Modelo MILP para la Programación de la Producción en Ambientes Job-Shop Flexibles con División de Lotes

Juan M. Novas

CIEM (Universidad Nacional de Córdoba - CONICET), Medina Allende s/n,
Cdad. Universitaria, Córdoba, Argentina
jmnovas@famaf.unc.edu.ar

Abstract. Mediante la presente propuesta se aborda el problema de la programación de tareas de manufactura en ambientes job-shop flexibles, considerando posible la división de lotes de trabajo en sublotes de menor tamaño. Ante un dado problema, el modelo desarrollado permite establecer si es necesaria la división de cada lote y, en caso positivo, obtener el tamaño óptimo de los sublotes generados. La propuesta determina los tiempos de inicio y fin de las operaciones de manufactura que se realizan sobre los sublotes, así como también asigna dichos sublotes a equipos adecuados. Se presenta aquí un modelo mixto entero lineal (MILP), el cual fue puesto a prueba mediante la resolución de distintos casos de estudio, obteniendo buenos resultados en bajos tiempos de CPU.

Keywords: scheduling, programación de la producción, programación matemática, job-shop flexible.

1 Introducción

Los entornos de trabajo conocidos como “job-shop” flexibles (“Flexible Job-Shop”, FJS) se caracterizan por tener máquinas que pueden realizar más de un tipo de operación (equipos multipropósito) y por contar con más de un equipo como alternativa para ejecutar cada proceso de maquinado. Es esta última característica la que diferencia a los FJSs de los “job-shops” tradicionales, donde la ruta de cada orden de trabajo o “job” se encuentra definida de antemano [1, 2].

El problema de programación de operaciones (“scheduling”) en entornos FJSs consiste en, para un dado conjunto de “jobs”, establecer una agenda para las actividades de manufactura requeridas por cada trabajo (tiempos de inicio y fin de cada tarea) y definir qué equipo ejecutará cada operación de maquinado. La complejidad de este problema se identifica como NP-Hard.

En ambientes FSJs, se define una orden de trabajo como un conjunto de partes o ítems iguales, también conocido como “batch” o lote, que requieren de los mismos procesos para transformarse en un producto terminado. Normalmente se ha tratado el problema de programación de la producción en FSJs considerando que cada lote atraviesa todas las operaciones requeridas como una unidad indivisible, cuya división en sublotes de menor tamaño no está permitida. Sin embargo, esta suposición no siempre

es válida, ya que existe una gran variedad de plantas industriales donde los lotes son aptos a ser divididos.

La separación de los lotes en sublotes más pequeños ("Lot Streaming", LS), permite que operaciones sucesivas demandadas por un mismo lote puedan superponerse temporalmente, siempre que se trate de distintos sublotes asignados a máquinas diferentes. Este procesamiento en paralelo de sublotes permite completar anticipadamente todo el proceso de maquinado vinculado a un dado lote. No obstante, determinar la cantidad y composición de los sublotes mientras se resuelve simultáneamente la programación de las tareas de manufactura requeridas, hace aún más complejo el problema.

En los últimos años se han realizado algunas contribuciones que abordan la temática de "scheduling" considerando la división de lotes. Estos aportes se han enfocado principalmente en los ambientes "flow-shop" [3, 4, 5], siendo escasos los trabajos sobre entornos "job-shop" [6, 7]. Sólo se ha encontrado una contribución que aborda la temática en plantas de tipo "job-shop" flexibles [8].

El presente aporte propone un modelo mixto entero lineal (MILP) que aborda el problema de la programación de operaciones en entornos "job-shop" flexibles, considerando la división de lotes (FJS-LS). En la sección 2 se describe el problema abordado. En la sección 3 se presenta el modelo desarrollado y, en la sección 4, los resultados alcanzados al resolver diferentes casos de estudio adaptados desde ejemplos extraídos de la literatura. Finalmente, se concluye sobre el aporte realizado y se establecen actividades de investigación futuras.

2 Definición del problema

Dada una determinada demanda de productos a elaborar en un ambiente de producción FJS, y con el fin de satisfacer la misma, se define un conjunto de órdenes de trabajo. Cada orden o "job" se vincula a un lote de partes a procesar. Cada lote recibirá todas las operaciones que sean requeridas para su transformación en producto final. Las operaciones que se ejecutan sobre cada lote dependen, en cantidad y tipo, del producto a elaborar.

Además, cada operación requerida por cada lote puede ser procesada en un equipo perteneciente a un conjunto de máquinas alternativas. El tiempo de procesamiento de cada operación sobre cada parte, depende entonces del equipo al que se asigne el lote que contiene dicha parte. En entornos FJS, las máquinas multipropósito pueden ejecutar diferentes tipos de operaciones, pero sólo pueden procesar una tarea u operación a la vez.

En este trabajo se considera que el entorno FJS que se aborda soporta la división de los lotes en sublotes, lo cual permite reducir el tiempo de finalización de las órdenes de producción. A pesar de que existen diversos aspectos a considerar al tratar un problema de LS, clasificación que se detalla en [5,7], esta propuesta se limita a tratar los problemas que poseen las siguientes características:

- Sublotes consistentes, los sublotes de cada lote pueden tener diferente tamaño, pero lo conservan durante toda la ruta de procesamiento,
- Sublotes discretos, los componen partes o piezas discretas,
- No existen tiempos de set-up considerables,
- No se permite intercalar sublotes de distintos lotes, es decir, está prohibido que una secuencia de sublotes de un lote j sea interrumpida por sublotes de un lote q, con $j \neq q$,
- El tiempo entre sublotes consecutivos de un lote en un equipo puede estar prohibido o habilitado,
- El tiempo de espera entre operaciones sucesivas de un mismo sublote puede encontrarse prohibido o habilitado.

A partir de las características del problema presentadas previamente, se plantea que la programación de operaciones en un entorno FJS, considerando la división de lotes, implica: (i) determinar la cantidad de sublotes en los que se divide cada lote, (ii) establecer el tamaño de cada sublote (cantidad de partes que contiene), (iii) asignar una máquina a cada operación requerida por cada sublote, (iv) secuenciar en cada equipo las operaciones de cada sublote, (v) especificar el tiempo de inicio y fin de cada operación de manufactura, sobre cada sublote. La duración de cada operación de cada sublote dependerá del equipo asignado y de la cantidad de partes que compongan el sublote.

La presente contribución realiza las siguientes suposiciones: (i) cada lote de partes es independiente del resto, (ii) no se consideran tiempos de traslado entre máquinas (el traslado entre equipos es inmediato), (iii) la interrupción de las operaciones no se encuentra permitida, (iv) las máquinas no se averían ni reciben mantenimiento durante el horizonte de planificación, (v) no se producen fallas en la partes durante su producción, así como tampoco ningún otro evento disruptivo de la agenda, (vi) la demanda de cada producto y por ende los lotes a procesar, se conocen de antemano, (vii) existe una cantidad máxima de sublotes permitidos por cada lote y ésta es conocida de antemano.

En cuanto a las medidas de desempeño empleadas, se han considerado dos funciones objetivo a minimizar, vinculadas a la eficiencia de la agenda de operaciones: (i) makespan, que consiste en el tiempo de finalización de la operación más tardía, y (ii) el tiempo de finalización total de los sublotes, definido como la suma de los tiempos de finalización de la última operación, de todos los sublotes, de todos los lotes (“Sublots Total Completion Time”, STCT), y (iii) la tardanza de todos los lotes.

3 Descripción del Modelo

El modelo fue implementado mediante el lenguaje OPL de ILOG-IBM, en el entorno ILOG CPLEX Optimization Studio 12.5 [9]. A continuación se presenta la nomenclatura empleada, para luego describir las restricciones que componen la formulación, así como las medidas de desempeño consideradas.

3.1 Nomenclatura

Conjuntos/Índices

J/j	Lotes (también denominados "jobs" o trabajos)
I/i	Operaciones
K/k	Máquinas
I_j	Operaciones demandadas por el lote j
$K_{i,j}$	Máquinas con capacidad de procesar la operación i sobre el lote j
S_j	Posibles sublotes para el lote j. Su cardinalidad representa el número máximo de sublotes en que puede dividirse j.

Parámetros

$pt_{k,i,j}$	Tiempo de procesamiento de una unidad del lote j en la operación i en la máquina k
d_j	Cantidad de unidades o partes a elaborar demandadas por j
dd_j	Fecha de entrega o "duedate" del lote j
M	Número entero suficientemente grande

Variables

$Cmax$	Makespan
T_j	Tardanza del lote j respecto a su fecha de entrega.
$W_{k,s,i,j}$	Variable binaria que asume el valor 1 cuando la operación i del lote j es ejecutada mediante el sublote s en la máquina k, y 0 de otra manera.
$X_{k,i,j}$	Variable binaria que asume el valor 1 cuando la operación i del lote j es ejecutada en la máquina k, independientemente del sublote sobre el que se ejecuta i; y 0 de otra manera.
$U_{s,i,j}$	Tamaño del sublote s sobre el que se ejecuta la operación i del lote j.
$Cs_{s,i,j}$	Tiempo de finalización de la operación i sobre el sublote s del lote j.
$Ss_{s,i,i}$	Tiempo de inicio de la operación i sobre el sublote s del lote j.
$C_{i,j}$	Tiempo de finalización de la operación i sobre todo el lote j.
$S_{i,j}$	Tiempo de inicio de la operación i de todo el lote j.

3.2 Modelo

Para la asignación de operaciones a equipos, se utilizan tres expresiones. Mediante la expresión (1), se asigna una operación i de un dado lote j, a una única máquina k entre aquellas que cuentan con capacidad para ejecutar la actividad i de dicho j. Como se observa, la variable X_{kij} es independiente de los sublotes en que pueda dividirse el lote j. Por medio de la expresión (2), se asignan los sublotes s de cada lote j, en cada operación i, a una única máquina k. La expresión (3) es utilizada para vincular las variables empleadas en (1) y (2), asegurando que si una operación i está asignada a una máquina k, entonces todos los sublotes s del lote j están asignados a la misma máquina k.

$$\sum_{k \in K_{ij}} X_{kij} = 1 \;\; ; \;\; \forall i \in I_j, \forall j \in J \tag{1}$$

$$\sum_{k \in K_{ij}} W_{ksij} = 1 \;\; ; \;\; \forall i \in I_j, \forall s \in S_j, \forall j \in J \tag{2}$$

$$W_{ksij} = X_{kij} \;\; ; \;\; \forall j \in J, \forall i \in I_j, \forall s \in S_j, \forall k \in K_{ij} \tag{3}$$

El siguiente par de desigualdades tienen por objeto establecer los tiempos de procesamiento de cada sublote *s* del lote *j* cuando sobre éste se ejecuta la operación *i* en la máquina *k*. Notar que el tiempo de procesamiento de todo el sublote *s* de *j* en la operación *i* dependerá del tiempo requerido por cada parte, pt_{kij}, y la cantidad de partes que componen dicho sublote, U_{sij}.

$$Cs_{sij} \geq Ss_{sij} + pt_{kij} \,.\, U_{sij} - \left(1 - W_{ksij}\right) M \;; \quad \forall s \in S_j \,, \forall i \in I_j \,, \forall k \in K_{ij} \,, \forall j \in J \tag{4}$$

$$Ss_{sij} \leq Cs_{sij} + W_{ksij} \,.\, M \;; \quad \forall s \in S_j \,, \forall i \in I_j \,, \forall k \in K_{ij} \,, \forall j \in J \tag{5}$$

La expresión (6) asegura la precedencia entre operaciones sucesivas, *i* e *i'*, demandadas por un mismo sublote *s* del lote *j*. La desigualdad refleja una política de trabajo donde los tiempos de espera entre operaciones están permitidos. En el caso que se requiera modelar esta precedencia en una planta donde la espera entre operaciones sucesivas esté prohibida, la desigualdad en (6) debe modificarse por una igualdad.

$$Ss_{si'j} \geq Cs_{sij} \;\; ; \;\; \forall s \in S_j \,, \forall i, i' \in I_j \,| i < last\left(I_j\right) \wedge i' = next(i), \forall j \in J \tag{6}$$

También se hace necesario asegurar la precedencia entre sublotes consecutivos, *s* y *s'*, pertenecientes al conjunto de sublotes de *j* y sobre los que se ejecuta una misma operación *i*, lo cual queda establecido por la expresión (7). Esta expresión es válida cuando se permite que exista tiempo libre entre los sublotes *s* y *s'*, mientras que cuando éste está prohibido, se debe reemplazar la desigualdad por una igualdad.

$$Ss_{s'ij} \geq Cs_{sij} \;\; ; \;\; \forall s, s' \in S_j | s < last\left(S_j\right) \wedge s' = next(s) \,, \forall i \in I_j \,, \forall j \in J \tag{7}$$

Las expresiones (8) y (9) tienen por fin sincronizar el inicio y fin de cada operación *i* de *j*, con el inicio de la operación *i* sobre el primer sublote *s* de *j* y el fin de la operación *i* sobre el último sublote *s* de *j*, respectivamente.

$$S_{ij} = Ss_{sij} \;\; ; \;\; \forall j \in J, \forall i \in I_j, \forall s \in S_j | s = first\left(S_j\right) \tag{8}$$

$$C_{ij} = Cs_{sij} \; ; \quad \forall j \in J, \forall i \in I_j, \forall s \in S_j | s = last(S_j) \tag{9}$$

Por las características del problema abordado, es posible definir la secuencia entre las operaciones de los trabajos, sin tener en cuenta los sublotes, ya que no hay sublotes de diferentes lotes intercalados en una misma secuencia de operaciones.

La sucesión de las operaciones sobre un dado equipo *k*, se logra mediante las expresiones lógicas (10) y (11), las cuales son soportadas por la herramienta de modelado empleada. El uso de las mismas asegura que no exista superposición entre las distintas actividades de maquinado asignadas a un mismo equipo, a la vez que evita crear una variable dedicada a la secuencia para lograrlo. La expresión (10) modela la secuencia de operaciones de distintos lotes sobre un dado equipo *k*; mientras que la (11) hace lo propio para operaciones sucesivas de un mismo trabajo. Estas restricciones, junto a las previas, aseguran la correcta secuencia de las tareas.

$$\begin{gathered} X_{kij} = 1 \; \wedge \; X_{klj'} = 1 \rightarrow S_{lj'} \; \geq \; C_{ij} \; \vee \; S_{ij} \geq \; C_{lj'}; \\ \forall i \in (I_j \; \cap I_{j'}), \forall k \in (K_{ij} \cap K_{lj'}), \forall j, j' \in J, j \neq j' \end{gathered} \tag{10}$$

$$\begin{gathered} X_{kij} = 1 \; \wedge \; X_{ki'j} = 1 \rightarrow S_{i'j} \; \geq \; C_{ij} \; ; \\ \forall j \in J, \forall k \in (K_{ij} \cap K_{i'j}), \forall i, i' \in I_j \mid i < last(I_j) \; \wedge \; i' = next(i) \end{gathered} \tag{11}$$

Las expresiones (12) y (13) permiten definir el tamaño de los sublotes para cada lote. La restricción (12) establece que el sublote *s* de *j*, en la operación *i*, se compone de cero o más partes. Cuando U_{sij} toma el valor cero, el sublote referido no existe en la solución (no contiene partes). La expresión (13), por su parte, busca que la suma de las unidades entre todos los sublotes *s* de cada lote *j*, en cada operación *i*, cumpla con la demanda de unidades requeridas por *j* al final del proceso.

$$U_{sij} \geq 0 \; ; \quad \forall s \in S_j, \forall i \in I_j \, , \forall j \in J \tag{12}$$

$$\sum_{s \in S_j} U_{sij} \geq d_j \; ; \quad \forall i \in I_j \, , \forall j \in J \tag{13}$$

La siguiente expresión establece sublotes consistentes; es decir, cada sublote *s* de *j*, conserva su tamaño a medida que atraviesa todas las operaciones que requiera.

$$U_{sij} = U_{si'j} \; ; \quad \forall s \in S_j, \forall j \in J, \forall i, i' \in I_j \mid i < last(I_j) \; \wedge \; i' = next(i) \tag{14}$$

Se proponen un par de expresiones auxiliares, (15) y (16), que permiten al modelo ordenar los sublotes de cada lote de acuerdo a su tamaño, de manera creciente o decreciente, respectivamente. Es decir, para cada operación de cada lote *j*, ésta se ejecuta de manera secuencial sobre los sublotes de *j*, de acuerdo al tamaño de los mismos. Ambas reglas de ordenamiento permiten modelar una política operativa, como la

producción por orden creciente/decreciente de tamaño de lotes, así como también buscar y evaluar soluciones alternativas.

$$U_{sij} \leq U_{s'ij} \;\; ; \;\; \forall i \in I_j, \forall j \in J, \forall s, s' \in S_j | s < last(S_j) \wedge s' = next(s) \tag{15}$$

$$U_{sij} \geq U_{s'ij} \;\; ; \;\; \forall i \in I_j, \forall j \in J, \forall s, s' \in S_j | s < last(S_j) \wedge s' = next(s) \tag{16}$$

Como función objetivo, la propuesta utiliza alternativamente tres medidas de desempeño. Por un lado, al emplear como función la minimización del valor de la variable *Cmax* (17), se busca optimizar el tiempo de finalización de todas las operaciones de todos los lotes. En este caso, el modelo debe considerar la restricción (18). Por otra parte, se plantea minimizar la función objetivo expresada en (19). La misma se define como la suma de los tiempos de finalización de la última operación sobre todos los sublotes, de todos los lotes ("Sublots Total Completion Time", STCT). Por último, si se considera la minimización de la tardanza de los lotes, la medida de performance a optimizar es la expresión (20), mientras que las desigualdades (21) y (22) deben incluirse en el modelo.

$$\min Cmax \;\; ; \tag{17}$$

$$Cmax \geq C_{sij} \;\; ; \;\; \forall s \in S_j, \forall i \in I_j, \forall j \in J \tag{18}$$

$$\min \sum_{\forall s \in S_j} \sum_{\forall i \in I_j | i = last(I_j)} \sum_{\forall j \in J} Cs_{sij} \;\; ; \tag{19}$$

$$\min \sum_{\forall j \in J} T_j \;\; ; \tag{20}$$

$$T_j \geq C_{ij} - dd_j \;\; ; \;\; , \forall j \in J, \forall i \in I_j \mid i = last(I_j) \tag{21}$$

$$T_j \geq 0 \;\; ; \;\; \forall j \in J \tag{22}$$

4 Casos de estudio

Con el fin de evaluar la formulación propuesta, se abordaron y resolvieron diversos casos de estudio. Para definir las instancias que corresponden a problemas de "scheduling" en entornos FJS considerando división de lotes, fue necesario adaptar los ejemplos de "scheduling" en FJS sin división de lotes, introducidos por Fattahi y colab. [10]. Esto se realizó debido a que no se hallaron en la literatura casos de benchmark específicos para el problema aquí tratado (en [8] el problema es diferente ya que se define de antemano la cantidad de sublotes que debe tener la solución). En la Tabla

1 se presentan algunos de los problemas resueltos, junto a su tamaño establecido por la cantidad de lotes *j*, operaciones *i* y máquinas *k* del entorno.

Tabla 1. Tamaño de los casos de estudio

Problema	En Fattahi y colab.[10]	# j	# i	# k
P1	SFJS1	2	2	2
P2	SFJS3	3	2	2
P3	SFJS6	3	3	3
P4	SFJS7	3	3	5
P5	SFJS8	3	3	4

Para cada problema P*x*, se definieron 3 conjuntos de datos variando los parámetros de demanda de cada lote y cantidad (máxima) de sublotes por lote (d_j y S_j, respectivamente), estableciendo entonces 15 instancias (P*x*-*i*). Los valores de demanda de cada lote *j* fueron definidos aleatoriamente entre 5 y 50, mientras que la cantidad de sublotes permitidos para cada *j* fue establecida entre 2 y 6 (ver Tabla 2).

Para los casos de minimización de la tardanza, se estableció la fecha de entrega de cada lote, dd_j, como la suma de los tiempos mínimos de procesamiento de cada operación, considerando la demanda del lote.

Tabla 2. Valores d_j, S_j para cada lote y cada instancia de problema.

		Lote		
Problema	**Instancia**	**1**	**2**	**3**
P1	1	7,2	11,3	-
	2	20,4	5,2	-
	3	33,6	18,4	-
P2	1	10,3	20,4	23,3
	2	21,3	32,4	5,3
	3	28,2	17,5	25,3
P3	1	19,3	43,5	7,2
	2	20,2	12,3	31,3
	3	43,4	11,3	32,3
P4	1	45,4	40,4	20,3
	2	10,2	15,2	16,3
	3	8,2	29,5	10,3
P5	1	8,3	18,3	42,4
	2	34,3	28,3	12,2
	3	25,3	42,4	10,1

Los resultados reportados en este trabajo fueron obtenidos al ejecutar el modelo en una computadora HP con 6 Gb de RAM, procesador AMD A10-4600M 2.3 GHz, empleando la configuración por defecto que provee el solver CPLEX en el entorno IBM-ILOG Studio [9].

En la Tabla 3 se reportan las soluciones halladas y el tiempo de cómputo empleado al minimizar makespan, STCT y tardanza, como medidas de performance. Estas funciones objetivo fueron empleadas de manera independiente, una por vez. El empleo y evaluación de una función multiobjetivo quedó fuera del alcance del presente trabajo.

Como se observa en la Tabla 3, se obtuvo la solución óptima para las 15 instancias de problema y todas las funciones objetivo empleadas. Al optimizar makespan, en 14 casos las soluciones óptimas fueron encontradas en menos de 3 segundos de CPU, existiendo una única instancia resuelta en un tiempo de 11.9 segundos. Cuando se empleó STCT como medida de desempeño, los tiempos de resolución se incrementaron algunos segundos, aunque siguieron siendo muy bajos, con 13 casos donde la agenda se obtuvo en menos de 20 segundos. Por otra parte, las soluciones óptimas obtenidas mediante la minimización de la tardanza se instanciaron de manera inmediata, en menos de 5 segundos.

Tabla 3. Resultados al minimizar makespan, STCT o tardanza

Problema	MILP					
	Makespan	Tiempo CPU	STCT	Tiempo CPU	Tardanza	Tiempo CPU
P1-1	726	< 1	4354	< 1	66	< 1
P1-2	805	< 1	5070	< 1	0	< 1
P1-3	1962	< 1	16692	< 1	360	1,4
P2-1	4175	< 1	36956	1,1	546	1,2
P2-2	4032	< 1	36532	< 1	840	1,1
P2-3	5404	< 1	48924	1,1	1403	< 1,0
P3-1	7440	1,9	80646	13,9	0	< 1
P3-2	6670	2,9	75720	19,8	140	1,0
P3-3	6950	1,4	99735	8,3	0	< 1,0
P4-1	9448	1,9	140133	13,7	0	1,7
P4-2	3777	1,8	48333	5,9	0	< 1
P4-3	4612	1,7	50229	2,0	0	< 1,0
P5-1	4966	1,7	58350	82,4	0	1,1
P5-2	5194	1,9	65145	217,5	0	4,4
P5-3	4744	11,9	61860	8,8	60	3,4

Estos resultados se alcanzaron sin utilizar ningún tipo de cota superior sobre las variables que representan los tiempos de inicio y fin. Pruebas preliminares usando límites superiores, redujeron en un orden de magnitud los tiempos de resolución para los problemas de mayor tamaño P4 y P5. La definición y análisis de estas cotas serán tratadas en trabajo futuro.

En la Tabla 4 se presentan los resultados cuando el modelo considera el uso de las reglas de ordenamiento de sublotes de acuerdo a su tamaño. Al ordenar de manera creciente, se debe considerar la restricción (15), mientras que si se pretende un orden decreciente, se utiliza la restricción (16). Se muestran en Tabla 4 los resultados correspondientes a las 9 instancias de mayor tamaño.

Cuando el modelo emplea las reglas para ordenar los sublotes por tamaño, los tiempos de resolución siguen siendo cortos. Para todos los casos en los que se minimiza STCT, el uso de la regla de orden creciente de tamaño de sublotes permite llegar a un mejor resultado que si se utiliza orden decreciente. Por el contrario, este tipo de patrón no es posible de identificar al minimizar makespan.

Tabla 4. Resultados empleando makespan, STCT o tardanza, considerando las reglas para el ordenamiento de sublotes

Problema	Orden de sublotes	MILP			
		makespan	Tiempo CPU	STCT	Tiempo CPU
P3-1	I.O. [a]	7670	1,1	80646	6,2
	D.O. [b]	7440	1,9	119262	1,7
P3-2	I.O.	7000	1,7	75720	3,3
	D.O.	6950	1,8	94320	1,5
P3-3	I.O.	7290	3,2	99735	13,2
	D.O.	7250	4,4	127059	1,1
P4-1	I.O.	9448	1,2	140133	11,6
	D.O.	9714	1,8	178224	1,6
P4-2	I.O.	3777	2,3	48333	2,6
	D.O.	3795	1,6	54825	1,9
P4-3	I.O.	5136	1,3	50229	2,8
	D.O.	4612	1,1	77427	1,4
P5-1	I.O.	4966	6,8	58350	33,3
	D.O.	5166	1,2	78009	2,8
P5-2	I.O.	5194	1,5	65145	32,5
	D.O.	5664	1,6	87795	5,1
P5-3	I.O.	4744	6,1	61860	5,9
	D.O.	5022	3,5	83094	1,1

[a] Sublotes ordenados por tamaño creciente, para cada i, j
[b] Sublotes ordenados por tamaño decreciente, para cada i, j

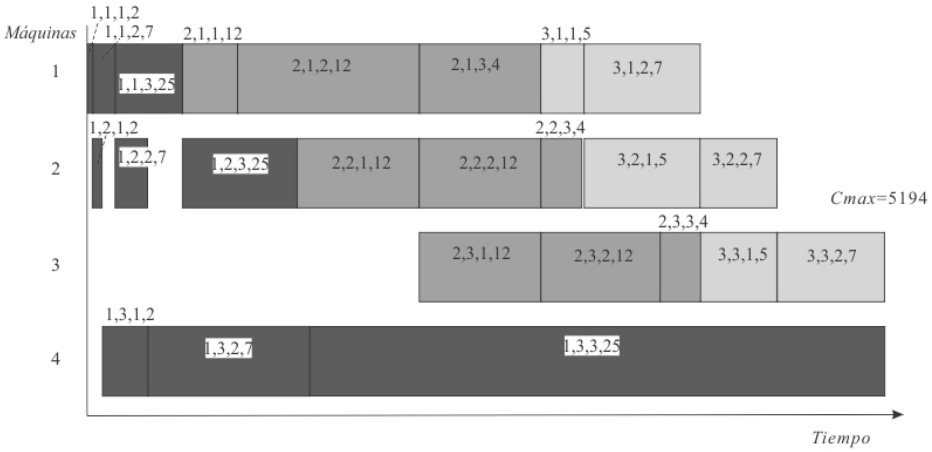
A modo de ilustración, en la Figura 1 se muestra el diagrama de Gantt que representa la solución óptima al minimizar makespan, para el problema P5-2. Las etiquetas de cada tarea representan lote, operación, sublote, y tamaño de sublote, respectivamente. El efecto práctico de la división en sublotes puede verse, por ejemplo, en el procesamiento de *j*2, donde al mismo tiempo hay piezas de dicho lote recibiendo las operaciones *i*1, *i*2 y *i*3, ya que se encuentran en diferentes sublotes asignados a distintas máquinas. La posibilidad de contar con piezas terminadas más temprano es también otra de las importantes ventajas de esta política de operación. Esto se puede ilustrar mediante la finalización del primer sublote de *j*1, muy pronto de haber iniciado el programa, en contraposición a la finalización de todas las partes de *j*1, que coincide con el makespan.

Se hace notar que al emplear la variable X_{rij} en lugar de W_{rsij} en las restricciones que modelan la secuencia de las operaciones, la reducción de la dimensión del pro-

blema es de un orden de magnitud. Por ejemplo, la cantidad de binarias en el problema P2-2 es de 317, mientras que si hubiéramos utilizado la variable W_{rsij} en lugar de X_{rij} , dicha cantidad sería de 3176.

El empleo de ambas variables y su sincronización, permite en esta propuesta abordar con eficiencia el problema definido. Sin embargo, no es factible el empleo de esta aproximación si el problema considerara posible intercalar sublotes de distintos lotes en una misma secuencia. Este tipo de problema está fuera de la definición del que aquí se ha abordado (ver Sección 2), y es tema de futuras investigaciones.

Fig. 1. Diagrama de Gantt. Solución al problema P5-2, minimizando makespan.



Ya ha sido planteado que el problema de “scheduling” en FSJ considerando división de lotes es más complejo que cuando no se consideran sublotes, como es el caso en [1] y [10]. Como se definió anteriormente, los problemas aquí resueltos toman como base los datos reportados en dichas contribuciones y, si bien no es posible comparar los resultados respecto del makespan reportado, ya que en [1] y [10] no tratan la división de lotes, el tiempo de cómputo sí es empleado para analizar cuánto más costoso es resolver el problema cuando existe división de lotes.

Al comparar los resultados alcanzados por la presente propuesta con los mejores resultados aportados por Demir y Kürşat Işleyen [1] al minimizar makespan, se observa que el tiempo de CPU para alcanzar la solución óptima es sólo un par de segundos mayor, para la mayor parte de los casos. Dichos autores lograron soluciones óptimas para los problemas P1 a P5 (FSJ sin división de lotes) en menos de 1 segundo de CPU (con procesador similar al utilizado aquí); mientras que en este trabajo, la mayor parte de las instancias encuentra su óptimo en un tiempo menor a 3 segundos, a pesar de la mayor complejidad del problema.

5 Conclusiones y trabajo futuro

El presente trabajo aborda el problema de la programación de operaciones en ambientes industriales de tipo “job-shop” flexible, donde se trabaja bajo política de división

de lotes en sublotes. El problema es de gran interés para el ambiente industrial, ya que la división de lotes permite disponer de productos terminados de manera más temprana, y por lo tanto, lograr una atención más rápida a las necesidades del mercado. También lo es desde la perspectiva académica, debido a la complejidad matemática del problema.

La contribución presenta una formulación matemática capaz de agendar operaciones y encontrar el número y tamaño adecuado de los sublotes de cada trabajo o lote. El modelo fue testeado en problemas pequeños respecto a la dimensión de los problemas reales, con el fin de verificar y validar su correcto funcionamiento. Se emplearon distintas medidas de desempeño, como makespan, tardanza y STCT ("Sublots Total Completion Time"), encontrándose soluciones óptimas en pocos segundos para todas las instancias de problema resueltas.

Este trabajo permite avanzar hacia el estudio de problemas de mayor tamaño y diversas características. A futuro, también se propondrán otras funciones objetivo, como las vinculadas al costo de producción, y se evaluará una función multiobjetivo. Al abordar casos de mayor dimensión, se considerará la posibilidad de aplicar otros métodos de resolución, como heurísticas o programación con restricciones.

Referencias

1. Y. DEMIR, AND S. KÜRŞAT IŞLEYEN, *Evaluation of mathematical models for flexible job-shop scheduling problems,* Applied Mathematical Modelling, 37 (2013), pp. 977-988.
2. W. XIA, AND Z. WU, *An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems,* Computers & Industrial Engineering, 48 (2005), pp. 409-425.
3. M. FELDMANN, AND D. BISKUP, *Lot streaming in a multiple product permutation flow shop with intermingling,* International Journal of Production Research, 46 (2008), pp. 197-216.
4. S.C. SARIN, AND P. JAIPRAKASH, *Flow Shop Lot Streaming*, Springer, 2007.
5. N. MORTEZAEI, AND N. ZULKIFLI, *Integration of Lot Sizing and Flow Shop Scheduling with Lot Streaming*, Journal of Applied Mathematics, Volume 2013, Article ID 216595, 9 pages.
6. T.C. WONG, FELIX T.S. CHAN, AND L.Y. CHAN, *A resource-constrained assembly job shop scheduling problem with Lot Streaming technique*, Computers & Industrial Engineering, 57 (2009), pp. 983-995.
7. C. LOW, C. HSU, AND K. HUANG, *Benefits of lot splitting in job-shop scheduling,* International Journal of Advanced Manufacturing Technology, 24 (2004), pp. 773–780.
8. F. M. DEFERSHA, AND M. CHEN, *A Coarse-Grain Parallel Genetic Algorithm for Flexible Job-Shop Scheduling with Lot Streaming* in Proceedings of 2009 International Conference on Computational Science and Engineering (2009), pp. 201-208.
9. IBM ILOG CPLEX Optimization Studio. http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/. Accedido: 30/12/2014.
10. P. FATTAHI, M. SAIDI MEHRABAD, AND F. JOLAI, *Mathematical modeling and heuristic approaches to flexible job shop scheduling problems,* Journal of Intelligent Manufacturing, 18 (2007), pp. 331-342.