Volume 43
March 2014
ISSN 0305-0548
ELSEVIER
computers & operations research
Editor: Stefan Nickel
Available online at www.sciencedirect.com
ScienceDirect

Computers & Operations Research 43 (2014) 215–224

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A better online algorithm for the parallel machine scheduling to minimize the total weighted completion time[☆]

Jiping Tao [*]

Department of Automation, Xiamen University, Xiamen 361005, China

ARTICLE INFO

Available online 7 October 2013

Keywords:
Online scheduling
Parallel machine
Competitive ratio
Total weighted completion time
Instance reduction

ABSTRACT

The identical parallel machine scheduling problem with the objective of minimizing total weighted completion time is considered in the online setting where jobs arrive over time. An online algorithm is proposed and is proven to be $(2.5-1/2m)$-competitive based on the idea of instances reduction. Further computational experiments show the superiority over other algorithms in the average performance.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

In the online setting, parallel machines scheduling problem has been paid much attention [1–4]. In this work, we consider the classical online scheduling over identical parallel machines with the objective of minimizing the total weighted completion time. Formally, there is a sequence of jobs arriving over time and must be scheduled on m identical machines without preemption allowed. Each job J_j is characterized by a release date r_j, a processing time p_j and a weight w_j. All the information about one job is not revealed until it is released. Also the total number n of jobs cannot be known in advance. The goal is to find a schedule that minimizes the total weighted completion time, $\sum w_jC_j$, where C_j is the completion time of job J_j. The problem can be denoted by $Pm|r_j, online|\sum w_jC_j$ in terms of the standard three-field notation for scheduling problems in [5].

An online algorithm is often assessed by its competitive performance. An algorithm is called *ρ-competitive* if, for any instance, the objective function value of the schedule generated from this algorithm is no worse than ρ times the objective value of the optimal offline schedule [6].

For the case of $m=1$, the problem degenerates into a single machine problem. For this problem, it is well known that the optimal deterministic online algorithm is presented with the competitive ratio of 2 in [7]. For the case of multiple machines, the first deterministic online algorithm is given by Hall et al. [8]. They design a $(4+\varepsilon)$ -competitive online algorithm, where ε is an arbitrarily small positive constant. The result is improved to a value of 3.28 by using the technique of shifting releasing times [9]. To the best of our knowledge, the current best deterministic online algorithm is given by Correa and Wagner [1]. They propose a 2.618-competitive algorithm based on linear programming relaxation techniques and the concept of α-point. In a recent study [2], Sitters designs an online algorithm named by ONLINE(ε) by using the technique of shifting releasing times. He proves that the competitive ratio of ONLINE(ε) is not greater than $(1+1/\sqrt{m})^2(3e-2)/(2e-2)$, which is much greater than the current best value of 2.618 in [1] for less machine number, although which tends to 1.79 when the machine number m tends to infinity. When randomization is allowed, better competitive algorithms have been proposed. Detailed results can be found in [1,10,11].

In this work, we only consider the deterministic setting. By generalizing the algorithm for the single machine problem in [7], we propose an online algorithm for $Pm|r_j, online|\sum w_jC_j$ and prove that it is $(2.5-1/2m)$-competitive. The competitive analysis is based on the idea of instance reduction, which is first introduced for two semi-online single scheduling problems in [12,13]. In general, the method is in an attempt to directly search for the worst-case instance in the instance space. It starts from an arbitrary instance and modifies the instance such that it possesses the possible structure of the worst-case one with respect to the given online algorithm. The modification guarantees that the performance ratio does not decrease. Eventually, the reduction procedure ends up with one or several types of relatively simple instances with special structures. These special structures make it possible to analyze the performance ratios. Thus an upper bound on the competitive ratio can be derived.

The remaining sections are organized as follows. In Section 2, the online algorithm is presented. Its competitive performance is analyzed in Section 3. Computational experiments are shown in Section 4. Conclusions and remarks are given in Section 5.

[☆] This work is supported by the National Science Foundation of China (No. 11201391).

[*] Tel.: +86 18205992171.
E-mail address: jipingtao@gmail.com

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.09.016

2. The AD-SWPT online algorithm

It is well known that the delayed shortest weighted processing time (D-SWPT) rule proposed by Anderson and Potts [7] is an optimal online algorithm 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 [14] for the case of identical weights.

Inspired by the same idea in both [14] and [7], we further generalize the D-SWPT rule in [7] and construct an online algorithm for the parallel machine problem. Informally, when there are some idle machines and unscheduled jobs, we choose the job with the smallest ratio of the processing time to the weight as the candidate for processing. We also choose the current time as the comparison reference to determine whether the selected candidate is immediately scheduled or not. Differently, we not only consider the candidate as in the single machine case, but also take all the jobs being processed at other machines into account. Quantitatively, we compare the current time with the average remaining processing time over all the machines to make a decision of processing. We call the proposed rule the average delayed shortest weighted processing time rule. It is thereafter abbreviated to AD-SWPT, which is described in detail as follows with some notations listed in Table 1.

Algorithm AD-SWPT: Whenever there is one idle machine and some jobs are available, choose a job with the smallest value of the ratio p_j/w_j (hereafter we use weighted processing time to refer to the ratio) among all the arrived and unscheduled jobs. When ties occur, choose the one with the smallest index. For example, J_i is chosen. Calculate the total remaining processing time at all the busy machines at time t. The value can be written as $\sum_{S_j \le t} \hat{p}_j(t)$ according to the notations in Table 1. Then if

$$\frac{p_i + \sum_{S_j \le t} \hat{p}_j(t)}{m} \le t, \tag{1}$$

we schedule J_i from t at the idle machine; otherwise, wait until the next time and repeat the whole procedure above.

At the first glance, it can be readily found out that AD-SWPT is reduced to the D-SWPT rule in [7] for the case of single machine. Such a reduction implies that AD-SWPT is optimal for the single machine case. For the case of multiple machines, we will show that AD-SWPT performs better in the worst case than the best algorithm in the related literature. The result is given in the following theorem and will be proved in the next section.

Theorem 1. *The competitive ratio of the AD-SWPT algorithm lies in the interval of* $[2, 2.5 - 1/2m]$ *for the online scheduling problem of* $Pm|r_j, online|\sum w_j C_j$.

Table 1
Symbols/Notations description.

Notation	Description
t	the current decision time
$\hat{p}_j(t)$[a]	the remaining processing time of job J_j at time t in a feasible schedule
$\sigma(\cdot)$	the schedule constructed by AD-SWPT for a given instance. It also refers to the objective value of the schedule when no confusion arises
S_j	the starting time of job J_j in the online schedule $\sigma(\cdot)$
C_j	the completion time of job J_j in the online schedule $\sigma(\cdot)$
$\pi(\cdot)$	the optimal schedule for a given instance. It also refers to the objective value of the schedule when no confusion arises

[a] According to its definition, $\hat{p}_j(t)$ equals p_j if J_j have not started processing until t, $\hat{p}_j(t)$ equals zero if J_j have been completed before or at t, and $\hat{p}_j(t)$ equals the unfinished processing time if J_j is being processed at t.

3. Competitive analysis of the AD-SWPT algorithm

Although the AD-SWPT algorithm can be regarded as a direct and even intuitive extension from D-SWPT in [7] for the single machine problem, it seems difficult to follow the proof techniques in [7] to analyze the competitive performance of AD-SWPT. In this work, we develop a competitive analysis method based on the idea of instance reduction, which is first introduced for two semi-online single scheduling problems in [12,13]. Although the competitive ratio is defined as the maximal performance ratio achieved in the set of all the instances, an exhaustive search is infeasible since the set includes infinite number of instances. The idea of instance reduction is in an attempt to reduce the search space by showing that some instances cannot achieve greater performance ratios than other instances do. Thus we can analyze the worst-case performance in smaller sets. The key point is that the smaller sets are composed of some instances with some types of special structures, which permit further analysis of performance ratios.

For the AD-SWPT rule proposed in Section 2, we first show that the worst-case instances can be achieved among two types of instance sets. For each instance in the first set, each job is associated with the same weighted processing time. For each instance in the other set, there are some jobs with weights tending to positive infinity. For the two types of instances, we further prove that their performance ratios are not greater than 2.5–1/2m.

3.1. Structure of the AD-SWPT schedule

The AD-SWPT schedule includes some idle time intervals at some machines due to the waiting strategy of AD-SWPT. For the convenience of presentation, let us state that one machine is "*idle*" at the time t if the machine remains idle during the interval of $(t-\varepsilon, t+\varepsilon)$, and that one machine is "*busy*" at the time t if it is busy during the interval of $(t-\varepsilon, t+\varepsilon)$, where ε is an infinitely small positive number. 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*" (abbreviated to `SPoint`) at one machine 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*" (`EPoint`) at one machine if the machine is busy in $(t-\varepsilon, t)$ and idle in $(t, t+\varepsilon)$.

Next we show that the worst-case instances can be obtained among those whose AD-SWPT schedules do not involve a time t between the earliest `SPoint` and the latest `EPoint` over all the machines such that each machine is *idle* at t. The reason follows. Assume that $\sigma(I)$ does not possess the aforementioned characteristic. In other words, there exists a time t between the earliest `SPoint` and the latest `EPoint` when all the machines are *idle*. Thus we can split the instance I into two smaller instances that consist of jobs scheduled before t and after t. Denote the two instances by I' and I''. According to the AD-SWPT rule, we can readily discover that $\sigma(I')$ and $\sigma(I'')$ maintain the starting times of all the jobs same as in $\sigma(I)$, i.e.,

$$\sigma(I) = \sigma(I') + \sigma(I''). \tag{2}$$

Given any feasible schedule of I, we can construct two feasible schedules for I' and I'' by keeping the starting times unchanging. Since the optimal schedule is the one with the minimal objective value among all the feasible schedules, it follows that

$$\pi(I) \ge \pi(I') + \pi(I''). \tag{3}$$

Combining (2) and (3), we can obtain

$$\frac{\sigma(I)}{\pi(I)} \le \frac{\sigma(I') + \sigma(I'')}{\pi(I') + \pi(I'')} \le \max\left\{\frac{\sigma(I')}{\pi(I')}, \frac{\sigma(I'')}{\pi(I'')}\right\}, \tag{4}$$

i.e., at least one of the two smaller instances can achieve a performance ratio not less than the original instance I's. Therefore, from the perspective of worst-case instances, we only need to focus on those whose AD-SWPT schedules possess the aforementioned characteristic. Denote any one of these instances by I_1.

Assume that jobs are processed in the order of $J_1, J_2, \ldots, J_n$ in terms of their starting times in $\sigma(I_1)$. We can further partition jobs into sub-queues such that jobs within each sub-queue are ordered according to the WSPT 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. The processing order of I_1 is illustrated in Fig. 1(a).

3.2. Instance reduction

First let us introduce an important lemma in [13]. The lemma will be repeatedly utilized in the competitive analysis.

Lemma 2 (*Tao et al.* [13]). *Let f(x) and g(x) be two positive functions defined on the interval [u,v], furthermore f(x) is convex and g(x) is concave. Then $f(x)/g(x)$ reaches its maximum at either endpoint of the interval, i.e.,*

$$\frac{f(x)}{g(x)} \le \max\left\{\frac{f(u)}{g(u)}, \frac{f(v)}{g(v)}\right\} \quad \forall x \in [u, v].$$

It can be shown that the lemma remains valid when the interval is open at some endpoint on the condition that the limit of $f(x)/g(x)$ exists at the corresponding endpoint.

Next we will develop two lemmas to show that I_1 can be reduced to one of two new instances with the performance ratio not decreasing. For the two new instances, their AD-SWPT schedules are with more simple and special sub-queues structures. In the first instance, each job is associated with the same weighted processing time. For another instance, each job in the last sub-queue in its AD-SWPT schedule has the same weighted processing time with weights tending to infinity. Denote the two instances by I_2 and I_3.

Denote by I_1' an intermediate instance whose AD-SWPT schedule satisfies that all the jobs in the last sub-queue have the same weighted processing time. To simplify the presentation, along with I_1, I_2 and I_3 mentioned above, we restate these notations in Table 2 with processing sub-queues illustrated in Fig. 1.

Lemma 3. *For any instance I_1, an intermediate instance I_1' can be constructed by modifying the weights of jobs in I_1, such that*

$$\frac{\sigma(I_1)}{\pi(I_1)} \le \frac{\sigma(I_1')}{\pi(I_1')}. \tag{5}$$

Proof. As shown in Fig. 1(a), denote the first job in the last sub-queue in $\sigma(I_1)$ by J_f. According to the division rule of sub-queues in Subsection 3.1, all the jobs in the last sub-queue have weighted processing times not less than p_f/w_f. Denote the set of all the jobs of the last sub-queue by Q'. Then divide Q' into two subsets in terms of the relations between their weighted processing times and p_f/w_f:

$$Q_1' = \left\{ J_j | J_j \in Q', \frac{p_f}{w_f} < \frac{p_j}{w_j} < +\infty \right\} \tag{6}$$

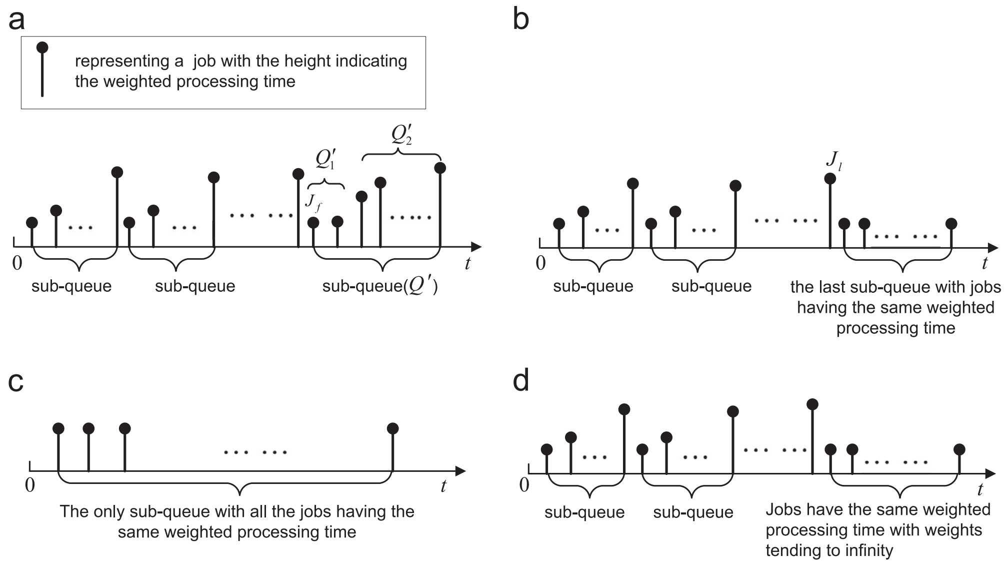
a
representing a job with the height indicating the weighted processing time
Q_1'
Q_2'
J_f
0
t
sub-queue
sub-queue
sub-queue(Q')
b
J_l
0
t
sub-queue
sub-queue
the last sub-queue with jobs having the same weighted processing time
c
0
t
The only sub-queue with all the jobs having the same weighted processing time
d
0
t
sub-queue
sub-queue
Jobs have the same weighted processing time with weights tending to infinity

Fig. 1. Processing sub-queues in terms of starting times in AD-SWPT schedules for I_1, I_1', I_2 and I_3.

Table 2
Four types of special instances.

I_1:	an instance for which there does not exist a time t between the earliest `SPoint` and the latest `EPoint` in the AD-SWPT schedule such that all the machines remain *idle* at t.
I_1':	an instance which not only possesses the same structure as I_1 but also satisfies that all the jobs in the last sub-queue in the AD-SWPT schedule have the same weighted processing time.
I_2:	an instance which not only possesses the same structure as I_1 but also satisfies that each job has the same weighted processing time.
I_3:	an instance which not only possesses the same structure as I_1' but also satisfies that jobs in the last sub-queue in the AD-SWPT schedule have positive infinite weights.

and

$$Q_2' = \left\{ J_j | J_j \in Q', \frac{p_j}{w_j} = \frac{p_f}{w_f} \right\}. \tag{7}$$

Construct an intermediate instance $I'(\delta)$ by modifying the weight w_j of each job J_j in Q_1' to δw_j where δ is a parameter to be chosen later. Let

$$\overline{\delta} = \frac{\min\left\{ \frac{p_j}{w_j} | J_j \in Q_1' \right\}}{p_f / w_f}. \tag{8}$$

For any $\delta \in (0, \overline{\delta}]$, this modification does not change the mutual relations of the weighted processing times among jobs in the last sub-queue. So jobs are scheduled in the same time intervals as in $\sigma(I_1)$ after this modification.

Since $\sigma(I'(\delta))$ maintains the starting times of jobs unchanging when δ changes in $(0, \overline{\delta}]$, $\sigma(I'(\delta))$ must be a monotonously increasing linear function with respect to δ. It is also a trivial convex function. Next we will explain how $\pi(I'(\delta))$ changes with δ. Note that any feasible schedule for $I'(\delta)$ remains feasible while δ changes in $(0, \overline{\delta}]$. Furthermore, given a feasible schedule for $I'(\delta)$, its objective value is a monotonously increasing linear function with respect to δ. Since the optimal schedule is the one with the minimal objective value among all the feasible schedules, $\pi(I'(\delta))$ becomes a minimum of multiple linear functions. It follows that $\pi(I'(\delta))$ must be a piecewise linear function with respect to δ, with its slope not increasing with δ. It also implies that $\pi(I'(\delta))$ is a concave function.

According to Lemma 2 and the convexity and concavity of $\sigma(I'(\delta))$ and $\pi(I'(\delta))$, we have that

$$\frac{\sigma(I'(\delta))}{\pi(I'(\delta))} \le \max\left\{ \lim_{\delta \to 0} \frac{\sigma(I'(\delta))}{\pi(I'(\delta))}, \frac{\sigma(I'(\overline{\delta}))}{\pi(I'(\overline{\delta}))} \right\}. \tag{9}$$

Two cases exist:

- If the maximum of the right-hand side of (9) is achieved at the first term, all the jobs in Q_1' have weights tending to 0. These jobs will be processed at the last both in $\sigma(I'(\delta \to 0))$ and $\pi(I'(\delta \to 0))$. They can be deleted since they contribute nothing to the objective value and have no effect on other jobs.
- If the maximum of the RHS of (9) is achieved at the second term, i.e., δ is chosen as $\overline{\delta}$, then there is at least one job belonging to Q_1' whose weighted processing time is modified to p_f / w_f according to (8).

For the resulting instances $I'(\delta \to 0)$ or $I'(\overline{\delta})$ in the two cases above, update Q_1' and Q_2' according to (6) and (7). If there remain jobs in Q_1', modify their weights by repeating the procedure above. The expected instance I_1' can be eventually obtained where all the jobs in the last sub-queue have the same weighted processing time of p_f / w_f. □

Lemma 4. *For any I_1', either an I_2 or an I_3 can be constructed by modifying the weights of jobs in I_1', such that*

$$\frac{\sigma(I_1')}{\pi(I_1')} \le \max\left\{ \frac{\sigma(I_2)}{\pi(I_2)}, \frac{\sigma(I_3)}{\pi(I_3)} \right\}. \tag{10}$$

Proof. Assume that $\sigma(I_1')$ is composed of K sub-queues. If $K=1$, I_1' is clearly the expected instance I_2 in the lemma. Thus we only need to consider the case of $K > 1$. We will show that an intermediate instance I' can be constructed by modifying the weights, such that either $\sigma(I')$ is composed of $K-1$ sub-queues, or all the jobs in the last sub-queue of $\sigma(I')$ have positive infinite weights.

As shown in Fig. 1(b), denote by J_l the last job of the next-to-last sub-queue in $\sigma(I_1')$. Denote the set of all the jobs of the last sub-queue by Q'. All the jobs in Q' have the same processing time according to the definition of I_1', say p_f / w_f. Then p_f / w_f is less than p_l / w_l according to the division rule of sub-queues in Subsection 3.1.

Construct an intermediate instance $I'(\delta)$ by modifying the weight w_j of each job J_j in Q' to δw_j where δ is a parameter to be chosen later. Let

$$\underline{\delta} = \frac{p_f / w_f}{p_l / w_l}. \tag{11}$$

Similar to the analysis in the proof of Lemma 5, we can discover that $\sigma(I'(\delta))$ is a convex function with respect to δ, and $\pi(I'(\delta))$ is a concave function with respect to δ for $\delta \in [\underline{\delta}, +\infty)$. According to Lemma 2

$$\frac{\sigma(I'(\delta))}{\pi(I'(\delta))} \le \max\left\{ \frac{\sigma(I'(\underline{\delta}))}{\pi(I'(\underline{\delta}))}, \lim_{\delta \to +\infty} \frac{\sigma(I'(\delta))}{\pi(I'(\delta))} \right\}. \tag{12}$$

Two cases exist:

- The maximum of the RHS of (12) is achieved at the first term, the weights of all the jobs in Q' are modified to p_f / w_f. So in $\sigma(I'(\underline{\delta}))$ these jobs can be combined into the original next-to-last sub-queue according to the division rule of sub-queues in Subsection 3.1, i.e. $\sigma(I'(\underline{\delta}))$ is composed of $K-1$ sub-queues.
- The maximum of the RHS of (12) is achieved at the second term, all the jobs in Q' have positive infinite weights. The resulting instance $I'(\delta \to +\infty)$ is just the expected instance I_3 in the lemma.

If the first case occurs and $K-1$ remains greater than 1, we can repeatedly carry out the procedure above. Eventually we can obtain either the expected I_2 in the first case or I_3 in the second case. □

3.3. *Lower bound on the optimal schedule*

Note that we do not need to know what the optimal schedules look exactly like in the procedure of instance reduction in the previous subsection. However, 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.

First let us introduce the concepts of *mean-busy-time* and *LP schedule*.

Definition 5 (*Goemans et al.* [15]). Given a preemptive schedule, the *mean-busy-time* of a job J_j is defined as

$$M_j := \frac{1}{p_j} \int_{r_j}^{T} \delta_j(t) \cdot t \, dt, \tag{13}$$

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 6 (*Goemans et al.* [15]). For any instance of the single machine problem $1|r_j, pmtn|\sum w_j C_j$, 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:

$$M_j = C_j - \frac{p_j}{2}. \tag{14}$$

Taking the mean-busy-times of jobs as optimization variables, Goemans et al. [15] develop a linear programming relaxation for $1|r_j, pmtn|\sum w_j C_j$ 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. [16] further extend the lower bound to the parallel machine problem by introducing a virtual m-times faster single machine problem.

Lemma 7 (*Chou et al.* [16]). *For any instance* $I=\{r_j, p_j, w_j\}$ *of* $P|r_j, pmtn|\sum w_jC_j$, *let* $\mu(I)$ *be the optimal objective value. Construct a single machine instance* $I'=\{r_j, p_j/m, w_j\}$ *(hereafter we refer to it 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*

$$\mu(I) \geq \sum_{j\in I} w_j M_j^{LP} + \frac{1}{2}\sum_{j \in I} w_j p_j \tag{15}$$

Since the preemptive problem is a trivial relaxation of the non-preemptive problem, the lower bound in Lemma 7 is clearly applicable to $P|r_j|\sum w_jC_j$. 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 according to Definition 6 because all the jobs have the same weighted processing time. Along with (14), we can readily obtain the following corollary.

Corollary 8. *For an instance I of* $P|r_j|\sum w_jC_j$ *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*

$$\pi(I) \geq \sum_{j\in I} w_j C_j^{FF} + \frac{1}{2}\left(1-\frac{1}{m}\right)\sum_{j \in I} w_j p_j. \tag{16}$$

Hereafter we refer to the lower bound in Corollary 8 as the LP lower bound, and denote it by $LB^m(\cdot)$.

3.4. *performance analysis of I_2 and I_3*

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 9. Let $J_1, J_2, \ldots, J_n$ be a sequence of jobs with the same weighted processing time, for example, $p_j/w_j = 1/\eta$ for $i=1,2,\ldots,n$. 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

$$\sum_{j=1}^{n} w_jC_j = \eta t \sum_{j=1}^{n} p_j + \frac{\eta}{2}\left(\sum_{j=1}^{n} p_j\right)^2 + \frac{\eta}{2}\sum_{i=1}^{n} p_j^2 \tag{17}$$

Lemma 10.

$$\frac{\sigma(I_2)}{\pi(I_2)} \leq 2.5 – 1/2m \tag{18}$$

Proof. It does not change the performance ratio to multiply the weights of all the jobs by a positive constant. All the jobs in I_2 have the same weighted processing time, we can normalize the ratio of p_j/w_j to 1, i.e., w_j equals p_j.

Consider the latest `SPoint` 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 Fig. 2(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 $J_1, J_2, \ldots, J_n$. 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 the other jobs in I_2 except $J_1, J_2, \ldots, J_n$. Then we have

$$\sigma(I_2) = \sigma(I_2') + \sum_{j=1}^{n} (S_j + p_j) w_j. \tag{19}$$

Jobs are continuously processed after r_L at each machine. So we can limit the starting time of the jth job in $\{J_1, J_2, \ldots, J_n\}$ as

$$S_j \leq r_L + \frac{\sum_{S_i<r_L}\hat{p}_i(r_L) + \sum_{1\leq i<j} p_i}{m} \qquad j=1,2,\ldots,n, \tag{20}$$

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, and $\sum_{S_i<r_L}\hat{p}_i(r_L)$ represents the total remaining processing time at all the machines at time r_L. Let $\sum_{S_i<r_L}\hat{p}_i(r_L){:=}A$, and $\sum_{j=1}^{n} p_j{:=}B$. Along with (19), we can limit $\sigma(I_2)$ by an upper bound as

$$\begin{aligned}\sigma(I_2) &= \sigma(I_2') + \sum_{j=1}^{n}(S_j+p_j)w_j \\ &\leq \sigma(I_2') + \sum_{j=1}^{n}\left(r_L + \frac{A+\sum_{i=1}^{j-1} p_i}{m} + p_j\right)p_j \\ \sigma(I_2) &= \sigma(I_2') + \sum_{j=1}^{n}\left(r_L + \frac{A}{m} + \frac{\sum_{i=1}^{j} p_i}{m}\right)p_j + \left(1-\frac{1}{m}\right)\sum_{j=1}^{n} p_j^2\end{aligned} \tag{21}$$

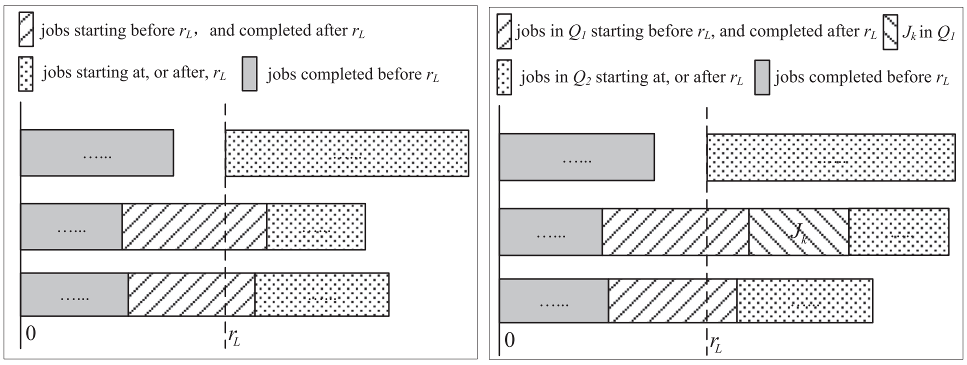
jobs starting before r_L, and completed after r_L
jobs starting at, or after, r_L
jobs completed before r_L
……
0
r_L
jobs in Q_1 starting before r_L, and completed after r_L
J_k in Q_1
jobs in Q_2 starting at, or after r_L
jobs completed before r_L
J_k
0
r_L

Fig. 2. The AD-SWPT schedule $\sigma(I_2)$. (a) Case 1 and (b) case 2.

$$\sigma(I_2)=\sigma(I_2')+\left(r_L+\frac{A}{m}\right)B+\frac{B^2}{2m}+\left(1-\frac{1}{2m}\right)\sum_{j=1}^{n}p_j^2. \quad (22)$$

The second term in (21) can be regarded as the total weighted completion time of the jobs of $J_1,J_2,\ldots,J_n$, 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 Lemma 9.

Consider the set of $\{J_1,J_2,\ldots,J_n\}$ as a separate instance, and further relax the release times of all the jobs to r_L, then we can develop a lower bound of the optimal schedule $\pi(I_2)$ according to Lemma 8:

$$\begin{aligned}\pi(I_2)&\geq\pi(I_2')+\pi(\{J_1,J_2,\ldots,J_n\})\\ &\geq\pi(I_2')+LB^m(\{J_1,J_2,\ldots,J_n\})\\ &=\pi(I_2')+\sum_{j=1}^{n}w_jC_j^{FF}+\frac{1}{2}\left(1-\frac{1}{m}\right)\sum_{j\in I}w_jp_j\\ &=\pi(I_2')+r_LB+\frac{B^2}{2m}+\frac{1}{2}\sum_{j=1}^{n}p_j^2, \quad (23)\end{aligned}$$

where C_j^{FF} is the completion time of J_j in the FCFS schedule for the virtual m-times single machine problem of $\{J_1,J_2,\ldots,J_n\}$. The last equation is simplified according to Lemma 9.

According to the AD-SWPT rule, we have $\sum_{S_j\leq r_L}\hat{p}_j(r_L)/m\leq r_L$, so $A/m\leq r_L$. Combining (22) and (23), we have

$$\begin{aligned}\frac{\sigma(I_2)}{\pi(I_2)}&\leq\max\left\{\frac{\sigma(I_2')}{\pi(I_2')},\frac{\left(r_L+\frac{A}{m}\right)B+\frac{B^2}{2m}+\left(1-\frac{1}{2m}\right)\sum_{j=1}^{n}p_j^2}{r_LB+\frac{B^2}{2m}+\frac{1}{2}\sum_{j=1}^{n}p_j^2}\right\}\\ &\leq\max\left\{\frac{\sigma(I_2')}{\pi(I_2')},2\right\}. \quad (24)\end{aligned}$$

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 $\sigma(I_2)$. See Fig. 2(b). According to the AD-SWPT rule, J_k must satisfy

$$\frac{p_k+\sum_{S_j<r_L}\hat{p}_j(r_L)}{m}\geq r_L. \quad (25)$$

Otherwise J_k would be scheduled before r_L.

Consider these jobs which are completed after r_L. According to the increasing order of their staring times, denote these jobs by $J_1,J_2,\ldots,J_n$. Construct an intermediate instance I_2' which includes all the other jobs in I_2 except $J_1,J_2,\ldots,J_n$.

Divide the set of $\{J_1,J_2,\ldots,J_n\}$ into two subsets as follows:

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

Let $\sum_{J_j\in Q_1}p_j:=A$, and $\sum_{J_j\in Q_2}p_j:=B$. Similar to (20), we have

$$S_j\leq r_L+\frac{\sum_{1\leq i<j}p_i}{m}\quad j=1,2,\ldots,n. \quad (26)$$

Then similar to the derivation of (22), we can limit $\sigma(I_2)$ as

$$\sigma(I_2)\leq\sigma(I_2')+r_L(A+B)+\frac{(A+B)^2}{2m}+\left(1-\frac{1}{2m}\right)\sum_{J_j\in Q_1\cup Q_2}p_j^2. \quad (27)$$

By relaxing the releasing times of jobs in $\{J_1,J_2,\ldots,J_n\}$ to 0, similar to analysis of (23), we can limit $\pi(I_2)$ as

$$\pi(I_2)\geq\pi(I_2')+\frac{(A+B)^2}{2m}+\frac{1}{2}\sum_{J_j\in Q_1\cup Q_2}p_j^2. \quad (28)$$

In addition, we can derive $A/m\geq r_L$ from (25). Furthermore we can obtain $\sum_{J_j\in Q_1}p_j^2\geq A^2/m$ because there are at most m jobs in Q_1. Combining these relations with (27) and (28), we can limit the performance ratio of I_2 as

$$\begin{aligned}\frac{\sigma(I_2)}{\pi(I_2)}&\leq\max\left\{\frac{\sigma(I_2')}{\pi(I_2')},\frac{r_L(A+B)+\frac{(A+B)^2}{2m}+\left(1-\frac{1}{2m}\right)\sum_{J_j\in Q_1\cup Q_2}p_j^2}{\frac{(A+B)^2}{2m}+\frac{1}{2}\sum_{J_j\in Q_1\cup Q_2}p_j^2}\right\}\\ &\leq\max\left\{\frac{\sigma(I_2')}{\pi(I_2')},\frac{\frac{A(A+B)}{m}+\frac{(A+B)^2}{2m}+\left(1-\frac{1}{2m}\right)\sum_{J_j\in Q_1\cup Q_2}p_j^2}{\frac{(A+B)^2}{2m}+\frac{1}{2}\sum_{J_j\in Q_1\cup Q_2}p_j^2}\right\}\\ &=\max\left\{\frac{\sigma(I_2')}{\pi(I_2')},2+\frac{\frac{A^2}{2m}-\frac{B^2}{2m}-\frac{1}{2m}\sum_{j\in Q_1\cup Q_2}p_j^2}{\frac{(A+B)^2}{2m}+\frac{1}{2}\sum_{j\in Q_1\cup Q_2}p_j^2}\right\}\\ &\leq\max\left\{\frac{\sigma(I_2')}{\pi(I_2')},2+\frac{\frac{A^2}{2m}-\frac{1}{2m}\cdot\frac{A^2}{m}}{\frac{A^2}{2m}+\frac{1}{2}\cdot\frac{A^2}{m}}\right\}\\ &=\max\left\{\frac{\sigma(I_2')}{\pi(I_2')},2.5-1/2m\right\}. \quad (29)\end{aligned}$$

The next-to-last inequality is obtained by letting $B=0$, then applying $\sum_{J_j\in Q_1}p_j^2\geq A^2/m$.

The two cases above show that we can limit the performance ratio of I_2 from above by $(2.5-1/2m)$ or the performance ratio of an intermediate instance I_2'. In $\sigma(I_2')$ there are no jobs starting processed at, or after, r_L. In other words, in $\sigma(I_2')$ r_L is not the latest `SPoint` anymore. Rewrite I_2' as I_2 and repeat the analysis above. Ultimately I_2' would become an empty set. It follows that the performance ratio of I_2 can be limited from above by 2.5-1/2m. □

Lemma 11.

$$\frac{\sigma(I_3)}{\pi(I_3)}\leq 2.5-1/2m \quad (30)$$

The proof is similar to that of Lemma 10 and is attached in Appendix A.

Summarizing the analysis above, we first apply the idea of instance reduction and show that the worst-case instances can be achieved among two types of instances by Lemmas 3 and 4. Then the performance ratios of the two types of instances are proved less than $2.5-1/2m$ in Lemmas 10 and 11.

Proof of Theorem 1. Following Lemmas 3, 10 and 11, we can directly obtain that AD-SWPT is $(2.5-1/2m)$-competitive. Next we will construct a special instance to show that the AD-SWPT algorithm is at most 2-competitive.

Consider an instance I_4 with $m+1$ jobs whose parameters are showed in Table 3, where ε is an arbitrarily small positive value. According to the AD-SWPT rule, J_1, J_2, …, and J_m are assigned to one of m machines respectively with the starting time of $S_j=j(m+1)/m(m+j)$ and the same completion time of $(m+1)/m$. J_{m+1} starts at time of $(m+1)/m$. It should start at its releasing time of $(m+1)/2m+\varepsilon$ in the optimal schedule since it has infinite

Table 3
Instance I_4 with $m+1$ jobs.

Job characteristics	J_1	J_2	…	J_j	…	J_m	J_{m+1}
r_j	0	0	…	0	…	0	$\frac{m+1}{2m}+\varepsilon$
p_j	1	$\frac{m+1}{m+2}$	…	$\frac{m+1}{m+j}$	…	$\frac{m+1}{m+m}$	0
w_j	1	$\frac{m+1}{2(m+2)}$	…	$\frac{m+1}{j(m+j)}$	…	$\frac{m+1}{m(m+m)}$	$w\to\infty$

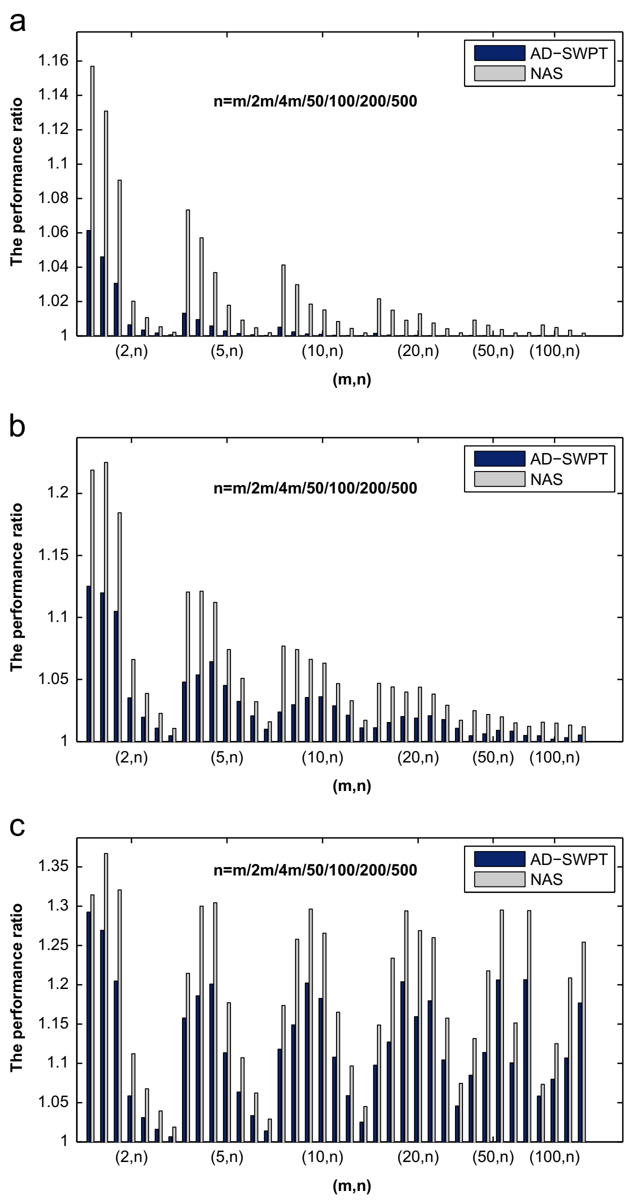

Fig. 3. The average performance ratio of AD-SWPT vs NAS at different pairs of (m,n) and three kinds of machine load. (a) Light load, (b) balanced load and (c) heavy load.

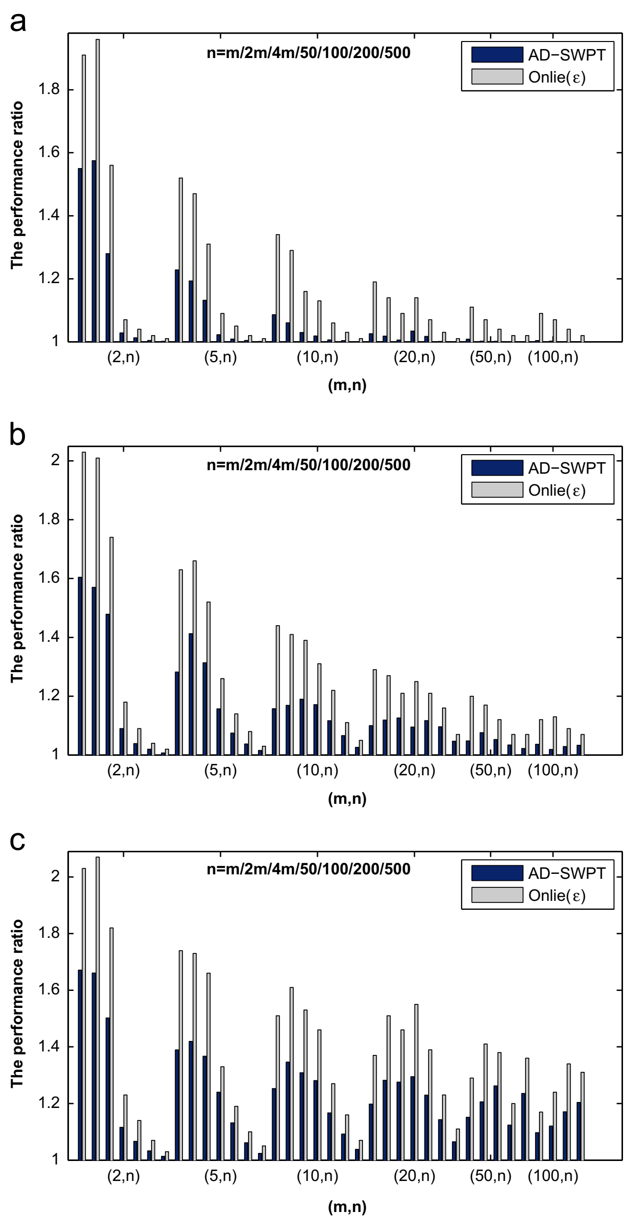

Fig. 4. The average performance ratio of AD-SWPT vs Online (ε) at different pairs of (m,n) and three kinds of machine load. (a) Light load, (b) balanced load and (c) heavy load.

weight. So we can obtain the performance ratio of the AD-SWPT on this instance:

$$\frac{\sigma(I_4)}{\pi(I_4)} \to \frac{\dfrac{m+1}{m}}{\dfrac{m+1}{2m}+\varepsilon} \to 2 \qquad \square \qquad (31)$$

4. Computational results

In this section, we perform a computational study to investigate the average performance for randomly generated instances. The proposed AD-SWPT algorithm is compared with the existing best deterministic online algorithm NAS in [1] and Online(ε) in [2]. We also compare the algorithm with the DSPT rule proposed in [3] for the non-weighted case

We generate the instance sets by similar methods in [17,18]. Three categories of instances are considered in terms of the machine load, say *light load*, *balanced load* and *heavy load*. We do this by generating jobs according to a Poisson process with a parameter of arrival rate λ, which indicates the average released processing time on each machine per unit time [18]. Let λ be 0.5, 1.0, and 3.0 for, respectively, light load, balanced load and heavy load. For all instances, processing times and weights are generated uniformly in [1, 100] with rounding off to integer numbers. We consider different pairs of machine number m and job number n by letting $m \in \{2, 5, 10, 20, 50, 100\}$ and $n \in \{m, 2m, 4m, 50, 100, 200, 500\}$. There are totally 37 different pairs of (m,n). For each pair of (m,n) and each category of machine load, we randomly generate 1000 instances and calculate the mean of the performance ratios of all the instance with respect to the online algorithms. Since it is very hard to calculate the offline optimal schedule, the LP-based lower bound in [16] is utilized. We first compare AD-SWPT with NAS and Online(ε). Figs. 3 and 4 show the statistic results in three different cases of machine load. By setting the weights of all the jobs to 1, we further compare AD-SWPT with DSPT and show the results in Fig. 5. From the statistic data, we surprisedly observe that the average

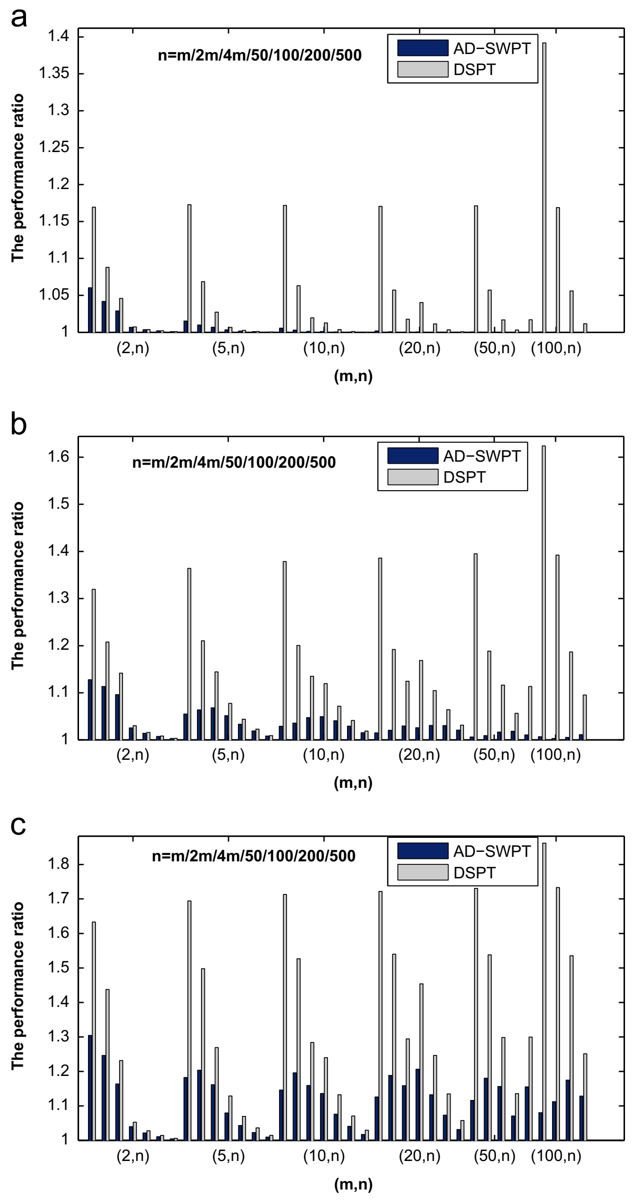

Fig. 5. The average performance ratio of AD-SWPT vs DSPT at different pairs of (m,n) and three kinds of machine load. (a) Light load, (b) balanced load and (c) heavy load.

performance ratio of the proposed AD-SWPT algorithm is consistently less than that of NAS, Online(ε) and DSPT for all pairs of (m,n) and all cases of machine load.

Sitter [2] proves that the competitive ratio of Online(ε) is not greater than $(1+1/\sqrt{m})^2(3e-2)/(2e-2)$. It is not hard to observe that AD-SWPT is superior to Online(ε) when $m \le 31$ in terms of the proved competitive performance, and conversely, Online(ε) is superior to AD-SWPT when $m > 31$. However, from the perspective of average performance, AD-SWPT surprisingly shows consistent superiority to Online(ε).

5. Conclusions

In this work, we consider the identical parallel machine online scheduling problem of minimizing the total weighted completion time. We design an online algorithm named by AD-SWPT and prove that it is $(2.5-1/2m)$-competitive. The result not only includes the algorithm presented in [7] as a special case, but also defeats the current best deterministic online algorithm presented in [1]. In the competitive analysis, we introduce an intuitive and systematic method. The method exploits the possible structure of the worst-case instance with respect to the given online algorithm. 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. The analysis method is deserved to be extended to other online algorithms in our further work. In order to investigate the average performance of the proposed algorithm, we further compare it with other algorithms and show the superiority.

Appendix A. The Proof of Lemma 11

Proof. Jobs in the last sub-queue of $\sigma(I_3)$ have the same weighted processing time with weights tending to infinity. Without loss of generality, let $w_j = \delta p_j$ for these jobs with δ tending to infinity. In the following calculating of performance ratios, they are all carried out in the sense of limit when δ tends to infinity, with the signs of limits omitted.

Denote by Q_∞ the set including all the jobs in the last sub-queue of $\sigma(I_3)$. Denote by r_f the earliest releasing time of jobs in Q_∞, and by r_L the latest `SPoint` in $\sigma(I_3)$. Next we analyze the performance ratio of I_3 by three cases.

Case 1: $r_L \le r_f$. Considering the time r_f, according to the AD-SWPT rule, we have

$$\frac{\sum_{S_i<r_f}\hat{p}_i(r_f)}{m} \le r_f. \tag{A.1}$$

After jobs which start before r_f are completed, jobs in Q_∞ are continuously processed. Assume that jobs in Q_∞ start processed in the order of $J_1, J_2, \ldots, J_n$. Let $\sum_{J_j \in Q_\infty} p_j := B$. Similar to the analysis in Case 1 in the proof of Lemma 10, along with (A.1), we can derive an upper bound of $\sigma(I_3)$ as

$$\begin{aligned}\sigma(I_3) &= O(1) + \sum_{j=1}^{n}(S_j+p_j)w_j \\ &\le O(1) + \sum_{j=1}^{n}\left(r_f + \frac{\sum_{S_i<r_f}\hat{p}_i(r_f) + \sum_{i=1}^{j-1}p_i}{m} + p_j\right)\delta p_j \\ &\le O(1) + \delta\left(2r_fB + \frac{B^2}{2m} + \left(1-\frac{1}{2m}\right)\sum_{j=1}^{n}p_j^2\right),\end{aligned} \tag{A.2}$$

where $O(1)$ indicates a limited value. By relaxing the releasing times of jobs in Q_∞ to r_f, we can similarly derive a lower bound of $\pi(I_3)$ as

$$\pi(I_3) \ge O(1) + \delta\left(r_fB + \frac{B^2}{2m} + \frac{1}{2}\sum_{j=1}^{n}p_j^2\right). \tag{A.3}$$

When δ tends to infinity, the relations above immediately imply

$$\frac{\sigma(I_3)}{\pi(I_3)} \le 2. \quad \square \tag{A.4}$$

Case 2: $r_L > r_f$, furthermore, jobs being processed at r_L in $\sigma(I_3)$ all belong to Q_∞. Similar to the proof of Lemma 10, we can construct an intermediate instance I_3' by deleting some jobs from I_3 such that r_L is not the latest `SPoint` in $\sigma(I_3')$ anymore. Furthermore, it holds that

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, 2.5-1/2m\right\}. \tag{A.5}$$

Case 3: $r_L > r_f$, furthermore, there are one or more jobs which do not belong to Q_∞ and are being processed at r_L *in* $\sigma(I_3)$. According

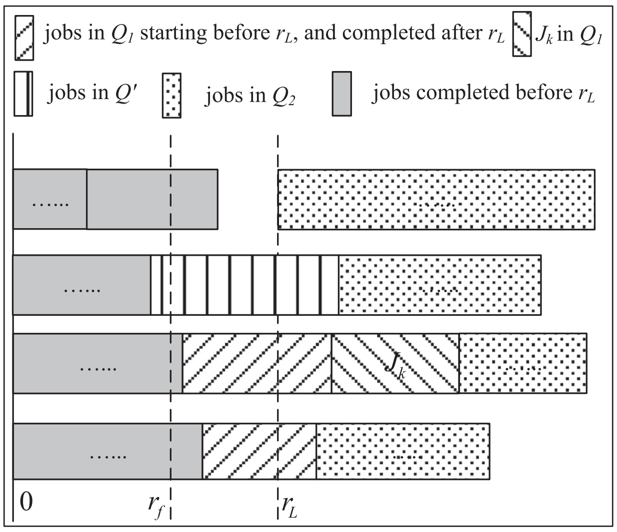
jobs in Q_1 starting before r_L, and completed after r_L
J_k in Q_1
jobs in Q'
jobs in Q_2
jobs completed before r_L
……
……
……
……
J_k
0
r_f
r_L

Fig. 6. The AD-SWPT schedule $\sigma(I_3)$.

to the AD-SWPT rule, these jobs must start before r_f. We further analysis the performance ratio in terms of two sub-cases.

Case 3.1: There does not exists a job in Q_∞ 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. Except these jobs, we can construct an intermediate instance I_3', which includes all the other jobs in I_3. Let $\sum_{S_j \ge r_L} p_j = B$. Similar to the analysis in Case 1 in the proof of Lemma 10, we can derive

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \frac{\sigma(I_3') + \delta\left(\left(r_L + \frac{\sum_{S_j < r_L} p_j}{m}\right)B + \frac{B^2}{2m} + \left(1 - \frac{1}{2m}\right)\sum_{S_j \ge r_L} p_j^2\right)}{\pi(I_3') + \delta\left(r_L B + \frac{B^2}{2m} + \frac{1}{2}\sum_{S_j \ge r_L} p_j^2\right)}$$
$$\le \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, 2\right\}. \tag{A.6}$$

Case 3.2: There exists at least a job J_k in Q_∞ which is released before r_L and is scheduled at, or after, r_L *in* $\sigma(I_3)$. See Fig. 6. First consider these jobs which do not belong to Q_∞ and are being processed at r_L in $\sigma(I_3)$. According to the AD-SWPT rule, these jobs must start processed before r_f. Denote the set including these jobs by Q'. Let $\sum_{j \in Q'} \hat{p}_j(r_L) := A'$. It follows that

$$\frac{A'}{m} \le \frac{\sum_{j \in Q'} \hat{p}_j(r_f)}{m} \le r_f. \tag{A.7}$$

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

$$\frac{p_k + \sum_{S_j < r_L} \hat{p}_j(r_L)}{m} \ge r_L. \tag{A.8}$$

Consider jobs in Q_∞ which are completed after r_L. Define two sets as follows:

$$Q_1 = \{J_j \in Q_\infty | S_j < r_L, C_j > r_L\} \cup \{J_k\}$$
$$Q_2 = \{J_j \in Q_\infty | S_j \ge r_L\} \backslash \{J_k\}.$$

Construct an intermediate instance I_3', which includes all the jobs in I_3 except jobs in Q_1 and Q_2. Let $\sum_{J_j \in Q_1} := A$, and $\sum_{J_j \in Q_2} := B$. Similar to the analysis in Case 2 in the proof of Lemma 10, 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

$$\sigma(I_3) \le \sigma(I_3') + \delta\left(\left(r_L + \frac{A'}{m}\right)(A+B) + \frac{(A+B)^2}{2m} + \left(1 - \frac{1}{2m}\right)\sum_{J_j \in Q_1 \cup Q_2} p_j^2\right). \tag{A.9}$$

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

$$\pi(I_3) \ge \pi(I_2') + \delta\left(r_f(A+B) + \frac{(A+B)^2}{2m} + \sum_{J_j \in Q_1 \cup Q_2} p_j^2/2\right). \tag{A.10}$$

Eq. (A.8) implies that $(A+A')/m \ge r_L$. Furthermore $\sum_{J_j \in Q_1} p_j^2 \ge A^2/m$ because there are at most m jobs in Q_1. Combining these relations with (A.(7), A.9) and (A.10), we have

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, 2, \frac{\frac{A}{m}(A+B) + \frac{(A+B)^2}{2m} + \left(1 - \frac{1}{2m}\right)\sum_{J_j \in Q_1 \cup Q_2} p_j^2}{(A+B)^2/2m + \sum_{J_j \in Q_1 \cup Q_2} p_j^2/2}\right\}$$
$$= \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, 2, 2 + \frac{\frac{A^2}{2m} - \frac{B^2}{2m} - \frac{1}{2m}\sum_{j \in Q_1 \cup Q_2} p_j^2}{(A+B)^2/2m + \sum_{j \in Q_1 \cup Q_2} p_j^2/2}\right\}$$
$$\le \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, 2.5 - 1/2m\right\}. \tag{A.11}$$

The first inequality is derived by applying $(A+A')/m \ge r_L$ and $A'/m \le r_f$. The last inequality is obtained by relaxing Q_2 to an empty set, then applying $\sum_{J_j \in Q_1} p_j^2 \ge A^2/m$.

The three cases above show that we can bound the performance ratio of I_3 from above by $(2.5 - 1/2m)$ 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_2 can be bounded from above by $2.5 - 1/2m$. □

References

[1] Correa J, Wagner M. LP-based online scheduling: from single to parallel machines. Mathematical Programming 2009;119(1):109–36.
[2] Sitters R. Efficient algorithms for average completion time scheduling. In: Eisenbrand F, Shepherd F, editors. Integer programming and combinatorial optimization. Lecture notes in computer Science, vol. 6080, 2010. p. 411–23.
[3] Liu P, Lu X. On-line scheduling of parallel machines to minimize total completion times. Computers & Operations Research 2009;36(9):2647–52.
[4] Krumke SO, Taudes A, Westphal S. Online scheduling of weighted equal-length jobs with hard deadlines on parallel machines. Computers & Operations Research 2011;38(8):1103–8.
[5] Pinedo M. Scheduling: theory, algorithms, and systems. Englewood Cliffs, NJ: Prentice Hall; 2002.
[6] Fiat A, Woeginger GJ. Competitive analysis of algorithms. In: Lecture notes in computer science, vol. 1442, 1998. p. 1–12.
[7] Anderson EJ, Potts CN. Online scheduling of a single machine to minimize total weighted completion time. Mathematics of Operations Research 2004;29 (3):686–97.
[8] Hall LA, Schulz AS, Shmoys DB, Wein J. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. Mathematics of Operations Research 1997;22(3):513–44.
[9] Megow N, Schulz AS. On-line scheduling to minimize average completion time revisited. Operations Research Letters 2004;32(5):485–90.
[10] Chakrabarti S, Phillips CA, Schulz AS, Shmoys DB, Stein C, Wein J. Improved scheduling algorithms for minsum criteria. In: Lecture notes in computer science, vol. 1099, 1996. p. 646–57.
[11] Schulz AS, Skutella M. Scheduling unrelated machines by randomized rounding. SIAM Journal on Discrete Mathematics 2002;15(4):450–69.
[12] Tao JP, Chao ZJ, Xi YG. A semi-online algorithm and its competitive analysis for a single machine scheduling problem with bounded processing times. Journal of Industrial and Management Optimization 2010;6(2):269–82.
[13] Tao JP, Chao ZJ, Xi YG, Tao Y. An optimal semi-online algorithm for a single machine scheduling problem with bounded processing time. Information Processing Letters 2010;110(8–9):325–30.
[14] Hoogeveen JA, Vestjens APA. Optimal on-line algorithms for single-machine scheduling. In: Lecture notes in computer science, vol. 1084, 1996. p. 404–14.

[15] Goemans MX, Queyranne M, Schulz AS, Skutella M, Wang Y. Single machine scheduling with release dates. SIAM Journal on Discrete Mathematics 2002;15 (2):165–92.
[16] Chou MC, Queyranne M, Simchi-Levi D. The asymptotic performance ratio of an on-line algorithm for uniform parallel machine scheduling with release dates. Mathematical Programming 2006;106(1):137–57.
[17] Savelsbergh MWP, Uma RN, Wein J. An experimental study of lp-based approximation algorithms for scheduling problems. INFORMS Journal on Computing 2005;17(1):123–36.
[18] Gu HY. Computation of approximate α-points for large scale single machine scheduling problem. Computers and Operations Research 2008;35(10):3262–75.