Contents lists available at ScienceDirect

Computers & Operations Research

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

An order scheduling problem with position-based learning effect

Jianyou Xu[a], Chin-Chia Wu[b], Yunqiang Yin[c], Chuanli Zhao[d], Yi-Tang Chiou[b], Win-Chin Lin[b,*]

[a] College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
[b] Department of Statistics, Feng Chia University, Taichung 40724, Taiwan
[c] Faculty of Science, Kunming University of Science and Technology, Kunming 650093, China
[d] School of Mathematics and Systems Science, Shenyang Normal University, Shenyang, Liaoning 110034, China

ARTICLE INFO

Article history:
Received 29 July 2015
Received in revised form
10 March 2016
Accepted 24 April 2016
Available online 3 May 2016

Keywords:
Order scheduling
Simulated annealing
Particle swarm optimization method
Tardiness
Learning effect

ABSTRACT

The order scheduling problem is receiving increasing attention in the relatively new but creative area of scheduling research. In order scheduling, several orders are processed on multiple machines, and each order comprises multiple components. The order completion time is defined as the time at which all components in an order are completed. In previous studies, the processing times of all components were fixed in order scheduling problems. This is unreasonable because a steady decline in processing time usually occurs when the same task is performed repeatedly in practical situations. Therefore, we propose a multiple-machine order scheduling problem with a learning effect to minimize the total tardiness. We develop a branch-and-bound algorithm incorporating certain dominance rules and three lower bounds for obtaining the optimal solution. Subsequently, we propose simulated annealing, particle swarm optimization, and order-scheduling MDD algorithms for obtaining a near-optimal solution. In addition, the experimental results of all proposed algorithms are provided.

1. Introduction

In many manufacturing and service environments, a product development team independently develops modules for several products, and the product design is deemed completed once all modules have been designed; this situation is termed “customer order scheduling.” Ahmadi et al. [1] investigated one such example in the manufacturing industry: a manufacturer producing semi-finished lenses and competing globally. The lenses are sold to professional lens finishing labs and large optometry stores. This manufacturer produces three types of plastic lenses: CR-39, polycarbonate, and spectralite. CR-39 is the industry standard for plastic lenses. The polycarbonate lenses are fabricated from higher index materials compared with CR-39 and thus are more durable. The spectralite lenses are composed of high index materials but do not require curing as much as polycarbonate lenses do. Because of the processing requirements, these lenses are produced on dedicated production lines according to confirmed customer orders; each order comprises different quantities of the three lens types. After completion on the different production lines, the lenses are sent to the packaging area separately and are shipped to the customer as a single shipment.

A common assumption in conventional scheduling models is that job processing times are fixed, known integer numbers [25,29]. This is untrue in applications where production time can be shortened if the production can be deferred, because the efficiency of the production facility, particularly for high-technology manufacturing processes. Moreover, Biskup [3] claimed that learning primarily occurs because of repeated processing time-independent operations, such as assembling, controlling, and operating machines and processing data.

Furthermore Biskup [3], claimed that the learning effect in scheduling may occur in a company where similar jobs are produced on one machine or on parallel and identical machines for several customers. Jobs usually have varying normal processing times because of varying order quantities and slightly differing product components. On the basis of this observation, Biskup [3] introduced a position-based learning model in which a job is processed in its normal processing time if the job is scheduled first; the processing times of the following jobs are shorter than their normal processing times because of the learning effect. However, this learning factor is not considered in customer order scheduling. Therefore, in this study, we investigated a multiple-machine order scheduling problem with a job position-based learning function to minimize the total tardiness of all given jobs.

Many studies have investigated order scheduling problems to minimize the total weighted completion time. Sung and Yoon [30] demonstrated that this problem is nondeterministic polynomial-time

* Corresponding author.
E-mail address: linwc@fcu.edu.tw (W.-C. Lin).

http://dx.doi.org/10.1016/j.cor.2016.04.021
0305-0548/© 2016 Elsevier Ltd. All rights reserved.

(NP) hard. Wang and Cheng [36] and Leung et al. [16,19,21] have proposed heuristic algorithms for obtaining near-optimal solutions, and Yoon and Sung [52] developed a branch-and-bound algorithm for obtaining the optimal solution. Ahmadi et al. [1] too showed that the order scheduling problem to minimize the total completion time is NP-hard and developed heuristic algorithms for it. Considering the due dates as criterion, Leung et al. [17] addressed the order scheduling problem to minimize the maximum lateness (i.e., the number of tardy orders) and established heuristic algorithms. Lee [15] considered the order scheduling problem to minimize total tardiness. Additional information on order scheduling problems on identical machines in parallel settings can be obtained from [46,18,20–22].

Biskup [3] and Cheng and Wang [5] were pioneers who introduced learning to scheduling. Koulamas and Kyparisis [13] and Kuo and Yang [12] modified Biskup's [3] learning model for increased practicability; the learning occurs not because of repeated processing time-independent operations, such as setups, but because of repeated production activity. The learning effect has since continued to receive much attention. For some relevant studies, we refer the readers to [4,6,8,14,23,24,31–35,37,39–45,49,51]. In addition, relevant studies on time-dependent processing are available in [47,50], for two special issues, [48], for a deterioration model in which the actual processing time of a job depends not only on the starting time of the job but also on its scheduled position, and Wang and Wang [35,37] and Wang and Zhang [38], for flowshop scheduling with a learning effect.

Only Lee [15] considered total tardiness as the objective measure for order scheduling, but without considering the learning effect. Therefore, this study focused on minimizing the total tardiness of the order scheduling problem with a learning effect. The remainder of this paper is organized as follows. The notation and problem formulation are presented in Section 2. In Section 3, several dominant lemmas and three lower bounds are derived and used in the branch-and-bound algorithm to accelerate the search efficiency for obtaining the optimal solution. Section 4 describes the SA, particle swarm optimization (PSO), and order-scheduling MDD heuristics. Section 5 describes the details of the branch-and-bound algorithm. Section 6 reports the simulation results of the proposed methods. Conclusions are presented in Section 7.

2. Problem statement

First, we define the notations used in the paper.

n: number of orders;

m: number of machines;

M_i: machine i, $i=1, 2,\ldots, m$;

S,S': schedules of n orders;

π,π': partial sequences of n orders;

p_{ik}: processing time for order i on machine k, $k=1, 2, \ldots,m$;

d_i: due date for order i, $i=1, 2, \ldots, n$;

a: learning effect, where $a\le 0$;

$C_i(S)$ and $C_j(S)$: completion times of orders i and j in S, and $C_j(S')$ and $C_i(S')$: completion times of orders j and i in S';

[ ]: position of jobs in a sequence;

$T_i(S)$ and $T_j(S)$: tardiness of orders i and j in S;

$T_j(S')$ and $T_i(S')$: tardiness of orders j and i in S', where

$T_i(S)=\max\left\{0,C_i(S)-d_i\right\}$.

The problem is formally formulated as follows. Consider n orders from n different clients, and each order comprises m components. Consider a facility with m different machines arranged in parallel. Each machine can produce one particular component, that is, each component can be processed on one dedicated machine. All orders are ready at time zero. Assume that no assembling times are involved and that the orders are processed without interruptions or preemptions. Let p_{ik} denote the processing time of the kth component of order i to be processed on machine k, and d_i denote the due date of order i. Because of the learning effect, we assume the actual processing time of the kth component of order i to be processed on machine k in the rth position in a given sequence, that is, $p_{ik[r]}=p_{ik}r^a$, where a is a learning effect with $a<0$, and $[r]$ denotes job i to be scheduled in the rth position in a given schedule. This study aimed to obtain an optimal sequence to minimize the total tardiness of all given n orders. The problem without learning effect was shown to be NP-hard in the strong sense by [15], and therefore our problem is NP-hard in the strong sense. A branch-and-bound algorithm incorporating certain dominance rules and three lower bounds for the optimal solution is developed in the following sections.

3. Dominances and lower bounds

Let $S=(\pi, i, j, \pi')$ and $S'=(\pi, j, i, \pi')$ denote two schedules in which π and π' are partial sequences. To show that S dominates S', it suffices to show that $T_i(S)+T_j(S)\le T_j(S')+T_i(S')$ and $C_j(S)\le C_i(S')$. Let $r-1$ be the number of orders in π, t_k be the completion time of the last order in partial schedule π on machine k, $k=1,2, \ldots,m$, u (resp., x) denote the machine index which finishes lastly when the order i (or j) in S, and v (resp., y) denote the machine index which finishes lastly when the order j (or i) in S'. It follows that

$C_i(S)=t_u+p_{iu}r^a=\max_{1\le k\le m}\left\{t_k+p_{ik}r^a\right\}$,

$C_j(S)=t_x+p_{ix}r^a+p_{jx}(r+1)^a=\max_{1\le k\le m}\{t_k+p_{ik}r^a+p_{jk}(r+1)^a\}$,

$C_j(S')=t_v+p_{jv}r^a=\max_{1\le k\le m}\{t_k+p_{jk}r^a\}$, and

$C_i(S')=t_y+p_{jy}r^a+p_{iy}(r+1)^a=\max_{1\le k\le m}\left\{t_k+p_{jk}r^a+p_{ik}(r+1)^a\right\}$.

Proposition 1. *If*

$\forall\ 1\le k\le m$, $p_{jk}(1-(1+1/r)^a)<p_{ik}\le p_{jk}$, $d_i-d_j\le(p_{jk}-p_{ik})(r^a-(r+1)^a)$, *then S dominates S'.*

Proof. It follows from $p_{ik}\le p_{jk}$ that $t_k+p_{ik}r^a\le t_k+p_{jk}r^a$ and

$t_k+p_{ik}r^a\le t_k+p_{ik}r^a+p_{jk}(r+1)^a\le t_k+p_{jk}r^a+p_{ik}(r+1)^a,\quad \forall\ 1\le k\le m$.

Thus,

$C_i(S)=\max_{1\le k\le m}\left\{t_k+p_{ik}r^a\right\}\le\max_{1\le k\le m}\{t_k+p_{jk}r^a\}=C_j(S')$

$C_i(S)=\max_{1\le k\le m}\left\{t_k+p_{ik}r^a\right\}\le\max_{1\le k\le m}\{t_k+p_{ik}r^a+p_{jk}(r+1)^a\}=C_j(S)$

$\le\max_{1\le k\le m}\{t_k+p_{jk}r^a+p_{ik}(r+1)^a\}=C_i(S')$

Now, we split the proof into two parts.

Case 1; $d_i\le d_j$. In this case, it follows from of [7] that $T_i(S)+T_j(S)\le T_j(S')+T_i(S')$.

Case 2; $d_i>d_j$. In this case, on one hand, we have

$(C_j(S')-d_j)-(C_i(S)-d_i)=(C_j(S')-C_i(S))+(d_i-d_j)>0$,implying that $T_i(S)\le T_j(S')$. On one hand, it follows from

$d_i-d_j\le(p_{jk}-p_{ik})(r^a-(r+1)^a)$that , $\forall\ 1\le k\le m$, and thus

$C_j(S)-d_j=\max_{1\le k\le m}\{t_k+p_{ik}r^a+p_{jk}(r+1)^a-d_j\}$

$\le\max_{1\le k\le m}\{t_k+p_{jk}r^a+p_{ik}(r+1)^a-d_i\}=C_i(S')-d_i$,

implying that $T_j(S)\le T_i(S')$.

Summing up the above analysis, in both cases, we have $T_i(S)+T_j(S)\le T_j(S')+T_i(S')$, as required.

As a direct consequence of Proposition 1, one can obtain the following result.

Proposition 2. *If $\forall\ 1\le k\le m, p_{ik}\le p_{jk}, d_i\le d_j$, then S dominates S'.*

Proposition 3. *If $\forall\ 1\le k\le m$, $p_{ik}\le p_{jk}(1-(1+1/r)^a)$, then S dominates S'.*

Proof On one hand, $p_{ik}\le p_{jk}(1-(1+1/r)^a)$ implies $p_{ik}<p_{jk}$, $\forall\ 1\le k\le m$, and thus, by Proposition 1, $C_i(S)<C_j(S)\le C_i(S')$. On the