Información Tecnológica
Vol. 26(3), 157-172 (2015)
doi: 10.4067/S0718-07642015000300019

Reducción del Tiempo de Terminación en la Programación de la Producción de una Línea de Flujo Híbrida Flexible (HFS)

Juan C. Lopez, Jaime A. Giraldo y Jaime A. Arango
Universidad Nacional de Colombia, Facultad de Ingeniería y Arquitectura, Departamento de Ingeniería Industrial, Manizales-Colombia (e-mail: jclopezva@unal.edu.co, jaiagiraldog@unal.edu.co, jaarangom@unal.edu.co)

Recibido Oct. 24, 2014; Aceptado Dic. 11, 2014; Versión final recibida Dic. 22, 2014

Resumen

Se propone un modelo de programación de la producción mediante la aplicación de una meta heurística para lograr la reducción del tiempo de terminación del último trabajo (*makespan*). Se eligió como objeto de estudio una empresa del sector textil y a partir de la codificación de un algoritmo genético simple se desarrolló una metodología de programación de la producción para configuraciones tipo línea de flujo (*flow shop*) híbrida flexible. El algoritmo arroja resultados de buena calidad con tiempos computacionales bastante razonables y con coeficientes de variación del orden de 2%, en relación a los valores del makespan. Este modelo también muestra un nivel de eficiencia comparable a otras aplicaciones encontradas en la literatura, en las cuales se utilizan algoritmos evolutivos. Se concluye que a partir de la codificación del sistema productivo y sus principales restricciones, el algoritmo genético ejecuta bien la programación de producción reduciendo el makespan.

Palabras clave: programación de la producción, makespan, flow shop hibrido flexible, algoritmos genéticos

Reduction of the Makespan in Scheduling the Production of a Hybrid-Flexible Flow Shop (HFS)

Abstract

A production programming model is proposed through the application of a metaheuristic technique to achieve the reduction of the makespan. A company in the textile sector was chosen as a case study and a production programming methodology for hybrid-flexible flow shop configurations was developed with the coding of a simple genetic algorithm. The algorithm gives good quality results with very reasonable computational times and coefficients of variation of about 2% in relation to the values of the makespan. This model also shows a level of efficiency comparable to other applications found in the literature, in which evolutionary algorithms are used. It is concluded that from the encoding of the production system and its main restrictions, the genetic algorithm well performs production scheduling by reducing the makespan.

Keywords: production programming, makespan, flexible hybrid flow shop, genetic algorithms

INTRODUCCIÓN

En el marco de la dirección de producción y operaciones, la programación de producción es una etapa clave, ya que en el corto plazo se toman las decisiones de carácter operativo que buscan el cumplimiento de las metas propuestas mediante la asignación de los recursos disponibles a las tareas de producción requeridas. La programación de producción o *scheduling*, se puede describir como la asignación de un conjunto de recursos en un período de tiempo, para llevar a cabo un conjunto de tareas (Phanden et al., 2012). Los objetivos considerados por los programadores de producción pueden clasificarse en tres grupos (Naderi et al., 2010): i) utilización eficiente de los recursos: entre ellos, el tiempo máximo de terminación de un conjunto de trabajos o *makespan*, ii) respuesta rápida a las demandas, o minimización del trabajo en proceso (WIP), y iii) conformidad con los plazos establecidos: tardanza total y el número de trabajos retrasados.

En un sistema de producción, la programación dependerá del entorno de la planta productiva (Phanden et al., 2012), y con el fin de que el programa propuesto sea exitoso es vital que quien lo desarrolle conozca a nivel de detalle la configuración productiva (Castrillón et al., 2010). Específicamente, la configuración *flow shop* es considerada un caso especial del *job shop*, en el que el problema de programación radica en secuenciar una cantidad de trabajos, que siguen la misma ruta de procesamiento, en una serie de máquinas ordenadas linealmente (Pinedo, 2005; Akhshabi et al., 2012; Shabtay, 2012). El *flow shop* híbrido flexible (HFS) es un caso especial del *flow shop* que posee un conjunto de n trabajos a ser procesados en una serie de m etapas (Yao et al., 2012) y en el cual al menos una de las etapas tiene más de una máquina en paralelo y se busca optimizar la programación en términos de cierta función objetivo (Wang y Liu, 2013). El problema consiste en resolver la distribución de las máquinas paralelas y la secuencia de los trabajos programados en la misma máquina con el fin de minimizar el máximo tiempo de flujo o *makespan* (Qiao y Sun, 2011). El enfoque en este tipo de configuración ha sido ampliamente estudiado, ya que es una de las configuraciones más comunes en los ambientes reales de fabricación (Hojjati y Sahraeyan, 2009; Yalaoui et al., 2011; Yang, 2011; Yaurima et al., 2009). En la programación clásica se utiliza un conjunto sumamente restrictivo de suposiciones tales como: tiempos de procesamiento determinísticos, la imposibilidad de dividir trabajos y la disponibilidad de las máquinas en todo momento (Schroeder, 2004). Sin embargo, estas suposiciones resultan no ser válidas para el modelo de muchos procesos industriales de la actualidad (Sun et al., 2012).

Los sistemas de producción que poseen máquinas paralelas pueden ser clasificados en tres grupos, en función de los tiempos de procesamiento de los trabajos en las máquinas (Cevikcan et al., 2011): i) máquinas paralelas idénticas, si los tiempos de procesamiento son los mismos para cada máquina, ii) máquinas paralelas uniformes, si las máquinas tienen una relación paramétrica en términos del tiempo de procesamiento, o iii) máquinas paralelas no relacionadas, si las diferencias entre los tiempos de procesamiento en las máquinas no pueden ser expresadas en una relación paramétrica. Dos enfoques distintos buscan explicar el concepto de flexibilidad en los entornos productivos, que derivan en el *flow shop* híbrido flexible. El primero, proviene de la capacidad del sistema para que los trabajos puedan saltar una o más etapas, es decir, se trata de trabajos que no necesitan ser procesados en todas las etapas del proceso (Zandieh y Karimi, 2010), el cual corresponde a la industria de tejidos técnicos. También, la flexibilidad del sistema radica tanto en la flexibilidad en el procesamiento como en la flexibilidad del enrutamiento, es decir, el procesamiento de un trabajo puede dividirse entre dos etapas, siempre con un enrutamiento unidireccional (Castrillón et al., 2010).

Existen además, restricciones al interior del sistema que definen el problema y éstos deben ser considerados por igual, como la satisfacción de la demanda, la capacidad limitada de los recursos y los tiempos de alistamiento de productos (Toledo et al., 2013). En algunos casos, el tiempo de preparación de un trabajo puede depender del trabajo inmediatamente anterior (Ángel-Bello et al., 2011). La magnitud del alistamiento de un trabajo que depende de aquel inmediatamente anterior en una misma máquina, se conoce como tiempos de alistamiento dependientes de la secuencia (Jabbarizadeh et al., 2009). Así, los problemas de programación de máquinas paralelas con tiempos de alistamiento dependientes de la secuencia y restricciones de fechas de los trabajos, son problemas complejos que requieren estudios adicionales en la práctica (Lin et al., 2011). La situación de tener la capacidad compartida entre productos fabricados para inventario (MTS) y productos fabricados bajo pedido (MTO), permite maximizar la capacidad al mantener una producción nivelada, pero agrega una alta variabilidad en los tiempos de entrega de productos MTO y en los niveles de inventario de productos MTS (Escobar et al., 2012), poniendo en riesgo el nivel de servicio ofrecido en ambos mercados, y, por consiguiente, la competitividad de la empresa, lo que hace justificable la búsqueda sistemática de una metodología de programación que tenga en cuenta ambos entornos en el contexto particular de la empresa seleccionada.

El problema clásico del *flow shop* hibrido flexible (HFS, por sus siglas en inglés) supone un conjunto de n trabajos que deben ser procesados en m etapas (Jabbarizadeh et al., 2009). Cada etapa contiene varias máquinas paralelas idénticas y la ruta de proceso es similar para cada trabajo; teniendo en cuenta un conjunto de suposiciones estándar, el objetivo es encontrar la programación que logre el mínimo *makespan* (Amin-Naseri y Beheshti-Nia, 2009; Baker y Altheimer, 2012). El tiempo que transcurre entre el inicio del primer trabajo en la primera máquina y la terminación del último trabajo en la última máquina se conoce con el término de *makespan* (Hojjati y Sahraeyan, 2009; Hmida et al., 2011). El *makespan* o $Cmax$ (como también se conoce en la literatura) es uno de los criterios de optimización más estudiado en los trabajos publicados, a través de la minimización del tiempo máximo de terminación de la programación (Vallada y Ruiz, 2011). Se considera que minimizar el *makespan* equivale a maximizar la utilización de las máquinas, ya que se busca disminuir los tiempos de alistamiento y los tiempos de ocio de las máquinas (Hekmatfar et al., 2011). El *makespan* es un criterio de decisión que se aplica de manera muy común en los problemas de programación (Jing et al., 2011). Por ejemplo, algunos trabajos que aplican este criterio son: Rebaine (2010) construye un algoritmo que minimiza el *makespan* en un *flow shop* sencillo; Chang y Chen (2011) analizan el problema de programación de máquinas paralelas no relacionadas con el objetivo de minimizar el *makespan*; Yang (2011) propone una heurística que busca el menor tiempo total de terminación; mientras que Zhao y Tang (2012) consideran la programación de un *flow shop* de dos estaciones para minimizar el *makespan* teniendo en cuenta restricciones de precedencia en el proceso.

El propósito de la programación de producción en un *flow shop* hibrido es obtener la asignación de los trabajos i, en las máquinas k, en los procesos j, en una secuencia de procesamiento l, de tal manera que logre reducirse el tiempo máximo de procesamiento o *makespan* ($Cmax$). En la Tabla 1 se muestra de manera resumida el modelo matemático del problema a resolver, en términos de la función objetivo y restricciones a satisfacer.

Tabla 1: Función objetivo y restricciones del problema a resolver.

Función a optimizar/Restricción	Descripción
$Min\ (C_{Max}) = Max(TF_j)$	Función objetivo. Se define que el $Cmax$ o *makespan* será el máximo tiempo de terminación de todos los trabajos en la última etapa j. Es decir, se garantiza que la función objetivo estará relacionada con el tiempo de terminación de todos los trabajos en la última etapa de procesamiento j.
$\sum_{k=1}^{n} X_{ikj} = 1, \quad X_{ikj} \in [0,1]$	La variable X_{ikj}, determina que cada trabajo i será asignado sólo una vez a la máquina k en cada etapa j.
$\sum_{i=1}^{n} X_{ikj} = 1, \quad X_{ikj} \in [0,1]$	La condición de que toda máquina k de la etapa j puede procesar un trabajo i como máximo.
$TF_{ikj} = TI_{ikj} + TA_{ikj} + (TP_{ikj} * E_k)$	Con máquinas paralelas no relacionadas y tiempos de alistamiento dependientes de la secuencia, cada trabajo i tendrá un tiempo de procesamiento y de alistamiento en cada máquina k de la etapa j, representado por TF_{ikj} en el cual se incluye: TA_{ikj} como el tiempo de montaje del trabajo i en la máquina k del proceso j, y TP_{ikj} es el tiempo de procesamiento del trabajo i, en la máquina k del proceso j. Además TP_{ikj} es afectado por el factor de eficiencia E_k que presenta la máquina k, causado por los problemas de calidad, errores humanos y condiciones de la máquina.
$TI_{ikj} \geq TF_{ikjo} + TA_{ikj}, \quad j_o, j \in RP_i$	La operación subsiguiente j de un trabajo i no puede iniciar antes de terminar la operación anterior j_o. Siempre que j y j_o estén definidas dentro de la ruta de procesamiento, RP_i, del trabajo i.
$E_k = U_k - M_k, \quad U_k, M_k \in [0,1]$	U_k define el porcentaje de utilización de la máquina k, y M_k equivale al factor de mantenimiento que reducirá el nivel de eficiencia de la máquina k.
$\sum_{i=1}^{n} TP_{ikj} \leq CP_k$	Los tiempos de procesamiento de los trabajos asignados a la máquina k (TP_{ikj}), no podrán superar el tiempo máximo que dispone su capacidad de procesamiento CP_k.

Reducir los tiempos de terminación es un método efectivo para eliminar demoras y tardanzas en los trabajos. Disminuir el *makespan* lleva también a la reducción del inventario en proceso (WIP), y minimiza los desórdenes de la planta por los trabajos no completados; por lo que la minimización de los tiempos de culminación es uno de los criterios más importantes para las empresas (Tavakkoli-Moghaddam et al., 2009). Una heurística clásica para el problema de programación de la producción con base en el *makespan* es la conocida regla de Johnson formulada en 1954 (Baker y Altheimer, 2012). Esta herramienta ha sido adaptada por muchos autores para resolver el problema del *flow shop* de 2 etapas con un tiempo de programación muy razonable (Carpov et al., 2012). Sin embargo, cuando el proceso productivo presenta más de dos etapas (m>2) y los trabajos a realizar poseen múltiples características, el problema pasa a ser NP-Completo, debido al crecimiento exponencial de la cantidad de soluciones alternativas que se crean para estos casos (Yao et al., 2012; Kasperski et al., 2012; Zhang y van de Velde, 2012).

Adicional a lo anterior, se considera que el *flow shop* está sujeto a una serie de características y restricciones de procesamiento que condicionan su programación (Pinedo, 2005). Inclusive, se ha demostrado desde la literatura clásica que, aunque los problemas con complejidad NP-Completo, se restrinjan desde sus variables de entrada, éstos mantendrán su nivel de complejidad NP-Completo para su solución (Garey et al., 1976). Esto ha causado la poca aplicación práctica de las heurísticas que consideran las reglas óptimas de secuenciación a pesar de su gran interés teórico. Ello se debe a que los verdaderos problemas de secuenciación implican una gran variabilidad en los tiempos de procesamiento (Schroeder, 2004). Ante la necesidad de encontrar otras alternativas distintas a las tradicionales heurísticas (que en su mayoría no reconocen la complejidad de los sistemas reales de fabricación), han surgido nuevas herramientas que buscan dar solución a la problemática de la programación de la producción en entornos como el *flow shop*. Entre estas meta heurísticas se reconocen los sistemas expertos, los agentes inteligentes, los algoritmos aleatorios y los algoritmos genéticos, entre otros (Castrillón et al., 2010).

En el campo de la programación de producción y operaciones, amplios estudios y aplicaciones se han propuesto con el uso de la meta heurística de los algoritmos genéticos. Anteriormente, se usaron algoritmos evolutivos como acercamiento a la programación de producción en la industria textil (Shiroma y Niemeyer, 1998); mientras que otra aplicación, trabajó con un algoritmo genético para la programación de un *flow shop*, teniendo como criterio la minimización de la tardanza (Onwubolu y Mutingi, 1999). Más adelante, un algoritmo genético se utilizó para reducir el *makespan* en la programación del *flow shop* híbrido, desarrollando una lista de programación de las tareas según el criterio de secuenciación FIFO (Primer trabajo en entrar, primero en ser atendido) (Xiao et al., 2000). Mesghouni y Rabenasolo (2002) usaron un algoritmo genético extendido aplicado al problema de programación bajo un modelo de producción con demanda incierta.

En el trabajo de Arango (Arango et al., 2013), se describe una propuesta de solución al problema de procesar n trabajos en m máquinas paralelas no relacionadas considerando solo la etapa de tejeduría. Por otra parte, en una fábrica con tiempos de alistamiento dependientes de la secuencia, se construyeron algoritmos genéticos con distintas soluciones iniciales y métodos de mutación para el problema de programación (Chang et al., 2003). De manera similar, Serifoglu y Ulusoy (2004) hacen un acercamiento con algoritmos genéticos para un *flow shop* híbrido multi-etapa, esta vez con reglas de prioridad SPT (Tiempo de procesamiento más corto), LPT (tiempo de procesamiento más largo) y STPT (tiempo de procesamiento total más corto). Igualmente, se publica una investigación en la que se aplican los algoritmos genéticos en un sistema multi-agente de planificación de los procesos en una industria textil (Solari y Ocampo, 2006). Mahdavi construye un algoritmo genético para el *scheduling* del *flow shop* híbrido con resultados eficientes para cantidades pequeñas de trabajos (Mahdavi et al., 2008). Sin embargo, a pesar de la notable cantidad de trabajos publicados en los últimos tiempos relacionados con el problema de la programación del *flow shop* híbrido (HFS), han sido pocos los que se han centrado en aplicar estas metodologías a los casos reales de la producción (Ribas et al., 2010).

Se concluye parcialmente que los algoritmos genéticos se han desarrollado con el fin de aplicarse en varias áreas de interés, entre las cuales, se incluyen los problemas de programación y asignación de máquinas. En la revisión de la literatura se ha seleccionado sólo unas pocas aplicaciones (por cuestiones de espacio) en entornos reales de manufactura, ya que hay una inmensa cantidad de casos donde se aplica esta herramienta exitosamente. Esta técnica ofrece la oportunidad de modelar gran cantidad de restricciones presentes en la realidad para llegar a resultados sumamente interesantes. Los autores citados logran demostrar la eficiencia y el buen rendimiento de los algoritmos genéticos para obtener alternativas de solución de muy buena calidad y aplicables a las decisiones de corto plazo (programación de la producción), con un costo de tiempo y recursos computacionales relativamente bajos.

ALGORITMO USADO

Como alternativa al caso propuesto, se aplicó un algoritmo genético simple o estándar (AGS). Los algoritmos genéticos (AG), desarrollados en la década de los 70 por Holland (Holland, 1992; Gómez-Gasquet, 2007; Mahdavi et al., 2008; Serifoglu y Ulusoy, 2004), son técnicas de búsqueda heurística que toman la analogía de los conceptos de la selección natural, empleando una población de soluciones candidatas y combinándolas en formas específicas con el fin de obtener mejores soluciones (Feng et al., 2009). En el algoritmo genético, la representación de una solución factible se conoce con el término de cromosoma, que puede ser construido, entre otras maneras, a través de una cadena de valores enteros, cada uno de los cuales se denomina gen (Tang et al., 2010). La calidad de un cromosoma se denomina *fitness*, que establece la concordancia de la alternativa de solución de acuerdo a la función objetivo definida (Chiou et al., 2012). Las combinaciones de los genes van evolucionando a través de las operaciones genéticas (selección, cruce y mutación) para crear la descendencia, de manera que los cromosomas se van aproximando a la solución óptima generación tras generación (Engin et al., 2011). Sin embargo, es necesaria una codificación adecuada para cada problema y tener una función de ajuste que represente claramente la medida de calidad para cada solución alternativa (Hekmatfar et al., 2011).

El procedimiento que se sigue al emplear un algoritmo genético consta fundamentalmente de cinco etapas: i) generar la población inicial: se establece el método de selección de los mejores individuos, ya sea de manera determinística o aleatoria (Jabbarizadeh et al., 2009); ii) selección: mediante esta los cromosomas compiten entre sí para ser seleccionados como padres y llevar a cabo la operación genética, con la finalidad de generar la descendencia (Toledo et al., 2013; Safari y Sadjadi, 2011; Toro et al., 2005), siendo cada solución candidata evaluada según su valor de *fitness*, que indica la calidad de la solución representada, y aquellas que califiquen mejor tendrán mayor probabilidad de sobrevivir (Yaurima et al., 2009; Solari y Ocampo, 2006); iii) operación genética: el propósito de esta etapa es crear los individuos de la siguiente población a través de los operadores genéticos de cruce y de mutación, que también son de naturaleza probabilística (Tavakkoli-Moghaddam et al., 2009; Gallego et al., 2006), siendo el operador de cruce el más utilizado y cuya función es crear nuevos individuos a partir de dos cromosomas seleccionados con el propósito de obtener la descendencia; estos nuevos individuos se generan a través de la combinación de las características de los cromosomas padres (Zandieh y Karimi, 2010; Zhan y Qiu, 2008). Otra función importante de este operador es la de encontrar una región entre los candidatos que tenga una alta probabilidad de contener buenas soluciones aptas para la nueva generación (Su y Li, 2009). De otra parte, el operador de mutación genera una solución a partir de la modificación aleatoria de las características de un cromosoma padre (Gómez-Gasquet et al., 2012; Rajkumar y Shahabudeen, 2009).

La mayoría de algoritmos genéticos incorporan un operador de mutación con el fin de preservar un nivel razonable de diversidad poblacional, para proveer un mecanismo de escape a un óptimo local, y recuperar material genético perdido (Ruiz y Maroto, 2006); iv) actualización de la población: después de aplicar los operadores genéticos a los elementos de la antigua generación, se crea una nueva población que estará conformada por aquellos cromosomas sobrevivientes de la antigua generación con mayores valores de *fitness*, en lo que se conoce como estrategia elitista (Gendreau y Potvin, 2010); y v) cierre: pueden utilizarse ciertas pautas para determinar cuándo detener la búsqueda. El proceso iterativo del algoritmo finaliza si se cumple una determinada condición: que se haya procesado una cierta cantidad de generaciones, que la aptitud de una candidata supere cierto valor definido, o que el valor de la función objetivo no mejore en cierto número de generaciones, en caso contrario, se continúa con el proceso de selección (Solari y Ocampo, 2006). Procurando tener no sólo una visión resumida sino también más clara de la metodología diseñada para programar la producción a partir de la ejecución de un algoritmo genético, la Figura 1 muestra el seudocódigo para el algoritmo genético propuesto.

En vista de que el algoritmo buscará la alternativa de solución que presente el menor *makespan*, es importante determinar la estructura de los cromosomas que conformarán la población y los cuales se modificarán durante las iteraciones del algoritmo genético. Se define que el cromosoma tendrá un número de celdas o posiciones igual al producto del número de trabajos i por el número de máquinas k de tal manera que los valores de la cadena logren representar la asignación de los trabajos en las máquinas del sistema, teniéndose así una estructura del cromosoma de $i * k$ posiciones. De acuerdo a la codificación del algoritmo planteado, la construcción del cromosoma se hace de una manera secuencial, es decir, el cromosoma se construye desde la asignación del primer trabajo en la primera etapa del proceso, hasta la asignación del último trabajo en la última etapa del proceso productivo.Teniendo en cuenta que existen restricciones técnicas que implican que no todos los trabajos pueden ser procesados en cualquier máquina, se crea una matriz binaria de unos y ceros que indique para cada trabajo si puede ser procesado o no en cada máquina, en todas las m etapas productivas. Así, con un valor de 1 (uno), un trabajo i puede procesarse en la máquina k, y con un valor de 0 (cero) el trabajo i no podrá ser procesado en la máquina k, según se muestra en la matriz de factibilidad mostrada en la Tabla 2.

```
A G S (Número de trabajos, Población, Tasa Mutación, Iteraciones, Puntos Cruce)
1   Generar aleatoriamente la población inicial
2   Aplicar las restricciones de producción a los individuos de la población
3   Secuenciación de los trabajos según la regla SPT
4   Evaluar el makespan
5   for i=1 hasta Iteraciones
6      Seleccionar los padres mediante la ruleta proporcional
7      Operación de cruce
8      if rand ≤ Tasa Mutación
9         Operación de Mutación
10     end-if
11     Candidato ← Mejor descendiente
12     if Candidato ≤ Peor individuo de la población
13        Población ← Candidato
14     end-if
15  end-for
16  Seleccionar el individuo con el menor makespan
17  Mostrar información de salida
end A G S
```

Fig. 1: Seudocódigo del algoritmo genético

Tabla 2: Matriz de factibilidad de procesamiento de los trabajos en cada máquina.

Etapas (m)	1			2		…		$m-1$						m
Máquinas (k) / Trabajos (i)	1	2	3	4	5	6	7	8	9	10	11	…	$k-1$	k
1	1	1	1	0	1	1	1	0	0	1	1	…	1	0
2	1	1	1	1	0	1	1	1	1	0	0	…	1	1
3	1	1	1	1	1	1	1	1	1	1	0	…	1	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
$i-1$	1	1	1	1	1	1	1	1	0	0	1	…	0	0
i	1	1	1	0	1	1	1	0	0	1	1	…	0	1

Para que se logre cumplir la asignación de cada trabajo a una sola máquina en cada etapa del proceso y se cumpla con las condiciones reales de la producción, se lleva un proceso de verificación del cromosoma completo. Este inicia desde las posiciones que representan la asignación del primer trabajo en las máquinas de la primera etapa, y termina en las posiciones que representan la asignación del último trabajo en las máquinas de la última etapa. Se selecciona aleatoriamente una de las posiciones que representa una máquina en la etapa respectiva y se cambia su valor por un 0 (cero). Estos cambios se realizan hasta que, la suma de los valores de las celdas correspondientes a las máquinas de cualquier etapa, sumen 1 (uno). El procedimiento se repite para las demás etapas del proceso y para todos los trabajos programados. Con este método se garantiza que cada trabajo será procesado en una sola máquina en cada etapa de la producción. El cromosoma es, por tanto, la representación de una alternativa de solución para la asignación de todos los trabajos en las máquinas de cada etapa, como se ilustra en la Figura 2. Como cada trabajo sólo se programa una vez en cada etapa, la suma de los X_{ik} debe ser igual a 1 (para cada trabajo i en cada máquina k). El cromosoma queda construido entonces cuando todos los trabajos hayan sido asignados a las máquinas en todas las etapas (López, 2013).

Trabajo	1											2										
Etapa	1			2		3		4			5	1			2		3		4			5
Máquina	1	2	3	4	5	6	7	8	9	...	k	1	2	3	4	5	6	7	8	9	...	k
Posición	1	2	3	4	5	6	7	8	9	..	k	$k+1$	$k+2$	$k+3$	$k+4$	$k+5$	$k+6$	$k+7$	$k+8$	$k+9$	..	$2k$
Asignación	0	0	1	0	1	0	1	1	0	..	1	1	0	0	1	0	1	0	0	1	...	1

Trabajo	3											...										
Etapa	1			2		3		4			5	1			2		3		4			5
Máquina	1	2	3	4	5	6	7	8	9	...	k	1	2	3	4	5	6	7	8	9	...	k
Posición	$2k+1$	$2k+2$	$2k+3$	$2k+4$	$2k+5$	$2k+6$	$2k+7$	$2k+8$	$2k+9$	..	$3k$	$3k+1$	$3k+2$	$3k+3$	$3k+4$	$3k+5$	$3k+6$	$3k+7$	$3k+8$	$3k+9$	..	$4k$
Asignación	0	0	1	0	1	0	1	0	1	..	1	0	1	0	0	1	0	1	1	0	...	1

Trabajo	$i-1$											i										
Etapa	1			2		3		4			5	1			2		3		4			5
Máquina	1	2	3	4	5	6	7	8	9	...	k	1	2	3	4	5	6	7	8	9	...	k
Posición	$(i-2)k+1$	$(i-2)k+2$	$(i-2)k+3$	$(i-2)k+4$	$(i-2)k+5$	$(i-2)k+6$	$(i-2)k+7$	$(i-2)k+8$	$(i-2)k+9$	..	$(i-1)k$	$(i-1)k+1$	$(i-1)k+2$	$(i-1)k+3$	$(i-1)k+4$	$(i-1)k+5$	$(i-1)k+6$	$(i-1)k+7$	$(i-1)k+8$	$(i-1)k+9$	..	ik
Asignación	0	1	0	1	0	1	0	0	1	..	1	0	0	1	0	1	0	1	0	1	...	1

Fig. 2: Estructura básica del cromosoma diseñado en el algoritmo genético

Población inicial

A pesar del componente aleatorio en la asignación de los trabajos a las máquinas, las restricciones de secuenciación y de procesamiento deberán cumplirse, obligando a que los trabajos se asignen aleatoriamente a cualquier máquina en cada etapa del proceso, siempre y cuando cada máquina tenga la capacidad de procesar el trabajo asignado. Dado que la población es el conjunto de individuos o cromosomas del algoritmo, se trabajará entonces con una matriz de unos y ceros de tamaño $i * k$ columnas, y tamaño de la población cuantas filas se generen, es decir, cada fila resulta ser un cromosoma o individuo de la población del algoritmo genético. La construcción de la población inicial mediante la codificación planteada, se realiza repitiendo el procedimiento para la obtención de un cromosoma, como se definió en el apartado anterior, tantas veces como lo defina el tamaño de la población.

Selección

A partir de la asignación aleatoria de los trabajos a las máquinas, la selección en cada una de ellas se hará mediante la regla de prioridad SPT (tiempo más corto de procesamiento). Esta regla se aplicará de acuerdo a los tiempos de procesamiento de los trabajos en la etapa donde se presentan cuellos de botella y se llevará este orden de secuenciación desde la primera etapa del proceso productivo. Luego de calcular los tiempos de inicio y terminación de cada trabajo en cada máquina (considerando los factores de eficiencia, mantenimiento, velocidad y cantidades), partiendo de una asignación y secuenciación previa, se obtiene un tiempo máximo de terminación, en otras palabras, se obtiene el valor de *makespan* para esa distribución de los trabajos. Este cálculo se repite entonces para todos los individuos de la población obteniendo una lista de los valores de *makespan* para cada una de las soluciones alternativas. Partiendo de que la función objetivo del modelo viene representada por el valor de *makespan*, definida como $Cmax$, cada individuo tendrá por tanto una función de *fitness* (f) calculada mediante la siguiente expresión:

$$f = 1 \,/\, C\_max \tag{1}$$

De manera que aquellos individuos con menor valor de *makespan*, tienen mayor valor *fitness* y consecuentemente tendrán mayor probabilidad de supervivencia y de ser utilizados para llevar a cabo las operaciones genéticas. En lo que concierne al proceso de selección del algoritmo, se ha establecido usar la metodología de la ruleta proporcional, la cual es una herramienta de clasificación que otorga una probabilidad de selección basada en la frecuencia relativa de la función de *fitness* de cada individuo en relación a la sumatoria de todos los valores de la población. De esta forma, tendrán más posibilidades de selección los individuos con mayor valor de *fitness*, y consecuentemente, aquellos con valores más bajos tendrán menos probabilidad de ser seleccionados.

Operación genética de cruce y mutación

Con los cromosomas seleccionados para el proceso de cruce, se crean dos descendientes que inicialmente tendrán vacías sus posiciones. Los genes vacantes de estos nuevos cromosomas tendrán la información relacionada con la asignación de los trabajos a las máquinas proveniente de los padres seleccionados, cuando los genes se copien de manera permutada en función de la cantidad de puntos de cruce definidos en el algoritmo y cuyas posiciones se obtienen de manera aleatoria. La figura 3 muestra una representación gráfica de la operación de cruce de dos individuos para crear dos descendientes nuevos; la figura 3 ofrece

un ejemplo para dos cromosomas con 38 celdas o posiciones y tres puntos de cruce. Los dos individuos descendientes son creados de tal manera que, y de acuerdo a los puntos de cruce, no se asigne un trabajo a más de una máquina en una etapa específica, o que un trabajo no sea asignado a ninguna de las máquinas perteneciente a cualquier etapa del proceso.

El operador genético de mutación actuará sobre cada uno de los dos descendientes creados en el cruce de los cromosomas padres. Partiendo de una tasa de mutación (pm) definida al inicio del algoritmo, se compara frente un aleatorio generado entre 0 y 1 que, de resultar este valor menor a pm, la mutación se llevará a cabo sobre el primer descendiente. De manera aleatoria se selecciona una posición del cromosoma y teniendo en cuenta la máquina y la etapa dentro del proceso que represente dicha celda, se reasignará el trabajo a otra máquina de la etapa respectiva. Este procedimiento se repite para el segundo cromosoma descendiente. Con los dos descendientes constituidos mediante las operaciones genéticas, se procede a analizar las alternativas de solución respecto a las soluciones que contiene la población actual.

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Cromosoma 1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	1
Cromosoma 2	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Descendiente 1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1
Descendiente 2	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1

Fig. 3: Representación de la operación de cruce entre dos cromosomas

Actualización de la población

Para ello, deberá considerarse cuál de los dos candidatos presenta mejor rendimiento en la asignación y la secuenciación de los trabajos. Por tanto, se procede a calcular los tiempos de procesamiento y evaluar el valor de *makespan* para ambos cromosomas, con la finalidad de establecer el mejor candidato descendiente, descartando y eliminando el segundo. Paralelamente se selecciona el individuo de la población que contenga el peor valor *fitness*, en otras palabras, el cromosoma que presente una configuración con el mayor valor de *makespan* dentro del conjunto de soluciones de la población, será el candidato a ser reemplazado. El paso siguiente es comparar el mejor descendiente y el peor individuo de la población en términos del valor de *makespan*. Si el descendiente candidato presenta mejor rendimiento, entrará a ser parte de la nueva población en reemplazo del peor individuo detectado. Queda establecido entonces, que la codificación logra la actualización mediante el reemplazo de un sólo individuo de la población, por cada iteración que realice el algoritmo.

Cierre

Mediante el proceso de codificación se ha establecido como criterio de parada el número de iteraciones que deberá procesar el algoritmo y en los cuales las generaciones llevarán a mejores alternativas de solución para el problema. Por tanto, si inicialmente se define un número de iteraciones igual a 3000, el algoritmo repetirá el ciclo genético tal cantidad de veces para cumplir la condición de parada. Una vez sea verificado el criterio de parada, se procede a recolectar la información de salida del algoritmo. En primer lugar debe seleccionarse de la población final el individuo con el mayor valor de *fitness*, es decir, aquella solución que posea el menor *makespan*. De esta solución, se recogerá la información que muestre la asignación de los trabajos en cada máquina dispuesta en la planta y el cálculo del tiempo máximo de procesamiento. Se trata de presentarle al usuario la secuenciación de los trabajos asignados a cada máquina en cada etapa del proceso productivo, al tiempo que se presenta el valor de la función objetivo de la mejor solución: el menor *makespan* encontrado.

RESULTADOS Y DISCUSIÓN

Con el objetivo de crear un modelo de programación de producción lo más cercano a los entornos reales de la industria textil, se decidió trabajar con una configuración productiva que contenga las 5 etapas típicas de la producción en la industria de tejidos técnicos: hilandería, urdizaje, remetido, tejeduría y acabados. En la Figura 4 se muestra un típico sistema productivo que contiene más de una máquina en al menos una de sus etapas y algunos trabajos se caracterizan por no requerir el proceso de alguna de las etapas productivas (López, 2013).

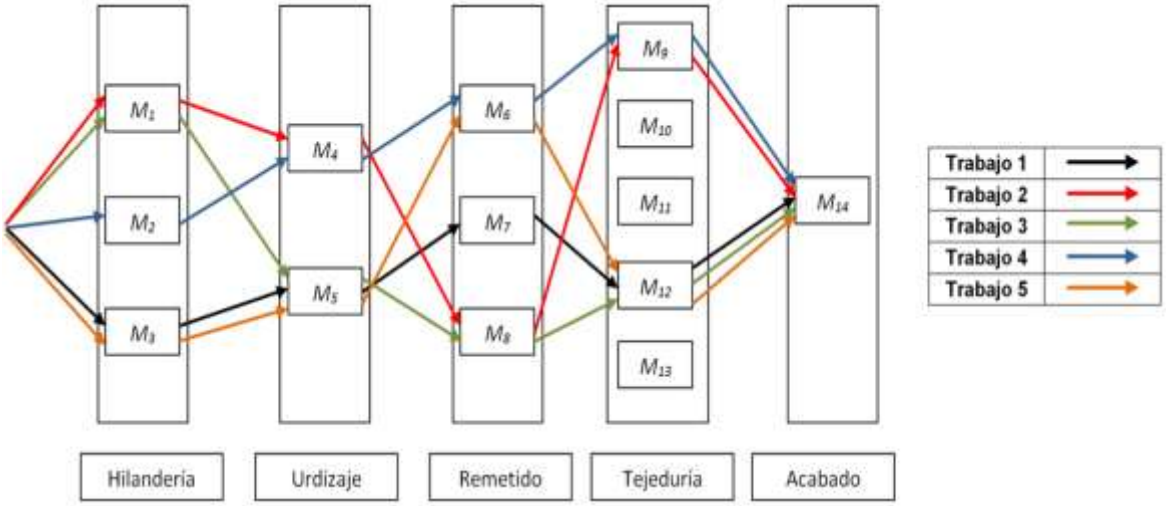

Fig. 4: Rutas de procesamiento de 5 trabajos asignados aleatoriamente a las máquinas a través de la codificación propuesta del cromosoma

Se contempló, para la ejecucion del algoritmo propuesto, un total de 19 máquinas en las etapas productivas distribuidas de la siguiente manera: hilandería (3), urdizaje (2), remetido (3), tejeduría (10) y acabados (1). La codificación del algoritmo mencionado (se puede acceder al algoritmo y a los datos de prueba en el sitio www. algoritmogeneticoenhfs.wordpress.com), se desarrolló mediante el *software* MATLAB en su versión R2012a, y se ejecutó en un computador personal con un procesador de 1,7 GHz y una capacidad de memoria RAM de 2 GB. Se creó una interfaz gráfica de usuario (ver Figura 5) que permite ingresar datos relacionados con: el número de trabajos o pedidos pendientes a procesar, el número de iteraciones del algoritmo, el tamaño de la población, la tasa de mutación y los puntos de cruce. Con el botón “CALCULAR”, se ejecuta el algoritmo y permite observar posteriormente el resultado del *makespan* obtenido.

En la experimentación requerida para validar el algoritmo propuesto, se ha considerado la evaluación con distintos valores de las variables de entrada. Se decidió utilizar dos valores para cada variable de entrada, según se muestra en la Tabla 3, lo que arroja experimentar con 32 combinaciones distintas (2^5). Para cada combinación de variables de entrada, se ejecutó el programa 30 veces con la finalidad de obtener variación estadística que dé validez a la hora de proponer elementos de análisis frente a los resultados obtenidos. Tres estadísticos se consideraron durante las pruebas: la media o el promedio del *makespan* obtenido con las 30 ejecuciones, la desviación estándar muestral de estos valores y el coeficiente de variación. Este último cálculo se efectúa para determinar la magnitud relativa de la desviación de los resultados, teniendo en cuenta que la dimensión de los valores de *makespan* está en miles de horas.

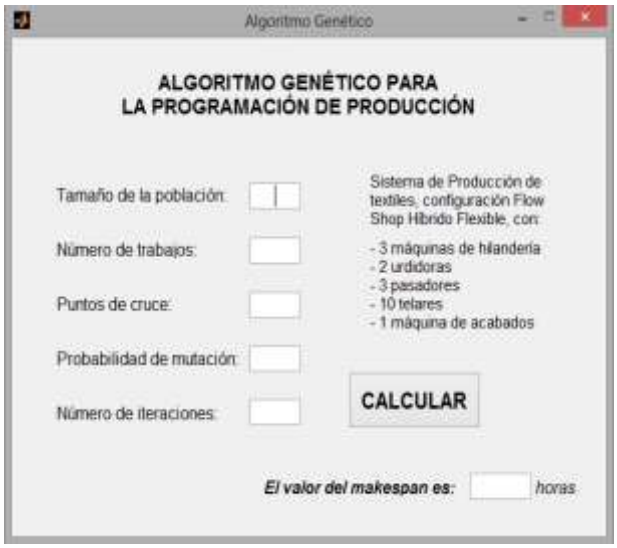

Fig. 5: Interfaz gráfica del algoritmo genético. Fuente:

Tabla 3: Variables de experimentación del algoritmo genético.

Variables de entrada	Valores
Tamaño de la población	50, 100
Número de trabajos	30, 50
Puntos de cruce	20, 30
Probabilidad de mutación	0.1, 0.3
Número de iteraciones	3000, 5000

La Tabla 4 muestra los resultados obtenidos al ejecutar el algoritmo para las combinaciones posibles cuando $i = 30$ trabajos por programar, con tiempo computacional promedio por ejecución de 101 segundos. Para cada combinación, se presenta la mejor solución obtenida cuya desviación de los datos tiende a disminuir cuando las variables número de iteraciones y tamaño de la población presentan valores elevados; consecuentemente con el coeficiente de variación se presenta el mismo fenómeno, cuyos valores no superan 2%, lo cual podría indicar para las configuraciones con valores más altos de las variables, un comportamiento nivelado en las soluciones arrojadas.

Tabla 4: Resultados de ejecución del algoritmo con $i = 30$ trabajos

	Variables de entrada				*Makespan* (Tiempo en horas)			
Combinación	Iteraciones	Población	T. mutación	P. cruce	Mejor	Media	Desv. Est.	Coef. Variación %
1	3000	50	0,1	20	1435,6	1506,09	51,51	3,4
2	3000	50	0,1	30	1454,5	1519,65	38,46	2,5
3	3000	50	0,3	20	1449,7	1499,20	33,00	2,2
4	3000	50	0,3	30	1430,5	1500,76	30,36	2,0
5	3000	100	0,1	20	1440,5	1487,61	31,24	2,1
6	3000	100	0,1	30	1439,5	1481,43	30,72	2,1
7	3000	100	0,3	20	1439,3	1475,67	26,92	1,8
8	3000	100	0,3	30	1440,5	1475,05	26,66	1,8
9	5000	50	0,1	20	1442,4	1523,62	39,90	2,6
10	5000	50	0,1	30	1440,5	1511,36	48,53	3,2
11	5000	50	0,3	20	1430,4	1496,87	50,82	3,4
12	5000	50	0,3	30	1444,4	1498,95	38,69	2,6
13	5000	100	0,1	20	1440,5	1476,80	36,46	2,5
14	5000	100	0,1	30	1440,5	1475,45	28,65	1,9
15	5000	100	0,3	20	1438,9	1463,66	15,67	1,1
16	5000	100	0,3	30	1439,3	1468,65	22,49	1,5

Con el propósito de presentar los resultados arrojados por el algoritmo en función de sus variables de una manera gráfica, se han construido a partir de la Tabla 4, las Figuras 6 y 7 donde se relacionan los promedios de los valores del *makespan* calculados para cada combinación de variables de entrada. La Figura 6 muestra las 8 combinaciones construidas a partir de establecer los parámetros de número de trabajos igual a 30 y de 20 puntos de cruce, mientras que la Figura 7, presenta la información con 30 trabajos y 30 puntos de cruce. La convergencia de los resultados de las ejecuciones hechas al algoritmo a partir de las instancias definidas en las variables de entrada, se muestra en la Tabla 5, donde se presenta el análisis de varianza ANOVA elaborado para el caso de estudio con $i = 30$ trabajos (Para ver los resultados del análisis de varianza con $i = 50$ trabajos, se recomienda al lector dirigirse al sitio web www. algoritmogeneticoenhfs.wordpress.com para mayor información). Al hacer un paralelo entre las Figuras 6 y 7, puede observarse un comportamiento bastante similar, a excepción de la primera configuración de valores (población=50 y mutación=0.1), que es precisamente la combinación más simple de valores y cuyo

comportamiento muestra mayor variabilidad en las soluciones obtenidas. Se evidencia una disminución del *makespan* en todos los casos cuando la tasa de mutación aumenta de 0.1 a 0.3, indicando esto que el algoritmo genético tiene mejor desempeño cuando la posibilidad de buscar alternativas diferentes aumenta y al tiempo decrece el riesgo de estancarse en un óptimo local. La influencia del tamaño de la población y el número de iteraciones en el procesamiento del algoritmo es bastante considerable. En la mayoría de los casos, la línea perteneciente a las 5000 iteraciones presentó mejores resultados que la línea de las 3000 iteraciones. Además, los mejores tiempos en promedio, se encontraron cuando el tamaño de la población se definió en 100 individuos, de tal manera que puede afirmarse nuevamente, que los mejores resultados se encuentran cuando las variables de entrada toman valores más altos.

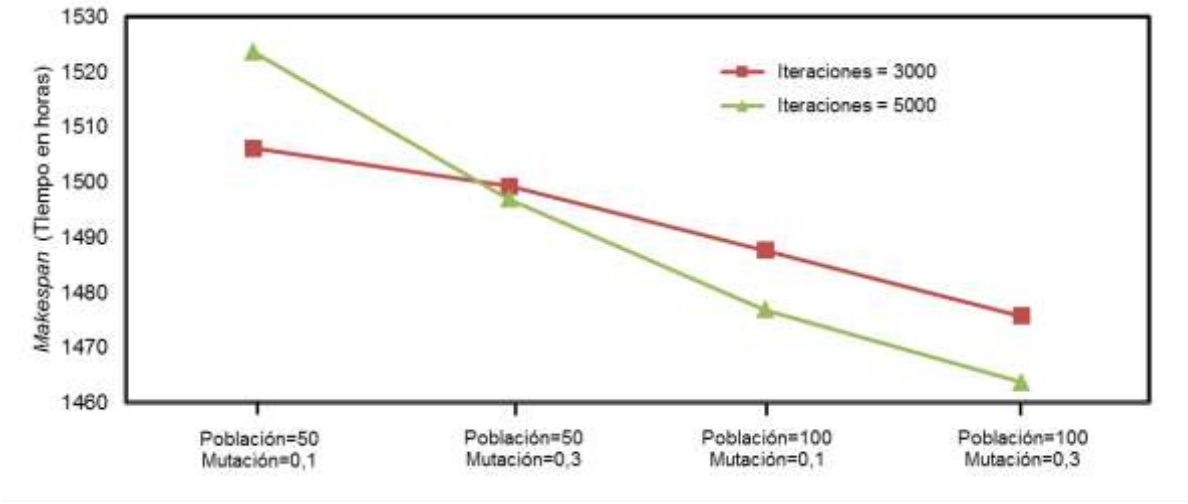

Fig. 6: *Makespan* promedio con 30 trabajos y 20 puntos de cruce

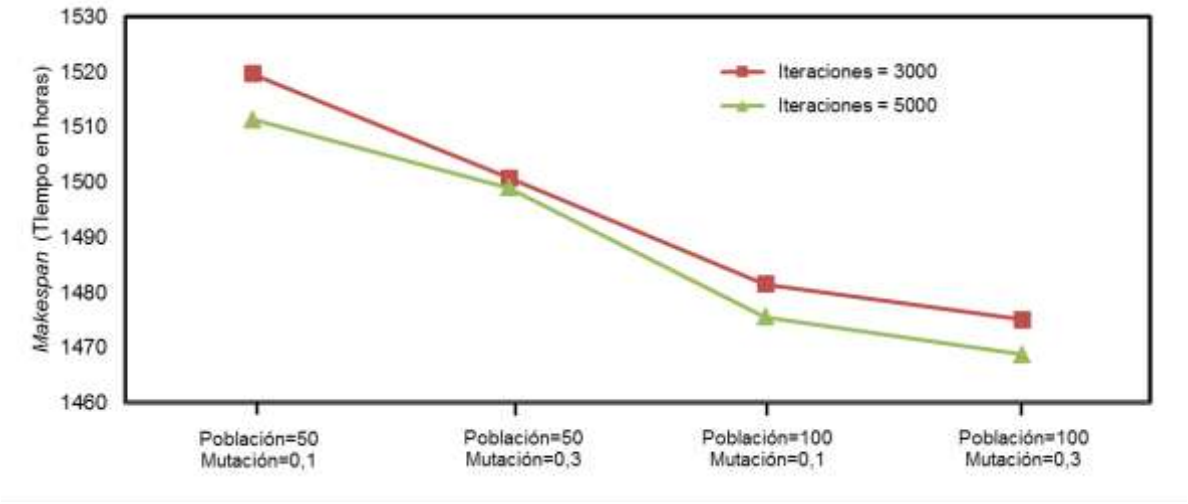

Fig. 7: *Makespan* promedio con 30 trabajos y 30 puntos de cruce

Tabla 5: Tabla ANOVA para las ejecuciones del algoritmo con 30 trabajos

Variable	*Suma cuadrados*	*GL*	*Cuadrados medios*	*Valor F*	*Probabildad*
Población	1697,264	1	1697,26	1,33593412	0,2483285
Iteraciones	119252,770	1	119253	93,8650892	2,171E-20
Mutación	19969,200	1	19969,2	15,7179639	8,474E-05
P. cruce	5,985	1	5,98533	0,00471112	0,9453067
Error	603473,200	475	1270,47		
Totales	744398,420	479			

Con fines de comparación, se construyó un algoritmo aleatorio capaz de llevar a cabo la programación de producción en el mismo entorno *flow shop* híbrido flexible, con igual cantidad de máquinas y etapas en el proceso, las mismas restricciones de procesamiento y las mismas consideraciones para los cálculos de los tiempos de producción. Este algoritmo asigna los trabajos requeridos a las máquinas de manera aleatoria y se calculan entonces los tiempos de inicio y terminación de cada trabajo en cada máquina obteniendo al final, el valor del *makespan* para la solución planteada. La Tabla 5 muestra los resultados obtenidos a partir de la ejecución del algoritmo aleatorio 30 veces. Claramente, tanto para 30 como para 50 trabajos, los datos presentan una desviación bastante significativa frente a los obtenidos mediante el algoritmo genético con coeficiente de variación cercanos al 20%.

Tabla 5: Resultados de la ejecución del algoritmo aleatorio

	Makespan (Tiempo en horas)			
Trabajos	Mejor	Media	Desv. Est.	Coef. variación %
30	2129,3	3706,87	838,94	22,6%
50	4532,8	5919,64	1152,34	19,5%

Los valores del *makespan*, aspecto considerado más crítico, resultaron ser demasiado elevados. En promedio, los resultados para 30 trabajos están por encima de las 3700 horas, mientras que para $i = 50$ trabajos, el promedio del *makespan* se acerca a las 5900 horas de producción; inclusive, las mejores soluciones encontradas por el algoritmo aleatorio siguen estando bastante por encima de las peores soluciones encontradas por el algoritmo genético tanto para 30 como para 50 trabajos a programar. Esta cuestión favorece ampliamente al algoritmo genético propuesto, que ofrece soluciones que rondan las 1440 horas para $i = 30$ trabajos, y las 2400 horas para $i = 50$ trabajos.

Finalmente, se invita al lector a revisar los siguientes trabajos sobre programación en un *flow shop* híbrido flexible reportados en la literatura consultada, con el fin que pueda valorar la efectividad del algoritmo que se reporta en el presente artículo: Costa et al. (2013), Jabbarizadeh et al. (2009) y Hsu et al. (2009).

CONCLUSIONES

De los resultados mostrados, su interpretación y discusión se pueden obtener las siguientes conclusiones:

El modelo matemático que representa al sistema productivo elegido como objeto de estudio, ha facilitado la codificación de un algoritmo genético, que consideró las complejidades de la producción que suceden en los entornos reales de la manufactura, y en específico, los factores que afectan la continuidad y la dinámica de los procesos dentro de una planta de producción textil.

Los cálculos que realiza el algoritmo logran considerar los principales aspectos que condicionan los sistemas reales de producción. Entre los que se destacan: los tiempos de alistamiento dependientes de la secuencia; tiempos de mantenimiento, nivel de utilización, velocidades y ritmos de las maquinas, la capacidad de procesamiento de cada máquina sobre algunos trabajos y las cantidades específicas de cada pedido a procesar. Estos elementos unidos a las demás restricciones planteadas en el modelo matemático conllevan a valores del *makespan* de buena calidad por parte del algoritmo genético propuesto.

El proceso de experimentación arrojó resultados muy aceptables donde se evidencia que a través del algoritmo de programación de producción, logra reducirse el *makespan* como función objetivo, con igual o mejor desempeño que otros algoritmos propuestos para situaciones similares. Aspecto de inmensa importancia teniendo en cuenta que la reducción de los tiempos de procesamiento no sólo implica beneficios a la planta en cuanto a productividad, eficiencia y costos, sino que a nivel organizacional significa mejores tiempos de entrega, mejor servicio al cliente y mayor competitividad.

AGRADECIMIENTOS

Los autores desean dar reconocimiento a la Universidad Nacional de Colombia por su apoyo al desarrollo de esta investigación (Convocatoria de Apoyo a Tesis de Posgrado-DIMA 2012. Proyecto: "Mejora de Tiempos de Entrega en un *Flow Shop* Híbrido Flexible Usando Técnicas Inteligentes. Aplicación en la Industria de Tejidos Técnicos", código Hermes 15917). Este trabajo hace parte de la tesis doctoral del coautor Jaime Antero Arango M. y la tesis de maestría del coautor Juan Camilo López V.

REFERENCIAS

Akhshabi, M., Haddadnia, J., y Akhshabi, M. *Solving flow shop scheduling problem using a parallel genetic algorithm*, Procedia Technology, 1, 351-355, (2012)

Amin-Naseri, M. R. y Beheshti-Nia, M. A. *Hybrid flow shop scheduling with parallel batching*, International Journal of Production Economics, 117, 185-196, (2009)

Ángel-Bello, F., Álvarez, A., Pacheco, J. y Martínez, I. *A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times*, Computers and Mathematics with Applications, 61, 797-808, (2011)

Arango, J. A., Giraldo, J. A., y Castrillón, O. D. *Programación de Máquinas Paralelas no Relacionadas con Tiempos de Montaje dependientes de la Secuencia y Entrada Dinámica usando Algoritmos Genéticos*. Información Tecnológica, 24, (3), 73-84, (2013)

Baker, K. R. y Altheimer, D. *Heuristic solution methods for the stochastic flow shop problem*, European Journal of Operational Research, 216, 172-177, (2012)

Carpov, S., Carlier, J., Nace, D. y Sirdey, R. *Two-stage hybrid flow shop with precedence constraints and parallel machines at second stage*, Computers & Operations Research, 39, 736-745, (2012)

Castrillón, O. D., Giraldo, J. A. y Sarache, W. A. *Técnicas Inteligentes y Estocásticas en Scheduling. Un Enfoque en la Producción y las Operaciones*, Universidad Nacional de Colombia, Manizales, Colombia, (2010)

Cevikcan, E., Bulent Durmusoglu, M. y Baskak, M. *Integrating parts design characteristics and scheduling on parallel machines*, Expert Systems with Applications, 38, 13232-13253, (2011)

Chang, P. C., Hsieh, J. C. y Wang, Y. W. *Genetic algorithms applied in BOPP film scheduling problems: minimizing total absolute deviation and setup times*, Applied Soft Computing, 3, 139-148, (2003)

Chang, P. C. y Chen, S. H. *Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times*, Applied Soft Computing, 11, 1263-1274, (2011)

Chiou, C. W., Chen, W. M., Liu, C. M. y Wu, M. C. *A genetic algorithm for scheduling dual flow shops*, Expert Systems with Applications, 39, 1306-1314, (2012)

Costa, A., Cappadonna, F. A. y Fichera, S. *A dual encoding-based meta-heuristic algorithm for solving a constrained hybrid flow shop scheduling problem*, Computers & Industrial Engineering, 64, 937-958, (2013).

Engin, O., Ceran, G. y Yilmaz, M. K. *An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems*, Applied Soft Computing, 11, 3056-3065, (2011)

Escobar, P., Giraldo, J. A., y Cárdenas, D. M. *Programación de Sistemas de Producción Híbridos, Para inventario/Bajo pedido, mediante un Proceso Analítico Jerárquico de Ordenación Grupal (GAHPO)*. Información Tecnológica, 23, (5), 33-46, (2012)

Feng, H., Lu, S. y Li, X. *Genetic algorithm for hybrid flow-shop scheduling with parrel batch processors*, Actas del WASE International Conference on Information Engineering, 2009. ICIE '09, 2, 9-13, Taiyuan, China, 10 y 11 de Julio, (2009)

Gallego, R. A., Escobar, A. H. y Romero, R. A. *Técnicas de Optimización Combinatorial*. Universidad Tecnológica de Pereira, Pereira, Colombia, (2006)

Garey, M. R., Johnson, D. S. y Stockmeyer, L. *Some simplified NP-Complete graph problems*, Theoretical Computer Science, 1, 237-267, (1967)

Gendreau, M. y Potvin, J. Y. *Handbook of Metaheuristics. Second Edition*. Springer Science + Business Media, LLC, New York, USA, (2010)

Gómez-Gasquet, P. *Un nuevo algoritmo genético basado en un sistema multiagente para la programación de la producción en un taller de flujo híbrido*, Actas del International Conference on Industrial Engineering & Industrial Management - CIO 2007, 1675-1685, Madrid, España, 5 al 7 de Septiembre, (2007)

Gómez-Gasquet, P., Andrés, C. y Lario, F. C. *An agent-based genetic algorithm for hybrid flowshops with sequence dependent setup times to minimise makespan*, Expert Systems with Applications, 39, 8095-8107, (2012)

Hekmatfar, M., Fatemi-Ghomi, S. M. T. y Karimi, B. *Two stage reentrant hybrid flow shop with setup times and the criterion of minimizing makespan*, Applied Soft Computing, 11, 4530-4539, (2011)

Hmida, A. B., Haouari, M., Huguet, M. J. y Lopez, P. *Solving two-stage hybrid flow shop using climbing depth-bounded discrepancy search*, Computers & Industrial Engineering, 60, 320-327, (2011)

Hojjati, S. M. H. y Sahraeyan, A. *Minimizing makespan subject to budget limitation in hybrid flow shop*, Actas del International Conference on Computers & Industrial Engineering, 2009. CIE 2009, 18-22, Troyes, Francia, 6 al 9 de Julio, (2009)

Holland, J. H. *Adaptation in Natural and Artificial Systems*, The MIT Press, Michigan, USA, (1992)

Hsu, H. M., Hsiung, Y., Chen, Y. Z. y Wu, M. C. *A GA methodology for the scheduling of yarn-dyed textile production*, Expert Systems with Applications, 36, 12095-12103, (2009)

Jabbarizadeh, F., Zandieh, M. y Talebi, D. *Hybrid flexible flowshops with sequence-dependent setup times and machine availability constraints*, Computers & Industrial Engineering, 57, 949-957, (2009)

Jing, C., Huang, W. y Tang, G. *Minimizing total completion time for re-entrant flow shop scheduling problems*, Theoretical Computer Science, 412, 6712-6719, (2011)

Kasperski, A., Kurpisz, A. y Zielinski, P. *Approximating a two-machine flow shop scheduling under discrete scenario uncertainty*, European Journal of Operational Research, 217, 36-43, (2012)

Lin, S.-W., Lee, Z.-J., Ying, K.-C. y Lu, C.-C. *Minimization of maximum lateness on parallel machines with equence-dependent setup times and job release dates*, Computers & Operations Research, 38, 809-815, (2011)

López, J. C., Metodología de Programación de Producción en un *Flow Shop* Híbrido Flexible con el Uso de Algoritmos Genéticos para Reducir el *Makespan*. Aplicación en la Industria Textil, Tesis de Maestría, Departamento de Ingeniería Industrial, Universidad Nacional de Colombia, Manizales, Colombia (2013)

Mahdavi, I., Mojarad, M. S., Javadi, B. y Tajdin, A. *A genetic approach for solving a hybrid flow shop scheduling problem*, Actas del IEEE International Conference on Industrial Engineering and Engineering Management, 2008. IEEM 2008, 1214-1218, Singapur, Singapur, 8 al 11 de Diciembre, (2008)

Mesghouni, K. y Rabenasolo, B. *Multi-period predictive production scheduling with uncertain demands*, Actas del 2002 IEEE International Conference on Systems, Man and Cybernetics, 6, Yasmine Hammamet, Túnez, 6 al 9 de Octubre, (2002)

Naderi, B., Tavakkoli-Moghaddam, R. y Khalili, M. *Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan*, Knowledge-Based Systems, 23, 77-85, (2010)

Onwubolu, G. C. y Mutingi, M. *Genetic algorithm for minimizing tardiness in flow-shop scheduling*, Production Planning & Control, 10, (5), 462-471, (1999)

Phanden, R. K., Jain, A. y Verma, R. *A genetic algorithm-based approach for job shop scheduling*, Journal of Manufacturing Technology Management, 23, (7), 937-946, (2012)

Pinedo, M. L. *Planning and Scheduling in Manufacturing and Services*. Springer Science + Business Media, LLC, New York, USA, (2005)

Qiao, P. y Sun, C. *Research on hybrid flow-shop scheduling problem based on improved immune particle swarm optimization*, Actas del 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 4240-4243, Deng Feng, China, 8 al 10 de Agosto, (2011)

Rajkumar, R. y Shahabudeen, P. *Bi-criteria improved genetic algorithm for scheduling in flowshops to minimise makespan and total flowtime of jobs*, International Journal of Computer Integrated Manufacturing, 22, (10), 987-998, (2009)

Rebaine, D. *Scheduling flexible flowshops with unit-time operations and minimum time delays*, Electronic Notes in Discrete Mathematics, 36, 1193-1200, (2010)

Ribas, I., Leisten, R. y Framiñan, J. M. *Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective*, Computers & Operations Research, 37, 1439-1454, (2010)

Ruiz, R. y Maroto, C. *A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility*, European Journal of Operational Research, 169, 781-800, (2006)

Safari, E. y Sadjadi, S. J. *A hybrid method for flowshops scheduling with condition-based maintenance constraint and machines breakdown*, Expert Systems with Applications, 38, 2020-2029, (2011)

Schroeder, R. *Administración de Operaciones: Casos y Conceptos Contemporáneos*. McGraw-Hill, México D.F., México, (2004)

Serifoglu, F. S. y Ulusoy, G. *Multiprocessor task scheduling in multistage hybrid flow-shops: a genetic algorithm approach*, Journal of the Operational Research Society, 55, 504-512, (2004)

Shabtay, D. *The just-in-time scheduling problem in a flow-shop scheduling system*, European Journal of Operational Research, 216, 521-532, (2012)

Shiroma, P. J. y Niemeyer, G. *Production scheduling in the textile industry: a practical approach using evolutionary algorithms with domain-dependent information*, Actas del 24th Annual Conference of the IEEE Industrial Electronics Society, 1998. IECON '98, 1, 269-273, Aachen, Alemania, 31 de Agosto al 4 de Septiembre, (1998)

Solari, M. y Ocampo, E. *Aplicación de algoritmos genéticos en un sistema multiagente de planificación en una industria manufacturera*, Actas del XXXII Conferencia Latinoamericana de Informática (CLEI 2006), Santiago de Chile, Chile, 21 al 25 de Agosto, (2006)

Su, Z. y Li, T. *Genetic algorithm for minimizing the makespan in hybrid flow shop scheduling problem*, Actas del International Conference on Management and Service Science, 2009. MASS '09, 1-4, Wuhan, China, 20 al 22 de Septiembre, (2009)

Sun, L. H., Sun, L. Y., Wang, M. Z. y Wang, J. B. *Flow shop makespan minimization scheduling with deteriorating jobs under dominating machines*, International Journal of Production Economics, 138, 195-200, (2012)

Tang, J., Zhang, G., Lin, B. y Zhang, B. *Hybrid genetic algorithm for flow shop scheduling problem*, Actas del 2010 International Conference on Intelligent Computation Technology and Automation (ICICTA), 2, 449-452, Changsha, China, 11 y 12 de Mayo, (2010)

Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, M., Izadi, M. y Sassani, F. *Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints*, Computers & Operations Research, 36, 3224-3230, (2009)

Toledo, C. F. M., de Oliveira, R. R. R. y Morelato França, P. *A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging*, Computers & Operations Research, 40, 910-919, (2013)

Toro, E. M., Granada, M. y Romero, R. *Algoritmo memético aplicado a la solución del problema de asignación generalizada*, Revista Tecnura, 8, (16), 55-63, (2005)

Vallada, E. y Ruiz, R. *A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times*, European Journal of Operational Research, 211, 612-622, (2011)

Wang, S. y Liu, M. *A heuristic method for two-stage hybrid flow shop with dedicated machines*, Computers & Operations Research, 40, 438-450, (2013)

Xiao, W., Hao, P., Zhang, S. y Xu, X. *Hybrid flow shop scheduling using genetic algorithms*, Actas del 3rd World Congress on Intelligent Control and Automation, 1, 537-541, Hefei, China, 28 de Junio al 2 de Julio, (2000)

Yalaoui, N., Mahdi, H., Amodeo, L. y Yalaoui, F. *A particle swarm optimization under fuzzy logic controller to solve a scheduling problem*, Actas del 2011 International Conference on Communications, Computing and Control Applications (CCCA), 1-6, Hammamet, Túnez, 3 al 5 de Marzo, (2011)

Yang, J. *Minimizing total completion time in two-stage hybrid flow shop with dedicated machines*, Computers & Operations Research, 38, 1045-1053, (2011)

Yao, F. S., Zhao, M. y Zhang, H. *Two-stage hybrid flow shop scheduling with dynamic job arrivals*, Computers & Operations Research, 39, 1701-1712, (2012)

Yaurima, V. Burtseva, L. y Tchernykh, A. *Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers*, Computers & Industrial Engineering, 56, 1452-1463, (2009)

Zandieh, M. y Karimi, N. *An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times*, Journal Intelligent Manufacturing, 22, (6), 979-989, (2010)

Zhan, Y. y Qiu, C. *Genetic algorithm application to the hybrid flow shop scheduling problem*, Actas del IEEE International Conference on Mechatronics and Automation, 2008. ICMA 2008, 649-653, Takamatsu, Japón, 5 al 8 de Agosto, (2008)

Zhang, X. y van de Velde, S. *Approximation algorithms for the parallel flow shop problem*, European Journal of Operational Research, 216, 544-552, (2012)

Zhao, C. y Tang, H. *Two-machine flow shop scheduling with deteriorating jobs and chain precedence constraints*, International Journal of Production Economics, 136, 131-136, (2012)