An approach for the production scheduling problem when lot streaming is enabled at the operational level

Juan M. Novas[1,2]

[1]CIEM (UNC-CONICET)
Medina Allende s/n – Cdad. Universitaria
5000, Córdoba, Argentina
[2]CIDS (UTN Córdoba)
Maestro Lopez esq. Cruz Roja Argentina – Cdad. Universitaria
5000, Córdoba, Argentina
jmnovas@famaf.unc.edu.ar

Abstract. By means of the present work, the production scheduling and the lot streaming problems are simultaneously addressed at flexible manufacturing environments. The proposal is based on a Constraint Programming (CP) formulation that can efficiently tackle the scheduling of manufacturing operations and the splitting of lots into smaller sublots. The approach is capable to define the number of sublots for each lot and the number of parts belonging to each sublot, as well as the assignment of the operations on sublots to machines, with their corresponding start and completion times. The CP model can be easily adapted to cope with different problem issues and several operational policies, which constitutes the main novelty of the contribution. A set of case studies were solved in order to validate the proposal and good quality solutions were found when minimizing the makespan.

Keywords: lot streaming; scheduling; flexible job shop; constraint programming

1. Introduction

The manufacturing companies are facing a challenging era, where product life cycles are getting shorter over time and customers' demands constantly change. In this context, having an agenda of operations and knowing the availability of resources at the shop floor, have turned into crucial concerns. Thus, the production scheduling (PS) activity has been getting increasing attention from practitioners at industrial environments. PS allows the schedulers or shop managers to define an agenda for a set of production orders or jobs, which optimizes one or more performance criteria while it satisfies a set of constraints. Most academic contributions addressing the PS problem at manufacturing environments represent a job as an indivisible entity, where job splitting during the process is not permitted. However, this assumption does not always reflect what happens in practice.

At industrial facilities, a job demands to manufacture products that can be elaborated by means of one or more lots. A lot (or batch) consists of a set of similar items or parts, which require an ordered sequence of manufacturing operations to be processed (i.e. the product manufacturing route). In many real industries, lots that were predefined at the

planning level are split when facing the manufacturing process. The operational decision of dividing the lots into smaller sublots (or transfer batches) is known as lot streaming (LS) in the literature. The streaming of lots is a practice aimed at achieving an anticipated completion of smaller sets of parts. It is an operational procedure that satisfies those internal or external customers by enabling sooner deliveries of elaborated products. The time savings are due to the parallel processing of two or more consecutive operations required by a lot [1].

By solving the LS problem, the number of sublots for each lot of product and the size of those sublots are determined. The LS is normally tackled simultaneously with production scheduling. Therefore, scheduling decisions have to be extended for each sublot, turning the joined solving of PS and LS problems into a more complex issue to cope with. The PS-LS problem has many variants, depending on the characteristics being addressed. In [2], Sarin and Jaiprakash summarize PS-LS main issues:

- Single product/Multiple products. Either a single product or multiple products are considered.
- Fixed/Equal/Consistent/Variable sublots. Fixed sublots refer to the case when all sublots of all lots (of all products) have the same size for all the operations. Equal sublots mean that the number of items of each sublot is fixed for each lot. When each sublot of a lot must maintain its size during the entire processing route, the sublot is termed consistent. Instead, variable sublots can differ in size during the manufacturing process.
- Discrete/Continuous sublots. At manufacturing industries that produce discrete parts, such as cars, valves, and gears, the number of items of a sublot must be an integer. Instead, at process facilities such as pharmaceutical, paint, and gas, sublots (batches) may take either integer or continuous sizes.
- Non-idling/Intermitted idling. On the one hand, under a non-idling policy, the sublots of a lot must be processed one after the other without idle time between their processing. On the other hand, with intermitted idling, the idle time between the processing of two consecutive sublots on a machine is allowed.
- No-Wait/Wait schedules. Under a no-wait policy, when a sublot of a lot needs to be transferred to another machine for the next operation, it must be done without any delay after it has finished the preceding operation. In a wait schedule mode, a sublot may wait for processing between consecutive operations on different machines.
- Attached/Detached/No setups. If the setups cannot begin until the sublot is available at a machine, attached setups are required. Detached setups occur when the setup is independent of the availability of the sublot. In some cases, there are no setups or they are neglected.
- Intermingling/Non-intermingling sublots. In a multiproduct facility, if intermingling is allowed, the sequence of sublots of a lot j, on a given machine, can be interrupted by sublots of a lot k, with $j \neq k$. On the contrary, no interruption in the sequence of sublots of a lot is allowed at non-intermingling sublots settings.

During the last decades, the number of research works addressing the PS-LS problem has increased. It has been studied using diverse methodologies, such as mathematical models, heuristics, and meta-heuristics. The Constraint Programming (CP) [3] techniques are also a promising technology to tackle it. Mostly, the PS-LS

problem has been faced on flow-shops (FS), hybrid flow-shops (HFS), and job-shops (JS) environments, as in [1,4-6], [7-9] and [10-12], respectively.

To the knowledge of the author, there are just a few research works addressing the problem at flexible job-shops (FJS). The FJS environment is one of the more challenging plant configurations, which is an extension of the classical JS problem. In FJS, each job has a flexible processing route; i.e. each operation of a job needs to be assigned to a machine among a set of alternative resources. Concerning the FJS problem with LS (FJSP-LS), some authors [13], authors developed a multi-objective particle swarm optimization algorithm. Particularly, they tackled the consistent sublots problem. In [14], an approach for scheduling jobs at virtual manufacturing cells was presented. The authors cope with transport times and consistent sublots. A MILP model was presented, while a genetic-based algorithm was developed to solve medium size instances. In [15], authors extended their previous proposal by formalizing the problem as a MILP model and developing an island-model parallel genetic algorithm. Relevant features such as sequence-dependent setups, attached/detached setups, machine ready times and lag times were considered. A job can be split in unequal sublots. Later, in [16], an evolutionary algorithm to address the batch splitting in a dyeing facility was proposed. The approach deals with equipment capacity constraints and transition times between jobs, while a cost-based performance measure is optimized.

As it has been described, most contributions have used heuristics or metaheuristics to address the PS-LS problem and, some works, have also formalized the problem by means of a mathematical formulation. To the knowledge of the author, there are no contributions that extensively describe an approach based on CP to cope with the problem. CP techniques have received increasing attention from researchers during last years since they can efficiently solve constraint satisfaction problems (CSP) and handle combinatorial problems, especially scheduling ones [17, 18].

In the present contribution, the flexible job-shop scheduling problem and the lot streaming problem are simultaneously addressed by means of a novel CP approach. The proposal can easily handle, just by the implementation of small changes on the CP model, a variety of different FJSP-LS problems. Thus, features such as non-idling or intermitted idling, intermingling or non-intermingling sublots, no-wait or wait schedules, can be tackled. The rest of the work is organized as follows. In section II, the problem characteristics are described. Section III presents the CP model, while section IV shows the computational results when solving a set of test case studies. Finally, section V concludes and posts future challenges.

2. Problem statement

There is a set of orders requiring different products to be elaborated in an FJS environment. Each product order is associated with a lot or batch of similar parts. The number of parts that each lot contains, the lot size, has already been defined at a planning level. Each product has its manufacturing route; therefore, the parts belonging to a lot follow an ordered set of machining operations, in order to become an elaborated

product. The number and type of operations that are required to be executed on each lot also depend on the demanded product.

Lots are allowed to be split into smaller sublots of parts. An operation on a sublot can be processed on a machine that belongs to a set of alternative multipurpose machines. Thus, the processing time to execute an operation on a single part of a sublot, depends on the product being elaborated and the machine where the sublot is allocated, while the required time to process the complete sublot also depends on its size.

Fig. 1 illustrates the main difference between agendas for an FJS without LS and an FJS with LS consideration. In Fig. 1(a), a possible solution for two lots requiring three operations each, is depicted. Different arrow types represent the selected manufacturing route for each lot. Lots are not permitted to be divided. A Gantt chart shows the agenda for this case, where the label x.y stands for the number of the lot and the operation, respectively. Fig. 1(b) represents a solution for the same illustrative environment but considering lot splitting. In this case, both lots are divided into two sublots. Each sublot follows its assigned manufacturing route to turn into sublots of final products. By means of the Gantt chart it can be easily observed how a lot splitting policy leads to a better makespan (label x.y.z stands for a lot, operation and sublot). This is caused by the parallel processing of operations of sublots created from the same lot. For instance, the first operation of lot L1 is executed simultaneously over its two sublots, on machines M1 and M3.

On the one hand, the main benefits of LS, besides a shorter makespan, are: (i) products can be delivered sooner in partial quantities since sublots are fully processed in shorter periods of time than the case with no lot splitting, allowing a more agile response to customer demands (as it is illustrated in Fig.1); (ii) better use of production resources, i.e. reduction of machines idle time; (iii) reduction of work-in-process (WIP) since, generally, a lower number of parts will be waiting to be processed and these sublots will wait less time, compared to the no splitting case, (v) decrease of mean flow time. On the other hand, splitting the lots has its associated drawbacks, such as increasing setup costs and demand of transport devices.

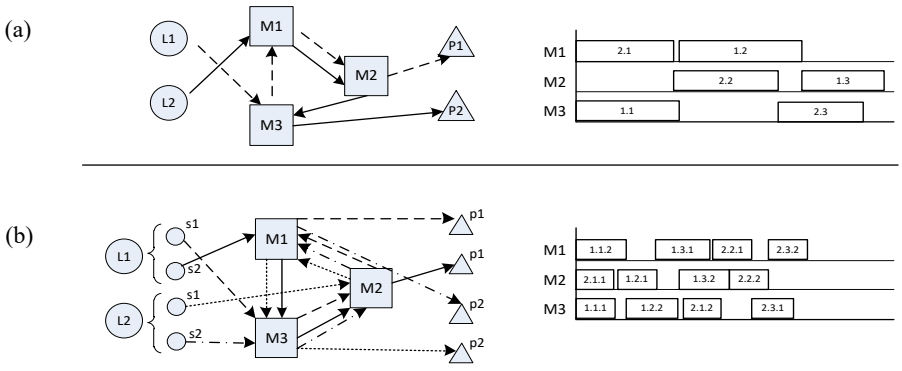

Fig. 1. (a) An FJS processing two lots without lot splitting. (b) Same FJS processing two lots with lot streaming.

The approach presented in this work is able to handle the FJSP-LS problem considering multiple products, consistent and discrete sublots. It also copes with opposite policies by simple adjustments in the formulation, such as non-idling and intermitted idling, wait and no-wait schedule, intermingling and non-intermingling sublots. Sequence-dependent detached setups, a characteristic that turns the addressed problem into a more complex one, are taken into consideration too.

The following assumptions are taken into account: (i) lots/sublots of parts are independently from each other, (ii) the demand of final products, and therefore the size of the lots, are known in advance, (iii) pre-emption is not allowed, (iv) transfer times between machines are neglected, (v) machines are capable of processing one operation at a time, (vi) machines do not receive maintenance during the scheduling horizon, (vii) disruptive events, such as rush orders or machine failures, are not taken into consideration.

When solving the FJSP-LS problem, it is required to: (i) determine the number of sublots for each lot, (ii) determine the size of each sublot, defined as the number of parts that it comprises, (iii) assign to a single machine each operation demanded by a sublot, (iv) sequence at each machine all the assigned operations, (v) determine the start and completion time of each machining operation needed by sublots. All the previous goals must be accomplished while satisfying the domain features and constraints. In this proposal, makespan was considered as the performance measure to minimize.

3. CP Formulation

3.1 Nomenclature

Set/Index

J/j	Lots (or jobs)
O/o	Operations
M/m	Machines
O_j	Operations required by lot j
$M_{j,o}$	Machines that can process operation o of lot j
S_j	Instantiable sublots of lot j. The cardinality of this set represents the maximum possible number of sublots for j.

Parameters

$pt_{j,o,m}$	The processing time of operation o on a part belonging to lot j, when it is executed on machine m
z_j	Size of lot j
sDS	Set of triplets $<j,j',s>$ representing the sequence-dependent detached setup times s, between lot j and j'.

Cumulative Function

mU_m	Represents the usage profile of machine m

Variables

cmax	Makespan
$t_{j,o,s}$	Interval variable that represents the machining operation *o* on sublot *s* of lot *j*
$tm_{j,o,s,m}$	Optional interval variable that represents the operation *o* on sublot *s* of lot *j*, executed on machine *m*
$u_{j,o,s}$	Size of sublot *s* of lot *j* on operation *o*
$spLot_{j,o,m}$	Interval variable used to span over all sublots of lot *j* under operation *o* on machine *m*
$mSpLotS_m$	Sequence variable on machine *m*, which represents the arrangement of spanned lot interval variables (*spLot*) allocated in *m*
mS_m	Sequence variable on machine *m*, which represents the arrangement of task interval variables (*t*) allocated in *m*

3.2 CP model

The CP formulation relies on the ILOG-IBM OPL language and the CP Optimizer, which underlies the CPLEX Optimization Studio [19]. These tools provide some constraints, functions, and type of variables that are used to model scheduling domain issues. Some OPL keywords, such as *alternative*, *span*, *endBeforeStart*, among others, are OPL built-in functions. Some of them were introduced and illustrated by the authors in [18] and are not explained here because of lack of space. The expressions used by the proposed model are:

$$alternative\left(t_{j,o,s}, tm_{j,o,s,m}\right); \quad \forall j \in J, \forall o \in O_j, \forall s \in S_j, \forall m \in M_{j,o} \tag{1}$$

$$mU_m = \sum_{\forall j \in J, \forall o \in O_j, \forall s \in S_j} pulse\left(tm_{j,o,s,m}, 1\right); \quad \forall m \in M_{j,o} \tag{2}$$

$$mU_m \leq 1; \quad \forall m \in M \tag{3}$$

$$endBeforeStart\left(t_{j,o,s}, t_{j,o',s}\right); \quad \forall j \in J, \forall o, o' \in O_j : o' = o + 1, \forall s \in S_j \tag{4}$$

$$endAtStart\left(t_{j,o,s}, t_{j,o',s}\right); \quad \forall j \in J, \forall o, o' \in O_j : o' = o + 1, \forall s \in S_j \tag{4'}$$

$$sizeOf\left(tm_{j,o,s,m}\right) = pt_{j,o,m} * u_{j,o,s} * presenceOf\left(tm_{j,o,s,m}\right); \quad \forall j \in J, \forall o \in O_j, \forall s \in S_j, \forall m \in M_{j,o} \tag{5}$$

$$span\left(spLot_{j,o,m}, all(s\ in\ S_j) tm_{j,o,s,m}\right); \ \forall j \in J, \forall o \in O_j, , \forall m \in M_{j,o} \tag{6}$$

$$noOverlap(mSpLotS_m); \quad \forall m \in M \tag{7}$$

$$sizeOf\left(spLot_{j,o,m}\right) \leq \sum_{s \in S_j} sizeOf\left(tm_{j,o,s,m}\right); \quad \forall j \in J, \forall o \in O_j, \forall m \in M_{j,o} \tag{8}$$

$$noOverlap(mS_m, sDS); \quad \forall m \in M \tag{9}$$

$$\sum_{s \in S_j} u_{j,o,s} = z_j \; ; \quad \forall j \in J, \forall o \in O_j \tag{10}$$

$$\sum_{s \in S_j} u_{j,o,s} \geq z_j \; ; \quad \forall j \in J, \forall o \in O_j \tag{10'}$$

$$u_{j,o,s} = u_{j,o',s} \; ; \quad \forall j \in J, \forall o, o' \in O_j : o \neq o', \forall s \in S_j \tag{11}$$

$$min\, cmax \; ; \tag{12}$$

$$cmax \geq endOf(task_{j,o,s}); \quad \forall j \in J, \forall o \in O_j, \forall s \in S_j \tag{13}$$

The assignment of sublots to machines, at each required operation, is modeled by (1). This expression ensures that an operation *o* executed on a sublot *s* of a lot *j*, is assigned to exactly one machine *m*, belonging to $M_{j,o}$. The alternative construct synchronizes each $t_{j,o,s}$ with just one instance of the optional interval variable $tm_{j,o,s,m}$, which is present in the solution.

Expressions (2) and (3) are formulated to constraint the number of operations that a machine can process simultaneously. In expression (2), the usage profile of each machine is modeled by a cumulative function. Each time an instance of the variable *tm* is present in the solution, the pulse construct enforces the mU_m function to increase its value by one unit, at the beginning of the period and returns to zero when the task associated with *tm* finishes. Expression (3), jointly with (2), ensures that only one sublot is processed at each unit of time on machine *m*.

Expression (4) defines the precedence relationships between consecutive operations required by each sublot of each lot. The *endBeforeStart* construct is used, which forces that the operation *o* on a sublot *s* of a lot *j*, to end before the start of the next operation *o'* required by the same sublot. This accounts for the wait schedule operational mode. If a no-wait policy is considered, then the expression (4') must be replaced by the expression (4). By (4'), the *endAtStart* construct ensures that the later operation starts without any delay after the preceding task has finished.

Expression (5) uses the construct *sizeOf* to define the duration each task. The processing time of the activity $tm_{j,o,s,m}$ depends on the processing time required by an individual part of the original lot *j* on machine *m*, when executing *o*, $pt_{j,o,m}$, and the number of parts that constitutes the sublot *s*, $u_{j,o,s}$. The duration of $tm_{j,o,s,m}$ can adopt a zero value when (i) the instance of that variable is not present in the solution, $presenceOf(tm_{j,o,s,m}) = 0$, which means that task $t_{j,o,s}$ is not assigned to machine *m*, or (ii) the sublot is not instantiated, meaning that it is empty, $u_{j,o,s} = 0$. Otherwise, the size of the task is greater than zero.

Previous constraints (1-5) are used to address the FJSP-LS under the intermitted idling policy, by which the idle time between the machining of two consecutive sublots on a machine is allowed. It is also permitted the processing, on different machines, of the same operation on different sublots.

Under an intermitted idling policy, intermingling can be allowed or forbidden. Constraints (1-5) also permit the intermingling policy, by means of which a sequence

of sublots of a lot on a machine can be interrupted by sublots of a different lot. On the contrary, under a non-intermingling policy, the expressions (6-7) have to be considered. These expressions enforce that a sequence of sublots of a lot is not interrupted by any sublot from a different lot. While by (6) an auxiliary task $spLot_{j,o,m}$ is instantiated to comprise all sublots of lot j on machine m under operation o, expression (7) ensures that those *spLot* intervals do not overlap each other, i.e. an operation o of a sublot of a lot k cannot be scheduled between tasks on sublots corresponding to another lot j, with $j \neq k$.

If no-idling policy is required to be modeled, then expressions (1-7) has to be considered and expression (8) must be added. By a no-idling mode, different sublots of a lot must be processed consecutively when they are allocated to the same machine, without idle time between tasks. The expression (8) forces the $spLot_{j,o,m}$ interval to have the same size as the sum of the durations of operations o, on all sublots belonging to j, on machine m. Thus, no idle time can exist between to $tm_{j,o,s,m}$ intervals.

The expression (9) accounts for the sequence-dependent detached setup times. A setup time interval exists every time two consecutive sublots in a machine sequence, belong to different lots (in this work it is assumed that lots pertain to different products). The duration of the setup depends on which product is being elaborated previously and which is processed next on a machine. To model this feature, the *noOverlap* construct is used. It enforces tasks assigned to machine m to not overlap each other and, when sublots are from different lots, imposes a period of time between those tasks that represent the predefined setup time.

A sublot s belonging to S_j can be either empty or not, that is a CP model decision to make. Not all declared elements in the set S_j will necessarily have a positive value, i.e. not all sublots s of lot j will be instantiated in the solution, some of them can be empty. The number of declared sublots for each j instantiated and the number of parts belonging to it (size of s), are decisions addressed by the CP model. Note that the cardinality of S_j defines the maximum number of sublots in which lot j can be split.

By means of (10), it is ensured that the sum of all the sublots sizes belonging to a lot must be equal to the size of that original lot. In some practical situations, this equality needs to be relaxed, as in expression (10'), in order to obtain feasible solutions. For instance, when there is a minimum requirement on the number of parts belonging to a sublot and the size of the lot is not a multiple of that parameter. Expression (11) enforces sublots of each lot to be consistent. This means that, even when sublots of a lot can have different sizes, each sublot maintains the same size during the whole manufacturing route.

In the present CP model, makespan has been chosen as the performance measure to minimize. Then, expressions (12) and (13) must be included in the formulation. Other objective functions, such as total tardiness or multi-objective functions, can be easily considered in the model and will be discussed in future works.

4. Computational results

4.1 Data and scenarios

The CP formulation presented in Section 3 has been tested with several case studies of different size and characteristics. To the best of the author's knowledge, there is a lack of benchmark problem instances in the literature for the FJSP-LS problem (with its corresponding complete set of data). Hence, the test instances addressed in this work have been generated adapting several of the FJSP without LS case studies introduced in [20], whose main characteristics are summarized in Table 1.

In [20], the processing times are expressed in terms of job, operation and machine. In this work, those values are considered as the processing time that requires an operation on a single part belonging to a lot, on a given machine. Therefore, the duration of an operation depends also on the size of the sublot being manufactured.

The problems in Table 1 were extended to consider the LS issue, by defining the maximum number of sublots for each lot (S_j) and the size of each lot (z_j). The size z_j was randomly generated between 5 and 50. Three sets of data per problem instance were defined by varying the values of z_j. Each of these instances is identified as Px-z, where x-z are the problem and the data set identifiers, respectively. The value of the S_j parameter was defined as 4 and 6, for small and medium-size problems, respectively, except for the case when a different value is explicitly indicated.

Table 1. Number of lots, operations, shop machines, and alternative machines per lot-operation

ID	ID in [20]	#J	#O	#M	$\#M_{j,o}$
P1	SFJS7	3	3	5	2
P2	SFJS8	3	3	4	2
P3	SFJS9	3	3	3	2
P4	SFJS10	4	3	5	1 - 2
P5	MFJS7	8	4	7	2 - 3
P6	MFJS8	9	4	8	2 - 3
P7	MFJS9	11	4	8	2 - 3
P8	MFJS10	12	4	8	2 - 3

A main contribution of this approach is its capability to address several FJSP-LS types of problems. With the aim of addressing these variants, a set of 10 possible scenarios were defined as FJSP-LS problems at shops operating under different operational characteristics and policies. Thus, a scenario is related to a precise set of constraints that models its features. Table 2 presents a classification of the scenarios, the features that typify them, and the associated expressions that must be considered in the formulation. Group A scenarios represent the possible combinations of the LS problem characteristics identified in the literature and enumerated in Section 1. The

scenarios of group B result from the combination of any scenario from group A, and constraint (9). For instance, S1.6 addresses a problem characterized by intermitted idling, wait schedule, intermingling sublots, and sequence-dependent setup times.

The case studies were solved to a maximum time limit that depends on its size. A desktop computer consisting of an Intel i7-7700 CPU, 3.60 GHz processor with 16 Gb of RAM was used.

Table 2. Set of the addressed scenarios

Group	Scenario ID	Operational features	Expressions
A	S1	Intermitted idling/Wait schedule/ Intermingling policy	1,2,3,4,5,10,11,12,13
	S2	Intermitted idling/No-wait schedule/ Intermingling policy	1,2,3,4',5,10,11,12,13
	S3	No-idling/ Wait schedule/ Non-intermingling policy	1,2,3,4,5,6,7,8,10,11,12,13
	S4	No-idling/No-wait schedule/ Non-intermingling policy	1,2,3,4',5,6,7,8,10,11,12,13
	S5	Intermitted idling/Wait schedule/ Non-intermingling	1,2,3,4,5,6,7,10,11,12,13
B	SA.6.	Sequence Dep. Setup	any from A & 9

4.2 Results

Table 3 shows the solutions when addressing the entire group A scenarios, for 4 medium size problem instances. A CPU time limit of 5000 seconds was imposed.

The solutions for S1 scenarios reach the lower makespan among all other scenarios for each problem instance, while the scenarios S4 are the ones with the highest makespan value. This behavior is related to the fact that S4 is the most constrained scenario since it forbids any idle time between consecutive operations on sublots and between sublots of a job in a given machine. On the contrary, the S1 scenario is the most flexible, allowing idle time among operations and sublots sequences. For all the test problems solved, the performance of the agenda degrades between 29% and a 33%, when solutions from S1 and S4 are compared.

The first solutions for the whole set of problems were instantiated in less than 2 seconds of CPU time. Before the imposed time limit of 5000 seconds, all solutions improve its first instantiation between 17% and 57%, with an average of 33%. In most problems, good solutions are found in short CPU times. Problems reach good quality solutions before 500 seconds. After that point, the ratio of improvement gets shorter with the time. In most cases, the later in time a new solution is found, the smaller the improvement of the solution is. This behavior replicates in all the problems and scenarios.

Table 3. Solutions for different medium-size problems with diverse lot streaming characteristics

ID	Scenario	First solution	Best solution	
		Makespan [a]	Makespan	CPU time (s)[b]
P5.3	S1	35631	25142	3480
	S2	34684	28797	2660
	S3	46468	25594	3098
	S4	43177	32337	1435
	S5	62348	26911	4756
P6.3	S1	37444	27478	3342
	S2	52196	30561	3110
	S3	49981	28699	4623
	S4	50080	35192	2697
	S5	57561	31072	3393
P7.3	S1	38156	29939	4377
	S2	44967	34880	4111
	S3	52646	32712	3910
	S4	56436	39901	4354
	S5	56670	32700	4781
P8.3	S1	40592	30267	2850
	S2	47658	37344	1411
	S3	47828	30725	4861
	S4	66417	40310	662
	S5	44039	32864	4856

[a] All first solutions were found in less than 2 seconds of CPU time
[b] Time required to instantiate the solutions within the 5000 CPU seconds

The CP approach allows considering sequence-dependent detached setup times. When this feature is addressed, the impact on the obtained solutions depends on the relation between the setup times and the duration of manufacturing tasks, which in turn depends on the size of the sublot being processed. An extended analysis of test instances considering setups is not included in this contribution. As an illustrative example, solution for problem P8.3, scenario 5, when the setup issue is present (S5.6), shows a Makespan of 42950 units of time. This solution was the best found within 5000 seconds of CPU, and it was instantiated at 2839 seconds. As it can be observed, in this particular case, the setups considerably degrade the Makespan value obtained when they are missed.

For two of the solved problem instances, Table 4 compares their solutions when addressing the problem with and without lot streaming consideration. In the P3.1-S2 case, the makespan reached considering LS outperforms the objective value when it is not permitted, in a 28%. If the processing times are assumed to be in seconds, as they

are in [20], then, by allowing the splitting of lots the saved time for the P3.1-S2 problem was 28 minutes. For P5.1-S1, the improvement is of 29%, representing 124 minutes in that case. Furthermore, the first delivery (first sublot completed) corresponds to the *j4* at time 3782, while without LS, the first completion corresponds to *j1* at time 13260. This situation clearly shows one of the main advantages of LS, which is the sooner finishing of some sublots of parts. Fig. 2 shows the Gantt charts of the obtained solutions for the P5.1 S1 case. In Fig. 2 (a), lot streaming is forbidden and there is a single lot per demanded product. In Fig. 2 (b), lot streaming is allowed and there are several sublots per lot of product.

Table 4. Comparison of solutions for problems that allow or forbid LS

ID	Scenario	LS[b]	First solution		Best solution	
			Makespan	CPU time (s)	Makespan	CPU time (s) [a]
P3.1	S2	A	7660	1	5986	665
		F	7680*	1	NI	-
P5.1	S1	A	33368	1	25389	632
		F	32852*	1	NI	-

[a] Time required to instantiate the solutions within the 1500 CPU seconds
[b] A-Lot streaming is allowed / F-Lot streaming is forbidden
* A single solution was found within the time limit
NI No improvement in the already found solution.

Fig. 2. Gantt charts representing the obtained schedules for instance P5.1 S1, when (a) lot streaming is forbidden, and when (b) lot streaming is allowed. There are 8 lots to be manufactured in a shop with 7 machines. Each lot (with different colors) requires 4 operations, each of which can be processed in a set of 2 or 3 alternative machines.

Notice that the approach does not set a lower bound on the number of parts that each sublot can comprise. A strategy to cope with this issue is part of the future research work.

5. Conclusions and future directions

A novel CP formulation that addresses the complex FJSP-LS problem has been presented. The approach is able to easily cope with the complexity of the FJSs and, by means of few changes, it can address many different characteristics of the lot streaming problem. Several operational policies such as idling/ intermitted idling, wait/no-wait schedules, etc., can be handled by the proposal. The model has been tested with several problem instances adapted and extended from the literature of FJSP without LS. Good quality solutions were found for small- and medium-size case studies when minimizing makespan.

There are several challenges related to the scheduling and lot streaming problem that are going to be tackled as part of future research activities. The use of multi-objective performance measures will be discussed. Also, different methods to handle real large-size problems will be studied, such as MILP/CP integrated approaches or heuristics. Some rules observed in practice will also be analyzed.

At some manufacturing settings the lots are streamed at some point in the manufacturing route and, downstream, consolidated again. This is a complex issue to efficiently cope with and, therefore, an interesting research challenge to tackle as part of future activities.

Acknowledgments

The author wishes to acknowledge the financial support received from ANPCyT (PICT-2015-3743) and UTN (PID-4526).

References

[1] A.A Kalir and S.C. Sarin, “Evaluation of the potential benefits of lot streaming in flow-shop systems”, International Journal of Production Economics, 66, pp. 131-142, 2000.
[2] S.C. Sarin and P. Jaiprakash, Flow Shop Lot Streaming, Springer US, 2007.
[3] K. Marriot and P. Stuckey, “Programming with constraints. An introduction.” Cambridge, Massachusetts: The MIT Press, 1999.
[4] Q-K. Pan and R. Ruiz, “An estimation of distribution algorithm for lot-streaming flow shop problems with setup times”, Omega, 40, pp. 166-180, 2012.
[5] D. Davendra, R. Senkerik, I. Zelinka, M. Pluhacek, and M. Bialic-Davendra, “Utilising the chaos-induced discrete self organising migrating algorithm to solve the lot-streaming flowshop scheduling problem with setup time”, Soft Computing, 18, pp. 669-681, 2014.

[6] D. Rossit, F. Tohmé, M. Frutos, J. Bard, and D. Broz, "A non-permutation flowshop scheduling problem with lot streaming: A Mathematical model", International Journal of Industrial Engineering Computations, 7, pp. 507-516, 2016.
[7] M. Cheng and S.C. Sarin, "Two-stage, Multiple-lot, Lot Streaming Problem for a 1 + 2 Hybrid Flow Shop", in IFAC Proceedings Volumes. 7th IFAC Conference on Manufacturing Modelling, Management, and Control, 46, 448-453,2013.
[8] M. Nejati, I. Mahdavi, R. Hassanzadeh, N. Mahdavi-Amiri, and M. Mojarad, "Multi-job lot streaming to minimize the weighted completion time in a hybrid flow shop scheduling problem with work shift constraint", Int. J. Adv. Manuf. Technol, 70, 501-514,2014.
[9] L.C. Wang, Y. Y. Chen, T.L. Chen, C.Y Cheng, and C.W. Chang, "A hybrid flowshop scheduling model considering dedicated machines and lot-splitting for the solar cell industry", International Journal of Systems Science,45, 2055-2071,2014.
[10] T.C. Wong, F.T.S. Chan, and L.Y. Chan, "A resource-constrained assembly job shop scheduling problem with Lot Streaming technique", Computers & Industrial Engineering, 57, pp. 983-995,2009.
[11] D. Lei and X. Guo, "Scheduling job shop with lot streaming and transportation through a modified artificial bee colony", International Journal of Production Research, 51, 4930-4941,2013.
[12] X.Q. Xu and D.M. Lei, "Research on swarm intelligence algorithm with an artificial bee colony algorithm for lot streaming problemin job shop", Advanced Materials Research, 951, 239-244,2014.
[13] B. Jun-Jie, G. Yi-Guang, W. Ning-Sheng, and T. Dun-Bing, "An Improved PSO Algorithm for Flexible Job Shop Scheduling with Lot-Splitting", International Workshop on Intelligent Systems and Applications, Wuhan, 23-24 May 2009, 1-5,2009.
[14] S.E. Kesen and Z. Güngör, "Job scheduling in virtual manufacturing cells with lot-streaming strategy: a new mathematical model formulation and a genetic algorithm approach", Journal of the Operational Research Society, 63, 683-695,2012.
[15] F. M. Defersha and M. Chen, "Jobshop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time", International Journal of Production Research, 50, 2331-2352,2012.
[16] X. Xu, L. Li, L. Fan, J. Zhang, X. Yang, and W. Wang, "Hybrid Discrete Differential Evolution Algorithm for Lot Splitting with Capacity Constraints I n Flexible Job Scheduling," Mathematical Problems in Engineering, vol. 2013, Article ID 986218, 10 pages, 2013.
[17] P. Baptiste, C. Le Pape, and W. Nuijten, "Constrained-Based Scheduling: Applying Constraint Programming to Scheduling Problems". Springer, New York, 2005.
[18] F. Novara, J.M. Novas, and G.P. Henning, "A novel constraint programming model for large-scale scheduling problems in multiproduct multistage batch plants: Limited resources and campaign-based operation", Computers & Chemical Engineering Journal, 93, 101-117, 2016.
[19] IBM ILOG CPLEX Optimization Studio. https://www.ibm.com/products/ilog-cplex-optimization-studio. Last access: 30/04/2018.
[20] P. Fattahi, M. Saidi Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems", Journal of Intelligent Manufacturing, 18, pp. 331-342, 2007.