Spring 4-2012

Completion Time Scheduling and the WSRPT Algorithm

Christine Chung
Connecticut College, cchung@conncoll.edu

Bo Xiong
Connecticut College, bxiong@conncoll.edu

Completion Time Scheduling and the WSRPT Algorithm

Comments
Presented at ISCO 2012 (International Symposium on Combinatorial Optimization.

Completion Time Scheduling and the WSRPT Algorithm

Bo Xiong, Christine Chung

Department of Computer Science, Connecticut College, New London, CT
{bxiong,cchung}@conncoll.edu

Abstract. We consider the online scheduling problem of minimizing the total weighted and unweighted completion time on identical parallel machines with preemptible jobs. We show a new general lower bound of $21/19 \approx 1.105$ on the competitive ratio of any deterministic online algorithm for the unweighted problem and $\frac{16-\sqrt{14}}{11} \approx 1.114$ for the weighted problem. We then analyze the performance of the natural online algorithm WSRPT (Weighted Shortest Remaining Processing Time). We show that WSRPT is 2-competitive. We also prove that the lower bound on the competitive ratio of WSRPT for this problem is 1.215.

1 Introduction

We consider the well-studied online problem of preemptively scheduling jobs on identical parallel machines to minimize total completion time, both in the case that the jobs have weights and in the case that the jobs are unweighted. We have m machines and a set of n jobs that arrive over time. Each job j is characterized by a positive integer processing time, p_j, and an nonnegative integer release time, r_j. In the weighted case, each job also has a positive integer weight w_j. (The unweighted case is equivalent to all jobs having equal weights.) A job's processing time, release time, and weight are not known until a job arrives. We cannot schedule a job on a machine before its release time and each job is preemptible. We use c_j to denote the completion time of job j, and our goal is to minimize $\sum_j c_j$ in the unweighted case and $\sum_j w_j c_j$ in the weighted case. The unweighted problem is denoted $P \,|\, r_j, \text{pmtn} \,|\, \sum_j c_j$ and the weighted problem is $P \,|\, r_j, \text{pmtn} \,|\, \sum_j w_j c_j$ using the standard scheduling notation of [1]. The weighted problem is NP-Hard, even for the single-machine case [2], and the unweighted problem is NP-Hard for two or more machines [3].

The best known general lowerbound on the competitive ratio for any deterministic algorithm (for both the weighted and unweighted case) due to Vestjens [4] was 1.047 and has held since 1997. With respect to upperbounds, for weighted completion time scheduling, Megow and Schulz [5] showed that the algorithm WSPT (Weighted Shortest Processing Time), which schedules jobs in order of non-decreasing weight-to-size ratio, is 2-competitive. Sitters [6] then recently gave an algorithm that is 1.791-competitive for weighted completion time scheduling.

In this work, we give an updated general lower bound of $21/19 \approx 1.105$ for the competitive ratio of any deterministic online algorithm for the unweighted case and $\frac{16-\sqrt{14}}{11} \approx 1.114$ for the weighted case.

We then analyze the performance of the algorithm WSRPT (Weighted Shortest Remaining Processing Time). WSRPT is an extension of the famous SRPT (Shortest Remaining Processing Time) algorithm, which is used for unweighted job scheduling. At any point in the schedule, SRPT always runs the job(s) with the shortest remaining processing time. SRPT is optimal for the unweighted problem on one machine. The competitiveness of SRPT for parallel machines was first analyzed in 1995 by [7], has been evaluated experimentally (e.g., [8–10]), and has continued to be widely used and studied. It was a long-held belief that SRPT was close to optimal, and most recently, [6] finally showed that SRPT is 5/4-competitive. (Prior to that, the original proof of SRPT's 2-competitiveness [7] was the best known ratio for fifteen years until [11] showed it was 1.86-competitive.) The current lowerbound on the competitiveness of SRPT is $21/19 \approx 1.105$ [11].

WSRPT is a natural extension of SRPT to the weighted case of the problem, as it schedules jobs based on smallest remaining weight-to-size ratio. (WSPT, by contrast, schedules jobs based only on smallest *initial* weight-to-size ratio, without taking into account how much of a job has been processed at any point in the schedule.) Megow [12] proves WSRPT is 2-competitive for the single machine case. To the best of our knowledge, ours is the first analysis of WSRPT for the parallel machine case. We show that WSRPT is 2-competitive for the problem $P \,|\, r_j, \text{pmtn} \,|\, \sum_j w_j c_j$, and we suspect it actually yields a much better performance guarantee. We also exhibit an instance of the problem, by modifying a lowerbound instance from [12], where the WSRPT schedule has a total weighted completion time 1.215 times that of the optimal offline schedule.

While our result for WSRPT does not beat the competitiveness of the best known algorithm for the problem $P \,|\, r_j, \text{pmtn} \,|\, \sum_j w_j c_j$ (which is due to [6]), our contribution is with respect to the WSRPT algorithm and techniques for analyzing the algorithm as applied to parallel machines. While the algorithm itself is simple to state, it has proven somewhat difficult to analyze, due to the interplay of the static job weights with the changing nature of the remaining processing times. Further, analyzing the algorithm for multiple parallel machines rather than one machine (as in [12]) demands that we handle additional possible situations that can cause the priorities of two jobs to change order as time unfolds. Given that SRPT is so popular and effective an algorithm, we speculate that many implementations of SRPT are in application domains where weights on the jobs are becoming a relevant factor. Easily tweaking SRPT implementations into WSRPT implementations will then be quite alluring for programmers. Our conjecture is that not only is WSRPT a simpler and more natural algorithm, but it is also a better algorithm for the problem $P \,|\, r_j, \text{pmtn} \,|\, \sum_j w_j c_j$ than the current proven leading algorithm, with respect to competitiveness.

To summarize, our contributions are:

- We show that WSRPT is 2-competitive for the problem $P \,|\, r_j, \text{pmtn} \,|\, \sum_j w_j c_j$.

- We show that the competitive ratio of WSRPT for the problem $1 \mid r_j, \text{pmtn} \mid \sum_j w_j c_j$ (and hence $P \mid r_j, \text{pmtn} \mid \sum_j w_j c_j$) is no better than 1.215.
- We prove a general lower bound of 1.114 on the competitive ratio of any deterministic algorithm for the problem $P \mid r_j, \text{pmtn} \mid \sum_j w_j c_j$.
- We prove a general lower bound of 21/19 on the competitive ratio of any deterministic algorithm for the problem $P \mid r_j, \text{pmtn} \mid \sum_j c_j$.

2 Preliminaries

Each instance of our scheduling problem consists of m machines and a set of jobs $J = \{1, ..., n\}$ that arrive over time, and we start at time $t = 0$. Each of the machines can only process one of the n jobs at a time, and each job can be processed by at most one machine at a time. Each job j is characterized by a positive integer processing time p_j and an nonnegative integer release time r_j. For unweighted completion time scheduling, all jobs have weight of one. For weighted completion time scheduling, each job has a positive integer weight w_j. We cannot schedule a job j before its *release time* r_j, which is not known in advance. Each job is *preemptible*, so a job may be suspended and resumed later on any machine at any time at no extra cost. For convenience, and without loss of generality, we assume each job may only be suspended or resumed on any machine at integer times. In the unweighted problem, if σ is a scheduling of the jobs in J, we define $\text{cost}(\sigma) = \sum_j c_j(\sigma)$ (i.e., *total completion time*), where $c_j(\sigma)$ denotes the completion time of job j in the schedule σ. When σ refers to a schedule for the weighted problem, then we define $\text{cost}(\sigma) = \sum_j w_j c_j(\sigma)$, referred to as *total weighted completion time*. When the schedule being referenced is clear from context, we write $c_j(\sigma)$ simply as c_j. For both problems, our goal is to minimize $\text{cost}(\sigma)$ over all possible schedules.

A problem is an *online problem* if the inputs arrive over time and there is no prior knowledge of inputs that will arrive in the future. The *competitive ratio* of an online algorithm is the maximum (over all input instances) of the ratio of the total weighted completion time of the schedule produced by the algorithm and the total weighted completion time of an optimal offline schedule (one that knows all future inputs in advance). More formally, if $OPT(I)$ is the optimal offline schedule on an instance I and $A(I)$ is the schedule produced by online algorithm A on instance I, then the competitive ratio of an algorithm A can be expressed as

$$\max_{I \in \mathcal{I}} \frac{\text{cost}(A(I))}{\text{cost}(OPT(I))},$$

where $\mathcal{I}$ is the set of all possible input instances.

An algorithm is called ρ-*competitive* if it has a competitive ratio of at most ρ. The competitive ratio is the standard measure by which algorithms for online problems are evaluated.

3 General Lower Bound

In this section, we show a lower bound of $21/19 \approx 1.105$ on the competitive ratio of any algorithm for unweighted completion time scheduling and a lower bound of $\frac{16-\sqrt{14}}{11} \approx 1.114$ for weighted completion time scheduling. We prove these lower bounds by giving a simple scheme for constructing an instance where the nature of the jobs that arrive depends on the choices the algorithm has made so far. The scheme is based on the lower bound instance for the algorithm SRPT, from [11].

Theorem 1. *There is no deterministic algorithm that has a competitive ratio better than* $21/19$ *for the problem* $P \,|\, r_j, pmtn| \sum_j c_j$.

Proof. Consider the following instance with only two machines. At time 0, a set of three jobs arrives, two of which have a processing time of 1 and one of which has a processing time of 2.

In order to minimize the total completion time, we assume without loss of generality that the online algorithm always processes an available job if a machine is available. Since preemption is allowed only at integer times, the online algorithm will either have processed none or one unit of the job of length 2 at time 1.

Suppose the online algorithm does not process any of the job of length 2 by time $t = 1$. Then we release a set of 4 jobs of length 1 at time $t = 2$. Consequently, at this point the best schedule σ for this instance would be to schedule the job of length 2 at $t = 1$ and the 4 jobs of length 1 one after another when the machines become available. Thus total completion time is $\text{cost}(\sigma) = 1+1+3+3+4+4+5 = 21$. However, the optimal schedule σ^* is to begin processing the job of length 2 at $t = 0$, which achieves $\text{cost}(\sigma^*) = 1 + 2 + 2 + 3 + 3 + 4 + 4 = 19$. (Please refer to Case 1 of Figure 1.)

Now suppose the algorithm has processed one unit of the job of length 2 at $t = 1$. In this case, we release only one job of length 1 at $t = 1$. Consequently, the best schedule σ at this point for this instance would be to schedule any remaining jobs one after another when machines become available. Hence, $\text{cost}(\sigma) = 1 + 2 + 2 + 3 = 8$. However, if we schedule both jobs of length 1 at $t = 0$, we can achieve an optimal schedule σ^* with $\text{cost}(\sigma^*) = 1 + 1 + 2 + 3 = 7$.

The competitive ratio in the first case was $21/19$ and the second was $8/7$. Combining both cases, we conclude that there is no deterministic algorithm that has a competitive ratio better than $21/19$ for unweighted completion time scheduling.

Theorem 2. *There is no deterministic algorithm that has a competitive ratio better than* $\frac{16-\sqrt{14}}{11} \approx 1.114$ *for the problem* $P \,|\, r_j, pmtn| \sum_j w_j c_j$.

Proof. We use a similar idea as in the proof of previous theorem to prove this lower bound. However, we add weights to the jobs. We first release three jobs,

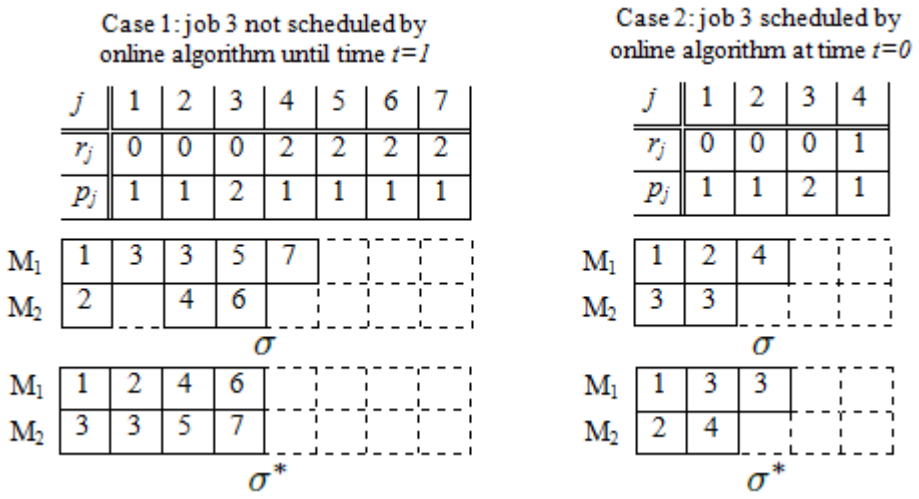

Fig. 1. The input instance for each case and the corresponding online (σ) and optimal (σ^*) schedules. The job number being processed is indicated within each integer time segment of each schedule.

two of which have processing time 1 and weight 1, the other one of which has processing time 2 and weight $\frac{2+\sqrt{14}}{5}$. In the first case, suppose the online algorithm does not process any of the job of length 2 at $t = 1$. We then release four jobs of weight $\frac{2+\sqrt{14}}{5}$ and processing time 1 at $t = 2$. In the second case, suppose the online algorithm has processed one unit of the job of length 2 at $t = 1$. We then release one job of weight 1 and processing time 1 at $t = 1$. The online and optimal schedules for these two cases reflects that of the two cases in Figure 1.

In the first case, $\text{cost}(\sigma) \geq 9.6 + \frac{19}{5}\sqrt{14}$ for any online schedule σ, and $\text{cost}(\sigma^*) = 9.4 + \frac{16}{5}\sqrt{14}$ for the optimal schedule σ^*. In the second case, $\text{cost}(\sigma) \geq 6.8 + \frac{2}{5}\sqrt{14}$ and $\text{cost}(\sigma^*) = 5.2 + \frac{3}{5}\sqrt{14}$. The competitive ratio in either case is at least $\frac{16-\sqrt{14}}{11}$. Thus, there is no deterministic algorithm that has a competitive ratio better than $\frac{16-\sqrt{14}}{11} \approx 1.114$ for weighted completion time scheduling.

4 WSRPT

In this section, we analyze the performance of the algorithm WSRPT (Weighted Shortest Remaining Processing Time). A related well-known algorithm, SRPT (Shortest Remaining Processing Time), applies to the variant of our problem where the jobs are unweighted. At every point in time, SRPT simply schedules the m jobs with shortest remaining processing time. SRPT is known to be 5/4-competitive for the unweighted variant of the problem [6]. WSRPT can be seen as the weighted version of SRPT.

The WSRPT algorithm proceeds as follows. Define the *remaining priority* of a job at a given time to be the weight of the job over the remaining processing time of the job. At any time, process the m available jobs with highest remaining priority, or fewer if less than m jobs are available. Ties are broken by choosing the job with the smaller job index. Note that the remaining priority of a job will change as the job is processed. When new jobs arrive, we recalculate the remaining priority of all jobs and reschedule the jobs based on the new values.

4.1 Lower Bound of WSRPT

Megow [12] showed that the algorithm WSRPT does not have a competitive ratio less than 1.21057 for the weighted completion scheduling problem on single machine. We slightly modify her instance to improve the lower bound from 1.21057 to 1.21568 and note that it also applies to the problem $P \,|\, r_j, \text{pmtn} \,|\, \sum_j w_j c_j$. For completeness, we reproduce the entire instance with modification here.

Theorem 3. *If the algorithm WSRPT is $\rho-$competitive for the scheduling problem $1|r_j, pmtn|\sum_j w_j c_j$, then ρ is at least* 1.21568.

Proof. Consider the following instance with one machine and $k + x + 1$ jobs: x high priority jobs with weight $1/x$ and processing time $1/x$, one low priority job ℓ with weight 1 and processing time p_ℓ and k small jobs of length $\epsilon = (p_\ell - 1)/k$. The job ℓ and the first small job are released at time 0.

The remaining small jobs are released at $r_j = (j-1)\epsilon$ for $j = 2, 3, ..., k$ and all the high priority jobs are released at $p_\ell - 1$. (Note that it is feasible for all small jobs to be completed by the time this high priority job is released.) The weight of the small jobs are $w_j = \epsilon/(p_\ell - (j-1)\epsilon)$ for $j = 1, 2, .., k$.

Since the priority of job ℓ and the first small job are the same, we assume without loss of generality that WSRPT starts processing job ℓ at time 0. Note that as each small job is released, it is tied in priority with the remaining priority of job ℓ. Hence, we do not preempt job ℓ until at $t = p_\ell - 1$ when all the jobs with high priority are released. Then we start processing all the jobs with high priority one after another. After all the jobs with high priority are finished, we then finish processing ℓ and all the small jobs. The total weighted completion time of WSRPT is thus

$$\sum_{i=1}^{x} \frac{(p_\ell - 1 + \frac{i}{x})}{x} + p_\ell + 1 + \sum_{i=1}^{k} (p_\ell + 1 + i\epsilon) \frac{\epsilon}{p_\ell - (k-i)\epsilon}$$

The optimal schedule in the instance should process all small jobs first, then all the jobs with high priority and job ℓ at last. The weighted completion time for optimal schedule is

$$\sum_{i=1}^{x} \frac{(p_\ell - 1 + \frac{i}{x})}{x} + 2p_\ell + \sum_{i=1}^{k} i\epsilon \frac{\epsilon}{p_\ell - (i-1)\epsilon}$$

As k and x tend to infinity, then the competitive ratio of WSRPT is no less than

$$\frac{p_\ell(3 - \ln \frac{1}{p_\ell - 1} + \ln \frac{p_\ell}{p_\ell - 1}) - 0.5}{0.5 + 2p_\ell + p_\ell \ln p_\ell} \geq 1.2156861$$

for $p_\ell \approx 5.17$.

4.2 Upper Bound of WSRPT

We begin this section by distinguishing the term *initial priority* from *remaining priority*. The *initial priority* of a job j is defined as the weight of j, w_j, divided by the total processing time of j, p_j. Initial priority does not change over time. We contrast this with *remaining priority* of a job j at time t, which we have defined as the weight divided by the *remaining* processing time of job j at time t, and we denote it $p_j(t)$. As the remaining processing time of a job decreases, its remaining priority increases. For convenience and without loss of generality, we assume all the jobs are indexed in non-decreasing *initial* priority: $w_1/p_1 \geq w_2/p_2 \geq ... \geq w_n/p_n$.

Megow and Schulz [5] showed that the algorithm WSPT (Weighted Shortest Remaining Time) is 2-competitive. They also give a matching lower bound on the competitive ratio of WSPT, proving that WSPT is no better than 2-competitive. Since the algorithm WSPT only considers the initial priority of the job, it has the helpful property that the job's scheduling priority never changes. However, WSRPT schedules jobs based on remaining priority, which changes as the job is processed. In this section, we show the algorithm WSRPT is 2-competitive. We note, however, that its counterpart SRPT is 5/4-competitive for the unweighted version of the problem, so we conjecture WSRPT in fact has a strictly lower competitive ratio than the algorithm WSPT.

Our overarching strategy will be to bound the total weighted idle time of all jobs in the schedule. We will then be able to bound an expression for total completion time that is broken down into two components: idle time and processing time. We now present three lemmas that will help us toward this goal.

The first two lemmas establish the fact that in a WSRPT schedule, if job j has higher remaining priority than job k at some point, and job k has higher remaining priority than job j later, then job j will never have higher remaining priority than job k again.

Lemma 1. *Consider two jobs a and b in a WSRPT schedule. After a and b are released, if we have $w_a/p_a(t_1) > w_b/p_b(t_1)$ and $w_a/p_a(t_2) < w_b/p_b(t_2)$ for some t_1 and t_2 such that $t_1 < t_2$, then $p_a(t_1) > p_b(t_1)$ and $w_a > w_b$.*

Proof. We prove the above lemma by contradiction. Assume $p_a(t_1) \leq p_b(t_1)$. Let t' be the earliest time such that $w_a/p_a(t') < w_b/p_b(t')$. According to the definition of WSRPT, and the fact that job a had greater remaining priority than job b in the interval $[t_1, t')$, we have

$$p_a(t_1) - p_a(t') \geq p_b(t_1) - p_b(t').$$

We can then conclude

$$\frac{p_a(t')}{p_b(t')} = \frac{p_a(t_1) - (p_a(t_1) - p_a(t'))}{p_b(t_1) - (p_b(t_1) - p_b(t'))} < \frac{p_a(t_1)}{p_b(t_1)}$$

By assumption, we also know that

$$\frac{w_a}{w_b} > \frac{p_a(t_1)}{p_b(t_1)}$$

and

$$\frac{w_a}{w_b} < \frac{p_a(t')}{p_b(t')}$$

It follows that

$$\frac{p_a(t')}{p_b(t')} > \frac{p_a(t_1)}{p_b(t_1)}$$

At this point we reach a contradiction. When we combine the fact that $p_a(t_1) > p_b(t_1)$ and $w_a/p_a(t_1) > w_b/p_b(t_1)$, we can conclude $w_a > w_b$.

Lemma 2. *Consider two jobs c and d in a WSRPT schedule. After c and d are released, if we have $w_c/p_c(t_1) > w_d/p_d(t_1)$ and $w_c/p_c(t_2) < w_d/p_d(t_2)$ for some t_1 and t_2 such that $t_1 < t_2$, then $w_c/p_c(t_3) < w_d/p_d(t_3)$ is true for any $t_3 > t_2$.*

Proof. Let t' be the earliest time such that $w_c/p_c(t') < w_d/p_d(t')$. From Lemma 1 we know $w_c > w_d$. Combined with the fact that $w_c/p_c(t') < w_d/p_d(t')$, we can conclude $p_c(t') > p_d(t')$. Assume for contradiction that at some $t_3 > t'$ we have $w_c/p_c(t_3) > w_d/p_d(t_3)$. Applying Lemma 1 again, we have $p_d(t') > p_c(t')$, a contradiction.

For the next lemma, we first define some notation. We define the *idle time* of a job as the total time that a job is not being processed between its release date and completion time. Let $d_j(\sigma)$ denote the idle time of job j in schedule σ. Then $d_j(\sigma) = c_j(\sigma) - r_j - p_j$. (We omit the σ when it is clear from context.) We can then rewrite the total weighted completion time for a schedule σ as $\text{cost}(\sigma) = \sum_j w_j(p_j + r_j) + \sum_j w_j d_j$. We use $W(\sigma)$ to denote the total weighted idle time for a schedule σ, that is, $W(\sigma) = \sum_j w_j d_j$.

Akin to [5], for any job j, we partition the interval between c_j and r_j into two non-overlapping set of subintervals $I(j)$ and $I'(j)$, where $I(j)$ denotes the set of subintervals in which job j is being processed and $I'(j)$ denotes the set of remaining subintervals. Note that in $I'(j)$, all machines are busy, otherwise job j would be processed. Also notice that the sum of lengths of the subintervals in $I'(j)$ is equal to d_j, the idle time of job j. Let $\delta_j(k)$ denote the amount of job k that is being processed in $I'(j)$. The idle time d_j can then be expressed as $\sum_{k \in D(j)} \delta_j(k)/m$ where $D(j)$ is the set of jobs that are being processed in $I'(j)$.

We partition set $D(j)$ into two sets $A(j)$ and $B(j)$, where $A(j) = \{k \in D(j) : k < j\}$ and $B(j) = \{k \in D(j) : k > j\}$.

We are now ready to prove the following lemma.

Lemma 3. *For any job j in a WSRPT schedule σ, if a job b is in set $B(j)$, we have*

$$w_b\delta_b(j) + w_j\delta_j(b) \leq w_b p_j$$

Proof. Let s_j and s_b refer to the times that jobs j and b first start processing in σ. We consider two cases.

Case 1: $s_j \leq s_b$. Since $\delta_j(b) > 0$ by definition of $B(j)$, then at some time in $I'(j)$, job b has higher remaining priority than job j. (For an illustration of this situation, see Figure 2.) Let t' be the earliest time that job b first has remaining priority higher than j, so $w_b/p_b(t') > w_j/p_j(t')$. Since, by Lemma 2, job j will never have higher remaining priority than job b after t', we can conclude $\delta_j(b) \leq p_b(t')$ and $\delta_b(j) \leq p_j - p_j(t')$. Thus, we have

$$w_j\delta_j(b) + w_b\delta_b(j) \leq w_j p_b(t') + w_b(p_j - p_j(t'))$$

We add nonnegative term $w_b p_j(t') - w_j p_b(t')$to the right hand side and obtain

$$w_j\delta_j(b) + w_b\delta_b(j) \leq w_b p_j$$

j	1	2	3
r_j	1	0	0
p_j	1	5	2
w_j	2	6	2

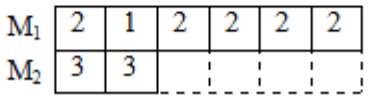

Fig. 2. An example instance where job 2 originally has higher priority than job 3, but at time $t' = 1$, the remaining priority of job 3 overtakes that of job 2. Hence $\delta_2(3) > 0$ even though job 3 has lower initial priority than job 2.

Case 2: $s_j > s_b$. In this case, since job j has higher initial priority than b, job b must be released before job j. When job j is released at time r_j, we have two subcases: either job j has higher remaining priority or job b does. (Note that at this point, remaining priority of j is the same as the initial priority).

If $w_j/p_j > w_b/p_b(r_j)$, then we are back in Case 1, by simply treating time r_j as time 0. (Note that the before time r_j, both $\delta_j(b)$ and $\delta_b(j)$ remain at 0.)

If $w_b/p_b(r_j) \geq w_j/p_j$, we can view the remainder of job b as a job with higher initial priority than job j. Hence, by switching the roles of j and b, and again by treating time r_j as time 0, we can again apply Case 1 above, which yields

$$w_b\delta_b(j) + w_j\delta_j(b) \leq w_j p_b(r_j).$$

By assumption, we have $w_b p_j \geq w_j p_b(r_j)$, concluding our proof.

We are now ready to bound the total weighted idle time in the schedule.

Lemma 4. *Let σ be a WSRPT schedule. Then*

$$W(\sigma) \leq \sum_{j \in J} w_j \sum_{k<j} \frac{p_k}{m}.$$

Proof. We begin by observing that

$$\begin{aligned} W(\sigma) &= \sum_{j \in J} w_j d_j = \sum_{j \in J} w_j \sum_{k \in D(j)} \frac{\delta_j(k)}{m} \\ &= \sum_{j \in J} w_j \sum_{k \in A(j)} \frac{\delta_j(k)}{m} + \sum_{j \in J} w_j \sum_{k \in B(j)} \frac{\delta_j(k)}{m} \qquad (1) \\ &\leq \sum_{a,b \in J : a<b} \left(\frac{w_b \delta_b(a)}{m} + \frac{w_a \delta_a(b)}{m} \right). \qquad (2) \end{aligned}$$

For inequality (2), note that if $a < b$, then $w_a \frac{\delta_a(b)}{m}$ can only appear in the second summation term of (1), and $w_b \frac{\delta_b(a)}{m}$ can only appear in the first summation term of (1). Finally, applying Lemma 3, the claim follows.

Theorem 4. *If WSRPT is c-competitive, then $c \leq 2$.*

Proof. In a WSRPT schedule σ, the total weighted completion time can be written:

$$\begin{aligned} \text{cost}(\sigma) = \sum_{j \in J} w_j (p_j + r_j + d_j) &= \sum_{j \in J} w_j d_j + \sum_{j \in J} w_j (p_j + r_j) \\ &\leq \sum_{j \in J} w_j \sum_{k<j} \frac{p_k}{m} + \sum_{j \in J} w_j (p_j + r_j), \end{aligned}$$

where the inequality holds by Lemma 4. At this point, we can echo the WSPT proof of [5] to complete our proof as follows.

On a single machine, when all jobs are released at time 0, the optimal strategy for minimizing weighted completion time is to schedule the job with the lowest initial priority first [13]. If we assume the single machine is m times faster than each of our m parallel machines, the optimal cost for this "easier" problem can be expressed $\sum_{j \in J} w_j \sum_{k<j} \frac{p_k}{m}$. Since the aforementioned single-machine problem is a relaxation of our problem, and any schedule for our m machines can be mapped to a single m-fast machine so that weighted completion time only improves, we have

$$\sum_{j \in J} w_j \sum_{k<j} \frac{p_k}{m} \leq \text{cost}(\sigma^*).$$

Clearly, we also have:

$$\sum_{j \in J} w_j (p_j + r_j) \leq \text{cost}(\sigma^*).$$

Thus, we have $\text{cost}(\sigma) \leq 2 \cdot \text{cost}(\sigma^*)$.

5 Conclusion

In this work we have provided a first analysis of the algorithm WSRPT for the problem $P \,|\, r_j, \text{pmtn} \,|\, \sum_j w_j c_j$. We show that WSRPT is 2-competitive, and demonstrate a lower bound of 1.215 on its competitive ratio. We conjecture the true competitive ratio of WSRPT for this problem is closer to the lower bound.

We also provide improved general lower bounds of $21/19 \approx 1.105$ and 1.114 on the competitive ratio of any deterministic algorithm for the problems $P \,|\, r_j, \text{pmtn} \,|\, \sum_j c_j$ and $P \,|\, r_j, \text{pmtn} \,|\, \sum_j w_j c_j$, respectively. We believe that $21/19$ is the correct answer for the unweighted problem, as we agree with the conjecture of [11] that the algorithm SRPT is $21/19$-competitive.

References

1. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling theory: a survey. ADM **5** (1979) 287 – 326
2. Labetoulle, J., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Preemptive scheduling of uniform machines subject to release dates (1982)
3. Du, J., Leung, J., Young, G.: Minimizing mean flow time with release time constraint. Theoretical Computer Science **75** (1990)
4. A.P.A.Vestjens: On-line machine scheduling, ph.d. thesis, Eindhoven University of Technology (1997)
5. Megow, N., Schulz, A.S.: On-line scheduling to minimize average completion time revisited. In: Operations Research Letters 32: 485-490. (2004)
6. Sitters, R.: Efficient algorithms for average completion time scheduling. In: Proceedings 14th Conference on Integer Programming and Combinatorial Optimization (IPCO). (2010)
7. Phillips, C.A., Stein, C., Wein, J.: Minimizing average completion time in the presence of release dates. Math. Program. **82** (1998) 199–223
8. Bansal, N., Harchol-Balter, M.: Analysis of srpt scheduling: investigating unfairness. In: Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. SIGMETRICS '01 (2001) 279–290
9. Harchol-Balter, M., Schroeder, B., Bansal, N., Agrawal, M.: Size-based scheduling to improve web performance. ACM Trans. Comput. Syst. **21** (2003) 207–233
10. Gong, M., Williamson, C.: Quantifying the properties of srpt scheduling. Modeling, Analysis, and Simulation of Computer Systems, International Symposium on **0** (2003) 126
11. Chung, C., Nonner, T., Souza, A.: Srpt is 1.86-competitive for completion time scheduling. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '10 (2010) 1373–1388

12. Megow, N.: Coping with incomplete information in scheduling - stochastic and online models, Technische Universitt Berlin (2006)
13. Smith, W.E.: Various optimizers for single-stage production. Naval Research Logistics Quarterly **3** (1956) 59–66