Computers & Industrial Engineering 88 (2015) 350–353

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Job selection in two-stage shops with ordered machines

Christos Koulamas [a,*], S.S. Panwalkar [b]

[a] College of Business, Florida International University, Miami, FL 33199, United States
[b] Carey Business School, Johns Hopkins University, 100 International Drive, Baltimore, MD 21202, United States

ARTICLE INFO

Article history:
Received 10 March 2015
Received in revised form 7 June 2015
Accepted 27 July 2015
Available online 1 August 2015

Keywords:
Scheduling
Flow shop
Open shop
Job shop

ABSTRACT

We consider job selection problems in two-stage flow shops, open shops and job shops. The objective is to select the best job subset with a given cardinality to minimize either the total job completion time or the maximum job completion time (makespan). An $O(n^2)$ algorithm is available for the two-stage flow shop with ordered machines and the minimum makespan objective; we utilize this algorithm to solve the same problem for the total job completion time objective. We then propose an $O(n^2)$ algorithm for the two-machine open shop job selection problem with ordered machines and the minimum makespan objective. We also consider a two-machine job shop in which the first operation of each job is no longer than the second one. We show that the job selection problem in this job shop with the minimum makespan objective is ordinary NP-hard and that the problem becomes solvable in $O(n \log n)$ time with the additional assumption of ordered jobs.

1. Introduction

Job selection problems in two-stage shops are concerned with the selection of the best job subset with a given cardinality to minimize a scheduling objective such as the total job completion time or the maximum job completion time (makespan) among others. Specifically, there is a set of n jobs, $j = 1, \ldots, n$ all of them available at time zero; job j must be processed non-preemptively on two stages (machines) M_1, M_2 with known positive processing times a_j, b_j on M_1, M_2 respectively. Each machine can process at most one job at a time and the two operations of each job cannot be processed concurrently. In a flow shop, all jobs must follow the $M_1 \rightarrow M_2$ processing sequence. In a job shop, a subset of the jobs must follow the $M_1 \rightarrow M_2$ processing sequence while the remaining jobs must follow the $M_2 \rightarrow M_1$ processing sequence. Each job is allowed to have only one operation on each machine. Therefore our two-stage job shop is actually a bi-directional two-stage flow shop. Finally, in an open shop, the two operations of each job can be processed in any order.

The objective is to select the best job subset with a given cardinality k for each $k = 1, \ldots, n$ to minimize either the total job completion time $\sum C_j$ or the makespan $C_{\max}$. It is assumed that all jobs are equally important. If this is not the case, then a job-specific rejection cost should be assigned to each non-processed job and the job selection problem becomes the corresponding scheduling problem with job rejection. The literature on shop scheduling problems with job rejection is reviewed by Shabtay, Gaspar, and Kaspi (2013). It is mentioned there that most two-stage shop problems with job rejection are NP-hard even with equal job rejection costs (see Choi & Chung, 2011; Shabtay & Gaspar, 2012 & T'kindt & Della Croce, 2012 among others). Furthermore, most of the papers on two-stage shops with job rejection utilize composite objective functions in which the total job rejection cost is added to the cost of the scheduling objective for the non-rejected jobs. In contrast, we consider all $k = 1, \ldots, n$ optimal solutions in this paper; this is facilitated by the observation that all jobs are viewed as equally important in the job selection problem and therefore the number of optimal solutions is equal to n (or equal to $n + 1$ if the solutions with either $\sum C_j = 0$ or $C_{\max} = 0$ are also considered).

We conclude our introduction by reviewing some related literature. T'kindt, Della Croce, and Bouquard (2007) analyzed the two-machine flow shop $F2/d_i = d/(d, n_T)$ problem of simultaneously determining a minimal common job due date d and the minimum number of tardy jobs n_T. For any given number of tardy jobs n_T (or for any given number of $k = n - n_T$ non-tardy jobs), the $F2/d_i = d/(d, n_T)$ problem is equivalent to the $F2/k\ jobs/C_{\max}$ problem in which the objective is to select the best job subset among all subsets with cardinality k to minimize the makespan. Della Croce, Koulamas, and T'kindt (2014) utilized a constraint generation approach to solve the $F2/k\ jobs/C_{\max}$ problem.

The minimum number of tardy jobs $F2/d_i = d/n_T$ problem in which the common due date d is given is reducible to the

* Corresponding author.
E-mail addresses: koulamas@fiu.edu (C. Koulamas), panwalkar@jhu.edu (S.S. Panwalkar).

http://dx.doi.org/10.1016/j.cie.2015.07.020
0360-8352/

$F2/d_i = d/(d, n_T)$ problem in polynomial time. T'kindt et al. (2007) showed that the $F2/k\ jobs/C_{\max}$ problem is ordinary NP-hard by utilizing the fact that the $F2/d_i = d/n_T$ problem is ordinary NP-hard.

The complexity of the $F2/k\ jobs/C_{\max}$ problem motivated Panwalkar and Koulamas (2012) to consider the $F2/k\ jobs/C_{\max}$ problem with a maximal machine. If $a_j \geqslant b_j$ $(b_j \geqslant a_j)$ for all $j = 1, \ldots, n$, then the first (second) machine is maximal and the problem has ordered machines. Panwalkar and Koulamas (2012) showed that the $F2/k\ jobs,\ M_2 - \max/C_{\max}$ problem is solvable in $O(n^2)$ time by implementing their PK algorithm; the symbol $M_2 - \max$ indicates that M_2 is maximal. Because of the symmetry of flow shop problems with the makespan objective, the $F2/k\ jobs,\ M_1 - \max/C_{\max}$ problem is also solvable in $O(n^2)$ time. In this paper, we utilize the PK algorithm to solve the $F2/k\ jobs,\ M_1 - \max/\sum C_j$ job selection problem in $O(n^2)$ time.

We then turn our attention to the open shop $O2/k\ jobs,\ M_1 - \max/C_{\max}$ (and its symmetric $O2/k\ jobs,\ M_2 - \max/C_{\max}$ problem) and show that the $O2/k\ jobs,\ M_1 - \max/C_{\max}$ problem can be solved in $O(n)$ time for each $k = 1, \ldots, n$ value.

Finally, we consider the two-stage job shop $J2/k\ jobs,\ o1 < o2/C_{\max}$ problem; the symbol $o1 < o2$ indicates that the first operation of each job is no longer than the second one. In this problem, there are two groups of jobs, G_1 and G_2 respectively. If $j \in G_1$, then the routing of j is $M_1 \rightarrow M_2$ and $a_j \leqslant b_j$. Similarly, if $j \in G_2$, then the routing of j is $M_2 \rightarrow M_1$ and $b_j \leqslant a_j$.

We show that the $J2/k\ jobs,\ o1 < o2/C_{\max}$ problem is ordinary NP-hard. We also show that the $J2/k\ jobs,\ o1 < o2/C_{\max}$ problem becomes solvable in $O(n \log n)$ time if we impose the additional condition of ordered jobs, according to which if $a_i \leqslant a_j$ for any two jobs i,j, then $b_i \leqslant b_j$ as well $\forall\{i,j\} \in \{1, \ldots, n\},\ i \neq j$. For more on the definitions of ordered machines and ordered jobs, the reader is referred to Panwalkar, Smith, and Koulamas (2013).

The rest of the paper is organized as follows. In Section 2, we show that the $F2/k\ jobs,\ M_1 - \max/\sum C_j$ problem can be solved in $O(n^2)$ time. In Section 3, we present an $O(n^2)$ algorithm for the $O2/k\ jobs,\ M_1 - \max/C_{\max}$ problem. A numerical example illustrating the implementation of the proposed algorithms is presented in Appendix A. In Section 4, we show that the $J2/k\ jobs,\ o1 < o2/C_{\max}$ problem is ordinary NP-hard and that the additional assumption of ordered jobs makes the problem solvable in $O(n \log n)$ time. The conclusions of this research are summarized in Section 5.

2. An $O(n^2)$ algorithm for the $F2/k\ jobs, M_1 - \max/\sum C_j$ problem

The $F2/k\ jobs/\sum C_j$ problem with arbitrary job processing times is strongly NP-hard because the $F2//\sum C_j$ problem is strongly NP-hard (Garey, Johnson, & Sethi, 1976). The $F2/M_1 - \max/\sum C_j$ problem (in which $a_j \geqslant b_j$) is solved optimally in $O(n \log n)$ time by arranging the jobs in the shortest processing time (SPT) order of their a_j values (Sarin & Eybl, 1978). This is a no-wait sequence because the start time of each job on M_2 is equal to its completion time on M_1 due to the $a_j \geqslant b_j$ inequalities and the SPT order on M_1.

We show next that the $F2/k\ jobs,\ M_1 - \max/\sum C_j$ problem can be solved in $O(n^2)$ time by implementing the PK algorithm of Panwalkar and Koulamas (2012) with slight modifications. The PK algorithm is summarized next for a regular objective function of the job completion times $f(C)$ (where $f(C) = \sum C_j$ in the case of the $F2/k\ jobs,\ M_1 - \max/\sum C_j$ problem).

Let j denote the job in position $[j]$ in a sequence S_n (with n jobs). If job j is removed from S_n, then, the difference between the old $f(C)$ and the new $f(C)$ value is called the ***contribution*** of job j to S_n and is denoted as δ_j.

The PK algorithm starts with all n jobs in S_n sequenced in the SPT order on M_1. Then, it identifies the job u with the maximum contribution δ_u as the ***candidate*** job and removes it from S_n. Once a job is removed, it is not added in subsequent iterations. The time complexity of the PK algorithm is $O(n^2)$.

Proposition 1. *The PK algorithm solves the $F2/k\ jobs,\ M_1 - \max/\sum C_j$ problem optimally.*

Proof. Let u be the candidate in S_n and assume that an optimal S_{n-2} sequence is obtained by removing two other jobs $v,\ w$ (and retaining u). Without loss of generality, let us assume that $v \leftarrow w$ (that is v precedes w) in the SPT sequence on M_1. Then, $\delta_u = C_u + \sum_{j \rightarrow u}(C_j - C'_j)$, where C_j (C'_j) denotes the completion time of job j before (after) the removal of u and $j \rightarrow u$ indicates that j follows u in the SPT sequence on M_1; C_j and C'_j are defined similarly when either v or w is removed instead of u. The following two cases should be considered.

Case 1: $v \leftarrow u \leftarrow w$ or $v \leftarrow w \leftarrow u$ in the SPT sequence on M_1. We remove v before w in S_{n-2}. Prior to the removal of $v,\ \delta_w = C_w + \sum_{j \rightarrow w}(C_j - C'_j)$ and $\delta_u = C_u + \sum_{j \rightarrow u}(C_j - C'_j)$. After the removal of $v,\ \delta_{vw} = \delta_w - a_v$ and $\delta_{uv} = \delta_u - a_v$ because the removal of v reduces $C_u,\ C_w$ by the same amount (a_v) and does not change the value of $\sum_{j \rightarrow u}(C_j - C'_j)$ and $\sum_{j \rightarrow w}(C_j - C'_j)$; observe that for each job j in these summations, both C_j and C'_j are reduced by the same amount (a_v) when v is removed. Therefore, if $\delta_w < \delta_u$, then $\delta_{vw} < \delta_{uv}$ as well and it is more beneficial to remove u and retain w in S_{n-2}.

Case 2: $u \leftarrow v \leftarrow w$. We remove v before w in S_{n-2}. By definition, $\delta_v = C_v + \sum_{j \rightarrow v}(C_j - C'_j)$ and $\delta_u = C_u + \sum_{j \rightarrow u}(C_j - C'_j)$ when w is part of the sequence. If w is removed, then both C_j and C'_j are reduced by the same amount (a_w) for all jobs $j \rightarrow w$ in the $\sum_{j \rightarrow u}(C_j - C'_j)$ and $\sum_{j \rightarrow v}(C_j - C'_j)$ summations respectively. The removal of w also eliminates the term $C_w - C'_w$ from these summations; observe that $C_w - C'_w = a_v$ in δ_v and that $C_w - C'_w = a_u$ in δ_u. Therefore, $\delta_{vw} = \delta_v - a_v$ and $\delta_{uw} = \delta_u - a_u$.

Because of the SPT order on M_1, $a_u < a_v$; also, $\delta_v < \delta_u$ by assumption. Therefore, $\delta_{vw} = \delta_v - a_v < \delta_u - a_u = \delta_{uw}$ and it is more beneficial to remove u and retain v in S_{n-2}. □

We illustrate the PK algorithm on a 7-job $F2/k\ jobs,\ M_1 - \max/\sum C_j$ problem in the Appendix A. For comparison purposes, we also solve the corresponding $F2/k\ jobs,\ M_1 - \max/C_{\max}$ problem using the same dataset. As expected, the total completion time decreases more rapidly than the makespan in most cases especially when the number of jobs is large.

3. An $O(n^2)$ algorithm for the $O2/k\ jobs,\ M_1 - \max/C_{\max}$ problem

Gonzalez and Sahni (1976) showed that an optimal schedule for the $O2/M_1 - \max/C_{\max}$ problem with makespan value $C^*_{\max} = \max\left\{\sum_{j=1}^{n} a_j,\ \max_{j=1,\ldots,n}(a_j + b_j)\right\}$ can be obtained in $O(n)$ time. We will use this result in the $O2/k\ jobs,\ M_1 - \max/C_{\max}$ problem for all $k = 1, \ldots, n$. Jozefowska, Jurisch, and Kubiak (1994) showed that the $O2/d_i = d/n_T$ problem (with arbitrary job processing times) is ordinary NP-hard which implies that the $O2/k\ jobs/C_{\max}$ problem is also ordinary NP-hard.

Koulamas and Kyparisis (1998) solved the $O2/M_1 - \max,\ d_i = d/n_T$ problem in $O(n \log n)$ time by first discarding all jobs j with $a_j + b_j > d$ and then selecting as non-tardy jobs the first k jobs in the SPT sequence on M_1 where k is such that $\sum_{j=1}^{k} a_j \leqslant d$