Computers & Operations Research 58 (2015) 24–31

Contents lists available at ScienceDirect

Computers & Operations Research

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

Permutation flowshop problems with bi-criterion makespan and total completion time objective and position-weighted learning effects

Jian-Jun Wang *, Bing-Hang Zhang

Faculty of Management and Economics, Dalian University of Technology, Dalian 116024, China

ARTICLE INFO

Available online 31 December 2014

Keywords:
Scheduling
Flowshop
Learning effect
Heuristic algorithm
Branch-and-bound algorithm

ABSTRACT

In this paper, we study the problem of minimizing the weighted sum of makespan and total completion time in a permutation flowshop where the processing times are supposed to vary according to learning effects. The processing time of a job is a function of the sum of the logarithms of the processing times of the jobs already processed and its position in the sequence. We present heuristic algorithms, which are modified from the optimal schedules for the corresponding single machine scheduling problem and analyze their worst-case error bound. We also adopt an existing algorithm as well as a branch-and-bound algorithm for the general m-machine permutation flowshop problem. For evaluation of the performance of the algorithms, computational experiments are performed on randomly generated test problems.

1. Introduction

Scheduling problems have received considerable attention for many years (see Gonzalez and Sahni [8], Pinedo [17], Bai and Tang [1]). Most research assumes that the production time of a given product is independent of its position in the production sequence. However, in many realistic settings, because firms and employees perform a task over and over, they learn how to perform more efficiently (i.e., performing setups, operating hardware and software, and handling raw materials and components). The production facility (a machine, a worker) improves continuously over time. As a result, the production time of a given product is shorter if it is scheduled later, rather than earlier in the sequence. This phenomenon is known as a "learning effect" in the literature. Extensive surveys of different scheduling models and problems involving jobs with learning effects can be found in Biskup [2] and Janiak and Rudek [9]. Recently, Cheng et al. [3], Cheng et al. [4], Janiak and Rudek [10], Janiak et al. [11], Niu et al. [15], Wang and Wang [26], Wu et al. [29], Wu et al. [31], Yin et al. [34], Yin et al. [35], and Yin et al. [36] considered single machine scheduling with learning effects. Yeh et al. [33] considered parallel-machine scheduling with fuzzy processing times and learning effects. Cheng et al. [3], Cheng et al. [5], Kuo et al. [12], Lee and Chung [13], Liu and Feng [14], Rudek [19], Rudek and Rudek [20], Sun et al. [21,22], Vahedi-Nouri et al. [23], Vahedi-Nouri et al. [24], Wang and Wang [25], Wang and Wang [27], Wang et al. [28], Wu and Lee [30], and Xu et al. [32] considered flowshop scheduling with learning effects.

In this paper we consider the same model as that of Cheng et al. [3], but with flowshop scheduling problem. The objective is to minimize the sum of makespan and total completion time. We present some heuristic algorithms and a branch-and-bound algorithm to solve this problem. The remaining part of this paper is organized as follows. Section 2 gives some general notations and assumptions. Section 3 presents heuristic algorithms for the problem. Some lower bounds are given in Section 4. Section 5 adopts a well-known algorithm for flowtime minimization and a branch-and-bound algorithm to the problem under consideration. Section 6 gives computational experiments of the branch-and-bound algorithm and the heuristic algorithms. The last section summarizes our findings and gives an outlook at future research questions.

2. Notations and assumptions

The permutation flowshop scheduling consists of scheduling n jobs $J=\{J_1,J_2,\ldots,J_n\}$ on m machines $M_1,M_2,\ldots,M_m$. Each job J_j consists of a chain operations $(O_{1j},O_{2j},\ldots,O_{mj})$, and must be processed without preemption on each machine with permutation schedule, i.e., all machines process the jobs in the same order. The (normal) processing time of operation O_{ij} is denoted by p_{ij}. The actual processing time p_{ijr} of job J_j on machine M_i is a function dependent on its position r in a schedule. As in Cheng et al. [3], in this paper, we consider flow shop scheduling with sum-of-logarithm-processing-times-based and job-position-based learning

* Corresponding author.
E-mail addresses: wmeagle717@163.com, wangjianjun_2008@163.com (J.-J. Wang).

http://dx.doi.org/10.1016/j.cor.2014.12.006
0305-0548/

effects in which the actual processing time of job job J_j on machine M_i is

$$p_{ijr}=p_{ij}\left(1+\sum_{l=1}^{r-1}\beta_l \ln p_{i[l]}\right)^a r^b, \quad i=1,2,\ldots,m,\ \ r,j=1,2,\ldots,n, \quad (1)$$

when scheduled in position r, where $a\le 0$ and $b\le 0$ are the learning indices, $\sum_{l=1}^{0}\beta_l \ln p_{i[l]}=0$, $\beta_l>0$ is the weight associated with the lth position, and $0<\beta_1\le\beta_2\le\ldots\le\beta_n$. "The real-world relevance of the formula (1) can be found in human learning. In the early stage of processing a given set of jobs, the worker is not familiar with the operations, so the learning effect on the jobs scheduled early is not apparent. On the other hand, when the worker has spent more time processing jobs, his learning improves. So the worker's learning effect on a job depends not only on the total processing time of the jobs that he has processed but also on the job's position" (Cheng et al. [3]). For a given schedule S, let $C_{ij}(S)=C_{ij}$ be the completion time of operation $O_{ij}, i=1,2,\ldots,m; j=1,2,\ldots,n$, $C_j(S)=C_j=C_{mj}$ represent the completion time of job J_j. The goal is to gain a schedule to minimize $\alpha C_{\max}+(1-\alpha)\sum C_j$, where $C_{\max}=\max\{C_j|j=1,2,\ldots,n\}, 0\le\alpha\le 1$. As in Cheng et al. [3], we denote the general learning effect (1) as LE_{pos}, the problem under study can be described as $Fm|prmu, LE_{pos}|\alpha C_{\max}+(1-\alpha)\sum C_j$.

Remark. On one hand, the weighting here allows to consider makespan minimization ($\alpha=1$) and total flowtime minimization ($\alpha=0$) as special cases. On the other hand, since total flowtime is (significantly) larger than makespan, one has to be careful when choosing alpha, especially if weighting is used as a means for multi-criteria consideration in this problems setting.

3. Worst-case behavior of heuristic algorithms

Lemma 1 (*Cheng et al. [3]*). *For the problems* $1|LE_{pos}|\sum C_j$ *and* $1|LE_{pos}|C_{\max}$, *an optimal schedule can be obtained by sequencing the jobs in non-decreasing order of* p_j *(i.e., the SPT rule).*

It is well known that the problem $Fm|prmu, LE_{pos}|\alpha C_{\max}+(1-\alpha)\sum C_j$ ($m\ge 3$) is NP-complete. In order to solve this problem approximately and examine "worst" schedules, we restrict ourselves to busy schedules (see Gonzalez and Sahni [8]), that is a schedule in which at all times from start to finish at least one machine is processing an operation.

From Lemma 1, we can use the SPT (in order of non-decreasing $L_j=\sum_{i=1}^{m}p_{ij}$) rule as an approximate algorithm to solve $Fm|prmu, LE_{pos}|\alpha C_{\max}+(1-\alpha)\sum C_j$.

Theorem 1. *Let* S^* *be an optimal schedule and* S *be an SPT schedule for the* $Fm|prmu, LE_{pos}|\alpha C_{\max}+(1-\alpha)\sum C_j$ *problem. Then*

$$\rho_1=\frac{\alpha C_{\max}(S)+(1-\alpha)\sum C_j(S)}{\alpha C_{\max}(S^*)+(1-\alpha)\sum C_j(S^*)}\le\frac{m}{(1+\ln P_{\max})^a n^b},$$

where $\ln P_{\max}=\max\{\sum_{l=1}^{n}\beta_l \ln p_{il}-\beta_1 \ln p_{i\ \min}|i=1,2,\ldots,m\}$ *and* $p_{i\ \min}=\min\{p_{ij}|j=1,2,\ldots,n\}$.

Proof. Without loss of generality we assume that S be an SPT schedule, i.e., $L_1\le L_2\le\cdots\le L_n$. Obviously, for the completion time, we have

$$\begin{aligned}C_j(S)&\le L_1+L_2(1+\beta_1 \ln p_{\min,1})^a 2^b\\&\quad+L_3(1+\beta_1 \ln p_{\min,1}+\beta_2 \ln p_{\min,2})^a 3^b\\&\quad+\cdots+L_j(1+\beta_1 \ln p_{\min,1}+\beta_2 \ln p_{\min,2}\\&\quad+\cdots+\beta_{j-1} \ln p_{\min,j-1})^a j^b\\&\le\sum_{l=1}^{j}L_l, \end{aligned}\quad (2)$$

where $p_{\min,j}=\min\{p_{ij}|i=1,2,\ldots,m\}$, so

$$C_{\max}(S)\le\sum_{j=1}^{n}L_j, \quad (3)$$

and

$$\sum_{j=1}^{n}C_j(S)\le\sum_{j=1}^{n}\sum_{l=1}^{j}L_l. \quad (4)$$

Let $S^*=(J_{[1]},J_{[2]},\ldots,J_{[n]})$ be the optimal schedule, where $[j]$ denotes the job that occupies the jth position in S^*, we have

$$\begin{aligned}C_{1[j]}&=p_{1[1]}+p_{1[2]}(1+\beta_1 \ln p_{1[1]})^a 2^b\\&\quad+\cdots+p_{1[j]}(1+\beta_1 \ln p_{1[1]}+\beta_2 \ln p_{1[2]}+\cdots+\beta_{j-1}\ln p_{1[j-1]})^a j^b\\C_{2[j]}&\ge p_{2[1]}+p_{2[2]}(1+\beta_1 \ln p_{2[1]})^a 2^b\\&\quad+\cdots+p_{2[j]}(1+\beta_1 \ln p_{2[1]}+\beta_2 \ln p_{2[2]}+\cdots+\beta_{j-1}\ln p_{2[j-1]})^a j^b\\C_{m[j]}&\ge p_{m[1]}+p_{m[2]}(1+\beta_1 \ln p_{m[1]})^a 2^b\\&\quad+\cdots+p_{m[j]}(1+\beta_1 \ln p_{m[1]}+\beta_2 \ln p_{m[2]}+\cdots+\beta_{j-1}\ln p_{m[j-1]})^a j^b,\end{aligned}$$

hence

$$C_{[j]}(S^*)\ge\frac{1}{m}\sum_{l=1}^{j}L_{[l]}(1+\ln P_{\max})^a j^b, \quad (5)$$

where $\ln P_{\max}=\max\{\sum_{l=1}^{n}\beta_l \ln p_{il}-\beta_1 \ln p_{i\ \min}|i=1,2,\ldots,m\}$ and $p_{i\ \min}=\min\{p_{ij}|j=1,2,\ldots,n\}$, so

$$C_{\max}(S^*)\ge\frac{1}{m}(1+\ln P_{\max})^a n^b\sum_{j=1}^{n}L_{[j]}=\frac{1}{m}(1+\ln P_{\max})^a n^b\sum_{j=1}^{n}L_j, \quad (6)$$

and

$$\sum_{j=1}^{n}C_j(S^*)\ge\frac{1}{m}(1+\ln P_{\max})^a n^b\sum_{j=1}^{n}\sum_{j=1}^{i}L_{[j]}\ge\frac{1}{m}(1+\ln P_{\max})^a n^b\sum_{i=1}^{n}\sum_{j=1}^{i}L_j. \quad (7)$$

Consequently, from (3), (4), (6) and (7), we have

$$\begin{aligned}\rho_1&=\frac{\alpha C_{\max}(S)+(1-\alpha)\sum C_j(S)}{\alpha C_{\max}(S^*)+(1-\alpha)\sum C_j(S^*)}\\&\le\frac{\alpha\sum_{j=1}^{n}L_j+(1-\alpha)\sum_{j=1}^{n}\sum_{l=1}^{j}L_l}{\frac{1}{m}(1+\ln P_{\max})^a n^b(\alpha\sum_{j=1}^{n}L_j+(1-\alpha)\sum_{i=1}^{n}\sum_{j=1}^{i}L_j)}\\&=\frac{m}{(1+\ln P_{\max})^a n^b}.\end{aligned}$$ □

Gonzalez and Sahni [8] proposed the ARB rule (any busy schedule) as an approximate algorithm to solve $Fm|prmu|\sum C_j$, hence, we can also use the ARB rule as an approximate algorithm to solve $Fm|prmu, LE_{pos}|\alpha C_{\max}+(1-\alpha)\sum C_j$.

Theorem 2. *Let* S^* *be an optimal schedule and* S *be any busy schedule (the ARB rule) for the problem* $Fm|prmu, LE_{pos}|\alpha C_{\max}+(1-\alpha)\sum C_j$. *Then*

$$\rho_2=\frac{\alpha C_{\max}(S)+(1-\alpha)\sum C_j(S)}{\alpha C_{\max}(S^*)+(1-\alpha)\sum C_j(S^*)}\le\frac{n}{(1+\ln P_{\max})^a n^b},$$

where $\ln P_{\max}=\max\{\sum_{l=1}^{n}\beta_l \ln p_{il}-\beta_1 \ln p_{i\min}|i=1,2,\ldots,m\}$ *and* $p_{i\min}=\min\{p_{ij}|j=1,2,\ldots,n\}$.