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.

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* $Pm|r_j, online|\sum w_jC_j$.

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

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