Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 34 (2007) 1033 – 1050

computers & operations research

www.elsevier.com/locate/cor

Makespan minimization in a no-wait flow shop problem with two batching machines

Ammar Oulamara*

MACSI Project, LORIA–INRIA Lorraine, Ecole des Mines de Nancy, Parc de Saurupt, 54042 Nancy, France

Available online 1 July 2005

Abstract

This paper deals with the problem of task scheduling in a no-wait flowshop with two batching machines. Each task has to be processed by both machines. All tasks visit the machines in the same order. Batching machines can process several tasks per batch so that all tasks of the same batch start and complete together. The batch processing time for the first machine is equal to the maximal processing time of the tasks in this batch, and for the second machine is equal to the sum of the processing times of the tasks in this batch. We assume that the capacity of any batch on the first machine is bounded, and that when a batch is completed on the first machine it is immediately transferred to the second machine. The aim is to make batching and sequencing decisions that allow the makespan to be minimized.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: No-wait flowshop problem; Batch processing machines; Complexity

1. Introduction

The no-wait flowshop scheduling problem with two batching machines considered in this article can be formulated as follows. There are n tasks to be batched and scheduled for processing on two machines. Each machine is a batching machine that can process several tasks in batch. Each task must be processed on both machines 1 and 2 and visiting them in this order. The processing time for task j on machines 1 and 2 is at least p_j and q_j time units, respectively. All tasks in a same batch start and finish their processing on the same batching machine at the same times (batch availability).

* Tel.: +33 3 83 58 17 83; fax: +33 3 83 58 43 28.
E-mail address: ammar.oulamara@loria.fr.

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.05.028

The first batching machine (1) is called a *max-batch* or a *parallel-batch* (*p-batch*) machine, and handles several tasks simultaneously. The batch processing time on this machine is equal to the maximum processing time of the tasks in this batch. This machine's batch capacities are bounded by some integer value $b(b < n)$. The second machine (2) is called *sum-batch* or a *serial-batch* (*s-batch*) machine, and handles tasks sequentially. The batch processing time on this machine is equal to the sum of the task processing time for this batch. Such machines require a setup time prior to batch execution.

There are two setup time variants. In the first variant—the anticipatory or detached model—the setup can be processed before the associated batch is available on the machine; in the second variant—the non-anticipatory or attached model—the setup time can be processed only when all the tasks assigned to a corresponding batch become available on the machine. The batches are processed in a no-wait fashion, so that each batch's completion time on machine 1 will coincide with its starting time on machine 2.

A schedule is characterized by partitioning the tasks into batches for each machine and then sequencing the batches. Due to the no-wait constraints, batch partitions and batch sequences are the same for both machines, and the objective is to find a schedule that will minimize the batch completion time. By using the general notation for scheduling problems introduced by Graham et al. [1] and the notation of batching machines [2] and [3], we write $F2|p\text{-}batch(1), s\text{-}batch(2), b, no\text{-}wait|C_{\max}$ to refer to this problem.

Motivation for the problem studied in this paper comes from processing products in iron and steel industry. During each working phase, metal sheets first pass through a multi-head hole-punching machine, which simultaneously punches batches of holes according to their position on the sheet. Then, following a short preparation period, the sheet undergoes a series of sequential bending operations [4].

Potts and Van Wassenhove [5], Webster and Baker [6], and Potts and Kovalyov [3] have published state-of-the-art surveys of the batch scheduling problem. Potts et al. [7] have studied the flowshop problem with two parallel batching machines, but without the no-wait constraint. They offered a polynomial algorithm for the problem of minimizing batch completion time when both parallel batching machines can process an unbounded number of tasks in same batch, and proved the NP-hardness of this problem when at least one of the machines can process up to b tasks in same batch $b < n$. Oulamara et al. [8] studied the no-wait flowshop problem with two parallel batching machines, and proposed a polynomial algorithm for the above problem, though they also extended their algorithm to the case of m parallel batching machines. Lin and Cheng [9] studied flowshops with two serial batching machines and a no-wait constraint. They showed that the problem of minimizing the completion time is strongly NP-hard and identified some cases that can be solved polynomially. Hall et al., [10] examined the problems of partitioning a set of identical tasks into batches in the no-wait flowshop with serial batching machines. They developed a dynamic programming algorithm for the single product problem and formulated the multi-product problem as generalized traveling salesman problem. Oulamara and Finke [11] studied the flowshop problem with two batching (p-batch and s-batch) machines, but without a no-wait constraint. They proved that the minimization of the makespan criterion is polynomial for the case in which batch capacity is unbounded on parallel batching machines. For the bounded case, they prove that the makespan minimization is NP-hard.

The remainder of this article is organized as follows. In Section 2, we introduce the notation used in this paper. In Section 3, we prove that the makespan minimization is strongly NP-hard. Section 4 addresses the case in which all tasks have the same processing times on the first machine, and we prove that when the capacity of the first machine is equal to two, the makespan minimization is reduced to a problem of maximum weight matching and that when this capacity is greater than or equal to three, the makespan minimization is strongly NP-hard. Section 5 addresses the case in which all tasks have the