Un Algoritmo Genético Híbrido para optimizar un caso especial del problema de Secuenciación de Tareas.

Rodolfo Najarro[a], Ringo Lopez[a], Elizabeth Racines[b], Amilkar Puris[c]

[a] Facultad de Ciencias de la Ingeniería, Universidad del Cotopaxi, extensión La Mana, La Mana, Ecuador
rodolfo.najarro@utc.edu.ec, ringo.lopez@utc.edu.ec

[b] Departamento de Talento Humano, Universidad Técnica Estatal de Quevedo, Quevedo, Ecuador
eracines@uteq.edu.ec

[b] Facultad de Ciencias de la Ingeniería, Universidad Técnica Estatal de Quevedo, Quevedo, Ecuador
apuris@uteq.edu.ec

Resumen. Este trabajo tiene como objetivo contribuir con el esfuerzo de recientes investigaciones de cerrar la brecha existente entre la teoría y la práctica en la solución a problemas de secuenciación de tareas, analizando el efecto de la inclusión de varias restricciones que influyen negativamente en la programación de la producción en un ambiente de manufactura real. Para la solución del problema, se introduce un eficiente Algoritmo Genético combinado con una Búsqueda Local de Vecindad Variable para problemas de *n* tareas y *m* máquinas minimizando el tiempo de completamiento total de las tareas o *makespan*. Las restricciones de fechas de liberación, tiempos de configuración dependientes de la secuencia y tiempos de transportación son introducidas. Estas son restricciones comunes que pueden ser encontradas en múltiples ambientes manufactureros donde existen máquinas herramientas y un conjunto de tareas debe procesarse en éstas siguiendo el mismo patrón de flujo. Los experimentos computacionales realizados sobre un conjunto de instancias de problemas de diferentes tamaños de complejidad demuestran que la metaheurística híbrida propuesta alcanza soluciones de alta calidad comparables con los óptimos reportados.

Palabras Clave: Algoritmo Genético híbrido; Secuenciación de Tareas; programación de la producción; búsqueda ocal de vecindad variable.

1 Introducción

El proceso de toma de decisiones se puede distinguir de tres maneras: a largo, mediano y corto plazo. A largo plazo, decisiones estratégicas en el ámbito productivo, por ejemplo, podrían ser la determinación de cuántas máquinas se deben comprar o qué productos se pueden ofrecer al mercado. La aceptación de pedidos de clientes o planificación de personal constituyen ejemplos de decisiones a mediano plazo. Sin embargo, una decisión a corto plazo, que es el tipo de decisión que más comúnmente es realizada en un entorno real de la producción sería: en qué momento y sobre cuáles máquinas se debe procesar un conjunto de tareas.

Un problema de secuenciación de tareas o problema de *scheduling*, como también se le conoce por su palabra en el idioma inglés, es precisamente esto: un proceso de asignación de tareas a un conjunto limitado de recursos disponibles en un intervalo de tiempo, donde determinados criterios son optimizados [1; 2].

De esta forma, la secuenciación de tareas está directamente asociada con la ejecutabilidad y optimalidad de un plan preestablecido y puede ser encontrada en una amplia gama de aplicaciones, tales como: programación de despacho de vuelos en los aeropuertos, programación de líneas de producción de una fábrica, programación de cirugías en un hospital, reparación de equipos o maquinarias en un taller, entre otros [3; 4].

En muchas ocasiones, la cadena de producción estructurada para la secuenciación de procesos requiere que cada una de las tareas a ejecutar transite por todas las etapas productivas en el mismo orden. Este problema es conocido de forma general como Variante de Flujo Regular o *Flow Shop Scheduling Problem* (FSSP), el cual, como muchos otros en este campo, son de difícil solución y están clasificados técnicamente como de solución en un tiempo no polinomial (NP-*Hard*) [5; 6; 7].

De forma general, desde la década del 50, científicos en el área de la Investigación de Operaciones (IO) se apoyan en métodos matemáticos de optimización para problemas de *scheduling* contribuyendo con una variedad amplia de metodologías en las que se incluyen la programación lineal [8; 9; 10; 11; 12] hasta diferentes técnicas avanzadas de Inteligencia Artificial (IA) [13; 14; 15; 16; 17; 18; 19].

La mayoría de los métodos descritos anteriormente, garantizan, en muchos de los casos, soluciones óptimas que son logradas en un período de tiempo razonable; sin embargo, de forma general, estas excluyen situaciones que se presentan en la práctica industrial. Desafortunadamente, los problemas de *scheduling* en el mundo real, al presentar un conjunto de restricciones propias de estos entornos, se convierten en problemas mucho más desafiantes, dificultando, en muchos casos, la aplicabilidad de dichos métodos [20].

El modelado es un tema a considerar al solucionar problemas de *scheduling* en un entorno real. Los innumerables tipos de procesos de fabricación, cada uno con sus propias particularidades, hacen difícil construir modelos generalmente aplicables. Algunas restricciones son difíciles de representar matemáticamente. Por otra parte, encontrar las restricciones correctas para modelar la realidad depende sobre todo del conocimiento extenso del dominio, que puede ser inasequible para los investigadores y que es solamente utilizable para un tipo particular de proceso de producción. Entre estas se pueden encontrar las siguientes:

- Fechas de liberación de las máquinas: En las empresas manufactureras antes de comenzar el proceso productivo, las máquinas son configuradas inicialmente según el tipo de trabajo que pueden realizar.
- Tiempos de configuración dependientes de la secuencia: Las operaciones de cambio de referencia en las máquinas varían constantemente. Estas se demoran más cuando el trabajo entrante es muy diferente al saliente, impactando el desempeño total de las operaciones.
- Tiempos de transportación: Los trabajos necesitan de un intervalo de tiempo para ser transportados de una máquina a otra.

Todo esto da lugar a que en las empresas se tenga que revisar la política de generar buenas programaciones de la producción. A su vez, refuerza la necesidad de automatizar el modelado de problemas aplicando algoritmos que puedan adaptarse a las características de un entorno real. Todo lo anteriormente planteado ilustra la necesidad de investigar otras estrategias, así como nuevos y eficientes algoritmos para resolver el problema en cuestión.

Por tal motivo, en este trabajo se presenta una propuesta de un Algoritmo Genético Híbrido el cual es combinado con una Búsqueda Local de Vecindad Variable para la solución del FSSP bajo el efecto de un conjunto de restricciones que se presentan con frecuencia en entornos manufactureros reales. El algoritmo propuesto es probado con varias instancias del problema mostrando que el mismo alcanza excelentes soluciones comparables con las óptimas.

2. Materiales y Métodos

En esta sección se realiza una formalización del problema de secuenciación de tareas que se aborda en esta investigación, se introducen los Algoritmos Genéticos como metodología computacional para la solución de problemas de optimización, y por último, se diseña la metaheurística híbrida propuesta que es empleada para la solución del problema tratado.

2.1 Variante de Flujo Regular o Flow Shop Scheduling

Dentro de la teoría de *scheduling* se pueden distinguir un gran número de problemas. En estos se tiene un conjunto de $\boldsymbol{N}$ trabajos que han de ser procesados sobre un conjunto de $\boldsymbol{M}$ recursos o máquinas físicas siguiendo un patrón de flujo o ruta tecnológica. Una secuenciación consiste en encontrar para cada trabajo un tiempo o un intervalo de tiempos en los que este pueda procesarse en una o varias máquinas [21; 22]. El objetivo es encontrar una secuencia sujeta a una serie de restricciones que optimice una o varias funciones objetivo [23; 24].

En general, en un problema de *scheduling* intervienen los siguientes elementos: trabajos, actividades, máquinas, tipo de *scheduling* (patrón de flujo) y objetivos. Con frecuencia, estas actividades deben ser ejecutadas para todos los trabajos en el mismo orden, implicando que estos sigan el mismo patrón de flujo. Las máquinas, entonces, son configuradas en serie y el ambiente es referido como Variante de Flujo Regular o FSSP.

En este trabajo, el FSSP está sujeto a las siguientes restricciones:

- Solo se cuenta con una máquina-herramienta de cada tipo por etapa.
- Las restricciones tecnológicas están bien definidas y son previamente conocidas, además de que son inviolables.
- No está permitido que dos operaciones del mismo trabajo se procesen simultáneamente.
- Ningún trabajo puede ser procesado más de una vez en la misma máquina.

- Cada trabajo es procesado hasta concluirse, una vez que se inicia una operación esta se interrumpe solamente cuando se concluye.
- Ninguna máquina puede procesar más de un trabajo a la vez.
- Se tiene en cuenta las fechas de liberación de las máquinas.
- Los tiempos de configuración dependientes de la secuencia se conocen de antemano.
- El tiempo de transportación de los trabajos entre etapas es considerado.

El objetivo es encontrar una secuencia de trabajos por etapas bajo la restricción de que el procesamiento de cada trabajo tiene que ser continuo con respecto al objetivo de minimizar el *makespan* o C_{max} como también se le conoce.

Por lo tanto: si tenemos a r(j) como el tiempo de liberación de la máquina j, t(i, j, l) como el tiempo de transportación del trabajo i desde la máquina j a la máquina l,p(i, j) como el tiempo de procesamiento del trabajo i en la máquina j, s(i, k, j) como el tiempo de configuración dependiente de la secuencia entre el trabajo i y el trabajo k en la máquina j, y una permutación de trabajos (J1, J2,…,Jn), entonces calculamos el tiempo de completamiento total denotado por C_{max} como sigue:

$$C(J_1, 1) = r(1) + p(J_1, 1) \quad . \tag{1}$$

$$C(J_i, 1) = s(J_{i-1}, J_i, 1) + C(J_{i-1}, 1) + p(J_i, 1), \qquad i = 2, \dots, n \quad . \tag{2}$$

$$C(J_1, j) = \max\{r(j), C(J_1, j-1) + t(i, j-1, j) + p(J_1, j)\}, \qquad j=2,\dots, m \quad . \tag{3}$$

$$C(J_i, j) = \max\{C(J_{i-1}, j) + s(J_{i-1}, J_i, j),\ C(J_i, j-1) + t(i, j-1, j) + p(J_i, j)\}, \tag{4}$$

$i= 2,\dots, n; j=2,..,m$.

$$C_{max} = C(J_n, m) \quad . \tag{5}$$

Bajo estas condiciones, el tiempo de procesamiento total corresponde al tiempo de culminación de procesamiento del último trabajo en la última máquina. En otras palabras, es el tiempo necesario para completar todos los trabajos [1].

2.2 Algoritmos Genéticos

Los Algoritmos Genéticos (AGs), introducidos por Holland [25], son métodos adaptativos basados en el proceso biológico de organismos que han sido ampliamente utilizados en problemas de optimización combinatoria [26; 23; 27]. De forma general, los Ags trabajan manteniendo una población fija (tamaño de la población) de individuos (cromosomas) las cuales representan soluciones del problema. Para representar una solución, una codificación adecuada debe ser diseñada. La misma codifica un conjunto de parámetros (genes) que unidos forman una cadena de valores (cromosomas). En cada generación, una función de aptitud es usada para evaluar (relativo a la función objetivo del problema) la aptitud de cada cromosoma. Entonces, se seleccionan aleatoriamente cromosomas (padres) de la población con el objetivo de aparearlos usando un esquema

que favorece a los cromosomas más aptos. Luego, operadores de cruzamiento y mutación son usualmente usados para combinar los cromosomas padres para generar una mejor descendencia. Luego, estos descendientes son insertados en la población actual para formar una nueva población para la nueva generación. El proceso se repite hasta que una condición de parada se cumple. La Figura 1 muestra la estructura básica de un AG.

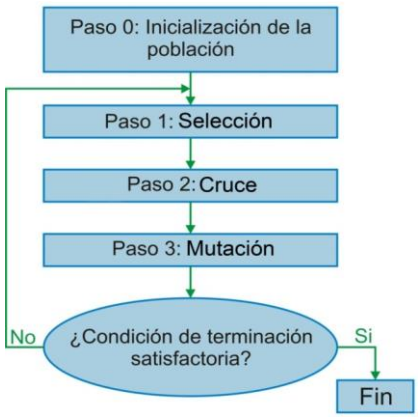

Fig. 1. Estructura básica de un AG

La Figura 1 recoge los pasos fundamentales de un AG. Se inicializa la población aleatoriamente y se evalúan los individuos de la población. En función de la aptitud de los individuos se seleccionan los padres para ser recombinados. El proceso de recombinación tiene como objetivo tomar información de varios individuos combinándola en otro, que supuestamente debe ser mejor. Durante la alteración del material genético, la mutación se realiza una exploración en una vecindad del individuo a mutar. Los criterios de parada involucran generalmente ideas relacionadas con la convergencia de la población de individuos, el número de generaciones obtenidas por el AG y el número de evaluaciones de la función objetivo.

El ciclo de evolución de un AG mostrado en la Figura 2, permite entender de forma sencilla su funcionamiento.

Fig. 2. Ciclo de la evolución de un AG

2.3 Algoritmo Genético Híbrido Propuesto

La solución de un problema utilizando AG comienza por el diseño de la representación de la solución del problema dado. La representación de las posibles soluciones dentro del espacio de búsqueda de un problema define la estructura del cromosoma que va ser manipulado por el algoritmo. Existen diferentes tipos de representaciones, la elección de cuál usar dependerá siempre de las características del problema a resolver [28; 19]. En el caso del problema estudiado en este trabajo se utiliza una representación similar a la utilizada en [29] en la cual el cromosoma representa la secuencia natural en la que se procesan los trabajos. La Figura 3 muestra el cromosoma que representa una solución para el FSSP donde son procesados 8 trabajos. La secuencia de números representa el número del trabajo y el orden en el que serán procesados. La aptitud de cada cromosoma estará dado por el cálculo del C_{max} a partir de la secuencia de trabajo representada en el el mismo.

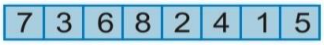

Fig. 3. Representación de orden

El flujo de trabajo del Algoritmo Genético Híbrido (AGHVNS) propuesto es presentado a través de un diagrama de bloques en la Figura 4.

.

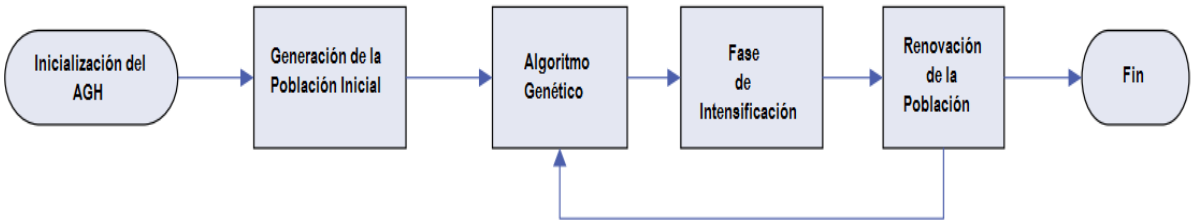

Fig. 4. Flujo de trabajo del AGHVNS

Cuando se genera la población inicial (pob_ini), el objetivo es adquirir una población diversificada. En el AGHVNS propuesto, la población inicial es generada aleatoriamente y el tamaño es variable en dependencia de la cantidad de trabajos (n) y de máquinas (m) donde $\text{pob}_{\text{ini}} = \text{n} * \text{m}$. En el paso 2, un AG básico (ver Figura 1) es empleado para mejorar la población. En nuestra implementación, la selección de padres está basada en el esquema clásico de selección por Ranking donde $\text{pob\_ini}/2$ parejas de padres son seleccionadas. Para la selección de cada pareja de padres se van apareando de dos en dos a los cromosomas de mayor a los de menor aptitud. Luego a esto, se realiza el cruzamiento de cada apareamiento. En nuestro caso se aplica el método Dos Punto de Cruce. Es preciso mencionar que al aplicar directamente este método pueden generarse cromosomas ilegales (en nuestro caso significa que en un mismo cromosoma puede estar un trabajo dos veces). Este problema es corregido utilizando la metodología estudiada en [30] la cual está basada en la sustitución del o de los trabajos que se repiten por los que no se encuentran en dicho cromosoma. Luego a la combinación se realizaría el proceso de mutación el cual consiste en seleccionar

dos puntos del cromosoma aleatoriamente y cambiar de posición estos trabajos. La Figura 5 muestra este proceso.

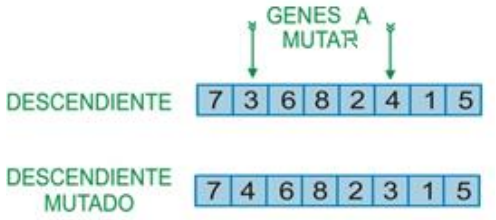

Fig. 5. Mutación por Valor para el FSSP

[31] Enfatizan la importancia del balance entre la búsqueda del AG y la búsqueda local para encontrar soluciones FSSP de alta calidad en tiempos computacionales aceptables. En el paso 3 de nuestro algoritmo, una Búsqueda Local de Vecindad Variable es aplicada para explorar la vecindad de un subconjunto de la población complementando el AG. En cada generación se seleccionan los pob_ini/2 individuos obtenidos en el paso 2. Esta intensificación opera sobre cada uno de los individuos seleccionados aplicando un operador de vecindad el cual es escogido aleatoriamente. El valor y detalle de cada operador se describe como sigue:

- Operador de Intercambio: este funciona de manera similar al operador de mutación.
- Operador de traslado o inserción: se selecciona un trabajo aleatoriamente y este se mueve hacia otra posición del cromosoma seleccionada aleatoriamente.
- Operador 2-Op: Se seleccionan dos trabajos consecutivos aleatoriamente, luego se selecciona una posición aleatoriamente dentro del cromosoma y estos son trasladados hacia esa posición.
- No se realiza ninguna operación sobre el cromosoma.

La idea principal detrás de esta intensificación es diversificar la población y lograr un balance entre la exploración y explotación.

Por último, en el paso 4 de nuestro algoritmo, se procede a renovar la población a partir de los nuevos individuos generados. Específicamente, los pob_ini/2 individuos menos aptos de la población actual son sustituidos por las soluciones obtenidas en la fase de intensificación.

3. Resultados y Discusión

Instancias de problemas FSSP han sido definidas por diferentes autores y ampliamente utilizadas por muchos investigadores en el campo de scheduling para analizar el rendimiento de las metodologías propuestas y comparar sus soluciones contra otros algoritmos. Estas instancias están disponibles en [32; 33]. Sin embargo, no existen instancias de problemas disponibles para el FSSP bajo el efecto de las restricciones mencionadas en la sección 2.1. Por esta razón, todos los datos para la experimentación computacional fueron generados aleatoriamente.

Con el objetivo de analizar el rendimiento del algoritmo propuesto, 10 instancias son utilizadas. Teniendo en cuenta que el espacio de búsqueda para el FSSP es n!, estas instancias fueron creadas con dimensiones pequeñas con el objetivo de realizar una búsqueda exhaustiva en este espacio para determinar la solución óptima y comparar estos valores con los obtenidos por el AGHVNS. De igual manera, estos resultados son comparados con la variante del AG propuesto en este trabajo excluyendo la fase de intensificación. De esta forma se analiza el efecto de esta fase en la solución del problema.

La dimensión de estas instancias es: 5x3, 5x4, 5x5, 7x6, 7x7, 8x8, 9x4, 9x9, 10x8 y10x10. Se generaron números aleatorios para crear los tiempos de procesamiento, tiempos de configuración dependiente de la secuencia, fechas de liberación y tiempos de transportación. Cada instancia fue ejecutada 10 veces.

Para determinar la calidad de las soluciones, el Error Relativo Medio (ERM) es definido como:

$$ERM = \sum_{i=1}^{10} \left[\frac{MK_i - UB_i}{UB_i} * 100 \right] / 10 \qquad (6)$$

Donde MK es la solución obtenida por nuestro algoritmo y UB es el óptimo obtenido mediante la búsqueda exhaustiva.

El AGHVNS fue implementado en Java y todas las ejecuciones fueron realizadas en un PC Pentium IV a 2.4 GHz con 1 GB de memoria RAM.

3.1 Calibración de Parámetros

Una gran cantidad de ejecuciones revelaron que utilizando el número de generaciones del AGH_{VNS} como condición de parada no era adecuado para controlar la convergencia del mismo. En muchos casos, el algoritmo alcanzó un óptimo local antes de completar la cantidad de generaciones. Como resultado, se introdujo un tiempo de procesamiento máximo el cual depende de la dimensión del problema a solucionar. El mismo fue arbitrariamente definido como $t_{max}=n*m/5$ segundos, el cual, en términos computacionales, es un tiempo práctico y en todos los casos, permite al algoritmo culminar en un tiempo menor que definiendo como condición de parada el número de generaciones.

Los valores de los demás parámetros principales del AGH_{VNS} son resumidos en la Tabla 1. Estos valores son tomados de [29], en la cual los autores realizan un estudio del comportamiento de los mismos en la solución del FSSP clásico.

Tabla 1. Parámetros iniciales del AG propuesto.

Parámetro	Valor
Factor de Cruzamiento	0.53
Factor de Mutación	0.021

Método de Selección	Ranking
Método de Cruzamiento	Dos Punto de Cruce
Método de Mutación	Mutación por Valor
Tipo de Modelo	Modelo Estacionario

3.2 Descripción de una Instancia

Para ilustrar el problema que se soluciona en esta investigación se introduce un ejemplo. Este describe una instancia con 5 trabajos que deben ser procesados sobre 5 máquinas (5x5). Las Tablas 2, 3, 4 y 5 muestran el tiempo de procesamiento de los trabajos sobre las máquinas, la fecha de liberación de cada una de las máquinas, los tiempos de configuración dependientes de la secuencia y los tiempos de transportación respectivamente.

Tabla 2. Tiempos de Procesamiento

Trabajo/ Máquina	Tiempos de Procesamiento				
	J_0	J_1	J_2	J_3	J_4
M_0	10	6	11	8	11
M_1	15	9	14	10	14
M_2	12	11	9	10	6
M_3	8	4	8	9	12
M_4	6	6	8	6	3

La Tabla 2 es interpretada como el tiempo que necesita un trabajo en procesarse en cada una de las máquinas. Por ejemplo, el trabajo J_0 necesita 10, 15, 8, 12 y 6 unidades de tiempo para procesarse en cada una de las 5 máquinas.

Tabla 3. Fechas de liberación

Maquina	Fecha de Liberación
M_0	9
M_1	3
M_2	8
M_3	16
M_4	23

Por su parte, la Tabla 3 muestra el tiempo que necesita cada máquina para ser liberada al comienzo de la planificación de la producción.

Tabla 4. Tiempos de Configuración dependientes de la secuencia

M_0	Tiempos de Configuración					M_1	Tiempos de Configuración					M_2	Tiempos de Configuración				
	J_0	J_1	J_2	J_3	J_4		J_0	J_1	J_2	J_3	J_4		J_0	J_1	J_2	J_3	J_4
J_0	0	2	4	6	5	J_0	0	3	6	8	9	J_0	0	4	6	3	3
J_1	3	0	7	9	4	J_1	2	0	5	4	1	J_1	2	0	5	7	3
J_2	3	2	0	5	6	J_2	1	1	0	3	4	J_2	4	7	0	3	5
J_3	2	4	6	0	7	J_3	4	4	5	0	7	J_3	8	8	5	0	12
J_4	3	3	2	5	0	J_4	5	4	2	10	0	J_4	4	4	3	10	0
M_3	Tiempos de Configuración					M_4	Tiempos de Configuración										
	J_0	J_1	J_2	J_3	J_4		J_0	J_1	J_2	J_3	J_4						
J_0	0	2	4	6	5	J_0	0	4	6	3	3						
J_1	3	0	7	9	4	J_1	2	0	5	7	3						
J_2	3	2	0	5	6	J_2	4	7	0	3	5						
J_3	2	4	6	0	7	J_3	8	8	5	0	12						
J_4	3	3	2	5	0	J_4	4	4	3	10	0						

En la Tabla 4 se detallan los tiempos de configuración dependiente de la secuencia. Esto significa, por ejemplo, que si la máquina M_0 procesa al trabajo J_0 y luego debe procesar al trabajo J_1, necesitaría 2 unidades de tiempo para configurar la máquina antes de procesar a J_1.

Por último, en la Tabla 5 se muestran los tiempos de transportación de los trabajos desde una máquina hacia la otra. Por ejemplo, para trasladar el trabajo J_0 desde la máquina M_0 hacia la máquina M_1, se necesitan 8 unidades de tiempo, de M1 hacia M_2 se necesitan 7 unidades de tiempo y así sucesivamente. En la tabla solo se muestran los valores para las primeras cuatro máquinas debido a que en la última etapa (M_5) el trabajo culmina su procesamiento.

Tabla 5. Tiempos de Transportación

Máquina/Trabajo	Tiempos de Trasportación				
	J_0	J_1	J_2	J_3	J_4
M_0	8	4	6	8	7
M_1	7	8	8	5	7
M_2	6	9	8	5	9
M_3	6	5	4	7	6

4. Resultados Experimentales

Como se mencionó anteriormente, dada la inexistencia de instancias del problema tratado en esta investigación, se generaron aleatoriamente instancias de diferentes tamaños de complejidad. Específicamente, se crearon 10 instancias (n x m) de 5x3, 5x4, 5x5, 7x6, 7x7, 8x8, 9x4, 9x9, 10x8 y 10x10, donde n representa la cantidad de trabajos y m la cantidad de máquinas respectivamente. El algoritmo propuesto fue ejecutado 10 veces para cada una de estas y el valor promedio de makespan (C_{max}(prom)) es calculado. Además, con el objetivo de analizar el rendimiento del algoritmo propuesto, el proceso de intensificación fue deshabilitado obteniendo un AG básico, el cual, al igual que el AGH_{VNS}, fue ejecutado 10 veces por cada una de las instancias. Estos resultados se tomaron en cuenta para el estudio comparativo.

Por otro lado, dado que estas instancias fueron generadas intencionalmente con un nivel de complejidad pequeño (dado por el número de trabajos y máquinas), se determinó el valor óptimo de C_{max} para cada una de ellas realizando una búsqueda exhaustiva. El ERM fue calculado a partir de estos valores y los obtenidos por las dos variantes de AGs.

La Tabla 6 resume los resultados obtenidos. En esta se muestra además el menor valor de makespan (C_{max}(min)) obtenido en las ejecuciones por ambas variantes. La última fila muestra el promedio de ERM.

Tabla 6. Resumen de los resultados obtenidos.

Instancia	Óptimo	AGH_{VNS}		ERM(%)	AG		ERM(%)
		C_{MAX}(min)	C_{MAX}(prom)		C_{MAX}(min)	C_{MAX}(prom)	
5x3	103	103	103.00	0.000	103	103.00	0.000
5x4	117	117	117.00	0.0000	117	117.90	0.0077
5x5	140	140	140.00	0.0000	144	144.00	0.0286
7x6	207	207	207.00	0.0000	208	208.50	0.0072
7x7	236	236	236.00	0.0000	236	236.80	0.0034
8x8	256	257	257.00	0.0039	262	262.90	0.0270
9x4	185	185	185.80	0.0043	193	194.30	0.0503
9x9	292	292	293.80	0.0062	300	302.00	0.0342
10x8	289	290	290.40	0.0048	296	297.10	0.0245
10x10	338	339	339.90	0.0056	347	347.00	0.0266
PROMEDIO:				**0.0025**			**0.0209**

4.1 Discusión de los Resultados

Basado en los resultados de la Tabla 6 se puede observar que el algoritmo propuesto es capaz de obtener buenos resultados mostrando la aplicabilidad del mismo a problemas

de scheduling teniendo en cuenta restricciones que se presentan con frecuencia en entornos manufactureros reales. Estos corresponden con lo planteado en la literatura especializada teniendo en cuenta la brecha que existe entre los problemas de la programación de la producción que comúnmente se resuelven por los especialistas en el tema y los que se presentan en la práctica.

Para las 10 instancias propuestas, excepto para la instancia de 8x8, 10x8 y 10x10, el AGHVNS alcanzó el valor óptimo de makespan. Para las primeras cinco instancias, este logró el valor óptimo en todas sus ejecuciones. El promedio de ERM fue de solo un 0.0025%.

Por otra parte, el AG sin la fase de intensificación solo alcanzó el valor óptimo en las dos primeras instancias aunque para el resto, los valores obtenidos de makespan estuvieron cercanos a los óptimos. El promedio de ERM fue de 0.0209%.

Los resultados obtenidos en la tabla 6 muestran la importancia de lograr un balance entre la exploración y la explotación a partir de la fase de intensificación. Esta le permitió al algoritmo propuesto, en el proceso de búsqueda en ese espacio, alcanzar mejores soluciones que la variante de AG sin el proceso de búsqueda local. Esta afirmación es evidenciada al establecer una comparación entre los promedios de ERM entre el AGHVNS y el AG básico. El ERM del AGHVNS, en todos los casos, fue menor que el obtenido por el AG básico y este, a su vez, muy por debajo del 1%.

5. Conclusiones y Recomendaciones

Este trabajo presentó la aplicación de un método híbrido para la solución del FSSP bajo el efecto de restricciones que se presentan en entornos reales de la producción. El algoritmo propuesto está constituido por tres componentes principales: una generación de la población inicial; un AG y un proceso de intensificación a través de una búsqueda local nombrada VNS. El AGH_{VNS} fue evaluado a través de un conjunto de diez instancias de problemas y los resultados obtenidos fueron comparados con los valores óptimos y por los obtenidos mediante la variante básica del AG demostrando eficiencia y efectividad del mismo en cuanto a la calidad de las soluciones. Es importante mencionar que la metaheurística propuesta constituye una interesante alternativa para resolver problemas complejos de optimización. Por otra parte, se debe resaltar que el algoritmo propuesto es simple y fácil de implementar.

Se sugiere que un esquema más sofisticado de intensificación sea utilizado en el futuro. Esta decisión podría lograr un mejor balance entre la explotación y exploración de las soluciones. Por otra parte, se propone incorporar otras restricciones como es el caso de máquinas paralelas no relacionadas, restricciones de precedencia y perturbaciones en las máquinas permitiendo modelar un entorno más realista de la producción. Esta extensión contribuiría aún más, a cerrar la brecha existente entre la teoría de scheduling y la aplicación en configuraciones industriales.

Referencia

1. Fonseca, Y., Martínez, M., Bermudez, J., Méndez, B.: A Reinforcement Learning Approach for Scheduling Problems. Revista Investigación Operacional, 36(3), 225 a 231 (2015).
2. Pinedo, M.: Scheduling Theory, Algorithms, and Systems (3th ed.). New Jersey: Prentice Hall Inc (2008).
3. Fonseca, Y., Martínez, Y., Figueredo, A. E., & Pernía, L. A.: Behavior of the main parameters of the Genetic Algorithm for Flow Shop Scheduling Problems. Revista Cubana de Ciencias Informáticas, 8(1), 99 a 111 (2014).
4. Toro, M., Restrepo, G., & Granada, M.: Adaptación de la técnica de Particle Swarm al problema de secuenciación de tareas. Scientia et Technica UTP, XII(32), 307 a 313 (2006).
5. Ancâu, M.: On Solving Flow Shop Scheduling Problems. Proceedings of the Romanian Academy, 13(1), 71 a79 (2012).
6. Čičková, Z., & Števo, S.: Flow Shop Scheduling using Differential Evolution. Management Information Systems, 5(2), 008 a 013 (2010).
7. Fattahi, P., Hassan-Hosseini, S. M., Jolai, F., Tavakkoli-Moghaddam, R., & Tarantilis, C. D.: A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations. Applied Mathematical Modelling(38), 119 a 134 (2014).
8. Manne, A. S.: On the Job-Shop Scheduling Problem. Operations Research, 8(2), 219 (1960).
9. Medina, P., Cruz, E., & Hernad-Restrepo, J.: Works programming in one machine uses a model Integer linear programming. Scientia et Technica, XIV(40), 111 a 116 (2008).
10. Ramezanian, R., Aryanezhad, M. B., & Heydar, M.: A Mathematical Programming Model for Flow Shop Scheduling Problems for Considering Just in Time Production. International Journal of Industrial Engineering & Production Research, 21(2), 97 a 104 (2010).
11. Sawik, T.: A lexicographic approach to bi-objective scheduling of single-period orders in make-to-order manufacturing. European Journal of Operational Research, 180, 1060 a 1075 (2007).
12. Šeda, M.: Mathematical Models of Flow Shop and Job Shop Scheduling Problems. World Academy of Science, Engineering and Technology, 1(31), 122 a 127 (2007).
13. Chandrasekaran, S., Ponnambalam, S., Suresh, R., & Vijayakumar, N.: Multi-objective particle swarm optimization algorithm for scheduling in flowshops to minimize makespan, total flowtime and completion time variance. Paper presented at the IEEE Congress on Evolutionary Computation (2007).
14. El-Bouri, A., Azizi, N., & Zolfaghari, S.: A comparative study of a new heuristic based on adaptive memory programming and simulated annealing: The case of job shop scheduling. European Journal of Operational Research, 177 1894 a 1910 (2007).
15. González, M.: Soluciones Metaheurísticas al Job-Shop Scheduling Problem with Sequence-Dependent Setup Times. (Tesis Doctoral), Universidad de Oviedo, Oviedo. (2011).
16. Naderi, B., Ruiz, R., & Zandieh, M.: Algorithms for a realistic variant of flowshop scheduling. Computers & Operations Research (37), 236 a 246. (2010)
17. Parviz, F., Seyed Mohammad, H. H., Fariborz, J., & Reza, T.-M.: A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations. Applied Mathematical Modelling, 38, 119 a 134 (2014).
18. Puris, A., Bello, R., Trujillo, Y., Nowe, A., & Martínez, Y.: Two-stage ACO to solve the job shop scheduling problem. Lecture and Notes in Computer Science. (2007).
19. Yamada, T.: Studies on Metaheuristics for Jobshop and Flowshop Scheduling Problems. (Tesis Doctoral), Kyoto University, Kyoto, Japan (2003).
20. Ruiz, R., Sivrikaya, F., & Urlings, T.: Modeling realistic hybrid flexible flowshop scheduling problems. Computers & Operations Research, 35, 1151 a 1175 (2008).
21. Mehmet, Y., & Betul, Y.: Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. Omega, 45 119 a 135 (2014).
22. Seido Naganoa, M., Almeida da Silva, A., & Nogueira Lorena, L. A.: A new evolutionary clustering search for a no-wait flow shop problem with set-up times. Engineering Applications of Artificial Intelligence, 25, 1114 a 1120 (2012).

23.Chie-Wun, C., Wen-Min, C., Chin-Min, L., & Muh-Cherng, W. (2012). A genetic algorithm for scheduling dual flow shops. Expert Systems with Applications(39), 1306–1314.
24.Pinedo, M.: Scheduling Theory, Algorithms, and Systems (3th ed.). New Jersey: Prentice Hall Inc. (2008).
25.Holland, J. H.: Adaption in natural and artificial systems. Ann Arbor: University of Michigan Press. (1975).
26.Chaudhry, I. A., & Munem khan, A.: Minimizing makespan for a no-wait flowshop using genetic algorithm. Sadhana, 36(6), 695 a 707. (2012).
27.Zhang, Y., Li, X., & Wang, Q.: Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. European Journal of Operational Research, 196, 869 a 876. (2009).
28.Márquez, J.: Genetic algorithm applied to scheduling in machine shops. Revista de Ingeniería Mecánica, 15(3), 201 a 212. (2012).
29.Fonseca, Y., Martínez, Y., Figueredo, A. E., & Pernía, L. A.: Behavior of the main parameters of the Genetic Algorithm for Flow Shop Scheduling Problems. Revista Cubana de Ciencias Informáticas, 8(1), 99 a 111. (2014).
30.Murata, T., Ishibuchi, H., & Tanaka, H.: Genetic algorithms for flowshop scheduling problems. Computers and industrial Engineering, 30, 1061 a 1071. (1996).
31.Ishibuchi, H., Yoshida, T., & Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Transactions on Evolutionary Computation, 7(2), 204 a 223. (2003).
32.Beasley, J. E. (1990). OR-Library. Retrieved January 17, 2014, from http://people.brunel.ac.uk/~mastjjb/jeb/info.html
33.Taillard, E. (1993). Benchmarks for basic scheduling problems. European Journal of Operational Research, 64(2), 278-285.