International Journal of Industrial Engineering Computations 7 (2016) 507–516

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

# A non-permutation flowshop scheduling problem with lot streaming: A Mathematical model

**Daniel Rossit[a, b*], Fernando Tohmé[c,d], Mariano Frutos[a,b], Jonathan Bard[e] and Diego Broz[d,f]**

[a]*Engineering Department, Universidad Nacional del Sur, Bahía Blanca, Argentina*
[b]*IIESS-CONICET, Argentina*
[c]*Economics Department, Universidad Nacional del Sur, Bahía Blanca, Argentina*
[d]*INMABB-CONICET, Argentina*
[e]*Graduate Program in Operations Research and Industrial Engineering, The University of Texas Austin, TX, United States*
[f]*Forestry Sciences School, Universidad Nacional de Misiones, Argentina*

**C H R O N I C L E**

*Article history*:
Received July 22 2015
Received in Revised Format
Septmber 26 2015
Accepted November 25 2015
Available online
November 25 2015

Keywords:
*Non-Permutation Flowshop*
*Scheduling*
*Lot Streaming*
*Makespan*

**A B S T R A C T**

In this paper we investigate the use of lot streaming in non-permutation flowshop scheduling problems. The objective is to minimize the makespan subject to the standard flowshop constraints, but where it is now permitted to reorder jobs between machines. In addition, the jobs can be divided into manageable sublots, a strategy known as lot streaming. Computational experiments show that lot streaming reduces the makespan up to 43% for a wide range of instances when compared to the case in which no job splitting is applied. The benefits grow as the number of stages in the production process increases but reach a limit. Beyond a certain point, the division of jobs into additional sublots does not improve the solution.



## 1. Introduction

One of the main issues in scheduling jobs during manufacturing is the efficient use of resources. Lot streaming, which involves dividing production lots or jobs into sublots, and then processing the overlapped sublots on different machines, has proven to be an effective way of addressing this issue. In recent years, lot streaming has received a great deal of attention as a strategy for minimizing any number of production objectives, including makespan, total tardiness, and maximum tardiness. Furthermore, it has been shown to reduce both work-in-process and the cycle time (Sarin & Jaiprakash, 2007). Since its conception by Reiter (1966), the development and applications of lot streaming have increased considerably, mostly in flowshop production systems. One of the first significant contributions, presented by Potts and Baker (1989), showed that for instances with two and three stages and a single product, a lot streaming problem could be solved optimally with consistent sublots; that is, sublots that are the same size at all stages. Following this work Trietsch and Baker (1993) proposed a problem classification

* Corresponding author. Tel +54 (0291) 4595156 Fax: +54 (0291) 4595119
E-mail: daniel.rossit@uns.edu.ar (D. Rossit)


doi: 10.5267/j.ijiec.2015.11.004

scheme, Vickson (1995) extended the analysis to several products, and Sriskandarajah and Wagneur (1999) added no-wait conditions to the problem. Larger instances were solved by means of evolutionary methods. Kumar et al. (2000) using this approach showed that the multi-product problem with continuous sized sublots could be reduced to the traveling salesman problem. Other research in this area includes the work of Yoon and Ventura (2002) who applied a genetic algorithm to find soluitons; Tseng and Liao (2008), who applied a particle swarm optimization approach, and Pan et al. (2011) who implemented a discrete bee colony algorithm. More recent work on lot streaming has been undertaken by Pan and Ruiz (2012) and Davendra et al. (2014).

Lot streaming has also been applied in the context of hybrid flowshop problems in which more than one machine (or resource unit) is available at each stage. The first application was by Zhang et al. (2005) who considered just two stages and parallel machines only in the first stage. Ruiz et al. (2008) tackled the hybrid flowshop problem with an approach somewhat related to lot streaming. In a similar vein, Defersha (2011) compared methods for finding solutions to the flexible hybrid flowshop problem with the lot streaming approach. Their ultimate goal was to close the gap between the two methods. Several authors followed this lead, including Azzi et al. (2012) and Wang et al. (2014).

All the aforementioned studies have a single feature in common: they all analyze the flowshop lot streaming scheduling problem under permutation schedules (permutation flowshop, PFS). In the PFS model, the sequence of jobs on all machines or at all stages is the same. However, in most cases of practical concern, this condition is not required by the production process. Usually there is no need for identical schedules on all machines. The permutation condition is generally assumed to make the scheduling problem more tractable. For the PFS, $n!$ possible schedules (where $n$ is the number of jobs) have to be examined. This is in a stark contrast to the non-permutation flow shop (NPFS) case in which the search space grows to $n!^m$ (where $m$ is the number of machines).. While the permutation condition has the advantage of lower complexity, achieving optimal solutions may still not be possible for production systems with more than three-stages (Pinedo, 2002). Moreover, keeping the sequence of jobs fixed may induce unproductive waiting time, for instance, when there is more than one job queuing for a busy machine. If the sequence is allowed to change (the queue is reordered) new solutions and, possibly, better results can be obtained. This is one of the goals that we explore in this paper.

With the increase in computational power and the development of improved optimization software, it is now possible to solve larger, more realistic instances of the general or non-permutation flowshop problem. Moreover, it has been shown that NPFS schedules dominate PFS schedules (Pugazhendhi, et al. 2003; Liao et al. 2006). This is particularly relevant for practitioners since it means that productivity can be increased without any extra investment in machinery or any modification of the actual production system or product design. Subsequent to the work of Pugazhendhi et al. (2003) and Liao et al. (2006), several algorithms and procedures have been proposed for solving NPFS (e.g., see Ying, 2008; Ying et al., 2010; Lin & Ying, 2009; Rossi & Lanzetta, 2014; Mehravaran & Logendran, 2012, 2013; Vahedi-Nouri et al., 2013, Shen et al., 2014). Although the interest in the lot streaming has grown in the past few years, it has not been adopted as an optimization strategy or even studied in the context of NPFSs. To the best of our knowledge the only paper that deals with a strategy similar to lot streaming is by Shen et al. (2014). They considered a cell manufacturing system in which the jobs belong to families of products and batching is used. The splitting procedure consists in dividing a family into jobs but not splitting the jobs themselves, i.e., the splitting strategy assumes that it is not necessary to process the entire family of jobs in a single run. Therefore, it is possible to divide the jobs belonging to a single family into groups of jobs and process them after processing jobs from other families. However, the authors do not give any indication of how to divide the jobs from a family into “optimal” groups, or equivalently, how to create optimal size sublots.

This brief review of the literature indicates that little work has been done in combining NPFS and lot streaming. In this paper, we develop a mathematical model that integrates lot streaming decisions and non-permutation flow in which the creation of sublots is a key decision variable. We assume that the

entire production process can be divided into some maximum number of sublots, but in such a way that the lot sizes may vary according to changing requirements. The solution of the model provides optimal job sequences for each machine, the number of sublots for each job, the size of each sublot, and the corresponding makespan.

The rest of the paper is as follows. Section 2 presents mathematical statements of both the standard non-permutation flowshop problem and of the non-permutation flowshop with lot streaming. Section 3 describes the instances on which our formulations are tested and present our computational results. Average makespan reductions ranged from 13% up to 43%. In Section 4, we conclude with the observation that lot streaming is, indeed, an efficient makespan-reduction strategy.

## 2. Mathematical formulation

We start by formulating an analytic version of the NPFS problem without lot streaming. The latter feature is then is added to get the new model.

### *2.1 Problem description and assumptions*

The flowshop system consists of several consecutive production stages with one machine for each stage. There are multiple products, having each of different parameter settings: unitary processing time, sublot setup time, and transfer and unloading times. Sarin and Jaiprakash (2007) pointed out the importance of the last two parameters for problems of lot streaming since they are the primary determinants of operational efficiency.

With lot streaming, each job or production lot is split into some maximum number of sublots. Discrete sublots belonging to the same job are processed consecutively. It is further assumed that all jobs require the same machine sequence, but the size of sublots is allowed to vary. Only the maximum number of sublots into which a job can be divided is specified (some sublots may be empty).

Additional assumptions include:

- Machines cannot process more than one job/sublot at a time
- A sublot cannot be processed by more than one machine at a time
- All machines are available at the start of the planning horizon
- All jobs are available at the start of the planning horizon
- Machines can be idle during the planning horizon
- Preemption is not allowed

### *2.2. NPFS MILP model*

*Indexes*

$M$ Number of machines; $m = 1,2,\ldots,M$

$J$ Number of jobs (or products); $j = 1,2,\ldots,J$

*Input data*

$U_j$ Number of units of product $j$ to be processed (size of the lot of product $j$)

$pr_{m,j}$ Processing time of a unit of job $j$ on machine $m$

$ts_{m,j}$ Setup time for job $j$ on machine $m$

$tr_{m,j}$ Unloading and transfer time associated with machine $m$ after processing one or more units of job $j$ ($tr_{m,j}$ does not depend on the number of units processed)

$\Omega$ Large positive number

*Variables*

$B_{j,m}$ Start time of processing of lot of job $j$ on machine $m$,

$c_{j,m}$ Completion time of job $j$ on machine $m$

$x_{j,j',m}$ Binary variable equal to 1 when job $j$ is processed any time before job $j'$ on machine $m$, 0 otherwise ($j$ is different from $j'$).

*MILP formulation*

Minimize $Z = c_{\max}$ (1)

subject to:

$$B_{j,m} + pr_{m,j} \cdot U_j + tr_{m,j} + ts_{m,j} \le B_{j,m+1};\ j = 1,....,J, m = 1,...,M-1 \quad (2)$$

$$B_{j,m} + pr_{m,j} \cdot U_j + tr_{m,j} + ts_{m,j} \le B_{j',m} + \Omega \cdot \left(1 - x_{j,j',m}\right);\ j \ne j'; m = 1,...,M \quad (3)$$

$$B_{j',m} + pr_{m,j'} \cdot U_{j'} + tr_{m,j'} + ts_{m,j'} \le B_{j,m} + \Omega \cdot x_{j,j',m};\ j \ne j'; m = 1,...,M \quad (4)$$

$$x_{j,j',m} + x_{j',j,m} = 1;\ m = 1,....,M, j \ne j' \quad (5)$$

$$c_{j,m} \ge B_{j,m} + pr_{m,j} \cdot U_j + tr_{m,j} + ts_{m,j};\ j = 1,...,J, m = 1,...,M \quad (6)$$

$$c_{\max} \ge c_{j,m};\ j = 1,...,J, m = M \quad (7)$$

$$x_{j,j',m},\ j \ne j', m = 1,...,M;\ \text{is binary} \quad (8)$$

$$B_{j,m},\ c_{j,m} \ge 0; j = 1,...,J, m = 1,...,M \quad (9)$$

The objective function (1) is aimed at minimizing the makespan of the production process. Constraints (2) require that each job finishes its operation at a given stage before moving to the next stage. Constraints (3) and (4) act jointly and set the production sequence for each machine $m$: if job $j$ is processed any time before $j'$ then (3) becomes operative and (4) becomes redundant. The logical precedence condition is restricted by (5). If job $j$ precedes job $j'$ on machine $m$, then $j'$ cannot precede $j$ on that machine. Expression (6) places a lower bound on the completion time of each job on each machine while (7) sets the makespan $c_{\max}$ as being at least equal to the completion time of the last job on the last machine. Finally, (8) and (9) are the integrality and feasibility constraints on the variables.

*2.3. NPFS with lot streaming MILP model*

To extend model (1) – (9) to an NPFS with lot streaming it is necessary to account for the division of jobs into sublots. This requires a new index $f$ for sublots and several new variables and constraints.

*Index*

$F$ *maximum number of possible sublots per job*; $f = 1,2,...,F$

*Variables*

$A_{f,j,m}$ Arrival time of sublot $f$ of job $j$ at machine $m$

$s_{f,j}$ Number of units of job $j$ in sublot $f$

$y_{f,j}$ Binary variable equal to 1 if sublot $f$ of job $j$ is not empty, 0 otherwise

The specification of setup time now requires two different parameters. Let $tsls_{m,j}$ denote the setup time for lot streaming when a new job $j$ starts processing on machine $m$, and let $tslss_{m,j}$ denote the setup time for lot streaming when a new sublot starts processing on machine $m$. The corresponding values represent,

respectively, the change of jobs on a machine ($tsls_{m,j}$), and the minor adjustments needed on a machine for the production of a new sublot ($tslss_{m,j}$).

$$\text{Minimize } Z = c_{\max} \tag{10}$$

subject to:

$$\sum_{f=1}^{F} s_{f,j} = U_j;\ j = 1,...,J \tag{11}$$

$$s_{f,j} \le \Omega \cdot y_{f,j}; j = 1,...,J, f = 1,...,F \tag{12}$$

$$A_{f=1,j,m} + pr_{m,j} \cdot s_{f=1,j} + \left(tr_{m,j} + tslss_{m,j}\right) \cdot y_{f=1,j} + tsls_{m,j} \le A_{f,j,m};\ j = 1,...,J, m = 1,...,M, f \ge 2 \tag{13}$$

$$A_{f,j,m} + pr_{m,j} \cdot s_{f,j} + \left(tr_{m,j} + tslss_{m,j}\right) \cdot y_{f,j} \le A_{f+1,j,m};\ j = 1,...,J, m = 1,...,M, f \ge 2 \tag{14}$$

$$A_{f=1,j,m} + pr_{m,j} \cdot s_{f=1,j} + \left(tr_{m,j} + tslss_{m,j}\right) \cdot y_{f=1,j} + tsls_{m,j} \le A_{f=1,j,m+1};\ j = 1,...,J, m = 1,...,M-1 \tag{15}$$

$$A_{f,j,m} + pr_{m,j} \cdot s_{f,j} + \left(tr_{m,j} + tslss_{m,j}\right) \cdot y_{f,j} \le A_{f,j,m+1};\ j = 1,...,J, m = 1,...,M-1; f \ge 2 \tag{16}$$

$$c_{j,m} \ge A_{f,j,m} + pr_{m,j} \cdot s_{f,j} + \left(tr_{m,j} + tslss_{m,j}\right) \cdot y_{f,j};\ j = 1,...,J, f = \text{last sublot} \tag{17}$$

$$c_{j,m} \le A_{f=1,j',m} + \Omega \cdot \left(1 - x_{j,j',m}\right);\ j \ne j'; m = 1,...,M \tag{18}$$

$$c_{j',m} \le A_{f=1,j,m} + \Omega \cdot x_{j,j',m};\ j \ne j', m = 1,...,M \tag{19}$$

$$c_{\max} \ge c_{j,m};\ j = 1,...,J, m = M \tag{20}$$

$$x_{j,j',m} + x_{j',j,m} = 1;\ m = 1,...,M, j \ne j' \tag{21}$$

$$x_{j,j',m},\ y_{f,j}; j \ne j', j = 1,...,J, m = 1,...,M, f = 1,...,F, \text{ are binary} \tag{22}$$

$$A_{f,j,m} \ge 0,\ s_{f,j} \ge 0; f = 1,...,F, j = 1,...,J, m = 1,...,M; \text{ are integer} \tag{23}$$

The objective in (10) is again makespan minimization. Equations (11) define the size $s_{f,j}$ of sublots and ensure that all of product $j$, $U_j$, is distributed among the sublots. The maximum allowable number of sublots is $F$. Expression (12) forces $y_{f,j}$ to be 1 when $s_{f,j}$ is non-zero. In (13), the start time of the first sublot of job $j$ is based on the setup time $tsls_{m,j}$ and is calculated even if sublot $f$ is empty. This ensures that machine $m$ is ready to process the remaining sublots of job $j$. Constraints (14) determine the sequence of sublots. Before the next sublot can be processed, the previous one has to finish its setup, processing and transfer operations.

Constraints (15) track the arrival time of the first sublot of job $j$ at the next machine $m$+1. The left-hand side includes setup time $tsls_{m,j}$ and processing time $pr_{m,j}$ on machine $m$, and transfer time $tr_{m,j}$ to machine $m$+1. For the remaining sublots, arrival time at downstream machines is governed by constraints (16). The completion time of each job on each machine is bounded by (17), taking into consideration the completion time of the last sublot of that job. The ordering of jobs at each stage is determined by expressions (18) and (19), and work in a similar way to (3) and (4). If job $j$ is processed any time before job $j'$ then (18) ensures that job $j'$ will be processed after job $j$ has finished and (19) becomes redundant. Otherwise, the role of the expressions is reversed. Expression (20) defines the makespan $c_{\max}$ as the completion time of the last job on the last machine. Logical restrictions in (21) are the same as in (5), and ensure that job $j$ is either processed before or after job $j'$ on machine $m$. Constraints (22) and (23) define the variables.

## 3. Computational Experience

### *3.1 Experimental design*

Model (10) – (23) provides a tool for evaluating the impact of lot streaming in NPFSs. Given that the literature indicates that the effort involved in solving non-permutation instances may have a positive return when the system has more than three stages, we test our model on different versions with 5 and 10 machines, and where the number of jobs is 2, 4 and 5. The parameter data are generated following the

standard flowshop literature (e.g., see Taillard 1993). In our case, unit processing times are drawn from a uniform distribution over [1,5] and the transfer time for job/sublot from a uniform distribution in the range [1,4]. For the total number of lots, we sample from a uniform distribution over the range [20, 50]. Note that multiplying the unit processing times by lot size, the range for the processing time of the entire job is similar to that found in the flowshop literature; that is, uniformly distributed over [1,99] as in the experiments prescribed by Taillard (1993). For no lot streaming, setup times are drawn from the uniform distribution over [1,50]; for lot streaming, we sample $tsls_{m,j}$ over [1,25] and $tslss_{m,j}$ over [1,10]. For each instance, different maximum allowable sublots sizes have been tested (ranging from no splitting up to 6 sublots). This provided insights into how NPFS systems respond under different lot streaming conditions. For each set of parameters, we randomly generated and evaluated five instances. All computations were performed under Windows 7 on a PC with an Intel core i7 2.00 GHz processor and 4 GB of memory. The optimization models were solved with CPLEX 12.6 using its default settings and its 4 core processors running in parallel.

### *3.2 Results*

Table 1 indicates the size of the various instances in terms of number of constraints as well as of the number of continuous and discrete variables. The average runtime of the five tests for each instance is also indicated.

**Table 1**
Size of instances in terms of constraints, variables and average runtime

| No. of machines, *M* | No. of jobs, *J* | | No splitting | No. of possible sublots, *F* | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 2 | 3 | 4 | 5 | 6 |
| *5* | *2* | *Constraints* | 42 | 81 | 113 | 145 | 177 | 209 |
| | | *cont. var.* | 33 | 62 | 86 | 110 | 134 | 158 |
| | | *discre. var.* | 10 | 18 | 22 | 26 | 30 | 34 |
| | | *Avg. runtime* | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec |
| | *4* | *Constraints* | 164 | 241 | 305 | 369 | 433 | 497 |
| | | *cont. var.* | 105 | 162 | 210 | 258 | 306 | 354 |
| | | *discre. var.* | 60 | 76 | 84 | 92 | 100 | 108 |
| | | *Avg. runtime* | < 1 sec | < 1 sec | 2 | 8 | 58 | 420 |
| | *5* | *Constraints* | 255 | 351 | 431 | 511 | 591 | 671 |
| | | *cont. var.* | 156 | 227 | 287 | 347 | 407 | 467 |
| | | *discre. var.* | 100 | 120 | 130 | 140 | 150 | 160 |
| | | *Avg. runtime* | < 1 sec | 3 | 20 | 88 | 1250 | 14268 |
| *10* | *2* | *Constraints* | 82 | 151 | 213 | 275 | 337 | 399 |
| | | *cont. var.* | 63 | 112 | 156 | 200 | 244 | 288 |
| | | *discre. var.* | 20 | 28 | 32 | 36 | 40 | 44 |
| | | *Avg. runtime* | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec | 3 |
| | *4* | *Constraints* | 324 | 461 | 585 | 709 | 833 | 957 |
| | | *cont. var.* | 205 | 302 | 390 | 478 | 566 | 654 |
| | | *discre. var.* | 120 | 136 | 144 | 152 | 160 | 168 |
| | | *Avg. runtime* | < 1 sec | < 1 sec | 4 | 70 | 752 | 1680 |
| | *5* | *Constraints* | 505 | 676 | 831 | 986 | 1141 | 1296 |
| | | *cont. var.* | 306 | 427 | 537 | 647 | 757 | 867 |
| | | *discre. var.* | 200 | 220 | 230 | 240 | 250 | 260 |
| | | *Avg. runtime* | < 1 sec | 5 | 107 | 890 | 13550 | 60300 |

The average results are shown in Table 2, indicating the average makespan for each instance (over the five different parameter sets) as well as the average total number of sublots required but the corresponding optimal makespans. The maximum total number of sublots is defined by splitting each job into 2, 3, 4, 5 or 6 sublots depending on the instance, and solving each case. We can see that lot streaming has a clear impact on makespan minimization. For example, in the case of 2 jobs and 5 machines the makespan goes down from 1187 when no lot splitting is applied to 862 when each job is allowed to be divided into 5 sublots. This reduction represents a 25% improvement.

**Table 2**
Average results under the different splitting conditions

| No. of machines, *M* | No. of jobs, *J* | | No splitting | No. of sublots, *F* | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 2 | 3 | 4 | 5 | 6 |
| 5 | 2 | *makespan* | 1187 | 914 | 872 | 863 | 862 | 862 |
| | | *No. sublots* | | 4 | 5.8 | 6.4 | 6.6 | 6.6 |
| | 4 | *makespan* | 1832 | 1545 | 1506 | 1502 | 1502 | 1502 |
| | | *No. sublots* | | 7.8 | 9.4 | 9.8 | 9.8 | 9.8 |
| | 5 | *makespan* | 2028 | 1763 | 1729 | 1726 | 1726 | 1726 |
| | | *No. sublots* | | 8.4 | 10.8 | 11.2 | 11.2 | 11.2 |
| 10 | 2 | *makespan* | 2278 | 1556 | 1371 | 1316 | 1305 | 1302 |
| | | *No. sublots* | | 4 | 5.8 | 7.2 | 7.6 | 8 |
| | 4 | *makespan* | 2950 | 2170 | 1996 | 1950 | 1942 | 1941 |
| | | *No. sublots* | | 8 | 10.8 | 12.4 | 13.2 | 13.4 |
| | 5 | *makespan* | 3112 | 2370 | 2221 | 2189 | 2186 | 2186 |
| | | *No. sublots* | | 9.8 | 12.8 | 14.2 | 15 | 15 |

Table 3 presents the makespan reduction when different lot streaming conditions are compared with the no-splitting condition, the comparison is expressed in percentage. From Table 3, it is shown that creating sublots reduce the makespan in every instance. The improvement ranges from 13% (for 5 jobs and 5 machines) up to 43% (2 jobs and 10 machines). This indicates that lot streaming is an efficient scheduling strategy. On the other hand, the limit for improvements seems to be reached at 4 sublots per job: for 2 jobs and 10 machines, 5 possible sublots yield an incremental improvement of 1%. Once splitting has reached this limit, any additional division does not impact on makespan reduction. This effect is illustrated in Fig.1, where percentages of relative makespan reduction for all the instances with 5 machines are presented.

**Table 3**
Average relative makespan reduction compared to the no-splitting condition

| No. of machines, *M* | No. of jobs, *J* | No. of possible sublots, *F* | | | | |
|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 5 | 6 |
| 5 | 2 | 23% | 27% | 27% | 27% | 27% |
| | 4 | 16% | 18% | 18% | 18% | 18% |
| | 5 | 13% | 15% | 15% | 15% | 15% |
| 10 | 2 | 32% | 40% | 42% | 43% | 43% |
| | 4 | 26% | 32% | 34% | 34% | 34% |
| | 5 | 24% | 29% | 30% | 30% | 30% |

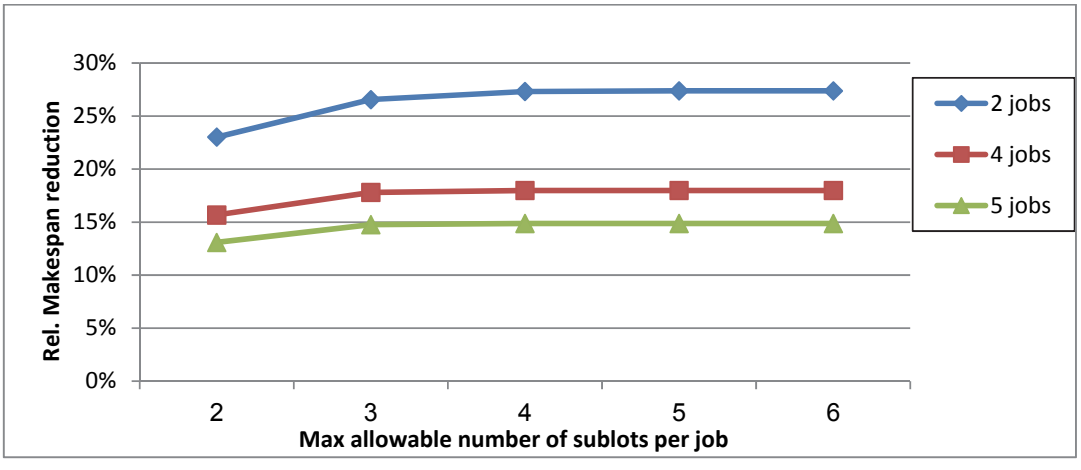


**Fig. 1.** Makespan reduction with respect to the maximum allowable number of sublots per job

Fig. 1 also shows similar behavior independently of the number of jobs: the relative makespan reduction reaches a maximum after which the use of more sublots does not affect the result. The maximum makespan reduction is different in each case: for 5 jobs it is around 15%, for 4 jobs about 18% and for 2 jobs it is over 25%. Fig 1 also shows that as the number of jobs increases, the relative makespan reduction decreases. That is, the impact of lot streaming is smaller for more jobs (products). From Table 3 we can see that the relative makespan reduction obtained with lot streaming for 2 jobs (always larger than 23%) is never reached with 4 and 5 jobs. The largest relative makespan reduction for these cases is 18% and 15%, respectively. Another observation from Table 2 is that makespan reduction depends on the number of operations required by each job. For example, consider the makespan reduction in the case of 5 jobs and 2 possible sublots. When the system involves 5 machines the relative makespan reduction with respect to the no-splitting case is 13% while for 10 machines it is 24%. For the rest of the product configurations the behavior is similar: when the number of machines increases the impact of lot streaming is increases. Fig. 2 illustrates this effect in the case of 5 jobs. Again, makespan reduction does not decrease indefinitely as more sublots are created. The main reason why improvement is bounded is related to the accumulation of fixed times in the schedule. Each new sublot for job $j$ requires minor setup changes on machine $m$ that take $tslss_{m,j}$ time and consume $tr_{m,j}$ time during transfer. These time-consuming activities increase proportionally with the number of sublots, since more sublots require more time for setup and transfer. Therefore, the benefits of lot streaming are restricted by these parameter values. If these activities were not taken into account, the optimal sublot size would be one.

The number of actual (nonempty) sublots may explain the greater impact of lot streaming on larger productive systems. Table 2 reports the number of sublots created for instances with 5 jobs and 10 machines when the maximum allowable number is 2 per job (9.8 on average). This value is larger than the average number of sublots created for the 5 machine instances (8.4 on average). For the rest of the instances the pattern of these entries is similar, for the same number of jobs with 10 machines, more sublots are used and achieve a larger relative makespan reduction than in the case with 5 machines. We can thus infer that the accumulative effect of lot streaming increases in significance as the size of the productive system grows.

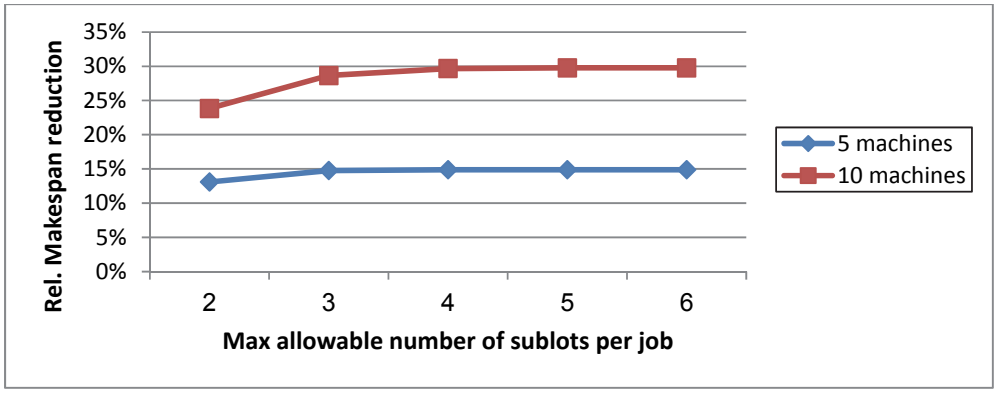


**Fig 2.** Makespan reduction with respect to the maximum allowable number of sublots per job, in the case of 5 jobs

Referring to Table 2 for the moment, we can see why a larger number of machines leads to a larger reduction in the makespan. Just consider the row corresponding to the number of sublots. Mathematically, this entry represents the average of the sum of the variables $y_{f,j}$, for the different runs. In the case of 5 jobs, 10 machines, and a maximum number of 2 sublots per job, the production lots have been split into an average of 9.8 sublots. In contrast, for 5 jobs, 5 machines and 2 sublots, the average is 8.4 sublots. The implication is that the number of sublots formed grows as the number of machines increases. This applies in all instances for the number of jobs held constant. As a consequence, we can say that the cumulative effect of lot streaming increases in significance as the size of the productive system grows. This proposition is further substantiated by the results presented in Table 4.

In Table 4, each entry indicates the percentage of sublots created with respect to the total number of potential sublots. For example, if the maximum number of possible sublots is 2 per job for the case of 2 jobs, then the maximum number of possible sublots is 4. The percentage represents the degree to which those possible sublots are used.

**Table 4**
Percentage of actual sublots compared to the total allowable number for each instance

| No. of machines, *M* | No. of jobs, *J* | No. of possible sublots, *F* = 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 5 | 2 | 100% | 97% | 80% | 66% | 55% |
| | 4 | 98% | 78% | 61% | 49% | 41% |
| | 5 | 84% | 72% | 56% | 45% | 37% |
| 10 | 2 | 100% | 97% | 90% | 76% | 67% |
| | 4 | 100% | 90% | 78% | 66% | 56% |
| | 5 | 98% | 85% | 71% | 60% | 50% |

In Table 4, we can also see that for the 10-machine instances all the percentages are equal to or larger than the respective values for the 5-machine instances. More splits induce a larger difference in makespan than with no splitting (as shown in Table 2). Note, in Table 4, that the average percentage of number of actual sublots for a given problem configuration, e.g., 10 machines and 5 jobs, decreases as *F* increases because the optimal number of sublots is reached and more sublots do not have any effect on the optimal solution. This reduces the proportion of the sublots actually used.

## 4. Conclusions

This work was motivated by a decisive lack of research on the use of lot streaming in non-permutation flowshops. The importance of this connection is twofold. First, NPFSs are a more representative model of manufacturing systems than the permutation version. Second, lot streaming has been proven to be an effective makespan reduction strategy that also reduces work-in-process and cycle time. The results of our analyses confirmed that the application of lot streaming to NPFSs effectively reduces the makespan of the production process. We have also shown that the strategy of splitting jobs has its limits. Our experiments have shown that the marginal improvement in the makespan becomes negligible as jobs are split into more than 4 sublots. Another finding of our research is that lot streaming has a larger impact on production systems as the number of stages in a flowshop increases. In accordance with this observation, the number of sublots actually used increases with the number of production stages. Again, this has a limit: for the same number of stages, the relative impact of lot streaming decreases with the number of sublots used in the process.

In future research we will consider the development of heuristics to analogously analyze large-scale instances and test other objective functions. This will involve the use of metaheuristics and optimization-based approaches such as column generation.

## References

Azzi, A., Faccio, M., Persona, A., & Sgarbossa, F. (2012). Lot splitting scheduling procedure for makespan reduction and machine capacity increase in a hybrid flow shop with batch production. *International Journal of Advanced Manufacturing Technology*, *59*(5-8), 775-786.

Davendra, D., Senkerik, R., Zelinka, I., Pluhacek, M., & Bialic-Davendra, M. (2014). Utilising the chaos-induced discrete self organising migrating algorithm to solve the lot-streaming flowshop scheduling problem with setup time. *Soft Computing*, *18*(4), 669-681.

Defersha, F. M. (2011). A comprehensive mathematical model for hybrid flexible flowshop lot streaming problem. *International Journal of Industrial Engineering Computations*, *2*(2), 283-294.

Defersha, F. M., & Chen, M. (2012). Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem. *The International Journal of Advanced Manufacturing Technology*, *62*(1-4), 249-265.

Kumar, S., Bagchi, T. P., & Sriskandarajah, C. (2000). Lot streaming and scheduling heuristics for m-machine no-wait flowshops. *Computers & Industrial Engineering*, *38*(1), 149-172.

Liao, C. J., Liao, L. M., & Tseng, C. T. (2006). A performance evaluation of permutation vs. non-permutation schedules in a flowshop. *International Journal of Production Research*, *44*(20), 4297-4309.

Lin, S. W., & Ying, K. C. (2009). Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems. *International Journal of Production Research*, *47*(5), 1411-1424.

Mehravaran, Y., & Logendran, R. (2012). Non-permutation flowshop scheduling in a supply chain with sequence-dependent setup times. *International Journal of Production Economics*, *135*(2), 953-963.

Mehravaran, Y., & Logendran, R. (2013). Non-permutation flowshop scheduling with dual resources. *Expert Systems with Applications*, *40*(13), 5061-5076.

Pan, Q. K., Tasgetiren, M. F., Suganthan, P. N., & Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information sciences*, *181*(12), 2455-2468.

Pan, Q. K., & Ruiz, R. (2012). An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. *Omega*, *40*(2), 166-180.

Pinedo, M. (2002). *Scheduling: Theory, Algorithms and Systems, 2nd ed., Prentice-Hall.*

Potts, C. N., & Baker, K. R. (1989). Flow shop scheduling with lot streaming. *Operations research letters*, *8*(6), 297-303.

Pugazhendhi, S., Thiagarajan, S., Rajendran, C., & Anantharaman, N. (2003). Performance enhancement by using non-permutation schedules in flowline-based manufacturing systems. *Computers & Industrial Engineering*, *44*(1), 133-157.

Reiter, S. (1966). A system for managing job shop production. Journal of Business, 34, 371–393.

Rossi, A., & Lanzetta, M. (2014). Native metaheuristics for non-permutation flowshop scheduling. *Journal of Intelligent Manufacturing*, *25*(6), 1221-1233.

Ruiz, R., Şerifoğlu, F. S., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, *35*(4), 1151-1175.

Sarin, S. C., & Jaiprakash, P. (2007). *Flow shop lot streaming*. Springer Science & Business Media.

Shen, L., Gupta, J. N., & Buscher, U. (2014). Flow shop batching and scheduling with sequence-dependent setup times. *Journal of Scheduling*, *17*(4), 353-370.

Sriskandarajah, C., & Wagneur, E. (1999). Lot streaming and scheduling multiple products in two-machine no-wait flowshops. *IIE transactions*, *31*(8), 695-707.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *european journal of operational research*, *64*(2), 278-285.

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

Tseng, C. T., & Liao, C. J. (2008). A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. *European Journal of Operational Research*, *191*(2), 360-373.

Vahedi-Nouri, B., Fattahi, P., & Ramezanian, R. (2013). Minimizing total flow time for the non-permutation flow shop scheduling problem with learning effects and availability constraints. *Journal of Manufacturing Systems*, *32*(1), 167-173.

Vickson, R. G. (1995). Optimal lot streaming for multiple products in a two-machine flow shop. *European Journal of Operational Research*, *85*(3), 556-575.

Wang, L. C., Chen, Y. Y., Chen, T. L., Cheng, C. Y., & Chang, C. W. (2014). A hybrid flowshop scheduling model considering dedicated machines and lot-splitting for the solar cell industry. *International Journal of Systems Science*,*45*(10), 2055-2071.

Ying, K. C. (2008). Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic. *The International Journal of Advanced Manufacturing Technology*, *38*(3-4), 348-354.

Ying, K. C., Gupta, J. N., Lin, S. W., & Lee, Z. J. (2010). Permutation and non-permutation schedules for the flowline manufacturing cell with sequence dependent family setups. *International Journal of Production Research*, *48*(8), 2169-2184.

Yoon, S. H., & Ventura, J. A. (2002). An application of genetic algorithms to lot-streaming flow shop scheduling. *IIE Transactions*, *34*(9), 779-787.

Zhang, W., Yin, C., Liu, J., & Linn, R. J. (2005). Multi-job lot streaming to minimize the mean completion time in m-1 hybrid flowshops. *International Journal of Production Economics*, *96*(2), 189-200.