Computers & Industrial Engineering 83 (2015) 237–243

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Minimizing total weighted completion time under makespan constraint for two-agent scheduling with job-dependent aging effects ☆

Jin Young Choi

Department of Industrial Engineering, Ajou University, Suwon, Republic of Korea

ARTICLE INFO

Article history:
Received 27 September 2014
Received in revised form 7 January 2015
Accepted 14 February 2015
Available online 9 March 2015

Keywords:
Total weighted completion time
Two-agent scheduling
Job-dependent aging ratio
Position-based processing time
Branch and bound
Simulated annealing algorithm

ABSTRACT

In this paper, we consider a two-agent single-machine scheduling problem with linear position-based aging effects and job-dependent aging ratios. The objective is to minimize the total weighted completion time of all jobs for two agents, where the makespan for one agent is constrained under an upper bound. After showing that this problem is at least NP-hard, we develop two solution algorithms: First, we devise a branch-and-bound algorithm to find an optimal solution through the establishment of several dominance and feasibility properties, and a lower bound. Second, we propose efficient simulated annealing algorithms, using three different methods to generate an initial solution. Through a numerical experiment, we demonstrate that the suggested algorithms can be applied to efficiently find near-optimal solutions within a reasonable amount of CPU time. In particular, we show that the initial solution method (arranging the jobs for one agent in non-increasing order of aging ratio, and scheduling the jobs for the other in the weighted shortest normal processing time order) is superior to others. Moreover, through scalability testing, we verify its consistent and relatively outstanding performance for larger systems with many processing jobs.

1. Introduction

In recent decades, many research works have been conducted on scheduling problems with two agents sharing a single common resource, where each agent has a set of jobs to process and its own objective function to achieve. This is called a two-agent single-machine scheduling problem in the literature. However, we cannot regard this as a simple bicriteria single-agent optimization problem, because two objective functions might have different measuring units, and all jobs contribute to all criteria. Therefore, the complexity of the two-agent single-machine scheduling problem is higher than that of a bicriteria scheduling problem, and we need more efficient and systematic approaches in order to solve it (Agnetis, Pacciarelli, & Pacifici, 2007).

The concept of the two-agent single-machine scheduling problem was first mooted by Baker and Smith (2003). They introduced a linear combination of the scheduling problem, with two agents and different performance objectives, and analyzed their computational complexity. Following this, Agnetis, Mirchandani, Pacciarelli, and Pacifici (2004) addressed some applications in which multiple agents compete for the usage of common processing resources, while minimizing the makespan, lateness, or (weighted) sum of completion times for different agents. Cheng, Ng, and Yuna (2006) extended the work by Agnetis et al. (2004) to the case of multi-agent scheduling problems. For more recent works, readers can refer to Leung, Pinedo, and Wan (2010), Ding and Sun (2010), Yin, Cheng, Cheng, Wu, and Wu (2013).

As a special case of the two-agent single-machine scheduling problem, we can consider the case that the job processing time depends on the position (or start time) of jobs in a sequence, called position-dependent processing time. In the literature, there are two cases for determining the actual processing time: (i) learning effect and (ii) aging effect. In the case of the learning effect, the actual processing time of a job decreases with its position in the sequence. Biskup (1999) introduced the concept of the learning effect into the scheduling problem for a single agent, and provided an extensive survey on the scheduling problems with learning effects (Biskup, 2008). Under the aging effect, the actual processing time of a job increases with its position in a sequence. Mosheiov (2001) addressed the aging effect, and showed that the optimal schedule for a single-machine flowtime minimization problem has a V-shape (Mosheiov, 2005). Chang, Chen, and Mani (2009) studied the learning/aging effect scheduling problem on a single machine with a common due date. For more recent results, refer to Wu, Yin, and Cheng (2011), Yin, Xu, Cheng, and Wu (2012), and Wang and Wang (2012).

☆ This manuscript was processed by Area Editor T.C. Edwin Cheng.
E-mail address: choijy@ajou.ac.kr

http://dx.doi.org/10.1016/j.cie.2015.02.024
0360-8352/© 2015 Elsevier Ltd. All rights reserved.

In addition, some recent research works address the scheduling problem where two-agent and position (or start time)-dependent processing times are considered simultaneously. Liu, Zhou, and Tang (2010), Lee, Wang, Shiau, and Wu (2010), Liu, Yi, and Zhou (2011), and Wu et al. (2013) investigated linear function-based approaches to compute actual processing times, whereas Cheng, Wu, Cheng, and Wu (2011), Wu, Huang, and Lee (2011), and Li and Hsu (2012) applied exponential function-based approaches. Furthermore, some further approaches assume that the actual processing time depends on the sum of the processing times of the jobs processed previously (Cheng, Cheng, Wu, Hsu, & Wu, 2011; Wu, Cheng, Wu, & Yin, 2012; Liu, Yi, Zhou, & Gong, 2013; Wu, 2014). All abovementioned works studied different combinations of objective functions, such as the total (weighted) tardiness; total (weighted) completion time; number of tardy jobs; or lateness for one agent, with a constraint on the upper bound of the makespan or the tardiness for the other agent.

However, in real-life scheduling environments, it is a more plausible scenario to consider the situation in which each processing job has a different learning or aging ratio, called a job-dependent learning or aging ratio. We notice that this can reflect the different degrees of processing difficulties of the jobs to be processed, meaning that the actual processing time of a job can be determined not only by its processing sequence, but also by the learning/aging ratio of the job. For example, at a major Korean electronics manufacturer, a cell production system to assemble multiple types of electronics could be modeled using job-dependent aging ratio as follows. Each worker has a job sheet with different types of products, and the processing sequence of them should be determined. The worker feels fatigue as operations are conducted, and it makes the processing time of a job lengthy if it is worked later rather than earlier in a sequence, represented by position-dependent aging effect. Furthermore, all jobs have different fatigue degrees depending on the weight and size of the product, making different degrees of aging effects. Each job has different performance objective to be achieved depending on (two) customer requests such as the completion time and lateness. There is another example in a picking system at warehouse. Each picker has a set of picking lists, where each picking list, corresponding to a job, consists of different items having various sizes and weights. Therefore, the processing time of a picking list can be lengthy if it is done later rather than earlier, and the degree of it can be different from jobs to jobs because all picking lists have different fatigue degrees, represented by job-dependent aging effects.

This modeling concept of considering job-dependent learning or aging ratios for the calculation of actual processing time was first suggested by Cheng and Wang (2000), in particular for the case of a single agent. Later, Bachman and Janiak (2004) analyzed a more simple learning effect formulation on the single-agent single-machine case. In the case of an increasing series of dominating machines, Wang and Xia (2005) considered multiple-machine flow-shop scheduling with a single-agent. However, to the best of the authors knowledge, few studies have simultaneously considered two-agent, position-based, and job-dependent learning or aging ratio. In this context, we believe that the application of this modeling concept to the case of the two-agent single-machine scheduling problem with linear position-based job-dependent learning/aging effects warrants some attention.

In this paper, motivated by these observations, we consider the two-agent single-machine scheduling with linear job-dependent position-based aging effects. The objective is to minimize the total weighted completion time of both agents, while satisfying the upper bound condition of the makespan for only one agent. We present two approaches to reaching a solution. First, we design a branch-and-bound algorithm to solve the problem optimally, by developing some feasibility and dominance conditions, and a lower bound. Second, as an efficient heuristic, we suggest a simulated annealing (SA) algorithm with three different variations in the methods of generating an initial solution. The performance of the proposed SAs is verified using a numerical experiment.

The rest of this paper is organized as follows. In Section 2, we define the two-agent single-machine scheduling problem with linear job-dependent position-based aging effects, and design a branch-and-bound algorithm to solve it optimally. In Section 3, we suggest an efficient simulated annealing algorithm, and three different ways of generating an initial solution. Section 4 verifies the performance of these algorithms using a numerical experiment. Finally, Section 5 presents our conclusions along with some directions of future work.

2. Problem definition and a branch-and-bound algorithm

2.1. Problem definition

The considered scheduling problem can be described as follows. There are two agents A and B, with a set of n jobs $J = \{J_1^X, J_2^X, \ldots, J_n^X\}$ to be scheduled, where X represents the agent such that $X \in \{A, B\}$. Some of the jobs, represented by a set $J^A = \{J_1^A, J_2^A, \ldots, J_{n_A}^A\}$, belong to agent A; others, represented by a set $J^B = \{J_1^B, J_2^B, \ldots, J_{n_B}^B\}$, belong to agent B, where $n_A + n_B = n, J^A \cup J^B = J$, and $J^A \cap J^B = \emptyset$. All of these jobs are available to be processed at the beginning of operation, and each of them is processed by a single common machine. Each job J_i^X $(i = 1, 2, \ldots, n)$ is assigned to a weight w_i^X, a job-dependent aging ratio $b_i^X (> 0)$, and a normal processing time p_i^X. However, the actual processing time of job J_i^X, processed at the rth order among n jobs, $p_i^X(r)$, is determined by the processing position r and job-dependent aging ratio b_i^X, represented as

$$p_i^X(r) = p_i^X + rb_i^X, \text{ for } i : J_i^X \in J = (J^A \cup J^B) \text{ and } r = 1, 2, \ldots, n. \quad (1)$$

In this scheduling environment, the objective is to minimize the total weighted completion time of all jobs for both agents, with the restriction that the makespan of agent B cannot exceed an upper bound U. By using the three-field notation $\Psi_1|\Psi_2|\Psi_3$ suggested by Graham, Lawler, Lenstra, and Rinnooy (1979), we can denote this scheduling problem as

$$1\left|p_i^A(r) = p_i^A + rb_i^A, \quad p_j^B(v) = p_j^B + vb_j^B\right|\sum_{i:J_i^X \in J} w_i^X C_i^X : C_{max}^B \leqslant U, \quad (2)$$

where C_i^X and C_{max}^B are the completion time of job J_i^X and the makespan of agent B, respectively. Moreover, the first component is the number of machines, the second component describes the processing times, and the third component represents the objective functions.

Regarding computational complexity, Agnetis et al. (2004) showed that $1\left|\right|\sum_{i=1}^{n_A} w_i^A C_i^A : C_{max}^B$ is binary NP-hard, implying that the complexity of the problem $1\left|\right|\sum_{i:J_i^X \in J} w_i^X C_i^X : C_{max}^B$ is at least binary NP-hard. Since our problem in Eq. (2) considers linear job-dependent position-based aging effects, it is more difficult than the problem $1\left|\right|\sum_{i:J_i^X \in J} w_i^X C_i^X : C_{max}^B$, implying that the problem in Eq. (2) is at least binary NP-hard. Therefore, we need to develop an efficient near-optimal solution approach to solve this problem.

2.2. Branch-and-bound algorithm

In this paper, we first suggest a branch-and-bound algorithm (Hillier & Lieberman, 2010) in order to find an optimal solution.