FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 39 (2014) No. 1

DOI: 10.2478/fcds-2014-0001 ISSN 0867-6356
e-ISSN 2300-3405

BATCH SCHEDULING IN A TWO-STAGE FLEXIBLE FLOW SHOP PROBLEM

Enrique Gerstl*, Gur Mosheiov * and Assaf Sarig**

Abstract. We study a special two-stage flexible flowshop, which consists of several parallel identical machines in the first stage and a single machine in the second stage. We assume identical jobs, and the option of batching, with a required setup time prior to the processing of a new batch. We also consider the option to use only a subset of the available machines. The objective is minimum makespan. A unique optimal solution is introduced, containing the optimal number of machines to be used, the sequence of batch sizes, and the batch schedule. The running time of our proposed solution algorithm is independent of the number of jobs, and linear in the number of machines.

Keywords: Deterministic Scheduling, Flexible Flowshop, Batch Scheduling

1. Introduction

A flexible flowshop is a machine setting consisting of several stages in series, where each stage contains a number of parallel machines. Minimizing makespan on a general flexible flowshop is clearly NP-hard, since it is a generalization of both minimum makespan on parallel identical machines and minimum makespan on a classical flowshop (with at least three machines). In fact, minimum makespan on a flexible flowshop was shown to be strongly NP-hard even for the special case of (i) two stages, (ii) two identical machines in one stage and a single machine in the second stage, and (iii) when preemption is allowed (Hoogeveen et al. [11]). We refer the reader to the recent survey on flexible flowshops by

* Enrique Gerstl, School of Business Administration, The Hebrew University, Jerusalem, Israel;
Gur Mosheiov, School of Business Administration, The Hebrew University, Jerusalem, Israel.
** Assaf Sarig, The Center for Academics Studies, Or Yehuda, Israel

Acknowledgement: This paper was supported in part by The Recanati Fund and The Charles Rosen Chair of Management, The School of Business Administration, The Hebrew University, Jerusalem, Israel.

Ruiz and Vazquez-Rodriguez [25], which contains 225 references, dealing with various combinations of flexible flowshop settings and objective functions.

Several researchers studied the special two-stage flexible flowshop setting, where the first stage consists of m parallel identical machines and the second stage contains a single machine. This setting is known in the literature as *look-behind flowshop* (LBFS); see e.g., Lee and Vairaktarakis [15]. LFBS has numerous applications, including the standard manufacturing system of several parallel identical production machines followed by a single machine (station) for painting, rapping, loading, assembly, etc. [Lee and Vairaktarakis [15] also defined the symmetric setting, in which the first stage contains a single machine and the second stage consists of m parallel identical machines. They denoted this setting by LAFS (*look-ahead flexible flowshop*).] A similar setting of a two-stage flowshop with a single (critical) machine in one of these stages and several *dedicated* machines in the other stage has been studied by e.g., Oguz et al. [23], Cheng and Kovalyov [5], Lin [16], Kyparisis and Koulamas [14], Lin and Liao [17], Mosheiov and Yovel [21], Cheng et al. [6], Oguz and Ercan [22], and Gerstl and Mosheiov [9], among others.

In this paper we focus on an LFBS system, in the context of *batch scheduling*; see e.g., Monma and Potts [18] and Allahverdi et al. [1]. Recall that in batch scheduling jobs may be grouped and processed in batches. The processing time of a batch is identical to the total processing times of the jobs contained in the batch. Prior to starting a new batch (either on one of the first stage machines, or on the second stage machine), a *setup* time is performed, during which the production process is stopped. Each of the batches is performed on one of the first stage machines, and upon completion, continues (as a batch) to the second stage machine. Each first-stage machine can process (at most) a single batch. As commonly assumed in batch scheduling models, we consider: (i) *batch availability*, i.e., jobs can start processing on the second-stage machine only when their entire batch is completed on the first-stage machine; (ii) *non-anticipatory* setup times, i.e., prior to starting the batch setup on the second-stage machine, the entire batch must be completed and released from the first-stage machine; (iii) *batch consistency*, i.e., a batch remains unchanged on the first-stage and on the second-stage machine. Finally, we consider here the very practical setting of *identical processing time jobs*. The numerous studies of scheduling identical jobs reflect the many applications of this setting, in particular the many types of production lines of identical items (see e.g., the two recent surveys: Baptiste and Brucker [2], and Kravchenko and Werner [13]). We note that the special case of the problem studied here; where the parallel machines in each stage of the flexible flowshop are replaced by a single machine (i.e., minimizing makespan on an m-machine flowshop with identical jobs and batching) has been solved in Mosheiov and Oron [19].

In a recent paper, Fanjul-Peyro and Ruiz [7] introduced and tested numerically algorithms for solving systems with the option of using only a subset of all the available machines. They focused on the option of Not-All-Machines (NAM) on parallel-unrelated machines. Previously, Cao et al. [3] introduced a tabu search algorithm to minimize the machine holding cost for a NAM model on parallel identical machines; Chen and Li [4] considered the case that the number of machines can be reduced by outsourcing jobs to an external production line; Finke et al. [8] studied a NAM model with precedence constraints, and Kravchenko and Werner [12] considered NAM problems with release dates and deadlines. Among the many applications of the NAM decision, Fanjul-Peyro and Ruiz [7] mention a typical setting of a shop in which some machines remain idle due to the large capacity of the system, which exceeds the total demand.

Given all the above (unit jobs, batching, NAM), an optimal solution for our proposed LBFS model consists of the following decisions: (i) How many first-stage machines to use; (ii) Given the number of machines - how to allocate jobs to batches; and (iii) How to schedule these batches. First we introduce a lower bound on the optimal makespan, obtained by solving to optimality the relaxed version of the problem in which non-integer batch sizes are permitted. This solution consists of a unique increasing sequence of batch sizes. Then, we convert this solution into an optimal integer schedule. The total running time is shown to be independent of the number of jobs, and is linear in the number of first-stage machines. (As indicated later, the running time of the algorithm is not polynomial in the input size. However, since the number of batches that need to be calculated and stored is of the order of the number of the machines, this running time seems to be the smallest possible.)

In a recent paper, Gerstl and Mosheiov [10] studied the LAFS version of the problem, i.e., when a single machine is considered in stage 1, and m parallel identical machines in stage 2. LAFS and LBFS may have significantly different applications. An LAFS system may consist of a single common production machine followed by several parallel customization stations. LBFS may model a plant having a number of parallel manufacturing units in stage one, followed by e.g., a common quality-control/rapping/painting station. Despite the different nature of the two models, the analysis of both appears to be related. Using a similar approach to that used by Gerstl and Mosheiov [10], we obtain a non-standard sequence of optimal batch sizes for the relaxed version of the problem (where non-integer batch sizes are allowed). We then introduce a rounding procedure, which guarantees an optimal solution for the original (integer) version.

The paper is organized as follows: Section 2 contains the notation and the problem formulation; Section 3 provides the lower bound based on the solution of the relaxed version; Section 4 presents the optimal integer solution.

2. Formulation

We study a 2-stage flexible flowshop (FFs), where the first stage consists of m parallel identical machines, and the second stage consists of a single common machine (called *critical*). We denote this special flowshop structure by $FFs(m,1)$. There are n identical jobs, which are assumed to have unit processing times after appropriate scaling. Thus, if p_{ij} denotes the processing time of job j on machine i, we have $p_{ij} = 1, i = 1, \dots, m+1,\ j = 1, \dots, n$ (where machine $m+1$ is the critical machine). The jobs processed on machine i ($i = 1, \dots, m$) are processed later as a single block (batch) on the critical machine. An integer setup time, denoted by S, is required prior to starting the process of a batch on each of the first m machines, as well as on the critical machine. We assume a machine-independent setup time. If k machines are used in the first stage ($k \le m$), then for a given allocation of jobs to these k machines, the number of jobs assigned to machine i is denoted by $n_i, i = 1, \dots, k$. Clearly, $\sum_{i=1}^{k} n_i = n$. Finally, recall that our model assumes *batch availability*, *non-anticipatory* setup times, and *batch consistency*; see above.

For a given schedule, the completion time of the last job on machine i (i.e., the completion time of batch i on the first stage machine) is denoted by $C_i, i = 1, \dots, k$. The

completion time of batch i on the critical machine is denoted by $C_i^{(CR)}, i = 1, \dots, k$. Let $C_{max} = \max\{C_i^{(CR)}, i = 1, \dots, k\}$. Thus, the problem studied in this paper is: $FFs(m,1)/\, S, p_{ij} = 1\, /C_{max}$.

3. A lower bound on the optimal makespan value

A solution for the problem $FFs(m,1)/\, S, p_{ij} = 1\, /C_{max}$ consists of: (i) a decision on the optimal number of (the first-stage) machines to be used, (ii) the allocation of jobs to machines, and (iii) the job schedule. According to the well-known *reversibility* property in flowshops, the makespan does not change if the jobs go through the flowshop in the opposite direction in the reverse order; see e.g. Pinedo [24]. Thus, a possible solution procedure could be based on the reversed solution of $FFs(1,m)/\, S, p_{ij} = 1\, /C_{max}$, given in (10). However, due to the different structure of the optimal schedules in both cases, we introduce in the following the properties of an optimal schedule for $FFs(m,1)$, and consequently we provide a complete solution for this problem.

We first solve the *relaxed* version of the problem, allowing non-integer batch sizes. The optimal solution for the relaxed version is clearly a *lower bound* on the optimal makespan of $FFs(m,1)/\, S, p_{ij} = 1\, /C_{max}$. We use the following notation for the relaxed version: $n_i^{(R)}$ is the size of batch i in the relaxed version, $C_i^{(R)}$is the completion time of batch i on the first stage machine, and $C_i^{(CR,R)}$ is the completion time of batch i on the critical machine. $C_{max}^{(*,R)}$ is the optimal makespan for the relaxed version. In the following we prove several properties of an optimal schedule:

Property 1: If $S \geq n$, an optimal schedule exists such that a single machine is used in the first stage ($k = 1$).

Proof: For $k = 1$ (single *machine* is used in the first stage) the makespan is $2S + 2n$. For $k \geq 2$ the makespan is at least $(k+1)S + n$, which is larger than $2S + 2n$. ■

In the remainder of this paper we assume $S < n$.

Property 2: *For* a given number k of machines used, an optimal schedule exists such that $n_{i+1}^{(R)} = S + 2n_i^{(R)}, i = 1, \dots, k-1$.

Proof: We focus first on the *first* two batches, and prove that $n_2^{(R)} = S + 2n_1^{(R)}$.

Assume that an optimal schedule q exists such that: $n_2^{(R)} > S + 2n_1^{(R)}$. The completion time of the second batch on the critical machine is given by: $C_2^{(CR,R)}(q) = 2S + 2n_2^{(R)}$. Let $\epsilon = n_2^{(R)} - S - 2n_1^{(R)} > 0$. We create a schedule q' by increasing $n_1^{(R)}$ by $\epsilon/3$ ($n_1^{'} = n_1^{(R)} + \epsilon/3$), and decreasing $n_2^{(R)}$ by $\epsilon/3$ ($n_2^{'} = n_2^{(R)} - \epsilon/3$). We obtain: $C_2^{(CR,R)}(q') = 2S + 2n_2^{'} = 2S + 2(n_2^{(R)} - \epsilon/3) < C_2^{(CR)}(q)$.

Assume now that in schedule q: $n_2^{(R)} < S + 2n_1^{(R)}$. The completion time of the second batch on the critical machine is given by: $C_2^{(CR,R)}(q) = 3S + 2n_1^{(R)} + n_2^{(R)}$. Let $\epsilon = 2n_1^{(R)} + S - n_2^{(R)} > 0$. We create a schedule q' by decreasing $n_1^{(R)}$ by $\epsilon/3$ $(n_1^{'} = n_1^{(R)} - \epsilon/3)$, and increasing $n_2^{(R)}$ by $\epsilon/3$ $(n_2^{'} = n_2^{(R)} + \epsilon/3)$. We obtain: $C_2^{(CR,R)}(q') = 3S + 2n_1^{'} + n_2^{'} = 3S + 2(n_1^{(R)} - \epsilon/3) + (n_2^{(R)} + \epsilon/3) < C_2^{(CR,R)}(q)$.

We conclude that schedule q' (with $n_2^{(R)} = S + 2n_1^{(R)}$) is optimal as well.

The remaining proof is by induction. Assume that $n_{i+1}^{(R)} = S + 2n_i^{(R)}, i = 1, \dots, l$ for some $l < k - 1$. A similar proof leads to the equality $n_{l+2}^{(R)} = S + 2n_{l+1}^{(R)}$. It follows that for a given k values, $n_{i+1}^{(R)} = S + 2n_i^{(R)}, i = 1, \dots, k - 1$. ■

Property 3: For a given number k of machines used, an optimal schedule exists such that

$$n_1^{(R)} = \frac{n - S(2^k - k - 1)}{2^k - 1} \tag{1}$$

Proof: *Since* $n_{i+1}^{(R)} = S + 2n_i^{(R)}, i = 1, \dots, k - 1$ (Property 2), we can easily express $n_k^{(R)}$ as a function of $n_1^{(R)}$ and S:

$$n_k^{(R)} = 2^{k-1} n_1^{(R)} + (2^{k-1} - 1)S. \tag{2}$$

From $\sum_{i=1}^{k} n_i^{(R)} = n$ and Property 2, it follows: $n = \sum_{i=1}^{k} n_i^{(R)} = n_1^{(R)} + \sum_{i=2}^{k} n_i^{(R)} = n_1^{(R)} + \sum_{i=2}^{k} \left(2n_{i-1}^{(R)} + S\right) = n_1^{(R)} + (k-1)S + 2\sum_{i=1}^{k-1} n_i^{(R)} = n_1^{(R)} + (k-1)S + 2(n - n_k^{(R)})$.

Thus,

$$n_k^{(R)} = \frac{n + n_1^{(R)} + (k-1)S}{2}. \tag{3}$$

From equations (2) and (3) we obtain that $2^{k-1} n_1^{(R)} + (2^{k-1} - 1)S = \frac{n + n_1^{(R)} + (k-1)S}{2}$. It follows that:

$$n_1^{(R)} = \frac{n - S(2^k - k - 1)}{2^k - 1}. \tag{4}$$ ■

Based on Properties 2 and 3, the makespan value for a given k value is the sum of the idle time on the critical machine (which is identical to the setup time and the total processing time of the first batch, i.e., $S + n_1^{(R)}$), and the total processing time of the k batches on the critical machine (i.e., $kS + n$). Thus,

$$C_k^{(CR,R)} = S + n_1^{(R)} + kS + n = S(k+1) + \frac{n - S(2^k - k - 1)}{2^k - 1} + n. \tag{5}$$

In order to find the optimal makespan value (for the relaxed version), we have to solve (5) for all k values. Let k^* denote the optimal number of machines used. Clearly, $1 \le k^* \le m$. The following property introduces better bounds on k^*:

Property 4: The optimal number of machines to be used is bounded by:

$$log_2\left(1+\left(\frac{n}{S}+1\right)ln(2)\right) \le k^* \le log_2\left(1+\left(\frac{n}{S}+m\right)ln(2)\right). \quad (6)$$

Proof: *Allowing* k to be non-integer, $C_k^{(CR,R)}$ is continuous and convex in k. Thus, the optimal (non-integer) k value can be found by standard derivation. The derivative of (5) with respect to k is:

$$\frac{dC_k^{(CR,R)}}{dk} = S + \frac{\left(-S2^k\ln(2)+S\right)\left(2^k-1\right)-\left(n-S\left(2^k-k-1\right)\right)2^k\ln(2)}{\left(2^k-1\right)^2}.$$

$\frac{dC_k^{(CR,R)}}{dk} = 0$ leads to: $S2^k - Sk\ln(2) = S + n\ln(2)$, or

$$2^k - k\,ln(2) = 1 + \frac{n\ln(2)}{S}. \quad (7)$$

Since $1 \le k \le m$, the left-hand-side of (7) is bounded by:

$$2^k - m\ln(2) \le 2^k - k\ln(2) \le 2^k - \ln(2).$$

It follows that:

$$2^k - m\ln(2) \le 1 + \frac{n\ln(2)}{S} \le 2^k - \ln(2)\ .$$

We obtain the following bounds on the optimal number of machines used:

$$k^* \le \log_2\left(1+\left(\frac{n}{S}+m\right)\ln(2)\right);$$

$$k^* \ge \log_2\left(1+\left(\frac{n}{S}+1\right)\ln(2)\right). \quad \blacksquare$$

Since k is clearly bounded by m, and must be an integer, the actual upper bound on its value is

$$k^{UB} \equiv \min\left\{m, \left\lfloor \log_2\left(1+\left(\frac{n}{S}+m\right)\ln(2)\right)\right\rfloor\right\}. \quad (8)$$

Similarly, the actual lower bound on k is

$$k^{LB} \equiv min\left\{m, \left\lfloor \log_2\left(1+\left(\frac{n}{S}+1\right)\ln(2)\right)\right\rfloor\right\}. \quad (9)$$

The optimal number of machines, k^*, is a non-decreasing function of the number of jobs n, and a non-increasing function of the setup time S. Figure 1 demonstrates k^* as a function of n (for a given S value; $S = 20$). Similarly, Figure 2 demonstrates k^* as a function of S (for a given n value; $n = 1000$).

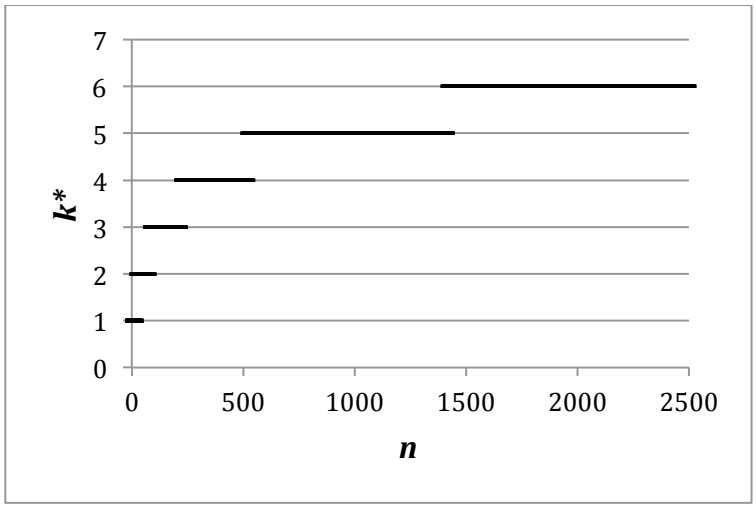

Figure 1. The optimal number of used machines as a function of the number of jobs ($S = 20$).

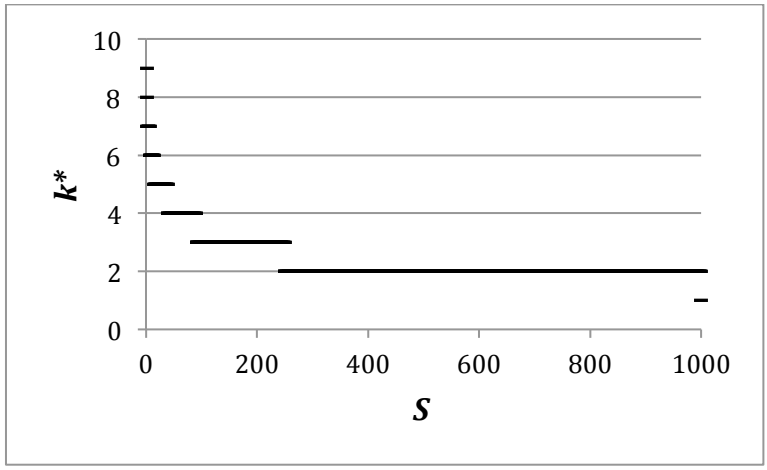

Figure 2. The optimal number of used machines as a function of the setup time ($n = 1000$).

Based on all the above, we introduce in the following a formal algorithm:

Lower Bound *Algorithm* (*optimum of the relaxed version*):
Input: m, n, S;
Step 1: Calculate k^{UB} and k^{LB} from (8) and (9).
Step 2: For $k = k^{LB}$ to k^{UB}
Calculate $C_k^{(CR,R)}$ from (5).
Step 3: The optimal makespan is given by (5): $C_{max}^{(*,R)} = \min_{k^{LB} \le k \le k^{UB}} \left\{ C_k^{(CR,R)} \right\}$.
k^* is the appropriate k value.

The optimal sequence of batch sizes is given by (4) and (2) (for k^*).

Running Time: Step 1 requires a constant time. Step 2 is performed $O(m)$ times, and each iteration requires a constant time. Finding $C_{max}^{(*,R)}$ in Step 3 requires $O(m)$, and then calculating the batch sizes requires $O(m)$ as well. Thus, the total running time is $O(m)$.

Numerical Examples: In the following we provide the solution for three 20-machine problems. The problems are different significantly from each other in the ratio n/S. As expected (see (8) and (9)), the number of candidates for the optimal number of machines decreases as this ratio increases.

Example 1: $n = 1{,}000, S = 8, m = 20$. In Step 1 of the algorithm we calculate the following bounds: $k^{UB} = 6.665$; $k^{LB} = 6.465$. It follows that $k = 6$ and $k = 7$ are candidates. We check the makespan of both k values (Step 2). This leads to the following optimal number of batches: $k^* = 7$, and to the optimal solution (Step 3): $n_1^{(R)} = 0.315, n_2^{(R)} = 8.630, n_3^{(R)} = 25.259, n_4^{(R)} = 58.520, n_5^{(R)} = 125.039, n_6^{(R)} = 258.079, n_7^{(R)} = 524.157$; $C_{max}^{(*,R)} = 1064.315$; see Figure 3.

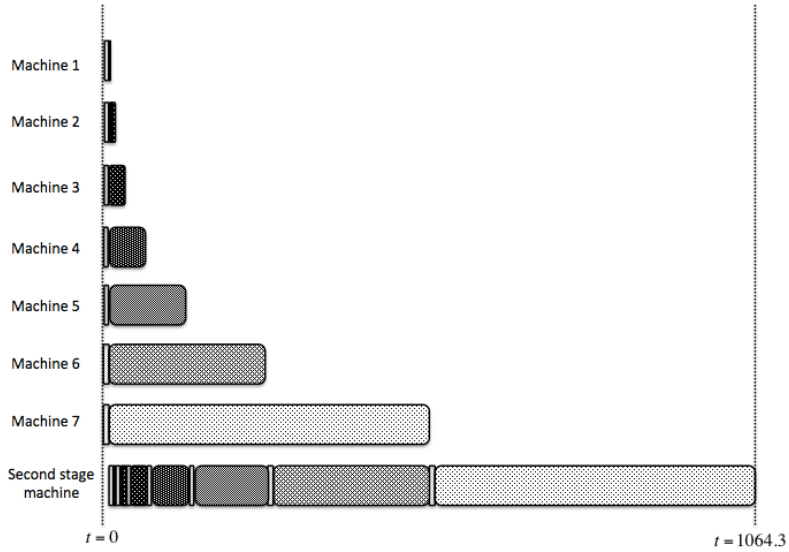

Figure 3. Optimal solution for Example 1 (the relaxed version): $k^* = 7$; $n_1^{(R)} = 0.315$, $n_2^{(R)} = 8.630$, $n_3^{(R)} = 25.259$, $n_4^{(R)} = 58.520$, $n_5^{(R)} = 125.039$, $n_6^{(R)} = 258.079$, $n_7^{(R)} = 524.157$; $C_{max}^{(*,R)} = 1064.315$.

Example 2: $n = 1{,}000, S = 75, m = 20$. The bounds on the optimal number of machines used are: $k^{UB} = 4.591$; $k^{LB} = 3.451$. The candidates are $k = 3, 4, 5$. After checking all

three candidates, we obtain the following optimum: $k^* = 4; n_1^{(R)} = 11.667, n_2^{(R)} = 98.333, n_3^{(R)} = 271.667, n_4^{(R)} = 618.333;\ C_{max}^{(*,R)} = 1386.667$.

Example 3: $n = 100{,}000, S = 8, m = 20$. The bounds on the optimal number of batches are: $k^{UB} = 13.083;\ k^{LB} = 13.081$. The candidates are $k = 13, 14$. The optimal solution consists of: $k^* = 13;\ n_1^{(R)} = 4.221, n_2^{(R)} = 16.442, n_3^{(R)} = 40.885, n_4^{(R)} = 89.770, n_5^{(R)} = 187.539, n_6^{(R)} = 383.079, n_7^{(R)} = 774.158, n_8^{(R)} = 1{,}556.316, n_9^{(R)} = 3{,}120.632, n_{10}^{(R)} = 6{,}249.264, n_{11}^{(R)} = 12{,}506.528, n_{12}^{(R)} = 25{,}021.055, n_{13}^{(R)} = 50,050.111;\ C_{max}^{(*,R)} = 110,116.221$.

4. An optimal integer solution

In the previous section we introduced a lower bound on the optimal makespan, obtained by solving the relaxed version of the problem, i.e., when non-integer batch sizes are permitted. In this section we convert this schedule into an optimal integer solution (with integer batch sizes). The optimal makespan value for the relaxed version, $C_{max}^{(*,R)}$, is a lower bound on the optimal makespan for the integer version. It is clear that even the smallest integer larger than or equal to $C_{max}^{(*,R)}$, i.e., $\lceil C_{max}^{(*,R)} \rceil$, remains a lower bound. Hence, an integer solution whose makespan is $\lceil C_{max}^{(*,R)} \rceil$ is optimal. In the following we introduce an algorithm that creates a schedule with this makespan value. We refer the reader to Mosheiov et al. [20], where a similar procedure was used to obtain an integer (not necessarily optimal) solution for a single machine batch-scheduling problem.

The optimal solution for the relaxed version of the problem consists of k^* (the optimal number of batches) and $n_1^{(R)}, n_2^{(R)}, \ldots n_{k^*}^{(R)}$ (the batch sizes). Let $\Delta_i = n_i^{(R)} - \lfloor n_i^{(R)} \rfloor, i = 1, \ldots, k^*$, i.e., Δ_i is the "non-integer" part of the size of batch i. Let $\Delta = \sum_{i=1}^{k^*} \Delta_i$. Since $n = \sum_{i=1}^{k^*} n_i^{(R)} = \sum_{i=1}^{k^*} (\lfloor n_i \rfloor + \Delta_i) = \Delta + \sum_{i=1}^{k^*} \lfloor n_i \rfloor$, Δ must be an integer. Based on these values, we convert the non-integer solution into an integer solution, using the following *Rounding Procedure*: we round up the first Δ batch sizes, and round down the remaining $k^* - \Delta$ batch sizes. We use the following notation for the integer solution obtained by this procedure: $n_i^{(I)}$ is the (integer) size of batch i, $C_i^{(I)}$ is the completion time of batch i on the first stage machine, and $C_i^{(CR,I)}$ is the completion time of batch i on the critical machine.

Note that the batch sizes obtained by the above procedure are given by:

$$n_i^{(I)} = \lceil n_i^{(R)} \rceil, i = 1, \ldots, \Delta;$$
$$n_i^{(I)} = \lfloor n_i^{(R)} \rfloor, i = \Delta + 1, \ldots, k^*. \quad (10)$$

Clearly, the total "rounded up processing time" is identical to the total "rounded down processing time", i.e.,

$$\sum_{i=1}^{\Delta} (1 - \Delta_i) = \sum_{i=\Delta+1}^{k^*} \Delta_i. \quad (11)$$

Property 5: The solution based on the integer batch sizes $n_i^{(I)}, i = 1, \dots, k^*$ contains no idle time between consecutive batches on the critical machine.

Proof: In order to prove this property, we have to show that $C_i^{(I)} \le C_{i-1}^{(CR,I)},\ i = 2, \dots, k^*$. We focus first on the first Δ batches (Claim 1 and Claim 2), and then on the remaining $k^* - \Delta$ batches (Claim 3).

Claim 1: $C_i^{(I)} \le C_{i-1}^{(CR,I)},\ i = 2, \dots, \Delta$ (i.e., there is no idle time between consecutive batches among the set of the first (rounded-up) Δ batches). We have to show that for $i = 2, \dots, \Delta$: $S + n_i^{(I)} \le 2S + 2n_{i-1}^{(I)}$, or

$$\lceil n_i^{(R)} \rceil \le S + 2\lceil n_{i-1}^{(R)} \rceil, \text{ or (by Property 2) } \lceil S + 2n_{i-1}^{(R)} \rceil \le S + 2\lceil n_{i-1}^{(R)} \rceil.$$

Since S is integer, we have to show that

$$S + \lceil 2n_{i-1}^{(R)} \rceil \le S + 2\lceil n_{i-1}^{(R)} \rceil, \tag{12}$$

which is always correct.

We conclude that for the first Δ batches, there is no idle time between any two consecutive batches on the critical machine. Note that (12) is either equality or strict inequality, implying that the difference $C_{i-1}^{(CR,I)} - C_i^{(I)}$ cannot decrease when proceeding from one batch to the next. The maximum difference is obtained after completing the entire set of the Δ rounded up batches. Denote by X the total rounded up processing times of the first Δ batches, i.e., $X \equiv \sum_{i=1}^{\Delta}(1 - \Delta_i)$.

Claim 2: $C_\Delta^{(CR,I)} - C_{\Delta+1}^{(I)} > \lfloor X \rfloor$.

$$C_\Delta^{(CR,I)} - C_{\Delta+1}^{(I)} = C_\Delta^{(CR,R)} + \sum_{j=1}^{\Delta}(1 - \Delta_j) + (1 - \Delta_1) - (C_{\Delta+1}^{(R)} - \Delta_{\Delta+1}).$$

(Note that $\Delta_{\Delta+1}$ is the non-integer part of the $\Delta + 1$-st batch.)

Since $C_\Delta^{(CR,R)} = C_{\Delta+1}^{(R)}$ (in the relaxed version the completion time of a given bacth on the first-sage machine is identical to the completion time of the previous batch on the critical machine), it follows that,

$$C_\Delta^{(CR,I)} - C_{\Delta+1}^{(I)} = \sum_{j=1}^{\Delta}(1 - \Delta_j) + (1 - \Delta_1) + \Delta_{\Delta+1} = X + (1 - \Delta_1) + \Delta_{\Delta+1} > \lfloor X \rfloor.$$

Since $C_\Delta^{(CR,I)} - C_{\Delta+1}^{(I)}$ is an integer, it follows from Claim 2 that $C_\Delta^{(CR,I)} - C_{\Delta+1}^{(I)} \ge \lceil X \rceil$.

Claim 3: $C_i^{(I)} \le C_{i-1}^{(CR,I)},\ i = \Delta + 1, \dots, k^*$ (i.e., there is no idle time between consecutive batches among the set of the last (rounded down) $k^* - \Delta$ batches).

Now we have to show that for $i = \Delta, \dots, k^* - 1$: $S + n_{i+1}^{(I)} \le S + n_1^{(I)} + iS + \sum_{j=1}^{i} n_j^{(I)}$, or

$S + \lfloor n_{i+1}^{(R)} \rfloor \le S + \lceil n_1^{(R)} \rceil + iS + \sum_{j=1}^{\Delta} \lceil n_j^{(R)} \rceil + \sum_{j=\Delta+1}^{i} \lfloor n_j^{(R)} \rfloor$, or

$S + n_{i+1}^{(R)} - \Delta_{i+1} \le$

$S + n_1^{(R)} + (1 - \Delta_1) + iS + \sum_{j=1}^{\Delta} n_j^{(R)} + \sum_{j=1}^{\Delta}(1 - \Delta_j) + \sum_{j=\Delta+1}^{i} n_j^{(R)} - \sum_{j=\Delta+1}^{i} \Delta_j$.

From the solution of the relaxed version we have:

$$S + n_{i+1}^{(R)} = S + n_1^{(R)} + iS + \sum_{j=1}^{i} n_j^{(R)} .$$

Thus, we have to prove that:

$$-\Delta_{i+1} \leq (1 - \Delta_1) + \sum_{j=1}^{\Delta}(1 - \Delta_j) - \sum_{j=\Delta+1}^{i} \Delta_j. \quad (13)$$

Recall that $\sum_{j=1}^{\Delta}(1 - \Delta_j) = X$ is the total rounded up processing times of the first Δ batches. $\sum_{j=\Delta+1}^{i} \Delta_j$ is the total rounded down processing times of batches $\Delta + 1, \Delta + 2, \dots, i$. From (11) it follows that $\sum_{j=\Delta+1}^{k^*} \Delta_j = X$. Thus, if $Y = \sum_{j=\Delta+1}^{i} \Delta_j$, then $Y \leq X$ for any $i = \Delta + 1, \dots, k^*$. Hence, the left-hand-side of (13) is not positive, whereas the right-hand-side is not negative, which completes the proof. ■

Corollary 6: The makespan value obtained by the batch sizes $n_i^{(I)}, i = 1, \dots, k^*$ (defined in (10)) is given by:

$$C_{max}^{(I)} \equiv S + \left\lceil n_1^{(R)} \right\rceil + k^* S + n = S(k^* + 1) + \left\lceil \frac{n - S(2^{k^*} - k^* - 1)}{2^{k^*} - 1} \right\rceil + n \quad (14)$$

Note that (14) is identical to the lower bound on the optimal makespan ($\left\lceil C_{max}^{(*,R)} \right\rceil$, see above), implying that our proposed procedure guarantees an optimal solution, i.e., $C_{max}^{*} = C_{max}^{(I)}$.

Running time: Given the batch sizes of the relaxed version, calculation of each Δ_i requires a constant time, i.e., an $O(m)$ effort for all batches (since $k^* \leq m$). Calculation of Δ as well the $C_{max}^{(I)}$ values requires $O(m)$ time as well. Hence, the running time of the solution procedure of the integer version requires $O(m)$ time.

It follows that the entire solution procedure (consisting of (i) obtaining the optimal batch sizes for the relaxed version by the algorithm specified in Section 3, and of (ii) the above procedure for obtaining integer batches) requires $O(m)$ time.

Comment 1: We note that the input contains three numbers only: m, n and S, implying that the proposed ($O(m)$) algorithm is *not polynomial* in the input size. However, as mentioned in the introduction, since there are $O(m)$ batch sizes to be calculated and stored, a faster algorithm appears to be impossible.

Comment 2: In Gerstl and Mosheiov [10], a rounding procedure for $FFs(1, m)$ was introduced, in order to convert the solution of the relaxed version into an integer solution. No proof of optimality of the resulting (integer) solution was provided. Due to (i) the fact that the rounding procedure suggested above (for $FFs(m, 1)$) was proved to be optimal, and (ii) the reversibility property in flowshops mentioned above, we claim that the rounding procedure guarantees an optimal solution for $FFs(1, m)$ as well.

Numerical Examples: In the following we provide the integer solution for Examples 1-3 solved above for the relaxed version.

Example 1 (integer): $n = 1{,}000, S = 8, m = 20$. Recall that the optimal solution of the relaxed version consists of: $k^* = 7$, $n_1^{(R)} = 0.315, n_2^{(R)} = 8.630, n_3^{(R)} = 25.259, n_4^{(R)} = 58.520, n_5^{(R)} = 125.039, n_6^{(R)} = 258.079, n_7^{(R)} = 524.157$, and $C_{max}^{(*,R)} = 1064.315$. We obtain $\Delta = 2$. It follows that the size of the first two batches is rounded up and that of the remaining (5 batches) is rounded down. Hence, an optimal solution to the problem is: $n_1^{(I)} = n_1^* = 1$, $n_2^{(I)} = n_2^* = 9$, $n_3^{(I)} = n_3^* = 25$, $n_4^{(I)} = n_4^* = 58$, $n_5^{(I)} = n_5^* = 125$, $n_6^{(I)} = n_6^* = 258$, $n_7^{(I)} = n_7^* = 524$; $C_{max}^{(I)} = C_{max}^* = 1065$.

Example 2 (integer): $n = 1{,}000, S = 75, m = 20$. Given the optimal solution of the relaxed version we obtain $\Delta = 2$. Thus, the size of two batches is rounded up and that of the remaining (2 batches) is rounded down. Hence, $n_1^* = 12, n_2^* = 99, n_3^* = 271, n_4^* = 618$; $C_{max}^* = 1387$.

Example 3 (integer): $n = 100{,}000, S = 8, m = 20$. We obtain $\Delta = 5$. Thus, 5 batch sizes are rounded up and 8 batch sizes are rounded down. Hence, $n_1^* = 5, n_2^* = 17, n_3^* = 41, n_4^* = 90, n_5^* = 188, n_6^* = 383, n_7^* = 774, n_8^* = 1{,}556, n_9^* = 3{,}120, n_{10}^* = 6{,}249, n_{11}^* = 12{,}506, n_{12}^* = 25{,}021, n_{13}^* = 50{,}050$; $C_{max}^* = 110{,}117$.

5. Conclusion and future research

We solved a makespan minimization problem on a 2-stage flexible flowshop with m parallel identical machines in stage 1 and a single machine in stage 2. We considered the option of batching (each first-stage machine processes a single batch), and focused on the special case of identical jobs. The paper introduces an efficient solution algorithm, which provides answers to the following questions: (i) the optimal number of first-stage machines to be used, (ii), the batch sizes, and (iii) the optimal schedule of the batches. The running time of the algorithm is independent of the number of jobs, and is linear with the number of machines.

Future research may focus on the extension to general job processing times and/or to more general (not necessarily 2-stage) flowshops.

References

[1] Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y., A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, **187**, 2008, 985-1032.

[2] Baptiste, P., Brucker, P., Scheduling equal processing time jobs. In: Leung J.Y.T. (Ed.), Handbook of Scheduling: Algorithms, Models, and Performance Analysis, Chapman & HALL/CRC, 2004.

[3] Cao, D., Chen, M., Wan, G., Parallel machine selection and job scheduling to minimize machine cost and job tardiness, *Computers & Operations Research*, **32**, 2005, 1995–2012.
[4] Chen, Z.L., Li, C.L., Scheduling with subcontracting options. *IIE Transactions*, **40**, 2008, 1171–1184.
[5] Cheng, T.C.E., Kovalyov, M.Y., An exact algorithm for batching and scheduling two part types in a mixed shop: A technical note, *International Journal of Production Economics*, **55**, 1998, 53-56.
[6] Cheng, T.C.E., Kovalyov, M.Y., Chakhlevich, K.N., Batching in a two-stage flowshop with dedicated machines in second stage. *IIE Transactions*, **36**, 2004, 87-93.
[7] Fanjul-Peyro, L., Ruiz, R., Scheduling unrelated parallel machines with optional machines and jobs selection, *Computers and Operational Research*, **39**, 2012, 1745–1753.
[8] Finke, G., Lemaire, P., Proth, J. M., Queyranne, M., Minimizing the number of machines for minimum length schedules, *European Journal of Operational Research*, **199**, 2009, 702–705.
[9] Gerstl, E., Mosheiov, G., A two-stage flow shop scheduling with a critical machine and batch availability, *Foundations of Computing and Decision Sciences*, **37**, 2012, 39-56.
[10] Gerstl, E., Mosheiov, G., The optimal number of used machines in a two-stage flexible flowshop scheduling problem, *Journal of Scheduling*, 2013, DOI 10.1007/s10951-013-0343-z.
[11] Hoogeveen J.A., Lenstra, J.K., Veltman, B., Preemptive scheduling in a two-stage multi-processor flow-shop is NP-hard, *European Journal of Operational Research*, **89**, 1996, 172-175.
[12] Kravchenko, S.A., Werner, F., Minimizing the number of machines for scheduling jobs with equal processing times, *European Journal of Operational Research*, **199**, 2009, 595–600.
[13] Kravchenko, S.A., Werner, F., Parallel machine problems with equal processing times: a survey, *Journal of Scheduling*, **14**, 2011, 435-444.
[14] Kyparisis, G.J., Koulamas, C., Flow shop and open shop scheduling with a critical machine and two operations per job, *European Journal of Operational Research*, **127**, 2000, 120-125.
[15] Lee C.-Y., Vairaktarakis, G.L., Performance comparison of some classes of flexible flowshops and jobshops, *The International Journal of Flexible Manufacturing Systems*, **10**, 1998, 379-405.
[16] Lin, M.T.B., The strong NP-hardness of two-stage flowshop scheduling with common second-stage machine. *Computers and Operations Research*, **26**, 1999, 695-69.
[17] Lin, H.–T., Liao, C.–J., A case study in a two-stage hybrid flow shop with setup time and dedicated machines, *International Journal of Production Economics*, **86**, 2003, 133-143.
[18] Monma, C.L., Potts, C.N., On the complexity of scheduling with batch setup times. *Operations Research*, **37**, 1989, 789-804.
[19] Mosheiov, G., Oron, D., A note on flow-shop and job-shop batch scheduling with identical processing-time jobs, *European Journal of Operational Research*, **161**, 2005, 285-291.
[20] Mosheiov, G., Oron, D., Ritov, Y., Minimizing flowtime on a single machine with integer batch sizes, *Operations Research Letters*, **33**, 2005, 497-501.

[21] Mosheiov, G., Yovel, U., Comments on "Flow shop and open shop scheduling with critical machine and two operations per job", *European Journal of Operational Research*, **157**, 2004, 257-261.
[22] Oguz, C., Ercan, M.F., A Genetic Algorithm for Hybrid Flow-Shop Scheduling with Multiprocessor Tasks, *Journal of Scheduling*, **8**, 2005, 323-351.
[23] Oguz, C., Lin, M.T.B., Cheng, T. C. E., Two-stage scheduling with a common second-stage machine, *Computers and Operations Research*, **24**, 1997, 1169-1174.
[24] Pinedo, M., Scheduling: Theory, Algorithms, and Systems. Prentice-Hall, 1995.
[25] Ruiz, R., Vazquez-Rodriguez, J.A., The hybrid flow shop scheduling problem, *European Journal of Operational Research*, **205**, 2010, 1-18.

Received October, 2013