Contents lists available at ScienceDirect

Int. J. Production Economics

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

A two-stage flexible flow shop problem with unit-execution-time jobs and batching

Enrique Gerstl[a,b], Gur Mosheiov[a,*]

[a] School of Industrial Engineering, Jerusalem College of Technology, Jerusalem, Israel
[b] School of Business Administration, The Hebrew University, Jerusalem, Israel

ARTICLE INFO

Article history:
Received 12 November 2013
Accepted 5 August 2014
Available online 13 August 2014

Keywords:
Deterministic scheduling
Flexible flowshop
Makespan
Batch scheduling

ABSTRACT

We study a batch-scheduling problem of unit-time jobs on a two-stage flexible flowshop. The objective functions are minimum makespan and minimum flowtime. Unlike previously studied models: (i) a general number of machines in both stages of the flowshop is allowed, and (ii) there is no restriction on the number of batches to be processed on each machine. Efficient exact dynamic programming algorithms are introduced. Extensions to the case of machine-dependent setup times are studied as well. All the proposed algorithms run in polynomial time in the number of jobs.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The well-known two-stage flowshop problem consists of a series of two machines, such that all the jobs must be processed on both machines, following the same order. Makespan minimization on this classical setting is known to be solvable in polynomial time (Johnson, 1954). Most of the following studies (e.g., extensions to more machines, or assuming other objective functions), lead to hard problems. Specifically, the extension to minimizing makespan on a two-stage flexible flowshop setting, which consists of several parallel machines in each stage of the flowshop, is known to be NP-hard. In fact, the latter problem was shown to be NP-hard even if one of the stages consists of a single machine, the other contains two parallel identical machines, and preemption is allowed (Hoogeveen et al., 1996). We refer the reader to the recent survey on flexible flowshops by Ruiz and Vazquez-Rodriguez (2010), and to more recent studies, e.g., Trung and Ng (2011), Defersha and Chen (2012), Almeder and Hartl (2013), Choi and Lee (2013), Emmons and Vairaktarakis (2013), Li et al. (2013), and Mainieri and Ronconi (2013).

A special case of a two-stage flexible flowshop, which appears to have many applications and has attracted a number of researchers, is that of a *critical machine* in one of the two stages. In this setting, all the jobs are processed on a single critical machine in one of the two stages, and on one of several parallel identical machines in the other stage. The case of a single (critical) machine in stage 1 and several machines in stage 2 is known in the literature as *look-ahead flowshop* (LAFS), and the case of several machines in stage 1 and a single (critical) machine in stage 2 is known in the literature as *look-behind flowshop* (LBFS); see e.g., Lee and Vairaktarakis (1998). As mentioned above, these problems are hard even when the number of parallel identical machines is two.

Recently, an interesting variant of both LAFS and LBFS has been studied. Specifically, the two-stage flexible flowshop has been considered with (i) *unit execution time* (*UET*) jobs, and (ii) the option of *batching*. Recall that when the option of batching is considered, jobs may be processed in groups, and a setup time is required prior to starting a new batch. In most scheduling models with batching, *batch availability* is assumed, i.e., the completion time of jobs contained in any given batch is defined as the completion time of the entire batch. Both assumptions of UET jobs (representing e.g. production lines of identical items) and production with batching, reflect numerous applications. We refer the reader to Gerstl and Mosheiov (2013a, 2013b) for LAFS with UET jobs and batching, and Gerstl et al. (2013) for LBFS with UET jobs and batching. In Gerstl and Mosheiov (2013a), a common (machine-independent) setup time is considered, and each batch is processed on a different second-stage machine. An important question referring to the optimal number of machines to be used becomes relevant, and it can be shown that the intuitive decision of using as many machines as possible is not necessarily optimal. Gerstl and Mosheiov (2013b) extends this LAFS model to a setting of machine-dependent setups. The option of using Not-All-Machines (NAM)

[*] Corresponding author. Tel.: +972 2 588 3108; fax: +972 2 588 1341.
E-mail address: msomer@huji.ac.il (G. Mosheiov).

http://dx.doi.org/10.1016/j.ijpe.2014.08.004
0925-5273/© 2014 Elsevier B.V. All rights reserved.

remains an important issue. [Note that the solution for the model introduced in Gerstl and Mosheiov 2013b is optimal only under the above assumption of a single batch per one second-stage machine. Thus, Property 1 in Gerstl and Mosheiov 2013b (implying that the number of batches is identical to the number of the second-stage used machines) is in fact wrong, and should be an *assumption* of the model rather than a property of an optimal schedule.] As mentioned, the case of LBFS with UET jobs and batching (with common machine-independent setup times) is studied in Gerstl et al. (2013).

In this paper we generalize the above models (of makespan and flowtime minimization), in two aspects. First, we allow a general number of machines in both stages of the flowshop. Secondly, we allow all the machines to process more than a single batch. These generalizations lead to a significantly different solution approach. In all previously published models (focusing on a critical machine in one of the two stages, and restricting all other machines to process a single batch), the *relaxed* version of the problem (i.e., when non-integer batch sizes are allowed) has been solved to optimality first. Then, a rounding procedure has been introduced. While this procedure was proved to be extremely fast (in most cases the solution of the relaxed version was shown to be given in closed form, and the rounding procedure is linear in the number of jobs), and very accurate, it does not guarantee an optimal schedule. In this paper we introduce a new approach: an *exact* polynomial time solution for the original (integer) version of the problem is presented. Specifically, for given numbers of machines in stages 1 and 2 of the flowshop, we introduce a polynomial time dynamic programming algorithm which guarantees an optimal schedule. If m_1 and m_2 are the numbers of the machines in stages 1 and 2, respectively, the total running time of our proposed algorithm is $O(n^{m_1+m_2})$ (where n is the number of jobs). [Note that while this running time is polynomial in the number of jobs (for given m_1 and m_2 values), it is not polynomial in the input size, which contains only four numbers: the number of jobs, the number of machines in stage 1, the number of machines in stage 2, and the (common) setup time.] We further extend these models (of both makespan and flowtime minimization) to the setting of machine-dependent setup times. The running times of our proposed algorithms for these extensions are shown to remain polynomial in the number of jobs.

In Section 2 we provide the notation and formulation of the problem. Section 3 presents the dynamic programming (DP) algorithm for makespan minimization. Section 4 contains the DP for flowtime minimization. A discussion and conclusion are provided in Section 5.

2. Formulation

There are n UET jobs to be processed on a two-stage flexible flowshop (FFS). Stage 1 consists of m_1 parallel identical machines, and stage 2 consists of m_2 parallel identical machines. We denote this machine setting by FFS(m_1, m_2). Jobs may be processed in batches. Each batch is processed first on one of the first-stage machines, and then on one of the second-stage machines. Prior to starting a new batch (in both stages), a setup time, denoted by S, is required. We assume (*i*) *batch availability*: all the jobs belonging to a given batch are completed when the entire batch (i.e., the last job of the batch) is completed; (ii) *non-anticipatory* setup times: processing a setup time of a batch on the second stage must start only after the entire batch has been completed on the first stage; (iii) *batch consistency*: allocation of jobs to batches is consistent throughout the production process (i.e., batches remain unchanged in both stages). The objective functions considered here are minimum makespan and minimum flowtime.

For a given job schedule consisting of k batches, let n_i denote the number of jobs assigned to batch $i, i=1,\dots,k$. Clearly, $\sum_{i=1}^{k} n_i = n$. The completion time of batch i in stage r is denoted by $C_i^r, i=1,\dots,k, r=1,2$. The makespan of a given schedule is the maximum completion time of any of the batches on the second stage. The flowtime is the sum of the job completion times, i.e., the sum of the completion time of the batches multiplied by the batch sizes. Thus, let $C_{max} \equiv \max\{C_i^2, i=1,\dots,k\}$, and $\sum C_j = \sum_{i=1}^{k} C_i^2 n_i, i=1,\dots,k$. It follows that formally, the two-stage flexible flowshop problems studied here are:

FFS$(m_1, m_2)/S$, UET $/C_{max}$.

FFS$(m_1, m_2)/S$, UET$/\sum C_j$.

When machine-dependent setup times are assumed. We denote by S_i the setup time of machine $i, i=1,\dots,m_1+m_2$. Thus, the more general problems solved here are:

FFS$(m_1, m_2)/S_i$, UET $/C_{max}$.

FFS$(m_1, m_2)/S_i$, UET$/\sum C_j$.

3. A dynamic programming algorithm for FFS$(\boldsymbol{m}_1, \boldsymbol{m}_2)/\boldsymbol{S}$, UET$/\boldsymbol{C_{max}}$

In this section we introduce a solution procedure for makespan minimization on a two machine flexible flowshop with any given number of machines in stages 1 and 2 (m_1 and m_2, respectively). Despite being a polynomial time solution, the proposed algorithm (a dynamic programming) contains a large number of state variables (specifically: $2m_1+m_2+1$). We thus focus first (for ease of exposition) on two important special cases with a relatively small number of jobs.

3.1. *A special case:* FFS(1, 2)

The first special case that we solve is *FFS*(1, 2), i.e., a two-stage flexible flowshop containing a single machine in stage 1 and two parallel identical machines in stage 2. Let M_1 denote the machine in stage 1, and M_2 and M_3 the parallel machines in stage 2. We present a dynamic programming algorithm (denoted *DP*1) based on the following state variables:

j – the number jobs already scheduled,
C_1 – the current completion time of the last job on M_1;