ELSEVIER

Journal of Materials Processing Technology 107 (2000) 459–465

Journal of Materials Processing Technology

www.elsevier.com/locate/jmatprotec

Heuristic algorithm for scheduling in the no-wait flow-shop

Edy Bertolissi

IRIDIA, ULB, Av. F. Roosevelt 50 CP 194/6, 1050 Brussels, Belgium

Abstract

Sequencing and scheduling are forms of decision making which play a key role in manufacturing. This paper presents the description of a scheduling algorithm for the solution of the *no-wait* flow-shop problem. The goal of the heuristic is to minimise the sum of the total flow-times, a criterion which at the same time minimises the average processing time. This is a sensible target in applications such as agile manufacturing where jobs are constantly added to the job list. When evaluated over a large number of problems of various sizes, the heuristic is found to be very effective in yielding near-optimal solutions. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Heuristic algorithm; Scheduling; No-wait flow-shop

1. Introduction

Scheduling is concerned with setting the timetable for the processing of a given set of jobs on a set of machines in order to optimise a given measure of performance. Many researchers have proposed approaches for solving these problems which have an immediate application in many fields of manufacturing. The large number of publications in this area is related to the fact that there is an enormous number of different characteristics and objectives according to the environment to schedule, associated with a wide choice of the parameters to optimise. Pinedo [13] offers a comprehensive review on optimisation and heuristic methods in production scheduling.

Scheduling algorithms are usually designed to solve special problems involving constraints on jobs and scheduling objectives [12]. Different problems may be characterised by different flow patterns of the jobs. The flow pattern may be the same for all the jobs (*flow-shop*), or each job may have its own individual flow pattern (*job-shop*), or no specified flow pattern may exist (*open-shop*). Parallel or duplicated machines may be present. Within these environments, the goal is to find a schedule which tries to optimise a specified objective, such as the *completion time*, the *tardiness*, or the *flow-time* of the jobs.

In many manufacturing cells, there is the constraint that all the jobs have to follow the same path, and that once a job is started it must be processed until completion without any interruption either on or between machines. Such a flow-shop is usually called *no-wait flow-shop*. This type of scheduling is necessary where operations are required to follow one immediately after the other due to production constraints, such as during the production of steel or plastic moulding. In addition, modern manufacturing environments such as *agile* manufacturing, where robots and industrial machines provide a highly coordinated process, can frequently be modelled as a no-wait scheduling problem. A comprehensive review on the state of the art in this area of scheduling is presented in [9].

This paper focuses on the objective of minimising the sum of all completion times of the jobs in a no-wait flow-shop. This scheduling problem can be synthetically denoted as an $F_m|\text{no-wait}|\sum C_j$ problem, where F_m indicates a flow-shop problem with m devices, and C_j is the completion time of the job j. Minimising the total flow-time is equivalent to minimising the sum of completion times, and consequently the average processing time. In other terms, the aim of the algorithm is to finish each job as soon as possible. This is an appropriate criterion in the case of manufacturing workcells, where minimising the average time for completing a job is a sensible target since new jobs are constantly added to the job list. At present, there are only a few available algorithms which have been proposed for solving this problem. Van Deman and Baker [6] proposed a *branch and bound* approach to find the optimal solution proposing a set of procedures for generating lower bounds on optimal values. Heuristic algorithms have also been proposed by Rajendran and Chauhuri [14] and Gangadharan and Rajendran [7]. These algorithms produce better schedules when compared with the ones derived using the heuristics developed by Bonney and Gundry [3] and King and Spachis [11]. The

E-mail address: eberto@iridia.ulb.ac.be (E. Bertolissi).

theoretical aspects of the continuous flow-shop have been investigated by Gupta [8], Szwarc [15], and Adiri and Pohoryles [1].

This article presents a heuristic algorithm based on *job insertion* for the no-wait flow-shop problem whose aim is to minimise total flow-time starting from a suboptimal sequence. The main idea of the proposed heuristic is to improve the results of the schedules generated using the heuristic described in [2] by using them as a starting point for a job insertion algorithm as described in the work of Rajendran and Chauhuri [14].

The remainder of the paper is structured as follows. Section 2 presents the foundations of the proposed heuristic approach. Section 3 exemplifies the proposed algorithm using a simple scheduling problem. Section 4 reports the details of the computational experience and a comparison of the performance of our algorithm with earlier proposed heuristics. Finally, Section 5 will present some final comments on the presented approach.

2. The heuristic algorithm

The following assumptions are taken in the no-wait flow-shop problem:

- job operation times are deterministic and known in advance;
- an operation once started on a machine cannot be interrupted before completion (no *pre-emption*);
- jobs consist of a strictly ordered sequence of operations;
- the same job sequence is assumed for each job, but machine skipping is allowed;
- once a job is started, it must be processed until completion without any interruption either on or between machines.

It is not felt that these assumptions deter from the possibility of using the described methodology in real applications.

The following notations are used in this paper:

n	number of jobs to be scheduled
m	number of machines in the flow-shop
t_{ij}	processing time for the ith job on the jth machine
$F_{m(ij)}$	total flow-time for the job i followed by the job j on m machines
σ	available partial schedule
n'	number of jobs in the set σ
d_{ij}	minimum delay on the first machine between the start of job i and job j rendered necessary by the no-wait constraint

The minimum delay terms d_{ij} can be calculated using the following expression:

$$d_{ij} = t_{i1} + \max_k \left(\sum_{p=2}^{k} t_{pi} - \sum_{p=1}^{k-1} t_{pj}, 0 \right) \quad \text{with } 2 \le k \le m \qquad (1)$$

Therefore, the total flow-time of n jobs is given by

$$F = \sum_{i=2}^{n} (n+1-i) d_{[i-1][i]} + \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} \qquad (2)$$

It has been shown by Wiser [16] that the flow-shop scheduling problem with the no-wait constraint is reducible to an asymmetric travelling salesman problem (ATSP). While he has considered the delays d_{ij} and the sum of the processing times as intercity costs, here the weighted sum of the delays for the flow-time objective is considered (Eq. (2)). Karg and Thompson [10] have suggested a heuristic algorithm for the ATSP based on city insertion to form a tour. However, this procedure suffers from the drawback of the random selection of the seed city pair and of the order of insertion of the cities. Rajendran and Chauhuri [14] have proposed an improved heuristic for the selection of the job seed and the insertion of the new jobs in the partial schedule based on the sum of the lower bounds on the job completion time.

This article proposes a heuristic approach based on the definition of an initial schedule which is used for generating the job seed and defining the order in which jobs will be considered by the job insertion algorithm. The algorithm for the selection of the seed job and the order of inserting the jobs in the available partial schedule is based on the work by Chan and Bedworth [4], which had been adapted to the *no-wait* flow-shop case by Bertolissi [2]. Originally, the algorithm was designed to solve the problem of minimising the mean flow-time in the n-job, m-machine case in static and dynamic environments. The extension of the methodology to the *no-wait* flow-shop environment has proved to be possible whilst maintaining the same approach. To derive the formula which is the basis of this approach, it is necessary to start from the fact that in a no-wait two-job, two-machine environment, where job i precedes job j, there are two possible flow-time sequences as shown in Fig. 1. From Fig. 1(A), if $t_{j1} > t_{i2}$, it is possible to see that the flow-time for job i is equal to $(t_{i1} + t_{i2})$. Similarly the flow-time for job j is $(t_{i1} + t_{j1} + t_{j2})$. The total flow-time is the sum of the two flow-times:

$$F_{2(ij)} = 2t_{i1} + t_{i2} + t_{j1} + t_{j2} \qquad (3)$$

If $t_{j1} < t_{i2}$, as in Fig. 1(B), the total flow-time is equal to

$$F_{2(ij)} = 2t_{i1} + 2t_{i2} + t_{j2} \qquad (4)$$

Therefore, in the case of a two-jobs two-machines environment, the total flow-time is equal to

$$\begin{aligned} F_{2(ij)} &= 2t_{i1} + t_{i2} + t_{j2} + \max(t_{j1}, t_{i2}) \\ &= 2t_{i1} + t_{i2} + t_{j2} + R_{2(ij)} \end{aligned} \qquad (5)$$

which in the case of the m machine problem becomes

$$F_{m(ij)} = 2t_{i1} + \sum_{k=2}^{m} t_{ik} + R_{m(ij)} \qquad (6)$$

The detailed mathematical derivation of this formula which