Implementación de un procedimiento basado en algoritmos evolutivos para programar la producción de marquillas estampadas por transferencia térmica*

Implementing a Scheduling Procedure based on Evolutionary Algorithms for the Production of Hot-Stamped Labels*

Implementação de um procedimento baseado em algoritmos evolutivos para programar a produção de selos estampados por transferência térmica*

*Fabián Vargas Nieto***
*Jairo Rafael Montoya Torres****

* Fecha de recepción: 9 de agosto de 2007. Fecha de aceptación para publicación: 9 de junio de 2008. Este artículo está basado en el trabajo de investigación de Maestría en Ingeniería Industrial en la Universidad del Norte, Barranquilla, Colombia, del primer autor.

* Date of submission: August 9, 2007. Date of acceptance for publication: June 9, 2008. This article is based on the research work carried out by the first author to receive his Master's degree in Industrial Engineering from Universidad del Norte.

* Data de recepção: agosto 9, 2007. Data de aceitação para publicação: junho 9, 2008. Este artigo esta baseado na pesquisa desenvolvida pelo primeiro autor a receber seu titulo de mestrado em Engenharia da Universidade do Norte.

** Ingeniero industrial. Magíster en Ingeniería Industrial, Universidad del Norte, Barranquilla, Colombia. Correo electrónico: fvargas@fnotex.com.

*** Ingeniero industrial, Universidad del Norte, Barranquilla, Colombia. Master of Science in Industrial Engineering and Management, Institut National Polytechnique de Grenoble, Francia. Doctor en Ingeniería Industrial, Ecole National Supérieure des Mines de Saint-Etienne y Université Jean Monnet, Francia. Profesor asociado, Universidad de La Sabana. Correo electrónico: jairo.montoya@unisabana.edu.co.

Resumen

En este artículo se estudia el problema de planificar la producción en una empresa manufacturera de marquillas para ropa, estampadas por transferencia térmica. El problema se modela utilizando la configuración de un *flowshop* flexible, con el objetivo doble de minimizar el lapso de fabricación (*makespan*) y el número de trabajos tardíos. Siendo este un problema novedoso en la literatura, se propone un enfoque de resolución biobjetivo, basado en algoritmos evolutivos. A partir de un conjunto de datos proveniente de órdenes reales de empresa, los experimentos muestran la pertinencia del procedimiento propuesto, tanto en términos comparativos, con respecto a la forma tradicional de programación manual de la producción, como con respecto a procedimientos conocidos en la literatura. Los resultados de la implementación en la empresa en los últimos meses muestran, igualmente, un mejoramiento considerable en los indicadores propios de desempeño de la línea de producción.

Palabras clave

Industrias manufactureras-Automatización, programación de la producción, algoritmo evolucionista.

Abstract

This paper considers the problem of production scheduling in a real-life manufacturing plant belonging to the apparel industry. The problem is modeled as a two-stage flexible flow shop, with minimization of both makespan and number of tardy jobs. For this new scheduling problem, we propose a bi-objective evolutionary algorithm. A set of experiments is performed using real data from the company's database. Experimental results show the relevance of the proposed procedure in terms of performance metrics, as opposed to the current manual scheduling procedure as well as against available commercial schedulers. The procedure was actually implemented as the scheduler procedure at the company. Such results also illustrate the improvement in key performance metrics of the production line.

Key words

Manufacturing industries-Automation, production scheduling, evolutionary algorithm.

Resumo

Neste artigo estuda-se o problema de programar a produção numa empresa de manufaturação de selos para roupa, estampados por transferência térmica. O problema é modelado utilizando a configuração de uma flowshop flexível, com o objetivo duplo de minimizar o lapso da fabricação (makespan) e o número de trabalhos atrasados. Sendo este um problema inovador na literatura, se propõe um enfoque da resolução biobjetivo, baseado em algoritmos evolutivos. A partir de um conjunto de dados de ordens reais da empresa, os experimentos mostram a pertinência do procedimento proposto, tanto em términos comparativos, com respeito à forma de programação manual da produção, como com respeito a procedimentos conhecidos na literatura. Os resultados da implementação na empresa nos últimos meses mostra, igualmente, um melhoramento considerável nos indicadores próprios do desempenho da linha de produção.

Palavras dicas

Indústrias de manufaturação, automatização, programação da produção, algoritmo evolucionista.

Introducción

La industria de las marquillas estampadas por transferencia térmica es una de las que mayor crecimiento ha presentado en la última década en Colombia (Vargas-Nieto, 2007). El mercado colombiano de confecciones demanda cada vez más que en las marquillas que aplican en sus prendas se maneje toda clase de información variada (tallas, estilos, referencias, códigos de barra, etc.). Al mismo tiempo, se exige una flexibilidad en el tamaño de la orden de pedido por referencia, que les permita ordenar solamente las cantidades exactas que necesitan. En los pedidos realizados por los clientes, muchas veces la cantidad requerida no sobrepasa las 500 unidades, lo cual dificulta su fabricación a través de los procesos tradicionales de producción: marquilla tejida y marquilla estampada en flexografía. Por lo anterior y por el manejo de tan variada información, la impresión de marquillas por transferencia térmica ha ganado mucho mercado en Colombia.

La empresa bajo estudio cuenta con centros de producción en las ciudades de Barranquilla, Bogotá, Medellín y Cali. Por razones de confidencialidad, no se puede mencionar el nombre de la compañía. En cada uno de sus centros de producción se programan pedidos que llegan diariamente. En los dos últimos años el crecimiento de las ventas ha ocasionado un incremento del 50% en el tiempo de entrega, a pesar de contar con la suficiente capacidad instalada para atender la demanda. Sin embargo, en el último semestre, este objetivo de entrega ha presentado un crecimiento de casi el 250%, lo cual ha empezado a amenazar la permanencia de cuentas importantes para la compañía.

Al revisar las causas de este incremento en entregas tardías, se ha identificado que con el crecimiento de la demanda, el número de pedidos por producir se ha hecho cada vez más difícil de programar y manejar. Por esto, se ha hecho necesario implementar una aplicación de *software*, basada en un modelo de programación de la producción (*scheduling*) multiobjetivo, a fin de planificar los pedidos, y así minimizar el lapso total de fabricación de cada orden y el número de trabajos entregados tardíamente.

Hasta la fecha, la programación de operaciones en la división de marquillas térmicas de esta empresa se ha realizado de forma empírica y muy intuitiva, basada en el conocimiento y la experiencia del supervisor de producción de cada centro de manufactura. Esto ha generado cierto desorden en este tema y ha ayudado a que el objetivo de entregas —establecido inicialmente de 48 horas— no se haya conseguido completamente. Lo anterior, al tiempo que le deja todo el trabajo de programación al buen juicio de una persona, ha generado una pérdida en el balance de carga de trabajo por máquina y ha hecho que la capacidad instalada con la que cuentan los centros de producción se esté mal utilizando y que diariamente el supervisor de producción se vea enfrentado a trabajar con un mayor número de órdenes "urgentes".

Por todo lo anterior, esta investigación pretende establecer un nuevo enfoque multiobjetivo para el problema de la planificación de las operaciones para la empresa en cuestión, perteneciente a la industria del sector de las marquillas estampadas por transferencia térmica. La herramienta de ayuda al proceso de toma de decisiones busca mejorar la productividad de los centros de producción, a través de la minimización del número de entregas tardías y la maximización de la productividad del sistema para una orden dada, para así crear ventajas competitivas para la empresa.

Este artículo está organizado de la siguiente manera. En primer lugar, se describe más detenidamente el sistema de fabricación de marquillas por transferencia térmica. En segundo lugar, se presenta el modelo de taller, conforme a la teoría clásica de programación de operaciones (*scheduling*), al igual que una breve revisión de la literatura relacionada. Esta sección presenta algunos conceptos básicos sobre optimización multiobjetivo y sobre algoritmos evolutivos. En tercer lugar, se describe el método propuesto, el cual está basado en un algoritmo evolutivo biobjetivo, junto con los desarrollos experimentales y el análisis de los resultados. En la cuarta y quinta partes se presentan los resultados de la implementación como herramienta de programación de la producción en la planta. Por último, el artículo termina presentando las conclusiones del trabajo y algunas perspectivas.

1. Descripción y modelado del proceso de fabricación

1.1 Descripción del proceso de fabricación de marquillas por transferencia térmica

En la división de marquillas por transferencia térmica de la compañía se fabrican dos tipos de marquillas: marquillas en base satín y marquillas en base nailon. Las marquillas pueden estamparse por el frente únicamente o por el frente y el

reverso. Todas las marquillas son corte cuchilla y son impresas en negro o en colores.

El proceso de fabricación consta de las operaciones de impresión y corte. Sin embargo, dependiendo de la máquina que se esté utilizando, se pueden hacer estas dos operaciones en un mismo proceso. La empresa cuenta con impresoras de transferencia térmica con un solo cabezal de impresión y sólo pueden imprimir un solo color por pasada. Estas máquinas, además, traen un cortador apilador apto para cortar marquillas en base de nailon únicamente.

También hay a disposición máquinas cortadoras ultrasonido, aptas para cortar marquillas en base satín, las cuales ejecutan la operación de corte para las marquillas estampadas en base satín. Esta operación de corte constituye la segunda etapa en este proceso. Para este estudio se tendrá en cuenta la distribución de máquinas más común que se encuentra en la empresa: cuatro impresoras y tres cortadoras ultrasonido.

Los siguientes son algunos aspectos importantes que se deben considerar respecto a la programación de operaciones en la industria de las marquillas estampadas por transferencia térmica:

- Si una marquilla que lleve más de un color se quiere programar en una máquina impresora, se deben hacer tantas pasadas como colores requiera.
- La velocidad de la máquina impresora cuando trabaja con el cortador adjunto se reduce en un 20% respecto a cuando no trabaja con este cortador.
- Los alistamientos de las producciones son muy cortos respecto al tiempo total de la operación. Por lo tanto, pueden pueden considerarse nulos.
- Existen operarios suficientes para atender todas las máquinas.

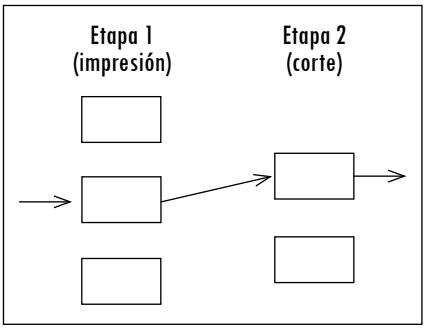
1.2 Modelamiento del proceso de fabricación y literatura relacionada

El problema de programación de operaciones para fabricar marquillas térmicas puede ser modelado como un *flowshop* flexible (FFS). De forma general, el FFS consiste en un taller con k etapas de fabricación en serie, y en cada una de las s ($s=1,...,k$) etapas se tienen m_s máquinas paralelas idénticas. No existe limitación en cuanto al inventario de producto de proceso entre dos etapas sucesivas. La tarea o la orden de producción (llamada *job*) se denota por j ($j=1,...,n$) y debe ser procesada en una máquina y sólo una en cada una de las etapas.

El tiempo de procesamiento de la tarea j en las diferentes etapas es conocido y denotado por p_{1j} ($p_{2j},...,p_{kj}$). El objetivo es encontrar una secuencia de ejecución sin interrupción de las tareas y una asignación de máquinas en cada una de las etapas, de manera que se optimicen una o varias funciones objetivo. La Figura 1

ilustra el esquema de un FFS con dos máquinas, tal como es el caso de la fabricación marquillas estampadas por transferencia térmica.

Figura 1. Esquema *flowshop* flexible



Fuente: presentación propia de los autores.

El problema de FFS cuando el objetivo es la minimización del *makespan* del lapso de fabricación o es conocido como *NP-completo* (Gupta, 1988). Esto quiere decir que no es posible encontrar la solución óptima para grandes instancias del problema en un tiempo de cálculo razonable. Por consiguiente, el problema multiobjetivo considerado en este trabajo es, al menos, igual de complejo.

Los trabajos de investigación presentados en la literatura se han focalizado, en su mayoría, en estudiar el problema con dos etapas. Se han propuesto varios métodos exactos, entre los cuales están formulaciones de programación matemática, algoritmos de ramificación y acotamiento, algoritmos de aproximación garantizada y heurísticas dedicadas (Brah y Hunsucker, 1991; Chen, 1995; Haouari y M'Hallah, 1997; Dessouky *et al*., 1998; Moursli y Pochet, 2000). Investigaciones sobre los últimos avances para este tema se proponen en (Chen, 1994; Linn y Zhang, 1999).

Para talleres de mayor tamaño, es decir, para configuraciones con más de dos etapas ($k \geq 3$) y una sola función objetivo, existen muy pocos trabajos. La función objetivo más estudiada ha sido el lapso de fabricación o *makespan*. Entre los procedimientos de resolución propuestos se encuentran métodos de ramificación y acotamiento, programación dinámica, modificaciones de los heurísticos para el problema con dos etapas, metaheurísticos, al igual que cotas inferiores para la solución óptima (Lee y Vairaktarakis, 1994; Soewandi y Elmaghraby, 2001; Riane *et al*., 1998; Vignier *et al*., 1997; Acero *et al*., 2004).

Algunos modelos estudiados tienen en cuenta configuraciones con otras funciones objetivo o con restricciones adicionales al problema clásico. Entre esos trabajos se encuentran: restricciones de disponibilidad de recursos (Allaoui y Artiba, 2006), minimización del tiempo de flujo o tiempo de terminación total (Azizoglu *et al*., 2001; Guinet y Solomon, 1996), recirculación de trabajos (Bertel y Billaut, 2004), restricciones de precedencia (Botta-Genoulaz, 2000; Tang *et al*., 2006), tiempos de procesamiento dependientes del estado de las máquinas (Sriskandarajah y Wagneur, 1991), almacenamiento limitado del producto en proceso (Sawik, 2002), interrupción de procesamiento de trabajos (Djellab y Djellab, 2002), o inclusive procesamiento por lotes de trabajos (Xuan y Tang, 2007).

Así mismo, en la literatura se han propuesto y analizado procedimientos metaheurísticos para varias versiones del problema FFS (por ejemplo, Portmann *et al*., 1998; Jin *et al*., 2006; Ruiz y Maroto, 2006). Para el problema con objetivos múltiples, la mayoría de los trabajos publicados se ha concentrado en el problema de minimización del *makespan* y del tiempo promedio de flujo o de terminación de las tareas (T'Kindt y Billaut, 2006). Algunos trabajos también consideran las mismas funciones objetivo, pero con algunas restricciones adicionales (por ejemplo, Bertel y Billaut, 2004; Morita y Shio, 2005).

El caso del FFS con optimización simultánea del lapso de fabricación y del número de trabajos tardíos no se ha estudiado hasta la fecha, al menos en nuestro conocimiento de la literatura. En este trabajo se presenta el diseño y la implementación de un procedimiento que se basa en algoritmos evolutivos para resolver el problema de minimización lapso de fabricación y el número de trabajos tardíos en un FFS con dos estaciones y múltiples máquinas en cada estación.

Como ya se mencionó, este problema se inspiró de una aplicación real en el sector de la manufactura de marquillas estampadas por transferencia térmica. Los datos necesarios para la implementación y la experimentación se tomaron de la fábrica bajo estudio.

2. Conceptos fundamentales

2.1 Fundamentos de la optimización multiobjetivo

Los problemas reales usualmente requieren la búsqueda de soluciones que satisfagan de forma simultánea múltiples criterios de desempeño u objetivos, los cuales pueden ser contradictorios. Cuando es factible combinar los objetivos de un problema de manera adecuada, es posible considerar un único objetivo por optimizar. En

este caso, para obtener la solución del problema basta con encontrar el mínimo o el máximo de una única función (transformación lineal) que pondere todos los objetivos que se desean optimizar. Sin embargo, lo usual es que no se conozca la manera óptima de combinar los diferentes objetivos, cuando no imposible hacerlo. En este caso, se dice que el problema es un problema de optimización multiobjetivo o *multiobjective optimization problem* (MOP).

En problemas de optimización multiobjetivo con objetivos contradictorios no siempre existe una solución única que pueda considerarse la mejor, sino un conjunto de soluciones que representa los mejores compromisos entre los distintos criterios. Dicho conjunto es llamado conjunto Pareto-óptimo y su imagen en el espacio objetivo es denominada *frente Pareto*.

Mientras en la optimización monobjetivo se busca un vector de decisión *n*-dimensional que optimice una función escalar, en la optimización multiobjetivo se intenta encontrar un vector que optimice una función vectorial, cuyos elementos representan las distintas funciones objetivo. Así, un problema de optimización multiobjetivo puede definirse formalmente de la siguiente manera (Ehrgott y Gandibleux, 2002; Collete y Siarry, 2004):

Definición 1. Problema de optimización multiobjetivo: un MOP general optimiza una función de la forma:

$$y = F(x) = (f_1(x), \ldots, f_k(x)) \tag{1}$$

Sujeto a:

$$g(x) = (g_1(x), \ldots, g_k(x)) \leq 0 \tag{2}$$

Donde: $x = (x_1, \ldots, x_n) \in X \subseteq \Re^n; y = (y_1, \ldots, y_k) \in Y \subseteq \Re^k$. x es una variable de decisión vectorial *n*-dimensional, y es un vector objetivo *k*-dimensional, $X \subseteq \Re^n$ denota el espacio de decisión y $Y \subseteq \Re^k$ denota el espacio objetivo. El conjunto de restricciones dadas por la ecuación (2) define la región de factibilidad $X_f \subseteq \Re^n$ y cualquier punto $x \in X_f$ es una solución factible.

En un MOP con objetivos contradictorios, el espacio de búsqueda se encuentra sólo parcialmente ordenado y dos soluciones pueden ser indiferentes entre sí. Es poco usual que una única variable de decisión optimice de manera simultánea todos los objetivos. Consecuentemente, para los MOP se extienden los operadores $<$, $>$e$=$ del siguiente modo:

Definición 2. Dominancia de Pareto: sean los vectores objetivo $u=(u_1,...,u_k)$ y $v=(v_1,...,v_k)$. Se dice que u domina a v y se denota como $u \succ v$, si y sólo si $\forall i \in \{1,...,k\}$, u_i es mejor o igual que $v_i \wedge \exists i \in \{1,...,k\} \mid u_i$ es mejor que v_i.

Es decir, u domina a v, si u es mejor o igual a v en todos los objetivos y estrictamente mejor en al menos un objetivo. Si ninguno de los vectores domina al otro se dice que son no comparables entre sí, ya que ninguno puede ser considerado mejor que el otro, considerando todos los objetivos.

Definición 3. Se dice que una solución $x \in X_f$ es un Pareto-óptimo respecto a un conjunto $\Omega \subseteq X_f$ si y sólo si $\exists x' \in \Omega$ para el cual $v'=F(x')$ domina a $u=F(x)$.

Cuando resulta claro a qué conjunto Ω se hace referencia, simplemente no se menciona. La solución x es Pareto-óptima si y sólo si es no dominada con respecto al conjunto X_f. Esto es, un vector x es Pareto-óptimo si no existe otro vector de decisión factible $x' \in X_f$ que lo domine. El conjunto de soluciones en un MOP, denotado por P^*, está compuesto por todos estos vectores Pareto-óptimos.

Las soluciones Pareto-óptimas son también llamadas soluciones no inferiores, admisibles o soluciones eficientes, mientras sus vectores objetivo correspondientes son denominados *no dominados*. El conjunto de vectores no dominados en el espacio objetivo, correspondiente al conjunto de soluciones P^*, es llamado *frente Pareto-óptimo* y denotado por PF^*.

Definición 4. Conjunto Pareto-óptimo y frente Pareto-óptimo. Dado un problema de optimización mutiobjetivo $F(x)$, el conjunto Pareto-óptimo, denotado por P^*, se define como: $P^* = \{x \in X_f \mid \exists x' \in X_f$ para el cual $F(x')$ domine $F(x)\}$. El frente Pareto-óptimo PF^* correspondiente se define como $PF^*=\{u=F(x)/x \in P^*\}$.

2.2 Algoritmos evolutivos

Los algoritmos evolutivos son métodos de optimización que utilizan el modelo computacional de selección natural. Los algoritmos evolutivos han probado ser particularmente exitosos en la solución de problemas difíciles de formalizar matemáticamente y que, por ende, no se pueden solucionar a través del análisis clásico, basado en herramientas de ingeniería industrial e investigación de operaciones.

El hecho de que el funcionamiento de los algoritmos evolutivos no esté basado en heurísticas de dominio específico los hace atractivos para ser utilizados en sistemas que son altamente no lineales, estocásticos y poco entendidos. Esto gracias a que se requiere muy poca información del problema para implementar un algoritmo evolutivo. Para problemas bien entendidos, que son lineales y para los cuales existen soluciones confiables, los algoritmos evolutivos producen resultados probablemente poco competitivos.

Para los algoritmos evolutivos es posible aportar varias soluciones diferentes a un problema, pero igualmente buenas en calidad, debido a que utilizan una población inicial. Por lo tanto, los algoritmos evolutivos son un método de búsqueda y optimización robusto que está en capacidad de enfrentarse a la multimodalidad, discontinuidad, variación en el tiempo, aleatoreabilidad y ruido que se puede encontrar fácilmente en varios problemas de optimización.

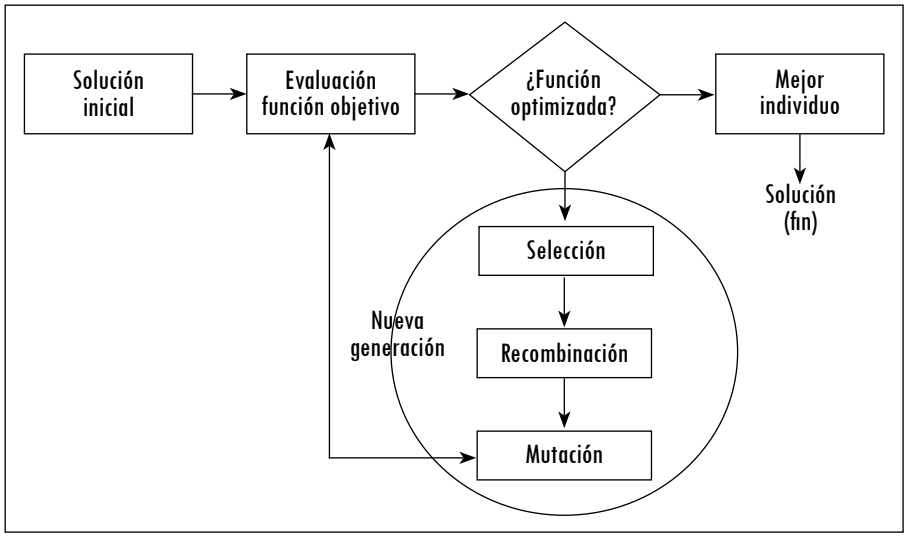
Los algoritmos evolutivos se han aplicado amplia y exitosamente a diseños de aplicaciones para trabajar fuera-de-línea. En el campo de sistema de controles en ingeniería, estas aplicaciones incluyen diseño de controladores, identificación de modelos, análisis de estabilidad robusta, sistemas de confiabilidad, diagnóstico de fallas, programación de operaciones, entre otros (Ehrgott y Gandibleux, 2002).

Los algoritmos evolutivos tienen varias implementaciones con leves diferencias, como son: programación evolutiva, estrategias evolutivas, algoritmos genéticos y programación genética. La selección de la técnica apropiada y el ajuste de parámetros para la técnica seleccionada requieren el conocimiento de estas técnicas.

Los algoritmos evolutivos trabajan con una población de potenciales soluciones a un problema, en la cual cada individuo que pertenece a la población representa una solución particular, generalmente representada en la forma de código genético. Se define el valor *fitness* de cada individuo como el valor que expresa qué tan buena es la solución representada por el individuo para resolver el problema. A las mejores soluciones se les asignan valores de *fitness* más altos que las peores soluciones. La clave de los algoritmos evolutivos es que el valor *fitness* de un individuo determina también qué tan bueno será este individuo para propagar sus genes en las generaciones siguientes.

La Figura 2 describe un algoritmo evolutivo típico. Primero se genera una población inicial de manera aleatoria, luego se evalúa el *fitness* de cada individuo con base en las funciones objetivo del problema y, finalmente, evolucionan de generación en generación a través de la aplicación de los pasos de evaluación, selección, mutación y recombinación. En el paso de selección, el algoritmo elige los padres para la próxima generación. La población es sujeta a la "presión del ambiente", lo que significa que sólo el individuo con el mejor *fitness* prevalecerá a lo largo de las generaciones. Los más importantes métodos de selección de soluciones para algoritmos evolutivos son el muestreo uniforme estocástico, la selección por torneo, la selección por ordenamiento de *fitness* y la selección proporcional de *fitness* (Ehrgott y Gandibleux, 2002).

Figura 2. Algoritmo evolutivo típico



Fuente: presentación propia de los autores.

La población de soluciones evoluciona a lo largo de las generaciones para producir mejores soluciones al problema. Esta evolución se realiza utilizando un conjunto de operadores genéticos estocásticos, los cuales manipulan el código genético usado para representar potenciales soluciones. La mayoría de los algoritmos evolutivos incluyen operadores que seleccionan individuos para reproducirlos y crear nuevos individuos basados en aquellos individuos seleccionados, lo que determina así la composición de la población de la generación subsiguiente.

Después de la selección de individuos, los nuevos individuos de la próxima generación (llamados hijos) son creados a través de la recombinación y la mutación. El método de recombinación (llamado también *crossover*) intercambia información genética entre dos individuos seleccionados para crear uno o dos hijos. El método de mutación hace pequeños cambios aleatorios a la información genética de un individuo. El paso final de los algoritmos evolutivos es el reemplazamiento, que ocurre cuando los nuevos individuos son insertados en la nueva población. Una vez la nueva generación ha sido construida, se comienzan de nuevo todos los procesos anteriores.

En muchos problemas de optimización se desea optimizar simultáneamente varios objetivos y restricciones. Los algoritmos evolutivos han sido capaces de hacerlo de una manera efectiva. En algunos casos se trabaja con la combinación

de objetivos y restricciones en un único valor de *fitness*, pero en otros se han utilizado técnicas basadas en el frente Pareto.

3. Procedimiento de programación propuesto

Este trabajo presenta un nuevo modelo de programación de operaciones biobjetivo, basado en algoritmos evolutivos para la industria de la marquilla estampada por transferencia térmica. En el momento de utilizar el heurístico se considera que un pedido no visita la misma máquina dos veces, cada máquina sólo puede procesar un pedido a la vez, no se puede interrumpir la fabricación de un pedido en cada máquina y las máquinas están disponibles en cualquier intervalo de tiempo. A continuación se describe la configuración del algoritmo.

3.1 Representación de la solución

El algoritmo evolutivo desarrollado utiliza la representación de solución o cromosoma ilustrado en la Figura 3. Este vector contiene los números de los trabajos (1, 2, 3,…, *n*) y los números de las impresoras termales disponibles en el momento de hacer la programación (I_1, I_2, I_3,…, I_m). Para el ejemplo se ilustra un cromosoma con 4 impresoras y 10 pedidos. Este vector define la secuencia de programación para los pedidos en cada impresora. Su correspondiente orden de salida del área de impresión determina la secuencia que se manejará en el área de corte: *Fist in First out* (FIFO), con asignación de máquina menos cargada.

Figura 3. Ejemplo del cromosoma

I_1	1	2	I_2	5	6	I_3	8	10	3	I_4	4	7	9

Fuente: presentación propia de los autores.

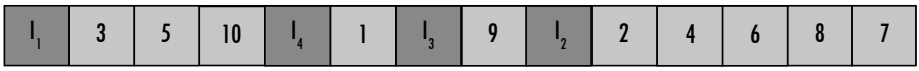
3.2 Operador de crossover o de cruzamiento

El operador de *crossover* manejado en el algoritmo corresponde a un *crossover* uniforme y se ilustra a continuación:

Padre 1:

I_1	1	2	I_2	5	6	I_3	8	10	3	I_4	4	7	9

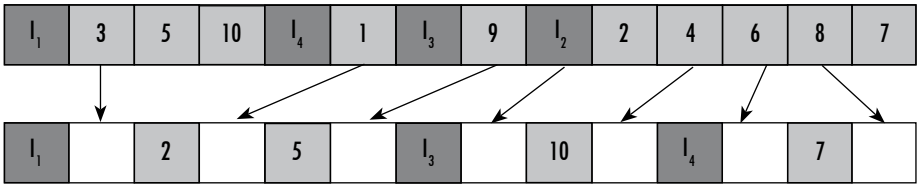
Padre 2:



Los genes del cromosoma Padre 1 que pasarán al Hijo 1 se seleccionan de manera uniforme:

Hijo 1:

Luego se llenan los genes vacíos del Hijo 1 con los genes que no están en el Hijo 1, pero que sí están en el Padre 2, de manera secuencial:

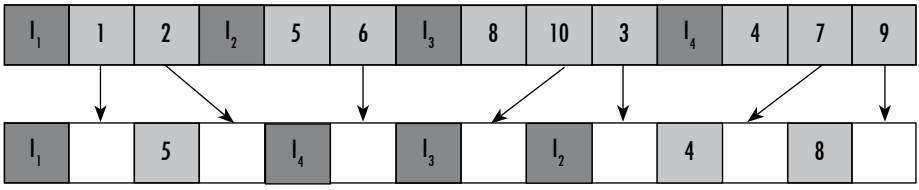


El Hijo 1 resultante sería:

A continuación se seleccionan de manera uniforme los genes del cromosoma Padre 2 que pasarán al Hijo 2:

Hijo 2:

Luego se llenan los genes vacíos del Hijo 2 con los genes que no están en el Hijo 2, pero que sí están en el Padre 1, de manera secuencial:



El Hijo 2 resultante sería:

3.3 Operador de mutación

Para la mutación, el algoritmo utiliza un operador de mutación por intercambio. Se selecciona un cromosoma padre:

Luego se seleccionan aleatoriamente dos genes dentro de este Padre, los cuales serán intercambiados de posición:

El cromosoma hijo resultante será:

3.4 Parámetros del algoritmo evolutivo

Para este algoritmo evolutivo se definieron los siguientes parámetros: la probabilidad de *crossover* se tomará con valores de 0,5 y 0,8. La probabilidad de mutación se mantendrá fija con un valor de 0,3. El tamaño de la población inicial será de 167 individuos, mientras el tamaño de la población de no dominados será de 20 individuos. Algunas variaciones en otros de los parámetros del algoritmo se presentan en el estudio experimental. Es importante anotar que el algoritmo está diseñado para que se puedan variar todos los parámetros y así realizar más pruebas.

3.5 Pasos del algoritmo desarrollado

El modelo de algoritmo evolutivo desarrollado consiste en los siguientes pasos:

Paso 1. Generar la población inicial. La población inicial se genera utilizando un procedimiento iterativo, que asigna el primer gen para todos los individuos de la población, escogiéndolo del conjunto de máquinas impresoras disponibles. El resto de genes se selecciona aleatoriamente de ambos conjuntos: impresoras y trabajos.

Paso 2. Calcular los tiempos de impresión. Los tiempos de impresión se calculan para cada trabajo en cada solución como el tiempo en el cual un trabajo terminó su proceso de impresión, de acuerdo con la secuencia establecida por el cromosoma de la solución para cada individuo de la población.

Paso 3. Asignar los trabajos a las máquinas de corte. Los trabajos se asignan a las máquinas de corte de acuerdo con la regla *Modified First-in-First-Out* (M-FIFO), que considera el balance de carga entre las máquinas de corte en el momento de asignar un trabajo.

Paso 4. Calcular los tiempos de terminación. Los tiempos de terminación para cada trabajo se calculan para cada individuo como el tiempo en el cual cada trabajo terminó su operación de corte, de acuerdo con la secuencia establecida por el cromosoma de la solución para cada individuo de la población.

Paso 5. Calcular el lapso de fabricación y el número de trabajos tardíos. El lapso se calcula como el tiempo de terminación que sea el más largo entre todos los individuos de la población. El número de trabajos tardíos se calcula a través de una comparación de los tiempos de terminación para cada trabajo con las fechas de entrega de cada trabajo para cada individuo de la población.

Paso 6. Calcular el fitness. El *fitness* para el individuo i de la población se calcula a partir de la fórmula $fitnes_i = n/N$, donde n es el número de soluciones que el individuo i domina entre la población, y N es el tamaño de la población.

Paso 7. Seleccionar los individuos. Los individuos de la población con mejor valor *fitness* se seleccionan y copian al archivo de soluciones no dominadas.

Paso 8. Recombinar las soluciones no dominadas. Aplicar los operadores de *crossover* y de mutación descritos a las soluciones no dominadas, teniendo en cuenta las probabilidades respectivas.

Paso 9. Selección de los nuevos individuos. Copiar los nuevos individuos resultantes de la recombinación del paso 8 y luego regresar al paso 6.

Este procedimiento se debe repetir hasta que se alcance el número máximo de generaciones.

4. Experimentación con el procedimiento

Para implementar el algoritmo evolutivo en la empresa se tomaron 21 trabajos, 4 máquinas impresoras y 3 máquinas cortadoras. Los datos utilizados para esta simulación están en la Tabla 1. Como se mencionó, los objetivos bajo los cuales se evalúa el desempeño del algoritmo son el lapso de fabricación (denotado como

C_{max} en la notación clásica) y el número de trabajos tardíos. La escogencia de estos datos de la base de datos de la empresa se hizo de forma aleatoria. Se consideró de particular importancia hacer seguimiento de la producción real de este conjunto de órdenes de fabricación, con el fin de poder comparar el desempeño del algoritmo propuesto respecto al desempeño real en la planta de manufactura. Dicha comparación se presenta al final de esta sección, en el análisis de resultados.

Tabla 1. Datos para la experimentación con el algoritmo evolutivo

Trabajo	Tiempo en impresión (h)	Tiempo en corte (h)	Tiempo total (h)	Fecha de entrega (h)
1	0,84	0,35	1,19	16
2	1,63	0,42	2,05	16
3	1,70	0,00	1,70	16
4	3,90	0,00	3,90	32
5	2,80	3,47	6,27	32
6	15,55	7,50	23,05	16
7	6,40	1,74	8,14	16
8	14,11	3,40	17,51	16
9	6,40	0,00	6,40	16
10	6,40	0,00	6,40	16
11	1,52	0,31	1,83	16
12	0,36	0,15	0,51	16
13	0,35	0,21	0,55	16
14	0,03	0,01	0,04	16
15	0,16	0,14	0,30	16
16	2,24	1,39	3,63	16
17	7,20	0,00	7,20	16
18	0,26	0,00	0,26	16
19	0,64	0,69	1,33	16
20	47,84	15,97	63,81	64
21	40,77	13,61	54,38	64

Fuente: presentación propia de los autores.

Con base en estos datos, se realizaron los experimentos de parametrización del algoritmo, presentados a continuación. Al constatar el comportamiento interesante del algoritmo propuesto, se llevó a cabo un seguimiento continuo

de la implementación del método en la programación real de la planta. Para esto se capacitó a los diferentes supervisores de producción en el uso de la herramienta. Los resultados sobre dicha implementación se presentan y explican con detalle más adelante.

4.1 Experimentos y análisis de resultados

En la Tabla 2 se presentan los parámetros definidos para los experimentos realizados con el algoritmo evolutivo desarrollado. Con diversas combinaciones de estos parámetros, se realizaron un total de 13 experimentos. Los resultados se presentan en las figuras 3 a 15. La programación del algoritmo se realizó utilizando macros de Microsoft® Excel. Los experimentos se corrieron en un computador de escritorio con procesador AMD® 2400 Plus, 2,2 GHz y 512 MB de RAM. Bajo estas condiciones, el tiempo de cálculo requerido para obtener la solución final después de 100 generaciones es cercano a 3 minutos y 10 segundos. Esto hace atractivo el algoritmo para ser empleado en el proceso real de toma de decisiones, puesto que el tiempo de cálculo es competitivo respecto a la frecuencia con la que se necesita programar las operaciones en la compañía (alrededor de 2 veces al día).

Tabla 2. Valores de los parámetros para los experimentos

Parámetro	Notación	Valores
Tamaño de la población inicial	N	50, 167 y 250
Tamaño del archivo de no dominados	*ND*	20
Probabilidad de *crossover*	P_c	0,5 y 0,8
Probabilidad de mutación	P_m	0,3
Número de generaciones	*G*	1, 5, 10, 20 y 100

Fuente: presentación propia de los autores.

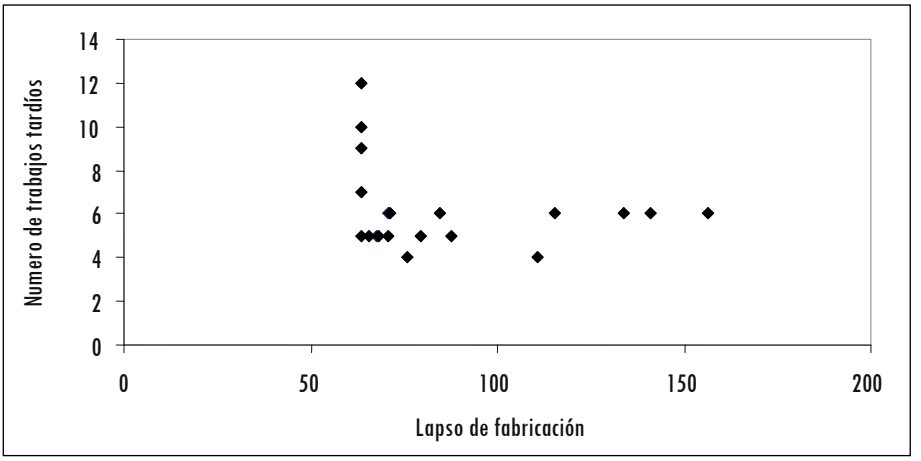
Experimento 1: N=167; ND=20; P_C=0,5; P_M=0,3; G=1.

Figura 3. Frente Pareto para el Experimento 1

Fuente: presentación propia de los autores.

Experimento 2: N=167; ND=20; P_C=0,8; P_M=0,3; G=1.

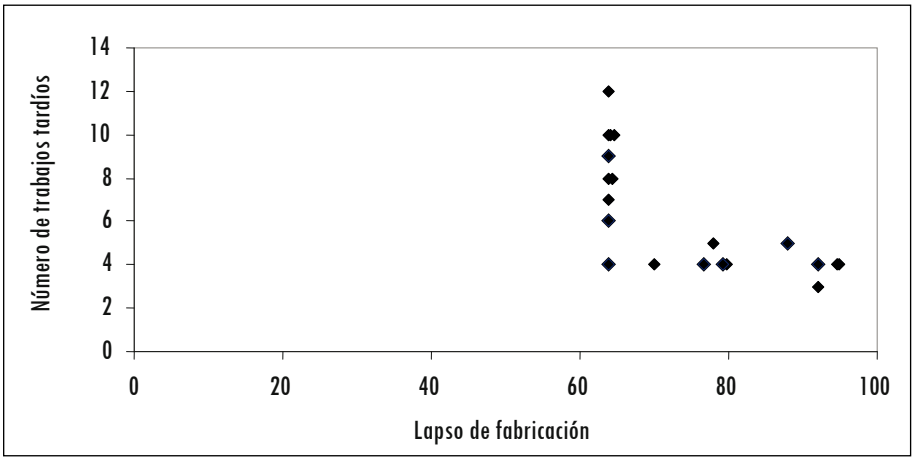
Figura 4. Frente Pareto para el Experimento 2



Fuente: presentación propia de los autores.

Experimento 3: N=167; ND=20; P_C=0,5; P_M=0,3; G=5.

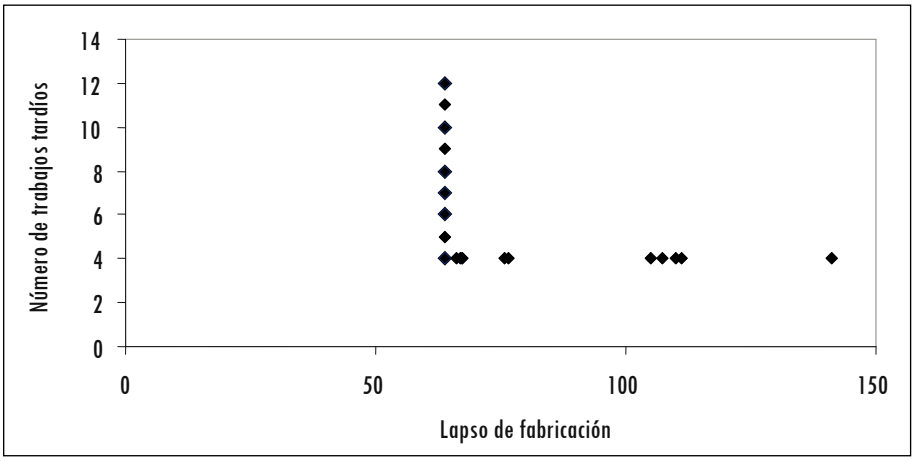
Figura 5. Frente Pareto para el Experimento 3



Fuente: presentación propia de los autores.

Experimento 4: N=167; ND=20; P_C=0,8; P_M=0,3; G=5.

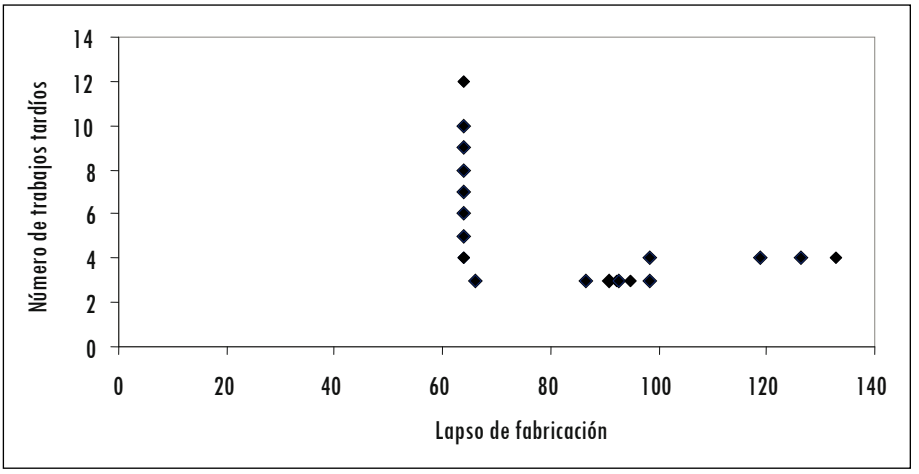
Figura 6. Frente Pareto para el Experimento 4



Fuente: presentación propia de los autores.

Experimento 5: N=167; ND=20; P_C=0,5; P_M=0,3; G=10.

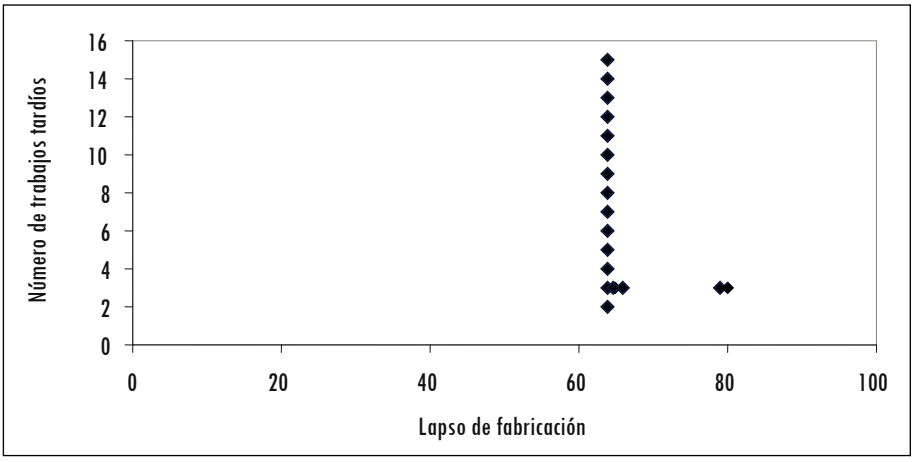
Figura 7. Frente Pareto para el Experimento 5



Fuente: presentación propia de los autores.

Experimento 6: N=167; ND=20; P_C=0,8; P_M=0,3; G=10.

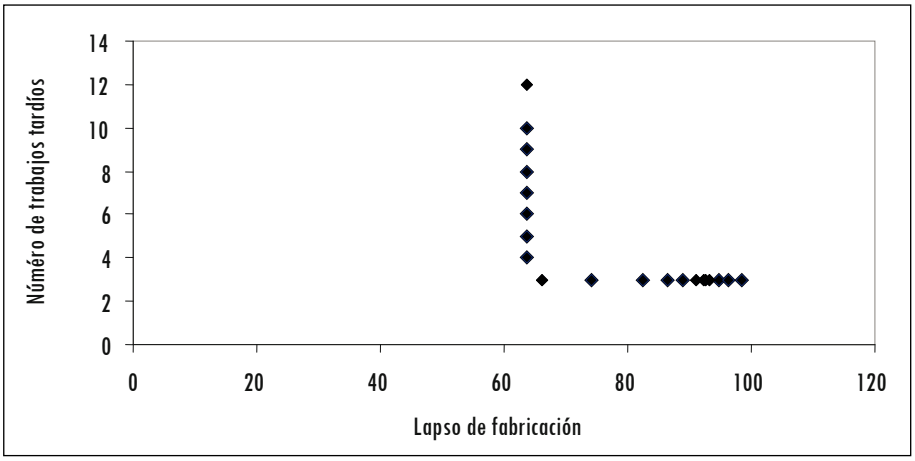
Figura 8. Frente Pareto para el Experimento 6



Fuente: presentación propia de los autores.

Experimento 7: N=167; ND=20; P_C=0,5; P_M=0,3; G=20.

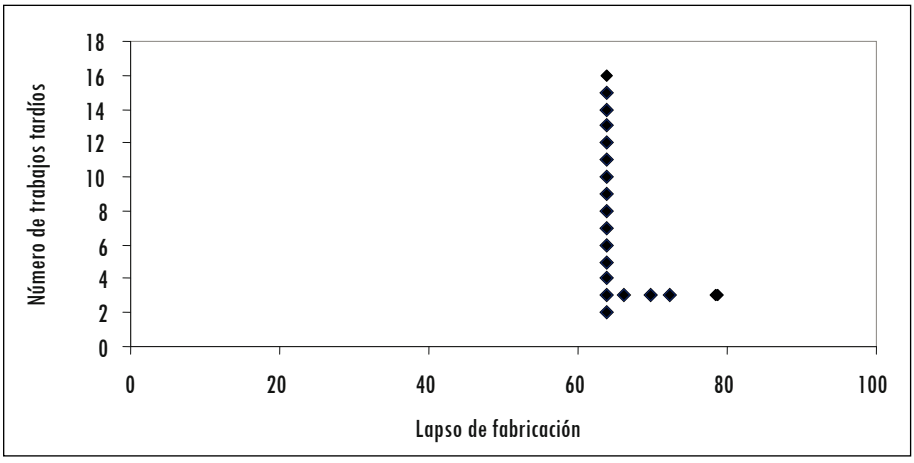
Figura 9. Frente Pareto para el Experimento 7



Fuente: presentación propia de los autores.

Experimento 8: N=167; ND=20; P_C=0,8; P_M=0,3; G=20.

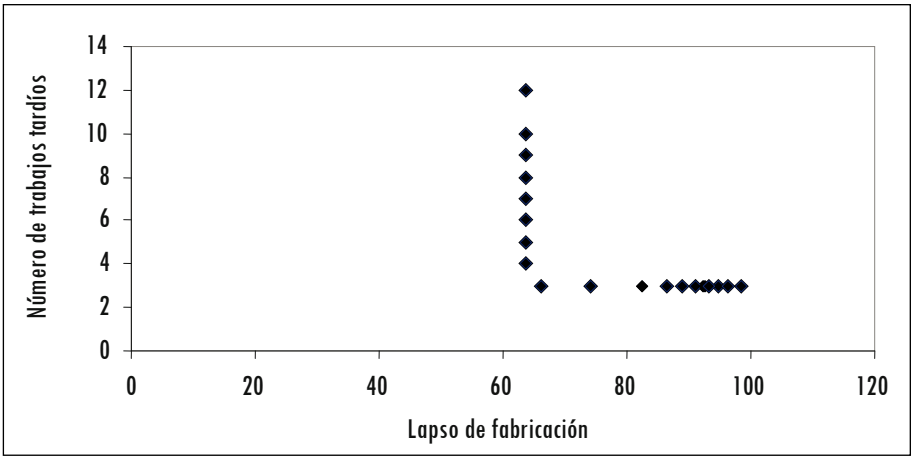
Figura 10. Frente Pareto para el Experimento 8



Fuente: presentación propia de los autores.

Experimento 9: N=167; ND=20; P_C=0,5; P_M=0,3; G=100.

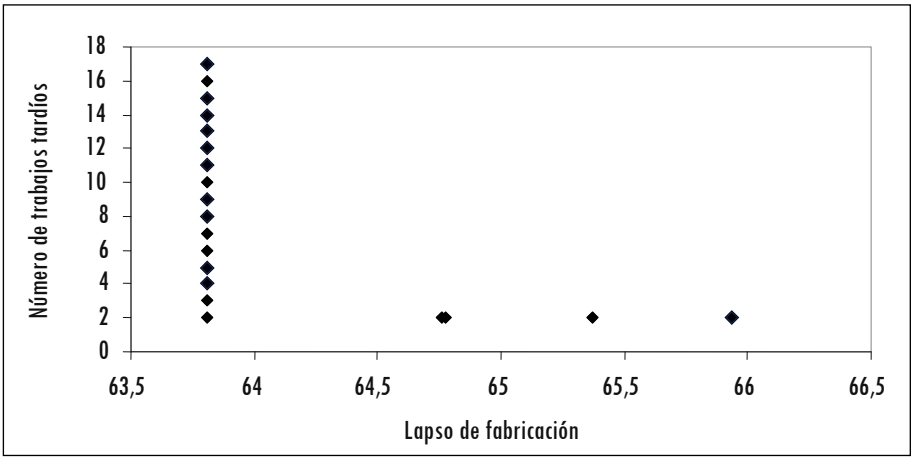
Figura 11. Frente Pareto para el Experimento 9



Fuente: presentación propia de los autores.

Experimento 10: N=167; ND=20; P_C=0,8; P_M=0,3; G=100.

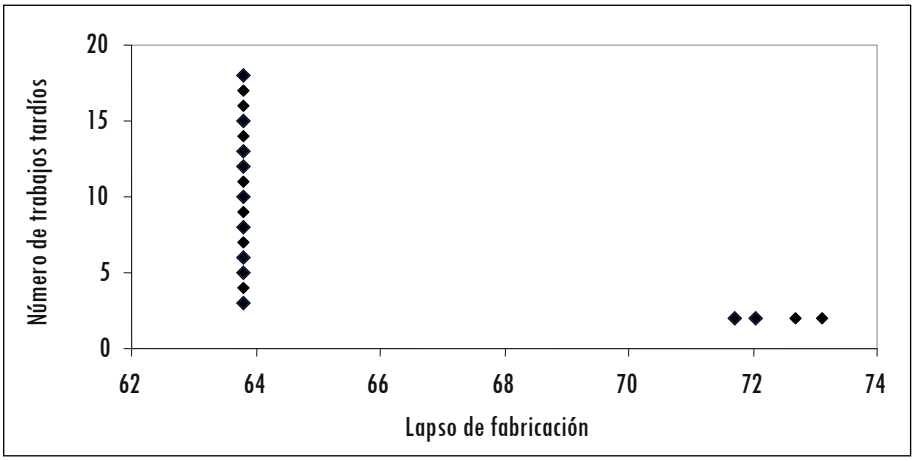
Figura 12. Frente Pareto para el Experimento 10



Fuente: presentación propia de los autores.

Experimento 11: N=50; ND=20; P_C=0,8; P_M=0,3; G=100.

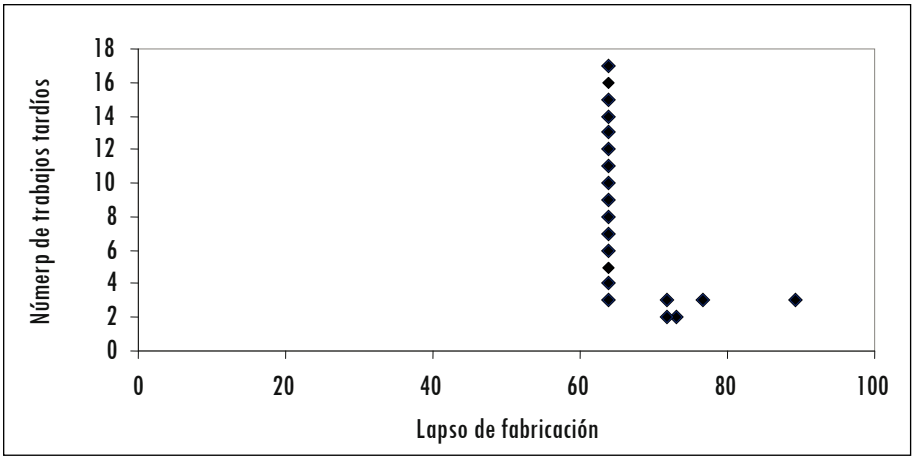
Figura 13. Frente Pareto para el Experimento 11



Fuente: presentación propia de los autores.

Experimento 12: N=50; ND=20; P_C=0,8; P_M=0,3; G=20.

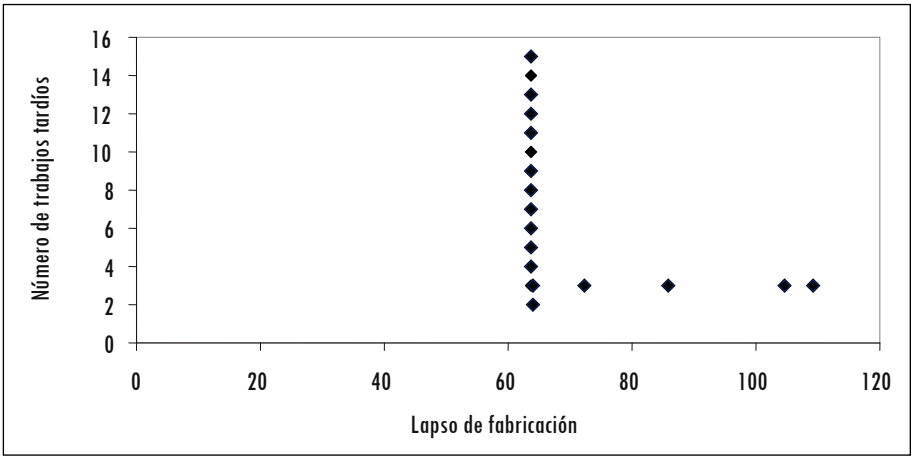
Figura 14. Frente Pareto para el Experimento 12



Fuente: presentación propia de los autores.

Experimento 13: N=250; ND=20; P_C=0,8; P_M=0,3; G=20.

Figura 15. Frente Pareto para el Experimento 13



Fuente: presentación propia de los autores.

Después de analizar los resultados de los experimentos anteriores, se evidencia que cuando se utiliza una probabilidad de *crossover* de 0,8, el frente Pareto es visible a partir de la primera generación, mientras que con una probabilidad de *crossover* de 0,5 se encuentran todavía varias soluciones que no están cercanas al frente Pareto. De esta forma, con la probabilidad 0,8, al arribar a la generación 20, el frente Pareto está prácticamente definido, tal como se muestra en el Experimento 8. En contraste, al utilizar una probabilidad de *crossover* de 0,5, como se hizo en los experimentos 1, 3, 5 ,7 y 9, es necesario llegar a un número más alto de generaciones para poder descubrir el frente Pareto.

Estos experimentos, aplicados en este caso de estudio a una industria real, verifican y confirman las hipótesis presentadas en la mayoría de trabajos literarios del tema de algoritmos evolutivos, en los cuales siempre se sugiere trabajar con probabilidades de *crossover* elevadas (superiores o iguales a 0,8) y con probabilidades de mutación bajas (inferiores o iguales a 0,3). Adicionalmente, se verifica que para el algoritmo evolutivo desarrollado en particular, los parámetros de tamaño de población inicial (*N*), tamaño del archivo (*ND*) y probabilidad de mutación (P_M) no influyen en la formación prematura del frente Pareto, como sí lo hacen la probabilidad de *crossover* (P_C) y el número de generaciones (*G*). Esto permite afirmar que un algoritmo evolutivo entregará soluciones con mejor calidad si el número de generaciones es superior a 100 y la probabilidad de *crossover* es alta ($P_C \geq 0{,}8$).

4.2 Selección de una solución única del frente Pareto

Después de correr este algoritmo evolutivo, el supervisor de producción de la compañía aún tiene que seleccionar cuál de las soluciones del frente Pareto va a ser implementada como un programa real en la planta. Con miras a ayudar al supervisor o programador de producción en esta labor, el algoritmo tiene un procedimiento adicional que selecciona una solución del frente Pareto.

Este procedimiento utiliza el criterio de balance de carga de trabajo entre todas las máquinas en ambas etapas del proceso productivo: impresión y corte. Teniendo en cuenta lo anterior, el algoritmo calcula la carga de trabajo por máquina para cada individuo perteneciente al frente Pareto resultante después del número de generaciones establecidas como parámetro. La solución con menor diferencia en la carga de trabajo entre la máquina más cargada y la máquina menos cargada en cada etapa (impresión y corte) es seleccionada para utilizarla como programa real para la planta.

Después de aplicar este criterio a todos los experimentos realizados y expuestos en este artículo, la mejor solución encontrada nos entregó los valores presentados en la Tabla 3 para las dos funciones objetivos en estudio. También se presenta una comparación con los valores actuales obtenidos en fabricación real bajo el esquema actual de programación manual. Al comparar estos resultados con los datos históricos suministrados por la compañía, se puede evidenciar una diferencia del 32% en el lapso de fabricación y del 400% en el número de trabajos tardíos.

Tabla 3. Comparación de resultados

Indicador de desempeño	Solución actual	Solución propuesta
Lapso de fabricación	84,52 h	63,81 h
Número de trabajos tardíos	8	2

Fuente: presentación propia de los autores.

Estos resultados nos llevan a pensar que con una correcta implementación de este algoritmo evolutivo en los distintos centros de fabricación de la compañía, los resultados pueden ser muy atractivos y pueden contribuir a que la compañía mejore sus ventajas competitivas y su nivel de servicio, respecto a su situación actual.

4.3 Comparación con otros procedimientos de la literatura

Con fin de complementar el análisis de desempeño del método propuesto, se decidió comparar los indicadores de lapso de fabricación y número de trabajos tardíos respecto a los valores que pueden obtenerse empleando las herramientas de programación de operaciones disponibles en el mercado. Para el mismo conjunto de datos descrito se utilizó el *software* Lekin® Scheduler (Pinedo, 2002). El modelo empleado fue el *flowshop* híbrido, y se resolvió la instancia estudiada utilizando el procedimiento *Shifting Bottleneck* (SB), rutina general para la minimización del lapso de fabricación (SB/C_{max}), y las variaciones correspondientes a la minimización del retraso máximo (SB/L_{max}) y de la tardanza total (SB/TT).

Estas dos últimas funciones objetivo se escogieron como indicador indirecto del nivel de servicio al cliente, en vista de que la rutina SB no tiene incorporada una función que permita minimizar directamente el número de trabajos tardíos. Los resultados obtenidos se presentan en la Tabla 4.

Tabla 4. Resultados comparativos con Lekin® Scheduler

Algoritmo	Mono o biobjetivo	Valor del lapso de fabricación	Número de trabajos tardíos	Nota
Propuesto	Biobjetivo	63,7	4	
General Shifting Bottleneck	Monoobjetivo	63,7	18	Minimiza el lapso de fabricación
General Shifting Bottleneck/L_{max}	Monoobjetivo	71,7	17	Minimiza el retraso máximo
General Shifting Bottleneck/*Total tardiness*	Monoobjetivo	74,0	6	Minimiza la tardanza total

Fuente: presentación propia de los autores.

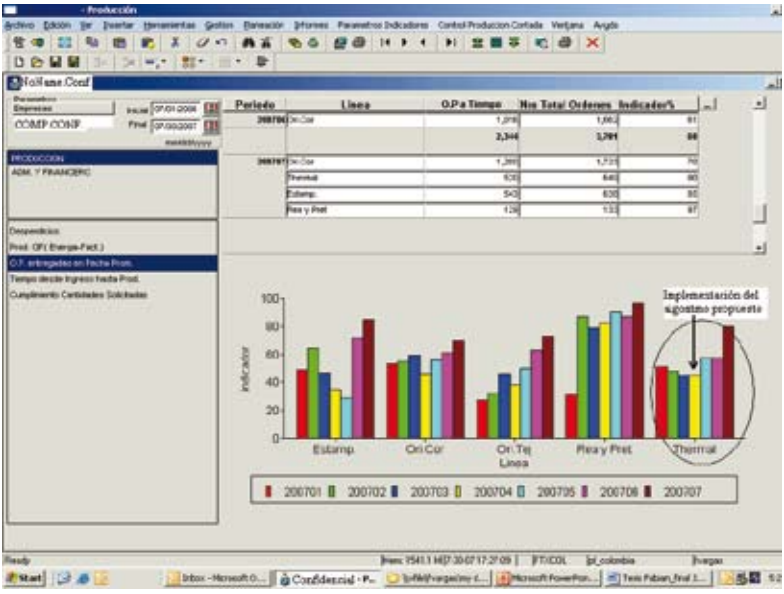
Con el fin de calcular la desviación porcentual del algoritmo propuesto respecto a cada una de las versiones del procedimiento SB, se empleó la relación $\%dev=(Z_{heur}-Z_{prop})/Z_{heur}$, donde Z_{heur} corresponde al valor del indicador (lapso de fabricación o número de trabajos tardíos), y Z_{prop}, al valor del indicador dado por el algoritmo presentado en este artículo. En vista de los resultados, se puede observar que con respecto a SB/C_{max} se obtiene el mismo valor para el lapso de fabricación; pero el procedimiento propuesto es mejor en un 78% en cuanto al

número de trabajos tardíos. Con respecto al procedimiento SB/L_{max}, el algoritmo propuesto es mejor en un 13% para el valor del lapso de fabricación y en un 325% para el número de trabajos tardíos. Finalmente, con respecto a SB/TT, los resultados muestran que el método propuesto es mejor en un 16% para el lapso de fabricación y en un 50% para el número de trabajos tardíos.

5. Resultados de la implementación en la empresa

Con base en el conjunto de datos presentado, se realizaron los experimentos de parametrización del algoritmo propuesto. Al constatar el comportamiento interesante de este algoritmo, se llevó a cabo un seguimiento real de la implementación del método en la programación real de la planta. Para esto se capacitó a los diferentes supervisores de producción en el uso de la herramienta. La Figura 16 presenta este seguimiento para el período enero-julio de 2007. Los diferentes grupos de columnas corresponden a las diferentes divisiones de la compañía. La línea de estudio es la correspondiente a la fabricación de marquillas para ropa, en la que se emplean procesos de transferencia térmica. El interés se centra en la evolución del porcentaje de entregas oportunas para el grupo de columna, ubicado más a la derecha de la figura (señalado dentro de un óvalo).

Figura 16. Porcentaje de cumplimiento de órdenes entre enero y julio de 2007



Fuente: presentación propia de los autores.

Igualmente, allí se observa el punto donde se empezó a implementar la programación de la producción, empleando el algoritmo presentado en este artículo (a partir de abril de 2007). Al tomar ese punto como referencia (la cuarta columna del último grupo de indicadores), precedente a la implementación del algoritmo, se veía un porcentaje de cumplimiento de las entregas cercano al 50%. A partir del punto de referencia, el cumplimiento de las entregas se ve incrementado, hasta llegar a un 80% en julio, y con ello iguala e incluso supera el indicador de cumplimiento de algunas de las otras líneas de fabricación de la empresa.

6. Conclusiones y perspectivas

En este trabajo se consideró el tema de la programación de operaciones en una planta industrial de la vida real. Se definió el problema como un modelo biobjetivo de un FFS de dos etapas. Los objetivos considerados fueron la minimización del lapso de fabricación o *makespan* y la minimización del número de trabajos tardíos. Con el objetivo de resolver este problema, se diseñó e implementó un algoritmo evolutivo con el cual se realizaron experimentos computacionales y se obtuvieron resultados. Estos resultados nos motivaron a extender el grupo de experimentos y llevarlo a una etapa de implementación comparativa con los datos históricos que para los mismos objetivos tenía la compañía.

Los parámetros del algoritmo evolutivo deben definirse muy bien para poder obtener un desempeño óptimo, en términos de calidad y rapidez de la solución final. De esta forma, como se muestra a lo largo de este trabajo, una idea bien definida sobre los parámetros adecuados para el algoritmo sólo se obtiene a través de la prueba de varios escenarios que se establezcan según el interés del analista que está interactuando con el algoritmo propuesto.

En términos de perspectivas de trabajo, la primera extensión natural del trabajo propuesto es la consideración de otras restricciones en el modelo de taller. Algunas de estas pueden ser la inclusión de restricciones de espacio de almacenamiento limitado para el producto en proceso o la consideración de más de dos estaciones en la planta.

Desde el punto de vista de la calidad del servicio al cliente, es bien conocido en la literatura que el objetivo de minimizar el número de trabajos tardíos puede inducir a tener entregas atrasadas con una gran demora respecto a la fecha de entrega. Una consideración interesante para el método propuesto podría ser considerar una cota superior para dicho retraso. Esto permitiría minimizar las entregas atrasadas, al mismo tiempo que les garantizaría a los clientes una fecha determinada para el peor de los casos posibles.

Agradecimientos

Los autores agradecen el apoyo y colaboración de los miembros del Departamento de Producción de la línea de marquillas por transferencia térmica de la empresa con la cual se realizó este trabajo. Así mismo, el apoyo de la Escuela Internacional de Ciencias Económicas y Administrativas de la Universidad de La Sabana.

Referencias

ACERO, M. J. R.; MONTOYA-TORRES, C. D. and PATERNINA-ARBOLEDA C. Scheduling jobs on a *k*–stage flexible flow shop. En OULAMARA, A. and PORTMAN, M. C. (eds.). *Proceedings of the Ninth International Workshop on Project Management and Scheduling (PMS*2004*)*, April 26-28. Nancy, France: EURO Working Group on Project Management and Scheduling, 2004, pp. 242-245.

ALLAOUI, H. and ARTIBA, A. Scheduling two-stage hybrid flow shop with availability constraints. *Computer & Operation Research.* 2006, núm. 33, pp. 1399-1419.

AZIZOGLU, M.; CAKMAK, E. and KONDAKCI, S. A flexible flowshop problem with total flow time minimization. *European Journal of Operational Research.* 2001, núm. 132, pp. 528-538.

BERTEL, S. and BILLAUT, J. C. A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research.* 2004, núm 159, pp. 651-662.

BOTTA-GENOULAZ, V. Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economic.* 2000, núm. 64, pp. 101-111.

BRAH, S. A. and HUNSUCKER, J. L. Branch and Bound algorithm for the flow shop with multiple processors. *European Journal of Operation Research.* 1991, núm. 51, pp. 88-99.

CHEN, B. Scheduling multiprocessor flowshops. En *Advances in Optimization and Approximation*. Kluwer: Dordrecht, 1994.

—. Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *Journal of the Operational Research Society*. 1995, núm. 46, pp. 234-244.

COLLETE, Y. and SIARRY, P. *Multiobjective optimization: principles and case studies*. New York: Springer Verlag, 2004.

DESSOUKY, M.; DESSOUKY, M. and VERMA, S. Flowshop scheduling with identical jobs and uniform parallel machines. *European Journal of Operational Research.* 1998, núm. 109, pp. 620-631.

DJELLAB, H. and DJELLAB, K. Preemptive hybrid flowshop scheduling problem of interval orders. *European Journal of Operational Research.* 2002, núm. 137, pp. 37-49.

EHRGOTT, M. and GANDIBLEUX, X. *Multiple criteria optimization: state of the art annotated bibliographic surveys.* Springer: Verlarg, 2002.

GUINET, A. and SOLOMON, M. Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. *International Journal of Production Research*. 1996, núm. 34, pp. 1643-1654.

GUPTA, J. Two stage hybrid flow shop scheduling problem. *Journal of the Operations Research Society*. 1988, núm. 38, pp. 359-364.

HAOUARI, M. and M'HALLAH, R. Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters*. 1997, núm. 21, pp. 43-53.

JIN, Z.; YANG, Z. and ITO, T. Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics.* 2006, núm. 100, pp. 322-334.

LEE, C. Y. and VAIRAKTARAKIS, G. Minimizing makespan in hybrid flowshops. *Operations Research Letter.* 1994, núm. 16, pp. 149-158.

LINN, R. and ZHANG, W. Hybrid flow shop scheduling: A survey. *Computers and Industrial Engineering.* 1999, núm. 37, pp. 57-61.

MORITA, H. and SHIO, N. Hybrid branch and bound method with genetic algorithm for flexible flowshop scheduling problem. *JSME International Journal Series C.* 2005, vol. 48, pp. 46-52.

MOURSLI, O. and POCHET, Y. A branch and bound algorithm for the hybrid flow shop. *International Journal of Production Economics*. 2000, núm. 64, pp. 113-125.

PINEDO, M. *Scheduling: theory, algorithms, and systems*. New Jersey: Prentice-Hall-Englewood Cliffs, 2002.

PORTMANN, M. C.; VIGNIER, A.; DARDILHAC, D. and DEZALAY, D. Branch and bound crossed with GA to solve hybrid flowshops. *European Journal of Operational Research.* 1998, núm. 107, pp. 389-400.

RIANE, F.; ARTIBA, A. and ELMAGHRABY, S. E. A hybrid three-stage flexible flowshop problem: Efficient heuristics to minimize makespan. *European Journal of Operational Research.* 1998, núm. 109, pp. 321-329.

RUIZ, R. and MAROTO, C. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research.* 2006, núm. 169, pp. 781-800.

SAWIK, T. An exact approach for batch scheduling in flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling.* 2002, vol. 36, núms. 4-5, pp. 461-471.

SOEWANDI, H. and ELMAGHRABY, S. E. Sequencing three-stage flexible flowshops with identical machines to minimize makespan. *IIE Transactions.* 2001, núm. 33, pp. 985-993.

SRISKANDARAJAH, C. and WAGNEUR, E. Hierarchical control of the two processor flowshop with state dependent processing times: Complexity analysis and approximate algorithms. *INFOR, Canadian Journal of Information Systems and Operational Research.* 1991, núm. 29, pp. 193-205.

TANG, L.; XUAN, H. and LIU, J. A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time. *Computers & Operations Research.* 2006, núm. 33, pp. 3344-3359.

T'KINDT, V. and BILLAUT, J. C. *Multicriteria Scheduling: Theory, models and algorithms*. 2a. ed. Springer: Verlag, 2006.

VARGAS-NIETO, F. *Diseño e implementación de un modelo de programación de operaciones biobjetivo basado en algoritmos evolutivos aplicado a la industria de marquillas estampadas por transferencia térmica*. Trabajo de grado de Maestría en Ingeniería Industrial. Barranquilla: Universidad del Norte, 2007.

VARGAS-NIETO, F. and MONTOYA-TORRES, J. R. Design and implementation of a bi-objective EA-based scheduling strategy: Case study in thermal-printed label manufacturing. *Proceedings of the International Industrial Simulation Conference* (*ISC-2007*). Delft: The Netherlands, 2007a, pp. 219-222.

—. Scheduling a thermal-printed label manufacturing plant using an evolutionary algorithm. *Proceedings of the 19th International Conference on Production Research (ICPR-19).* [CD-ROM]. Valparaiso: Chile, 2007b.

VIGNIER, A.; COMMANDEUR, P. and PROUST, C. New lower bound for the hybrid flowshop scheduling problem. *Emerging Technologies and Factory Automation Proceedings, 1997. ETFA '97, 1997 6th International Conference on*, 9-12 de septiembre de 1997, Los Angeles, pp. 446-451.

XUAN, H. and TANG, L. Scheduling a hybrid flowshop with batch production at the last stage. *Computers & Operations Research*. 2007, núm. 34, pp. 2718-2733.