Computers & Operations Research 40 (2013) 1314–1327

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

An optimal method for the preemptive job shop scheduling problem

Abbas Ebadi, Ghasem Moslehi *

Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran

A R T I C L E   I N F O

Available online 20 December 2012

Keywords:
Scheduling
Branch and bound
Preemption
Job shop
Disjunctive graph

A B S T R A C T

In this study, a powerful solution methodology is developed for minimizing makespan in the preemptive Job Shop Scheduling Problem (pJSSP). Some new properties of the problem are stated and proved via theorems on the basis of which a new dominant set is introduced for the problem. These properties give rise to a dramatic decrease in the search space and provide the potential for exact methods to be successfully used in the solution of this notoriously NP-hard problem. The exact method presented here is a branch and bound algorithm developed on the basis of a new disjunctive graph. Its efficiency is enhanced by the effective use of such techniques as dominance rules or lower bounds. The capability of the approach is investigated by using it to solve the well-known benchmark problems and comparing the results obtained with those from the best methods in common use. The results indicate that the proposed method is capable of optimally solving 24 open benchmark problems including the famous 10×10 problems. Additionally, it is the first optimal method ever developed to find optimal solutions to some large-scale problems of the size 30×10 and 50×10.

1. Introduction

Job shop environments, as a generalization of single machine and flow shop environments, are associated with products of a wide variety, each of which has a unique production process. Scheduling in these environments can, therefore, have a considerable impact on the system's efficiency that can be further enhanced by preemption forming one of the most important scheduling assumptions. Preemption may be allowed in such processes as machining operations in which the amount of processing a job receives is not lost after preemption. Among other applications of preemption is included the assignment of human resources to different project activities. As illustrative examples, reference may be made to Guo et al. [1] who used a special case of this problem in the apparel industry, or to Anderson et al. [2] who, taking the hardware CPU and I/O for machines, used the 2-machine version of the problem to schedule computer systems.

Throughout this paper, a scheduling problem is designated by the triple notation $\alpha|\beta|\gamma$ [3]. It must be mentioned that preemption may decrease problem complexity in certain cases but it is not certainly the case with Job Shop Scheduling Problems (JSSP). For example, the two machine pJSSP with only three jobs ($J2|n=3,\text{prmp}|C_{\max}$) is an NP-hard problem; however, the nonpreemptive version with any arbitrary number of jobs ($J2|n=k|C_{\max}$) is solvable in a polynomial time [4]. The high complexity of the pJSSP has led researchers to use heuristic or approximation algorithms to solve the problem. This is verified by the literature review below.

A number of researchers have addressed a special case of the pJSSP with two machines. For example, Sevastianov and Woeginger [5] proposed a polynomial time approximation algorithm with the worst case ratio of 1.5 for the problem $J2|\text{prmp}|C_{\max}$. Kimbrel and Saia [6] considered this problem in online and offline environments and developed randomized heuristics with the worst case ratio of 2. Anderson et al. [2] studied the same problem and developed an algorithm that yields a schedule of length at most $P_{\max}/2$ greater than the optimal schedule length, where $P_{\max}$ is the length of the longest job.

Others have developed approxim\ation algorithms for the problem in its general form. For instance, Goldberg et al. [7] proposed approximation algorithms with the ratio $O(\log m\mu/\log\log m\mu)$ for the problem $J_m|\text{prmp}|C_{\max}$, where m is the number of machines and μ is the maximum number of operations per job. Bansal et al. [8] developed approximation algorithms with the ratio $O(\log m/\log\log m)$ for the problems $J_m|\text{prmp}|C_{\max}$ or $J_m|p_{ij}=1|C_{\max}$. They presented a $(2+\varepsilon)$ approximation, in which $0<\varepsilon<1$, for a constant number of machines and an algorithm with the ratio 1.45 for the problem $J2|\text{prmp}|C_{\max}$. Jansen et al. [9] presented a polynomial time approximation scheme for JSSP and pJSSP with a fixed number of machines and a fixed number of operations per job. Leighton et al. [10,11] developed approximation algorithms with a fixed ratio for the problem $J_m|p_{ij}=1|C_{\max}$.

* Corresponding author. Tel.: +98 311 3915509; fax: +98 311 3915526.
E-mail addresses: a.ebadi@in.iut.ac.ir (A. Ebadi), moslehi@cc.iut.ac.ir, moslehi@istt.ir (G. Moslehi).

0305-0548/$ - see front matter
http://dx.doi.org/10.1016/j.cor.2012.12.004

Another group of authors have addressed pJSSP with some additional assumptions. Jansen et al. [12] studied JSSP and pJSSP with controllable processing times, in the sense that processing time may be decreased at an additional cost. They developed polynomial time algorithms to minimize both makespan and cost. Akcora et al. [13] studied a case of pJSSP in which a reward is given to each operation based on its completion time. Considering the maximization of the total rewards as the objective function, they developed a heuristics using a mathematical model.

Exact solution methods for solving the $J_m|\text{prmp}|C_{\max}$ problem have been rarely reported in the literature if ever. The most important attempt in this field is due to Le Pape and Baptiste in the form of scientific reports [14], conferences presentations [15], and papers [16,17] during the period from 1994 to 1999. Unlike other researchers, they developed optimal methods in addition to heuristics to solve the problem. Common in all their work is the use of constraint programming and its related techniques. In 1998, they proposed different constraint propagation techniques based on time tables, disjunctive constraints, and edge finding to optimally solve the problem $J_m|\text{prmp}|C_{\max}$ [16]. To the best of our knowledge, this study includes the best records in optimally solving the benchmark pJSSPs. In their following study in1999, they combined different heuristic search strategies with various constraint propagation techniques [17].

More recently, Baptiste et al. [18] have shown that there exists an optimal schedule for preemptive job shop scheduling problems with integral data in which all interruptions and all starting and completion times occur at integral dates. This implies that we can break down each operation into a chain of unit length components. They have also introduced new upper bounds on the minimal number of interruptions.

The desirability of optimal solutions and the unavailability of exact methods reported in the literature for the problem $J_m|\text{prmp}|C_{\max}$ encouraged us to conduct this study aimed at developing efficient, exact methods for the optimal solution of such problems. We recently developed novel mathematical models for the preemptive shop scheduling problems using a commercial software for their solution [19]. The dimension in these models, unlike those of the previously reported ones, depends solely on the number of jobs and machines irrespective of processing times. Here, we develop a branch and bound solution that is capable of solving tougher instances.

In Section 2, the problem $J_m|\text{prmp}|C_{\max}$ is represented by a disjunctive graph which serves as the basis of a branch and bound method presented in Section 3. Dominance rules, lower bounds, and a preference rule are used to improve its efficiency. In Section 4, the method is used to solve the famous benchmark problems [20]. The results are then compared with those obtained from the best methods available. The last Section provides conclusions and suggestions for future studies.

2. Disjunctive graph

In the pJSSP, there are n jobs and m machines. Each job has its own sequence of operations, and each operation should be processed on a particular machine. The objective is to schedule operations on machines so that the maximum completion time is minimized. Problem assumptions can be stated as follows. Processing times are deterministic and sequence independent. All jobs are ready to be processed at time zero. Only one job can be processed on each machine at a given period of time. Each job visits each machine once at most and preemption is allowed; in other words, processing of any operation may be interrupted and resumed later. The notation and definitions used in the problem $J_m|\text{prmp}|C_{\max}$ are as follows.

n	number of jobs
m	number of machines
p_{ik}	processing time for job i on machine k. ($i=1, 2, \ldots, n$, $k=1, 2, \ldots, m$)
P	total processing time ($P=\sum_{i=1}^{n}\sum_{k=1}^{m} p_{ik}$)
O_{ik}	chain of job i on machine k including p_{ik} operations with unit processing times ($i=1, \ldots, n$, $k=1, \ldots, m$)
O_{ikj}	jth unit-operation of O_{ik}. ($i=1, \ldots, n$, $k=1, \ldots, m$, and $j=1, \ldots p_{ik}$)
$O_{ik\rho}$	last unit-operation of chain O_{ik}. This notation is used instead of $O_{ik\,p_{ik}}$ to avoid two levels of subscripts.
$G=(F, W, Y)$	graph G, where F is set of nodes, W is set of conjunctive arcs, and Y is set of disjunctive arcs
$M(i, l)$	the machine required to process the lth operation of job i ($i=1, 2, \ldots, n$, $l=1, 2, \ldots, m$)
A	an arbitrary nondelay schedule for the problem $J_m|\text{prmp}|C_{\max}$
A_k	schedule on machine k ($k=1, \ldots, m$) in schedule A; so $A=\{A_1, A_2, \ldots, A_m\}$
S_{ik}^A	start time of job i on machine k in schedule A. ($i=1, \ldots, n$, $k=1, \ldots, m$)
C_{ik}^A	completion time of job i on machine k in schedule A ($i=1, \ldots, n$, $k=1, \ldots, m$)
D_{ik}^A	due date of job i on machine k determined according to schedule A_k; $D_{ik}^A=C_{ik}^A$ ($i=1, \ldots, n$, $k=1, \ldots, m$)
$R_{i,M(i,l)}^A$	ready time of job i on machine $M(i, l)$ determined according to schedule $A_{M(i,l-1)}$; $R_{i,M(i,l)}^A=C_{i,M(i,l-1)}^A$ ($i=1, \ldots, n$, $l=2, \ldots, m$) The ready time of each job on its first machine is taken to be equal to zero, i.e., $R_{i,M(i,1)}^A=0$.
B	a schedule for the problem $J_m|\text{prmp}|C_{\max}$ where jobs are scheduled on any machine k ($k=1, \ldots, m$) according to the preemptive version of the Earliest Due Date rule (pEDD) based on times R_{ik}^A and D_{ik}^A
B_k	schedule on machine k ($k=1, \ldots, m$) in schedule B; so $B=\{B_1, B_2, \ldots, B_m\}$
$T_{\max}$	maximum tardiness; $\underset{1\le i\le n}{\text{Max}}(\text{Max}\{0, C_i-D_i\})$

Based on the above definitions, it may be concluded that the release date of the next operation of a job is the completion date of the preceding operation, or its due date:

$$R_{i,M(i,l)}^A=D_{i,M(i,l-1)}^A \quad l=2,\ldots,m \tag{1}$$

Considering a scheduling objective, a sub set of all feasible schedules is said to be dominant when, for any schedule not belonging to the dominant set, there is a better or equivalent schedule in it.

In the problem $J_m|\text{prmp}|C_{\max}$, the nondelay schedules, say *A Schedules* or briefly *AS*, constitute the dominant set because the objective function $C_{\max}$ is a regular performance measure and preemption is allowed [3]. In Theorems 1 and 2 below, a new dominant set, say *B Schedules* or briefly *BS*, is introduced, which is a subset of *AS* and whose cardinality is considerably smaller than that of *AS*. Set *BS* is a set of nondelay schedules in which the schedule on each machine follows the pEDD rule provided the completion time of any job on each machine ($C_{i,M(i,l)}$) is considered as the due date ($D_{i,M(i,l)}$) and the completion time of its previous operation ($C_{i,M(i,l-1)}$) is considered as the ready time ($R_{i,M(i,l)}$). The pEDD rule is the preemptive version of Jackson's algorithm [21] and can be executed as follows. Whenever a new job becomes available, if its due date is earlier than that of the job being processed, then the processing of the current job is interrupted for the new job to be processed. Whenever a job is completed, the unscheduled job with the earliest due date is processed.