Online Non-Preemptive Scheduling to Optimize Max Stretch on a Single Machine

Pierre-Francois Dutot, Erik Saule, Abhinav Srivastav, Denis Trystram

Online Non-Preemptive Scheduling to Optimize Max Stretch on a Single Machine

Pierre-Francois Dutot[1], Erik Saule[2], Abhinav Srivastav[1], Denis Trystram[1]

University of Grenoble Alpes, France[1]
Dept. of Computer Science, University of North Carolina at Charlotte, USA[2]
{srivasta,trystram,pfdutot}@imag.fr, esaule@uncc.edu

Abstract. We consider in this work a classical online scheduling problem with release times on a single machine. The quality of service of a job is measured by its *stretch*, which is defined as the ratio of its response time over its processing time. Our objective is to schedule the jobs non-preemptively in order to optimize the maximum stretch. We present both positive and negative theoretical results. First, we provide an online algorithm based on a waiting strategy which is $(1+\frac{\sqrt{5}-1}{2}\Delta)$-competitive where Δ is the upper bound on the ratio of processing times of any two jobs. Then, we show that no online algorithm has a competitive ratio better than $1+\frac{\sqrt{5}-1}{2}\Delta$. The proposed algorithm is asymptotically the best algorithm for optimizing the maximum stretch on a single machine.

1 Introduction

Scheduling independent jobs that arrive over time is a fundamental problem that arises in many applications. Often, the aim of a scheduler is to optimize some function(s) that measure the performance or quality of service delivered to the jobs. The most popular and relevant metrics include throughput maximization, minimization of maximum or average completion times and optimizing the flow time [1]. These metrics have received a lot of attention over the last years in various scenarios: on single or multiple machines, in online or offline settings, in weighted or unweighted settings, etc. One of the most relevant performance measures in job scheduling is the *fair* amount of time that the jobs spend in the system. This includes the waiting time due to processing some other jobs as well as the actual processing time of the job itself. Such scheduling problems arise for instance while scheduling jobs in parallel computing platforms. The *stretch* is the factor by which a job is slowed down with respect to the time it takes on an unloaded system [2].

Here, we are interested in scheduling a stream of jobs to minimize the maximum stretch (*max-stretch*) on a single machine. This problem is denoted as $1|r_i, online|S_{max}$ in the classical 3-fields notation of scheduling problems [3]. While this problem admits no constant approximation algorithm in the offline case [2], interesting results can be derived by introducing an instance-dependent parameter Δ: the ratio between the largest and the smallest processing time in the instance.

We show using an adversary technique, that no online algorithm can achieve a competitive ratio better than $1+\alpha\Delta$ where $\alpha = \frac{\sqrt{5}-1}{2}$ (the golden ratio). This improves upon the previously best known lower bound of $\frac{1+\Delta}{2}$ by Saule *et al.* [4].

Based on the observation that no greedy algorithm can reach this lower bound, we designed Wait-Deadline Algorithm (WDA) which enforces some amount of waiting time for large jobs, before they can be scheduled. We prove that WDA has a competitive ratio of $1 + \alpha\Delta$, which improves upon the best known competitive ratio of Δ achieved by First-Come First-Served and presented by Legrand *et al.* [5].

The competitive ratio of WDA and the lower bound on best achievable competitive ratio are both asymptotically equal to $1+\alpha\Delta$, for large values of Δ. In other words, this paper essentially closes the problem of minimizing max-stretch on a single machine.

This paper is organized as follows. Section 2 defines the problem formally and summarizes the main positive and negative results that relate to optimizing the maximum stretch objective. Section 3 provides lower bounds on the competitive ratio of deterministic algorithms for both objectives and it indicates that algorithms with good competitive ratios have to wait before executing large jobs. Section 4 presents the wait-deadline algorithm (WDA). Then we provide the corresponding detailed analysis for the competitive ratio of max-stretch in Section 5. Finally, we provide concluding remarks in Section 6 and discuss future issues for the continuation of this work.

2 Problem Definition and Related Works

We study the problem of scheduling on a single machine n independent jobs that arrive over time. A scheduling instance is specified by the set of jobs J. The objective is to execute the continuously arriving stream of jobs. We consider the clairvoyant version of the problem where the processing time p_i of each job i is only known at its release time r_i. Without loss of generality, we assume that the smallest and largest processing times are equal to 1 and Δ, respectively.

In a given schedule σ_i, C_i and S_i denote the start time, completion time and stretch of job i, respectively where $S_i = \frac{C_i - r_i}{p_i}$. We are interested in minimizing $S_{max} = \max_{j \in J} S_j$.

An online algorithm is said to be ρ-competitive if the worst case ratio (over all possible instances) of the objective value of the schedule generated by the algorithm is no more that ρ times the performance of the optimal (offline clairvoyant) algorithm [6].

Bender *et al.* introduced the stretch performance objective to study the fairness for HTTP requests arriving at web servers [2]. They showed that the problem of optimizing *max-stretch* in a non-preemptive offline setting cannot be approximated within a factor of $\Omega(n^{1-\epsilon})$, unless $P = NP$. They also showed that any online algorithm has a competitive ratio in $\Omega(\Delta^{\frac{1}{3}})$. Finally, they provided an online preemptive algorithm using the classical EDF strategy (*earliest deadline first*) and showed that it is $O(\sqrt{\Delta})$ competitive.

Later, Legrand *et al.* showed that the First-Come First-Served algorithm (FCFS) is Δ-competitive for the *max-stretch* problem on a single machine [5]. Since preemption is not used in FCFS, the above bound is also valid in the non-preemptive case. They also showed that the problem of optimizing *max-stretch* on a single machine with preemption cannot be approximated within a factor of $\frac{1}{2}\Delta^{\sqrt{2}-1}$. Saule *et al.* showed

that all approximation algorithms for the single machine problem and m parallel machine of optimizing max-stretch cannot have a competitive ratio better than $\frac{1+\Delta}{2}$ and $(1 + \frac{\Delta}{m+1})/2$, respectively [4]. Bansal *et al.* [7], Golovin *et al.* [8], Im *et al.* [9] and Anand *et al.* [10] studied similar problems with resource augmentation.

3 Lower Bounds on Competitive Ratios for Max-Stretch

Observation 1. *Any greedy algorithm for scheduling jobs on a single machine has a competitive ratio of at least Δ for max-stretch.*

For non-preemptive schedules, it is easy to prove that any greedy algorithm is at least Δ-competitive using the following adversary technique. At time 0 a large job of processing time Δ arrives. Any greedy algorithm schedules it immediately. At time ϵ, a small job of processing time 1 is released. Since preemption is not allowed, the greedy algorithm can only schedule the small job at time $t = \Delta$ and thus $S_{max} \approx \Delta$. The optimal algorithm finishes the small job first and hence has a stretch close to 1; more precisely of $S^* = \frac{\Delta+\epsilon}{\Delta}$.

Hence, for an improved bound, the algorithm should incorporate some waiting time strategies. We show below a lower bound on the competitive ratio of such algorithms using a similar adversary technique.

Theorem 2. *There is no ρ-competitive non-preemptive algorithm for optimizing* max-stretch *for any fixed $\rho < \frac{\sqrt{5}-1}{2}\Delta$.*

Proof. Let ALG be any scheduling algorithm. Consider the following behaviour of the adversary. At time 0 a job of size Δ is released. On the first hand, if ALG schedules this job of size Δ at time t such that $0 \leq t \leq \frac{\sqrt{5}-1}{2}\Delta$, then the adversary sends a job of size 1 at time $t + \epsilon$ where $0 < \epsilon \ll 1$. In this case, ALG achieves a max stretch of $S_{max} = \Delta + 1$ while the optimal schedule has a max stretch of $S^*_{max} = \frac{t+1}{\Delta} + 1$. Therefore, the competitive ratio of ALG is greater than (or equal to) $\frac{\sqrt{5}-1}{2}\Delta$, for sufficiently large values of Δ. On the other hand if $\Delta > t > \frac{\sqrt{5}-1}{2}\Delta$, then the adversary sends a job of size 1 at time Δ. ALG reaches a max-stretch of $S_{max} = t + 1$ while the optimal solution has a max-stretch of $S^*_{max} = 1$. Hence, ALG has a competitive ratio greater than (or equal to) $\frac{\sqrt{5}-1}{2}\Delta$. Lastly, if ALG schedules the job at time t such that $t \geq \Delta$, then the adversary releases a job of size 1 at time $t + \epsilon$, where $0 < \epsilon \ll 1$. The competitive ratio of ALG is greater than $\frac{\sqrt{5}-1}{2}\Delta$ times the optimal schedule, since ALG achieves a max-stretch of $S_{max} = \Delta + 1$ while the optimal schedule has a max-stretch of $S^*_{max} = 1$. □

4 The Wait-Deadline Algorithm (WDA) for Streams of Jobs

We design an online non-preemptive algorithm for optimizing max-stretch on a single machine. To develop the intuition, we briefly consider the case where all the jobs have been released. The feasibility of scheduling the set of jobs within a given maximum stretch *S* can be easily determined since the stretch formula sets a deadline for each job.

Data: Ready queue Q_R at time t
Result: Job to be scheduled at time t
Perform binary search on *max-stretch* to find the appropriate deadline to schedule all the jobs of Q_R;
Store the *max-stretch* estimate as a lower bound for the next binary search;
Return the job of Q_R with the earliest deadline where ties are broken according to the processing time of the job (the shortest job is returned);

Algorithm 1: Job selection in WDA

Knowing these deadlines, the best order of execution for the jobs is determined by the Earliest Deadline First (EDF) algorithm which schedules the jobs as soon as possible in the order of non-decreasing deadlines. EDF is known to schedule all released jobs before their deadlines on a single machine if such a schedule exists [1].

In the online setting, these deadlines cannot be computed in advance. Our algorithm emulates these deadlines in two ways: firstly by holding the large jobs for a fixed amount of time to avoid small worst cases as explained below, secondly by computing a feasible deadline for the currently available jobs and using it to select the next one to start.

Observation 1 indicates that any algorithm with a competitive ratio better than Δ for max-stretch must wait for some time before it starts scheduling large jobs due to the non-clairvoyant nature of arrival times of the jobs. Waiting strategies have been studied for the problem of minimizing weighted completion time[11, 12]. To best our knowledge, this is the first work which studies waiting time strategies in the context of flow time. As stated before, our algorithm also needs to maintain an estimate of the *max-stretch* and adjust this estimate whenever EDF can not produce a feasible schedule.

We now describe the Wait-Deadline algorithm (WDA). We classify the jobs into two sets, namely *large set* and *small set* (denoted by J_{large} and J_{small}, respectively), based on their processing time. More specifically, $J_{small} = \{i \in J : 1 \leq p_i \leq 1+\alpha\Delta\}$ and $J_{large} = \{i \in J : 1 + \alpha\Delta < p_i \leq \Delta\}$

We maintain two separate queues : the *Ready queue* (denoted by Q_R) and the *Wait queue* (denoted by Q_W). Whenever a job $i \in J_{small}$ is released, it is placed directly into the *Ready queue*. On the other hand, when a job $i \in J_{large}$ is released, it is initially placed in the *Wait queue* for αp_i units of time and then moved to the *Ready queue*.

Our algorithm is based on three kinds of events: (i) a job is released, (ii) a waiting period ends and (iii) a job ends. Whenever an event occurs the queues are updated, then if the *Ready queue* is not empty and the machine is idle, a job is selected as depicted in the job selection pseudo-code in Algorithm 1.

Intuitively, we modify the release time of every job $i \in J_{large}$ to a new value $r_i + \alpha p_i$. Let t be the time at which the machine becomes idle. Then the algorithm sets the deadline $d_i(t)$ for each job $i \in Q_R$ where $d_i(t) = r_i + S(t)p_i$ and $S(t)$ is the estimated *max-stretch* such that all the jobs in Q_R can be completed. Note that the deadline $d_i(t)$ uses the original release time r_i rather than the modified release date. For already released jobs, $S(t)$ can be computed in polynomial time using a binary search similarly to the technique used in [2]. The upper bound for the binary search can be derived from the FCFS schedule, while 1 is a natural lower bound at time $t = 0$. At any later time

Data: Q_R and Q_W are initially empty sets
Result: An online schedule
Wait for events to occur.
Let t be the time at which events occured.
while *At least one event occuring at time* t *has not been processed* **do**
  switch *Event* **do**
    case *Job* i *has been released*
      if *the new job is in* J_{small} **then**
        Update Q_R.
      else
        Create a new event at time $t + \alpha p_i$ and update Q_W.
    case *Job* i *finished its waiting period*
      Remove i from Q_W and add it to Q_R.
    case *Job* i *finished its execution*
      Nothing special to do in this case for Q_R and Q_W.
if $Q_R \neq \emptyset$ *and the machine is idle* **then**
  Select a new job to execute using Algorithm 1 and remove it from Q_R.
Return to the first line to wait for the next time instant when events occur.

Algorithm 2: Wait-Deadline algorithm

$t > 0$, whenever a job has to be selected for execution, WDA uses the previous stretch estimate as a lower bound for the new binary search. As indicated in Algorithm 1, the job with the earliest deadline is scheduled. Note that $S(t)$ is increasing with respect to time t. We also assume that Δ is already known to WDA, which is a common hypothesis for online scheduling algorithms. The entire procedure is summarized in Algorithm 2.

Before we start with the competitive analysis, remember that $\alpha = \frac{\sqrt{5}-1}{2}$. Indeed Theorem 2 suggests that for an instance of two jobs with size 1 and Δ, it is optimal to wait for $\alpha\Delta$ time units before the job of size Δ is scheduled. When the size of the jobs can take any values between 1 and Δ, the partitioning of jobs in J_{small} and J_{large} ensures that small jobs can be scheduled as soon as they arrive while large jobs wait a fraction α of their processing time before they can be scheduled.

5 WDA is $(1 + \alpha\Delta)$-competitive for Max-Stretch

5.1 General Framework

Our goal is to show that WDA is $(1 + \alpha\Delta)$-competitive for the non-preemptive *max-stretch* problem. It can been seen that the local-competitiveness techniques used in preemptive cases do not work for our algorithm. Therefore, our approach has rather more of the combinatorial flavour.

We denote WDA the schedule produced by our algorithm and OPT some fixed optimal schedule. For the rest of this analysis, a supercript of $*$ indicates that the quantities in question refer OPT. We use r'_i to denote the modified released time of job i, that is $r'_i = r_i$ if job $i \in J_{small}$, otherwise $r'_i = r_i + \alpha p_i$. Moreover $d_i(t)$ denotes the estimated deadline of job i at time t *i.e.*, $d_i(t) = r_i + S(t)p_i$.

Let z be the job in WDA that attains the *max-stretch* among the jobs in J. We remove all jobs from the instance J that are released after the start of job z without

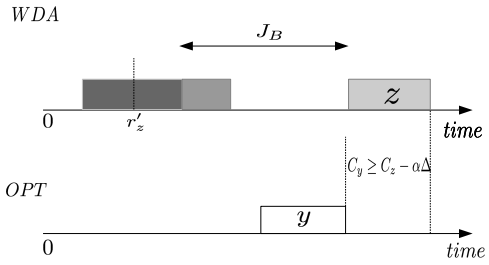

Fig. 1. Representation of jobs z and y in WDA and OPT schedule, respectively

changing the S_z and without increasing the optimal stretch. Similarly, we also remove the set of jobs that are scheduled after the job z in WDA, without changing S_z and without increasing the optimal stretch. Therefore, we assume, without loss of generality, that z is the latest job in J that is processed in WDA.

Definition 3. *We define the set of jobs* Before z*, denoted by J_B, as the set of jobs that are scheduled during the interval $[r'_z, \sigma_z)$, that is: $J_B = \{i \in J : r'_z \leq \sigma_i < \sigma_z\}$*

Property 4. *For all jobs in set* Before z*, at their start times, the deadlines of jobs are at most the deadline of job z. More formally, $d_i(\sigma_i) \leq d_z(\sigma_i) : \forall i \in J_B$*

This simply stems from the fact that the job i starting at time $t = \sigma_i$ is selected because its deadline is the earliest.

Property 5. *The schedule WDA ensures that $\forall i \in J$ the machine is busy for during time interval $[r'_i, C_i)$.*

As soon as a job is completed, an event will be generated and a new job is selected to run if Q_R is not empty. Job i is in Q_R from its modified release date r'_i until its starting time σ_i.

Our general approach is to relate the stretch of job z with the stretch of another job in the optimal schedule. The completion time of job z in WDA can be written as $C_z = r_z + S_z p_z$.

In the optimal schedule OPT, there is a job which completes at or after time $C_z - \alpha\Delta$. This is due to the fact that $\alpha\Delta$ is the maximum difference between the makespan of schedules WDA and OPT. In the rest of this analysis, we denote such a job by y (refer to Figure 1). Hence, the completion time of job y can be written as $C^*_y = r_y + S^*_y p_y \geq r_z + S_z p_z - \alpha\Delta$. Isolating S_z in the previous equation, we get:

$$S_z \leq S^*_y \left(\frac{p_y}{p_z}\right) + \frac{r_y - r_z}{p_z} + \frac{\alpha\Delta}{p_z} \tag{1}$$

Theorem 6. *WDA is $(1 + \alpha\Delta)$-competitive for the problem of minimizing max-stretch non-preemptively.*

The proof is constructed mainly in three separate parts: Lemma 7, Lemma 11 and Lemma 17. Each part mostly relies on refining Equation 1 in different cases. They are

devised based on ratio of processing time of job z and job y, as defined earlier. We further divided them into few sub cases depending upon the execution time of job y in WDA. In most of the sub cases, the lower bound on *max-stretch* are different and are derived using tricky mathematical arguments. To elaborate the proof more specifically, Lemma 7 considers the case when $p_y \leq p_z$; Lemma 11 consider the case when $p_z < p_y \leq (1+\alpha\Delta)p_z$; Lastly, Lemma 17 considers the case when $(1+\alpha\Delta)p_z < p_y$.

Frequently, we refer to the intermediate stretch at time t. As aforementioned, we use the notation $S(t)$ to refer to the intermediate maximum stretch at time t such that all jobs in the *Ready queue* can be scheduled within their respective deadlines. Note that $S(\sigma_i) \geq S_i$ for all job $i \in J$.

5.2 Proving the bound when $p_y \leq p_z$

Lemma 7. *If $p_y \leq p_z$, then $S_z \leq S^*_y + \alpha\Delta$.*

Proof. We consider two cases:

1. *Suppose $y \in J_B$.* Then Property 4 implies that $r_y + S(\sigma_y)p_y \leq r_z + S(\sigma_y)p_z$. Since the stretch of job z is greater than the intermediate stretch at any time, we have $S(\sigma_y) \leq S_z$, which leads to $r_y - r_z \leq S_z(p_z - p_y)$. Substituting this inequality in Equation 1 we get,

$$S_z \leq S^*_y \left(\frac{p_y}{p_z}\right) + \left(1 - \frac{p_y}{p_z}\right) S_z + \frac{\alpha\Delta}{p_z}$$
$$S_z \leq S^*_y + \frac{\alpha\Delta}{p_y} \qquad \leq S^*_y + \alpha\Delta$$

2. *Suppose $y \notin J_B$.* Let δ be a binary variable such that it is 0 when job z belongs to class J_{small}, otherwise it is 1. Then the modified release time of job z can we re-written as $r'_z = r_z + \delta\alpha p_z$. The start time of job y is earlier than the modified released time of job z, that is $r_y \leq \sigma_y < r'_z$. This implies that $r_y < r_z + \delta\alpha p_z$. Substituting this inequality in Equation 1 we get,

$$S_z \leq S^*_y \left(\frac{p_y}{p_z}\right) + \frac{\alpha\Delta}{p_z} + \delta\alpha \qquad \leq S^*_y + \frac{\alpha\Delta}{1+\delta\alpha\Delta} + \delta\alpha \qquad \leq S^*_y + \alpha\Delta$$

When $\delta = 1$, the last inequality follows from that fact that $\frac{\alpha\Delta}{1+\alpha\Delta} + \alpha < \alpha\Delta$ when $\Delta \geq 2$.

□

5.3 Proving the bound when $p_z < p_y \leq (1+\alpha\Delta)p_z$

Observation 8. *In WDA, there does not exist a job i such that job z is released no later than job i and the processing time of job i is more than that of job z. More formally, $\nexists i \in J : r_i \geq r_z$ and $p_i > p_z$.*

For the remaining cases, it follows that job z is processed before job y in OPT, $p_z < p_y$ and $r_y < r_z$. Before moving on to analysis of such cases, we define the notion of *limiting* jobs which play a crucial role in the analysis to follow.

Definition 9. *We say that job i* limits *job j if the following statements are true.*

- *processing time of job i is more than that of job j, $p_i > p_j$*
- *job i is scheduled at or after the modified released time of job j, both in WDA and OPT*
- *job i is processed earlier than job j in WDA, $\sigma_i < \sigma_j$*
- *job j is processed earlier than job i in OPT, $\sigma^*_j < \sigma^*_i$*

Property 10. *If i* limits *j then the stretch of job i in WDA is at least $1 + \frac{p_i}{p_j} - \frac{p_j}{p_i}$.*

Now we have all the tools to show the bound for *max-stretch* in the case where $p_y \leq (1 + \alpha\Delta)p_z$.

Lemma 11. *If $p_z < p_y$ and $p_y \leq (1 + \alpha\Delta)p_z$ then $S_z \leq S^*(1 + \alpha\Delta)$.*

Proof. Suppose that the completion time of job z in schedule WDA is no later than the completion time of job y in OPT, that is $C^*_y \geq C_z$. Similar to Equation 1, the relationship between the stretch of job z in WDA and the stretch of job y in OPT can be written as $S_z \leq S^*_y \frac{p_y}{p_z} + \frac{r_y - r_z}{p_z}$. From Observation 8, it follows that job y is released earlier than job z, *i.e.* $r_y - r_z \leq 0$. Thus combining both inequalities, we have $S_z \leq S^*_y \frac{p_y}{p_z} \leq S^*_y(1 + \alpha\Delta) \leq S^*(1 + \alpha\Delta)$. Therefore, we assume $C^*_y < C_z$ for the rest of this proof. We further split the analysis in three separate cases.

Case A: *Job $y \in J_B$.* Observe that the start time of job y is at or after the modified release time of job z *i.e.* $\sigma_y \geq r'_z$. Applying property 4, we have $r_y + S(\sigma_y)p_y \leq r_z + S(\sigma_y)p_z$. Since $p_y > p_z$ and the stretch of any job is at least 1, we can re-write the above inequality as $r_y - r_z \leq p_z - p_y$. Using this inequality in Equation 1 along with the fact that $p_y \leq (1 + \alpha\Delta)p_z$ proves that the bound holds in this case.

Case B: *Job $y \notin J_B$ and $C_y \leq r'_z$.* The assumption $C_y \leq r'_z$ implies that $r_y + S_y p_y \leq r_z + \delta\alpha p_z$ where $\delta = 0$ if $z \in J_{small}$ or 1 otherwise. Since the stretch of job y is greater than 1 or $1 + \alpha$, depending upon class of job y, job y is released at least p_y time units earlier than job z, that is $r_z - r_y \geq p_y$. Using this inequality with Equation 1 proves that the bound holds in this case.

Case C: *Job $y \notin J_B$ and $C_y > r'_z$.* Since $C^*_y < C_z$, there exists a job k such that $[\sigma_k, C_k) \subseteq [\sigma_y, C_z)$ and $[\sigma^*_k, C^*_k) \not\subseteq [\sigma_y, C_z)$.

Case C.1: *Consider $r_k \geq \sigma_y$.* Since job k is released after the start time of job y, the completion time of job k in OPT is strictly larger than the completion time of job z in WDA, *i.e.* $C^*_k > C_z$. Suppose that $p_k \leq p_z$, then Lemma 7 implies that bound is true. On the contrary if $p_k > p_z$, then Observation 8 implies that job k is released earlier than job z. Moreover, the difference in the release time of job z and job k is at most p_y. Hence $r_z - r_k \leq (1 + \alpha\Delta)p_z$. Using Property 4, we have $r_k + S(\sigma_k)p_k \leq r_z + S(\sigma_k)p_z$ and $S(\sigma_k) \geq \frac{p_k + p_z}{p_z}$. Consequently, we get that the difference in release time of job z and job k is at least $\frac{p_k^2 - p_z^2}{p_z}$. Equating this lower bound with upper bound on $r_k - r_z$, we get $p_k \leq p_z(\sqrt{2 + \alpha\Delta})$. As $C^*_k > C_z$ and $p_k \leq p_z(\sqrt{2 + \alpha\Delta})$, we get $S_z \leq S^*_k(\sqrt{2 + \alpha\Delta}) \leq S^*(1 + \alpha\Delta)$.

Case C.2: *Consider* $r_k < \sigma_y$. If $p_k \leq p_z$ then by Property 10, we have $S_y > 1 + \frac{p_y}{p_k} - \frac{p_k}{p_y} > 1 + \frac{p_y}{p_k} - \frac{p_z}{p_y}$. Since $r'_z \leq C_y$ and $y \notin J_B$, we have $r_y + S_y p_y - p_y < r_z$. Using both inequalities in Equation 1 proves that our bound holds in this case. Conversely suppose that $p_k > p_z$. Since $k \in J_B$, using Property 4 we have $r_k + S(\sigma_k)p_k \leq r_z + S(\sigma_k)p_z$. As intermediate stretch estimate is a non-decreasing function of time, $p_k > p_z$ and $\sigma_y \leq \sigma_k$, we have $r_k + S(\sigma_y)p_k < r_z + S(\sigma_y)p_z$. Hence $r_y + S(\sigma_y)p_y < r_k + S(\sigma_y)p_k < r_z + S(\sigma_y)p_z$. The above facts imply that $r_y - r_z < S(\sigma_y)(p_z - p_y) < p_z - p_y$ since $p_z - p_y < 0$. Substituting this inequality in Equation 1 gives $S_z \leq S^*_y \frac{p_y}{p_z} + 1 - \frac{p_y}{p_z} + \frac{\alpha\Delta}{p_z} \leq (S^*_y - 1)\frac{p_y}{p_z} + 1 + \alpha\Delta \leq S^*_y(1 + \alpha\Delta)$.

□

5.4 Proving the bound when $(1 + \alpha\Delta)p_z < p_y$

Now we build up the tools for the last major Lemma 17 which shows that $S_z \leq S^*(1 + \alpha\Delta)$ when $p_z(1 + \alpha\Delta) \leq p_y$. Observe that for this particular case job z and job y belongs to class J_{small} and J_{large}, respectively. To simplify the notations, from here on we will refer to r'_z as r_z.

Definition 12. *At any time t, we define $J_U(t)$ as set of jobs that are unfinished at time t, i.e. $J_U(t) = \{i \in J : r_i \leq t < C_i\}$*

Then the following lemma relates the stretch estimates $S(t)$ shortly after r_z with the jobs in $J_U(r_z)$.

Lemma 13. *Denote by j the first job started in WDA after r_z. For $t \geq \sigma_j$, $S(t)$ is at least $\frac{\sum_{i \in J_U(r_z)} p_i + \sigma_j - r_z}{p_z}$.*

Before we proceed onto last case analysis in Lemma 17, we define two sets of jobs that are useful for the further analysis. Our aim is to relate the set of jobs in WDA and OPT that are executed after r_z. Informally, we first define a set consisting of jobs that were processed during the interval $[r_z, C^*_y)$, in OPT, such that for each job, their processing time is at most the processing time of job z.

Definition 14. *We define J_S as the set of all jobs in OPT for which the following conditions are met:*

- *job i starts no earlier than r_z, i.e. $\sigma^*_i \geq r_z$.*
- *$p_i \leq p_z$ or the deadline of job i is at most the deadline of job z, according to the optimal stretch S^*, i.e. $r_i + S^* p_i \leq r_z + S^* p_z$.*
- *Job i completes before job y, i.e. $C^*_i < C^*_y$.*

Observe that job z belongs to J_S. Hence J_S is a non-empty set. Now we define the set of big jobs that were processed *consecutively* [1] just before job y (see Figure 2).

[1] Here we assume that the optimal schedule is non-lazy, that is all jobs are scheduled at the earliest time and there is no unnecessary idle time

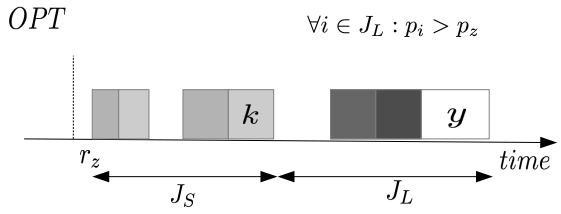

Fig. 2. Representing set of jobs in J_S and J_L

Definition 15. *We define J_L as the set of jobs in schedule OPT that are executed between the completion time of latest job in set J_S and completion time of job y (refer to Figure 2). Formally, $J_L = \{i \in J : \sigma_i^* \in [C_k^*, C_y^*)\}$ where $k \in J_S$ and $\sigma_k^* \geq \sigma_i^*, \forall i \in J_S$. Moreover, λ and $|J_L|$ denote the length of time interval $[C_k^*, C_y^*)$ and the number of jobs in J_L, respectively.*

Note that job y belongs to J_L(hence $\lambda \geq p_y$) and $\forall i \in J_L$, we have $p_i > p_z$ and $r_z + S^* p_z < r_i + S^* p_i$.

Property 16. *If $p_z(1 + \alpha\Delta) < p_y \leq \Delta$, then the total processing time of the jobs in $J_U(r_z)$ is at least $\lambda - p_y + \alpha\Delta$.*

Now we have all the tools necessary to prove the lemma 17.

Lemma 17. *If $p_z(1 + \alpha\Delta) < p_y \leq \Delta$, then $S_z < S^*(1 + \alpha\Delta)$, where S^* is the maximum stretch of some job in OPT.*

Proof. Let k be the latest job in set J_S (see Figure 2). More formally, $k \in J_S$ and $\forall i \in J_S : \sigma_i^* \leq \sigma_k^*$. From Definition 15, we have $C_k^* = C_y^* - \lambda$. We can re-write this equality in terms of the stretch of job y and k as $p_y S_y^* = p_k S_k^* + \lambda + r_k - r_y$. Substituting this expression in Equation 1, we get:

$$S_z \leq S_k^* \frac{p_k}{p_z} + \frac{r_k - r_z}{p_z} + \frac{\alpha\Delta + \lambda}{p_z} \tag{2}$$

Remember that in this subsection we denote by j the first job that starts its execution after time r_z, that is $\sigma_j \leq \sigma_i : \forall i \in J_B$. Now we organize this proof into two parts.

Case A :*Suppose $\sigma_y \geq r_z$*. From Property 4 we have $r_y + S(\sigma_y)p_y < r_z + S(\sigma_y)p_z < r_z + S_z p_z$. Using this inequality in Equation 1, we get $S^* \geq S(\sigma_y) - 1$. Since $\sigma_y \geq r_z$, it follows that job $y \in J_B$ and $S(\sigma_j) \leq S(\sigma_y)$. Also note that job y *limits* job z. Therefore using Property 16 and Lemma 13, we have $S(\sigma_z) \geq S(\sigma_j) \geq 1 + \frac{\lambda - p_y + \alpha\Delta}{p_z}$ Therefore, we have $S^* > \frac{\lambda - p_y + \alpha\Delta}{p_z}$.

Case A.1*: Assume $r_k \leq r_z$*. Plugging $r_k - r_z \leq 0$ and the above lower bound on S^* in Equation 2 we have the desired results.

***Case A.2**: Assume $r_k > r_z$.* From Observation 8, we have $p_k < p_z$. Observe that job k belongs to J_B. From Property 4, we have the $r_k - r_z \leq S(\sigma_k)(p_z - p_k) \leq S_z(p_z - p_k)$. Combining this with above lower bound on S^* and using in Equation 2, we obtain bounded competitive ratio.

Case B : *Suppose that $\sigma_y < r_z$.* Again by Properties 16 and 13, it follows that $S(\sigma_j) \geq 1 + \frac{\lambda - p_y + \alpha\Delta}{p_z}$.

***Case B.1**:* Suppose that there exists some job l such that $l \in J_L$ and $l \in J_B$. [2] Then replace job y with job l in *Case A* and the proof follows.

***Case B.2**:* Now assume that there does not exist any job l such that $l \in J_L$ and $l \in J_B$. Recall that $|J_L| \geq 2$ as stated in case hypothesis B.1. Let v be the smallest job in J_L. Observe that v starts before time r_z in schedule WDA since $v \notin J_B$. Therefore there must be a job $w \in J_B$ such that $\sigma^*_w < r_z$. Now we split the proof into two sections based on processing times of such jobs.

Assume that there exists at least one such job w with $p_v \leq p_w$. Job v is scheduled before job w in the WDA, this implies that $r_v + S(\sigma_v)p_v \leq r_w + S(\sigma_v)p_w$. Since $\sigma_v < r_z \leq \sigma_j$ and $p_v \leq p_w$, we have $r_v + S(\sigma_j)p_v \leq r_w + S(\sigma_j)p_w$. Also z is the last job to be scheduled, which states that $r_w + S(\sigma_j)p_w \leq r_z + S(\sigma_j)p_z$. Hence, we have $r_v + S(\sigma_j)p_v \leq r_w + S(\sigma_j)p_w \leq r_z + S(\sigma_j)p_z$. Since job $v \in J_L$, we also have $r_z + S^*p_z \leq r_v + S^*p_v$. This implies that $S(\sigma_j) \leq S^*$. Using this lower bound in Equation 2, our competitive ratio holds.

On the contrary, we assume that there exists no job w such that $p_v \leq p_w$. Then it implies that there are at least $|J_L|$ are jobs in J_B such that they are started before time r_z in OPT (call such jobs J_M). Moreover $\forall i \in J_M$, $p_i \leq p_v$. Since all jobs belonging to set J_L starts execution before r_z in OPT, there exist a job (denoted by x) in J_M that is delayed at least by λ time units before its start time in WDA. Hence $S^* > S(\sigma_v) \geq \frac{\lambda + p_x}{p_x}$. Now we look at two cases together. First, as we assume that $p_x < 2p_z$. This implies that $S^* \geq \frac{\lambda + 2p_z}{2p_z}$. Second, if $S^* \geq \frac{\lambda + p_x}{p_x} \geq (2|J_L| + 1)$. Using last terms as lower bounds on S^* in Equation 2, our bound holds.

It remains to prove the case where $\frac{\lambda + p_x}{p_x} < (2|J_L| + 1)$ and $p_x \geq 2p_z$. Then we have $p_x > \frac{\lambda}{2|J_L|} \geq \frac{p_v}{2}$. Since job x belongs to set J_B, we have $r_x + S(\sigma_j)p_x \leq r_z + S(\sigma_j)p_z$. Note that at time σ_v, we have $r_v + S(\sigma_v)p_v < r_x + S(\sigma_v)p_x$. Since $p_x < p_v$, we have $r_v < r_x$. Moreover as $v \in J_L$, we also have $r_z + S^*p_z \leq r_v + S^*p_v$. This implies that $r_z + S^*p_z \leq r_v + S^*p_v \leq r_x + 2S^*p_x$. Combining this with $r_x + S(\sigma_j)p_x \leq r_z + S(\sigma_j)p_z$, we get $S^* \geq \frac{S(\sigma_j)(p_x - p_z)}{(2p_x - p_z)}$. Using this as lower bound in Equation 2, we have our desired results. □

6 Concluding remarks

We investigated the online non-preemptive problem scheduling of a set of jobs on a single machine that are released over time so as to optimize the maximum stretch of the

[2] Note that job l starts processing after time r_z in both schedule OPT and WDA.

jobs. We showed that no algorithm can achieve a competitive ratio better than $\frac{\sqrt{5}-1}{2}\Delta$ for the maximum stretch objective. We proposed a new algorithm which delays the execution of large jobs and achieves a competitive ratio $1+\frac{\sqrt{5}-1}{2}\Delta$. This paper essentially closes the problem of optimizing the maximum stretch on a single machine. Indeed, when Δ goes to infinity, these upper and lower bounds are both equal to $\frac{\sqrt{5}-1}{2}\Delta$

The following questions will receive our attention next. Is WDA competitive for the average stretch? Can the waiting strategy of WDA be extended to the more general weighted flow time objectives? Can we design an algorithm better than Δ competitive for max-stretch when multiple machines are available?

Acknowledgments. This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir. Erik Saule is a 2015 Data Fellow of the National Consortium for Data Science (NCDS) and acknowledges the NCDS for funding parts of the presented research.

References

1. P. Brucker. *Scheduling Algorithms*. Springer-Verlag New York, Inc., 3rd edition, 2001.
2. M. Bender, S. Chakrabarti, and S. Muthukrishnan. Flow and stretch metrics for scheduling continuous job streams. In *Proc. of SODA*, pages 270–279, 1998.
3. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. *Handbooks in operations research and management science*, 4:445–522, 1993.
4. E. Saule, D. Bozdag, and Ü. V. Çatalyürek. Optimizing the stretch of independent tasks on a cluster: From sequential tasks to moldable tasks. *J. of Para. and Dist. Comp.*, 2012.
5. A. Legrand, A. Su, and F. Vivien. Minimizing the stretch when scheduling flows of divisible requests. *Journal of Scheduling*, 11(5):381–404, 2008.
6. D.S. Hochbaum. *Approximation Algorithms for NP-hard problems*. PWS, 1997.
7. N. Bansal and K. Pruhs. Server scheduling in the Lp norm: A rising tide lifts all boat. In *Proc. of ACM STOC*, pages 242–250, 2003.
8. D. Golovin, A. Gupta, A. Kumar, and K. Tangwongsan. All-norms and all-Lp-norms approximation algorithms. *Proc. of FSTTCS*, pages 199–210, 2008.
9. S. Im and B. Moseley. An online scalable algorithm for minimizing l_k-norms of weighted flow time on unrelated machines. *Proc. of ACM-SIAM SODA*, pages 98–108, 2011.
10. S. Anand, N. Garg, and A. Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proc. of ACM-SIAM SODA*, pages 1228–1241, 2012.
11. X. Lu, R. A. Sitters, and L. Stougie. A class of on-line scheduling algorithms to minimize total completion time. *Operation Research Letters*, 31:232–236, 2003.
12. N. Megow and A. S. Schulz. On-line scheduling to minimize average completion time revisited. *Operation Research Letters*, 32:485–490, 2003.