IIE Transactions (2002) **34**, 953–970

Lot splitting to minimize average flow-time in a two-machine flow-shop

JOSEPH BUKCHIN[1], MICHAL TZUR[1] and MICHAL JAFFE[2]

[1]*Industrial Engineering Department, Tel Aviv University, Ramat Aviv, Tel Aviv, 69978 Israel*
E-mail: bukchin@eng.tau.ac.il or tzur@eng.tau.ac.il
[2]*EL AL Israel Airlines Ltd., Ben Gurion Airport, 70100 Israel*
E-mail: michaly0@netvision.net.il

Received July 1999 and accepted October 2001

Lot splitting is a technique for accelerating the flow of work by splitting job lots into sublots. In this paper we investigate the lot splitting scheduling problem in a two-machine flow-shop environment with detached setups and with batch availability. The performance measure considered is the average flow-time which is indicative of the increasingly important manufacturing lead-time. Our contribution is both theoretic and practical for the case of general (not necessarily equal) sublots. We identify properties of the optimal solution and develop a solution procedure to solve the problem. We then present a computational study which indicates that our solution technique is very efficient.

1. Introduction and literature review

Traditional scheduling problems typically assume that lot sizes are given from upper level production planning. The problem then becomes how to determine the order in which jobs are processed (see for example, Conway *et al.* (1967), Baker (1974), and Pinedo (1995)). The lot-sizing problem which is solved in the first stage does not take into account operational performance measures such as flow-time. Based on the lot-sizing decisions a lot is transferred from one-machine to the next only when all items in the lot have completed their processing. Thus, when expensive setups induce large lots, an item may spend most of the time waiting to be processed or waiting for the rest of its lot to be processed.

Lot splitting is a technique for accelerating the flow of work by splitting job lots into sublots (Kropp and Smunt, 1990). This technique improves the overall performance of the production system by releasing all items included in a sublot upon its completion. When only one-machine is involved, the flow-times of these items become shorter than in the single lot case, at least for the early sublots. If more than one-machine is involved, this allows the items in a sublot to proceed to the next machine even before all items have completed their processing on the current machine, thus creating overlapping consecutive operations and releasing items sooner.

The vast majority of research on lot splitting has been concerned with the makespan objective function and is often referred to as the *lot streaming* problem (see Trietsch and Baker (1993), Baker and Jia (1993) and many others). Other terminology that is used especially in the one-machine case is *batching* (Dobson *et al.* 1987; Santos and Magazine, 1985; Naddef and Santos, 1988). The problem is typically how to split a single lot into sublots given various other considerations such as the existence of a setup at the beginning of each sublot, or possible restrictions on the number and the size of the sublots.

In this paper we investigate the lot splitting problem with the performance measure of *average flow-time* which is indicative of the increasingly important manufacturing lead-time. We consider a two-machine flow-shop environment, where a setup time is attached to each sublot and both variable unit processing times, as well as the sublot setup times, are machine-dependent. We consider the case of general (not necessarily equal) sublots/batches under the assumption of batch availability. This problem was not considered before. We provide a theoretical contribution to the problem on hand in the two machines framework which may be used in future research as a building block for analyzing and solving problems of multiple products and more than two machines. A practical contribution arises from it in two ways: one is by enhancing managerial intuition and the other is by considering cases in which planning is carried out hierarchically.

We first identify properties of the optimal solution. We then develop a solution procedure to solve the problem. For some parameter combinations of the problem this solution procedure provides an optimal solution,

otherwise the suggested procedure becomes a heuristic method. We then present a computational study that indicates that our solution technique is very efficient. Finally, in the second part of the computational study, we compare the results of our model to the same model, albeit with the makespan objective function. This comparison enhances managerial intuition regarding differences and similarities of using flow-time and makespan objectives.

The motivation for implementing lot splitting techniques is well-addressed in Santos and Magazine (1985). The first motivation is bridging the gap mentioned above between the lot-sizing decision making and the traditional scheduling problems. Another somewhat related motivation is the contribution of these problems to the scheduling/sequencing literature. These problems can be viewed as sequencing problems with processing times that are decision variables.

The third motivation is related to the resulting flow-time reduction. Reducing flow-time is associated with lead-time reduction which has recently become an important goal in industrial environments that produce expensive products. An example for such an environment is the semiconductor industry where a stepper machine is used in a photolithographic process. In this process a setup time is associated with each sublot of wafers which are loaded on the machine. Subsequently, the wafers in the sublot are processed sequentially and hence the total variable process time of a sublot increases linearly with the number of parts in the sublot. In the semiconductor industry, flow-time (also known as 'cycle time') is a very important performance measure and large efforts are made in an attempt to reduce it. For example, in the photolithographic process described above, loading less than the maximum possible number of wafers on the machine may improve average flow-time. To obtain the maximum improvement the quantities to be loaded on the machine at each sublot need to be determined.

Other examples for using lot splitting in industrial environments are given in Cheng *et al.* (1996), where flexible manufacturing systems environments are addressed. In these production systems a wide variety of part types are produced and items are generally mounted on standard pallets while moving from one machining center to another. Again it may not be beneficial to fill a pallet to its full capacity due to the existence of setup times. Then a trade-off exists between producing more sublots, thereby incurring more setup times and producing fewer sublots, which entails longer job completion times. Therefore, a technique to determine the exact sublot sizes is again required.

The nature of the machine setups used in this paper is related to other machine setups mentioned in the literature in the following way: Truscott (1985) distinguishes between two major setups: the first where the setup can be done in advance and when the first item arrives the machine is ready. The other is when the machine requires *one* item to be set up, but items that follow do not require a setup. Chen and Steiner (1996, 1998), who consider makespan minimization use the term *detached* (or anticipatory) setup for the former type and *attached* (or non-anticipatory) setup for the latter, but with a slight variation which requires a *minimal* number of items (usually one) to be available during the setup. Potts and Kovalyov (2000) define an attached or non-anticipatory setup of a batch as a setup that cannot be initiated before *all* jobs of the batch are released and have completed their processing on any previous machine. In our model all three definitions of non-anticipatory setup are satisfied since we assume batch availability, hence all items in a sublot arrive together to the machine. Chen and Steiner (1998) assume that a setup exists only for the first sublot but we assume an attached or non-anticipatory setup at the beginning of every sublot.

Truscott (1986) addresses a resource called *transporter*, which requires the same processing time regardless of the sublot size. Trietsch and Baker (1993) observe that transporters are equivalent to *ovens*, which also require the same processing time regardless of the sublot size. The setup structure we use is of the transporter/oven type. However, unlike the transporter/oven system, we consider in addition to the setup, a variable processing time for each individual item.

As mentioned above, the setup structure we use is appropriate for the semiconductor industry and for other environments where loading a sublot is associated with a setup time in which the machine cannot operate. Another example for this setup structure can be found in a recent paper by Cheng *et al.* (2000), who describe a problem observed by a manufacturer of pneumatic valves, which are produced on a two-stage production line. In the first stage the valves are produced and the second stage is an inspection station. The part-types are mounted on pallets for machining and the rest of the procedure is described as follows: "The part-types mounted on the same pallet are processed together as a batch for their first operation at the machining center and are transferred to the inspection station only when all part-types on the same pallet have finished processing. Thus, the part-types assigned to the same pallet share the same completion time which is called the batch completion time. A setup time is needed to remove a processed pallet and to install a new one on either the machining center or the inspection station." They also mention that: "loading a pallet to its full capacity may not necessarily be beneficial as it may lead to long batch processing times. Hence, the number of part-types to be mounted on the pallet is a decision variable."

Other assumptions in Cheng *et al.* (2000) differ from ours in the following points: (i) they assume identical setup times on both machines; (ii) they use makespan as an objective function; (iii) they consider combining different part-types into a batch, rather than splitting one lot into

sublots. Finally, as in our model, in addition to the above described setup times, there are variable processing times.

There are numerous studies on lot splitting, lot streaming and batching. We complete the literature review by discussing additional references that deal with the flow-time objective function in models which include setups, since this is the part which is most related to our study. The single machine single- and multi-products batching problem is addressed in Santos and Magazine (1985) and Dobson *et al.* (1987), who assume continuous batch (sublot) sizes. Closed-form formulas for the number of batches and the batch sizes are given for the single product case. In Naddef and Santos (1988) and in Shallcross (1992) the single machine problem with integer batch sizes is addressed; the latter concludes that the integer solution resembles the continuous solution.

The multiple machine case is mainly investigated using the makespan objective function and only a few address the flow-time objective function. One type of multiple machine problem is the multiple parallel machines, which is discussed in Dobson *et al.* (1989) and Cheng *et al.* (1996). The former solves the continuous case of a single product problem, finding a closed-form solution for a special case of equal setup times on the machines. The latter discusses the discrete case of the multi-product problem with setups, developing a dynamic programming algorithm to solve the problem.

The lot streaming problem in a two-stage flow-shop environment for minimizing flow-time is discussed in Şen *et al.* (1998) where the number of sublots is given and no setup times are considered. Kropp and Smunt (1990) consider the multi-machine flow-shop where the number of sublots is given and a setup time exists for starting the first sublot only; the problem is solved using a quadratic programming approach. The work of Kalir and Sarin (2001) deals with the lot streaming problem for minimizing the mean flow-time in a flow-shop environment where a setup is attached to every sublot and the sublot sizes are restricted to be equal. A closed-form formula for the number of sublots and the sublot sizes is presented in the continuous case and an algorithm for solving the discrete problem is developed.

The problem we consider in this paper generalizes the current state of the art research on lot splitting from two directions. First, our problem is a generalization of the single machine problem studied in Santos and Magazine (1985) and in Dobson *et al.* (1987). Second, our problem relaxes the restriction of equal sublot sizes imposed in Kalir and Sarin (2001), although their analysis is for the m-machine flow-shop setting.

The rest of the paper is organized as follows: in Section 2 we present the model formulation and discuss some preliminary results. In Section 3 we define and analyze a property called: 'Single Machine Bottleneck' (SMB). In particular, we prove that the property is satisfied in all optimal solutions under various parameter combinations of the problem. In Section 4 we restrict our attention to solutions that satisfy the SMB property and develop a solution procedure that finds the optimal solution in this class. Experiments, including a comparison with the makespan objective function are presented in Section 5, and conclusions are drawn in Section 6.

2. Problem formulation and preliminaries

We summarize here the definitions and assumptions that describe our model:

1. A two-machine flow-shop environment is considered.
2. A *sublot* is defined as the number of parts processed continuously on a machine with a single setup.
3. The sublots are *consistent*, that is sublot sizes are the same on both machines.
4. Sublot sizes are *general*, that is they don't have to be equal.
5. An *attached* or *non-anticipatory* setup time is associated with each sublot, that is a setup time is incurred before the beginning of the processing of each sublot on each machine. This setup is associated with a time when the machine is not working, for example, in order to load the parts on the machine, or in order to set the machine up again.
6. We assume *batch availability*, that is all items in a sublot leave the machine together at the end of the processing of the last item in the sublot. Hence, the flow time of each item is defined as the finish time of the last item of the sublot to which it belongs.
7. We assume that the sublot sizes are *continuous* in order to be able to gain an insight on the problem without being affected by integrality issues. We expect that rounding the continuous solution will be a good approximation to the integral solution, as was concluded in Shallcross (1992) for the single machine problem.

The problem is defined by the following parameters:

d = demand, number of parts waiting to be processed (all parts are identical);

s_m = setup time on machine m prior to processing a sublot (the setup time is independent of the sublot size), $m = 1, 2$;

t_m = processing time (of each part) on machine m, $m = 1, 2$.

The decision variables are:

q_k = size of sublot k, that is the number of parts it contains, $k = 1, \ldots, M$; (As in Dobson *et al.* (1987, 1989), we use M as an upper bound on the number of actual sublots, where only sublots with $q_k > 0$ are counted as actual ones; in practical situations we expect the actual number of sublots to be smaller than the number of units that need to be processed.)

n = the number of actual sublots (i.e., with a positive size); $n \leq M$ and it is given by the highest index of the positive sublots.
The values of the q_k variables determine the flow-time of all parts, as well as the idle time on the machines, which we represent through the following notation:

I_k = cumulative idle time on machine 2, up to the time when the kth sublot finishes its processing on machine 1;
We assume that there exists an infinite buffer between machines 1 and 2, therefore machine 1 is never blocked, operating continuously.

f_k = flow-time of sublot k (ready time of all parts is zero).

The formulation of the problem is described next:

$$\min \sum_{k=1}^{M} f_k q_k, \tag{1}$$

subject to

$$f_k = ks_2 + t_2 \sum_{i=1}^{k} q_i + I_k \quad k = 1, \ldots, M, \tag{2}$$

$$I_1 = s_1 + t_1 q_1, \tag{3}$$

$$I_k = \max\left\{ I_{k-1}, \left(t_1 \sum_{i=1}^{k} q_i + ks_1 \right) - \left(t_2 \sum_{i=1}^{k-1} q_i + (k-1)s_2 \right) \right\}$$
$$k = 2, \ldots, M, \tag{4}$$

$$\sum_{k=1}^{M} q_k = d, \tag{5}$$

$$q_k \geq 0 \quad k = 1, \ldots, M. \tag{6}$$

The objective function describes the total flow-time of all parts in all sublots. Constraint (2) defines the flow-time of sublot k as the processing time of all sublots up to k on machine 2, including setups, plus the cumulative idle time on machine 2 at the end of this sublot. Constraint (3) defines the first idle time on machine 2 as the processing time of the first sublot on machine 1, including setups. Constraint (4) defines the cumulative idle time at the end of sublot k as the maximum between the cumulative idle time at the end of the former sublot and the cumulative idle time that may have been created by the current sublot. The latter is the difference between the processing time of all sublots (including setups) on machine 1, and the process time of all sublots up to the former sublot (including setups) on machine 2. Constraint (5) states that the total number of parts to be processed equals the demand, and finally constraint (6) requires non-negative sublot sizes. We refer to the problem defined in (1)–(6) as *the general problem*.

Constraint (4) may be replaced by two linear constraints, however the formulation remains non-linear due to the objective function. Another important observation regarding the formulation (1)–(6) is that although the flow time f_k is convex in the vector $\boldsymbol{q}$ (since it is the maximum and sum of convex functions), the objective function is not necessarily convex. Only if the number of sublots n is restricted to two, the objective function can be shown to be convex (and the solution is easily obtained in closed-form), but this is no longer true for $n \geq 3$. For $n = 3$, for example, the objective function is combined of four cases; each of these cases represents a combination of the expressions which achieve the maximum in constraint (4) for $k = 2$ and $k = 3$. In other words, each case represents the situation at the completion of the second and third sublots on machine 1, that is: $I_1 = I_2 = I_3$, $I_1 = I_2 < I_3$, $I_1 < I_2 = I_3$ and $I_1 < I_2 < I_3$. A strict inequality between I_{k-1} and I_k implies that the right-hand expression in (4) is larger than the left-hand one. On the other hand, equality between I_{k-1} and I_k occurs when the left-hand expression in (4) is no smaller than the right-hand one. Using some algebra to obtain closed-form expressions for the objective function for each of the cases, one discovers that the expression in the third case is not convex for all parameter values, therefore the entire objective function is not convex in general.

Examining the meaning of the third case ($I_1 < I_2 = I_3$) raises the question whether such a solution is likely to be optimal. Indeed, one would intuitively expect to find in an optimal solution the existence of a *single machine bottleneck* situation (denoted as *SMB* from now on) throughout the production process. We refer to an SMB situation when in Equation (4) the same expression always achieves the maximum. For example, if the first expression in (4) always achieves the maximum, then no new idle time is created on machine 2 as a result of sublot k. Therefore machine 2 is the bottleneck in proceeding from sublot $k-1$ to sublot k (for all $k > 1$) on machine 2. If the second expression in (4) always achieves the maximum, then the reverse situation occurs and machine 1 is the bottleneck throughout the process. (A rigorous definition of the SMB property is given in the next section.)

In the following example we show that the SMB property is not always satisfied in an optimal solution, hence non-convex situations as described above may occur in an optimal solution. We demonstrate it by presenting an instance of the problem and a solution to it, in which not the same expression in (4) achieves the maximum for all sublots. This solution has a better objective value than the optimal solution, which does satisfy the SMB property (in Section 4 we describe how an optimal solution, which satisfies the SMB property can be obtained.)

Example: the parameters are: $d = 50, t_1 = 2.1, t_2 = 1.0$, $s_1 = 26$, $s_2 = 30$. A solution which does not satisfy the SMB property is: $q_k = (22.21, 15.63, 9.35, 2.81)$, and is associated with cumulative idle times: $I_k = (72.64, 79.25, 79.25, 79.25)$ and an objective function value of 8265.13

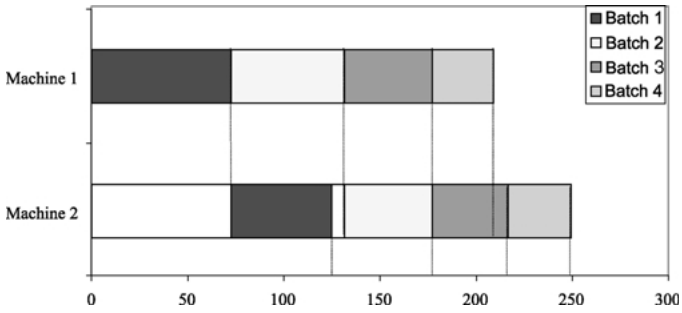

Fig. 1. Shifting bottleneck example.

(see Fig. 1). In this solution, for $k = 2$, 3 and 4, the left-hand expression in (4) is smaller, equal and larger, respectively, than the right-hand expression in (4).

The example was also solved for the two cases in which either machine 1 or machine 2 were a bottleneck machine for all sublots. In the first (second) case, respectively, the right- (left-hand) term in the maximum expression in Equation (4) was constrained to achieve the maximal value for all sublots. The optimal solution when machine 1 (2) was constrained to be a bottleneck, produced a solution with three (four) sublots, where $q_k = (23.00, 16.67, 10.33)$ (and $q_k = (23.21, 12.96, 8.08, 5.75)$), respectively, and the resulting objective value was 8268.46 (8280.96), respectively. We conclude from this example that the SMB property is not necessarily satisfied in an optimal solution.

3. The single machine bottleneck property

In the previous section we showed that the SMB property does not hold in general. However, we prove now that for a restricted set of parameters it is satisfied. In Section 5 a numerical study is performed, providing empirical support for using this property.

Definition 1. *Machine* 1(2) *is a bottleneck at sublot* $k + 1$ *if the completion time of sublot* $k + 1$ *on machine* 1 *occurs no earlier* (*no later*) *than the completion time of sublot* k *on machine* 2.

Definition 2. *Machine* 1(2) *is a unique bottleneck at sublot* $k + 1$ *if the completion time of sublot* $k + 1$ *on machine* 1 *occurs later* (*earlier*) *than the completion time of sublot* k *on machine* 2.

Definition 3. *Machine* 1(2) *is a bottleneck machine, if it is a bottleneck at sublot* $k + 1$, *for all* $k \geq 1$.

Theorem 1. *If* $t_1 \geq t_2$ and $s_1 \geq 2s_2$ *then in all optimal solutions machine* 1 *is a bottleneck machine.*

Proof. Assume in contradiction that there exists an optimal solution in which machine 2 is a unique bottleneck at sublot $k + 1$ for some $k \geq 1$, and that this is the earliest sublot for which this is the case. In particular, this means that machine 1 is a bottleneck at sublot k (possible jointly with machine 2) and therefore the starting time of sublot $k + 1$ on machine 1 and sublot k on machine 2 are identical (see Fig. 2, where w.l.o.g. machine 2 is also a bottleneck at sublot k). Therefore we get:

$$s_2 + q_k t_2 > s_1 + q_{k+1} t_1. \tag{7}$$

We now show that the solution may be improved by transferring $\varepsilon > 0$ units of the product from sublot k to sublot $k + 1$. Note that (7) states that the time required to perform sublot k on machine 2 is larger than the time required to perform sublot $k + 1$ on machine 1; we choose ε to be sufficiently small so that this situation is preserved after the transfer of ε also, hence:

$$s_2 + (q_k - \varepsilon) t_2 > s_1 + (q_{k+1} + \varepsilon) t_1. \tag{8}$$

This transfer does not affect the completion time of sublots $1, \ldots, k - 1$ and the completion times of sublot k and all subsequent sublots remains unchanged or become earlier (in case that machine 2 is not a bottleneck at sublot k). Therefore the total contribution to the objective function of all sublots other than k and $k + 1$ is either unchanged or reduced as a result of this transfer. It remains to examine the change in contribution of sublots k and $k + 1$ to the objective function as a result of the transfer.

Let S be the contribution of sublots k and $k + 1$ to the objective function before transferring ε, and S' is the contribution after transferring ε. Then:

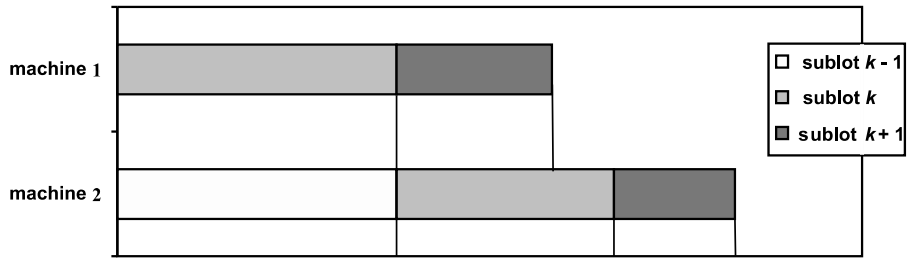

Fig. 2. Machine 2 is a unique bottleneck of sublot $k + 1$ for the first time.

$$S = q_k f_k + q_{k+1} f_{k+1},$$

and

$$S' \le (q_k - \varepsilon)(f_k - \varepsilon t_2) + (q_{k+1} + \varepsilon) f_{k+1}.$$

We now show that $S - S' > 0$, which implies that the solution is improved.
Note that:

$$f_{k+1} - f_k = s_2 + t_2 q_{k+1}, \tag{9}$$

since machine 2 is a unique bottleneck at sublot $k+1$.
Therefore, the difference between the new and the old solution is:

$$S - S' = \varepsilon(-s_2 + q_k t_2 - q_{k+1} t_2 - \varepsilon t_2). \tag{10}$$

From (8) we get:

$$q_k \ge \frac{s_1 - s_2 + q_{k+1} t_1 + \varepsilon t_1}{t_2} + \varepsilon. \tag{11}$$

Using (11) in (10) and we get:

$$S - S' \ge \varepsilon(s_1 - 2s_2 + q_{k+1}(t_1 - t_2) + \varepsilon t_1). \tag{12}$$

Now, given $t_1 \ge t_2$ and $s_1 > 2s_2$, $S - S' > 0$ and the modified solution is better than the original one, a contradiction. Therefore, in none of the sublots it is possible for machine 2 to be a unique bottleneck, concluding our proof. ■

Theorem 2. *If* $t_2 \ge t_1$ *and*

$$s_2 \ge s_1 \times \frac{2t_2}{t_1 + 2t_2},$$

then in all optimal solutions machine 2 *is a bottleneck machine.*

Proof. Assume in contradiction that there exists an optimal solution in which machine 1 is a unique bottleneck at sublot $k+1$ for some $k \ge 1$. Since sublot k cannot start on machine 2 before its completion time on machine 1 (it may start even later than that, as can be observed without loss of generality in Fig. 3), we have: $s_1 + t_1 q_{k+1} > s_2 + t_2 q_k$.

We next show that the objective function can be improved in this case, by transferring $\varepsilon > 0$ units of the product from sublot $k+1$ to sublot k. We choose ε to be sufficiently small so that machine 1 remains a bottleneck at sublot $k+1$. This transfer does not affect the completion time of sublots $1, \ldots, k-1$. The completion times of sublots $k+2$ and all subsequent sublots has not increased as a result of the transfer, and even may have been reduced (see Fig. 3). Therefore, the total contribution to the objective function of all sublots, other than k and $k+1$, has not increased as a result of this transfer. It remains to examine the change in contribution of sublots k and $k+1$ to the objective function.

Let S be the contribution of sublots k and $k+1$ to the objective function before transferring ε, and S' the contribution after the transfer. Then

$$S = q_k f_k + q_{k+1} f_{k+1},$$

and

$$S' \le (q_k + \varepsilon)(f_k + \varepsilon t_1 + \varepsilon t_2) + (q_{k+1} - \varepsilon)(f_{k+1} - \varepsilon t_2). \tag{13}$$

(Inequality (13) will be satisfied as equality if originally sublot k started on machine 2 immediately when it finished on machine 1, in which case the completion time after the transfer becomes: $f_k + \varepsilon t_1 + \varepsilon t_2$. Otherwise it is satisfied as a strict inequality, for example, according to Fig. 3 the completion time after the transfer becomes: $f_k + \varepsilon t_2$.)

We show that the new solution is improved, namely that $S - S' > 0$.

$$S - S' \ge \varepsilon(f_{k+1} - f_k + q_{k+1} t_2 - q_k t_2 - q_k t_1 - \varepsilon(t_1 + 2t_2)). \tag{14}$$

We use now two more inequalities: first, since machine 1 is a unique bottleneck at sublot $k+1$,

$$f_{k+1} - f_k > s_2 + q_{k+1} t_2. \tag{15}$$

Second, since the starting time of sublot $k+1$ on machine 2 has not changed as a result of the transfer of ε (according to our choice of ε that leaves machine 1 a bottleneck at sublot $k+1$), $s_1 + t_1(q_{k+1} - \varepsilon) \ge s_2 + t_2(q_k + \varepsilon)$,

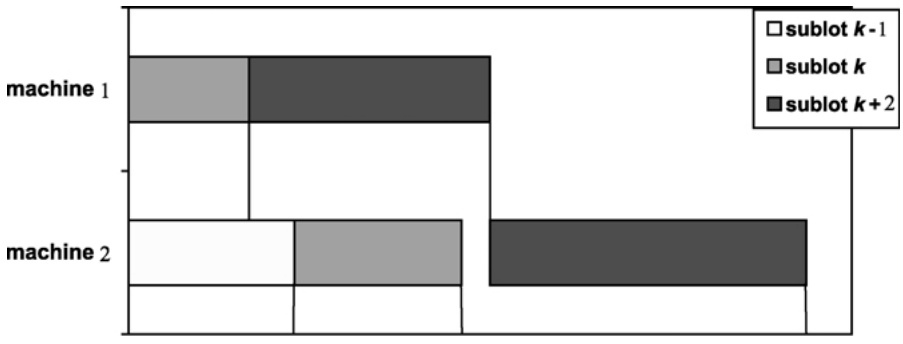

Fig. 3. Machine 1 is a unique bottleneck of sublot $k + 1$.

Therefore

$$q_{k+1} \geq \frac{s_2 - s_1 + \varepsilon t_1 + q_k t_2 + \varepsilon t_2}{t_1}. \quad (16)$$

Now using inequalities (15) and (16) in (14), we get:

$$S - S' > \varepsilon(s_2 + 2(s_2 - s_1 + \varepsilon t_1 + q_k t_2 + \varepsilon t_2)\frac{t_2}{t_1} - q_k t_2 - q_k t_1 - \varepsilon(t_1 + 2t_2)). \quad (17)$$

Given $t_2 \geq t_1$ and

$$s_2 \geq s_1 \times \frac{2t_2}{t_1 + 2t_2},$$

the right-hand-side of (17) is non-negative, therefore $S - S' > 0$ and the modified solution is better than the original one, a contradiction. Therefore, in none of the sublots it is possible for machine 1 to be a unique bottleneck, concluding our proof. ■

Remark. *When $t_2 \geq t_1$, the ratio $2t_2/(t_1 + 2t_2)$ yields values between 2/3 and 1. Thus, independently of the exact values of t_1 and t_2, the second condition of the theorem may be replaced by $s_2 \geq s_1$.*

Remark. *From Theorems 1 and 2 we note that it is "easier" for machine 2 to become a bottleneck machine, since at the beginning it is idle until sublot 1 on machine 1 is completed. In particular, when both the setup and the processing times are equal for the two machines, i.e., $t_2 = t_1$ and $s_2 = s_1$, machine 2 is the bottleneck machine in all optimal solutions.*

4. Formulation and analysis of SMB problems

In this section we analyze the problem in which the set of admissible solutions is restricted to those who satisfy the SMB property. We first show in Section 4.1 (4.2) how to obtain an optimal solution to the problem in which the set of admissible solutions is restricted to those in which machine 1 (2) is the bottleneck machine. In Section 4.3, the complete solution procedure is presented.

4.1. *Machine 1 is the bottleneck machine*

The problem formulation (1)–(6) is simplified now in the following way: in Equation (4), since machine 1 is the bottleneck at sublot k for all k, the right-hand expression always achieves the maximum. Substituting I_k in Equation (2) by the right-hand expression of the maximum, we get (after some algebra) a new expression for f_k which may also be obtained directly:

$$f_k = s_2 + t_2 q_k + t_1 \sum_{i=1}^{k} q_i + k s_1. \quad (18)$$

Equation (18) states that the flow-time of sublot k is equal to the total processing time of all k sublots on machine 1, plus the processing time of sublot k on machine 2. This represents the fact that sublot k can start processing on machine 2 immediately after it is completed on machine 1, since the latter is the bottleneck machine.

The objective function can now be written as follows:

$$\sum_{k=1}^{M} f_k q_k = s_1 \sum_{k=1}^{M} q_k k + s_2 d + \left(\frac{1}{2}t_1 + t_2\right)\sum_{k=1}^{M} q_k^2 + \frac{1}{2}t_1 d^2. \quad (19)$$

Now that we have ignored the possibility of machine 2 being a bottleneck, an additional constraint is required, to make sure that this assumption is satisfied by any admissible solution (otherwise, the flow-time expression in (18) is not the correct one):

$$s_1 + t_1 q_k - s_2 - t_2 q_{k-1} \geq 0 \quad 2 \leq k \leq n, \quad (20)$$

where n is the number of positive sublots.

Since n is not known, (20) may be written in the following way:

$$(s_1 + t_1 q_k - s_2 - t_2 q_{k-1}) \times q_k \geq 0 \quad 2 \leq k \leq M. \quad (21)$$

We summarize the problem formulation for the case of machine 1 as the bottleneck machine, denoted (SMB1):

$$\min s_1 \sum_{k=1}^{M} q_k k + s_2 d + \left(\frac{1}{2}t_1 + t_2\right)\sum_{k=1}^{M} q_k^2 + \frac{1}{2}t_1 d^2, \quad (22)$$

subject to

$$(s_1 + t_1 q_k - s_2 - t_2 q_{k-1}) \times q_k \geq 0 \quad 2 \leq k \leq M, \quad (23)$$

$$\sum_{k=1}^{M} q_k = d, \quad (24)$$

$$q_k \geq 0 \quad k = 1, \ldots, M. \quad (25)$$

We note that the objective function expression is quadratic, therefore convex. Constraints (24) and (25) are linear, but constraint (23) is neither linear nor convex. Therefore, problem (22)–(25) is not a convex programming problem, and cannot be immediately solved.

4.1.1. *The pure SMB1 case*

We define the *pure (SMB1) problem* denoted as (P-SMB1) as a relaxation of the (SMB1) problem, in which constraint (23) is removed. It is clear that if the optimal solution to (P-SMB1) satisfies (23), that solution is also the optimal solution to (SMB1). We refer to an instance in which this occurs as *the pure case*, and to the solution obtained as *the pure solution*. Pure cases are easily solvable since (22), (24) and (25) define a convex programming problem that can be solved through its KKT conditions. Derivation of the pure solution of SMB1 is given in Appendix 1. The resulting solution can be represented by closed-form formulas, as follows:

$$n = \left\lceil \sqrt{\frac{1}{4} + \frac{2d(t_1 + 2t_2)}{s_1}} - \frac{1}{2} \right\rceil, \tag{26}$$

$$q_k = \frac{d}{n} + \frac{s_1(n+1)}{2(t_1+2t_2)} - k\frac{s_1}{t_1+2t_2} = \frac{d}{n} + \frac{s_1(n+1-2k)}{2(t_1+2t_2)}$$

$$k = 1, \ldots, n, \tag{27}$$

$$q_k = 0 \quad k > n. \tag{28}$$

We observe that the sublot sizes are linearly decreasing with a slope of $s_1/(t_1 + 2t_2)$. Note also that n, the number of sublots, is non-increasing with s_1; in the extreme case when $s_1 \to \infty$, n would be equal to one, and in the other extreme case when $s_1 \to 0$, n would approach infinity. All these results are independent of the value of s_2, since machine 1 is the bottleneck machine in this case.

As claimed earlier, if the solution described by (26)–(28) satisfies constraint (23) then the optimal solution to (SMB1) is found. Using (27) we are able to identify sufficient conditions under which this will occur, as follows:

For $q_k > 0$, (23) implies that $s_1 + t_1 q_k - s_2 - t_2 q_{k-1} \geq 0$ should be satisfied (for $q_k = 0$, (23) is trivially satisfied). Using

$$q_{k-1} = q_k + \frac{s_1}{t_1 + 2t_2},$$

from (27) results in the condition:

$$s_1 + t_1 q_k - s_2 - t_2 q_k - t_2 \frac{s_1}{t_1 + 2t_2} \geq 0,$$

or

$$q_k(t_1 - t_2) + s_1 - s_2 - \frac{t_2 s_1}{t_1 + 2t_2} \geq 0.$$

If $t_1 \geq t_2$, then the above condition is satisfied when:

$$s_1\left(\frac{t_1 + t_2}{t_1 + 2t_2}\right) - s_2 \geq 0. \tag{29}$$

Therefore, we have the following result:

Theorem 3. *If* $t_1 \geq t_2$ *and*

$$s_1\left(\frac{t_1 + t_2}{t_1 + 2t_2}\right) - s_2 \geq 0,$$

then the pure (*SMB1*) *solution given in* (26)–(28) *is optimal for problem* (*SMB1*).

When $t_1 = t_2$ condition (29) requires $s_1 \geq 1.5s_2$, when $t_1 \gg t_2$ the condition approaches the requirement $s_1 \geq s_2$ (reduces exactly to $s_1 \geq s_2$ when $t_2 = 0$).

Theorem 3 states sufficient but not necessary conditions for a pure solution to be feasible. Therefore, when Theorem 3 is not satisfied, it is still possible that the pure solution is feasible, in which case it is also optimal for (SMB1). If the pure solution is not feasible for the (SMB1) problem, then an alternative solution technique is required, and is described in Section 4.1.2.

Combining Theorems 1 and 3 results in the following corollary:

Corollary 1. *If* $t_1 \geq t_2$ *and* $s_1 \geq 2s_2$ *then the pure* (*SMB1*) *solution given in* (26)–(28) *is optimal for the general problem* (1)–(6).

4.1.2. *The non-pure (SMB1) case*

The formulation of (SMB1) was given in (22)–(25), but as discussed there due to constraint (23) the problem is not convex and cannot be solved in a straightforward manner. The problem *does* become convex if we fix the number of sublots, n, in which case we replace Equation (23) by Equation (20), and consider only $q_1, \ldots, q_n$ in all other constraints. We denote the resulting problem by (SMB1(n)). Given n, (SMB1(n)) is a convex programming problem, hence it can be solved by a standard non-linear optimization software. In addition, the KKT conditions of (SMB1(n)) can be established. However, closed-form formulas for the solution cannot be obtained without the knowledge of which of the indices $2, \ldots, n$ satisfy Equation (20) as equality, and which as a strong inequality. Therefore, we used the AMPL software (Fourer *et al.*, 1993) with the MINOS solver to solve the (SMB1(n)) problem.

The next question is how to determine the optimal value of n. Here we use the following conjecture:

Conjecture 1. *The values of the optimal solution of* (*SMB1*(n)) *form a unimodal function of* n.

We note that in all the numerous examples that we have examined the conjecture was satisfied.

Given that (SMB1(n)) is a convex programming problem when n is fixed, we suggest solving (SMB1) by solving a sequence of (SMB1(n)) problems, searching for the optimal value of n. If the conjecture is used, this can be done for example by performing a binary search on the value of n. Without using the conjecture the search would be performed on all possible values of n. The resulting solution is optimal for the (SMB1) problem.

4.2. *Machine 2 is the bottleneck machine*

The general problem formulation (1)–(6) is simplified now in the following way: in Equation (4), since machine 2 is the bottleneck at sublot k for all k, the left-hand expression always achieves the maximum. Consequently, the idle time on machine 2 remains I_1, therefore the flowtime of sublot k is:

$$f_k = s_1 + t_1 q_1 + k s_2 + t_2 \sum_{i=1}^{k} q_i. \tag{30}$$

The objective function becomes (after some algebra):

$$\sum_{k=1}^{M} f_k q_k = s_1 d + t_1 q_1 d + s_2 \sum_{k=1}^{M} k q_k + \frac{1}{2} t_2 \sum_{k=1}^{M} q_k^2 + \frac{1}{2} t_2 d^2. \tag{31}$$

The constraint that is required to ensure that machine 2 is a bottleneck is:

$$\sum_{j=1}^{k-1} (s_2 + t_2 q_j) - \sum_{j=2}^{k} (s_1 + t_1 q_j) \geq 0 \quad 2 \leq k \leq n. \tag{32}$$

Here the constraint is on cumulative times, reflecting the fact that there is an infinite buffer at machine 1, therefore it is never idle.

Since n, the number of positive sublots is not known, we may write (32) as follows:

$$\left(\sum_{j=1}^{k-1} (s_2 + t_2 q_j) - \sum_{j=2}^{k} (s_1 + t_1 q_j) \right) \times q_k \geq 0 \quad 2 \leq k \leq M. \tag{33}$$

We summarize the problem formulation for the case of machine 2 as the bottleneck machine, denoted (SMB2):

$$\min \; s_1 d + t_1 q_1 d + s_2 \sum_{k=1}^{M} k q_k + \frac{1}{2} t_2 \sum_{k=1}^{M} q_k^2 + \frac{1}{2} t_2 d^2, \tag{34}$$

subject to

$$\left(\sum_{j=1}^{k-1} (s_2 + t_2 q_j) - \sum_{j=2}^{k} (s_1 + t_1 q_j) \right) \times q_k \geq 0 \quad 2 \leq k \leq M, \tag{35}$$

$$\sum_{k=1}^{M} q_k = d, \tag{36}$$

$$q_k \geq 0 \quad k = 1, \ldots, M. \tag{37}$$

Again, the objective function is quadratic, therefore convex. Constraints (36) and (37) are linear, but constraint (35) is neither linear nor convex. Therefore, problem (34)–(37) is not a convex programming problem and cannot be immediately solved.

4.2.1. *The pure (SMB2) case*

Similarly to the pure (SMB1) case we define the *pure* (*SMB2*) *problem*, denoted as (P-SMB2), as a relaxation of the (SMB2) problem in which constraint (35) is removed. If the optimal solution to the pure (SMB2) problem satisfies (35), that solution is also the optimal solution to (SMB2). We refer to an instance in which this occurs as *the pure case* and to the solution obtained as *the pure solution*. The pure problem defined by (34), (36) and (37) is easily solvable since it is a convex programming problem that can be solved through its KKT conditions.

Derivation of the pure solution of (SMB2) is given in Appendix 2. The resulting solution can be represented by closed form formulas as follows:

$$n = \left\lceil \sqrt{\frac{1}{4} + \frac{2d(t_1 + t_2)}{s_2}} - \frac{1}{2} \right\rceil, \tag{38}$$

$$q_k = \begin{cases} -\dfrac{t_1 d}{t_2} + \dfrac{d}{n}\left(\dfrac{t_1 + t_2}{t_2}\right) + \dfrac{s_2}{t_2}\left(\dfrac{n+1}{2} - k\right) & k = 1, \\ \dfrac{d}{n}\left(\dfrac{t_1 + t_2}{t_2}\right) + \dfrac{s_2}{t_2}\left(\dfrac{n+1}{2} - k\right) & 2 \leq k \leq n, \end{cases} \tag{39}$$

$$q_k = 0 \quad k > n. \tag{40}$$

The sublot sizes are again linearly decreasing with a constant slope (here the slope is s_2/t_2), but only from the second sublot on. From the first to the second sublot there is an increase in the amount of $t_1 d/t_2$. Here, the value of n is non-increasing in s_2 and is independent of s_1.

As opposed to the pure (SMB1), here we were not able to find simple conditions that ensure the optimality of the above pure solution to the (SMB2) problem, i.e., problem (34)–(37). Therefore, in finding the best solution of the (SMB2)-type, we first check whether the solution given by (38)–(40) satisfies constraint (35). If it does, then this is the optimal solution to (34)–(37); otherwise we develop another solution procedure in a similar way as we did for the (SMB1) case.

4.2.2. *The non-pure (SMB2) case*

We consider the formulation of (SMB2) given in (34)–(37) with the following modifications: fix the number of sublots to be n, replace constraint (35) by constraint (32) and consider only $q_1, \ldots, q_n$ in all other constraints. Given n the resulting problem denoted by (SMB2(n)), is a convex programming problem and can be solved by standard non-linear optimization software. Again, closed-form formulas are not easily obtained from the KKT conditions.

Determining the optimal solution is easy to obtain in the following case:

Theorem 4. *If $s_2 \geq s_1$ and M is an upper bound on the number of sublots, then the optimal solution for* (*SMB2*) *is given by the optimal solution of* (*SMB2*(M)).

Proof. We prove the theorem by showing that any solution with n sublots ($n \leq M$) is feasible for problem (SMB2(M)). Consider a solution with n sublots denoted by $q_1(n)$, $q_2(n), \ldots, q_n(n)$, and consider the associated solution for (SMB2(M)) given by: $q_j(M) = q_j(n)$ for $j = 1, \ldots, n$ and $q_j(M) = 0$ for $j = n+1, \ldots, M$. Given $s_2 \geq s_1$, the latter solution is feasible where $s_2 \geq s_1$ is required in order to satisfy the kth constraint in (32), for $k = n+1, \ldots, M$. Since any solution with n sublots is feasible for problem (SMB2(M)) this also holds for the optimal solution for (SMB2). ■

Combining Theorems 2 and 4 results in the following corollary:

Corollary 2. *If* $t_2 \geq t_1$ *and* $s_2 \geq s_1$ *then the solution of problem* (*SMB2*(*M*)) *is optimal for the general problem* (1)–(6).

If the condition $s_2 \geq s_1$ is not satisfied we search again for the optimal value of n. We again have a conjecture:

Conjecture 2. *The values of the optimal solution of* (*SMB2*-(*n*)) *form a unimodal function of n.*

As with Conjecture 1 we observed this behavior in all instances that we examined. Again, we suggest solving problem (SMB2) through a sequence of (SMB2-(*n*)) problems (on all possible values of n or through a binary search).

4.3. *A solution procedure for the general problem*

We combine the results obtained thus far in order to obtain a complete solution procedure to solve the general problem (1)–(6). If according to Theorem 1 (Theorem 2), machine 1 (2) is known to be a bottleneck machine in the optimal solution, we solve (SMB1), ((SMB2)) to optimality, and the solution obtained is optimal for the general problem. (Obtaining an optimal solution for problems (SMB1) and (SMB2) was described in Sections 4.1 and 4.2, respectively.) Otherwise, we solve both the (SMB1) and (SMB2) problems to optimality, and select the better solution of the two. The resulting solution is optimal for the class of solutions that satisfy the SMB property, but it is a heuristic for the general problem. A summary of the solution procedure is shown in Fig. 4.

5. Experiments

In this section we investigate empirically various aspects of our solution procedure (Sections 5.1 and 5.2). In addition we present (in Section 5.3) a comparison with a problem that is identical to ours except for using the makespan as its objective function.

5.1. *SMB Justification*

We have investigated 200 problems (problem set 1), in which the parameters were randomly generated from a uniform probability distribution of the following ranges: $d \sim U(50{,}10\,000)$, $s_m \sim U(0{,}1000)$ for $m = 1{,}2$, $t_m \sim U(0{,}10)$ for $m = 1{,}2$. Relatively wide ranges were chosen in order to enable all combinations among the parameters, while the setup time would almost always be larger than the process time as in real world environments.

The general formulation of the problem (1)–(6) was used. The 200 instances were solved using the AMPL software with the MINOS solver for non-linear problems. Recall that the general formulation of the problem is not convex and therefore the solution found is not guaranteed to be optimal. Therefore, a 'shifting bottleneck' solution could be either better or worse than an SMB solution.

The results indicate that only in 18 out of the 200 problems a 'shifting bottleneck' solution type was obtained namely 9%. The rest of the problems provided solutions that were SMB-type. In addition comparing the objective function of the 'shifting bottleneck' solutions with SMB solutions obtained for those problems indicated that the average difference was 0.022% and the maximal was 0.27%. We conclude that SMB-type solutions provide satisfying results.

5.2. *SMB results*

We have investigated 40 additional problems (problem set 2). For all instances the unit processing time was generated from a uniform distribution between zero and one, and $d = 500$ was used. The setup time in the first 20 instances was generated from a uniform distribution between one and 10 and in the other 20 from a uniform distribution between one and 100.

In 16 out of the 40 instances the identity of the bottleneck machine was known according to Theorems 1 and 2. Out of those 16 instances, in six the pure solution was known to be optimal (according to Theorem 4) and in the other 10 the problem (SMB2(*d*)) was known to provide the optimal solution (according to Corollary 2). Therefore in those 16 instances (40%) obtaining the optimal solution was immediate, either through a closed-form formula or through one run of a convex programming problem. In these cases our procedure found the optimal solution for the *general* problem. Out of the other 24 instances, which imply the need to solve 48 problems (looking for the (SMB1) solution and the (SMB2) solution for each instance), we obtained 17 cases in which the pure solution was feasible and therefore optimal for its category. In the remaining 31 problems a search over the optimal n value was required.

The solutions obtained by our SMB procedure were compared to those obtained in Kalir and Sarin (2001), where the m-machine flow-shop problem was investigated with the restriction that the sublot sizes are equal. Given this restriction only one decision variable exists namely the number of sublots (or equivalently – the sublot size) and the optimal continuous solution may be obtained by a closed-form formula. The parameters used and the results are presented in Table 1 where the objective function refers to the average flow-time.

The differences between our solutions and the equal sublot size solutions are shown in the second column

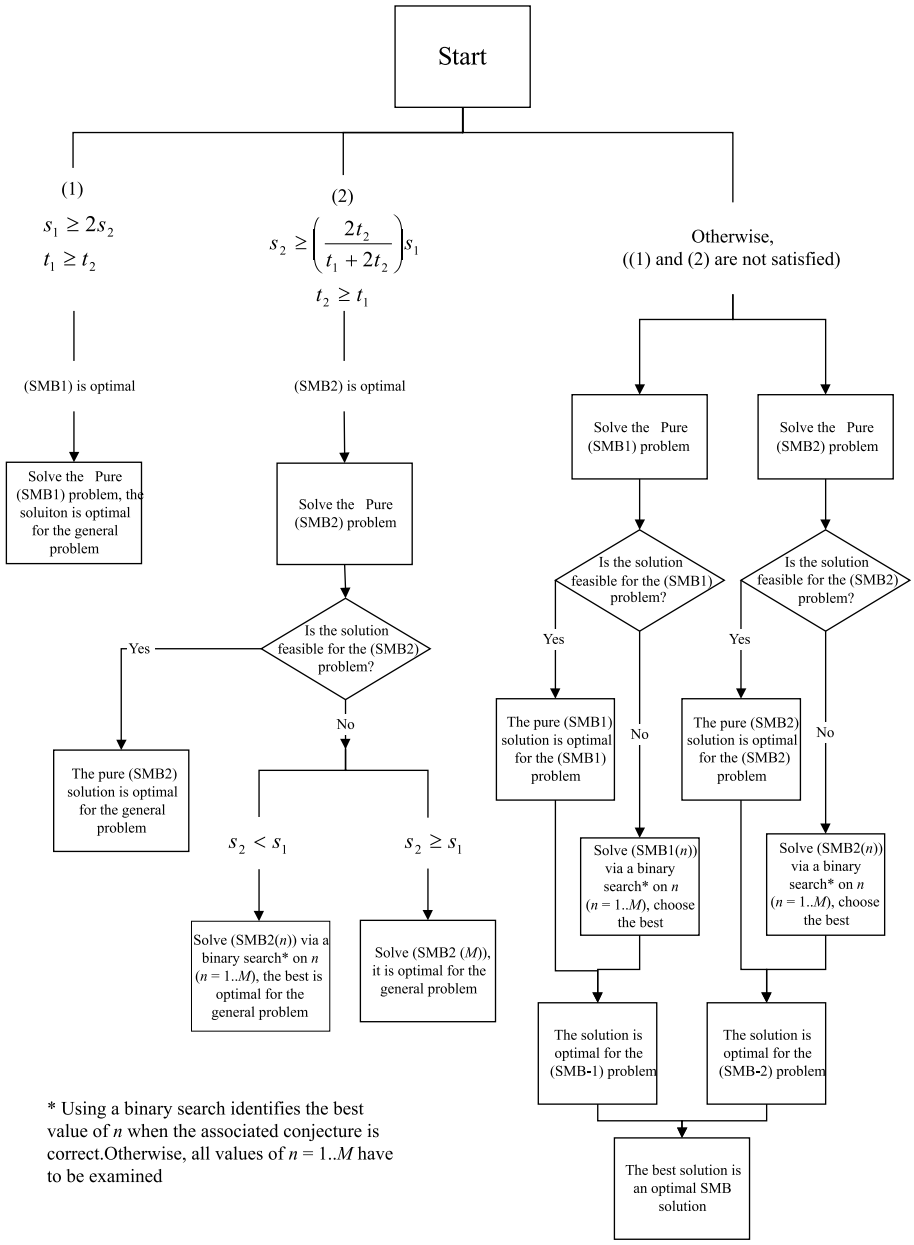

Fig. 4. The solution procedure.

from the right. The average and maximal differences were 2.28 and 6.62% respectively. An interesting observation is that in all cases the number of sublots in the general sublot size solutions was larger than in the equal sublot size solutions. On average it was 10.9 versus 8.6 batches respectively.

We also developed a lower bound for the objective value (see Appendix 3) and it is presented in the right-hand column of Table 1. We observe the magnitude of the lower bound compared with the optimal solution value. Empirically, for these 40 instances, the lower bound is on average around 70% of the optimal solution value.

5.3. *Comparison with the makespan objective function*

As most of the literature on lot splitting and lot streaming consider the makespan as the objective function, we incorporate here a comparison of the flow-time and the makespan objective functions, for the model analyzed in this paper. Our aim is to help the reader develop an understanding of the differences in scheduling that are caused by the objective function's choice. Another purpose is to learn whether a solution that minimizes flow-time may be achieved by considering the makespan objective. As can be observed from the results below the answer is no and the two solutions are in fact quite different.

Table 1. Results of problem set 2

Problem	s_1	s_2	t_1	t_2	*Equal sublot size*		*General sublot size*		*Improvement (in %)*	*LB*
					Avg. F.T.	*Number of sublots*	*Avg. F.T.*	*Number of sublots*		
1	6.72	2.32	0.77	0.69	282.1	13	278.3	18	1.35	198.4
2	9.01	3.40	0.97	0.46	343.3	10	337.5	14	1.73	252.0
3	6.21	4.50	0.97	0.95	343.7	15	339.1	21	1.35	247.9
4	7.70	3.58	0.98	0.74	351.0	13	344.2	18	1.96	253.8
5	1.70	1.03	0.61	0.22	184.8	18	182.6	25	1.24	154.6
6	1.35	7.32	0.80	0.23	236.4	22	235.6	26	0.35	200.6
7	7.45	7.69	0.58	0.15	215.2	8	210.5	11	2.26	153.7
8	1.20	6.04	0.77	0.09	224.3	20	221.7	25	1.17	194.9
9	6.11	7.44	0.68	0.34	245.4	11	241.3	14	1.71	176.7
10	1.16	3.82	0.43	0.35	139.9	15	139.1	18	0.59	109.0
11	6.22	7.61	0.10	0.79	268.9	8	265.8	11	1.20	211.2
12	6.61	8.54	0.13	0.63	228.9	7	225.7	9	1.43	171.5
13	1.42	9.22	0.72	0.84	319.4	11	299.5	12	6.62	221.4
14	5.90	8.60	0.45	0.95	337.3	10	323.5	12	4.25	252.4
15	1.78	8.03	0.14	0.66	231.8	8	227.4	10	1.97	174.4
16	6.61	3.73	0.07	0.70	223.9	11	220.9	14	1.36	186.1
17	9.00	1.84	0.64	0.97	298.2	23	296.5	25	0.56	253.6
18	3.26	1.82	0.43	0.95	283.0	22	274.9	25	2.92	243.3
19	4.02	2.27	0.05	0.91	265.9	15	264.8	21	0.44	233.3
20	6.42	5.75	0.32	0.56	207.4	10	200.2	12	3.60	151.4
21	60.52	25.86	0.89	0.42	507.0	4	495.3	5	2.38	282.3
22	93.52	56.48	0.43	0.03	365.0	2	356.6	2	2.37	200.5
23	97.84	16.77	0.81	0.61	583.2	3	567.1	5	2.83	299.5
24	77.79	61.93	0.66	0.02	431.5	2	423.6	3	1.87	243.1
25	78.05	23.18	0.61	0.56	474.9	3	461.1	5	2.99	230.1
26	3.75	19.37	0.85	0.54	302.0	10	300.4	11	0.53	217.3
27	52.52	76.76	0.97	0.31	550.6	4	539.2	5	2.11	295.4
28	55.92	74.82	0.75	0.62	530.5	4	528.5	4	0.37	242.7
29	93.19	98.98	0.40	0.03	396.9	2	386.8	2	2.60	192.9
30	63.52	86.24	0.90	0.54	594.5	4	583.7	5	1.86	289.6
31	32.88	52.96	0.62	0.90	524.1	5	503.0	5	4.18	311.3
32	3.23	55.69	0.13	0.87	426.6	3	421.6	4	1.19	276.1
33	59.66	60.68	0.59	0.70	505.7	4	496.0	5	1.94	296.4
34	1.14	16.98	0.36	0.97	373.0	7	357.6	9	4.30	261.3
35	3.85	69.37	0.48	0.64	436.5	3	414.0	4	5.42	233.4
36	88.40	5.77	0.38	0.56	403.4	3	389.5	4	3.58	234.1
37	40.90	7.90	0.03	0.07	97.6	2	93.1	2	4.86	65.9
38	96.77	32.60	0.02	0.63	374.5	3	370.6	4	1.05	285.9
39	73.35	42.22	0.47	0.53	434.7	3	421.2	5	3.22	248.0
40	53.62	2.81	0.54	0.76	400.8	4	386.9	6	3.59	247.3

Remarks: (i) $d = 500$ in all instances. (ii) the numbers are associated with the average flow-time.

Towards that and since no earlier study has considered the same model as ours with the makespan objective function, we have adjusted our mathematical programming formulation which was presented in Section 2 to consider the makespan rather than the flow-time objective function. In addition to changing the objective function itself, we had to introduce new binary variables and define them through the constraints. Due to the change in the objective function there are no more non-linearities in the formulation.

The resulting formulation is presented in Appendix 4. We used the AMPL software with the CPLEX solver for integer programming to solve the resulting problem for the 40 instances investigated in Section 5.2 (problem set 2). The results are presented in Table 2. In this table we present a comparison of two solutions: one is the solution obtained by our procedure which is designed to minimize the average flow-time, and the other is the solution that minimizes the makespan, obtained by solving the ILP discussed above. For each solution we compute both the

Table 2. A comparison of the flow-time and the makespan objective functions

Problem	*Min. of flow-time*			*Min. of makespan*			$\frac{FT(MS)}{FT(FT)}$	$\frac{MS(FT)}{MS(MS)}$	*Number of sublots FT/MS*
	FT	*MS*	*Number of sublots*	*FT*	*MS*	*Number of sublots*			
1	278.3	509.1	18	299.9	456.6	8	1.078	1.115	2.25
2	337.5	616.7	14	459.5	534.2	4	1.361	1.155	3.50
3	339.1	622.8	21	357.1	587.3	9	1.053	1.061	2.33
4	344.2	633.0	18	397.0	559.0	7	1.153	1.132	2.57
5	182.6	348.7	25	283.5	315.4	5	1.553	1.106	5.00
6	235.6	443.5	26	370.3	416.9	4	1.572	1.064	6.50
7	210.5	380.1	11	269.0	306.5	4	1.278	1.240	2.75
8	221.7	421.7	25	379.2	395.8	3	1.710	1.065	8.33
9	241.3	435.4	14	313.9	384.2	4	1.301	1.133	3.50
10	139.1	255.9	18	149.3	243.9	9	1.073	1.049	2.00
11	265.8	488.5	11	382.6	421.9	2	1.439	1.158	5.50
12	225.7	401.0	9	297.6	349.0	3	1.319	1.149	3.00
13	299.5	539.2	12	311.3	498.2	7	1.039	1.082	1.71
14	323.5	590.4	12	380.4	526.7	4	1.176	1.121	3.00
15	227.4	413.8	10	307.8	356.9	3	1.354	1.159	3.33
16	220.9	410.5	14	336.1	367.5	2	1.522	1.117	7.00
17	296.5	554.5	25	320.1	526.6	8	1.080	1.053	3.13
18	274.9	526.5	25	341.2	492.6	6	1.241	1.069	4.17
19	264.8	507.8	21	440.9	465.0	2	1.665	1.092	10.50
20	200.2	362.9	12	220.0	323.6	5	1.099	1.121	2.40
21	495.3	786.1	5	588.0	648.1	2	1.187	1.213	2.50
22	356.6	463.2	2	380.0	380.0	1	1.066	1.219	2.00
23	567.1	913.2	5	636.3	713.6	2	1.122	1.280	2.50
24	423.6	626.4	3	479.7	479.7	1	1.132	1.306	3.00
25	461.1	723.9	5	511.5	592.0	2	1.109	1.223	2.50
26	300.4	518.3	11	330.3	498.4	6	1.100	1.040	1.83
27	539.2	838.2	5	623.7	710.2	2	1.157	1.180	2.50
28	528.5	759.6	4	594.2	710.5	2	1.124	1.069	2.00
29	386.8	489.8	2	407.2	407.2	1	1.053	1.203	2.00
30	583.7	914.3	5	656.7	773.1	2	1.125	1.183	2.50
31	503.0	788.7	5	523.7	694.6	3	1.041	1.136	1.67
32	421.6	682.4	4	539.2	551.2	2	1.279	1.238	2.00
33	496.0	778.5	5	551.0	665.5	2	1.111	1.170	2.50
34	357.6	646.5	9	471.3	546.7	3	1.318	1.182	3.00
35	414.0	632.3	4	441.0	537.2	3	1.065	1.177	1.33
36	389.5	569.9	4	400.6	488.4	3	1.028	1.167	1.33
37	93.1	113.8	2	98.8	98.8	1	1.061	1.152	1.00
38	370.6	546.0	4	454.4	454.4	1	1.226	1.202	4.00
39	421.2	646.2	5	459.4	547.9	2	1.091	1.180	2.50
40	386.9	608.4	6	394.8	537.7	4	1.020	1.132	1.50
Average			10.9			3.6	1.212	1.147	3.191

average flow-time (denoted FT) and the makespan (denoted MS), and indicate the number of sublots included in the solution. In the third, second and most right-hand columns we compare the two solutions with respect to the ratio of the flow-time values, the ratio of the makespan values and the ratio of the number of sublots, respectively. In these columns the performance measure and the solution type are indicated outside and inside the parenthesis, respectively.

We draw some important conclusions from the results. As expected, the solution which minimizes the makespan objective, is not optimal for the flow-time objective function. In fact, the differences are quite large. The optimal makespan solution yields an objective value for the flow-time that is on average 21.2% higher than the objective value obtained by our procedure. The maximum difference among all 40 problems was 71%. Therefore, it is clear that using the makespan solution does not

produce good solutions for the flow-time minimization problem.

The opposite direction is also true, but the differences are smaller. Namely, our procedure for minimizing the flow-time is not optimal for the makespan objective, but the average and the maximum differences are only 14.7 and 30.6%, respectively.

Finally, it is interesting to compare the number of sublots that are obtained under both solutions. In all 40 problems the number of sublots obtained for the flow-time solution was much higher than the number of sub-lots obtained for the makespan solution. On average the flow-time solution yields a number of sublots that is 3.19 higher than in the makespan solution. The maximum ratio is obtained for problem 19 in which the flow-time solution used 21 sublots, while the makespan solution used two sublots only (a ratio of 10.5). We explain the difference in the number of sublots used as follows: when minimizing flow-time – all sublots participate in the objective function, therefore it is important to release initial sublots early by using additional sublots. In makespan minimization only the last sublot is considered in the objective function, therefore it is important to perform it as early as possible, regardless of the flow-time of initial sublots.

One case in which the results of the flow-time model would be similar to the results of the makespan model is in the extreme case in which the setup is negligible. In that case in both models the sublots would be as small as possible, therefore there would be a large number of sublots. This intuition is used in developing our lower bound for the flow time model (see Appendix 3).

6. Conclusions

We presented a solution procedure for the two-machine flow-shop lot splitting problem with sublot-attached setup times and a flow-time objective function. This is the first work which suggests how to obtain a solution which may consist of non-equal sublot sizes for this system. We developed a solution procedure which is based on an intuitive solution structure, namely the SMB (*Single Machine Bottleneck*) property. We say that a solution satisfies the SMB property, when the same machine is the bottleneck machine (i.e., has no idle time between sublots) throughout the production process. Although the SMB property does not always achieve the optimal solution, it is shown empirically to be very close to optimal. For some cases we proved that the SMB property *is* satisfied in all optimal solutions.

Empirical results demonstrate an improvement of the objective value over the optimal value given equal sublot sizes. Note that when the sublot sizes are restricted to be identical the solution automatically satisfies the SMB property. In addition, we compared the solution obtained by our solution procedure to the solution of the same model with the makespan objective function and observed that the solutions behave differently. The superiority of the objective value when considering minimization of flow-time is quite significant. The applicability of lot splitting decisions to industries that produce expensive products implies that any reduction in the average flow-time is very meaningful.

References

Baker, K.R. (1974) *Introduction to Sequencing and Scheduling*, Wiley, New York. NY.

Baker, K.R. and Jia, D. (1993) A comparative study of lot streaming procedures. *OMEGA International Journal of Management Science*, **21**(5), 561–566.

Chen, J. and Steiner, G. (1996) Lot streaming with detached setups in three-machine flow shops. *European Journal of Operational Research*, **26**, 591–611.

Chen, J. and Steiner, G. (1998) Lot streaming with attached setups in three-machine flow shops. *IIE Transactions*, **30**, 1075–1084.

Cheng, T.C.E., Chen, Z.L., Kovalyov, M.Y. and Lin, B.M.T. (1996) Parallel-machine batching and scheduling to minimize total completion time. *IIE Transactions*, **28**, 953–956.

Cheng, T.C.E., Lin, B.M.T. and Toker, A. (2000) Makespan minimization in the two-machine flowshop batch scheduling problem. *Naval Research Logistics*, **47**, 128–144.

Conway, R.W., Maxwell, W.L. and Miller, L.W. (1967) *Theory of Scheduling*, Addison-Wesley, Reading, MA.

Dobson, G., Karmarkar, U. and Rummel, J. (1987) Batching to minimize flow times on one machine. *Management Science*, **33**(6), 784–799.

Dobson, G., Karmarkar, U.S. and Rummel, J.L. (1989) Batching to minimize flow times on parallel heterogeneous machines. *Management Science*, **35**(5), 607–613.

Fourer, R., Gay, D.M. and Kernighan, B.W. (1993) *AMPL: A Modeling Language for Mathematical Programming*, Boyd and Fraser, Danvers, MA.

Kalir, A.A. and Sarin, S.C. (2001) Optimal solutions for the single batch flow-shop lot streaming problem with equal sublots. *Decision Sciences*, **32**(2), 387–397.

Kropp, D.H. and Smunt, T.L. (1990) Optimal and heuristic models for lot-splitting in a flow-shop. *Decision Sciences*, **21**(4), 691–709.

Naddef, D. and Santos, C. (1988) One-pass batching algorithm for the one-machine problem. *Discrete Applied Mathematics*, **21**, 133–145.

Pinedo, M. (1995) *Scheduling*: *Theory*, *Algorithms and Systems*, Prentice-Hall, Englewood Cliffs, NJ.

Potts, C.N. and Kovalyov, M.Y. (2000) Scheduling with batching: a review. *European Journal of Operational Research*, **120**, 228–249.

Santos, C. and Magazine, M. (1985) Batching in single operation manufacturing systems. *Operations Research Letters*, **4**(3), 99–103.

Şen, A., Topaloğlu, E. and Benli, Ö.S. (1998) Optimal streaming of a single job in a two-stage flow shop. *European Journal of Operational Research*, **110**, 42–62.

Shallcross, D.F. (1992) A polynomial algorithm for a one machine batching problem. *Operations Research Letters*, **11**, 213–218.

Trietsch, D. and Baker, K.R. (1993) Basic techniques for lot streaming. *Operations Research*, **41**(6), 1065–1076.

Truscott, W.G. (1985) Scheduling production activities in multi-stage manufacturing systems. *International Journal of Production Research*, **23**(2), 315–328.

Truscott, W.G. (1986) Production scheduling with capacity constrained transportation activities. *Journal of Operations Management*, **6**(3), 1986, 333–348.

Appendices

Appendix 1: The pure solution for (SMB1)

We derive the pure solution of (SMB1) by using the KKT conditions of problem (22)–(25) excluding constraint (23).

The problem in hand, denoted as (P-SMB1) is the following:

$$\min s_1 \sum_{k=1}^{M} q_k k + s_2 d + \left(\frac{1}{2}t_1 + t_2\right)\sum_{k=1}^{M} q_k^2 + \frac{1}{2}t_1 d^2, \quad \text{(A1)}$$

subject to

$$\sum_{k=1}^{M} q_k = d, \quad \text{(A2)}$$

$$q_k \geq 0 \quad k = 1, \ldots, M. \quad \text{(A3)}$$

Associating a Lagrange multiplier v with constraint (A2) and a Lagrange multiplier u_k with the kth non-negativity constraint (A3), we get the following KKT conditions:

$$s_1 k + (t_1 + 2t_2)q_k + u_k + v = 0 \quad 1 \leq k \leq M, \quad \text{(A4)}$$

$$u_k q_k = 0 \quad 1 \leq k \leq M, \quad \text{(A5)}$$

$$u_k \leq 0 \quad 1 \leq k \leq M. \quad \text{(A6)}$$

A solution that satisfies conditions (A4)–(A6) together with constraints (A2)–(A3) is an optimal solution to (P-SMB1). We find it by solving a system of $2M + 1$ variables (the q variables, the u variables and v) and $2M + 1$ equations ((A2), (A4), and (A5)) under the additional conditions of (A3) and (A6).

We solve (A4) for q_k:

$$q_k = -\frac{s_1 k}{t_1 + 2t_2} - \frac{u_k}{t_1 + 2t_2} - \frac{v}{t_1 + 2t_2}. \quad \text{(A7)}$$

We use n to denote the number of positive sublots and note that for $q_k > 0$, (A5) implies $u_k = 0$.

Then, summing (A7) for $k = 1, \ldots, n$ we get:

$$d = -\frac{n(n+1)}{2} \times \frac{s_1}{t_1 + 2t_2} - n \times \frac{v}{t_1 + 2t_2},$$

and therefore:

$$-\frac{v}{t_1 + 2t_2} = \frac{d}{n} + s_1 \frac{n+1}{2(t_1 + 2t_2)}. \quad \text{(A8)}$$

Using (A7) and (A8), we get for $q_k > 0$:

$$q_k = \frac{d}{n} + \frac{s_1(n+1)}{2(t_1 + 2t_2)} - k\frac{s_1}{t_1 + 2t_2} = \frac{d}{n} + \frac{s_1(n+1-2k)}{2(t_1 + 2t_2)}. \quad \text{(A9)}$$

Since n is the largest index of k for which $q_k > 0$, n is the largest index for which (A9) results in: $q_n > 0$ and $q_{n+1} \leq 0$, which can be written as:

$$q_n = \frac{d}{n} + \frac{s_1(n+1-2n)}{2(t_1 + 2t_2)} = \frac{d}{n} - \frac{s_1(n-1)}{2(t_1 + 2t_2)} > 0, \quad \text{(A10)}$$

$$q_{n+1} = \frac{d}{n} + \frac{s_1(n+1-2n-2)}{2(t_1 + 2t_2)} = \frac{d}{n} - \frac{s_1(n+1)}{2(t_1 + 2t_2)} \leq 0. \quad \text{(A11)}$$

After some algebra, we get:

$$n(n-1) < \frac{2d(t_1 + 2t_2)}{s_1} \leq n(n+1), \quad \text{(A12)}$$

where the first inequality in (A12) results from (A10) and the second from (A11).

Since the interval defined in (A12) by n does not overlap with the interval defined in (A12) by $n + 1$, there is only one value of n that satisfies (A12). This value is found by solving (A12) and we obtain:

$$n = \left\lceil \sqrt{\frac{1}{4} + \frac{2d(t_1 + 2t_2)}{s_1}} - \frac{1}{2} \right\rceil. \quad \text{(A13)}$$

To conclude the verification of the optimality conditions, we show that for $k > n$ we can solve (A7) subject to the condition $u_k \leq 0$ is satisfied. Plugging (A8) into (A7) for $k > n$ for which $q_k = 0$ results in:

$$\frac{u_k}{t_1 + 2t_2} = -\frac{s_1 k}{t_1 + 2t_2} + \frac{d}{n} + \frac{s_1(n+1)}{2(t_1 + 2t_2)},$$

which indicates that $u_k \leq 0$ due to: $k \geq n + 1$ which implies $n(n+1) \leq nk$ and also:

$$\frac{2d(t_1 + 2t_2)}{s_1} \leq n(n+1),$$

from (A12), implying:

$$\frac{2d(t_1 + 2t_2)}{s_1} \leq nk.$$

Appendix 2: The pure solution for (SMB2)

We derive the pure solution of (SMB2) by using the KKT conditions of problem (34)–(37), excluding constraint (35).

The problem in hand denoted as (P-SMB2) is the following:

$$\min s_1 d + t_1 q_1 d + s_2 \sum_{k=1}^{M} k q_k + \frac{1}{2}t_2 \sum_{k=1}^{M} q_k^2 + \frac{1}{2}t_2 d^2, \quad \text{(A14)}$$

subject to

$$\sum_{k=1}^{M} q_k = d, \quad \text{(A15)}$$

$$q_k \geq 0 \quad k = 1, \ldots, M. \quad \text{(A16)}$$

Associating a Lagrange multiplier v with constraint (A15) and a Lagrange multiplier u_k with the kth non-negativity constraint (A16), we get the following KKT conditions:

$$\begin{cases} t_1 d + s_2 k + t_2 q_k + u_k + v = 0 & k = 1, \\ s_2 k + t_2 q_k + u_k + v = 0 & 2 \leq k \leq M, \end{cases} \quad \text{(A17)}$$

$$u_k q_k = 0 \quad 1 \le k \le M, \tag{A18}$$

$$u_k \le 0 \quad 1 \le k \le M. \tag{A19}$$

A solution that satisfies conditions (A17)–(A19) together with constraints (A15)–(A16) is an optimal solution to (P-SMB2). We find it by solving a system of $2M+1$ variables (the q variables, the u variables and v) and $2M+1$ equations ((A15), (A17), and (A18)) under the additional conditions of (A16) and (A19). We solve (A17) for q_k:

$$q_k = \begin{cases} -\dfrac{t_1 d}{t_2} - \dfrac{s_2 k}{t_2} - \dfrac{u_k}{t_2} - \dfrac{v}{t_2} & k = 1, \\ -\dfrac{s_2 k}{t_2} - \dfrac{u_k}{t_2} - \dfrac{v}{t_2} & 2 \le k \le M. \end{cases} \tag{A20}$$

We use n to denote the number of positive sublots and note that for $q_k > 0$, (A18) implies $u_k = 0$. Then, summing (A20) for $k = 1, \ldots, n$ we get:

$$d = -\frac{t_1 d}{t_2} - \frac{n(n+1)}{2} \times \frac{s_2}{t_2} - n \times \frac{v}{t_2},$$

and therefore:

$$-\frac{v}{t_2} = \frac{d}{n} + \frac{t_1}{t_2} \times \frac{d}{n} + \frac{n+1}{2} \times \frac{s_2}{t_2}. \tag{A21}$$

Using (A20) and (A21) we get for $q_k > 0$:

$$q_k = \begin{cases} -\dfrac{t_1 d}{t_2} + \dfrac{d}{n}\left(\dfrac{t_1 + t_2}{t_2}\right) + \dfrac{s_2}{t_2}\left(\dfrac{n+1}{2} - k\right) & k = 1, \\ \dfrac{d}{n}\left(\dfrac{t_1 + t_2}{t_2}\right) + \dfrac{s_2}{t_2}\left(\dfrac{n+1}{2} - k\right) & 2 \le k \le n. \end{cases} \tag{A22}$$

Since n is the largest index of k for which $q_k > 0$, n is the largest index for which (A22) results in: $q_n > 0$ and $q_{n+1} \le 0$, which, given that $n \ge 2$, can be written as:

$$q_n = \frac{d}{n}\left(\frac{t_1 + t_2}{t_2}\right) + \frac{s_2}{t_2}\left(\frac{n+1}{2} - n\right) > 0, \tag{A23}$$

$$q_{n+1} = \frac{d}{n}\left(\frac{t_1 + t_2}{t_2}\right) + \frac{s_2}{t_2}\left(\frac{n+1}{2} - (n+1)\right) \le 0. \tag{A24}$$

After some algebra, we get:

$$n(n-1) < \frac{2d(t_1 + t_2)}{s_2} \le n(n+1), \tag{A25}$$

where the first inequality in (A25) results from (A23) and the second from (A24). Since the interval defined in (A25) by n does not overlap with the interval defined in (A25) by $n+1$, there is only one value of n that satisfies (A25). This value is found by solving (A25) and we obtain:

$$n = \left\lceil \sqrt{\frac{1}{4} + \frac{2d(t_1 + t_2)}{s_2}} - \frac{1}{2} \right\rceil. \tag{A26}$$

To conclude the verification of the optimality conditions, we show that for $k > n$ we can solve (A20) subject to the condition $u_k \le 0$ is satisfied. Plugging (A21) into (A20) for $k > n$, for which $q_k = 0$, results in:

$$\frac{u_k}{t_2} = \frac{d}{n}\left(\frac{t_1 + t_2}{t_2}\right) + \frac{s_2}{t_2}\left(\frac{n+1}{2} - k\right),$$

which indicates that $u_k \le 0$ due to: $k \ge n+1$ which implies $n(n+1) \le nk$ and also:

$$\frac{2d(t_1 + t_2)}{s_2} \le n(n+1),$$

from (A25), implying:

$$\frac{2d(t_1 + t_2)}{s_2} \le nk.$$

Appendix 3: A lower bound

We develop a lower bound on the total flow-times (over all units) by considering a solution in which:

1. the sublot sizes are as small as possible; and
2. the setup cost is incurred on each machine only at the beginning of the first sublot.

The total flow-time of the specified solution forms a lower bound on the total flow-time of any solution. This is true since the first condition provides a lower bound on the contribution of the variable processing times and the second condition provides a lower bound on the contribution of the setup times.

Accordingly, we assume that in the specified solution, there are d/ε sublots and the size of each sublot is ε, where $\varepsilon \ge 0$ is arbitrarily small. We analyze the total flow-time of the above solution by distinguishing between two cases:

Case 1: $t_1 > t_2$

In this case, the flow-time of the kth sublot is: $s_1 + \varepsilon k t_1 + \varepsilon t_2$, see Fig. A1.

Since the size of the kth sublot is ε its contribution to the total flow-time is: $\varepsilon(s_1 + \varepsilon k t_1 + \varepsilon t_2)$, and the total flow-time of all d/ε sublots is:

$$\begin{aligned} \sum_{k=1}^{d/\varepsilon} \varepsilon(s_1 + \varepsilon k t_1 + \varepsilon t_2) &= d s_1 + \varepsilon^2 t_1 \sum_{k=1}^{d/\varepsilon} k + \varepsilon d t_2 \\ &= d s_1 + \varepsilon^2 t_1 \frac{d}{\varepsilon}\left(\frac{d}{\varepsilon} + 1\right) \Big/ 2 + \varepsilon d t_2 \\ &\ge d s_1 + \frac{1}{2} t_1 d^2 \\ &\equiv LB_1. \end{aligned}$$

Case 2: $t_2 \ge t_1$

In this case the flow time of the kth sublot is: $s_1 + s_2 + \varepsilon t_1 + \varepsilon k t_2$, see Fig. A2.

Since the size of the kth sublot is ε, its contribution to the total flow-time is: $\varepsilon(s_1 + s_2 + \varepsilon t_1 + \varepsilon k t_2)$ and the total flow-time of all d/ε sublots is:

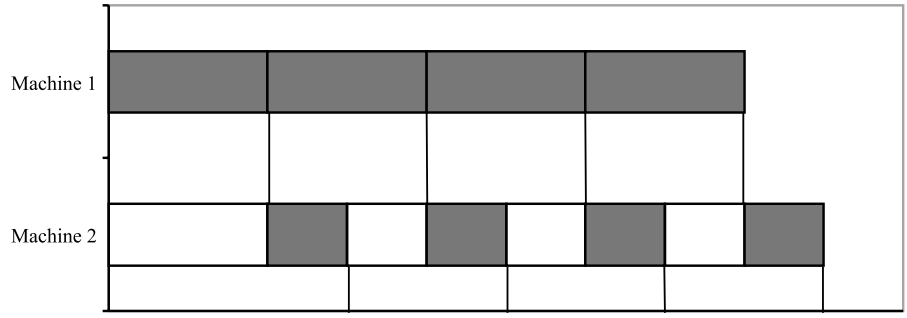

Fig. A1. Case 1.

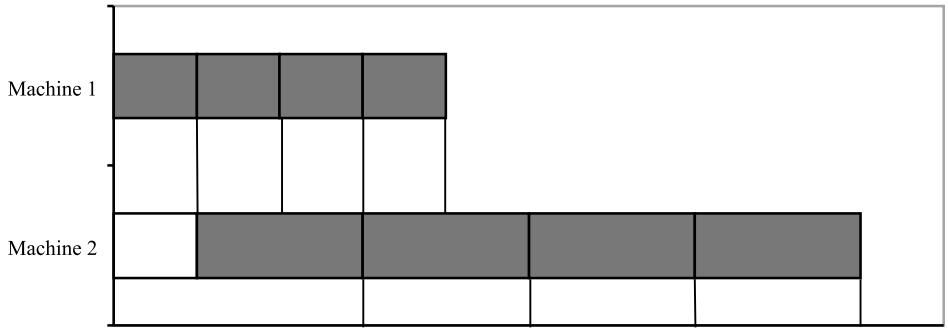

Fig. A2. Case 2.

$$\sum_{k=1}^{d/\varepsilon} \varepsilon(s_1 + s_2 + \varepsilon t_1 + \varepsilon k t_2) = ds_1 + ds_2 + \varepsilon d t_1 + \varepsilon^2 t_2 \sum_{k=1}^{d/\varepsilon} k$$
$$= ds_1 + ds_2 + \varepsilon d t_1 + \varepsilon^2 t_2 \frac{d}{\varepsilon}\left(\frac{d}{\varepsilon} + 1\right)\Big/ 2$$
$$\geq d(s_1 + s_2) + \frac{1}{2} t_2 d^2 \equiv LB_2.$$

Conclusion:

A lower bound on the total flow-time of *any* instance is:

$$ds_1 + \frac{1}{2} d^2 \max\{t_1, t_2\}.$$

A higher lower bound can be determined when $t_2 \geq t_1$ as follows:

$$LB_2 = d(s_1 + s_2) + \frac{1}{2} t_2 d^2.$$

That is:

$$LB = \begin{cases} ds_1 + \frac{1}{2} t_1 d^2 & \text{if } t_1 > t_2, \\ d(s_1 + s_2) + \frac{1}{2} t_2 d^2 & \text{if } t_2 \geq t_1. \end{cases}$$

Appendix 4: Formulation for the makespan objective

We use the following additional variables:

$$p_k = \begin{cases} 1 & \text{if sublot } k \text{ is the last positive sublot,} \\ 0 & \text{otherwise.} \end{cases}$$

The definition of the rest of the variables remain unchanged.

$$\min I_M + dt_2 + s_2 \sum_{k=1}^{M} p_k, \tag{A27}$$

subject to

$$I_1 = s_1 + t_1 q_1, \tag{A28}$$
$$I_k \geq I_{k-1} \quad k = 2, \ldots, M, \tag{A28}$$
$$I_k \geq \left(t_1 \sum_{i=1}^{k} q_i + s_1 \sum_{i=1}^{k} p_i\right) - \left(t_2 \sum_{i=1}^{k-1} q_i + s_2 \sum_{i=1}^{k-1} p_i\right) \quad k = 2, \ldots, M, \tag{A29}$$
$$\sum_{k=1}^{M} q_k = d, \tag{A30}$$
$$q_k \leq d p_k \quad k = 1, \ldots, M, \tag{A31}$$
$$p_k \leq p_{k-1} \quad k = 2, \ldots, M, \tag{A32}$$
$$q_k \geq 0 \quad k = 1, \ldots, M. \tag{A33}$$

Biographies

Joseph Bukchin is a faculty member of the Department of Industrial Engineering at Tel Aviv University. He received his B.Sc., M.Sc. and D.Sc. degrees in Industrial Engineering from the Technion Israel Institute of Technology. He is a member of the IIE and INFORMS. In 2000–2002 he was a visiting professor at the Grado Department of Industrial and Systems Engineering at Virginia Tech. His main research interests are in the areas of assembly systems design, assembly line balancing, facility design, design of cellular manufacturing systems, operational scheduling as well as work station design with respect to cognitive and physical aspects of the human operator.

Michal Tzur is a senior lecturer in the Department of Industrial Engineering at Tel Aviv University, Israel. She joined Tel Aviv University in 1994 after spending 3 years at the Operations and Information Management Department at The Wharton School, University of Pennsylvania. She received her B.A. from Tel Aviv University and her M.Phil and Ph.D. from Columbia University. Michal is a member of IIE, INFORMS and POMS. Her research interests are in the areas of supply chain management, multi-echelon inventory management, operations scheduling and production planning.

Michal Jaffe received her B.Sc. in Industrial Engineering from Tel Aviv University in 1995. The research on lot splitting was performed, partly, when she was a graduate student at Tel Aviv University in 1996–1998. Michal has worked for EL AL Israeli Airlines since 1995. She is currently Vice Director of Marketing, and Manager of Product Development. Previously, she served as Vice Director of Transportation, and Manager of Flight Scheduling, as well as the manager of the Operations Research Department and an operations research engineer, in charge of aircrew planning.

Contributed by the Scheduling Department