Computers & Operations Research 46 (2014) 91–101

Contents lists available at ScienceDirect

Computers & Operations Research

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

Exact and parallel metaheuristic algorithms for the single processor total weighted completion time scheduling problem with the sum-of-processing-time based models

Radosław Rudek*

Wrocław University of Economics, Komandorska 118/120, 53-345 Wrocław, Poland

ARTICLE INFO

Available online 24 January 2014

Keywords:
Scheduling
Learning
Deteriorating
Aging
Dynamic programming
Parallel algorithm

ABSTRACT

In this paper, the single processor scheduling problem to minimize the total weighted completion times is analysed, where the processing times of jobs are described by functions dependent on the sum of the normal processing times of previously processed jobs, which can model learning or aging (deteriorating) effects. We construct the exact pseudopolynomial time algorithm based on the dynamic programming, which solves the problem, where the processing time of each job is described by an arbitrary stepwise function. Moreover, the parallel metaheuristic algorithms are provided for the general version of the problem with arbitrary sum-of-processing time based models. The efficiency of the proposed algorithms is evaluated during numerical analysis.

1. Introduction

The presence of the learning effect in computers or manufacturing systems has usually a positive impact on their performances. For instance, machine learning [6,25,31], intelligent agents [6,23], advanced algorithms [1], iterative learning control methods [2,5] or human workers [16,37,42] can adapt to occurring changes and (often autonomously) increase their efficiency, which can decrease time or cost of processed jobs. Moreover, in such cases the optimization objectives can be further controlled by the sequence of processed jobs, which in fact utilizes learning abilities (see [3,7]. Since this approach does not require changes in a structure of an optimized system and due to the growing meaning of autonomous learning, it is not surprising that this direction of research has attracted particular attention during last few years, especially in scheduling theory (e.g., [4,9,17,22,39]).

In the scheduling context, the learning effect is modelled by job processing times described by non-increasing functions dependent on the experience of the processor. This experience is usually equivalent to the number of previously processed jobs (*position based/dependent learning models*, e.g., [3,7,43]) or to the sum of the normal processing times of previously processed jobs (*sum-of-processing time based learning models*, e.g., [8,18,20,33,34,38]), where the normal processing time of a job is defined as the time required to process a job if no learning exists. The discussion on differences between these two approaches is presented in [4].

On the other hand, the efficiency of a processor (e.g., a human worker, CNC machine, tool, chemical cleaning bath) can also decrease with the processed jobs (see [24,29,10]). This phenomenon in scheduling theory is called deteriorating or the aging effect and in particular it can be modelled by job processing times described by non-decreasing functions dependent on the sum of the normal processing times of already processed jobs (e.g., [11,19,35]).

In this paper, we will analyse the single processor scheduling problem to minimize the total weighted completion times, where the processing time of each job is described by a function dependent on the sum of the normal processing times of previously processed jobs, which can be non-increasing (learning) or non-decreasing (deteriorating). Since a survey on this group of problems with the learning effect was provided in [26], whereas different sum-of-processing-time based models were discussed in [32] (learning) and [27] (deteriorating), we will only briefly complement it by results related with deteriorating models.

Namely, Sun [30] analysed the single-processor total weighted completion time minimization problem, where the processing time of job j that is scheduled in the vth position in a sequence was given by $\tilde{p}_j(v)=p_j(1+\sum_{l=1}^{v-1}p_{[l]})^a v^b$, where p_j is the normal processing time of job j, $[l]$ denotes the index of the lth job in a sequence, $a\geq 1$ and $b<0$ are deteriorating and learning indices, respectively. It was proved that the Shortest Processing Time (SPT) rule solves optimally the special case of the analysed problem. Later on, it was shown (see [13]) that the same rule is optimal for a special case of the considered problem with the following model: $\tilde{p}_j(v)=p_j(1+\sum_{l=1}^{v-1}p_{[l]})^a b^{v-1}$, where $a>1$ and $0<b\leq 1$ are deteriorating and learning indices, respectively. In [19], a more general

* Tel.: +48 71 368 0378; fax.: +48 71 368 0376.
E-mail addresses: rudek.radoslaw@gmail.com, radoslaw.rudek@ue.wroc.pl

0305-0548/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2014.01.008

problem was analysed with job processing times defined as $\tilde{p}_j(v)=p_j h(\sum_{l=1}^{v-1}p_{[l]},v)$, where $h:[0,M)\times\mathbb{N}\to[1,\infty)$ is a differentiable non-decreasing function with respect to each variable (deteriorating), $(\partial/\partial x)h(x,y_0)$ is non-decreasing with respect to x for every fixed y_0. Moreover, it was assumed that $(\partial/\partial x)h(B+x,y_0)\geq h(B+x,y)/x$ for each x and y in domain and $0\leq B\leq M$. It was proved that a special case of this problem can be solved by the Weighted Shortest Processing Times (WSPT) rule. Another general model was presented in [21], i.e., $\tilde{p}_j(v)=p_j h'(\sum_{l=1}^{v-1}\beta_l p_{[l]},v)$, where $\beta_1,\ldots,\beta_n$ is a sequence of numbers such that $0\leq\beta_1\leq\beta_2\leq\cdots\leq\beta_n$ and $h':[0,\infty)\times[1,\infty]\to(0,\infty)$ is a differentiable non-decreasing function with respect to the first variable x (deteriorating), non-increasing with respect to the second variable y (learning), whereas $(\partial/\partial x)h'(x,y_0)$ is non-decreasing with respect to x for every fixed y_0 and $h'(0,1)=1$. The WSPT rule was still optimal for some special cases of the analysed problem with this model.

It can be observed that the results presented in the scientific literature concerning the single processor total weighted completion time scheduling problem with the sum-of-processing time based models mostly focus on polynomially solvable special cases or the worst case analysis of heuristic algorithms (e.g., [34]). Nevertheless, in [26], the NP-hardness was proved for the discussed problem with non-increasing step functions describing job processing times (learning) as well as an exact pseudopolynomial dynamic programming algorithm was proposed, which can be applied for the problem with job processing times described by non-increasing (learning) or non-decreasing (deteriorating) functions. Furthermore, fast approximation algorithms were provided for the general version of the problem, where job processing times are described by arbitrary functions (learning or deteriorating) dependent on the sum of the normal job processing times.

Note that the pseudopolynomial time algorithm in [26] can solve the problem with job processing times described by step functions only, therefore in this paper, we will construct the dynamic programming algorithm that optimally solves the more general problem, where job processing times are described by arbitrary *stepwise* functions. Moreover, the approximation algorithms in [26] are efficient, however, they can be still significantly improved. It will be done by their parallelization. Thus, the dynamic programming as well as parallel algorithms constitutes the contribution of this paper.

The remainder of this paper is organized as follows. The considered problem is formulated in Section 2. Next, the proposed exact dynamic programming and parallel algorithms are described in Section 3, whereas their numerical analysis is provided subsequently. Finally, the last section concludes the paper.

2. Problem formulation

In this section, we will define formally the considered scheduling problem with a general model of job processing times, which can model both learning and deteriorating (aging) phenomena.

A single processor and a set $J=\{1,\ldots,n\}$ of n jobs that have to be performed by the processor are given; there are no precedence constraints between jobs. The processor is continuously available and can process at most one job at a time. Once it begins processing a job it continues until this job is finished. Each job j is characterized by its weight parameter, denoted w_j, and the processing time, denoted $\tilde{p}_j(v)$. When the job j is scheduled as the vth in a sequence, its processing time is given as follows:

$$\tilde{p}_j(v)=p_j\cdot f\left(\sum_{l=1}^{v-1}p_{[l]}\right), \tag{1}$$

where p_j is the normal processing time that is the time required to perform the job if the job is processed as the first one (i.e., $p_j\triangleq p_j(1)$) and $f:[0,+\infty)\to(0,+\infty]$ is an arbitrary function common for all jobs, where $f(0)=1$. Function f depends on the sum of the normal processing times of jobs performed before job j, i.e., $\sum_{l=1}^{v-1}p_{[l]}$, where $p_{[l]}$ denotes the normal processing time of a job scheduled in the lth position in a sequence.

We focus on the following stepwise function characterizing the variability of job processing times:

$$f(x)=\begin{cases}\alpha_0, & x<g_1\\ \alpha_1, & g_1\leq x<g_2\\ \vdots & \\ \alpha_m, & g_m\leq x\end{cases}, \tag{2}$$

where m is the number of steps, α_i are the arbitrary rational numbers (where $\alpha_0=1$), and $g_1<g_2<\cdots<g_m<\sum_{j=1}^{n}p_j$ are the thresholds that describe function f. If $m=0$ or $\alpha_0\ldots=\alpha_m$, then job processing times are constant.

If $\alpha_0=1>\alpha_1>\cdots>\alpha_m>0$, then $f:[0,+\infty)\to(0,1]$ is a non-increasing function that models the learning effect, i.e., the processor (e.g., an autonomous agent, an algorithm, a human, an intelligent system or a learning system in general) can increase its efficiency during processing jobs (e.g., tasks, packets, products) and as a result job processing times decrease. The non-increasing function $\tilde{p}_j(v)$ is called the *learning curve*. Observe that $m=0$ means there is no learning and the job processing times are constant. Furthermore, the values $(1-\alpha_i)$ can be perceived as the learning ratios of the processor (the higher the values the better the learning performance of the processor). The stepwise learning curve characterizes *inter alia* computer systems working on the basis of machine learning algorithms (see [36]). It can also be used as an approximation of other learning curves in manufacturing [14] or computer systems (see [15], where a problem was described by the sum-of-processing-time based learning model). Obviously, learning is noticeable in long (e.g., [16]) as well as in short periods (e.g., [2,6,16,36]). Therefore, it is clearly justified to take it into consideration during determination of short term schedules as well as long horizon planning.

On the other hand, if $\alpha_0=1<\alpha_1<\cdots<\alpha_m<0$, then $f:[0,+\infty)\to[1,+\infty]$ is a non-decreasing function that models the deteriorating (aging) effect, i.e., the efficiency of the processor (e.g., a single worker or a group of human workers, CNC machine, tool, chemical cleaning bath) can decrease during processing jobs and as a result job processing times increase. The non-decreasing function $\tilde{p}_j(v)$ is called the *deteriorating (aging) curve*. An example of a long term aging is deterioration of tools, whereas tiredness of a human worker during particular shifts is a short period. For more details concerning scheduling problems with this phenomenon see [40] or [41]. Obviously, for a given instance the function f is either non-increasing (learning) or non-decreasing (aging/deteriorating), but it cannot be both.

Let $\pi=\langle\pi(1),\ldots,\pi(i),\ldots,\pi(n)\rangle$ denote the sequence of jobs (permutation of the elements of the set J), where $\pi(i)$ is the job processed in position i in this sequence. Let Π denote the set of all such permutations. For the given sequence (permutation) $\pi\in\Pi$, we can easily determine the completion time $C_{\pi(i)}$ of a job placed in the ith position in π from the following formula:

$$C_{\pi(i)}=C_{\pi(i-1)}+\tilde{p}_{\pi(i)}(i)=\sum_{l=1}^{i}\tilde{p}_{\pi(l)}(l), \tag{3}$$

where $C_{\pi(0)}=0$ and according to (1), we have $\tilde{p}_{\pi(i)}(i)=p_{\pi(i)}\cdot f(\sum_{l=1}^{i-1}p_{\pi(l)})$.

The objective is to find such control decisions, i.e., sequence (schedule) π of jobs on the single processor, which minimizes the