Hindawi Publishing Corporation
Advances in Operations Research
Volume 2009, Article ID 153910, 10 pages
doi:10.1155/2009/153910

Research Article

Batch Scheduling on Two-Machine Flowshop with Machine-Dependent Setup Times

Lika Ben-Dati,[1] Gur Mosheiov,[1,2] and Daniel Oron[3]

[1] *Department of Statistics, The Hebrew University of Jerusalem, Jerusalem 91905, Israel*
[2] *School of Business Administration, The Hebrew University of Jerusalem, Jerusalem 91905, Israel*
[3] *Faculty of Economics and Business, The University of Sydney, NSW 2006, Australia*

Correspondence should be addressed to Daniel Oron, d.oron@econ.usyd.edu.au

Received 19 August 2008; Revised 21 April 2009; Accepted 23 June 2009

Recommended by George Steiner

We study a batch scheduling problem on a 2-machine flowshop. We assume unit processing time jobs, batch availability, and machine-dependent setup times. The objective is to find a job allocation to batches of integer size and a batch schedule that minimize makespan. We introduce a very efficient closed form solution for the problem.

1. Introduction

The recent survey paper "A Survey on Scheduling Problems with Setup Times and Costs" [1], classifies batch scheduling problems into those with batching and nonbatching considerations, with sequence-independent and sequence-dependent setup times, and according to the machine environment and the objective function. The survey contains approximately 300 references, a clear indication of the importance and relevance of this topic.

In this paper we study batch scheduling problems on a 2-machine flowshop. The objective function is minimum makespan. The underlying assumptions in the model studied are as follows.

(i) *Batch availability*: all the jobs are completed when the last job of the batch is completed (and then the entire batch is available for processing on the second machine, or for delivery to customers).

(ii) *Nonanticipatory setups*: setups on the second machine may be performed only after processing of the batch on the first machine has been completed.

(iii) *Batch consistency*: batch formation is identical on both machines.

(iv) *Machine-dependent setup times*: machine 1 and machine 2 have different setup times (which are identical for all batches).

The 2-machine flow-shop under the above assumptions was shown to be strongly NP-hard by Cheng et al. [2], and Glass et al. [3]. We focus, however, on an important special case, which is shown to have an elegant closed form solution. Specifically, we assume *unit processing time jobs*. This setting is known to have various applications, and in the manufacturing context, for example, it reflects the common systems in which sequences of identical items are produced. Scheduling with unit jobs has been studied extensively in many scheduling contexts. Batch scheduling on flowshops with unit processing time jobs have been studied by Mosheiov and Oron [4, 5], Ng and Kovalyov [6], and Mosheiov, et al. [7].

In all the above papers, the authors consider identical (i.e., machine-independent) setup times. Our study, as mentioned, focuses on *machine-dependent* setups. Scheduling problems containing machine-dependent setups exist in numerous real-life applications, and have been extensively studied in various contexts. It appears, however, that a model assuming both unit jobs and machine-dependent setups is considered in this paper for the first time. Possible applications for such a setting, when similar or identical operations, are performed on the jobs on the different machines, but an additional operation requiring constant time must be performed on one of the machines (or different constant time operations are performed on the different machines). These operations may be viewed as batching operations, where all jobs of a given batch undergo some type of process which is performed in constant time, independent of the number of jobs in the batch. Such operations may include special loading or recording of arriving batches on the first machine, initial quality assurance, or special preparation of batches prior to processing. Constant time operations on the last machine include, again, preparation of batches for shipment and dispatching of batches (stamping, spray painting), quality assurance procedures (random testing of a fixed number of items or batching procedures such as scanning), or administrative operations which may include paperwork, recording of data, and so forth.

We distinguish between the following two (similar but nonsymmetric) cases: (*i*) the setup on the first machine is larger than the setup on the second machine, and, (*ii*) the setup on the first machine is smaller than the setup on the second machine. For both cases, we show that the makespan minimization problem is solved efficiently in $O(\sqrt{n})$ time, where n is the number of jobs. It is important to note that the existing algorithms solve a special case of the problem studied in this paper, namely, when $s_1 = s_2$, in $O(n)$ time. Thus, the algorithm presented in this paper reduces the computational effort required to obtain an optimal solution even if the problem is more complex. It should be noted that both the existing algorithms and the current solution procedure are not polynomial in the problem input size. Since we consider unit processing time jobs, the input consists of three values ($O(1)$), the number of jobs, and the setup time on each of the two machines. Thus, even a procedure requiring $O(\sqrt{n})$ time is exponential in the input size. Nevertheless, we doubt that there is a way to express the optimal solution to this problem more efficiently.

It should be noted that a first step in the solution consists of solving the "relaxed version" in which integrality of the batch sizes is not required (see below). This, in fact, is a *lot streaming* problem, as it consists of splitting given jobs into sublots in order to allow their overlapping processing on the two successive machines of the flowshop. Chen and Steiner [8, 9], and Liu [10] addressed lot streaming problems on flow-shops with batches of integer size. Vickson [11] addressed lot streaming of a flow-shop with machine-dependent setup times. However, to the best of our knowledge, the model considered in this paper (assuming machine-dependent setups, and requiring integrality of batch sizes), has not been studied before.

In Section 2, we present the notation and formulation of the problem. The optimal solution is provided in Section 3 (and the appendix). Section 4 contains concluding remarks and ideas for future research.

2. Formulation

n independent jobs are available for processing on a 2-machine flowshop at time zero. p_{ij} denotes the processing time of job j ($j = 1, 2, \ldots, n$) on machine i ($i = 1, 2$). We assume unit processing times, therefore $p_{ij} = 1$ for $j = 1, 2, \ldots, n$ and for $i = 1,2$. The scheduler's task is to partition jobs into batches (i.e., find their optimal number and size), and to schedule the batches so as to minimize makespan.

Prior to processing a new batch, an (integer) machine dependent setup time, s_i, $i = 1, 2$, is incurred. As mentioned above, setups are assumed to be *nonanticipatory*. For a given allocation to batches, let k denote the number of batches, and let n_j denote the size of the jth batch. The total processing time of batch j is clearly equal to its size. We assume (see above) *batch-availability* and *batch consistency*. C_j denotes the completion time of batch j, which is the completion time of all the jobs contained in batch j. Using the conventional three field notation, the problem studied in this paper is $F2/s\text{-}batch,\ p_{ij} = 1,\ s_i/C_{\max}$.

Comment 1. A standard assumption is that all input parameters are nonnegative integers, implying in our case that both the (identical) processing times and the (machine-dependent) setups are integers. One can assume that after appropriate scaling all processing times have unit time, but clearly the setups do not necessarily remain integers. Thus, we do allow and investigate the case of noninteger setup times; see Comment 3 in Section 3 and Comment A.1 in the appendix. (Note that the computational effort for this more general case remains $O(\sqrt{n})$ through the use of the proposed algorithm.)

3. A Closed Form Solution for $F2/p_{ij} = 1, s_i/C_{\max}$

We focus here on the case that the setup time on the first machine is smaller, that is $s_1 < s_2$. (The case $s_1 > s_2$, although not completely symmetric, is similar and its analysis appears in the appendix. The case of machine independent setups, that is, when $s_1 = s_2$, is studied in [4, 6].) For convenience we begin by solving the *relaxed* version of the problem (denoted by P_R), in which batch sizes are allowed to have noninteger values.

First, we introduce a lower bound on the optimal makespan for a given number of batches, k. As in the classical 2-machine flow-shop problem, a lower bound is obtained when no idle time is incurred between consecutive batches on the second machine. Given the unavoidable idle time prior to the processing of the first batch on the second machine, we obtain

$$\mathrm{LB} = s_1 + n_1 + ks_2 + n. \tag{3.1}$$

A schedule attaining this lower bound must contain no idle time between consecutive jobs on the second machine. A necessary and sufficient condition for no idle time is the following: $n_{j+1} \le n_1 + j(s_2 - s_1),\ j = 1, \ldots, k-1$. This is true since the completion time of batch $j+1$ on the first machine ($= (j+1)s_1 + n_1 + n_2 + \cdots + n_{j+1}$) cannot exceed the completion time of batch j on the second machine ($= s_1 + n_1 + js_2 + n_1 + n_2 + \cdots + n_j$). One schedule

satisfying these conditions consists of batch sizes obtained by the following set of *equalities*: $n_{j+1} = n_1 + j(s_2 - s_1),\ j = 1, \ldots, k-1$. The resulting schedule consists of the following sequence of increasing batch sizes: $n_{j+1} = n_j + (s_2 - s_1),\ j = 1, \ldots, k-1$. Given the fact that $\sum_{j=1}^{k} n_j = n$, we easily obtain that

$$n_1 = \frac{n}{k} - \frac{k-1}{2}(s_2 - s_1), \qquad n_{j+1} = n_j + (s_2 - s_1), \quad j = 1, \ldots, k-1. \tag{3.2}$$

The above schedule has a makespan value which is equal to the lower bound, and is therefore optimal for problem P_R and a given k value.

The makespan value of the above schedule is

$$C_{\max}(k) = s_1 + \frac{n}{k} - \frac{k-1}{2}(s_2 - s_1) + ks_2 + n = \frac{n}{k} + n + \frac{k+1}{2}(s_1 + s_2). \tag{3.3}$$

Note that (3.3) is a strictly convex function in k and the unique minimum is given by

$$k^* = \sqrt{\frac{2n}{s_1 + s_2}}. \tag{3.4}$$

Recall that k denotes the optimal (integer) number of batches, and therefore, due to the convexity of(3.3)

$$k = \lceil k^* \rceil \quad \text{or} \quad k = \lfloor k^* \rfloor. \tag{3.5}$$

Both $k = \lceil k^* \rceil$ and $k = \lfloor k^* \rfloor$ should be considered.

We conclude that an optimal solution for the relaxed version of the problem (P_R) is given by (3.4), (3.5), and (3.2).

We now consider the original problem, where batch sizes are restricted to be integers, denoted byP_{INT}. A lower bound on the optimal solution is

$$\text{LB}(k) = \left\lceil s_1 + \frac{n}{k} - \frac{k-1}{2}(s_2 - s_1) + n + ks_2 \right\rceil = s_1 + \left\lceil \frac{n}{k} - \frac{k-1}{2}(s_2 - s_1) \right\rceil + n + ks_2. \tag{3.6}$$

(Note that as defined above n_1 is the size of the first batch of the solution of relaxed version. In the special case that $n_1 = n/k - ((k-1)/2)(s_2 - s_1)$ is integer, then, clearly, the "ceiling" is redundant, and the lower bound is identical to the makespan value given in (3.3).)

The lower bound given in (3.6) is the smallest integer which is larger than or equal to the makespan value for problem P_R. Any schedule that consists of batches of *integer size* yielding this makespan is clearly optimal.

If n_1 (given in (3.2)) is integer, then clearly all batches are integers and the solution for problem P_R is optimal for the original problem, P_{INT}. For the general case, where n_1 is not necessarily integer, let $\Delta = n_1 - \lfloor n_1 \rfloor\ (= n_i - \lfloor n_i \rfloor,\ i = 1, \ldots, k)$.

Since $n = \sum_{i=1}^{k} n_i = \sum_{i=1}^{k} (\lfloor n_i \rfloor + \Delta) = \sum_{i=1}^{k} \lfloor n_i \rfloor + k\Delta$, we obtain that $k\Delta$ is an integer (strictly smaller than k). Based on the solution for problem P_R, we construct a solution

for problem P_{INT} by rounding down the size of l batches and rounding up the size of the remaining $k-l$ batches, where $l=(1-\Delta)k$. One option is to round up the size of the first $k-l$ batches, and round down the size of the last l batches. The resulting job allocation to batches is $\lceil n_1\rceil, \lceil n_2\rceil, \ldots, \lceil n_{k-l}\rceil, \lfloor n_{k-l+1}\rfloor, \lfloor n_{k-l+2}\rfloor, \ldots, \lfloor n_k\rfloor$. It is easily verified that in the above schedule: (i) the idle time prior to batch 1 on machine 2 is $s_1+\lceil n_1\rceil = s_1+\lceil n/k-((k-1)/2)(s_2-s_1)\rceil$, and (ii) there is no idle time between consecutive batches on machine 2. Thus, the makespan value of this schedule is

$$C_{\max}(k)=s_1+\left\lceil \frac{n}{k}-\frac{k-1}{2}(s_2-s_1)\right\rceil+n+ks_2\ . \tag{3.7}$$

Note that the above value for the makespan is identical to the lower bound for problem P_{INT} given in (3.6). We conclude that for any value of k, the above schedule is optimal.

Comment 2. It is worth mentioning that the optimal solution is not unique, and other rounding procedures may lead to other optimal schedules. One alternative example consists of rounding down the first l batches and rounding up the last $k-l$ batches.

The resulting job allocation to batches is $\lfloor n_1\rfloor, \lfloor n_2\rfloor, \ldots, \lfloor n_l\rfloor, \lceil n_{l+1}\rceil, \lceil n_{l+2}\rceil, \ldots, \lceil n_k\rceil$.

In this case, it is easily verified that (i) there is no idle time between consecutive batches on machine 2 for $j=1,2,\ldots,l$, (ii) there exists 1 unit of idle time between batches l and $l+1$, and (iii) there is no idle time between consecutive batches on machine 2 for $j=l+1,\ldots,k$. The makespan value of this schedule (clearly, given that n_1 is not integer) is

$$C_{\max}(k)=\lfloor n_1\rfloor+s_1+n+ks_2+1=\left\lfloor \frac{n}{k}-\frac{k-1}{2}(s_2-s_1)\right\rfloor+s_1+n+ks_2+1. \tag{3.8}$$

This value for the makespan is identical to the value given in (3.7), implying that this schedule is optimal as well.

It remains to find the optimal number of batches, k. Note that due to the integrality of s_1 and s_2, the makespan function (3.7) can be slightly modified as follows:

$$C_{\max}(k)=s_1+\left\lceil \frac{n}{k}-\frac{k-1}{2}(s_2-s_1)\right\rceil+n+ks_2\ =\left\lceil s_1+\frac{n}{k}-\frac{k-1}{2}(s_2-s_1)+n+ks_2\right\rceil. \tag{3.9}$$

It follows that the k value minimizing (3.7) also minimizes (3.9). Hence the optimal k value for the relaxed version given in (3.4) and (3.5) is optimal for the integer version. We denote the latter by k^{opt}. A formal algorithm is provided in Algorithm 3.1 .

Algorithm 3.1 (flowshop_makespan_$s_1<s_2$). Input: $n,\ s_1,\ s_2$.

Step 1 (Optimal number of batches). Calculate $k_1=\lceil k^*\rceil$and $k_2=\lfloor k^*\rfloor$ (from (3.4) and (3.5)).

If $C_{\max}(k_1)\le C_{\max}(k_2)$ (in (3.7)), then $k^{\text{opt}}=k_1$; otherwise $k^{\text{opt}}=k_2$.

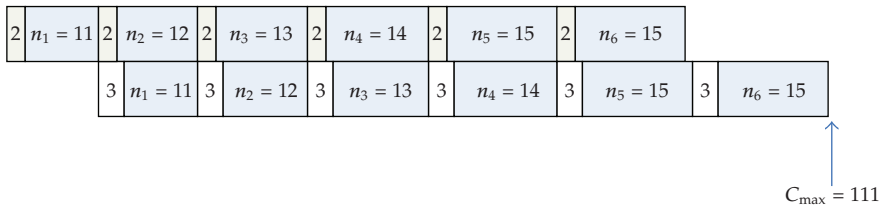

Figure 1: Optimal schedule of a 2-machine flowshop (Example 3.2: $n = 80, s_1 < s_2, s_1 = 2, s_2 = 3$).

Step 2 (Optimal batches).

$$n_1 = \frac{n}{k^{\text{opt}}} - \frac{k^{\text{opt}} - 1}{2}(s_2 - s_1), \qquad n_{j+1} = n_j + (s_2 - s_1), \quad j = 1, \ldots, k^{\text{opt}} - 1 \tag{3.10}$$

(from (3.2)).

Step 3 (Rounding). Calculate the non-integer part of the batch size, $\Delta = n_1 - \lfloor n_1 \rfloor$.

Calculate the number of batches to be rounded down, $l = (1 - \Delta)k^{\text{opt}}$.

An optimal integer allocation to batches is $\lceil n_1 \rceil, \lceil n_2 \rceil, \ldots, \lceil n_{k^{\text{opt}}-l} \rceil, \lfloor n_{k^{\text{opt}}-l+1} \rfloor, \lfloor n_{k^{\text{opt}}-l+2} \rfloor, \ldots, \lfloor n_{k^{\text{opt}}} \rfloor$.

Running Time

It is easily verified that the algorithm is performed in $O(\sqrt{n})$ time. While the calculation of the optimal number of batches and the optimal makespan are performed in constant time, Step 2 of the algorithm requires calculating the batch sizes. Since the number of batches is $O(\sqrt{n})$, the computational effort required is $O(\sqrt{n})$. Step 3 can easily be performed in $O(\sqrt{n})$ time, and consequently, the total running time Algorithm 3.1is $O(\sqrt{n})$.

The use of Algorithm 3.1 is demonstrated in the following example.

Example 3.2. Assume $n = 80$, and $s_1 = 2$, $s_2 = 3$. In Step 1 we calculate $k^* = 5.66$, and $k^{\text{opt}} = 6$, (the associate optimal makespan value is 111). In Step 2 we obtain the batch sizes (for the relaxed problem): $n_1 = 10.833, n_2 = 11.833, n_3 = 12.833, n_4 = 13.833, n_5 = 14.833$, and $n_6 = 15.833$. The rounding procedure (Step 3) leads to the following optimal sequence of batch sizes: $n_1 = 11, n_2 = 12, n_3 = 13, n_4 = 14, n_5 = 15$, and $n_6 = 15$; see Figure 1. The makespan as a function of the number of batches (both for the relaxed and for the integer versions) is presented in Figure 2 .

Comment 3. Consider the interesting generalization to the case of *non-integer* setup times. In this case, the optimal k value for (3.7) may be different from that of (3.3), (e.g., if $n = 80$, $s_1 = 2.1$ and $s_2 = 2.2$, we obtain $k^* = 6.099$, implying that the optimal k value for the relaxed problem is either 6 or 7, however, the optimal k value for the integer version is 5.) It appears that in this general case (non-integer setups), a closed form expression for the optimal k value does not seem to exist, and a search must be performed. Clearly, a search over all possible k values (1 through n) guarantees an optimal solution. However, a limited search appears to be sufficient, due to the fact that the smallest batch n_1 must be nonnegative. From (3.2)

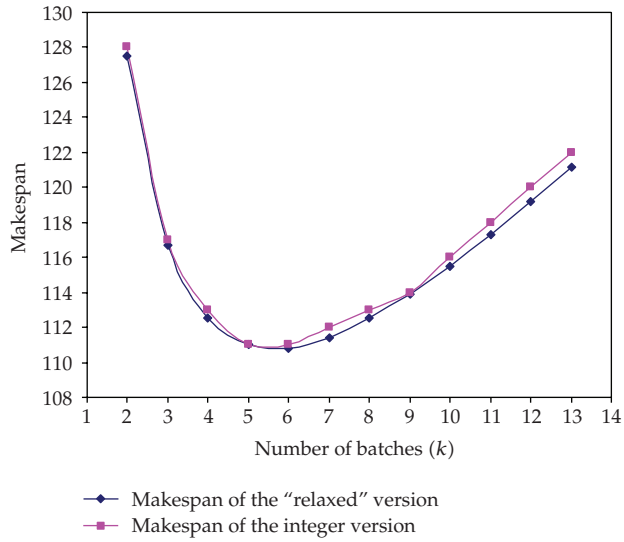

Figure 2: Makespan as a function of the number of batches for the relaxed and the integer versions (Example 3.2: $n = 80, s_1 < s_2, s_1 = 2, s_2 = 3$).

we obtain $n_1 = n/k - ((k-1)/2)(s_2 - s_1) \geq 0$. Consequently, $1/2 - \sqrt{1/4 + 2n/(s_2 - s_1)} \leq k \leq 1/2 + \sqrt{1/4 + 2n/(s_2 - s_1)}$. The first term is strictly negative (recall that $s_2 > s_1$), implying that k is an integer in the range $1, \ldots, m$, where

$$m = \left\lfloor \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{2n}{s_2 - s_1}} \right\rfloor. \tag{3.11}$$

We conclude that despite the need to consider a range of possible values for the number of batches, k, the computational effort required for solving the problem with non-integer setup times remains $O(\sqrt{n})$.

Comment 4. It is easily verified that the algorithm proposed in this section generalizes the algorithm introduced in Mosheiov and Oron [4], for the case of machine-independent setups. When substituting $s_1 = s_2 = s$ in the expressions along the above algorithm, we obtain the optimal solution for this special case. In particular, substituting $s_1 = s_2 = s$ in Step 2 leads to "equal allocation" to batches (using the terminology of [4]), and the rounding procedure, that is, rounding up the size of the first $k - l$ batches and rounding down the size of the remaining l batches generates an optimal integer solution.

4. Conclusion

We study a new version of makespan minimization on a 2-machine flowshop. We consider unit time jobs which may be grouped into batches. The setup times are assumed to be machine-dependent. We introduce an elegant closed form solution for two different cases.

A challenging extension is to the setting of an m-machine flowshop ($m \geq 3$). Another interesting question refers to a 2-machine jobshop setting. Although not published yet,

both problems appear to have similar properties and structures and may have closed form solutions for the number of batches and their sizes. The m-machine jobshop case is known to be strongly NP-hard even with no setups [12]. Thus, future research dealing with this extension may focus either on an introduction of heuristics/approximations or on developing exact (pseudopolynomial) solution algorithms.

Appendix

The Case $s_1 > s_2$

As mentioned, the analysis of the case $s_1 > s_2$ is similar to that of the case $s_1 < s_2$. Again, we focus first on the relaxed version. It is easily verified that lower bound (3.1) for a given number of batches k, still holds. As in the previous case, a schedule attaining this lower bound must have no idle time between consecutive jobs on the second machine. The conditions $n_{j+1} \leq n_1 + j(s_2 - s_1),\ j = 1, \ldots, k-1$ are still valid, and since $s_1 > s_2$, these could be written as $n_{j+1} \leq n_1 - j(s_1 - s_2),\ j = 1, \ldots, k-1$. A schedule based on equalities clearly satisfies these conditions, leading, again, to a constant difference between the size of consecutive batches: $n_{j+1} = n_j - (s_1 - s_2),\ j = 1, \ldots, k-1$. Since the total size of all the batches is n, we obtain the following schedule of decreasing batch sizes:

$$n_1 = \frac{n}{k} + \frac{k-1}{2}(s_1 - s_2), \qquad n_{j+1} = n_j - (s_1 - s_2), \quad j = 1, \ldots, k-1. \tag{A.1}$$

Note that the resulting schedule is identical to the one obtained for the case $s_1 < s_2$, in a reversed order: the largest batch becomes first, the second-to-largest second, and so forth. This schedule has a makespan value identical to the lower bound (3.1), and is therefore optimal for the relaxed problem and a given k value.

The makespan value is given by

$$C_{\max}(k) = s_1 + \frac{n}{k} + \frac{k-1}{2}(s_1 - s_2) + ks_2 + n = \frac{n}{k} + n + \frac{k+1}{2}(s_1 + s_2). \tag{A.2}$$

(Note that (A.2) is identical to the makespan value (3.3) obtained earlier.) Since the function (A.2) is strictly convex, the optimal (integer) k-value is obtained by a simple differentiation and rounding to one of the two nearest integers (see (3.4) and (3.5)). Given this optimal solution for the relaxed problem, we consider again the integer version. We suggest the rounding procedure introduced earlier, that is, round up the first k-l batches, and round down the remaining l batches (where $l = (1 - \Delta)k$, and $\Delta = n_1 - \lfloor n_1 \rfloor$). This schedule of integer batch sizes is optimal since its makespan value (given by (3.7)) is identical to the lower bound (3.6). Our last argument refers to the optimal number of integer batches. As in the previous case, due to the integrality of s_1 and s_2, the (integer) makespan value is given by

$$C_{\max}(k) = \left\lceil s_1 + \frac{n}{k} + \frac{k-1}{2}(s_1 - s_2) + n + ks_2 \right\rceil, \tag{A.3}$$

which is minimized by the same k value that minimizes the objective of the relaxed version (A.2). Hence the optimal number of batches is given, as before, by (3.4) and (3.5).

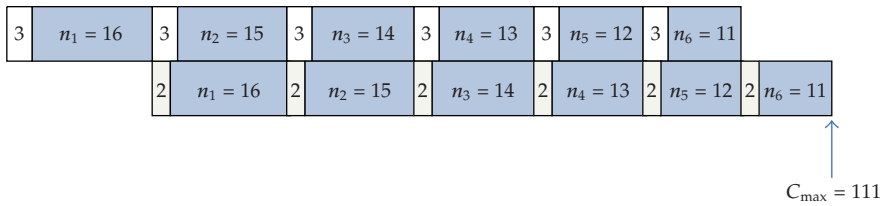

Figure 3: Optimal schedule of a 2-machine flowshop (Example A.1: $n = 80, s_1 > s_2, s_1 = 3, s_2 = 2$).

We conclude that an optimal solution for the case $s_1 > s_2$ is obtained by a very similar (constant time) algorithm to the one introduced for the case $s_1 < s_2$. (Step 2 is slightly different due to the fact that the sequence of batch sizes is decreasing.) For the sake of briefness, we omit the formal algorithm.

Example A.1. We replace the values of s_1 and s_2 in Example 3.2: $n = 80$, $s_1 = 3$, $s_2 = 2$. The optimal number of batches (Step 1) remains 6. In Step 2 we obtain the batch sizes (for the relaxed problem): $n_1 = 15.833$, $n_2 = 14.833$, $n_3 = 13.833$, $n_4 = 12.833$, $n_5 = 11.833$, and $n_6 = 10.833$. (Note that this sequence is the reversed sequence obtained for the relaxed problem in Example 3.2.) An optimal integer solution (Step 3) consists of the following sequence of batch sizes: $n_1 = 16$, $n_2 = 15$, $n_3 = 14$, $n_4 = 13$, $n_5 = 12$, $n_6 = 10$; see Figure 3.

Comment A.1. In the case where the setup times are not necessarily integers, finding the optimal number of batches requires a search. Using the fact that the size of the smallest batch (now $n_k = n/k - ((k-1)/2)(s_1 - s_2)$) must be non-negative, we obtain (as in the previous case) the following upper bound on k: $m = \lfloor 1/2 + \sqrt{1/4 + 2n/(s_1 - s_2)} \rfloor$. Since the search is limited to the integers in the interval $[1,m]$, the total running time in this case becomes $O(\sqrt{n})$.

References

[1] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 985–1032, 2008.
[2] T. C. E. Cheng, B. M. T. Lin, and A. Toker, "Makespan minimization in the two machine flowshop batch scheduling problem," *Naval Research Logistics*, vol. 47, no. 2, pp. 128–144, 2000.
[3] C. A. Glass, C. N. Potts, and V. A. Strusevich, "Scheduling batches with sequential job processing for two machine flow and open shops," *INFORMS Journal on Computing*, vol. 13, no. 2, pp. 120–137, 2001.
[4] G. Mosheiov and D. Oron, "A note on flow-shop and job-shop batch scheduling with identical processing-time jobs," *European Journal of Operational Research*, vol. 161, no. 1, pp. 285–291, 2005.
[5] G. Mosheiov and D. Oron, "Open-shop batch scheduling with identical jobs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 1282–1292, 2008.
[6] C. T. Ng and M. Y. Kovalyov, "Batching and scheduling in a multimachine flow shop," *Journal of Scheduling*, vol. 10, no. 6, pp. 353–364, 2007.
[7] G. Mosheiov, D. Oron, and Y. Ritov, "Flow-shop batch scheduling with identical processing time jobs," *Naval Research Logistics*, vol. 51, no. 6, pp. 783–799, 2004.
[8] J. Chen and G. Steiner, "Approximation methods for discrete lot streaming in flow shops," *Operations Research Letters*, vol. 21, no. 3, pp. 139–145, 1997.
[9] J. Chen and G. Steiner, "Discrete lot streaming in two-machine flow shops," *INFOR Journal*, vol. 37, no. 2, pp. 160–173, 1999.
[10] S. C. Liu, "A heuristic method for discrete lot-streaming with variable sublots in a flow-shop," *International Journal of Advanced Manufacturing Technology*, vol. 22, no. 9-10, pp. 662–668, 2003.

[11] R. G. Vickson, "Optimal lot streaming for multiple products in a two-machine flow shop," *European Journal of Operational Research*, vol. 85, no. 3, pp. 556–575, 1995.
[12] J. K. Lenstra and A. H. G. Rinnooy Kan, "Computational complexity of discrete optimization problems," *Annals of Discrete Mathematics*, vol. 4, pp. 121–140, 1979.