Applied Mathematics and Computation 221 (2013) 819–832

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

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

The computational complexity analysis of the two-processor flowshop problems with position dependent job processing times

Radosław Rudek *

Wrocław University of Economics, Komandorska 118/120, 53-345 Wrocław, Poland
Faculty of Computer Science and Management, Wrocław University of Technology, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland

ARTICLE INFO

Keywords:
Scheduling
Learning
Aging
Deteriorating
Flowshop
Computational complexity

ABSTRACT

In this paper, we analyse the makespan minimization problem in the two-processor flowshop environment, where the processing time of each job depends on its position in a sequence (a position dependent job processing time) that is equivalent to the number of previously processed jobs. Namely, if processing times of jobs increase with the number of processed jobs, then the aging (fatigue/deterioration) effect is modelled, whereas the non-increasing dependency describes the learning effect. We prove that the considered problem becomes strongly NP-hard if the processing time of each job is described by a piecewise linear function dependent on its position in a sequence; we analyse two types of functions: non-decreasing (aging) and non-increasing (learning). Furthermore, we describe the strong NP-hardness proof supporting method and elucidate correctness of our other strong NP-hardness proofs. Additionally, we provide a modification of the Johnson's algorithm and prove that it optimally solves the considered problem in the two-processor flowshop environment if job processing times are characterized by a common linear (non-increasing or non-decreasing) dependency on a job position.

1. Introduction

Flowshop scheduling problems constitute a significant part of scheduling theory, since they describe manufacturing settings, where a final product has to flow through all production stages to be completed (see [1,2]). However, the accuracy of solving real-life problems strongly depends on a reliability of mathematical models used during designing process of solution algorithms. Therefore, flowshop scheduling problems that assume constant job processing times are not sufficient for modelling problems that occur in the real-life systems ([3]) *inter alia* where the processing times of jobs depend on the number of earlier processed jobs.

In many industrial and computer systems, the job processing times decrease with the number of processed jobs due to learning, e.g., the total hours to assemble a product can decrease as the number of assembled products increases due to the increasing experience of workers. This phenomenon is called the learning effect and for the first time it was discovered by Wright [4] in the aircraft industry. The first significant venture that took into consideration the existence of this effect was the production planing of airplanes for the World War II needs by USA War Production Board (see [5]). Further study on the learning effect revealed its relevant presence in various industrial manufacturing, economy, management, services sectors, computer networks and systems (e.g., [6–16]).

* Address: Wrocław University of Economics, Komandorska 118/120, 53-345 Wrocław, Poland.
E-mail address: rudek.radoslaw@gmail.com

0096-3003/$ - see front matter
http://dx.doi.org/10.1016/j.amc.2013.06.086

Although the theory of the learning effect enables for efficient estimation of the variable production time caused by learning, it is not possible to optimize time objectives beyond reductions resulting from learning-by-doing (see [17]). However, it has been observed in [18] that optimization of time objectives in learning systems can be controlled by a schedule of jobs processed by such systems. Although the proposed approach does not influence learning itself (that anyway in many cases is impossible), but it allows to efficiently utilize learning abilities of the system. Thereby, time objectives (e.g., the maximum completion time of products or the number of late orders) can be additionally improved (in a specified range) by the sequence (schedule) of jobs. In other words, additional benefits from learning can be gained. It is worth highlighting this scheduling approach does not interfere a system nor require any changes of its structure. Since learning is noticeable in long (e.g., [9]) as well as in short periods (e.g., [14,15]), therefore, it can be taken into consideration during determining short term schedules as well as in long horizon planning. Thus, it is a significant advantage, which makes this non-invasive method universal and applicable to improve (optimize) different (manufacturing or computer) systems, where learning is present and a sequence of processed jobs can be (at least partially) controlled. Therefore, this direction of research has attracted particular attention in scheduling theory since the results can be applied in learning systems (based on human or artificial intelligence) to improve their performances, e.g., [17,19–23].

On the other hand, in industrial environments job processing times can increase with the number of produced items that results in decreasing of the production efficiency. This phenomenon in scheduling theory is called the aging effect (or position dependent deteriorating), e.g., [24–33]. It is observed in many manufacturing companies equipped in CNC lathe machines, where knives or drills blunt with the number of machined elements (e.g., [34]). Similar dependency occurs in chemical industry, i.e., if more elements are cleaned in a chemical bath, then more active substance in the bath is used and the cleaning time is increased (e.g. [35]). Furthermore, in various assembly lines operated by human workers their tiredness grows if more control or assembling activities they perform (e.g., [36,37]).

The common feature of scheduling problems with the aging and learning effects is that both are usually modelled by job processing times described by functions dependent on the number of already processed jobs (see [38–40]). In the scientific literature it is often called position dependent models, since the processing time of a job depends on its position in a sequence. The difference between these two models is that the aging effect is modelled by non-decreasing functions (e.g., [23,30]) whereas the learning effect is modelled by non-increasing dependencies (e.g., [18,38,41–44]).

Having a scheduling problem that models real-life settings, the goal is to find an efficient algorithm that provides an optimal (or satisfying) solution. If (in spite of the effort) such an algorithm cannot be found, then the determination of the NP-hardness of the problem can be attempted for its simplest case that is not polynomially solvable. If the considered scheduling problem reveals to be NP-hard, then there is no optimal polynomial time algorithm for solving it (under $P \neq NP$). Hence only approximation methods can be applied or exact algorithms, but characterized by an above polynomial complexity.

In this paper, we analyse the makespan minimization problem in the two-processor flowshop environment, where processing times of jobs depend on the number of previously processed jobs, that is equivalent to a job position in a sequence, thus, it is called a scheduling flowshop problem with position dependent job processing times. Although there is a significant number of papers that focus on this problem with learning and/or aging effects (e.g., [45–50]), the strong NP-hardness is proved only for general non-increasing (learning) and non-decreasing (aging) position dependent function of job processing times (see [51,52]). Note that efficient solution algorithms that solve even general version of the considered problem with position dependent job processing times have been provided in [52]. However, from the perspective of computational complexity analysis and to decrease the boundary between hard and polynomially solvable cases, it is crucial to prove the NP-hardness for simpler (less complex) problems such as with piecewise linear functions of job processing times.

Therefore, to fill this gap, we prove that the analysed problem becomes strongly NP-hard even if the job processing times are described by piecewise-linear functions dependent on a job position in a sequence. We analyse the computational complexity of the problem individually for two such functions: non-decreasing (aging) and non-increasing (learning); it is the main result of this paper. We also describe the strong NP-hardness proof supporting method and elucidate correctness of our other strong NP-hardness proofs. Additionally, we provide a modification of the Johnson's algorithm ([53]) and prove that it optimally solves the considered problem if job processing times are characterized by a common linear (non-decreasing or non-increasing) dependency on a job position.

The results presented in this paper have – to the best of our knowledge – never been investigated in the scheduling domain.

This paper is organized as follows. Section 2 contains problem formulation and its computational status for aging and learning models is established in Section 3. An optimal polynomial time algorithm for a special case is provided in Section 4. Final remarks are presented in the last section.

2. Problem formulation

There are given a set $J = \{1, \ldots, n\}$ of n jobs and two processors, namely M_1 and M_2. Each job j consists of two operations $O_{1,j}$ and $O_{2,j}$. Operation $O_{z,j}$ has to be processed on processor M_z $(z = 1, 2)$. Moreover operation $O_{2,j}$ may start only if $O_{1,j}$ is completed. If it is assumed that processors have to process jobs in the same order, then the problem is called a permutation flowshop and such problem is considered in this paper. It is also assumed that each processor can process one operation at a time, and there are no precedence constraints between jobs. Operations are non-preemptive and are available for processing at time 0 on M_1. Further, instead of operation $O_{z,j}$, we say job j on processor M_z.