Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 359 (2006) 430–439

Theoretical Computer Science

www.elsevier.com/locate/tcs

New resource augmentation analysis of the total stretch of SRPT and SJF in multiprocessor scheduling

Wun-Tat Chan[a], Tak-Wah Lam[a], Kin-Shing Liu[a], Prudence W.H. Wong[b,*]

[a]*Department of Computer Science, University of Hong Kong, Hong Kong*
[b]*Department of Computer Science, University of Liverpool, UK*

Received 15 October 2005; received in revised form 23 May 2006; accepted 1 June 2006

Communicated by Ding-Zhu Du

Abstract

This paper studies online job scheduling on multiprocessors and, in particular, investigates the algorithms Shortest Remaining Processing Time First (SRPT) and Shortest Job First (SJF) for minimizing total stretch, where the stretch of a job is its flow time (response time) divided by its processing time. SRPT is perhaps the most well-studied algorithm for minimizing total flow time or stretch. This paper gives the first resource augmentation analysis of the total stretch of SRPT, showing that it is indeed O(1)-speed 1-competitive. This paper also gives a simple lower bound result showing that SRPT is not s-speed 1-competitive for any $s < 1.5$.

This paper also makes contribution to the analysis of SJF. Extending the work of [L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, K. Pruhs, Online weighted flow time and deadline scheduling, in: RANDOM-APPROX, 2001, pp. 36–47], we are able to show that SJF is O(1)-speed 1-competitive for minimizing total stretch. More interestingly, we find that the competitiveness of SJF can be reduced arbitrarily by increasing the processor speed (precisely, SJF is $O(s)$-speed $(1/s)$-competitive for any $s \geqslant 1$). We conjecture that SRPT also admits a similar result.

Keywords: Multiprocessor scheduling; Competitive analysis; Resource augmentation; Online algorithms

1. Introduction

We study the problem of online job scheduling for minimizing total stretch. There is a pool of $m \geqslant 1$ processors. Jobs arrive at arbitrary times, and their processing times are known only when they arrive. Jobs are sequential in nature and can be scheduled on at most one processor at a time. Preemption is allowed. The flow time (or response time) of a job is the amount of time the job spent before it is completed, and the stretch of the job is the ratio of its flow time to its required processing time. We are interested in scheduling algorithms that minimize the total stretch (or equivalently the average stretch) of the jobs. Stretch is a useful indicator of system performance, and has received a lot of attention in recent years (see, e.g., [3,5–7,11,12,19]). Roughly speaking, if the average stretch is λ, a job on average takes λ

* Corresponding author. Tel.: +44 151 794 3694; fax: +44 151 794 3715.
E-mail addresses: wtchan@cs.hku.hk (W.-T. Chan), twlam@cs.hku.hk (T.-W. Lam), ksliu@cs.hku.hk (K.-S. Liu), pwong@csc.liv.ac.uk (P.W.H. Wong).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.06.003

Table 1
SRPT and SJF using faster processors can be 1-competitive (or even better) for minimizing total stretch. Results given in this paper are marked with asterisks

	Single processor	Multiprocessors
SRPT	2-competitive [19]	14-competitive [19]
	2-speed 1-competitive *	5-speed 1-competitive *
SJF	$(1+\varepsilon)$-speed $(1+\frac{1}{\varepsilon})$-competitive [4]	$(2+2\varepsilon)$-speed $(1+\frac{1}{\varepsilon})$-competitive [4]
	2-speed 1-competitive [4,20]	$(24s)$-speed $(\frac{1}{s})$-competitive, for $s \geqslant 1$ *

times the required processing time to complete, i.e., it appears to be processed by a $1/\lambda$-speed processor. [1] Competitive analysis is often used to measure the performance of an online algorithm with respect to total stretch (or any other objective function). An online scheduling algorithm A is said to be c-competitive for any number $c > 0$ if for any input job sequence, the total stretch of the jobs as defined by A is at most c times that of the optimal offline algorithm.

Shortest Remaining Processing Time First (SRPT) is a popular online algorithm when the concern is the total flow time or total stretch. With respect to total flow time, SRPT is 1-competitive for a single processor [2]. But for multiprocessors ($m \geqslant 2$), Leonardi and Raz [17] showed that SRPT is $\Theta(\min(\log P, \log n/m))$-competitive, where n is the number of jobs and P is the ratio of the maximum possible processing time to the minimum possible processing time. To obtain better performance guarantee for multiprocessor scheduling, Phillips et al. [20] applied resource augmentation analysis (which was pioneered by Kalyanasundaram and Pruhs [16]) to SRPT, showing that SRPT is 1-competitive when using processors that are two times faster, or in short, 2-speed 1-competitive. This result means that a modest increase in the processor speed of the online scheduler can compensate its lack of future information. Recently, McCulloguh and Torng [18] further showed that SRPT is s-speed $(1/s)$-competitive for any $s \geqslant 2 - 1/m$. In a wider context, resource augmentation analysis has been found useful in a number of difficult scheduling problems (see, e.g., [8–10,13–16,20]).

Muthukrishnan et al. [19] were the first to study total stretch. They showed that SRPT is 2-competitive on a single processor and 14-competitive on multiprocessors, and no online algorithm can be 1-competitive. Chekuri et al. [12] proposed a different algorithm (called SG) that is 9.81-competitive on multiprocessors. Existing resource augmentation results are actually based on algorithms like Shortest Job First (SJF), which assigns fixed priorities to jobs independently of the schedule. On a single processor, the work of Phillips (on weighted flow time) [20] implies that an algorithm called Preemptively Schedule-by-Halves as well as SJF are 2-speed 1-competitive for minimizing total stretch. For multiprocessors, there are two algorithms known to be O(1)-speed O(1)-competitive (namely, SJF is $(2+2\varepsilon)$-speed $(1+1/\varepsilon)$-competitive [4], and IMD [1] is $(1+\varepsilon)$-speed $O(1+1/\varepsilon)$-competitive [11]). Though SRPT is believed to perform well and is known to be O(1)-competitive, no resource augmentation analysis has been given for SRPT and it is generally agreed that SRPT is more difficult for resource augmentation analysis (see, e.g., [21]). In this paper we show that SRPT is indeed 2-speed 1-competitive for minimizing total stretch on a single processor. A more elaborate analysis further reveals that SRPT is 5-speed 1-competitive on multiprocessors. This is the first result on exploiting extra speed to achieve 1-competitiveness. Table 1 gives a summary of the performance of SRPT and SJF. We also derive a simple lower bound that for any $s < 1.5$, SRPT is not s-speed 1-competitive.

Technically speaking, our analysis of SRPT is based on an observation that the optimal offline algorithm, at any time, has no more finished jobs than SRPT does, and more interestingly, each finished job of the optimal offline algorithm can be mapped to a unique finished job of SRPT with same or smaller processing time.

This paper also makes contribution to the analysis of SJF. It has been known that based on the result on weighted flow time, SJF is 2-speed 1-competitive for minimizing total stretch on a single processor, and $(2+2\varepsilon)$-speed $(1+1/\varepsilon)$-competitive on multiprocessors [4,20]. We improve the analysis of SJF on multiprocessors to show that SJF is indeed 24-speed 1-competitive and, in general, $(24s)$-speed $(1/s)$-competitive for any $s \geqslant 1$, for minimizing total stretch. We conjecture that SRPT also admits a similar result, i.e., SRPT is also $O(s)$-speed $(1/s)$-competitive.

Before moving on to the analysis of SRPT and SJF, we give a definition of these two algorithms. Suppose there are $m \geqslant 1$ processors. At any time, if there are at most m unfinished jobs, SRPT and SJF both schedule each job to a distinct

[1] A speed-s processor, where $s > 0$, can process s units of work in one unit of time.

processor; otherwise, SJF gives priority to the m jobs with the shortest processing times, and SRPT schedules the m jobs with the shortest remaining processing times. A tie is simply broken by job ID.

Formally speaking, we say that SRPT (or SJF) is s-speed c-competitive if for any job sequence, SRPT (or SJF) using m s-speed processors incurs a total stretch at most c times of that of the optimal offline algorithm using m unit-speed processors.

Organization of the paper: Section 2 gives three useful properties of an SRPT schedule regardless of processor speed. Section 3 presents a resource augmentation analysis of SRPT on multiprocessors, revealing that SRPT is 5-speed 1-competitive for minimizing total stretch. Section 4 shows a lower bound of SRPT. Section 5 analyzes the performance of SJF. Section 6 shows that SRPT is 2-speed 1-competitive on a single processor. Section 7 discusses some future work.

2. Preliminaries

In this section we give some basic definitions and three useful properties of an SRPT schedule regardless of processor speed. Let I be an input sequence of jobs to be scheduled on $m \geqslant 1$ processors. For any job $J \in I$, let $p(J)$ and $r(J)$ denote the processing time and release time of J, respectively. For any set K of jobs, $p(K)$ is defined to be $\sum_{J\in K} p(J)$. Let $x > 0$ be any number. A job J is said to be *x-large* if $p(J) > x$, and *x-small* if $p(J) \leqslant x$. Consider a schedule S for I on m processors. We assume that jobs can be preempted and later resumed at the point of preemption. The following notations are concerned with a particular time t in the schedule S.

- Let $w_t^S(J)$ and $rw_t^S(J)$ denote the processed work and the remaining work, respectively, of a job J in S at time t. Note that $w_t^S(J) + rw_t^S(J) = p(J)$. A job J is said to be *partially processed* if $0 < rw_t^S(J) < p(J)$.
- Let Q_t^S denote the set of jobs released at or before time t and unfinished at time t, and let $Q_t^S(x) \subseteq Q_t^S$ denote the set of x-small jobs in Q_t^S.
- Let $\text{Shrink}_t^S(x)$ denote the set of x-large jobs J in Q_t^S such that $rw_t^S(J) \leqslant x$ (note that any job J in $\text{Shrink}_t^S(x)$ is partially processed because $p(J) > x$).
- Let F_t^S denote the set of jobs finished at or before time t, and let $F_t^S(x) \subseteq F_t^S$ denote the set of x-small jobs in F_t^S.

When the context is clear, we will omit the superscript S in the above notations, which become $w_t(J)$, $rw_t(J)$, Q_t, $Q_t(x)$, F_t, and $F_t(x)$. Using the above definitions, the stretch of a schedule S can be expressed as

$$\int_0^\infty \sum_{J\in Q_t} \frac{1}{p(J)}\,\mathrm{d}t.$$

To ease our discussion, we use c-speed SRPT, for any $c \geqslant 1$, to denote an online scheduler running SRPT on m c-speed processors, and we let OPT denote an optimal schedule for I on m unit-speed processors. To compare the stretch of the schedule defined by c-speed SRPT and OPT, we focus on analyzing the corresponding Q_t and F_t. Hereafter, we use the notations Q_t^*, $Q_t^*(x)$, F_t^* and $F_t^*(x)$ to denote the above concepts for OPT.

Before we move on to the analysis of 5-speed SRPT, we show in the rest of this section three useful properties of an SRPT schedule regardless of processor speed. Precisely, let S denote the schedule defined by c-speed SRPT for any $c \geqslant 1$. Consider any $x > 0$. The properties are concerned with three categories of jobs defined at any time in S as follows.

- There are at most m unfinished x-large jobs whose remaining work is at most x (Lemma 1).
- While there is an x-small job J waiting (i.e., not being processed by an processor), jobs that can be scheduled include only x-small jobs or jobs in $\text{Shrink}_{r(J)}(x)$ (Lemma 2).
- The accumulated processed work on all unfinished x-small jobs is less than mx (Lemma 3).

Lemma 1. *At any time $t \geqslant 0$ and for any $x > 0$, $|\text{Shrink}_t(x)| \leqslant m$.*

Proof. We prove the lemma by contradiction. Suppose that $\text{Shrink}_t(x) = \{J_1, J_2, \cdots, J_{m'}\}$, for some $m' > m$. By definition, $rw_t(J_i) \leqslant x < p(J_i)$. Let y be a number such that $x < y < \min_{J\in\text{Shrink}_t(x)}\{p(J)\}$. For each job $J_i \in \text{Shrink}_t(x)$, let $t_i < t$ be the latest time such that $rw_{t_i}(J_i) = y$. Note that J_i must be processed by some processor at time t_i and its remaining work is strictly less than y immediately after t_i. Without loss of generality, we assume that $t_1 \leqslant \cdots \leqslant t_{m'} < t$.

Suppose that $t_{m'-k} < t_{m'-k+1} = \cdots = t_{m'}$ for some integer $k \in [1, m]$. At $t_{m'}$, the jobs $J_{m'-k+1}, \ldots, J_{m'}$ are each being processed by a processor. And, for $1 \leqslant i \leqslant m' - k$, $rw_{t_{m'}}(J_i) < y = rw_{t_{m'}}(J_{m'})$, which implies that all these J_i are also being processed at time $t_{m'}$ (because SRPT always processes jobs with smallest remaining work first). This leads to a contradiction that $m' > m$ jobs are being processed at the same time. □

Lemma 2. *Let J be any x-small job. Whenever J is waiting, S can only schedule other x-small jobs or jobs in* $\mathrm{Shrink}_{r(J)}(x)$.

Proof. Any x-large job $J' \notin \mathrm{Shrink}_{r(J)}(x)$ has $rw_{r(J)}(J') > x \geqslant p(J)$. Starting from $r(J)$, whenever J' is being processed, J is also being processed. Therefore, the remaining work of J is always less than that of J'. Thus, whenever J is waiting, J' also needs to wait as J' has more remaining work. □

Lemma 3. *At any time $t \geqslant 0$ and for any $x > 0$, $w_t(Q_t(x)) < mx$, where $w_t(Q_t(x)) = \sum_{J \in Q_t(x)} w_t(J)$.*

Intuitively, at any time t, there may be many unfinished x-small jobs, but the above lemma states that their total processed work up to time t is less than mx. To prove the lemma, we let L be the set of jobs in $Q_t(x)$ that are partially processed. Notice that $w_t(Q_t(x)) = w_t(L)$. Assume that $L = \{J_1, J_2, \ldots, J_{|L|}\}$ where $p(J_1) \geqslant p(J_2) \geqslant \cdots \geqslant p(J_{|L|})$.

Below we show that jobs in L can be partitioned into m disjoint sets $Y_1, Y_2, \cdots, Y_m$ such that for $1 \leqslant k \leqslant m$, $w_t(Y_k) < x$. Then $w_t(Q_t(x)) = \sum_{1 \leqslant k \leqslant m} w_t(Y_k) < mx$ and the upper bound follows. We construct the partition by inserting the jobs in L one by one into the m sets. Denote by $last(Y_k)$ the last job inserted to Y_k.

- Initially, set $Y_1 = \{J_1\}, Y_2 = \{J_2\}, \ldots, Y_m = \{J_m\}$.
- For $i = m + 1$ to $|L|$, insert J_i to the set Y_k with the largest $rw_t(last(Y_k))$ value.

The following lemma gives a property on $rw_t(last(Y_k))$.

Lemma 4. *Whenever a job J_i is inserted to a set Y_k, $p(J_i) \leqslant rw_t(last(Y_k))$.*

Proof. Suppose on the contrary that $rw_t(last(Y_k)) < p(J_i)$. Notice that for all $1 \leqslant z \leqslant m$, we have (1) $rw_t(last(Y_z)) \leqslant rw_t(last(Y_k))$ because J_i is inserted to Y_k; and (2) $p(J_i) \leqslant p(last(Y_z))$. Therefore, all the $m + 1$ jobs including J_i and $last(Y_1), \cdots, last(Y_m)$ have processing time at least $p(J_i)$ but remaining work at time t less than $p(J_i)$. So letting $x = \max\{rw_t(J_i), rw_t(last(Y_k))\}$, we have $x < p(J_i)$ and $|\mathrm{Shrink}_t(x)| \geqslant m+1 > m$, which is a contradiction to Lemma 1. □

Consider any Y_k. Suppose $J'_1 (= J_k), J'_2, \ldots, J'_h$, for some $h \geqslant 1$, are the jobs inserted to Y_k (in that order). Then by Lemma 4, $w_t(Y_k) = \sum_{1 \leqslant i \leqslant h} w_t(J'_i) = \sum_{1 \leqslant i < h} (p(J'_i) - rw_t(J'_i)) + w_t(J'_h) \leqslant \sum_{1 \leqslant i < h} (p(J'_i) - p(J'_{i+1})) + w_t(J'_h) = p(J'_1) - p(J'_h) + w_t(J'_h) < p(J'_1) \leqslant x$. The second last inequality holds because J'_h is not finished at time t. Therefore, we have $w_t(Q_t(x)) = \sum_{1 \leqslant k \leqslant m} w_t(Y_k) < mx$ and Lemma 3 follows.

3. Resource augmentation analysis of SRPT

In this section we show that SRPT is 5-speed 1-competitive for minimizing total stretch on $m \geqslant 2$ processors. We analyze the schedule of 5-speed SRPT, denoted by S_5 below, against OPT on a given input sequence, and in particular, we show in Lemma 6 that at any time t, S_5 outperforms OPT on finished jobs; precisely, for any $x > 0$, $p(F_t(x)) \geqslant p(F^*_t(x))$. Then we show in Lemma 7 (Section 3.2) that there is a one-to-one mapping from F^*_t to F_t such that each job $J^* \in F^*_t$ can be mapped to a unique job $J \in F_t$ with $p(J^*) \geqslant p(J)$. In other words, at any time t, we have $\sum_{J \in F_t} 1/p(J) \geqslant \sum_{J^* \in F^*_t} 1/p(J^*)$, implying that $\sum_{J \in Q_t} 1/p(J) \leqslant \sum_{J^* \in Q^*_t} 1/p(J^*)$. It is then easy to see that the total stretch of S_5 is no more than that of OPT (Theorem 8).

3.1. Outperforming the optimal schedule on finished jobs

In this section we show that 5-speed SRPT outperforms OPT on finished jobs. Consider the schedule S_5 defined by 5-speed SRPT. For any $x > 0$, a time interval is said to be a $\lambda(x)$-*interval* if at any time within the interval, there is an x-small job waiting.

Lemma 5. *Let J be a job with $p(J) = x$. Suppose that S_5 does not complete J at time $t \geqslant r(J) + x$.*

- *Then during $[r(J), t]$, S_5 schedules at least $3m(t-r(J))$ units of work on x-small jobs; and $p(F_t(x)) \geqslant p(F_{r(J)}(x)) + 2m(t - r(J))$.*
- *Furthermore, if $r(J)$ is inside a $\lambda(x)$-interval starting from $t' \leqslant r(J)$, then during $[t', t]$, the work scheduled by S_5 on x-small jobs is at least $3m(t - t')$; and $p(F_t(x)) \geqslant p(F_{t'}(x)) + 2m(t - t')$.*

Proof. During $[r(J), t]$, J incurs a waiting time longer than $t - r(J) - x/5 \geqslant \frac{4}{5}(t - r(J))$, and S_5 must process at least $5m \cdot \frac{4}{5}(t - r(J)) = 4m(t - r(J))$ units of work; otherwise, J should have been finished in S_5 at time t. By Lemma 2, while J is waiting, S_5 can only process other x-small jobs or jobs in $\text{Shrink}_{r(J)}(x)$. By Lemma 1, $|\text{Shrink}_{r(J)}(x)| \leqslant m$. Each job in $\text{Shrink}_{r(J)}(x)$, by definition, has remaining work at most x at time $r(J)$. Thus, during $[r(J), t]$, the work scheduled by S_5 on x-small jobs is at least $4m(t - r(J)) - mx \geqslant 3m(t - r(J))$.

By Lemma 3, $w_t(Q_t(x)) < mx$. During $[r(J), t]$, the work scheduled by S_5 on x-small jobs that are completed by time t is at least $3m(t - r(J)) - mx \geqslant 2m(t - r(J))$. Consider jobs in $F_t(x)$ but not in $F_{r(J)}(x)$. They are all x-small jobs scheduled by S_5 to completion during $[r(J), t]$, and their total processing time is at least the work scheduled by S_5 on them during $[r(J), t]$, i.e., at least $2m(t - r(J))$. Thus, $p(F_t(x)) \geqslant p(F_{r(J)}(x)) + 2m(t - r(J))$.

Furthermore, if $r(J)$ is inside a $\lambda(x)$-interval starting from $t' \leqslant r(J)$, we have $\text{Shrink}_{r(J)}(x) \subseteq \text{Shrink}_{t'}(x)$. During $[t', r(J)]$ and the waiting time of J, S_5 can only process x-small jobs or jobs in $\text{Shrink}_{t'}(x)$. Using the same argument above, we can conclude that during $[t', t]$, the work scheduled by S_5 on x-small jobs is at least $3m(t - t')$, and $p(F_t(x)) \geqslant p(F_{t'}(x)) + 2m(t - t')$. □

Lemma 6. *At any time $t \geqslant 0$ and for any $x > 0$, $p(F_t(x)) \geqslant p(F_t^*(x))$.*

Proof. Let $\Psi_u(x)$ denote the set of x-small jobs released before time u. We will use a property of $F_t^*(x)$ that for any time $u < t$, $p(F_t^*(x)) \leqslant p(\Psi_u(x)) + m(t - u)$.

We prove the lemma by contradiction. Suppose that $t \geqslant 0$ is the earliest time such that at time t, there is a smallest $x > 0$ such that $p(F_t(x)) < p(F_t^*(x))$. Then there must be an x-small job J with $p(J) = x$ such that at time t, J is finished in OPT but unfinished in S_5. Note that $t - r(J) \geqslant x$.

We consider two cases. First, if $r(J)$ is not within a $\lambda(x)$-interval in S_5, then at time $r(J)$ in S_5, no x-small jobs are waiting, and there are at most m unfinished x-small jobs. Thus, $p(\Psi_{r(J)}(x)) \leqslant p(F_{r(J)}(x)) + mx$, and $p(F_t^*(x)) \leqslant p(F_{r(J)}(x)) + mx + m(t - r(J)) \leqslant p(F_{r(J)}(x)) + 2m(t - r(J))$. By Lemma 5, $p(F_t(x)) \geqslant p(F_{r(J)}(x)) + 2m(t - r(J))$, and thus $p(F_t(x)) \geqslant p(F_t^*(x))$. A contradiction occurs.

Second, if $r(J)$ is within a $\lambda(x)$-interval starting from time $t' \leqslant r(J)$ in S_5, then we can upper bound $p(\Psi_{t'}(x))$ by $p(F_{t'}(x)) + mx$. Then $p(F_t^*(x)) \leqslant p(F_{t'}(x)) + 2m(t - t')$. Together with Lemma 5, we can again derive the contradiction that $p(F_t(x)) \geqslant p(F_t^*(x))$. □

3.2. 5-speed SRPT is 1-competitive

Based on Lemma 6, we can prove that 5-speed SRPT is 1-competitive. First, we show that at any time t, there is a one-to-one mapping between F_t^* and F_t.

Lemma 7. *Consider any time $t \geqslant 0$. Assume that $p(F_t(x)) \geqslant p(F_t^*(x))$ for any $x > 0$. Then there is a one-to-one mapping from F_t^* to F_t such that each job $J^* \in F_t^*$ is mapped to a unique job $J \in F_t$ with $p(J^*) \geqslant p(J)$.*

Proof. Suppose that the processing times of the jobs in F_t^* have d distinct values, denoted by $x_1 < x_2 < \cdots < x_d$. We construct a mapping from F_t^* to F_t incrementally, each time we consider jobs in F_t^* with the same processing time.

Consider all jobs in F_t^* that have the smallest processing time (i.e., equal to x_1). Given that $p(F_t(x_1)) \geqslant p(F_t^*(x_1))$, $F_t(x_1)$ must contain at least as many jobs as $F_t^*(x_1)$. Thus, each job in $F_t^*(x_1)$ can be mapped to a unique job in $F_t(x_1)$ with processing time at most x_1.

Assume that for some $k \geqslant 1$, we have constructed a mapping from $F_t^*(x_k)$ to F_t as required by Lemma 7. Next, we consider jobs in F_t^* with processing time x_{k+1}. Let $Y \subset F_t$ be the set of jobs to which jobs in $F_t^*(x_k)$ are mapped. As each job in $F_t^*(x_k)$ is mapped to a job with the same or shorter processing time, we have $p(F_t^*(x_k)) \geqslant p(Y)$.

The number of jobs in F_t^* with processing time x_{k+1} is exactly

$$\frac{p(F_t^*(x_{k+1})) - p(F_t^*(x_k))}{x_{k+1}}.$$

The number of unmapped jobs in $F_t(x_{k+1})$ is at least

$$\frac{p(F_t(x_{k+1})) - p(Y)}{x_{k+1}} \geqslant \frac{p(F_t^*(x_{k+1})) - p(F_t^*(x_k))}{x_{k+1}}$$

because $p(F_t(x_{k+1})) \geqslant p(F_t^*(x_{k+1}))$ and $p(F_t^*(x_k)) \geqslant p(Y)$. Thus, each job in F_t^* with processing time x_{k+1} can be mapped to a unique job in F_t with the same or shorter processing time. □

We are now ready to show our main theorem.

Theorem 8. *SRPT is* 5-*speed* 1-*competitive for minimizing total stretch.*

Proof. For any time $t \geqslant 0$, let Φ_t denote the set of jobs released at or before time t. (Note that Φ_t equals the union of the set of jobs released at time t and Ψ_t, the set of jobs released before time t.) The set of unfinished jobs in S_5 is $Q_t = \Phi_t - F_t$; and the set of unfinished jobs in OPT is $Q_t^* = \Phi_t - F_t^*$.

$$\text{Total stretch of 5-speed SRPT} = \int \sum_{J \in Q_t} \frac{1}{p(J)}\,\mathrm{d}t = \int \sum_{J \in \Phi_t} \frac{1}{p(J)}\,\mathrm{d}t - \int \sum_{J \in F_t} \frac{1}{p(J)}\,\mathrm{d}t.$$

By Lemma 7, each job J^* in F_t^* is mapped to a unique job in F_t with processing time at most $p(J^*)$, so we have

$$\int \sum_{J \in F_t} \frac{1}{p(J)}\,\mathrm{d}t \geqslant \int \sum_{J^* \in F_t^*} \frac{1}{p(J^*)}\,\mathrm{d}t.$$

Thus,

$$\text{total stretch of 5-speed SRPT} \leqslant \int \sum_{J \in \Phi_t} \frac{1}{p(J)}\,\mathrm{d}t - \int \sum_{J^* \in F_t^*} \frac{1}{p(J^*)}\,\mathrm{d}t = \int \sum_{J^* \in Q_t^*} \frac{1}{p(J^*)}\,\mathrm{d}t,$$

which is equal to the stretch of OPT. Hence, 5-speed SRPT is 1-competitive. □

4. Speed requirement for SRPT to be 1-competitive

In this section we give a lower bound on the speed requirement for SRPT to be 1-competitive.

Theorem 9. *For minimizing total stretch, SRPT is not c-speed* 1-*competitive for any $c<1.5$.*

Proof. Let $c = 1.5 - \varepsilon$ where $0 < \varepsilon < 1.5$. We construct a sequence of jobs such that the total stretch of c-speed SRPT schedule is larger than that of OPT. At time 0, m jobs $J_1, J_2, \ldots, J_m$ of equal processing time are released. (The processing time p will be fixed shortly.) Each job is processed in a distinct processor at time 0. Just after c-speed SRPT has started the last unit of work, i.e., at time $(p-1)/c + \delta$, for some small $0 < \delta < 1/c$, m more jobs $J_1', J_2', \ldots, J_m'$, all of which have processing time equal to 1, are released. In this case, c-speed SRPT continues processing J_i, and starts processing J_i' only after finishing all J_i.

We analyze the total stretch of c-speed SRPT and OPT. For c-speed SRPT, the stretch of J_i is $1/c$. Since J_i' is processed after J_i is finished, J_i' is finished at $p/c + 1/c$ and thus with stretch $2/c - \delta$. Therefore, the total stretch of c-speed SRPT is $m(3/c - \delta)$. On the other hand, a unit-speed schedule can start processing J_i' immediately after the job is released and then resume processing J_i; the stretch of J_i' and J_i is 1 and $(p+1)/p$, respectively. Therefore, the total stretch of OPT is at most $m(2 + 1/p)$. We can fix the value of $p > 1/(4\varepsilon/(3-2\varepsilon) - \delta)$, then we have $3/c - \delta = 3/(\frac{3}{2} - \varepsilon) - \delta > 2 + 1/p$. (Notice that $0 < \varepsilon < \frac{3}{2}$ and if we choose $\delta < 4\varepsilon/(3-2\varepsilon)$, then we can ensure p to be positive.) The total stretch of c-speed SRPT is greater than that of OPT and thus the theorem follows. □

5. Resource augmentation analysis of SJF

In this section we analyze the performance of SJF for scheduling $m \geqslant 2$ processors. We show that the total stretch of the schedule of $(24c)$-speed SJF is at most $1/c$ times the total stretch of an optimal schedule using unit-speed processors. Recall that SJF gives higher priority to jobs with shorter processing times, with tie broken by job ID.

Our analysis makes use of the result by Becchetti et al. [4] that HDF (Highest Density First) is $(2 + 2\varepsilon)$-speed $(1 + 1/\varepsilon)$-competitive for minimizing weighted flow time. If we define the weight of a job J to be $1/p(J)$, then the stretch of J is equal to the weighted flow time of J. Furthermore, HDF is equivalent to SJF (since the density of a job is defined to be its weight divided by its processing time). Thus, the work of Becchetti et al. [4] implies that SJF is $(2 + 2\varepsilon)$-speed $(1 + 1/\varepsilon)$-competitive for minimizing total stretch.

The framework of our analysis of SJF is as follows. Let $\tau = (2 + 2\varepsilon)$, for some $\varepsilon > 0$ to be fixed later, and let $c \geqslant 1$ be any number. We compare the schedules of $(c\tau)$-speed SJF and τ-speed SJF. We show that the flow time of each job in the former schedule is at most $3/c$ times of the flow time in the latter schedule. Combining with the result of Becchetti et al., we conclude that SJF is $c(2 + 2\varepsilon)$-speed $(3/c)(1 + 1/\varepsilon)$-competitive, or equivalently, $(24c)$-speed $((2 + 2\varepsilon)(1 + 1/\varepsilon)/(8c))$-competitive. Putting $\varepsilon = 1$ (so as to minimize $(2 + 2\varepsilon)(1 + 1/\varepsilon)$), we obtain the result that SJF is $(24c)$-speed $(1/c)$-competitive.

Lemma 10. *Consider any real numbers $z \geqslant z' \geqslant 1$. Given an input job sequence, denote the schedules of z-speed SJF and z'-speed SJF as S and S', respectively. At any time $t \geqslant 0$ and for any job J, we have $rw_t^S(J) \leqslant rw_t^{S'}(J)$.*

Proof. We prove the lemma by contradiction. Let t be the earliest time such that there is a job J with $rw_t^S(J) > rw_t^{S'}(J)$. If there are more than one such J, then we pick the one with the highest priority, i.e., the shortest processing time. We can assume that, at time t, J is processed by some processor in S' but not by any processor in S, otherwise, t can be smaller because $z \geqslant z'$. By the definition of SJF, as J is processed in S' at time t, there are at most $m - 1$ unfinished jobs with priority higher than J. On the other hand, since J is not processed in S at time t, there are at least m unfinished jobs with priority higher than J, and one of these m jobs must have already been finished in S'. It contradicts the assumption that t is the earliest time and J is the job with highest priority that $rw_t^S(J) > rw_t^{S'}(J)$. $\square$

Corollary 11. *Assume that $z \geqslant z' \geqslant 1$. For any job J, the flow time of J in the schedule of z-speed SJF is at most that of J in the schedule of z'-speed SJF.*

Lemma 12. *Consider a schedule S of (any speed) SJF. At the time when a job J is finished, the total remaining work of the unfinished jobs arrived before J is finished and with priority higher than J is at most $(m - 1)p(J)$.*

Proof. At the time when J is finished, there are at most $m - 1$ unfinished jobs arrived before $r(J)$ and with priority higher than J. Otherwise, J will be preempted and cannot finish at the time. Since a job with priority higher than J has processing time at most $p(J)$, the total remaining work of those jobs is at most $(m - 1)p(J)$. $\square$

Let S_τ denote the schedule of τ-speed SJF. We denote the flow time of a job J in S_τ as $flow_\tau(J)$, which can be divided into two parts, $wait_\tau(J)$ and $busy_\tau(J)$, corresponding to the amount of time J is waiting for a processor and J is being processed by a processor, respectively. Similarly, we use the notations $S_{c\tau}$, $flow_{c\tau}(J)$, $wait_{c\tau}(J)$ and $busy_{c\tau}(J)$ for the schedule of $(c\tau)$-speed SJF.

Consider any job J. Our goal is to show that $flow_{c\tau}(J) \leqslant (3/c) flow_\tau(J)$. This is done by proving $busy_{c\tau}(J) \leqslant (1/c) flow_\tau(J)$ and $wait_{c\tau}(J) \leqslant (2/c) flow_\tau(J)$. The former is straightforward because $busy_{c\tau}(J) = p(J)/(c\tau) = busy_\tau(J)/c \leqslant flow_\tau(J)/c$.

The rest of this section is devoted to showing that the work scheduled by $S_{c\tau}$ while J is waiting, denoted W below, is upper bounded by $2m\tau flow_\tau(J)$. Then it follows that $wait_{c\tau}(J) \leqslant W/(mc\tau) \leqslant (2/c) flow_\tau(J)$. Let G be the set of jobs that have ever been scheduled by $S_{c\tau}$ while J is waiting. Note that jobs in G must arrive before $r(J) + flow_{c\tau}(J)$, and they all have priority higher than J. We partition G into two subsets G_1 and G_2 such that G_1 contains jobs arriving before $r(J)$ and G_2 the rest. The work scheduled by $S_{c\tau}$ while J is waiting, i.e., W, is at most $\sum_{J' \in G_1} rw_{r(J)}^{S_{c\tau}}(J') + p(G_2)$. To relate W with the flow time of J in S_τ, we consider two sets of jobs H_1 and H_2 in the schedule S_τ.

- H_1 contains jobs with priority higher than J that arrive before $r(J)$ and are unfinished at $r(J)$ in S_τ.
- H_2 contains jobs J' with priority higher than J such that $r(J) \leqslant r(J') < r(J) + \mathit{flow}_\tau(J)$.

It is not difficult to see that $G_1 \subseteq H_1$ and $G_2 \subseteq H_2$ (see Lemma 13), and hence W can be bounded by the remaining work of H_1 in S_τ at $r(J)$ plus the processing time of H_2. Details are as follows. Consider any set K of jobs, at any time t, $rw_t^S(K)$ is defined to be $\sum_{J\in K} rw_t^S(J)$ for any schedule S.

Lemma 13. *$G_1 \subseteq H_1$ and $G_2 \subseteq H_2$. Furthermore, $rw_{r(J)}^{S_{c\tau}}(G_1) \leqslant rw_{r(J)}^{S_\tau}(H_1)$,*

Proof. Consider any job J' in G_1. By definition, J' is unfinished in $S_{c\tau}$ at $r(J)$ and has priority higher than J. By Corollary 11, J' is also unfinished in S_τ at $r(J)$. Therefore, we have $J' \in H_1$. Furthermore, by Lemma 10, $rw_{r(J)}^{S_{c\tau}}(J') \leqslant rw_{r(J)}^{S_\tau}(J')$. Together with $G_1 \subseteq H_1$, we have $rw_{r(J)}^{S_{c\tau}}(G_1) \leqslant rw_{r(J)}^{S_\tau}(H_1)$.

Consider any job J'' in G_2. By definition, $r(J) \leqslant r(J'') < r(J) + \mathit{flow}_{c\tau}(J)$. By Corollary 11, $\mathit{flow}_{c\tau}(J) \leqslant \mathit{flow}_\tau(J)$, and hence $r(J'') < r(J) + \mathit{flow}_\tau(J)$. Therefore, $G_2 \subseteq H_2$. □

Corollary 14. $W \leqslant rw_{r(J)}^{S_\tau}(H_1) + p(H_2)$.

Lemma 15 further shows that the upper bound of W is $2m\tau\,\mathit{flow}_\tau(J)$.

Lemma 15. $rw_{r(J)}^{S_\tau}(H_1) + p(H_2) \leqslant 2m\tau\,\mathit{flow}_\tau(J)$.

Proof. Let us consider how S_τ schedules the work in H_1 and H_2 starting from time $r(J)$. First, we note that the total amount of such work is exactly $rw_{r(J)}^{S_\tau}(H_1) + p(H_2)$. During $[r(J), r(J) + \mathit{flow}_\tau(J)]$, the work scheduled by S_τ is at most $m\tau\,\mathit{flow}_\tau(J)$. At time $r(J) + \mathit{flow}_\tau(J)$, S_τ may not complete all work in H_1 and H_2; yet, by Lemma 12, at the time when J is finished, all unfinished jobs arriving before J is finished and with priority higher than J have a total remaining work at most $(m-1)p(J)$, and thus, from $r(J) + \mathit{flow}_\tau(J)$ onwards, S_τ can schedule at most $(m-1)p(J)$ units of work on H_1 and H_2. In conclusion, $rw_{r(J')}^{S_\tau}(H_1) + p(H_2) \leqslant m\tau\,\mathit{flow}_\tau(J) + (m-1)p(J) \leqslant 2m\tau\,\mathit{flow}_\tau$ because $p(J) \leqslant \tau\,\mathit{flow}_\tau(J)$. □

The waiting time of the job J in $S_{c\tau}$ (i.e., $\mathit{wait}_{c\tau}(J)$) is at most $W/mc\tau$, which, by Corollary 14 and Lemma 15, is at most $(2/c)\mathit{flow}_\tau(J)$.

Corollary 16. $\mathit{flow}_{c\tau}(J) \leqslant (3/c)\mathit{flow}_\tau(J)$.

Proof. By definition, $\mathit{flow}_{c\tau}(J) = \mathit{wait}_{c\tau}(J) + \mathit{busy}_{c\tau}(J)$. The corollary follows from the facts that $\mathit{wait}_{c\tau}(J) \leqslant (2/c)\mathit{flow}_\tau(J)$ and $\mathit{busy}_{c\tau}(J) \leqslant (1/c)\mathit{flow}_\tau(J)$. □

The following theorem follows from Corollary 16 and that SJF is $(2 + 2\varepsilon)$-speed $(1 + 1/\varepsilon)$-competitive.

Theorem 17. *SJF is $(24c)$-speed $(1/c)$-competitive, for any $c \geqslant 1$.*

6. 2-speed SRPT on a single processor

In the previous sections we have analyzed SRPT and SJF in the multiprocessor setting. In this section, we show that on a single processor, SRPT admits a better analysis than that in Section 3; precisely, we show that SRPT is indeed 2-speed 1-competitive. The analysis takes a similar approach as before. First, we attempt to bound the total processing time of finished x-small jobs, yet based on one 2-speed processor, we can only derive a weaker bound, namely, at any time $t \geqslant 0$ and for any $x > 0$, $p(F_t(x)) + x > p(F_t^*(x))$ (see Lemma 19). This weaker result is still sufficient for establishing a one-to-one mapping between the finished x-small jobs of OPT and that of 2-speed SRPT exists (Lemma 20). Following the same argument as in the proof of Theorem 8, we can then conclude that SRPT is 2-speed 1-competitive.

Consider the schedule S defined by 2-speed SRPT. The following lemma helps to establish the bound in Lemma 19.

Lemma 18. *Let J be a job with $p(J) = x$. Suppose that S does not complete J at time $t \geqslant r(J) + x$ and $t' < r(J)$ is the latest time in S that there is no unfinished x-small job. Then during $[t', t]$, S schedules at least $2(t - t') - x$ units of work on x-small jobs*; *and $p(F_t(x)) + x > p(F_{t'}(x)) + t - t'$.*

Proof. Throughout $[t', t]$, there is at least one unfinished x-small job. With a single processor, any x-large jobs released after t' would not be processed during $[t', t]$; in other words, during $[t', t]$, S can only schedule unfinished x-small jobs or jobs in $\mathrm{Shrink}_{t'}(x)$. Since, by Lemma 1, $|\mathrm{Shrink}_{t'}(x)| \leqslant 1$, during $[t', t]$, the work scheduled by S on x-small jobs is at least $2(t - t') - x$.

Since, by Lemma 3, $w_t(Q_t(x)) < x$, during $[t', t]$, the work scheduled by S on x-small jobs that are completed by time t is strictly more than $2(t - t') - 2x$. Consider jobs in $F_t(x)$ but not in $F_{t'}(x)$. They are all x-small jobs scheduled by S to completion during $[t', t]$, and their total processing time is at least the work scheduled by S on them during $[t', t]$, i.e., strictly more than $2(t - t') - 2x$. Thus, $p(F_t(x)) + x > p(F_{t'}(x)) + 2(t - t') - 2x + x \geqslant p(F_{t'}(x)) + t - t'$ as $t - t' \geqslant x$. □

Lemma 19. *At any time $t \geqslant 0$ and for any $x > 0$, $p(F_t(x)) + x > p(F_t^*(x))$.*

Proof. We prove the lemma by contradiction. Suppose that t is the earliest time such that at time t, there is a smallest $x > 0$ such that $p(F_t(x)) + x \leqslant p(F_t^*(x))$. There must be an x-small job J with $p(J) = x$ such that J is finished in OPT at time t but not finished in S. Let $t' < r(J)$ be the latest time in S that there is no unfinished x-small job. Note that $t - t' > t - r(J) \geqslant x$. By Lemma 18, $p(F_t(x)) + x > p(F_{t'}(x)) + t - t'$.

On the other hand, since at time t' there is no unfinished x-small job, $p(F_{t'}(x)) = p(\Psi_{t'}(x))$ where $\Psi_{t'}(x)$ denotes the set of x-small jobs released before time t'. Using a property of $F_t^*(x)$ that $p(F_t^*(x)) \leqslant p(\Psi_{t'}(x)) + t - t'$, we have $p(F_t^*(x)) \leqslant p(F_{t'}(x)) + t - t' < p(F_t(x)) + x$, which is a contradiction. □

Lemma 20. *Consider any time $t \geqslant 0$. Assume that $p(F_t(x)) + x > p(F_t^*(x))$, for any $x > 0$. Then there is a one-to-one mapping between F_t^* and F_t such that each job $J^* \in F_t^*$ can be mapped to a unique job $J \in F_t$ with $p(J^*) \geqslant p(J)$.*

Proof. Suppose that the processing times of the jobs in F_t^* have d distinct values, denoted by $x_1 < x_2 < \cdots < x_d$. We construct a mapping from F_t^* to F_t incrementally, each time we consider jobs in F_t^* with the same processing time.

Consider all jobs in F_t^* that have the smallest processing time (i.e., equal to x_1). Given that $p(F_t(x_1)) + x_1 > p(F_t^*(x_1))$, $F_t(x_1)$ contains strictly more than $p(F_t^*(x_1))/x_1 - 1$ jobs, which is the number of jobs in $F_t^*(x_1)$ minus 1. Since both the number of jobs in $F_t(x_1)$ and $F_t^*(x_1)$ are integers, $F_t(x_1)$ must contain at least as many jobs as $F_t^*(x_1)$. Thus, each job in $F_t^*(x_1)$ can be mapped to a unique job in $F_t(x_1)$ with processing time at most x_1.

Assume that for some $k \geqslant 1$, we have constructed a mapping from $F_t^*(x_k)$ to F_t as required by Lemma 20. Next, we consider jobs in F_t^* with processing time x_{k+1}. Let $Y \subset F_t$ be the set of jobs to which jobs in $F_t^*(x_k)$ are mapped. As each job in $F_t^*(x_k)$ is mapped to a job with the same or shorter processing time, we have $p(F_t^*(x_k)) \geqslant p(Y)$. The number of jobs in F_t^* with processing time x_{k+1}, denoted by K is exactly

$$\frac{p(F_t^*(x_{k+1})) - p(F_t^*(x_k))}{x_{k+1}}.$$

On the other hand, the number of unmapped jobs in $F_t(x_{k+1})$, denoted by U is at least

$$\frac{p(F_t(x_{k+1})) - p(Y)}{x_{k+1}} > \frac{p(F_t^*(x_{k+1})) - p(F_t^*(x_k))}{x_{k+1}} - 1$$

because $p(F_t(x_{k+1})) + x_{k+1} > p(F_t^*(x_{k+1}))$ and $p(F_t^*(x_k)) \geqslant p(Y)$. Since both K and U are integers, we have $U \geqslant K$. Thus, each job in F_t^* with processing time x_{k+1} can be mapped to a unique job in F_t with the same or shorter processing time. □

Following the same argument as in the proof of Theorem 8, we have the following theorem.

Theorem 21. *SRPT is 2-speed 1-competitive for minimizing total stretch on a single processor.*

7. Concluding remarks

In this paper we have studied online job scheduling on multiprocessors and showed that, with respect to total stretch, SRPT is 5-speed 1-competitive and not c-speed 1-competitive for any $c < 1.5$. An interesting direction is to close the gap between the speed requirement for SRPT to be 1-competitive. We also show that SJF is $(24c)$-speed $(1/c)$-competitive, for any $c \geqslant 1$. We believe that SRPT also admits a similar result as SJF, and, in particular, we conjecture that SRPT is $(10c)$-speed $(1/c)$-competitive for large enough c (say, $c \geqslant 3$). SJF is simpler than SRPT in terms of implementation. Intuitively, SRPT is more adaptive than SJF in the sense that SRPT also takes into account the work that has already been done on the jobs. A job that has been processed would get a higher priority, thus, the set of unfinished small jobs could be minimized. SJF does not have this merit. Our results that SRPT is 5-speed 1-competitive while SJF is 24-speed 1-competitive on multiprocessors also suggest this is the case.

It is also interesting to analyze the performance of SRPT and SJF when the online algorithm is given extra processors instead of extra speed, and to determine whether 1-competitiveness can be achieved. Another open problem is to derive a c-speed 1-competitive online algorithm for minimizing weighted flow time on multiprocessors. Note that both SRPT and SJF require job migration. A further direction is to consider non-migratory algorithms, i.e., once a job is assigned to a processor, it cannot be migrated to other processors, though it may be preempted.

References

[1] N. Avrahami, Y. Azar, Minimizing total flow time and total completion time with immediate dispatching, in: Symp. on Parallelism in Algorithms and Architectures, 2003, pp. 11–18.

[2] K.R. Baker, Introduction to Sequencing and Scheduling, Wiley, New York, 1974.

[3] N. Bansal, K. Pruhs, Server scheduling in the L_p norm: a rising tide lifts all boat, in: Proc. Symp. on Theory of Computing, 2003, pp. 242–250.

[4] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, K. Pruhs. Online weighted flow time and deadline scheduling, in: RANDOM-APPROX, 2001, pp. 36–47.

[5] L. Becchetti, S. Leonardi, S. Muthukrishnan, Scheduling to minimize average stretch without migration, in: Proc. Symp. of Discrete Algorithms, 2000, pp. 548–557.

[6] M.A. Bender, S. Chakrabarti, S. Muthukrishnan, Flow and stretch metrics for scheduling continuous job streams, in: Proc. Symp. of Discrete Algorithms, 1998, pp. 270–279.

[7] M.A. Bender, S. Muthukrishnan, R. Rajaraman. Improved algorithms for stretch scheduling, in: Proc. Symp. of Discrete Algorithms, 2002, pp. 762–771.

[8] M. Brehob, E. Torng, P. Uthaisombut, Applying extra-resource analysis to load balancing, J. Scheduling 3 (5) (2000) 273–288.

[9] H.L. Chan, T.W. Lam, K.K. To, Non-migratory online deadline scheduling on multiprocessors, in: Proc. Symp. Discrete Algorithms, 2004, pp. 970–979.

[10] W.T. Chan, T.W. Lam, H.F. Ting, P.W.H. Wong, A unified analysis of hot video schedulers, in: Proc. Symp. on Theory of Computing, 2002, pp. 179–188.

[11] C. Chekuri, A. Goel, S. Khanna, A. Kumar, Multi-processor scheduling to minimize flow time with ε resource augmentation, in: Proc. Symp. on Theory of Computing, 2004, pp. 363–372.

[12] C. Chekuri, S. Khanna, A. Zhu, Algorithms for minimizing weighted flow time, in: Proc. Symp. on Theory of Computing, 2001, pp. 84–93.

[13] M. Chrobak, L. Epstein, J. Noga, J. Sgall, R. van Stee, T. Tichý, N. Vakhania, Preemptive scheduling in overloaded systems, in: Internat. Colloq. on Automata, Languages and Programming, 2002, pp. 800–811.

[14] J. Edmonds, Scheduling in the dark, in: Proc. Symp. on Theory of Computing, 1999, pp. 179–188.

[15] B. Kalyanasundaram, K. Pruhs, Maximizing job completions online, in: ESA, 1998, pp. 235–246.

[16] B. Kalyanasundaram, K. Pruhs, Speed is as powerful as clairvoyance, J. ACM 47 (4) (2000) 617–643.

[17] S. Leonardi, D. Raz, Approximating total flow time on parallel machines, in: Proc. Symp. on Theory of Computing, 1997, pp. 110–119.

[18] J. McCullough, E. Torng, SRPT optimally utilizes faster machines to minimize flow time, in: Proc. Symp. of Discrete Algorithms, 2004, pp. 350–358.

[19] S. Muthukrishnan, R. Rajaraman, A. Shaheen, J. Gehrke. Online scheduling to minimize average stretch, in: Proc. Symp. on Foundations of Computer Science, 1999, pp. 433–442.

[20] C.A. Phillips, C. Stein, E. Torng, J. Wein, Optimal time-critical scheduling via resource augmentation (extended abstract), in: Proc. Symp. on Theory of Computing, 1997, pp. 140–149.

[21] K. Pruhs, J. Sgall, E. Torng, Online scheduling, in: J. Leung (Ed.), Handbook of Scheduling: Algorithms, Models and Performance Analysis, CRC Press, Boca Raton, FL, 2004, pp. 15-1–15-41.