in U(SJF, r_i), p_j \leq p_i) \qquad (5) \\ & + \frac{1}{1+\epsilon} V(r_i, J_j \in U(SJF, r_i), p_j \leq p_i) \\ & + P(r_j \in (r_i, t'], p_j \leq p_i) \\ & - \frac{\epsilon}{4(1+\epsilon)} P(J_j \in \mathcal{D}, r_j \leq r_i, p_j \leq p_i) \\ \geq \quad & (\frac{4+5\epsilon}{4(1+\epsilon)})(t - r_i) \end{aligned}$$

Since obviously, $P(J_j \in U(SJF, r_i), p_j \leq p_i) \geq P(J_j \in \mathcal{D}, r_j \leq r_i, p_j \leq p_i)$ and also $V(r_i, J_j \in U(SJF, r_i), p_j \leq p_i) \leq P(J_j \in U(SJF, r_i), p_j \leq p_i)$, to prove equation 5 it suffices to show that

$$\begin{aligned} & \frac{4+3\epsilon}{4(1+\epsilon)} V(r_i, J_j \in U(SJF, r_i), p_j \leq p_i) \qquad (6) \\ & + P(r_j \in (r_i, t'], p_j \leq p_i) \geq (\frac{4+5\epsilon}{4(1+\epsilon)})(t - r_i) \end{aligned}$$

During $[r_i, t']$, *SJF* does $(1+\epsilon)(t'-r_i)$ work on jobs shorter than J_i. By algebraic simplification $(1+\epsilon)(t'-r_i) = \frac{4+3\epsilon}{4}(t-r_i)$. The jobs that *SJF* worked on during $[r_i, t']$ either had to arrive before r_i or during $[r_i, t']$. Therefore, it trivially follows that

$$\begin{aligned}(t-r_i) \quad &\leq \quad \frac{4}{4+3\epsilon}V(r_i, J_j \in U(SJF, r_i), p_j \leq p_i) \qquad (7)\\ &+\frac{4}{4+3\epsilon}P(r_j \in (r_i, t'], p_j \leq p_i)\end{aligned}$$

Hence by substitution using equation 7, to prove equation 6 it suffices to prove:

$$\begin{aligned}&\frac{4+3\epsilon}{4(1+\epsilon)}V(r_i, J_j \in U(SJF, r_i), p_j \leq p_i) \qquad (8)\\ &+P(r_j \in (r_i, t'], p_j \leq p_i)\\ \geq \quad &\frac{4+5\epsilon}{(4+3\epsilon)(1+\epsilon)}V(r_i, J_j \in U(SJF, r_i), p_j \leq p_i)\\ &+\frac{4+5\epsilon}{(4+3\epsilon)(1+\epsilon)}P(r_j \in (r_i, t'], p_j \leq p_i)\end{aligned}$$

Now, it is easy to see that equation 8 is true since,

$$1 \geq \frac{4+5\epsilon}{(4+3\epsilon)(1+\epsilon)}$$

and

$$\frac{4+3\epsilon}{4(1+\epsilon)} \geq \frac{4+5\epsilon}{(4+3\epsilon)(1+\epsilon)}$$

Thus we have proved our main theorem for this section.

THEOREM 3. *SJF is $(1+\epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive for the L_p norms of flowtime and stretch, for $p \geq 1$.*

Note that the above proof also holds for maximum flow and stretch, that is when $p = \infty$, which prompted the metaphorical portion of this paper's title.

6. ANALYSIS OF SRPT

In this section we prove identical results for SRPT. The analysis is somewhat more involved than that for SJF because we need to handle the remaining times of jobs under SRPT more carefully.

Let us define the *relaxed age*, denoted by $rAge^p(J_i, t)$, of a job J_i at time t to be 0 if $(t-r_i) \leq 8p_i/\epsilon$ and $(t-r_i)^{p-1}$ otherwise. Similarly, we define $rSAge^p(J_i, t)$ to be 0 if $(t-r_i) \leq 8p_i/\epsilon$ and $(t-r_i)^{p-1}/p_i^p$ otherwise.

Note that if $F(J_i) \leq 8p_i/\epsilon$, then $p\int_t rAge^p(J_i, t) = 0$ and if $F(J_i) > 8p_i/\epsilon$, then $p\int_t rAge^p(J_i, t) = F^p(J_i) - (8p_i/\epsilon)^p$. Thus, we always have that

$$F^p(J_i) \leq p\int_t rAge^p(J_i, t) + (8p_i/\epsilon)^p$$

Similarly, $S^p(J_i) \leq p\int_t rSAge^p(J_i, t) + (8/\epsilon)^p$

Define $rAge(X, t)$ and $rSAge(X, t)$ for a set of jobs X as the sum of $rAge$'s and $sAge$'s of jobs in X. Now, if we show that for all t,

$$\begin{aligned}&rAge^p(U(SRPT, t) \setminus U(Opt, t), t)\\ \leq \quad &O(1/\epsilon^p)Age^p(U(Opt, t), t)\end{aligned}$$

then it follows that,

$$rAge^p(U(SRPT, t), t) = O(1/\epsilon^p)Age^p(U(Opt, t), t)$$

and hence that

$$\int_t rAge^p(U(SRPT, t), t) = O(1/\epsilon^p)\int_t Age^p(U(Opt, t), t)$$

Now since the flow time of J_i it at least p_i, it is easy to see that $F^p(SRPT) \leq \int_t rAge^p(U(SRPT, t), t) + \sum_i (8p_i/\epsilon)^p = O(1/\epsilon)^p F^p(Opt)$. Similarly, it suffices to show that

$$\begin{aligned}&rSAge^p(U(SRPT, t) \setminus U(Opt, t), t)\\ \leq \quad &O(1/\epsilon^p)SAge^p(U(Opt, t), t)\end{aligned}$$

for proving $O(1/\epsilon)^p$ competitive for S^p.

To prove these, we modify the analysis for *SJF* as follows: Given a time t, define $\mathcal{D}$ to consist of only those jobs which have age more that $8/\epsilon$ times their size, and which *Opt* has finished but are present under *SRPT*. Mimicking the proof for *SJF*, for every job J_i in $\mathcal{D}$ will we show that we can associate $\epsilon p_i/4(1+\epsilon)$ units of work (which has not been already allocated to a lower indexed job in $\mathcal{D}$) from some job of size $\leq p_i$ in *Opt* and which has age at least $\frac{\epsilon(t-r_i)}{4(1+\epsilon)}$. Clearly, this implies that $rAge^p(U(SRPT, t) - U(Opt, t), t) \leq O(1/\epsilon^p)Age^p(U(Opt, t), t)$ and also that

$$\begin{aligned}&rSAge^p(U(SRPT, t) \setminus U(Opt, t), t)\\ \leq \quad &O(1/\epsilon^p)SAge^p(U(Opt, t), t)\end{aligned}$$

For job J_j, let $rem(j, t)$ denote its remaining processing requirement at time t. We begin by a result similar to Lemma 2.

LEMMA 4. *For all times u and values of p_i,*

$$\begin{aligned}&V(u, J_i \in U(Opt, u), r_j \leq u, p_j \leq p_i) \qquad (9)\\ \geq \quad &\frac{\epsilon}{1+\epsilon}P(J_j \in U(SRPT, u), p_j \leq p_i)\\ &+\frac{1}{1+\epsilon}V(u, J_j \in U(SRPT, u), p_j \leq p_i) - \frac{1}{1+\epsilon}p_i\end{aligned}$$

PROOF. First observe that by the nature of SRPT, at any time u and any value p, there can be at most one which has both size greater than p and remaining time less than p. Thus the term

$$V(u, J_j \in U(SRPT, u), rem(j, u) \leq p_i, p_j > p_i)$$

never exceeds p_i.

We now prove the lemma by induction on the time u. Whenever there is a job in with size less than p_i or remaining time less than p_i then the right hand side of the inequality decreases at least as fast as the left hand side. If the is no such job J_j, then the right hand side of the inequality is $-p_i$ where as the left hand side is 0. Now, it might happen that SRPT works on some job of size $> p_i$ and its remaining time becomes p_i or less. However, it is easy to see that in this case the right hand side is no more than 0. $\square$

We now apply Equation 9 with $u = r_i$ to Lemma 3. We also use the obvious facts that $P(J_j \in U(SRPT, r_i), p_j \leq p_i) \geq P(J_j \in \mathcal{D}, r_j \leq r_i, p_j \leq p_i)$ and $P(J_j \in U(SRPT, r_i), p_j \leq p_i) \geq V(r_i, J_j \in U(SRPT, r_i), p_j \leq p_i)$. Thus it suffices to show that

$$\begin{aligned}&\frac{4+3\epsilon}{4(1+\epsilon)}P(J_j \in U(SRPT, r_i), p_j \leq p_i) \qquad (10)\\ &+P(r_j \in (r_i, t'], p_j \leq p_i) - \frac{1}{1+\epsilon}p_i \geq (\frac{4+5\epsilon}{4(1+\epsilon)})(t-r_i)\end{aligned}$$

Since SRPT does not finish J_i by time t (and hence by time t'). It must be the case that $(1+\epsilon)(t'-r_i) \leq V(r_i, J_j \in U(SRPT, r_i), rem(j) \leq p_i) + P(r_j \in (r_i, t'], p_j \leq p_i)$. Since,

$$\begin{aligned} & V(r_i, J_j \in U(SRPT, r_i), rem(j) \leq p_i) \\ \leq \quad & V(r_i, J_j \in U(SRPT, r_i), p_j \leq p_i) + p_i \end{aligned}$$

we get that

$$\begin{aligned} \frac{4+3\epsilon}{4}(t-r_i) \quad \leq \quad & V(r_i, J_j \in U(SRPT, r_i), p_j \leq p_i)i \\ & + P(r_j \in (r_i, t'], p_j \leq p_i) + p_i \end{aligned}$$

Now, since $p_i \leq \frac{\epsilon}{8}(t-r_i)$ (this is where we use that we are considering relaxed ages) we have that,

$$\begin{aligned} \frac{4+2\epsilon}{4}(t-r_i) \quad \leq \quad & V(r_i, J_j \in U(SRPT, r_i), p_j \leq p_i)(11) \\ & + P(r_j \in (r_i, t'], p_j \leq p_i) - p_i \end{aligned}$$

Using 12, to prove 10 it suffices to show that

$$\begin{aligned} & \frac{4+3\epsilon}{4(1+\epsilon)} V(J_j \in U(SRPT, r_i), p_j \leq p_i) \quad (12) \\ & + P(r_j \in (r_i, t'], p_j \leq p_i) - \frac{1}{1+\epsilon} p_i \\ \geq \quad & (\frac{4+5\epsilon}{4(1+\epsilon)})(\frac{2}{2+\epsilon}) V(r_i, J_j \in U(SRPT, r_i), p_j \leq p_i) \\ & + (\frac{4+5\epsilon}{4(1+\epsilon)})(\frac{2}{2+\epsilon}) P(r_j \in (r_i, t'], p_j \leq p_i) - p_i \end{aligned}$$

However, comparing term by term, it is easy to see that equation 12 always holds.

7. FLOW ANALYSIS OF SETF

Our goal is to show that *SETF* is $(1+\epsilon)$-speed $O(1/\epsilon^{2+2/p})$-competitive for the L_p norm of flowtime. We first compare *SETF* to an intermediate policy *MLF*, and then compare *MLF* to *Opt*. Our *MLF* is a variation of the standard Multilevel Feedback Queue algorithm where the quanta for the queues are set as a function of ϵ. Let $\ell_i = \epsilon((1+\epsilon)^i - 1)$, $i \geq 0$, and let $q_i = \ell_{i+1} - \ell_i = \epsilon^2(1+\epsilon)^i$, $i \geq i$. In *MLF* a job J_j is in *level* k *at time* t if the work done on J_j by time t is $\geq \ell_k$ and $< \ell_{k+1}$. *MLF* maintains the invariant that it is always running the earliest arriving job in the smallest nonempty level.

LEMMA 5. *For any instance $\mathcal{I}$, and for any $J_j \in \mathcal{I}$, J_j completes in $SETF_{1+\epsilon}$ before J_j completes in MLF_1.*

PROOF. For any job $J_j \in \mathcal{I}$ and any time t, let $w(j,t)$ denote the work done on J_j by time t. For a job with $w(j,t) \leq x$, let $rem_{\leq x}(j,t) = \min(p_j, x) - w(j,t)$, that is, the amount of work that must be done on J_j until either J_j completes or until J_j has been processed for x units. Let $U(A, t, w(j,t) \leq x)$ denote the set of unfinished jobs in algorithm A at time t which have less than x work done on them. Let $W_{\leq x}(A,t)$ denote $\sum_{J_j \in U(A,t,w(j,t) \leq x)} rem_{\leq x}(j,t)$. Now, as *SETF* is always working on the job with least work done, it is easy to see that $W_{\leq p}(SETF, t) \leq W_{\leq p}(A,t)$ for any algorithm A with the same speed processor at *SETF*.

Suppose to reach a contradiction that there some job J_j which completes earlier in MLF_1 than in $SETF_{1+\epsilon}$. Clearly MLF_1 must finish the work $W_{\leq p_j}(MLF_1, r_j)$ before time $C_j^{MLF_1}$ since MLF_1 will complete earlier arriving shorter jobs before later arriving longer jobs. Moreover, at time $C_j^{MLF_1}$ all the jobs in $U(MLF_1, t)$ have at least $p_j/(1+\epsilon)$ amount of work done on them, otherwise J_j wouldn't be run by MLF_1 at time $C_j^{MLF_1}$ since there would be a lower level job than J_j at this time. Consider the schedule $A_{1+\epsilon}$ that at all times t, runs the job that MLF_1 is running at time t (and idles if this job is already completed). Hence, $A_{1+\epsilon}$ would have completed J_j, and all previously arriving jobs with processing time less that p_j, by time $C_j^{MLF_1}$ since $A_{1+\epsilon}$ has a $(1+\epsilon)$-speed processor. Hence, by the property of *SETF* from the previous paragraph, $SETF_{1+\epsilon}$ completes J_j by time $C_j^{MLF_1}$, which is our contradiction. □

Lemma 5 implies that

$$F^p(SETF(\mathcal{I}), s) \leq F^p(MLF(\mathcal{I}), s/(1+\epsilon)) \quad (13)$$

Now our goal will be to relate *MLF* and *Opt*. Let the original instance be $\mathcal{I}$. Let $\mathcal{J}$ be the instance obtained from $\mathcal{I}$ as follows. Consider a job J_j and let i be the smallest integer such that $p_j + \epsilon \leq \epsilon(1+\epsilon)^i$. The processing time of J_j in $\mathcal{J}$ is then $\epsilon(1+\epsilon)^i$. Let $\mathcal{K}$ be the instance obtained from $\mathcal{J}$ by decreasing each job size by ϵ. Thus, each job in $\mathcal{K}$ has size $\ell_k = \epsilon((1+\epsilon)^k - 1)$ for some k. Note that in this transformation from $\mathcal{I}$ to $\mathcal{K}$, the size of a job doesn't decrease, and it increases by at most a factor of $(1+\epsilon)^2$. Since *MLF* is has the property that increasing the length of a particular job will not decrease the completion time of any job, we can conclude that

$$F^p(MLF(\mathcal{I}), s/(1+\epsilon)) \leq F^p(MLF(\mathcal{K}), s/(1+\epsilon)) \quad (14)$$

Finally we create an instance $\mathcal{L}$ by replacing each job of size $\epsilon((1+\epsilon)^k - 1)$ job in $\mathcal{K}$ by k jobs of size $q_1, \ldots, q_{k-1}$. Note that $\ell_k = q_0 + q_1 + \ldots, q_{k-1}$. For a job of $J_j \in \mathcal{K}$, we denote the corresponding jobs in $\mathcal{L}$ by $J_{j_0}, J_{j_1}, \ldots, J_{j_{k-1}}$.

Notice that any time t, $SJF(\mathcal{L})$ is working on a job $J_{j_b} \in \mathcal{L}$ if and only if $MLF(\mathcal{K})$ is working on job $J_j \in \mathcal{K}$ that is in level b at time t. In particular, this implies that the completion time of J_j in $MLF(\mathcal{K})$ is exactly the completion time of some job $J_{j_b} \in SJF(\mathcal{L})$. Hence,

$$F^p(MLF(\mathcal{K}), s/(1+\epsilon)) \leq F^p(SJF(\mathcal{L}), s/(1+\epsilon)) \quad (15)$$

By Theorem 3, we know that

$$F^p(SJF(\mathcal{L}), s/(1+\epsilon) = O(1/\epsilon^p) F^p(Opt(\mathcal{L}), s/(1+\epsilon)^2) \quad (16)$$

We relate the optimal schedule for $\mathcal{L}$ back to the optimal schedule for $\mathcal{I}$. To do this we first relate $\mathcal{L}$ to $\mathcal{J}$ as follows. Let $\mathcal{L}(k)$ denote the instance obtained from $\mathcal{J}$ by multiplying each job size in $\mathcal{J}$ by $\epsilon/(1+\epsilon)^k$. Next, we remove from $\mathcal{L}(k)$ any job whose size is less than ϵ^2. We claim that $\mathcal{L} = \mathcal{L}(1) \cup \mathcal{L}(2) \cup \ldots$. To see this, let us consider some job $J_j \in \mathcal{J}$ of size $\epsilon(1+\epsilon)^i$. Then, $L(1)$ contains the corresponding job $J_{j_{i-1}}$ of size $\epsilon/(1+\epsilon) \cdot \epsilon(1+\epsilon)^i = \epsilon^2(1+\epsilon)^{i-1} = q_{i-1}$. Similarly $\mathcal{L}(2)$ contains the job $J_{j_{i-2}}$ of size q_{i-2} and so on. Thus, $\mathcal{L}$ is exactly $\mathcal{L}(1) \cup \mathcal{L}(2) \cup \ldots$. Summarizing, we have that the $\mathcal{L}(k)'s$ are geometrically scaled down copies of J and that $\mathcal{L}$ is exactly the union of these $\mathcal{L}(k)'s$.

Our idea at a high level is as follows: Given a *good* schedule of $\mathcal{J}$, we obtain a good schedule for $\mathcal{L}(k)$. This will be easy to do as $\mathcal{L}(k)$ is a scaled down version of $\mathcal{J}$. Finally, we will superimpose the schedules for each $\mathcal{L}(k)$ to obtain a schedule for $\mathcal{L}$. This will give us a procedure to obtain a *good* schedule for $\mathcal{L}$ given a good schedule for $\mathcal{J}$. To put everything in place, we need the following lemma from [5] while relates $\mathcal{J}$ and $\mathcal{L}(k)$.

LEMMA 6. *Let $\mathcal{L}(k)$ be as defined above. Let $F(s, J_i, \mathcal{G}))$ (respectively $S(s, J_i, \mathcal{G}))$ denote the flow time (respectively stretch) of job $J_i \in \mathcal{G}$ when SJF is run on $\mathcal{G}$ with a speed s processor. Then, for all $x \geq 1$,*

$$F(\epsilon(1+\epsilon)^{-k} \cdot x \cdot s, J_{i_k}, \mathcal{L}(k)) \leq \frac{1}{x} F(s, J_i, \mathcal{J})$$

$$S(\epsilon(1+\epsilon)^{-k} \cdot x \cdot s, J_{i_k}, \mathcal{L}(k) \leq \frac{(1+\epsilon)^k}{\epsilon x} S(s, J_i, \mathcal{J})$$

PROOF. We first show that for all jobs $J_j \in \mathcal{J}$, $F(s, J_j, \mathcal{J}) \leq (1/s) F(1, J_j, \mathcal{J})$. Let $w(x, s, J_j)$ denote the work done on job J_j, after x units of time since it arrived, under SJF using an s speed processor. We will show a stronger invariant that for all jobs J_j and all times t, $w((t - r_j)/s, s, J_j) \geq w(t - r_j, 1, J_j)$.

Consider some instance where this condition is violated. Let J_j be the job and t be the earliest time for which $w((t - r_j)/s, s, J_j) < w(t - r_j, 1, J_j)$. Clearly, SJF_s is not working on J_j at time t, due to minimality of t. Thus, SJF_s is working on some other smaller job i. Since SJF_1 is not working on i, SJF_1 has already finished i by some time $t' < t$. However, this means that $w((t' - r_i), s, J_i) < w(t' - r_i, 1, J_i)$, which contradicts the minimality of t.

We now show the main result of the lemma. It is easy to see that the flow time of every job $J_j(k) \in \mathcal{L}(k)$ (where $J_j(k)$ is job corresponding to $J_j \in \mathcal{J}$ but scaled down by $\epsilon(1+\epsilon)^{-k}$ times) under SJF with a speed $\epsilon(1+\epsilon)^{-k}$ processor is at most that of the corresponding job $J_j \in \mathcal{J}$, under SJF with a unit speed processor. Thus by the fact above, running SJF on $\mathcal{L}(k)$ with an $x \cdot \epsilon(1+\epsilon)^{-k}$-speed processor yields a $1/x$ times smaller flow time for each job in $\mathcal{L}(k)$ than the corresponding job in $\mathcal{J}$.

Finally, since the size of jobs in $\mathcal{L}(k)$ are $\epsilon(1+\epsilon)^{-k}$ times smaller than in $\mathcal{J}$, the result for stretch follows from the result for flow time. □

LEMMA 7.

$$F^p(Opt(\mathcal{L}), 1+2\epsilon) = O(1/\epsilon^2) F^p(SJF(\mathcal{J}), 1)$$

PROOF. Given the schedule $SJF(\mathcal{J})$, we construct a following schedule A for $\mathcal{L}$ as follows. The jobs in $\mathcal{L}(k)$ are run with a speed $x_k = \epsilon(1 + \frac{\epsilon}{1+\epsilon})^{-k}$ processor using the algorithm *SJF*. Note that the total speed required by A is at most

$$\sum_{i=1}^{\infty} x_i = \sum_{i=1}^{\infty} \epsilon \frac{1}{(1+\frac{\epsilon}{1+\epsilon})^i} = \frac{\epsilon}{1 - \frac{1+\epsilon}{1+2\epsilon}} = 1 + 2\epsilon$$

By Lemma 6, $F^p(A(\mathcal{L}(k)), 1+2\epsilon)$ will be at most $\left(\frac{\epsilon(1+\epsilon)^{-k}}{x_k}\right)^p = \left(\frac{(1+2\epsilon)}{(1+\epsilon)^2}\right)^{kp}$ times $F^p(SJF(\mathcal{J}, 1)$. Hence,

$$\begin{aligned} \frac{F^p(A(\mathcal{L}), 1+2\epsilon)}{F^p(SJF(\mathcal{J}), 1)} &\leq \sum_{i=1}^{\infty} \left(\frac{1+2\epsilon}{(1+\epsilon)^2}\right)^{ip} \\ &\leq \sum_{i=0}^{\infty} \left(1 - \frac{\epsilon^2}{(1+\epsilon)^2}\right)^i \\ &= O(1/\epsilon^2) \end{aligned}$$

The proof then follows because *Opt* is at least as good as A. □

Hence by Lemma 7, Theorem 3, and the fact that that jobs lengths in $\mathcal{J}$ are at most $(1+\epsilon)^2$ times as long as they are in $\mathcal{I}$, we get that

$$\begin{aligned} & F^p(Opt(\mathcal{L}), s/(1+\epsilon)^2) \\ = \ & O(1/\epsilon^2) \cdot F^p(SJF(\mathcal{J}), \frac{s}{(1+2\epsilon)(1+\epsilon)^2}) \\ = \ & O(1/\epsilon^{p+2}) \cdot F^p(Opt(\mathcal{J}), \frac{s}{(1+2\epsilon)(1+\epsilon)^3}) \\ = \ & O(1/\epsilon^{p+2}) \cdot F^p(Opt(\mathcal{I}), \frac{s}{(1+2\epsilon)(1+\epsilon)^5}) \quad (17) \end{aligned}$$

By stringing together the inequalities 13, 14, 15, 16, 17, we find that

$$F^p(SETF(\mathcal{I}), s) = O(1/\epsilon^{2p+2}) F^p(Opt(\mathcal{I}), \frac{s}{(1+2\epsilon)(1+\epsilon)^4})$$

Hence, we conclude that *SETF* is $(1+\epsilon)$-speed $O(1/\epsilon^{2+2/p})$-competitive for the L_p norm of flowtime.

8. STRETCH ANALYSIS OF SETF

We consider stretch analysis of *SETF*. The analysis is similar to the analysis for flow time. By Lemma 5, it follows that $S^p(SETF(\mathcal{I}), 1+\epsilon) \leq S^p(MLF(\mathcal{I}), 1)$. Similarly, since each job in $J_i \in \mathcal{I}$ is at most $(1+\epsilon)^2$ times smaller than the corresponding job in $\mathcal{K}$, and also has size no more than that of the corresponding job in $\mathcal{J}$, we get that $S^p(MLF(\mathcal{I}), 1) \leq (1+\epsilon)^{2p} S^p(MLF(\mathcal{K}), 1)$. These together give us that

$$S^p(SETF(\mathcal{I}), 1+\epsilon) \leq (1+\epsilon)^{2p} S^p(MLF(\mathcal{K}), 1) \quad (18)$$

For a job of size $\epsilon[(1+\epsilon)^k - 1]$ in $\mathcal{K}$, the corresponding job of size $\epsilon^2(1+\epsilon)^k \in \mathcal{L}$ has equal flow time. Thus the ratio of the contributions to the p^{th} power of stretch for $\mathcal{L}$ and $\mathcal{K}$ will be at least $(\frac{\epsilon^2(1+\epsilon)^k}{\epsilon[(1+\epsilon)^k - 1]})^p$, which is at least $(\frac{\epsilon}{1-(1+\epsilon)^{-k}})^p$. Now since $\epsilon[(1+\epsilon)^k - 1] \geq 1$ for all valid job sizes in $\mathcal{K}$, we get that $(1+\epsilon)^{-k} \leq \frac{\epsilon}{1+\epsilon}$ and hence that $(\frac{\epsilon}{1-(1+\epsilon)^{-k}})^p \leq (\epsilon(1+\epsilon))^p$. Thus we get that $S^p(MLF(\mathcal{K}), 1) \leq (\epsilon(1+\epsilon))^p S^p(SJF(\mathcal{L}), 1)$. Now, applying Theorem 3 it follows that $S^p(SJF(\mathcal{L}), 1+\epsilon) = O(1/\epsilon^p) S^p(Opt(\mathcal{L}), 1)$. Thus we get that

$$S^p(MLF(\mathcal{K}), 1) = O((1+\epsilon)^p) S^p(SJF(\mathcal{L}), 1) \quad (19)$$

Recall Theorem 3 that

$$S^p(SJF(\mathcal{L}), 1+\epsilon) = O(1/\epsilon^p) S^p(Opt(\mathcal{L}), 1) \quad (20)$$

We now prove a result similar to Lemma 7 for stretch.

LEMMA 8.

$$S^p(Opt(\mathcal{L}), 1+\epsilon) = O(\frac{\log_{1+\epsilon}^{p+1} B}{\epsilon^p}) S^p(SJF(\mathcal{J}), 1)$$

PROOF. Set $x_i = \epsilon(1+\epsilon)^{-i}$ for $i = 1, \ldots,$ $\log_{1+\epsilon} \log_{1+\epsilon}(B/\epsilon)$, and $x_i = \epsilon / \log_{1+\epsilon}(B/\epsilon)$ for $i > \log_{1+\epsilon} \log_{1+\epsilon}(B/\epsilon)$. We run the jobs in $\mathcal{L}(i)$ using *SJF* on a speed x_i processor. Notice that the total speedup required is $\sum_{i=1}^{\log_{1+\epsilon}(B/\epsilon)} x_i$ which is at most $1+\epsilon$.

By, lemma 6, $S^p(SJF(\mathcal{L}(i)), x_i)$ is at most $(1/x_i^p) S^p(SJF(\mathcal{J}), 1)$. By simple algebraic calculation it can be seen that $\sum_i (\frac{1}{x_i})^p$ is $O(\frac{1}{\epsilon^p} \log_{1+\epsilon}^{p+1} B)$. □

Combining equations 18, 19, and 20, and lemma 8, we get that

$$\begin{aligned} & S^p(SETF(\mathcal{I}), (1+\epsilon)^2) \\ = \ & O(\frac{(1+\epsilon)^{3p}}{\epsilon^p} \cdot \log_{1+\epsilon}^{p+1} B) S^p(SJF(\mathcal{J}), 1) \quad (21) \end{aligned}$$

Now by Theorem 3 we have that

$$S^p(SJF(\mathcal{J}), 1+\epsilon) = O(1/\epsilon^p)S^p(Opt(\mathcal{J}), 1) \qquad (22)$$

Also, since each job $J_i \in \mathcal{J}$ has size at most $(1+\epsilon)^2$ times more than the corresponding job $J_i \in \mathcal{I}$ (and is not smaller), we trivially have that

$$S^p(Opt(\mathcal{J}), (1+\epsilon)^2) \leq S^p(Opt(\mathcal{I}), 1) \qquad (23)$$

Now combining equations 21, 22 and 23 it follows that

$$S^p(SETF, (1+\epsilon)^5, \mathcal{I}) = O(\frac{1}{\epsilon^{2p}} \cdot \log_{1+\epsilon}^{p+1} B)S^p(Opt(\mathcal{I}), 1)$$

or equivalently that

$$S^p(SETF, (1+\epsilon)^5, \mathcal{I}) = O(\frac{1}{\epsilon^{3p+1}} \cdot \lg^{p+1} B)S^p(Opt(\mathcal{I}), 1)$$

Thus we shown that,

THEOREM 9. *SETF is a $(1+\epsilon)$-speed $O(\frac{1}{\epsilon^{3+1/p}} \cdot \lg^{1+1/p} B)$-competitive algorithm for the L_p norm of stretch.*

9. LOWER BOUND FOR ROUND ROBIN

We now show that for every $p \geq 1$, there is an $\epsilon > 0$ such that Round Robin (RR) is not an $(1+\epsilon)$-speed $n^{o(1)}$-competitive algorithm for the L_p norm of flow time. Observe, that as any non-clairvoyant algorithm is $\Omega(n)$ competitive with respect to the L_p norm of stretch, so is RR.

Suppose $\epsilon < 1/2$ and RR has a processor of speed $1+\epsilon$. Consider the following instance. Two jobs of size $p_0 = 1$ arrive at $r_0 = 0$. Next we have a collection of n jobs whose release times and sizes are defined as follows. The first job of size $p_1 = p_0(1 - \frac{1+\epsilon}{2})$ arrives at time $r_1 = p_0$. In general the i^{th} job has size $p_i = (1 - \frac{1+\epsilon}{i+1})p_{i-1}$ and $r_i = \sum_{j=0}^{i-1} p_j$. The instance has the following properties which are easily verified:

1. Except for one job of size 1 which arrives at time 0, each job under SRPT has a flow time equal to its size. The job of size 1 has flow time $t' = 1 + \sum_{j=0}^{n} p_j$.

2. Under RR (with a $1+\epsilon$ speed processor) all the jobs keep accumulating and finish simultaneously at time $t = (2p_0 + \sum_{j=1}^{n} p_i)/(1+\epsilon)$.

We now consider the relevant quantities. First observe that $p_i = \Pi_{j=1}^{i} \frac{(j-\epsilon)}{j+1} = \frac{1}{i+1}\Pi_{j=1}^{i}(1-\epsilon/j)$. Using the fact that for all $x \geq 0$, $1 - x \leq e^{-x}$ we get that $p_i \leq \frac{1}{i+1}e^{-H(i)\epsilon}$ where $H(i)$ is the i^{th} Harmonic number. Finally using that $H(i) \geq \ln i$, we get that $p_i \leq \frac{1}{i+1}i^{-\epsilon} \leq i^{-(1+\epsilon)}$.

We now upper bound $F^p(SRPT, 1)$, and hence $F^p(Opt, 1)$. Observing that $\sum_{j=0}^{n} p_j \leq \sum_{j=0}^{\infty} p_j \leq \sum_{j=0}^{\infty} j^{-(1+\epsilon)} = O(1)$ and that $\sum_{j=0}^{n} p_j^p \leq \sum_{j=0}^{\infty} j^{-(1+\epsilon)p} = O(1)$ we get that $F^p(SRPT, 1) = O(1)$.

On the other hand, in RR each job with size p_i has flow time at least $(i+2)p_i$ (since this job time-shares with at least $i+2$ jobs throughout its execution). We now lower bound p_i. Using the fact that $e^{-2x} \leq 1 - x$ for $x \leq \frac{1}{2}$, we get that $p_i = \frac{1}{i+1}\Pi_{j=1}^{i}(1-\epsilon/j) \geq \frac{1}{i+1}e^{-2\epsilon H(i)}$. Since $H(i) \leq \ln i + 1$ we get, $p_i \geq \frac{1}{i+1}(ei)^{-2\epsilon} = \Omega(i^{-(1+2\epsilon)})$. Thus $(i+2)p_i = \Omega(i^{-2\epsilon})$. This gives us that $F^p(RR, 1+\epsilon)$ is at least $\sum_i^n i^{-2\epsilon p}$ which is $\Omega(n^\delta)$ for $2\epsilon p \leq 1 - \delta$. In particular, if $p = 2$, then RR is not O(1)-competitive even with an $(1+\epsilon)$ speedup if $\epsilon < \frac{1}{4}$.

Acknowledgments: We thank Cliff Stein for helpful discussions.

10. REFERENCES

[1] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 493–500, 1997.

[2] A. Avidor, Y. Azar, and J. Sgall. Ancient and new algorithms for load balancing in the l_p norm. *Algorithmica*, 29(3):422–441, 2001.

[3] B. Awerbuch, Y. Azar, E. Grove, M. Kao, P. Krishnan, and J. Vitter. Load balancing in the l_p norm. In *IEEE Symposium on Foundations of Computer Science*, 1995.

[4] N. Bansal and K. Dhamdhere. Minimizing weighted flow time. In *Symposium on Discrete Algorithms SODA*, pages 508–516, 2003.

[5] N. Bansal, K. Dhamdhere, J. Konemann, and A. Sinha. Non-clairvoyant scheduling for minimizing mean slowdown. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, to appear, 2003.

[6] N. Bansal and M. Harchol-Balter. Analysis of srpt scheduling: Investigating unfairness. In *ACM Sigmetrics*, pages 279–290, 2001.

[7] L. Becchetti and S. Leonardi. Non-clairvoyant scheduling to minimize the average flow time on single and parallel machines. In *ACM Symposium on Theory of Computing (STOC)*, pages 94–103, 2001.

[8] L. Becchetti, S. Leonardi, A. M. Spaccamela, and K. Pruhs. Online weighted flow time and deadline scheduling. In *RANDOM-APPROX*, pages 36–47, 2001.

[9] M. Bender, S. Muthukrishnan, and R. Rajaraman. Improved algorithms for stretch scheduling. In *13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[10] A. Borodin and R. El-Yaniv. *On-Line Computation and Competitive Analysis.* Cambridge University Press, 1998.

[11] C. Chekuri and S. Khanna. Approximation schemes for preemptive weighted flow time. In *ACM Symposium on Theory of Computing (STOC)*, 2002.

[12] C. Chekuri, S. Khanna, and A. Zhu. Algorithms for weighted flow time. In *ACM Symposium on Theory of Computing (STOC)*, 2001.

[13] M. Crovella, R. Frangioso, and M. Harchol-Balter. Connection scheduling in web servers. In *USENIX Symposium on Internet Technologies and Systems*, pages 243–254, 1999.

[14] J. Edmonds. Scheduling in the dark. *Theoretical Computer Science*, 235(1):109–141, 2000.

[15] L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. In *European Symposium on Algorithms (ESA)*, pages 151–162, 1999.

[16] M. Harchol-Balter, M. Crovella, and S. Park. The case for srpt scheduling of web servers.

[17] `http://httpd.apache.org/docs/`.

[18] `http://www.netcraft.com/survey/`.

[19] B. Kalyanasundaram and K. Pruhs. Fault-tolerant scheduling. In *STOC*, 1994.

[20] B. Kalyanasundaram and K. Pruhs. Minimizing flow time nonclairvoyantly. In *IEEE Symposium on Foundations of Computer Science*, pages 345–352, 1997.

[21] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.

[22] D. Knuth. *The TeXbook*. Addison Wesley, 1986.

[23] J. Kurose and K. Ross. *Computer networking: A top-down approach featuring the Internet.* Addison-Wesley, 2002.

[24] R. Motwani, S. Phillips, and E. Torng. Nonclairvoyant scheduling. *Theoretical Computer Science*, 130(1):17–47, 1994.

[25] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. Gehrke. Online scheduling to minimize average stretch. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 433–442, 1999.

[26] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. In *ACM Symposium on Theory of Computing (STOC)*, pages 140–149, 1997.

[27] B. Schroeder and M. Harchol-Balter. Web servers under overload: how scheduling can help.

[28] A. Tanenbaum. *Operating systems: design and implementation.* Prentice-Hall, 2001.

[29] K. Thompson. Unix implementation. *The Bell System Technical Journal*, 57(6):1931–1946, 1978.