Un problema de programación de la producción en células de fabricación que incluye almacenes

Adrián A. Toncovich[a], Daniel A. Rossit[a,b] y Mariano Frutos[a,c]
[a]Departamento de Ingeniería, Universidad Nacional del Sur, Bahía Blanca, CP 8000, Argentina
[b]INMABB-CONICET-UNS, Bahía Blanca, CP 8000, Argentina
[c]IIESS-CONICET-UNS, Bahía Blanca, CP 8000, Argentina
{atoncovi, daniel.rossit, mfrutos}@uns.edu.ar

***Abstract*—En este trabajo se presenta un problema específico de programación de la producción flow-shop de interés práctico. El sistema de fabricación está configurado como una célula de fabricación y en el planteamiento del problema se consideran los almacenes de materias primas y de productos terminados. El desempeño de la programación se evalúa de una manera multi-objetivo, considerando el tiempo total de producción (makespan) y la tardanza total (tardiness). Se propone una formulación matemática para el problema. Además, se presenta una estrategia meta-heurística para resolver eficientemente dicho problema y obtener soluciones de buena calidad en un tiempo computacional razonable. El procedimiento aplicado se basa en una adaptación de la meta-heurística de recocido simulado. Se generaron conjuntos de problemas para evaluar el método propuesto, obteniendo soluciones óptimas o casi óptimas en tiempos significativamente menores que los requeridos por el enfoque de optimización resuelto mediante CPLEX. Además, el algoritmo fue probado con problemas de mayor tamaño, para evaluar su comportamiento en espacios de búsqueda más extensos.**

***Index terms*—Programación de la producción, flow-shop permutativo, recocido simulado, optimización multi-objetivo, almacenes.**

I. Introducción

La competencia que enfrentan las empresas manufactureras impulsa el desarrollo de métodos innovadores para una mejor toma de decisiones. Estos métodos están igualmente dedicados a ser lo suficientemente rápidos y lograr resultados altamente competitivos. Uno de los problemas que frecuentemente es abordados por estos métodos es la programación de las operaciones de fabricación. La complejidad asociada a estos problemas restringe considerablemente las posibilidades de obtener resultados óptimos, para los casos del mundo real [1], [2]. Dado que la programación de las operaciones tiene efectos económicos significativos en el rendimiento de las empresas manufactureras, se requiere, ante la dificultad de obtener soluciones óptimas, contar al menos con enfoques adecuados para resolver los problemas con la suficiente rapidez, para obtener soluciones competitivas [1], [2], [3]. Los procesos de fabricación comprenden las diferentes operaciones que transforman las materias primas y componentes en productos finales. La formulación general de un problema de programación de la fabricación dado se puede obtener recurriendo a un enfoque de programación matemática [4], [5], [6]. En este trabajo se aborda específicamente el problema de programación en un entorno flow-shop permutativo (PFSSP, Permutation Flow-shop Scheduling Problem). El PFSSP es un problema de programación de optimización combinatoria que pertenece a la clase NP-Completo [7]. El PFSSP se define como el problema de encontrar la mejor programación de trabajos que se deben procesar en un conjunto de máquinas, dado que todos los trabajos tienen la misma secuencia de procesamiento a través de las máquinas y todas las máquinas procesan los trabajos siguiendo la misma secuencia. Este tipo de configuración suele ser la conveniente para generar mejoras en la productividad industrial y, por ello, se encuentra implementada en más de un cuarto de las instalaciones productivas del mundo real [8]. Incluso, se intenta extender sus beneficios a otros procesos productivos que, por su naturaleza, no podrían considerarse configuraciones de tipo flow-shop. Tal es el caso de las celdas de manufactura, que consisten en grupos de máquinas (celdas) que procesan un conjunto de todos los trabajos que guardan características similares entre sí. Esto permite que dentro de cada celda la configuración sea semejante a un sistema flow-shop. Este tipo de estrategias de producción es ampliamente utilizada por la industria [9], [10]. Sin embargo, las mejoras en la productividad que pueden alcanzarse por este tipo de estrategias suelen verse comprometidas cuando el entorno industrial en el que operan no está correctamente integrado a la celda. Tal como puede ser el caso del abastecimiento de los materiales para producir cada trabajo, o como la coordinación de su paradero una vez finalizada su producción. Esta problemática no ha sido abordada en la literatura sobre la temática. Es por este motivo que, en este trabajo, se concibe a la celda de manufactura inmersa en un sistema productivo, que debe proveer los materiales necesarios para el proceso de manufactura, así como también, recibir y almacenar los productos terminados.

La medida de rendimiento más frecuentemente utilizada para evaluar la calidad de las soluciones en el PFSSP, ha sido el makespan, que se define como el tiempo total de producción requerido para procesar todos los trabajos del sistema. A lo largo del tiempo se han considerado otras medidas de

Este trabajo fue financiado por el Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) y por la Secretaría de Ciencia y Tecnología de la Universidad Nacional del Sur (UNS) (PGI 24/ZJ35 y PGI 24/ZJ34).

rendimiento y se han añadido diferentes condiciones operativas a la configuración básica. En nuestro caso, el sistema de fabricación está dispuesto como una instalación de célula de fabricación y comprende, además, el suministro de materias primas y la entrega del producto final. El proceso de optimización considera como objetivos el makespan y la tardanza total en la búsqueda de soluciones para el problema.
En este trabajo se introduce un problema realista, nuevo para la literatura, para el cual se propone una formulación de programación matemática y se desarrolla una metodología de solución adecuada que permite abordar su naturaleza combinatoria. La metodología utilizada para resolver el problema se basa en un procedimiento meta-heurístico que puede considerarse como una adaptación multi-objetivo del algoritmo de recocido simulado. Este enfoque meta-heurístico genera un conjunto de soluciones que proporciona una aproximación al frente óptimo de Pareto. Se realizan una serie de experimentos para evaluar el rendimiento de la técnica propuesta.
El trabajo está organizado de la siguiente forma. Primero se presenta el PFSSP y se establece en detalle su configuración. A continuación, se formula un modelo de programación matemática para el PFSSP. Luego, se presenta el procedimiento de recocido simulado. A continuación, se resuelve un conjunto de problemas utilizando ambos enfoques para comparar los resultados. Posteriormente, el algoritmo de recocido simulado se aplica a instancias más grandes para evaluar su rendimiento. Finalmente, se establecen las conclusiones y se proponen futuras líneas de investigación.

II. Un Problema Realista

Con la intención de ejemplificar el alcance y la complejidad de la investigación en esta área, se introduce un problema realista y se desarrolla una metodología de solución conveniente. El problema de programación de la producción introduce una serie de elementos de interés en entornos reales de producción conocidos como células flexibles de fabricación [9], [10], [11], [12]. El problema considerado se limita al análisis de las actividades de programación en un entorno de producción constituido esencialmente por una serie de máquinas dispuestas para configurar una celda, un almacén de materias primas y un almacén de productos terminados. Entre los almacenes y la celda de manufactura se deben realizar ciertas operaciones de traslado, ya sea de materias primas o de productos terminados, que impactarán en las medidas de desempeño consideradas para resolver el problema de secuenciación. Estas operaciones, naturalmente, dependerán del tipo de producto que se procese (cantidad de piezas, componentes, pesos, volúmenes, etc.).

En primer lugar, se propone una formulación de programación matemática para resolver el problema caracterizado como un sistema flow-shop permutativo con la posibilidad de que existan operaciones que omitan etapas (Permutation Skip Flow-Shop), es decir, donde algunos trabajos no necesariamente tienen que ser procesados en todas las máquinas de la celda. Estas características adicionales se encuentran en varios entornos de producción reales. A continuación, se propone el procedimiento meta-heurístico multi-objetivo para resolver el problema de programación de la producción [13], [14].

Las medidas de desempeño utilizadas para evaluar la calidad de las secuencias tienen en cuenta la optimización de: la utilización de los recursos de producción, los indicadores relacionados con las fechas de entrega y los tiempos de transporte de las materias primas y de los productos terminados. En particular, se tienen en cuenta dos objetivos al analizar la calidad de las soluciones candidatas: la minimización del tiempo para la realización de todos los trabajos, o makepan, y la minimización de la tardanza total, o tardiness. La figura 1 muestra un ejemplo de configuración de una celda de fabricación flow-shop constituida por tres máquinas (M_1, M_2 y M_3), un almacén de materias primas (A_{MP}) y un almacén de productos terminados (A_{PF}).

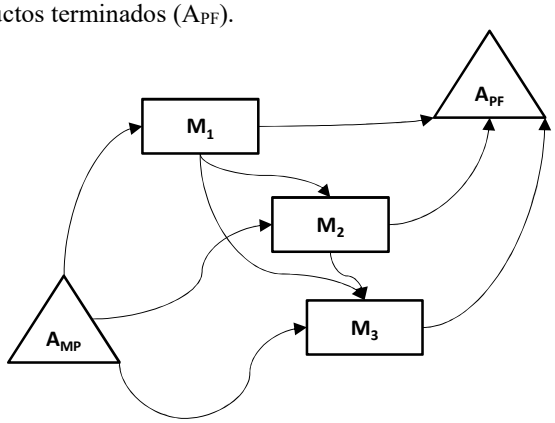

Fig 1. Ejemplo de configuración de una celda de fabricación con tres máquinas.

III. Formulación de Programación Matemática

El problema abordado en este trabajo es un problema de secuenciación en una celda de fabricación que se encuentra alejada de los almacenes de materias primas y de productos terminados. Los trabajos tienen una fecha de lanzamiento, y una vez producida la liberación, las materias primas requeridas para producir el trabajo pueden ser transportadas desde el almacén de materias primas hasta la celda de fabricación. Cuando las materias primas necesarias para realizar el trabajo llegan a la célula de fabricación, puede comenzar su procesamiento. La configuración productiva dentro de la celda de fabricación es del tipo flow-shop, por lo tanto, todos los trabajos tienen la misma secuencia de máquinas. Sin embargo, algunos de los trabajos no requieren ser procesados en todas las máquinas, es decir, pueden omitir algunas operaciones. Después de que cada trabajo ha terminado su procesamiento, debe ser transportado desde la célula de fabricación hasta el almacén de productos terminados. Este último tiempo de transporte repercute en el tiempo de entrega del trabajo, ya que debe añadirse al tiempo de finalización del trabajo en la celda de fabricación.

Otros supuestos que se hacen para esta formulación del problema son:

- Cada máquina puede procesar solo un trabajo a la vez.
- Cada trabajo puede ser procesado por una sola máquina a la vez.
- Una vez que se inicia el procesamiento de un trabajo, no puede interrumpirse.

- Las máquinas de la celda pueden estar ociosas durante el horizonte de planificación.

CONJUNTOS

j: índice para trabajos, j = 1, 2, …, n.

m: índice para máquinas, m = 1, 2, …, m.

PARÁMETROS

$p_{j,m}$: tiempo de procesamiento del trabajo j en la máquina m.

t_{MP_j}: tiempo de transporte de materias primas a la célula de fabricación.

r_j: liberación de la orden del trabajo j.

d_j: fecha de entrega del trabajo j.

t_{PF_j}: tiempo de transporte de productos finalizados al almacén.

M: un número positivo muy grande.

VARIABLES

$C_{j,m}$: tiempo de finalización del trabajo j en la máquina m.

T_j: tardanza del trabajo j.

$C_{máx}$: tiempo total de producción (makespan).

$x_{j',j}$: variable binaria, es 1 si el trabajo j' es procesado antes que el trabajo j, y 0 en otro caso.

Función objetivo (1), minimizar el tiempo total de producción (makespan) y la tardanza total (tardiness).

$$Min\ z = C_{máx} + \sum_j T_j \tag{1}$$

Restricción de precedencia (2), asegura que el trabajo j no comience su procesamiento en la máquina m antes de que haya terminado su procesamiento en la máquina m-1.

$$C_{j,m} \geq C_{j,m-1} + p_{j,m}\ ;\ \forall j, m > 1 \tag{2}$$

Restricción de ordenamiento (3), establece que el trabajo j no comience su procesamiento en la máquina m si antes no terminaron su procesamiento todos los trabajos j' precedentes.

$$C_{j,m} \geq C_{j',m} + p_{j,m} - \left(1 - x_{j',j}\right) \cdot M;\ \forall m, j' \neq j \tag{3}$$

Restricción lógica (4), si el trabajo j' está secuenciado antes que el trabajo j, entonces la inversa no es válida.

$$x_{j',j} + x_{j,j'} = 1\ ; j \neq j' \tag{4}$$

Abastecimiento de materias primas (5), cada trabajo no puede comenzar su procesamiento, si antes no fue liberada la orden correspondiente y las materias primas no fueron suministradas.

$$C_{j,m} \geq r_j + t_{MP_j} + p_{j,m}\ ; m = 1, \forall j \tag{5}$$

Makespan (6), para calcular la finalización del trabajo j, es necesario considerar el tiempo de traslado hacia el almacén de productos finalizados.

$$C_{máx} \geq C_{j,m} + t_{PF_j}\ ;\ \forall j, \forall m \tag{6}$$

Tardanza (7), para calcular la tardanza del trabajo j, es necesario considerar el tiempo de traslado hacia el almacén de productos finalizados.

$$T_j = \text{máx}\{0, \left(C_{j,m} + t_{PF_j}\right) - d_j\}\ ; \forall j, m = M \tag{7}$$

Restricciones (8) relacionadas con los valores no negativos y binarios de las variables.

$$T_j, C_{j,m} \geq 0;\ x_{j,j'} \in \{0,1\}; \forall j, \forall m \tag{8}$$

IV. SOLUCIÓN BASADA EN LA META-HEURÍSTICA DE RECOCIDO SIMULADO

En esta sección se describe el algoritmo basado en el recocido simulado utilizado para resolver el problema presentado anteriormente. El algoritmo tiene la estructura de un procedimiento de recocido simulado con archivo de soluciones de Pareto (PASA, Pareto Archived Simulated Annealing). El recocido simulado es un método de búsqueda local que se desarrolló a partir de una analogía con el fenómeno de recocido [15] para resolver problemas complejos de optimización. Los métodos de búsqueda local buscan la solución con un mejor valor del criterio elegido en el entorno de la solución actual, la aceptan como la solución actual y repiten este procedimiento hasta que no resulte posible mejorar la solución en el entorno explorado. Mediante la aplicación sistemática de este procedimiento se obtiene, en general, un óptimo local para el problema. Para evitar quedar atrapado en un óptimo local, se debe incorporar un mecanismo de diversificación con el fin de explorar adecuadamente el espacio de soluciones. En la meta-heurística de recocido simulado, la estrategia de diversificación permite movimientos, con cierta probabilidad, hacia soluciones que empeoran el valor actual de la función objetivo. El recocido simulado ya ha demostrado su capacidad de resolución en entornos productivos flow-shop regulares [16], [17], [18].

A. Algoritmo PASA

En [19] y [20] se propusieron dos procedimientos similares basados en el recocido simulado para resolver problemas de optimización combinatoria multi-objetivo. Posteriormente, en [21] y [22] se desarrollaron otras variantes de la metodología que consideran de forma más explícita las características multi-objetivo del problema [23]. En este trabajo, siguiendo lo propuesto en [22], se utiliza una función de agregación de las funciones objetivo, junto con un sistema de archivo de soluciones no dominadas encontradas durante el proceso de búsqueda. Se supone que las funciones objetivo a minimizar, f_p, p = 1,2,...,P, son positivas. En tal caso, el problema se puede

transformar en un problema de minimización mono-objetivo utilizando la siguiente función de agregación (9).

$$z(S) = \sum_{p=1}^{P} \ln\left(f_p(S)\right) \quad (9)$$

Donde S es una secuencia dada. Entonces, la expresión (10), representa la variación relativa media de las funciones objetivo entre la solución actual (S_A) y la solución candidata (S_C).

$$\Delta z = z(S_C) - z(S_A) = \sum_{p=1}^{P} \ln\left(\frac{f_p(S_C)}{f_p(S_A)}\right) \quad (10)$$

Si $\Delta z > 0$, S_C deteriora la media relativa del conjunto de funciones objetivo. Si $\Delta z < 0$, S_C mejora la media relativa del conjunto de funciones objetivo. En el primer caso se acepta la solución S_C con una probabilidad dada por $PA = e^{-\Delta z/T}$, donde T es el parámetro de control que simula la temperatura en el proceso metalúrgico del recocido. El método tiene en cuenta un archivo de soluciones no dominadas que se gestiona de la siguiente manera:

- Si S_C está dominada por al menos alguna de las soluciones del archivo, no se incorpora al conjunto.
- Si S_C domina a una o más soluciones en el archivo, es incorporada eliminando las soluciones que domina.
- Si S_C no domina, ni es dominada por las soluciones del archivo, es incorporada sin eliminar ninguna solución.

Para obtener una aproximación a toda la frontera eficiente durante el proceso de búsqueda, es necesario reiniciar la búsqueda regularmente desde una de las soluciones archivadas seleccionada al azar. El algoritmo PASA incorpora los parámetros clásicos del recocido simulado:

T: parámetro de control (temperatura), valor positivo que varía desde un valor inicial mayor, T_0, a otro que es menor, T_f, durante la ejecución del algoritmo.

N_T: número de iteraciones realizadas por el algoritmo para cierto valor de T.

α: función en T, α = α (T), que determina la variación de T. En general α (T) = α T, en la práctica α ∈ [0.8,0.9999].

N_{stop}: número máximo de iteraciones permitidas sin mejora.

Se aplica el siguiente pseudo-código para determinar un conjunto de soluciones potencialmente eficientes (óptimas de Pareto).

i. Inicio

Se utiliza un procedimiento constructivo aleatorizado para generar una solución inicial, S_0.

Se evalúa $f_p(S_0)$, ∀ p.

S_0 se incorpora al conjunto de soluciones eficientes: CE = $\{S_0\}$, $N_{cont} = t = 0$, $T = T_0$.

ii. Iteración t

Se genera aleatoriamente una solución en el entorno de S_A, $S_C \in V(S_A)$.

Se evalúa $f_p(S_C)$, ∀ p.

Se calcula $\Delta z = z(S_C) - z(S_A) = \sum_p \ln(f_p(S_C)/f_p(S_A))$.

Si $\Delta z \leq 0$, la nueva solución es aceptada: $S_A \leftarrow S_C$, $N_{cont} = 0$.

En otro caso, S_C es aceptada con la siguiente probabilidad $PA = e^{-\Delta z/T}$.

Se genera un número aleatorio ξ uniformemente distribuido en el intervalo [0, 1]:

$$\begin{cases} \text{si } \xi \leq PA, S_A \leftarrow S_C, N_{cont} = 0, \\ \text{si } \xi > PA, S_A \leftarrow S_A, N_{cont} = N_{cont} + 1. \end{cases}$$

Si corresponde, se actualiza CE teniendo en cuenta S_c.

$t \leftarrow t + 1$: Si t es un múltiplo de N_T, entonces $T = \alpha T$, en otro caso el valor de T se mantiene. Si $N_{cont} = N_{stop}$ o $T < T_f$, se detiene la ejecución, en otro caso se continúa la ejecución.

Es necesario señalar que la naturaleza multi-objetivo del problema no fue probada en este trabajo. Es decir, para comparar los resultados de la meta-heurística con los de la programación matemática solo se consideraron las mejores soluciones en términos de la función objetivo dada por la ecuación (1).

V. Experiencia computacional

A. Diseño experimental

Para evaluar los resultados de la formulación de programación matemática y del algoritmo propuesto en este trabajo, se consideraron instancias de diferentes tamaños. El número de trabajos se definió a partir de los valores 5, 10, 15 y 20, mientras que para el número de máquinas se tuvieron en cuenta las mismas posibilidades (5, 10, 15 y 20). No todas las combinaciones posibles fueron consideradas. La Tabla 1 presenta las instancias evaluadas.

TABLA I
DETALLE DE LAS INSTANCIAS EVALUADAS

Instancias	No. de trabajos	No. de máquinas
1	5	5
2	5	10
3	10	5
4	10	10
5	15	10
6	15	15
7	20	10

De acuerdo con lo indicado al introducir la formulación de programación matemática, resulta necesario definir los valores de varios parámetros para especificar los cinco escenarios considerados dentro de cada instancia. Para el caso de los tiempos de procesamiento, $p_{j,m}$, los datos se obtuvieron a partir de una distribución cuasi-uniforme [0;100]. Dado que el problema abordado contempla la posibilidad de que existan operaciones omitidas, se incluye el valor 0 en la distribución del tiempo de procesamiento, y como característica especial, la probabilidad de omitir operaciones se considera del 3%, es decir, no todos los valores posibles tienen la misma probabilidad, por ello se habla de una distribución cuasi-uniforme. Esto amplifica el impacto de saltar operaciones en el problema de programación. Para los tiempos de transporte

desde el almacén de materias primas hasta la celda de fabricación y desde la celda de fabricación hasta el almacén de productos finalizados, los datos se tomaron de una distribución uniforme [10;20]. Las fechas de lanzamiento de los trabajos se obtuvieron a partir de una distribución uniforme [1;100]. Respecto a la fecha de vencimiento de los trabajos, se siguieron como guía los lineamientos dados en [24], sin embargo en ese artículo la fecha de vencimiento se calcula de la siguiente manera: $d_j = r_j + \sum_m p_{j,m} \cdot (1 + aleatorio \cdot 3)$. Esta representación incluye todos los tiempos de procesamiento que el trabajo *j* debe recorrer para su finalización, una vez producido su lanzamiento, más un tiempo adicional dado por el segundo término de la expresión encerrada entre paréntesis, que considera un número al azar (*aleatorio*) aleatorio entre [0;1]. En este caso, deben tenerse en cuenta para cada trabajo otros tiempos adicionales a los ya mencionados, es decir, los tiempos de transporte desde el almacén de materias primas hasta la célula de fabricación y desde la célula hasta el almacén de producto terminado. Por lo tanto, en este caso la fecha de entrega del trabajo j, se calculará como: $d_j = r_j + t_{MP_j} + t_{PF_j} + \sum_m p_{j,m} \cdot (1 + aleatorio \cdot 0.3)$. Además, en la presente formulación, en el cálculo de la fecha de entrega la constante que multiplica el número aleatorio, se ha reducido de 3 a 0.3. Esto genera las fechas de entrega más ajustadas, dando una mayor relevancia en el enfoque multi-objetivo a la tardanza total con respecto al makespan. Para cada instancia (combinación de número de máquinas y número de trabajos) se generaron cinco conjuntos de parámetros diferentes.

Para los casos resueltos mediante el enfoque de programación matemática fue utilizado Pyomo [25], [26], como lenguaje de modelado, y CPLEX 12.5.0 como solver. Las pruebas se ejecutaron en una CPU con 8 GB de RAM e i-core 5, el sistema operativo es de 64 bits y el tiempo de ejecución límite fue establecido en una hora (3600 segundos). CPLEX pudo resolver instancias de hasta 10 trabajos y 10 máquinas. Para la siguiente instancia, 15 trabajos y 10 máquinas, no resultó posible encontrar una solución con una brecha (gap) inferior al 3%. En promedio la brecha reportada por CPLEX después de una hora de corrida fue mayor al 30%. Estas soluciones fueron descartadas. Los casos resueltos utilizando el algoritmo PASA fueron programados en VBA (Visual Basic for Applications) como lenguaje de modelado sobre una configuración de MS-Excel. Las pruebas se ejecutaron en una CPU con 8 GB de RAM e i-core 7 y un sistema operativo de 64 bits.

B. Resultados

La Tabla II presenta los resultados de CPLEX y del algoritmo PASA. Los valores reportados para el algoritmo PASA corresponden al promedio de 30 corridas y el valor de la mejor corrida para cada escenario. La tabla muestra el valor de la función objetivo global (FO), es decir, la suma de ambos objetivos y, a continuación, el valor de makespan ($C_{máx}$), la tardanza total (Tardiness), y el tiempo requerido para obtener la mejor solución, respectivamente. Para el primer grupo de escenarios (5T_5M) se puede ver que el algoritmo PASA funciona bastante bien, obteniendo para todos los casos la solución óptima (el valor medio coincide con el mejor valor). Esta precisión del algoritmo PASA se repite para el segundo grupo de escenarios (5T_10M). Para los siguientes grupos de escenarios, tenemos ligeras diferencias entre el valor de las funciones objetivo. Para tener una mejor percepción de esto, se presenta la Tabla III.

TABLA II
RESULTADOS DE CPLEX Y DEL ALGORITMO PASA

		MIP-CPLEX				PASA							
						Promedio (30 corridas)				Mejor valor obtenido			
Instancia	Escenario	FO	Cmáx	Tardiness	Tiempo (s)	FO	Cmáx	Tardiness	Tiempo (s)	FO	Cmáx	Tardiness	Tiempo (s)
5T_5M	1	970	510	460	< 1	970	510	460	< 1	970	510	460	< 1
	2	834	553	281	< 1	834	553	281	1	834	553	281	< 1
	3	951	517	434	< 1	951	517	434	1	951	517	434	< 1
	4	1201	621	580	< 1	1201	621	580	1	1201	621	580	< 1
	5	1156	591	565	< 1	1156	591	565	< 1	1156	591	565	< 1
5T_10M	1	1394	929	465	1	1394	929	465	1	1394	929	465	< 1
	2	1431	835	596	1	1431	835	596	1	1431	835	596	< 1
	3	1082	847	235	1	1082	847	235	< 1	1082	847	235	< 1
	4	1210	746	464	1	1210	746	464	1	1210	746	464	< 1
	5	1228	895	333	1	1228	895	333	1	1228	895	333	< 1
10T_5M	1	2742	845	1897	10	2757	843	1914	23	2742	845	1897	8
	2	2579	791	1788	10	2586	792	1795	21	2579	791	1788	< 1
	3	2882	902	1980	9	2898	914	1983	18	2882	902	1980	< 1
	4	2361	775	1586	12	2375	777	1598	30	2361	775	1586	8
	5	2903	755	2148	10	2909	756	2153	29	2903	755	2148	5
10T_10M	1	2746	1095	1651	22	2773	1097	1676	27	2746	1095	1651	< 1
	2	3358	1158	2200	17	3365	1160	2205	25	3358	1158	2200	7
	3	2560	1058	1502	19	2566	1060	1506	21	2560	1058	1502	< 1
	4	2571	1040	1531	23	2611	1077	1534	34	2571	1040	1531	10
	5	3262	1060	2202	14	3265	1063	2202	18	3262	1060	2202	< 1

TABLA III
COMPARACIÓN ENTRE LAS SOLUCIONES PROMEDIO DE CPLEX Y DEL ALGORITMO PASA

Instancia	CPLEX			PASA					
	FO	Cmáx	Tardiness	FO		Cmáx		Tardiness	
				Promedio	Diferencia	Promedio	Diferencia	Promedio	Diferencia
5T_5M	1022	558	464	1022	0.00%	558	0.0%	464	0.0%
5T_10M	1269	850	418	1269	0.00%	850	0.0%	419	0.0%
10T_5M	2693	813	1879	2705	-0.45%	816	-0.6%	1889	-1.1%
10T_10M	2899	1082	1817	2916	-0.59%	1091	-0.8%	1825	-0.6%

En la Tabla III, la comparación entre la solución obtenida mediante CPLEX y la solución generada a través de PASA se expresa en términos de la diferencia porcentual entre los valores correspondientes, con respecto a la solución obtenida al resolver con CPLEX. A partir de la Tabla III, es posible ver que todos los porcentajes están por debajo del 1%, además, muchos de ellos son directamente 0%, lo cual quiere decir que la solución óptima y la solución media encontrada por el algoritmo PASA coinciden perfectamente. A pesar de que la magnitud del error es más bien menor, vale la pena mencionar que el algoritmo PASA está equilibrado en la búsqueda de ambas funciones objetivo. Concretamente, para los casos de 10 trabajos y 5 máquinas el error porcentual para el objetivo del tiempo total de fabricación es menor que el error para el objetivo de tardanza. Mientras tanto, para los casos de 10 trabajos y 10 máquinas, ocurre lo contrario. Por lo tanto, el algoritmo PASA no está sesgado en la búsqueda de uno de los objetivos con respecto al otro, tal como se pretende.

TABLA IV
INSTANCIAS MÁS GRANDES RESUELTAS POR EL ALGORITMO PASA

PASA									
Instancia	Escenario	Promedio (30 corridas)				Mejor valor obtenido			
		FO	Cmáx	Tardiness	Tiempo (s)	FO	Cmáx	Tardiness	Tiempo (s)
15T_10M	1	6633	1503	5130	67	6370	1441	4929	71
	2	6520	1504	5016	57	6306	1499	4807	71
	3	6375	1444	4931	69	6173	1423	4750	55
	4	6109	1470	4639	58	5923	1400	4523	113
	5	6044	1480	4564	52	5789	1437	4352	114
15T_15M	1	5794	1714	4080	84	5562	1658	3904	105
	2	7452	1839	5613	99	7347	1705	5642	275
	3	7230	1859	5371	133	6983	1866	5117	140
	4	7497	1805	5693	145	7152	1804	5348	180
	5	6622	1753	4869	166	6416	1774	4642	191
20T_10M	1	11349	1789	9560	202	10957	1779	9178	177
	2	11286	1799	9486	172	10798	1789	9009	107
	3	10463	1636	8827	191	9846	1602	8244	157
	4	10830	1751	9078	143	10412	1711	8701	156
	5	11248	1801	9448	106	10649	1696	8953	101

Los casos más grandes fueron resueltos solo por el algoritmo PASA, ya que CPLEX no pudo encontrar una solución reportable en un tiempo de cómputo de una hora. En la Tabla IV, se puede ver la precisión del algoritmo PASA, ya que la diferencia entre el mejor valor con respecto al valor promedio (considerando 30 corridas para cada escenario) es relativamente pequeña. Para un análisis directo de esta cuestión se presenta la Tabla V. En la Tabla V se muestra la diferencia porcentual entre el mejor valor obtenido por el algoritmo PASA y el promedio obtenido por PASA después de 30 corridas. Estos valores indican que el algoritmo se comporta de manera estable y replicable, capaz de producir resultados similares en condiciones similares. Este aspecto constituye una fortaleza del algoritmo. La diferencia media entre el mejor valor y el valor medio de la función objetivo global es inferior al 4%, apoyando la capacidad del algoritmo para producir soluciones de alta calidad empleando pocas corridas. Para el objetivo del makespan la diferencia entre el mejor valor y el valor medio es del 2.4% y para el objetivo de la tardanza es del 4.4%.

TABLA V
PRECISIÓN DEL ALGORITMO PASA, DIFERENCIA RELATIVA DEL VALOR PROMEDIO OBTENIDO POR PASA CON RESPECTO AL MEJOR VALOR OBTENIDO POR PASA

Precisión del algoritmo PASA				
Instancia	Escenario	FO	Cmáx	Tardiness
15T_10M	1	4.1%	4.3%	4.1%
	2	3.4%	0.3%	4.4%
	3	3.3%	1.5%	3.8%
	4	3.2%	5.0%	2.6%
	5	4.4%	3.0%	4.9%
15T_15M	1	4.2%	3.4%	4.5%
	2	1.4%	7.8%	-0.5%
	3	3.5%	-0.4%	5.0%
	4	4.8%	0.0%	6.4%
	5	3.2%	-1.2%	4.9%
20T_10M	1	3.6%	0.6%	4.2%
	2	4.5%	0.6%	5.3%
	3	6.3%	2.1%	7.0%
	4	4.0%	2.4%	4.3%
	5	5.6%	6.2%	5.5%
	Promedio	4.0%	2.4%	4.4%

VI. Conclusión

En este trabajo se estudió un caso particular del PFSSP (Permutation Flow-Shop Scheduling Problem), que incluye características tales como: dos objetivos de rendimiento, tiempos de transporte, omisión de operaciones, fechas de lanzamiento y fechas de entrega variables. Para este problema se presentaron una formulación de programación matemática y un algoritmo de recocido simulado multi-objetivo (Algoritmo PASA). Los resultados de ambos enfoques son similares en cuanto a su calidad. Sus tiempos de ejecución, probados en problemas de ejemplo, difieren significativamente tan pronto como el tamaño del problema aumenta, con un mejor desempeño del enfoque meta-heurístico. Problemas de mayor tamaño también fueron abordados mediante el algoritmo PASA. Se observó que los resultados obtenidos eran estables en términos de precisión y las soluciones se obtuvieron en un tiempo de cálculo razonable.

En futuras investigaciones se considerarán instancias de mayor tamaño y diferentes medidas de desempeño, realizando comparaciones de resultados con otras estrategias de solución, tratando de realizar contribuciones en el desarrollo de método precisos y eficientes para resolver este tipo de problemas de programación de la producción.

Referencias

[1] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Handbook on scheduling: from theory to applications.* Springer Science & Business Media, 2007.

[2] M. Pinedo, *Scheduling: theory, algorithms, and systems*. New Jersey: Prentice-Hall, 2012.

[3] J. M. Framiñán, R. Leisten, and R. R. García, *Manufacturing scheduling systems.* Berlin: Springer, 2014.

[4] R. Ruiz, F. S. Şerifoğlu, and T. Urlings, "Modeling realistic hybrid flexible flowshop scheduling problems," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1151–1175, 2008.

[5] M. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab, "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems". *Int. J. Ind. Eng. Comput.*, vol. 1, no. 1, pp. 1–10, 2010.

[6] F. M. Defersha and M. Chen, "Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem," *Int. J. Adv. Manuf. Technol.*, vol. 62, no. 1, pp.249–265, 2012.

[7] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1 no. 2, pp. 117–129, 1976.

[8] Q. K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information sciences*, vol. 181, no. 12, pp. 2455–2468, 2011.

[9] J. A. Joines, C. T. Culbreth, and R. E. King, "Manufacturing cell design: an integer programming model employing genetic algorithms," *IIE Trans.*, vol. 28, no. 1, pp. 69–85, 1996.

[10] P. M. França, J. N. Gupta, A. S. Mendes, P. Moscato, and K. J. Veltink, "Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups," *Comput. Ind. Eng.*, vol. 48, no. 3, pp. 491–506, 2005.

[11] S. H. Hendizadeh, H. Faramarzi, Mansouri, S. A., Gupta, J. N., and T. Y. ElMekkawy, "Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times," *Int. J. Prod. Econ.*, vol. 111, no. 2, pp. 593–605, 2008.

[12] Y. Li, X. Li and J. N. Gupta, "Solving the multi-objective flowline manufacturing cell scheduling problem by hybrid harmony search," *Expert Systems with Applications*, vol. 42, no.3, pp. 1409–1417, 2015.

[13] G. Minella, R. Ruiz, and M. Ciavotta, "Restarted Iterated Pareto Greedy algorithm for multi-objective flowshop scheduling problems," *Comput. Oper. Res.*, vol. 38, no. 11, pp. 1521–1533, 2011.

[14] N. Shoaardebili and P. Fattahi, "Multi-objective meta-heuristics to solve three-stage assembly flow shop scheduling problem with machine availability constraints," *Int. J. Prod. Res.*, 53(3), pp. 944–968, 2015.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[16] I. H. Osman and C. N. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989.

[17] C. Low, "Simulated annealing heuristic for flow-shop scheduling problems with unrelated parallel machines," *Comput. Oper. Res.*, vol. 32, no. 8, pp. 2013–2025, 2005.

[18] B. Vahedi Nouri, P. Fattahi, and R. Ramezanian, "Hybrid firefly-simulated annealing algorithm for the flow-shop problem with learning effects and flexible maintenance activities," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3501–3515, 2013.

[19] P. Serafini, "Simulated Annealing for Multi Objective Optimization Problems," in *Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application*, Tzeng G. H., Wang H. F., Wen U. P., Yu P. L. (eds.), New York, NY, USA: Springer, 1992, pp. 283–292.

[20] P. Fortemps, J. Teghem, and B. Ulungu, "Heuristics for multiobjective combinatorial optimization by simulated annealing," in *Proc. XIth Int. Conf. on Multiple Criteria Decision Making*, University of Coimbra, Portugal, Aug. 1994.

[21] P. Czyzak and A. Jaszkiewicz, "Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization," *J. Multi-Crit. Decis. Anal.*, vol. 7, no. 1, pp.34–47, 1998.

[22] P. Engrand and X. Mouney, "Une méthode originale d'optimisation multiobjectif," Tech. Rep. HT-14/97/035/A (98NJ00005), EDF-DER Electricité de France, Direction des Études et Recherches, Clamart, France, 1998.

[23] Y. Collette and P. Siarry, "Multiobjective methods using metaheuristics," in *Multiobjective Optimization: Principles and Case Studies*, Berlin Heidelberg, Germany: Springer, 2004, pp. 109–133.

[24] R. Ruiz and T. Stützle, T. "An Iterated Greedy heuristic for the sequence dependent setup times flow-shop problem with makespan and weighted tardiness objectives," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 1143–1159, 2008.

[25] W. E. Hart, J. P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in Python," *Math. Prog. Comput.*, vol. 3, no. 3, pp. 219–260, 2011.

[26] W. E. Hart, C. Laird, J. P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python* (Vol. 67). Springer Science & Business Media, 2012.