A 2.28-COMPETITIVE ALGORITHM FOR ONLINE SCHEDULING ON IDENTICAL MACHINES

JIPING TAO, RONGHUAN HUANG AND TUNDONG LIU

Department of Automation, Xiamen University
422 South Siming Road, Xiamen, 361005, China

(Communicated by Yakov Shafransky)

Abstract. Online scheduling on identical machines is investigated in the setting where jobs arrive over time. The goal is to minimize the total completion time. A waiting strategy based online algorithm is designed and is proved to be 2.28-competitive. The result improves the current best online algorithm from the worse-case prospective.

1. Introduction. Identical machines scheduling is considered in the online setting. There is a sequence of jobs arriving over time and must be scheduled on m identical machines. Each job $J_j$ becomes available for processing at its release time $r_j$ and is associated with a processing time $p_j$ and a positive weight $w_j$. All the information about one job is not revealed until it is released. Each job must be processed on one of the m identical machines without interruption, and each machine can process at most one job at a time. The goal is to minimize the total weighted completion time, $\sum w_j C_j$, where $C_j$ is the completion time of job $J_j$. The problem can be denoted by $Pm|r_j, online|\sum w_j C_j$ in terms of the three-field notation for scheduling problems in [13].

An online algorithm is often assessed by its competitive performance, which is also known as the worst-case performance. An algorithm is called $\rho$-competitive if, for any instance, the objective function value of the schedule generated from this algorithm is no worse than $\rho$ times the objective value of the optimal offline schedule [4].

The first online algorithm with constant competitive performance guarantee for the problem above is given by Hall et al. [7]. They design a $(4 + \varepsilon)$-competitive online algorithm based on the Greedy-Interval technique and a dual PTAS of the Maximum Scheduled Weight Problem, where $\varepsilon > 0$ is an arbitrarily small positive value. The result is improved to a value of 3.28 by using the technique of shifting releasing times [11]. The idea of $\alpha$-point is widely applied to construct provably good schedules in the online setting as well as in the offline scheduling [6, 7, 12]. Correa and Wagner [3] combine the $\alpha$-point method with linear programming relaxation techniques and propose a 2.618-competitive algorithm for $Pm|r_j, online|\sum w_j C_j$. 

2010 Mathematics Subject Classification. Primary: 90B35; Secondary: 68Q25.

Key words and phrases. Identical machines scheduling, online algorithms, competitive analysis, instance reduction.

All correspondence should be addressed to ltd@xmu.edu.cn(Tundong Liu). This work was supported by the National Natural Science Foundation of China (No.11201391) and the Fundamental Research Funds for the Central Universities.
In a recent study [14], Sitters revisits the technique of shifting releasing times and designs an online algorithm named by ONLINE($\epsilon$). He proves that the competitive ratio of ONLINE($\epsilon$) is not greater than $(1 + 1/\sqrt{m})^2 (3e - 2)/(2e - 2)$. The value is much greater for less machine number (say $m \leq 31$), although it tends to 1.79 when $m$ tends to infinity. Tao et al. generalize the directly waiting strategy based idea, which is commonly applied in [1, 8, 9], and show a preliminary result about a $(2.5 - 1/2m)$-competitive online algorithm named by AD-SWPT in [15].

As mentioned in [8] for the single machine case, any online algorithm that schedules a job as soon as the machine is available has an unbounded worst-case performance. The conclusion also holds up for the case of identical machines. In other words, a waiting strategy is necessary in order to guarantee a better competitive performance. Two kinds of techniques are commonly used to design waiting strategies. The first one is to shift releasing times to later times. The revised releasing times are derived either by a heuristic method like in [10, 11, 12], or from a preemptive schedule by solving a related relaxation problem as in [3, 5, 6]. Another technique used in [1, 8, 9] is to apply a directly waiting strategy, which makes decisions to insert appropriate waiting time or to immediately schedule a job by directly comparing processing time with the current time. In this work, we revisit the second technique and further investigate the ideas behind the AD-SWPT rule and its competitive analysis in [15]. We note that an improvement can be achieved by introducing a parameter related to the machine number $m$ in the waiting strategy. Intuitively, we need to insert less waiting time in the online schedule in order to achieve a good competitive performance when more machines are available. The competitive analysis is based on the idea of instance reduction, which is commonly used for several semi-online and online scheduling problems in [15, 16, 17]. The main result in this paper is that an improved AD-SWPT rule with the parameter of $m$ is presented, and it is proved to be $\alpha_m$-competitive, where $\alpha_m = 1 + m^{-1} + \sqrt{17m^2 - 2m + 1}/4m$. $\alpha_m$ reduces to 2 when $m = 1$ and monotonously increases to about 2.28. The value together with the competitive performance of AD-SWPT in [15] is illustrated with respect to the number $m$ in Figure 1.

The remaining sections are organized as follows. In Section 2, the improved AD-SWPT rule is presented. Its competitive performance is analyzed in Section 3 based on the idea of instance reduction. Conclusions are given in Section 4.
2. The improved AD-SWPT rule. The key point to design a good online algorithm for the scheduling problem is to determine whether to insert appropriate waiting time or to immediately schedule a job when the machine is idle and some jobs are available. Anderson and Potts [1] prove that the delayed shortest weighted processing time (D-SWPT) rule is optimal for the single machine problem to minimize the total weighted completion time. Actually, D-SWPT can be regarded as a generalization of the delay shortest processing time rule proposed in [8] for the non-weighted case.

In view of the idea proposed in both [8] and [1], we generalize the D-SWPT algorithm and proposed the 2.5-competitive AD-SWPT rule in [15]. Informally, when there are some idle machines and unscheduled jobs, the AD-SWPT rule chooses the job with the smallest ratio of the processing time to the weight as the candidate for processing. It also chooses the current time as the comparison reference to determine whether the selected candidate is immediately scheduled or not. Differently, it not only considers the candidate as in the single machine case, but also takes all the jobs being processed at other machines into account. Quantitatively, the AD-SWPT rule compares the current time with the average remaining processing time over all the machines to make a decision of processing. In this work, we further improve the AD-SWPT rule by introducing the machine number \( m \) as an algorithm parameter. With some notations listed in Table 1, the improved AD-SWPT rule can be described in detail as follows:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>the current decision time</td>
</tr>
<tr>
<td>( \hat{p}_j(t) )</td>
<td>the remaining processing time of job ( J_j ) at time ( t ) in an online schedule</td>
</tr>
<tr>
<td>( \sigma(\cdot) )</td>
<td>the schedule constructed by the improved AD-SWPT rule for a given instance. It also refers to the objective value of the schedule when no confusion arises.</td>
</tr>
<tr>
<td>( S_j )</td>
<td>the starting time of job ( J_j ) in the online schedule ( \sigma(\cdot) )</td>
</tr>
<tr>
<td>( C_j )</td>
<td>the completion time of job ( J_j ) in the online schedule ( \sigma(\cdot) )</td>
</tr>
<tr>
<td>( \pi(\cdot) )</td>
<td>the optimal offline schedule for a given instance. It also refers to the objective value of the schedule when no confusion arises.</td>
</tr>
</tbody>
</table>

The improved AD-SWPT rule: Whenever there is an idle machine and some jobs are available, choose a job with the smallest value of the ratio \( p_j/w_j \) (hereafter referred to as the weighted processing time) among all the arrived and unscheduled jobs. For example, \( J_i \) is chosen. Calculate the total remaining processing time at all the busy machines at time \( t \). The value can be represented as \( \sum_{S_j \leq t} \hat{p}_j(t) \) according to the notations in Table 1. Then if

\[
\frac{p_i + \sum_{S_j \leq t} \hat{p}_j(t)}{m} \leq \alpha t,
\]

schedule \( J_i \) from \( t \) at the idle machine; otherwise, wait until the next time and repeat the whole procedure above, where \( \alpha \) is a parameter to be designed.

We note that the improved AD-SWPT is reduced to the AD-SWPT rule in [15] when \( \alpha \) is equal to 1. This reduction implies that the improved AD-SWPT can perform at least as well as the AD-SWPT rule. At the same time, the improved
AD-SWPT is further reduced to the D-SWPT rule in [1] for the single machine problem. Such a reduction implies that the improved rule can be optimal for the single machine case. Intuitively, when \( m \) tends to infinity and more machines are available, less waiting time has to be inserted in the online schedule in order to resolve the dilemma where an urgent job arrives immediately after all the machines are occupied. The intuition makes us believe that \( \alpha \) should be more than 1 for the case of multiple machines in order to achieve a better competitive performance. The formal result is given in Theorem 2.1 and will be proved in the next section.

**Theorem 2.1.** For \( Pm|\sum w_j C_j, \alpha \)\( m \leftarrow \alpha = m-1+\sqrt{17m^2-2m+1 \over 4m} \) is \((1+\alpha)\)-competitive.

3. Competitive analysis of the improved AD-SWPT rule.

3.1. Preliminaries. For any given positive value of \( \alpha \), the improved AD-SWPT rule works as a directly waiting strategy. The generated schedule contains some idle time intervals due to the waiting strategy. For the sake of convenience, we state that one machine is “idle”/“busy” at the time \( t \) if the machine remains idle/busy during the interval of \((t-\varepsilon, t+\varepsilon)\), where \( \varepsilon \) is an infinitely small positive value. In order to differentiate the switching time points between the busy and idle states, we further state that the time \( t \) is a “starting point of busy time” (hereafter referred to as \( \text{SPoint} \)) if the machine remains idle in \((t-\varepsilon, t)\) and busy in \((t, t+\varepsilon)\), and that the time \( t \) is an “ending point of busy time” (hereafter referred to as \( \text{EPoint} \)) if the machine is busy in \((t-\varepsilon, t)\) and idle in \((t, t+\varepsilon)\).

Assume that jobs are processed in the order of \( J_1, J_2, \ldots, J_n \) in terms of their starting times in the online schedule by the improved AD-SWPT rule. We can further partition the scheduling queue into sub-queues such that jobs within each sub-queue are ordered according to the SWPT rule, with the last job of a sub-queue having a greater weighted processing time than that of the first job of the succeeding sub-queue. To simplify the presentation, we list some instances with specified structure in Table 2 with processing sub-queues illustrated in Figure 2. We also use these notations to refer to the corresponding sets including all the instances with the specified structure when no confusion arises.

**Table 2.** Three types of instances with specified structure

| \( I_1 \) | An instance for \( Pm|\sum w_j C_j, \alpha \)\( m \leftarrow \alpha = m-1+\sqrt{17m^2-2m+1 \over 4m} \) is \((1+\alpha)\)-competitive.
| \( I_2 \) | An instance where each job has the same weighted processing time.
| \( I_3 \) | An instance which satisfies that jobs in the last sub-queue in the improved AD-SWPT schedule have the same weighted processing time with weights tending to positive infinity.

3.2. Instance reduction. Since it is very difficult to directly analyze the performance for an arbitrary instance, we wish that we can reduce the searching ranges for the worst-case instance. This is just the original motivation of the instance reduction based method, which is commonly used in [15, 16, 17]. The basic idea behind is to modify an arbitrary instance such that it has a worse performance ratio as well as a more special structure of which we can take advantage to analyze the performance ratio.
Next we show in Lemma 3.1 that any instance $I_1$ can be reduced to one of two new instances $I_2$ or $I_3$ with the performance ratio not decreasing. Here we only give some intuitive explanation while referring readers to [15] for the detailed proof. As mention above, jobs within each sub-queue are ordered according to the SWPT rule. For any $I_1$, we can multiply the weights of some jobs by a parameter $\delta$ with not changing the mutual relations of the weighted processing times among jobs. The result is that jobs are scheduled in the same time intervals as in $\sigma(I_1)$ after this modification. The fact means that the objective value of the online schedule for the modified instance is a monotonously increasing linear function with respect to $\delta$. At the same time, the optimal objective value is a concave function with respect to $\delta$ since any feasible schedule remains feasible after modification and the optimal schedule is the one with the minimal objective value among all the feasible schedules. Combining the observations above with Lemma 3.1 in [16] which states that the ratio of a convex function to a concave function is maximized at one endpoint of the supported interval, we can modify $I_1$ to an intermediate instance with a worse performance ratio by setting $\delta$ to a specified value. By repeatedly applying the modification, we can reduce $I_1$ to $I_2$ or $I_3$.

Lemma 3.1. [15] For any instance $I_1$, a new instance $I_2$ or $I_3$ can be constructed through modifying the weights in $I_1$, such that

$$\frac{\sigma(I_1)}{\pi(I_1)} \leq \max \left\{ \frac{\sigma(I_2)}{\pi(I_2)}, \frac{\sigma(I_3)}{\pi(I_3)} \right\}.$$  \hspace{1cm} (2)

3.3. The lower bound. An appropriate lower bound on the optimal schedule has to be established in order to further analyze the performance ratios of $I_2$ and $I_3$. In this work, we utilize the lower bound proposed in [2]. In order to make this work self-contained, we give a brief introduction about some related concepts and lemmas.

First let us introduce the concepts of mean-busy-time and LP schedule.
Definition 3.2. [6] Given a preemptive schedule, the mean-busy-time of a job $J_j$ is defined as

$$M_j = \frac{1}{p_j} \int_{r_j}^{T} \delta_j(t) \cdot t \, dt,$$

where $T$ is the schedule horizon, i.e., all the jobs are completed by $T$, and $\delta_j(t)$ is the indicator function of the processing of job $J_j$ at time $t$, i.e., $\delta_j(t) = 1$ when $J_j$ is being processed at time $t$, otherwise $\delta_j(t) = 0$.

Definition 3.3. [6] For any instance of the single machine problem $1|\sum\text{w}\text{j}\cdot C_j|\sum\text{w}\text{j}$, at any point in time, schedule the job with the smallest weighted processing time ($p_j/w_j$), then the resulting preemptive schedule is called LP schedule.

For a non-preemptive schedule, the following relation can be readily discovered between the mean-busy-time and the completion time.

$$M_j = C_j - \frac{p_j}{2}.$$  \hfill (4)

Taking the mean-busy-times of jobs as optimization variables, Goemans et al. [6] develop a linear programming relaxation for $1|\sum\text{w}\text{j}\cdot C_j|\sum\text{w}\text{j}$ and obtain a lower bound for the problem. They also prove that the relaxation can be optimally solved by the LP schedule for the problem. Chou et al. [2] further extend the lower bound to the parallel-machine problem by introducing a virtual $m$-times faster single machine problem.

Lemma 3.4. [2] For any instance $I = \{r_j, p_j, w_j\}$ of $\text{Pm} | r_j, \text{pmtn} | \sum w_j C_j$, let $\mu(I)$ be the optimal objective value. Construct a single machine instance $I' = \{r_j, p_j/m, w_j\}$ (hereafter referred to as the virtual $m$-times faster single machine problem). Denote the mean-busy-time of job $J_j$ in the LP schedule of $I'$ by $M_j^{LP}$. Then

$$\mu(I) \geq \sum_{j \in I} w_j M_j^{LP} + \frac{1}{2} \sum_{j \in I} w_j p_j.$$  \hfill (5)

Since the preemptive problem is a trivial relaxation of the non-preemptive problem, the lower bound in Lemma 3.4 is clearly applicable to $P|r_j|\sum w_j C_j$. We further consider the special case where all the jobs are associated with the same weighted processing time. For any instance in this case, considering the corresponding virtual $m$-times faster single machine problem, we can construct its LP schedule by the FCFS (First Come First Service) strategy because all the jobs have the same weighted processing time. Along with (4), we can readily obtain the following corollary.

Corollary 1. For an instance $I$ of $P|\text{r}_j, \text{online}|\sum w_j C_j$ with each job having the same weighted processing time, construct a non-preemptive schedule according to FCFS for the corresponding virtual $m$-times faster single machine problem. Denote the completion time of job $J_j$ in the FCFS schedule by $C_j^{FF}$. Then

$$\pi(I) \geq \sum_{j \in I} w_j C_j^{FF} + \frac{1}{2} (1 - \frac{1}{m}) \sum_{j \in I} w_j p_j.$$  \hfill (5)

Hereafter we refer to the lower bound in Corollary 1 as the LP lower bound, and denote it by $LB^m(\cdot)$. Based on the LP lower bound, we can analyze the performance ratios of $I_2$ and $I_3$. In the following analysis, we will repeatedly handle calculating
the total weighted completion time of some jobs with the same weighted processing
time. To simplify the proof, we first give a compact expression for the calculation,
which can be derived by a direct algebraic simplification.

**Statement 1.** Let $J_1, J_2, \cdots, J_n$ be a sequence of jobs with the same weighted
processing time, for example, $p_j/w_j = 1/\eta$ for $j = 1, 2, \cdots, n$. Assume that these
jobs are continuously processed starting from time $t$ at a single machine. Then the
total weighted completion time of these jobs can be expressed as

$$
\sum_{j=1}^{n} w_j C_j = \eta t \sum_{j=1}^{n} p_j + \frac{\eta}{2} \left( \sum_{j=1}^{n} p_j \right)^2 + \frac{\eta}{2} \sum_{j=1}^{n} p_j^2.
$$

3.4. The performance ratio analysis of the instance $I_2$ and $I_3$.

**Lemma 3.5.** For any instance $I_2$, the online schedule obtained by the improved
AD-SWPT rule with $\alpha \geq 1$ satisfies

$$
\frac{\sigma(I_2)}{\pi(I_2)} \leq \max\{1 + \alpha, \frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha}\}.
$$

**Proof.** It does not change the performance ratio to multiply the weights of all the
jobs by a positive constant. We can normalize the ratio of $p_j/w_j$ to 1 by letting
$w_j = p_j$ since all the jobs in $I_2$ have the same weighted processing time.

Consider the latest $S\text{Point}$ in $\sigma(I_2)$, and denote it as $r_L$. The “latest” implies
that jobs are continuously processed after $r_L$ at each machine without idle time
between jobs. Now we analyze the performance ratio by two cases.

**Case 1.** There does not exist a job which is released before $r_L$ and is
scheduled at, or after, $r_L$ in $\sigma(I_2)$. See Figure 3(a). Consider these jobs
that start processed at, or after, $r_L$. According to the increasing order of their staring
times, denote these jobs by $J_1, J_2, \cdots, J_n$. The assumption in this case implies that
these jobs must be released at, or after, $r_L$. Furthermore, these jobs have no effect
on jobs starting before $r_L$. Construct an intermediate instance $I'_2$ which includes
all other jobs in $I_2$ except $J_1, J_2, \cdots, J_n$. Then we have

$$
\sigma(I_2) = \sigma(I'_2) + \sum_{j=1}^{n} (S_j + p_j) w_j.
$$
Jobs are continuously processed after $r_L$ at each machine. So we can limit the starting time of the $j$th job in $\{J_1, J_2, \cdots, J_n\}$ as

$$S_j \leq r_L + \sum_{i<n} S_{i<n} \hat{p}_i(r_L) + \sum_{1\leq i<j} p_i/m, \quad j = 1, 2, \cdots, n, \quad (9)$$

where the second term is to average the total processing time which have to be finished between $r_L$ and $S_j$ over all the machines.

Let $\sum_{i<n} S_{i<n} \hat{p}_i(r_L) := A$, and $\sum_{j=1}^n p_j := B$. Along with (8) and (9), we can limit $\sigma(I_2)$ by an upper bound as

$$\sigma(I_2) = \sigma(I_2') + \sum_{j=1}^n (S_j + p_j)w_j \leq \sigma(I_2') + \sum_{j=1}^n \left( r_L + \frac{A + \sum_{i=1}^{j-1} p_i}{m} + p_j \right) p_j = \sigma(I_2') + \sum_{j=1}^n \left( r_L + \frac{A}{m} + \frac{\sum_{i=1}^{j} p_i}{m} \right) p_j + (1 - \frac{1}{m}) \sum_{j=1}^n (2m) p_j^2 \quad (10)$$

$$= \sigma(I_2') + (r_L + \frac{A}{m})B + \frac{B^2}{2m} + (1 - \frac{1}{m}) \sum_{j=1}^n p_j^2. \quad (11)$$

The second term in (10) can be regarded as the total weighted completion time of the jobs of $J_1, J_2, \cdots, J_n$, which are continuously processed starting from the time $r_L + A/m$ on a single machine, with the processing time of each job multiplied by a constant of $1/m$. So this term can be simplified according to Statement 1.

Consider the set of $\{J_1, J_2, \cdots, J_n\}$ as a separate instance, and further relax the release times of all the jobs to $r_L$, then we can develop a lower bound on the optimal schedule $\pi(I_2)$ according to Corollary 1.

$$\pi(I_2) \geq \pi(I_2') + \pi(\{J_1, J_2, \cdots, J_n\}) = \pi(I_2') + B^m(\{J_1, J_2, \cdots, J_n\})$$

$$= \pi(I_2') + \sum_{j=1}^n w_j CFF + \frac{1}{2} (1 - \frac{1}{m}) \sum_{j=1}^n w_j p_j$$

$$= \pi(I_2') + r_L B + \frac{B^2}{2m} + \frac{1}{2} \sum_{j=1}^n p_j^2, \quad (12)$$

where $CFF$ is the completion time of $J_j$ in the FCFS schedule for the virtual $m$-times single machine problem related to $\{J_1, J_2, \cdots, J_n\}$. The last equation is simplified according to Statement 1.

According to the improved AD-SWPT rule, we have $\sum_{i<n} S_{i<n} \hat{p}_i(r_L)/m \leq \alpha r_L$, so $A/m \leq \alpha r_L$. Combining (11) and (12), we have

$$\frac{\sigma(I_2)}{\pi(I_2)} \leq \max \left\{ \frac{\sigma(I_2')}{\pi(I_2')}, \frac{(r_L + \frac{A}{m})B + \frac{B^2}{2m} + (1 - \frac{1}{m}) \sum_{j=1}^n p_j^2}{r_L B + \frac{B^2}{2m} + \frac{1}{2} \sum_{j=1}^n p_j^2} \right\} \leq \max \left\{ \frac{\sigma(I_2')}{\pi(I_2')}, \frac{(r_L + \frac{A}{m})B}{r_L B} \right\}.$$
\[ \leq \max \left\{ \frac{\sigma(I_2)}{\pi(I_2)}, 1 + \alpha \right\}. \quad (13) \]

Case 2. There exists at least a job \( J_k \) which is released before \( r_L \) and is scheduled at, or after, \( r_L \) in \( \sigma(I_2) \). According to the improved AD-SWPT rule, \( J_k \) must satisfy

\[ p_k + \frac{\sum_{i \leq m} p_i}{m} \geq \alpha r_L. \quad (14) \]

Otherwise \( J_k \) would be scheduled before \( r_L \) because there exists an idle machine before \( r_L \) since \( r_L \) is an SPoint.

See Figure 3(b). Consider these jobs which are completed after \( r_L \). According to the increasing order of their starting times, denote these jobs by \( J_1, J_2, \cdots, J_n \). Construct an intermediate instance \( I'_2 \) which includes all the other jobs in \( I_2 \) except \( J_1, J_2, \cdots, J_n \). Divide the set of \( \{J_1, J_2, \cdots, J_n\} \) into two subsets as follows:

\[ Q_1 = \{J_j|S_j < r_L, C_j > r_L\} \cup \{J_k\}, \]
\[ Q_2 = \{J_j|S_j \geq r_L\}\{J_k\}. \]

Let \( \sum_{j \in Q_1} p_j := A \), and \( \sum_{j \in Q_2} p_j := B \). Similar to (9), we have

\[ S_j \leq r_L + \frac{\sum_{1 \leq i < j} p_i}{m}, \quad j = 1, 2, \cdots, n. \quad (15) \]

Then similar to the derivation of (11), we can limit \( \sigma(I_2) \) as

\[ \sigma(I_2) \leq \sigma(I'_2) + r_L(A + B) + \frac{(A + B)^2}{2m} + (1 - \frac{1}{2m}) \sum_{j \in Q_1 \cup Q_2} p_j^2. \quad (16) \]

By relaxing the releasing times of jobs in \( \{J_1, J_2, \cdots, J_n\} \) to 0, similar to analysis of (12), we can limit \( \pi(I_2) \) as

\[ \pi(I_2) \geq \frac{(A + B)^2}{2m} + \frac{1}{2} \sum_{j \in Q_1 \cup Q_2} p_j^2. \quad (17) \]

In addition, we can derive \( A/m \geq \alpha r_L \) from (14). Furthermore we can obtain \( \sum_{j \in Q_1} p_j^2 \geq A^2/m \) because there are at most \( m \) jobs in \( Q_1 \). Combining these relations with (16) and (17), we can limit the performance ratio of \( I_2 \) as

\[ \frac{\sigma(I_2)}{\pi(I_2)} \leq \max \left\{ \frac{(A + B)^2}{2m} + (1 - \frac{1}{2m}) \sum_{j \in Q_1 \cup Q_2} p_j^2 \right\} \]
\[ \leq \max \left\{ \frac{\sigma(I'_2)}{\pi(I'_2)} \cdot \frac{(A + B)^2}{2m} + \frac{1}{2} \sum_{j \in Q_1 \cup Q_2} p_j^2 \right\} \]
\[ = \max \left\{ \frac{(A + B)^2}{2m} + \frac{2 - \alpha}{\alpha m} A^2 - \frac{\alpha - 1}{\alpha m} AB - \frac{1}{2m} B^2 - \frac{1}{2m} \sum_{j \in Q_1 \cup Q_2} p_j^2 \right\} \]
\[ \leq \max \left\{ \frac{(A + B)^2}{2m} + \frac{2 - \alpha}{\alpha m} A^2 - \frac{1}{2m} \sum_{j \in Q_1} p_j^2 \right\} \quad (18) \]
Next we analyze the performance ratio of $I_\sigma$ we can derive an upper bound of

\[ \frac{\sigma(I_\sigma)}{\pi(I_\sigma)} \leq \max \left\{ \frac{\sigma(I'_2)}{\pi(I'_2)} \cdot \frac{2 - \frac{\alpha}{2m} A^2 - \frac{1}{2m} A^2}{\frac{2^2}{m} + \frac{2}{m}} \right\} \]

(20)

\[ = \max \left\{ \frac{\sigma(I'_2)}{\pi(I'_2)} \cdot \frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha} \right\} \]

(21)

where (18) comes from $A/m \geq \alpha r_L$, and (19) is obtained by letting $B = 0$ with $\alpha \geq 1$, and (20) is due to $\sum_{j \in Q} \hat{p}_j^2 \geq A^2/m$.

The two cases above show that the performance ratio of $I_2$ can be bounded from above by $(1 + \alpha)$ or $(\frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha})$ or the performance ratio of an intermediate instance $I'_2$. Furthermore, in $\sigma(I'_2)$, $r_L$ is not the latest $\text{SPoint}$ anymore. Rewrite $I'_2$ as $I_2$ and repeat the analysis above. Ultimately the performance ratio of $I_2$ can be bounded from above by the maximum of $1 + \alpha$ and $\frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha}$.

**Lemma 3.6.** For any instance $I_3$, the online schedule obtained by the improved AD-SWPT rule with $\alpha \geq 1$ satisfies

\[ \frac{\sigma(I_3)}{\pi(I_3)} \leq \max\{1 + \alpha, \frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha}\}. \]

(22)

**Proof.** Jobs in the last sub-queue of $I_3$ have the same weighted processing time with weights tending to infinity. Let $w_j = \delta p_j$ for these jobs with $\delta$ tending to infinity. In the following analysis, the calculations are all carried out in the sense of limit with the signs of limits omitted.

Denote by $Q_\infty$ the set including all the jobs in the last sub-queue of $I_3$. Denote by $r_f$ the earliest releasing time of jobs in $Q_\infty$, and by $r_L$ the latest $\text{SPoint}$ in $I_3$. Next we analyze the performance ratio of $I_3$ by three cases.

**Case 1.** $r_L \leq r_f$. See Figure 4(a). Considering the time $r_f$, according to the improved AD-SWPT rule, we have

\[ \frac{\sum_{i \leq r_f} \hat{p}_i(r_f)}{m} \leq \alpha r_f. \]

(23)

After jobs which start before $r_f$ are completed, jobs in $Q_\infty$ are continuously processed. Assume that jobs in $Q_\infty$ start processed in the order of $J_1, J_2, \ldots, J_n$. Let $\sum_{j \in Q_\infty} p_j := B$. Construct an intermediate instance $I'_3$ by excluding jobs in $Q_\infty$. Similar to the analysis in the Case 1 in the proof of Lemma 3.5, along with (23), we can derive an upper bound of $\sigma(I_3)$ as

\[ \sigma(I_3) = \sigma(I'_3) + \sum_{j=1}^{n} (S_j + p_j) w_j \]

\[ \leq \sigma(I'_3) + \sum_{j=1}^{n} \left( r_f + \frac{\sum_{i \leq r_f} \hat{p}_i(r_f) + \sum_{i=1}^{j-1} p_i}{m} + p_j \right) \delta p_j \]

\[ \leq \sigma(I'_3) + \delta \left( (1 + \alpha) r_f B + \frac{B^2}{2m} + (1 - \frac{1}{2m}) \sum_{j=1}^{n} p_j^2 \right). \]

(24)

By relaxing the releasing times of jobs in $Q_\infty$ to $r_f$, we can similarly derive a lower bound of $\pi(I_3)$ as

\[ \pi(I_3) \geq \pi(I'_3) + \delta \left( r_f B + \frac{B^2}{2m} + \frac{1}{2} \sum_{j=1}^{n} p_j^2 \right). \]

(25)


\(\sigma(I'_3)\) and \(\pi(I'_3)\) are both limited values. When \(\delta\) tends to infinity, the relations above immediately imply
\[
\frac{\sigma(I_3)}{\pi(I_3)} \leq 1 + \alpha. \tag{26}
\]

**Case 2.** \(r_L > r_f\), furthermore, jobs being processed at \(r_L\) in \(\sigma(I_3)\) all belong to \(Q_\infty\). Similar to the proof of Lemma 3.5, we can construct an intermediate instance \(I'_3\) by deleting some jobs from \(I_3\) such that \(r_L\) is not the latest SP\textit{O}int in \(\sigma(I'_3)\) anymore. Furthermore, it holds that
\[
\frac{\sigma(I_3)}{\pi(I_3)} \leq \max \left\{ \frac{\sigma(I'_3)}{\pi(I'_3)}, \frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha}, 1 + \alpha \right\}. \tag{27}
\]

**Case 3.** \(r_L > r_f\), furthermore, there are one or more jobs which do not belong to \(Q_\infty\) and are being processed at \(r_L\) in \(\sigma(I_3)\). We analyze the performance ratio in terms of two sub-cases.

**Case 3.1.** There does not exist a job in \(Q_\infty\) which is released before \(r_L\) and is scheduled at, or after, \(r_L\) in \(\sigma(I_3)\). This case implies that jobs starting at, or after, \(r_L\) are all released at, or after, \(r_L\). Excluding these jobs, we can construct an intermediate instance \(I'_3\), which includes all the other jobs in \(I_3\). Let \(\sum_{S_j \geq r_L} p_j = B\). Similar to the analysis in the Case 1 in the proof of Lemma 3.5, we can derive
\[
\frac{\sigma(I_3)}{\pi(I_3)} \leq \frac{\sigma(I'_3) + \delta \left( (r_L + \frac{\sum_{S_j < r_L} p_j}{m})B + \frac{B^2}{2m} + (1 - \frac{1}{2m}) \sum_{S_j \geq r_L} p_j^2 \right)}{\pi(I'_3) + \delta \left( r_L B + \frac{B^2}{2m} + \frac{1}{2} \sum_{S_j \geq r_L} p_j^2 \right)}
\leq \max \left\{ \frac{\sigma(I'_3)}{\pi(I'_3)}, 1 + \alpha \right\}. \tag{28}
\]

**Case 3.2.** There exists at least a job \(J_k\) in \(Q_\infty\) which is released before \(r_L\) and is scheduled at, or after, \(r_L\) in \(\sigma(I_3)\). See Figure 4(b). First consider these jobs which do not belong to \(Q_\infty\) and are being processed at \(r_L\) in \(\sigma(I_3)\). According to the improved AD-SWPT rule, these jobs must start processed before \(r_f\). Denote

**Figure 4.** The schedule for \(I_3\) by the improved AD-SWPT rule
the set including these jobs by $Q'$. Let $\sum_{j \in Q'} \hat{p}_j(r_L) := A'$. It follows that

$$\frac{A'}{m} \leq \frac{\sum_{j \in Q'} \hat{p}_j(r_f)}{m} \leq \alpha r_f. \quad (29)$$

Since $J_k$ is released before $r_L$ and is scheduled at, or after, $r_L$ in $\sigma(I_3)$, we have

$$\frac{p_k + \sum_{\sigma_j < r_L} \hat{p}_j(r_L)}{m} \geq \alpha r_L. \quad (30)$$

Consider jobs in $Q_{\infty}$ which are completed after $r_L$. Define two sets as follows.

$$Q_1 = \{ J_j \in Q_{\infty} | S_j < r_L, C_j > r_L \} \cup \{ J_k \},$$

$$Q_2 = \{ J_j \in Q_{\infty} | S_j \geq r_L \} \setminus \{ J_k \}.$$

Construct an intermediate instance $I'_3$, which includes all the jobs in $I_3$ except jobs in $Q_1$ and $Q_2$. Let $\sum_{j \in Q_1} := A$, and $\sum_{j \in Q_2} := B$. Similar to the analysis in the Case 2 in the proof of Lemma 3.5, considering that jobs in $Q_1$ and $Q_2$ can be continuously processed after jobs in $Q'$ are completed, we can derive an upper bound on $\sigma(I_3)$ as

$$\sigma(I_3) \leq \sigma(I'_3) + \delta \left( (r_L + \frac{A'}{m})(A + B) + \frac{(A + B)^2}{2m} + (1 - \frac{1}{2m}) \sum_{j \in Q_1 \cup Q_2} p_j^2 \right). \quad (31)$$

By relaxing the releasing times of jobs in $Q_1$ and $Q_2$ to $r_f$, we can also derive a lower bound on $\pi(I_3)$ as

$$\pi(I_3) \geq \pi(I'_3) + \delta \left( r_f (A + B) + \frac{(A + B)^2}{2m} + \sum_{j \in Q_1 \cup Q_2} p_j^2 / 2 \right). \quad (32)$$

Inequality (30) implies that $(A + A')/m \geq \alpha r_L$. Furthermore, $\sum_{j \in Q_1} p_j^2 \geq A^2/m$ because there are at most $m$ jobs in $Q_1$. Combining these relations with (29), (31) and (32), we have

$$\frac{\sigma(I_3)}{\pi(I'_3)} \leq \max \left\{ \frac{\sigma(I'_3)}{\pi(I'_3)}, 1 + \alpha, \frac{A(A + B)}{2m} + \frac{(A + B)^2}{2m} + \frac{(1 - \frac{1}{2m}) \sum_{j \in Q_1 \cup Q_2} p_j^2}{(A + B)^2 / 2m + \sum_{j \in Q_1 \cup Q_2} p_j^2 / 2} \right\} \leq \max \left\{ \frac{\sigma(I'_3)}{\pi(I'_3)}, 1 + \alpha, \frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha} \right\}. \quad (33)$$

The first inequality is derived by applying $(A + A')/m \geq \alpha r_L$ and $A'/m \leq \alpha r_f$. The last inequality comes from similar analysis as (21) by relaxing $Q_2$ to an empty set, then applying $\sum_{j \in Q_1} p_j^2 \geq A^2/m$.

The three cases above show that the performance ratio of $I_3$ can be bounded from above by $(1 + \alpha)$ or $(\frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha})$ or the performance ratio of an intermediate instance $I'_3$. Furthermore, in $\sigma(I'_3)$, $r_L$ is not the latest SPoint anymore. Rewrite $I'_3$ as $I_3$ and repeat the analysis above. Ultimately the performance ratio of $I_3$ can be bounded from above by the maximum of $1 + \alpha$ and $\frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha}$. \qed

Following Lemmas 3.5 and 3.6, we can readily prove Theorem 2.1.

Proof of Theorem 2.1. According to Lemmas 3.5 and 3.6, we can obtain an upper bound on the competitive performance $\rho$ of the improved AD-SWPT rule with
\( \alpha \geq 1 \) as
\[
\rho \leq \max\{1 + \alpha, \frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha}\}. \tag{34}
\]
In order to minimize the upper bound, let \( 1 + \alpha = \frac{3}{2} - \frac{1}{2m} + \frac{1}{\alpha} \), we can obtain that
\[
\alpha = m - 1 + \sqrt{m^2 - 2m + 1}.
\]
The resulted upper bound is
\[
\rho \leq 1 + \alpha = 1 + \frac{m - 1 + \sqrt{m^2 - 2m + 1}}{4m}, \tag{35}
\]
which tends to 2.28 when the machine \( m \) tends to infinity.

4. Conclusions. In this work, we revisit the problem \( Pm|r_j, online|\sum w_jC_j \). We modify the AD-SWPT rule in [15] by introducing the machine number as a parameter. The modified rule is proved to be \( (1 + m - 1 + \sqrt{m^2 - 2m + 1}) \)-competitive, which tends to 2.28 when \( m \) tends to infinity. The result not only includes the algorithm presented in [1] as a special case, but also improves the results in [3] and [15].

REFERENCES


Received March 2013; 1st revision November 2013; 2nd revision January 2014.

*E-mail address:* jipingtao@gmail.com  
*E-mail address:* ronghuanh@gmail.com  
*E-mail address:* ltd@xmu.edu.cn