Available online at www.sciencedirect.com

SCIENCE DIRECT®

European Journal of Operational Research 174 (2006) 1–10

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

A branch and bound algorithm to minimize the total tardiness for *m*-machine permutation flowshop problems

Chia-Shin Chung [a], James Flynn [a,*], Ömer Kirca [b]

[a] *Department of Operations Management and Business Statistics, Cleveland State University, 2121 Euclid Avenue BU 539, Cleveland, OH 44115-2214, United States*

[b] *Department of Industrial Engineering, Middle East Technical University, Ankara 06531, Turkey*

Received 3 January 2002; accepted 30 December 2004
Available online 14 April 2005

Abstract

The m-machine permutation flowshop problem with the total tardiness objective is a common scheduling problem, which is known to be NP-hard. Here, we develop a branch and bound algorithm to solve this problem. Our algorithm incorporates a machine-based lower bound and a dominance test for pruning nodes. We undertake a numerical study that evaluates our algorithm and compares it with the best alternative existing algorithm. Extensive computational experiments indicate that our algorithm performs better and can handle test problems with $n \leqslant 20$.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Scheduling; Permutation flowshop; Tardiness; Branch and bound

1. Introduction

In the *permutation flowshop problem*, each of n jobs has to be processed on machines $1,\ldots,m$ in that order. The processing times of each job on each machine are known. At any time, each machine can process at most one job and each job can be processed on at most one machine. Once the processing of a job on a machine has started, it must be completed without interruption. Also, each job must be processed in the same order at every machine. The usual objectives are the minimization of the make-span, flow-time, tardiness, lateness, and the number of jobs late (see Pinedo [13] for a review of the general flowshop problem). Here, the objective is to minimize the total tardiness. Tardiness equals the amount by which a job's completion time exceeds its due date.

* Corresponding author. Tel.: +1 216 687 4741/621 6349; fax: +1 216 687 9343.
E-mail address: j.flynn@csuohio.edu (J. Flynn).

0377-2217/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.12.023

Schedules where each job must be processed in the same order at every machine are called *permutation schedules*. When $m \leqslant 2$, the restriction to such schedules is harmless; however, when $m > 3$, there may exist a general schedule whose total tardiness is strictly less than the total tardiness of any permutation schedule (see [13]). Finding such a schedule is often computationally impractical; moreover, as discussed in Kim [8] there are many real situations where only permutation schedules are feasible. Most approaches to the m machine flowshop problem restrict attention to permutation schedules.

The general m machine flow time problem with the total tardiness time objective is NP-hard in the ordinary sense for $m = 1$ and NP-hard in the strong sense for $m \geqslant 2$ (the special case where the due dates are equal to 0 is NP-hard in the strong sense for $m \geqslant 2$): see Appendix D of [13]. Consequently, there has been great interest in developing heuristic solutions [5–7,16,22]. See [10] for more details on heuristic approaches and other general issues.

Most optimal algorithms for single machine problems combine dynamic programming or branch and bound with decomposition properties developed by Lawler [12], Potts and Wassenhove [15] and Szwarc [18]. Harikawa [4] proposes an effective algorithm, based on Lawler's decomposition theorem. Szwarc and Mukhopadhyay [20] and Tansel [21] develop branch and bound algorithms incorporating decomposition properties, which can handle problem with over 100 jobs. Recently, Szwarc et al. [19] employ an improved decomposition rule, which allows them to solve problems with 300 jobs.

Only a few articles deal with optimal algorithms for multiple machine problems. Kim [7] and Sen et al. [17] apply branch and bound to two machine problems. Kim [8] applies branch and bound algorithm to general m machine problems using a special branching scheme, which he calls "backward branching". In his scheme, a node corresponds to a partial sequence of jobs placed at the end of a whole job sequence. When a node branches, one or more nodes are defined by placing one more job in front of the partial sequence associated with the current branch node. After characterizing properties of nodes, Kim derives lower bounds for the branch and bound algorithm together with a dominance rule. The dominance rule leads to a problem size reduction procedure, which sometimes yields a problem with a smaller n.

In Section 2, we briefly describe the problem and derive a new machine based lower bound. Our bound employs estimates of the earliest start time for jobs assigned to each position in the job sequence. (These estimates were derived in Chung et al. [1], which dealt with the flow-time objective.) Section 3 presents a dominance test, which helps to prune nodes. An outline of the algorithm appears in Section 4. Section 5 reports on a numerical study that evaluates our algorithm and compares it with Kim [8]—the best alternative existing algorithm. Extensive computational experiments involving over 40,000 test problems suggest that our algorithm performs better and can handle problems with $n \leqslant 20$. Section 6 states our conclusions.

2. The problem description and a lower bound

Branch and bound algorithms are commonly applied to the permutation flowshop problem, which is known to be NP hard. The effectiveness of a branch and bound algorithm is heavily influenced by its lower bounds. Such bounds are more useful when they are tighter and easier to compute. In this section, we develop a useful machine-based lower bound.

Consider the search tree for our algorithm [11]. The root node $\emptyset$ represents the null schedule. Every other node represents a *partial schedule* $\sigma = (\sigma(1), \ldots, \sigma(s))$, indicating that job $\sigma(j)$ occupies the jth position on each machine, for $1 \leqslant j \leqslant s$, where $1 \leqslant s \leqslant n$. Any permutation $\bar{\sigma}$ of the set of unscheduled jobs defines a complete schedule $\sigma\bar{\sigma} = (\sigma(1), \ldots, \sigma(s), \bar{\sigma}(1), \ldots, \bar{\sigma}(n-s))$. By placing any unscheduled job i in position $(s+1)$, we produce a descendant node $\sigma i = (\sigma(1), \ldots, \sigma(s), i)$. This paper employs the following additional notation:

x^+	$\max\{x, 0\}$, for any real number x
N	$\{1, \ldots, n\}$, the set of all jobs