PERGAMON

Omega 29 (2001) 599–608

omega
The International Journal of Management Science

www.elsevier.com/locate/dsw

Heuristic algorithms for the two-machine flowshop with limited machine availability ☆

Jacek Błażewicz[a], Joachim Breit[b], Piotr Formanowicz[a], Wiesław Kubiak[c], Günter Schmidt[d, *]

[a] *Institute of Computing Science, Poznań University of Technology, Piotrowo 3a, 60-965 Poznań, Poland*
[b] *Lehrstuhl für Informations- und Technologiemanagement, Universität des Saarlandes, Postfach 15 11 50, D-66041 Saarbrücken, Germany*
[c] *Faculty of Business Administration, Memorial University of Newfoundland, St. John's, Nfld., Canada A1C 5S7*
[d] *Institut für Informatik, Technische Universität Clausthal, Julius Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany*

Received 15 August 2000; accepted 16 July 2001

Abstract

The paper studies a flowshop scheduling problem where machines are not available in given time intervals. The objective is to minimize the makespan. The problem is known to be NP-hard for two machines. We analyze constructive and local search based heuristic algorithms for the two-machine case. The algorithms are tested on easy and difficult test problems with up to 100 jobs and 10 intervals of non-availability. Computational results show that the algorithms perform well. For many problems an optimum solution is found. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Flowshop; Availability constraints; Scheduling; Heuristics

1. Introduction

The problem studied in this paper can be described as follows. Each of n jobs is to be processed on two machines M_1 and M_2 in this order. The processing times are given. Each machine may be unavailable for processing jobs in given time intervals which we call *holes* for convenience. At any time, each machine can process at most one job and each job can be processed on at most one machine. Preemption is allowed. The objective is to find a schedule which minimizes the maximum completion time $C_{\max}$. Fig. 1 shows a schedule for a problem instance with four jobs and three holes.

In scheduling theory the basic model assumes that all machines are continuously available for processing throughout the planning horizon. This assumption might be justified in some cases but it does not apply if certain maintenance requirements, breakdowns or other constraints that cause the machines not to be available for processing have to be considered.

This kind of constraints appear e.g. within MRP-II planning systems on a tactical level when a rolling horizon planning approach is used. Here, two consecutive time periods overlap where decisions taken in the first period constrain decisions for the second period. From this the problem arises that we have two sets of order requirements, one set having a fixed assignment to time intervals and the other being assigned to the remaining free processing intervals.

The same kind of problem is repeated on the operational level of production scheduling. Here, some jobs are fixed in terms of starting and finishing times and resource allocation. When new jobs come in there are already jobs assigned to time intervals and corresponding machines while the new ones have to be processed using the remaining free processing intervals.

☆ This research has been supported by the NATO Collaborative Research Grant CGR 950438.

* Corresponding author.

PII: S0305-0483(01)00048-2

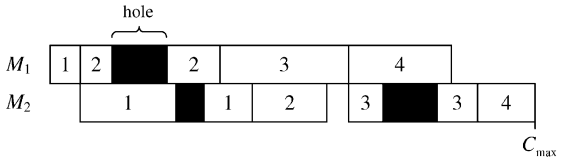

Fig. 1. Example of a schedule.

Another application of limited machine availability comes from operating systems for mono- and multi-processors, where subprograms with higher priority will interfere with the current program executed. Numerous other examples exist where the investigation of limited machine availability is of great importance. This has also been proven by a growing market demand for software packages that can handle this feature of scheduling problems.

For quite some time this type of scheduling problem has also attracted many researchers. For a recent survey of results on different kind of models refer to Sanlaville and Schmidt [1] and Schmidt [2]. However, research on flowshop problems with limited machine availability has started only recently.

Since Johnson [3] and Garey et al. [4] it is well known for the case of continuous machine availability that the problem of minimizing the makespan is easy to solve for two machines (Johnson's rule) and that it becomes NP-hard for more than two machines.

Lee [5] has shown that the problem gets more difficult if holes have to be considered. The problem already becomes NP-hard for two machines if there is a single hole on one machine only. In the same paper two approximation algorithms are presented, depending on the machine where the hole is located; one has a relative error of $\frac{3}{2}$ if the hole is on machine one, the other of $\frac{4}{3}$ if the hole is on machine two. Recently, Cheng and Wang [6] provided an improved approximation algorithm for the case with the hole on machine one; this algorithm has a relative error of $\frac{4}{3}$. Lee [5] also proposes a dynamic programming algorithm for the case with one hole only.

Kubiak et al. [7] prove that no polynomial time heuristic with a finite worst case bound can exist when there are at least two holes, provided that the later one occurs on machine two. They furthermore prove that makespan minimization becomes NP-hard in the strong sense even if an arbitrary number of holes occur on one machine only. On the other hand they show that there always exists an optimal schedule where the permutation of jobs scheduled between any two consecutive holes obeys Johnson's order.

Due to these negative results in the same paper a branch and bound algorithm is given to solve the problem. The approach uses Johnson's rule for jobs scheduled between two consecutive holes.

Taking into account that the branch and bound algorithm does not guarantee to find an optimal solution within a given time limit we want to analyze the performance of two constructive heuristics and one local search heuristic. The two constructive heuristics are Johnson's rule and a new *look-ahead heuristic* which is based on local optimization. From the variety of local search heuristics we choose simulated annealing because this method has proved to be very successful especially for solving flowshop scheduling problems [8–11]. In Section 2 we formulate the problem in greater detail. In Section 3 we present the heuristic algorithms we investigate. Section 4 describes the design of our experiment. The computational results are presented in Section 5. We finish with some conclusions.

2. Problem formulation

The flowshop scheduling problem considered relates to a set $\mathscr{J}$ of n jobs that have to be processed on two machines M_1 and M_2. Each job consists of two tasks. The first task of job j has to be processed on M_1 requiring p_{1j} time units, the second on M_2 requiring p_{2j} time units ($j=1,\ldots,n$). Before the second task can be processed on M_2 the first task has to be finished on M_1, i.e. each job visits machine M_1 first and then machine M_2. Both machines can process only one job at a time.

For each machine there may exist time intervals, during which the machine is not available for processing (holes). The total number of holes is H. The start time of hole h is denoted by s_h, the length of the same hole by $l_h, h=1,\ldots,H$. The machine on which hole h occurs is denoted by $M(h)\in\{M_1,M_2\}$. Without loss of generality we assume that no two holes overlap. When a hole on a machine interferes with the assignment of a task, the processing of the task is stopped at the beginning of the hole and it is resumed at the end. A task may be preempted by more than one hole. Let C_j denote the point of time job j is finished on M_2. The objective is to find a schedule for which $C_{\max}=\max_j\{C_j\}$ is minimized.

Using the exchange argument given in [3] it can be shown that it is sufficient to consider only permutation schedules. Thus a solution to the problem can be represented by a permutation $\pi=(\pi(1),\pi(2),\ldots,\pi(n))$ of length n; $\pi(k)$ denotes the job in position k. The permutation π shows the processing sequence of the jobs on both machines. The set of solutions is represented by the $n!$ possible permutations of the n jobs.

3. Heuristic algorithms

3.1. Constructive methods

3.1.1. Johnson's rule (JOH). This rule, introduced in [3], solves the problem to optimality if both machines are continuously available. The set of jobs $\mathscr{J}$ is split into disjoint subsets $\mathscr{J}_1$ and $\mathscr{J}_2$. $\mathscr{J}_1$ contains all jobs which have processing times $p_{1j}\leqslant p_{2j}$; $\mathscr{J}_2$ contains the remaining jobs. The jobs from $\mathscr{J}_1$ are scheduled first according to