Ingeniería Mecánica. Vol. 15. No. 3, septiembre-diciembre, 2012, p. 201-212 ISSN 1815-5944

Artículo Original

Algoritmo genético aplicado a la programación en talleres de maquinado

Genetic algorithm applied to scheduling in machine shops

José Eduardo Márquez-Delgado[I], Ricardo Lorenzo Ávila-Rondón[II], Miguel Ángel Gómez-Elvira-González[III], Carlos Rafael Herrera-Márquez[IV]

I. Universidad de Granma. Facultad de Ciencias Técnicas. Granma. Cuba
Correo electrónico: jmarquezd@udg.co.cu
II. Universidad de Holguín. Facultad de Ingeniería. Holguín. Cuba.
III. Universidad Politécnica de Madrid. Escuela Técnica Superior de Ingenieros Agrónomos. Madrid. España.
IV. Empresa de Acumuladores XX Aniversario. Manzanillo, Granma. Cuba.

Recibido: 28 de diciembre de 2011 Aceptado: 20 de julio de 2012

Resumen

En este trabajo se utiliza la metaheurística nombrada algoritmo genético, para dos variantes típicas de problemas de planificación presentes en un taller de maquinado de piezas: las variantes flujo general y flujo regular, y se ha seleccionado la minimización del tiempo de finalización de todos los trabajos o camino máximo, como objetivo a optimizar en un plan de trabajo. Este problema es considerado de difícil solución y es típico de la optimización combinatoria. Los resultados demuestran la calidad de las soluciones encontradas en correspondencia con el tiempo de cómputo empleado, al ser comparados con problemas clásicos reportados por otros autores. La representación propuesta de cada cromosoma genera el universo completo de soluciones factibles, donde es posible encontrar valores óptimos globales de solución y cumple con las restricciones del problema.

Palabras claves: algoritmo genético, cromosomas, flujo general, flujo regular, planificación, camino máximo.

Abstract

In this paper we use the metaheuristic named genetic algorithm, for two typical variants of problems of scheduling present in a in a machine shop parts: the variant job shop and flow shop, and the minimization of the time of finalization of all the works has been selected, good known as makespan, as objective to optimize in a work schedule. This problem is considered to be a difficult solution and is typical in combinatory optimization. The results demonstrate the quality of the solutions found in correspondence with the time of used computation, when being compared with classic problems reported by other authors. The proposed representation of each chromosome generates the complete universe of feasible solutions, where it is possible to find global good values of solution and it fulfills the restrictions of the problem.

Key words: genetic algorithm, chromosomes, flow shop, job shop, scheduling, makespan.

Introducción

La programación adecuada de trabajos en procesos de manufactura constituye un importante problema que se plantea dentro de la producción en muchas empresas. El orden en que estos son procesados, no resulta indiferente, sino que determinará algún parámetro de interés cuyos valores convendrá optimizar en la medida de lo posible. Así podrá verse afectado el coste total de ejecución de los trabajos, el tiempo necesario para concluirlos o el stock de productos en curso que será generado. Esto conduce de forma directa al problema de determinar cuál será el orden más adecuado para llevar a cabo los trabajos con vista a optimizar algunos de los anteriores parámetros u otros similares.

Debido a las limitaciones de las técnicas de optimización convencionales, en el siguiente trabajo se presenta una metaheurística basada en un Algoritmo Genético Simple *(Simple Genetic Algorithm, SAG)*, para resolver problemas de programación de tipo flujo general (*job shop Scheduling, JSS*) *y* flujo regular (*Flow shop Scheduling, FSS*), con el objetivo de minimizar el tiempo de finalización de todos los trabajos.

El desarrollo actual de las computadoras, y la aparición de nuevas técnicas de simulación y optimización heurística que aprovechan plenamente las disponibilidades de cálculo intensivo que estas proporcionan, han abierto una nueva vía para abordar los problemas de secuenciación o problemas de *scheduling* [1, 2] como también se le conocen, y han suministrado un creciente arsenal de métodos y algoritmos [3, 4, 5] cuyo uso se extiende paulatinamente al sustituir a las antiguas reglas y algoritmos usados tradicionalmente. En algunos estudios en este sentido, es posible encontrar una descripción general del problema de la programación de trabajos en el taller mecánico, comúnmente referenciado por la terminología anglosajona como *Job Shop Scheduling Problem (JSSP).* En el trabajo titulado: *"Neuronal Network Modeling and Simulation of the Scheduling"* [6], se utiliza como enfoque para su solución una red neuronal. Varios investigadores de esta temática han desarrollado múltiples algoritmos para resolver este problema, pero debido a su complejidad en instancias grandes [7, 8], no resulta posible contar con un método totalmente determinista para su solución general. De ahí, que en los últimos años se han aplicado diversas metaheurísticas, tales como: Algoritmos Genéticos [9, 10], Búsqueda Tabú [11, 12], Recocido Simulado [13], Colonia de Hormigas [14], Enjambre de Partículas [15], entre otras.

La función del s*cheduling* es la asignación de recursos limitados a tareas a lo largo del tiempo y tiene como finalidad la optimización de uno o más objetivos. En las variantes desarrolladas en esta investigación, identificadas en un taller de maquinado, se tiene en cuenta las relaciones de precedencia (ruta tecnológica en la fabricación de piezas), donde los recursos pasan a constituir los puestos de trabajo (máquinas-herramienta), los trabajos son las piezas a fabricar, y las tareas pasan a ser las operaciones que se realizan sobre los trabajos en las máquinas. En la variante flujo general *(JSS)* cada trabajo (pieza) puede seguir su propio orden tecnológico, mientras que en la variante flujo regular *(FSS)*, el orden de ejecución de las operaciones es el mismo para todos los trabajos, es decir, la misma ruta tecnológica para todas las piezas, constituyendo un caso particular. Cuando a esta última variante se le añade la disciplina de que todas las máquinas sean también visitadas en el mismo orden, al cumplir con: primer trabajo en entrar, primer trabajo en salir (*first in first out, FIFO*), entonces se convierte en el nombrado flujo permutacional (*Permutational Flow Shop Scheduling, PFSS*).

En este artículo se muestra la representación adecuada del cromosoma que deberá utilizar el algoritmo genético, basada en el cálculo apropiado de las permutaciones, necesario y suficiente para la obtención del universo total de soluciones factibles, en las cuales se encuentran los valores óptimos globales para el objetivo que se desea minimizar en este problema.

El *JSSP* es un problema de optimización catalogado como problema no polinomial completo *(NP-Hard),* pues se trata de unos de los problemas de optimización combinatoria más difíciles de resolver [16]. La complejidad de los problemas de secuenciamiento *(scheduling)* radica en la cantidad abrumadora de posibles soluciones. Aquí se presenta la solución dada a este problema utilizando una metaheurística de tipo evolutiva, como es el caso del algoritmo genético, y se optimiza la minimización del criterio del camino máximo, como medida de desempeño regular para este tipo de problema, perteneciente al campo de la investigación de operaciones. Para la comprobación de los resultados, se compara la calidad de las soluciones que se consiguen, con las obtenidas por otros autores y otros métodos empleados, tomando como ejemplo problemas clásicos que se encuentran disponibles en la Biblioteca de Investigación de Operaciones *(Operational Research Library, OR-Library)* [17].

Materiales y Métodos

Formulación del problema

El *JSSP* requiere planificar un conjunto de N trabajos $\{J_1, \dots, J_N\}$ sobre un conjunto de M máquinas $\{R_1, \dots, R_N\}$. Cada trabajo J_i consiste en una serie de operaciones $\{\theta_{i1}, \dots, \theta_{iM}\}$ que deben ser procesadas secuencialmente. Cada operación θ_{il} requiere el uso de una máquina $R_{\theta_{il}}$, tiene una duración $p_{\theta_{il}}$, y un tiempo de comienzo $st_{\theta_{il}}$, que debe ser determinado. Las restricciones de precedencia se expresan de la forma: $st_{\theta_{il}} + p_{\theta_{il}} \leq st_{\theta_{i(l+1)}}$, e indican que las operaciones de cada trabajo se ejecutarán secuencialmente. Las restricciones de capacidad son disyunciones de la forma: $st_v + p_v \leq st_w \vee st_w + p_w \leq st_v$, y expresan que una misma máquina no puede ser compartida de forma simultánea por dos operaciones.

El problema general asume ciertas hipótesis para que una planificación sea compatible con las resticciones tecnológicas, es decir, sea factible. Las hipótesis hacen que no todos los problemas de programación y secuenciación puedan ser modelados como un *job shop,* ni resueltos con las técnicas desarrolladas para este tipo de problemas, pero si constituyen una buena introducción a los conceptos incluidos en la teoría de secuenciación. En este caso estan presentes las siguientes:

- Hay solo una máquina de cada tipo, no existen varias máquinas para realizar una operación.
- Las máquinas no pueden procesar más de una operación a la vez.
- Las restricciones tecnológicas (ruta tecnológica) son conocidas e invariables.
- Cada trabajo es una entidad, y por lo tanto, no pueden procesarse dos operaciones de un mismo trabajo simultáneamente.
- No existe interrupción, es decir, cada operación una vez comenzada debe ser completada antes de que otra operación pueda hacerlo en esa misma máquina.
- Cada trabajo incluye una y solo una operación en cada máquina, por lo que todos los trabajos contienen una cantidad de operaciones no mayor al número de máquinas.
- Los tiempos de proceso son independientes de la secuencia seguida, lo que excluye tiempos de ajuste en las máquinas según la secuencia de los trabajos considerada o tiempos de transporte entre máquinas.
- Son conocidos y fijos todos los datos que intervienen: número de trabajos, número de máquinas, tiempos de proceso, entre otros.

Representación del problema. Estructura de datos utilizada

Como en todos los problemas que se enfrentan en el mundo real, para poder resolverlos se tiene que encontrar una forma de abstraerlos y poder representar sus posibles soluciones. Existen varias formas de representar el *JSSP* para su solución. Entre las más conocidas están, la representación con Grafos Disyuntivos [1, 2, 6] y la representación con Redes de Petri [18]. Para el desarrollo de este trabajo se utilizó una representación basada en grafos disyuntivos y se tuvo en cuenta, en primer lugar, que esta representación garantiza una mayor claridad de las soluciones a obtener debido a las restricciones que se tienen en el problema. Esta forma de representación ha encontrado mayor aceptación entre los investigadores de esta temática, lo cual no le quita el mérito ni la fortaleza a la representación basada en redes de Petri.

En este trabajo para la representación del *JSSP* se utilizó un grafo disyuntivo, ver figura 1. Un grafo es una pareja de conjuntos $G = (V, A)$, donde V es el conjunto finito de vértices, y A es el conjunto de aristas, este último es un conjunto de pares de la forma (u, v) tal que $u, v \in V$. En este caso se añade que $u \neq v$.

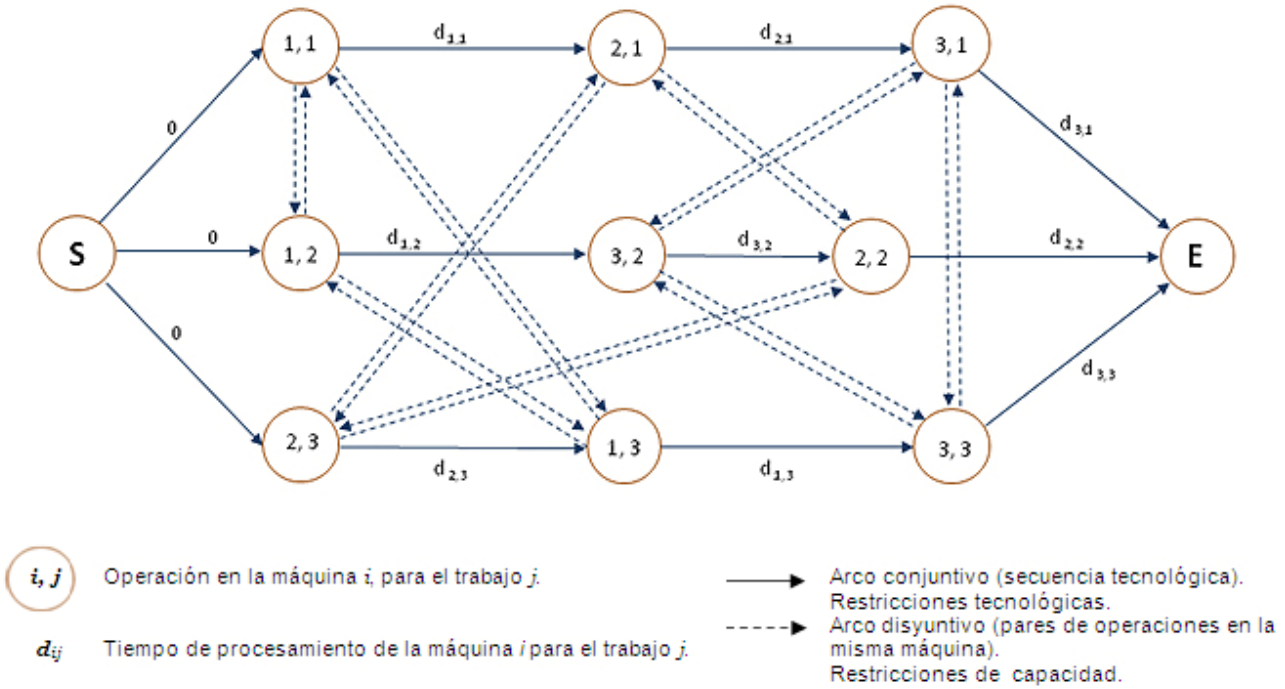

Fig. 1. Ejemplo de grafo disyuntivo para un problema de 3 trabajos en 3 máquinas (3 x 3). Variante *JSS*

Teniendo esto en cuenta se asume que: cada nodo del grafo representa la relación (máquina i, trabajo j). Existen dos nodos especiales: S y E que representan el inicio y el final de todas las operaciones. Para cada dos operaciones consecutivas en el mismo trabajo $(i,j) \in A$ existe un arco dirigido. Para cada par de trabajos que emplean la misma máquina $\{i,j\} \in V$ existen dos arcos (i,j) y (j,i) en sentidos opuestos, que indican cuál es la operación que se debe ejecutar antes. Cada nodo i tiene asociado un peso p_i que indica el tiempo que se necesita para completar la operación i. Una vez que se ha construido el grafo como se definió anteriormente y se tiene el valor de p_i como el peso de cada rama (i,j), entonces el *makespan* es igual a la longitud del camino máximo entre S y E también conocido como camino máximo o camino crítico *(critical path)*.

Un taller de fabricación o *job shop* es una configuración productiva que permite fabricar una gran cantidad de piezas diferentes. Este tipo de sistema se caracteriza por tener máquinas de propósito general, y porque cada pieza tiene una ruta de fabricación única, que no necesariamente se repite entre trabajos. Un taller de maquinado constituye un buen ejemplo de una estructura *job shop*, en el cual sus máquinas, tales como: tornos, fresadoras, taladradoras, rectificadoras, entre otras, pueden procesar una gama casi infinita de tales piezas. En estos talleres cada pieza puede tener una ruta diferente dentro del sistema; es decir, alguna orden irá primero al torno para luego pasar por la fresadora y por último a la taladradora, mientras que otra orden puede iniciar su proceso en la fresadora, continuar en el torno y terminar en la taladradora.

El objetivo perseguido en este problema es encontrar alguna planificación factible que optimice alguna medida de desempeño. Normalmente el más usado en la literatura es el de minimizar el *makespan* o $C_{máx}$ (1). Este objetivo es equivalente a minimizar los tiempos muertos, o a maximizar la utilización de las máquinas, y ésta es tal vez la razón por la cual ha sido abordado con mayor frecuencia por los investigadores.

Esta versión del problema es conocida en la literatura como $J||C_{máx}$ y se obtiene de la forma:

$$C_{máx} = máx_{j\in\{1\ldots n\}} C_j \qquad (1)$$

Espacio de búsqueda de soluciones

En esta investigación se realizó un estudio basado en tres formas distintas de obtener secuencias para problemas de naturaleza combinatoria [19]. Estas son:

1- Permutación de trabajos: las secuencias se obtienen solo por permutación de los trabajos N por lo tanto representan un orden fijo en que serán visitadas todas las máquinas. Este cálculo permite obtener el total de secuencias válidas para el caso flujo permutacional (*PFSS*), dentro de la variante de flujo regular (*FSS*). Las secuencias se obtienen de la expresión:

$$N! \tag{2}$$

2- Permutación de operaciones (partición) sin repetición: las secuencias se pueden obtener diferentes para cada máquina por permutación de las operaciones, las cuales no se pueden repetir para un mismo trabajo. Este cálculo permite obtener el total de secuencias válidas para la variante *FSS*, en la cual, aunque todos los trabajos presentan un mismo orden, cada máquina puede experimentar un orden distinto en la visita de los trabajos. Las secuencias se obtienen de la expresión:

$$(N!)^M \tag{3}$$

3- Permutación de operaciones (partición) con repetición: las secuencias se pueden obtener diferentes para cada máquina por permutación de las operaciones, las cuales se pueden repetir para un mismo trabajo. Este cálculo permite obtener el total de secuencias válidas para la variante *JSS*, en la cual los trabajos tienen distinto orden y por lo tanto visitan desigual las máquinas. Las secuencias se obtienen de la expresión:

$$\frac{(N*M)!}{(M!)^N} \tag{4}$$

La siguiente tabla 1 muestra algunos ejemplos del tamaño del espacio de búsqueda de soluciones factibles, en dependencia de las formas de cálculo anteriores, para obtener las secuencias a partir del número de trabajos y del número de máquinas.

Tabla 1. Ejemplos de dimensiones de problema y el tamaño del espacio de búsqueda de soluciones factibles correspondiente

N	M	Permutación de trabajos $N!$	Permutación de operaciones (partición)	
			Sin repetición $(N!)^M$	Con repetición $\frac{(N*M)!}{(M!)^N}$
3	3	6	216	1 680
4	4	24	331 776	63 063 000
5	5	120	24 883 200 000	623 360 743 125 120
6	6	720	139 314 069 504 000 000	2 670 177 736 637 149 247 308 800
...	...	...	...	...

Como se puede observar en la medida en que aumenta el número de trabajos y el número de máquinas, la dimensión del espacio de búsqueda de soluciones se incrementa y produce un aumento en la complejidad del problema.

A continuación, en la tabla 2, se muestra una porción del desarrollo de las secuencias para un problema de 3 trabajos en 3 máquinas. En la representación han sido utilizadas letras como símbolos, y las columnas con permutación de operaciones, no muestran su desarrollo de forma completa debido al espacio que ocuparían.

Tabla 2. Secuencias posibles del espacio total de búsqueda de soluciones factibles correspondiente a un problema de 3 trabajos (N) en 3 máquinas (M)

Permutación de trabajos $N!$	**Permutación de operaciones (partición)**	
	Sin repetición $(N!)^M$	**Con repetición** $\frac{(N*M)!}{(M!)^N}$
ABC	ABC-ABC-ABC	AAA-BBB-CCC
ACB	ABC-ABC-ACB	AAA-BBC-BCC
BAC	ABC-ABC-BAC	AAA-BBC-CBC
BCA	ABC-ABC-BCA	AAA-BBC-CCB
CAB	ABC-ABC-CAB	AAA-BCB-BCC
CBA	ABC-ABC-CBA	AAA-BCB-CBC
	ABC-ACB-ABC	AAA-BCB-CCB
	ABC-ACB-ACB	AAA-BCC-BBC
	ABC-ACB-BAC	AAA-BCC-BCB
	ABC-ACB-BCA	AAA-BCC-CBB
	ABC-ACB-CAB	AAA-CBB-BCC
	ABC-ACB-CBA	AAA-CBB-CBC
	...	...

Algoritmos Genéticos para solucionar problemas de Scheduling

Los algoritmos genéticos imitan el procedimiento de la selección natural sobre el espacio de soluciones del problema considerado, generalmente usados en problemas de búsqueda y optimización de parámetros. Se basan en la creación de generaciones sucesivas de individuos representativos de posibles soluciones al problema. La codificación de una solución se interpreta como el cromosoma del individuo compuesto de un cierto número de genes a los que les corresponden ciertos alelos. La codificación más común de los cromosomas que conforman las soluciones es a través de cadenas binarias, aunque se han utilizado también números reales y letras. Una función de aptitud *(fitness function)* debe ser diseñada para cada problema de manera específica. Una característica que esta debe tener es que debe ser capaz de "castigar" a las malas soluciones y de "premiar" a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez. La selección de padres se efectúa al azar usando un procedimiento que favorezca a los individuos mejor adaptados, ya que a cada individuo se le asigna una probabilidad de ser seleccionado que es proporcional a su función de aptitud. Se consideran dos operaciones básicas: el cruzamiento y la mutación, a partir de las cuales se obtienen nuevos individuos.

En la literatura existen numerosas propuestas que resuelven el *Job Shop Scheduling Problem* mediante el uso de algoritmos evolutivos, entre las cuales se cuenta los algoritmos genéticos [20, 21], con distintas estrategias de evolución, esquemas de codificación y diseño de operadores genéticos eficientes de cruce y mutación que producen cromosomas factibles.

Los algoritmos genéticos constituyen un método muy prometedor ya que se pueden combinar fácilmente con otras técnicas tales como la búsqueda tabú [11], vecindad variable [22], entre otras. Además, permiten explotar cualquier clase de conocimiento heurístico sobre el dominio del problema. De este modo son realmente competitivos con los mejores métodos para resolver problemas de *scheduling.*

En el presente artículo ha sido seleccionada esta metaheurística cómo método de solución, ya que se trata de una técnica robusta, y si bien no se garantiza encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de los algoritmos de optimización combinatoria. Además, son intrínsecamente paralelos, independientemente de si lo hayamos implementado de esa forma o no, ya que buscan en distintos puntos del espacio de soluciones de forma paralela, cualidad que permite que se ejecuten simultáneamente en varios procesadores.

Estructura utilizada para construir el cromosoma

Varias han sido las formas de codificar las soluciones para el problema de este tipo con el uso de algoritmos genéticos. En la siguiente tabla 3 se hace una clasificación de la bibliografía [23] donde se tiene en cuenta algunos tipos de representación que se han empleado.

Tabla 3. Algunas formas de representación según la estructura del cromosoma utilizada.

Representación	Autor
Binaria basada en pares de trabajo	Nakano y Yamada (1991)
Secuencia de operaciones con particiones basadas en listas de preferencias	Falkenauer y Bouffouix (1991), Croce et al. (1995)
Secuencia de operaciones con particiones y operadores especializados	Yamada y Nakano (1992)
Secuencia de trabajos	Holsapple et al. (1993)
Secuencia de operaciones sin particiones	Fang et al. (1993), Bierwirth (1995)
Secuencia de operaciones basadas en números aleatorios	Bean (1994)
Basadas en reglas de prioridad	Dorndorf y Pesch (1995)

Para el desarrollo de esta investigación se utilizó la secuencia de trabajos según el caso particular *PFSS*, debido a que los datos de los problemas resueltos, permiten aplicar la discplina *FIFO* y resulta suficiente para generar el universo completo de soluciones factibles donde encontrar valores óptimos de planificaciones, para el objetivo propuesto de minimizar el camino máximo. La longitud del cromosoma queda determinada por la cantidad de trabajos *N*, por lo tanto utiliza menos memoria para su almacenamiento. Adicionalmente, esta forma directa de representar el cromosoma aprovecha el conocimiento del problema a modelar lo que se traduce en menos intentos por violentar el orden del proceso o ruta tecnológica de fabricación de las piezas.

La figura 2 muestra un cromosoma conformado por una secuencia de caracteres que representa los distintos trabajos y donde coinciden genes y alelos. Con esta representación se le dio solución a problemas de 20 trabajos en 5 máquinas (20 x 5), los cuales cumplen de la forma particular *PFSS,* cuyos resultados se ofrecen más adelante en la sección de “resultados”.

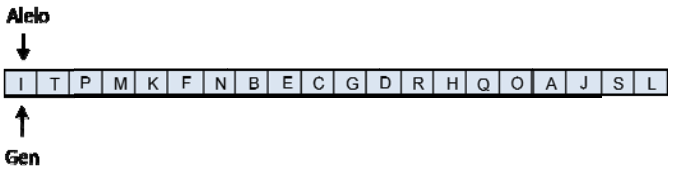

Fig. 2. Representación de un cromosoma de 20 trabajos para un problema del caso particular *PFSS*

Para la correcta modelación de la variante *JSS*, en el presente artículo se propone una representación basada en una secuencia con partición con repetición (permutación con repetición), que tiene en cuenta no solo el número de trabajos *N*, sino también el número de máquinas *M*. Cada partición o subsecuencia es equivalente al número de máquinas (Fig. 3) y está constituida por los elementos individuales (alelos) que conforman el cromosoma y contiene el orden en que son ejecutadas las operaciones de cada trabajo, de modo que cada plan *(schedule)* pueda ser representado por un único cromosoma. En cada una de ellas es posible la ubicación consecutiva de alelos (representados por caracteres), lo cual significa que se realizarían las operaciones consecutivas para un mismo trabajo. Hay que considerar, que esta forma de representar las planificaciones supone un mayor número de soluciones factibles dentro del universo total en comparación con la variante *FSS* y su caso particular *PFSS*, pero garantiza la posibilidad de encontrar el óptimo global, al poder ser representadas todas las soluciones posibles del problema. La longitud del cromosoma queda determinada por el producto del número de trabajos por el número de máquinas (NxM).

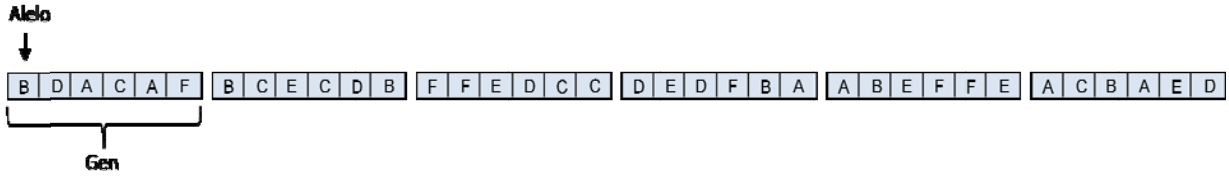

Fig. 3. Representación de un cromosoma para un problema tipo *job shop* de 6 trabajos en 6 máquinas

En esta representación se tienen en cuenta que quede asegurado el orden o ruta tecnológica de cada trabajo, restricción importante que para este tipo de problema, nunca se puede violar. Aunque es permitido la repetición de operaciones de forma consecutiva, se conservan los datos del trabajo, máquina, orden y tiempo individual, cada vez que se construye un cromosóma válido.

En ambas representaciones se tuvo en cuenta la generación de soluciones no factibles una vez que se efectúa el método de cruzamiento seleccionado. Así, en el cromosoma para el caso particular *PFSS* (Fig. 2), solamente se verifica que no exista un trabajo repetido, mientras que en el cromosoma para la variante *JSS* (Fig. 3), se comprueba que exista la misma cantidad de operaciones (representadas por los alelos) que máquinas (representadas por los genes).

Resultados

Dado que los algoritmos genéticos son un mecanismo de carácter estocástico y no exacto, su validez como método de búsqueda de soluciones debe ser realizada de forma experimental. En general se deben evaluar no solamente la eficacia y la eficiencia, como en cualquier otro método de búsqueda, sino también la estabilidad por tratarse de un método de naturaleza estocástica. En el caso de los problemas de *scheduling* existen bancos de problemas *(benchmark problems),* -disponibles en [17], de uso común entre los investigadores-, lo cual facilita la comparación entre distintos métodos de resolución.

La tabla 4 muestra una serie de resultados obtenidos en un estudio experimental para el cual se escogió un conjunto de casos (instancias de problemas) propuestos por su autor (Eric Taillard), conocidos como problemas Taillard [17], que sirven para comparar los resultados obtenidos con la representación y la solución ofrecida al problema en la presente investigación. Particularmente se seleccionaron 10 problemas de la variante flujo regular conformados por 20 trabajos en 5 máquinas (20 x 5). La primera columna ofrece el nombre dado a la instancia del problema. Las siguientes columnas indican el tamaño: número de trabajos (*N*) (número de máquinas (*M*). Luego los valores de las cotas inferior y superior. La última columna representa el mejor valor encontrado por nuestra propuesta.

Como se puede apreciar en todos los casos se alcanzó el límite o cota superior, valores que han sido obtenidos por otros métodos, habitualmente mediante la relajación de restricciones en un problema, y que son fundamentales tener en cuenta para la eficiencia de muchos algoritmos de búsqueda, principalmente cuando se desconoce el valor óptimo del objetivo que se persigue, en este caso la minimización del camino máximo en un plan de trabajo.

Tabla 4. Resultados obtenidos para problemas de la variante flujo regular. Las instancias corresponden a Eric Taillard. *(Taillard's benchmark problems).*

Instancia	N	M	Cota inferior	Cota superior	Algoritmo Genético
ta001	20	5	1232	1278	1278
ta002	20	5	1290	1359	1359
ta003	20	5	1073	1081	1081
ta004	20	5	1268	1293	1293
ta005	20	5	1198	1236	1235
ta006	20	5	1180	1195	1195
ta007	20	5	1226	1239	1239
ta008	20	5	1170	1206	1206
ta009	20	5	1206	1230	1230
ta010	20	5	1082	1108	1108

La figura 4 muestra un gráfico de Gantt con la solución obtenida para la primera instancia (ta001). Para una mejor comprensión se le ha asignado un color direferente a cada trabajo. Un mismo color significa las distintas tareas de un trabajo en cada máquina visitada.

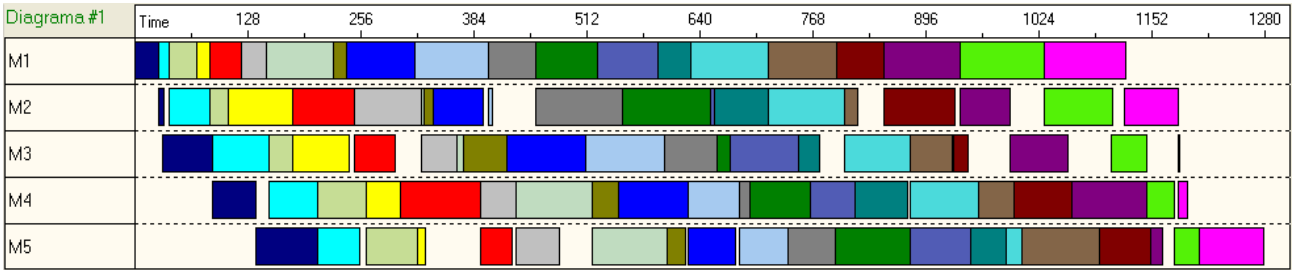

Fig. 4. Representación en forma de gráfico de Gantt de un plan de trabajo tipo *flow shop*. Corresponde a la solución de la instancia de problema ta001. Valor óptimo de $C_{máx} = 1278$.

Para las diferentes ejecuciones realizadas para el caso particular *PFSS*, el algoritmo genético fue regulado con los siguiente parámetros:

- tamaño de población: 100
- número de generaciones: 1000
- método de selección: ranking
- método de cruzamiento: 1 punto de cruce
- factor de cruzamiento: 0.6
- factor de mutación: 0.02
- elitismo: habilitado

Los resultados experimentales para problemas de la variante *Job Shop Scheduling,* han sido avalados, mediante la solución a problemas propuestos por otros investigadores. Un ejemplo lo constituye el conjunto (40 en total) de instancias de problemas propuestos por S. Lawrence [17], de las cuales se muestran los resultados de las primeras 20 en la siguiente tabla 5.

La primera columna ofrece el nombre dado a cada instancia. Las siguientes columnas indican el tamaño: número de trabajos (*N*), número de máquinas (*M*). Las demás corresponden a resultados obtenidos por otros investigadores, reportado en [19]. Por ese orden, T. Yamada, D. Applegate, P. van Laarhoven, H. Matsuo, J. Adams y P. Brucker. La última columna ha sido adicionada con el propósito de mostrar los resultados alcanzados por nuestra propuesta. Solo en la instancia 20 no se consiguió obtener el valor óptimo del camino máximo. El asterisco simple indica el valor óptimo o mejor valor mínimo conocido. Como se puede apreciar, en algunas instancias algunos investigadores no alcanzaron el valor óptimo correspondiente.

Tabla 5. Resultados obtenidos para problemas *Job Shop*. Las instancias corresponden a S. Lawrence. *(Lawrence's benchmark problems)*

Instancia	N	M	Yamada	Appl	Laar	Matsuo	Adams	Bruck	Algoritmo Genético
la01	10	5	*666	*666	*666	---	*666	*666	*666
la02	10	5	*655	*655	*655	*655	669	*655	*655
la03	10	5	*597	*597	606	*597	605	*597	*597
la04	10	5	*590	*590	*590	*590	593	*590	*590
la05	10	5	*593	*593	*593	---	*593	*593	*593
la06	15	5	*926	*926	*926	---	*926	*926	*926
la07	15	5	*890	*890	*890	---	*890	*890	*890
la08	15	5	*863	*863	*863	*863	*863	*863	*863
la09	15	5	*951	*951	*951	---	*951	*951	*951
la10	15	5	*958	*958	*958	---	*958	*958	*958
la11	20	5	*1222	*1222	*1222	---	*1222	*1222	*1222
la12	20	5	*1039	*1039	*1039	---	1239	*1039	*1039

la13	20	5	*1150	*1150	*1150	---	*1150	*1150	*1150
la14	20	5	*1292	*1292	*1292	---	*1292	*1292	*1292
la15	20	5	*1207	*1207	**1207	---	*1207	*1207	*1207
la16	10	10	*945	*945	956	959	978	*945	*945
la17	10	10	*784	*784	*784	*784	787	*784	*784
la18	10	10	*848	*848	861	*848	859	*848	*848
la19	10	10	*842	*842	848	*842	860	*842	*842
la20	10	10	907	*902	*902	907	914	*902	907

En la figura 5 se muestra en forma de gráfico de Gantt una de las soluciones obtenidas derivada de la investigación realizada. Cada trabajo tien un color direferente. Un mismo color significa las distintas tareas de un trabajo en cada máquina visitada. Constituye un típico problema *JSS* en el cual cada trabajo tiene un orden (ruta tecnológica) diferente.

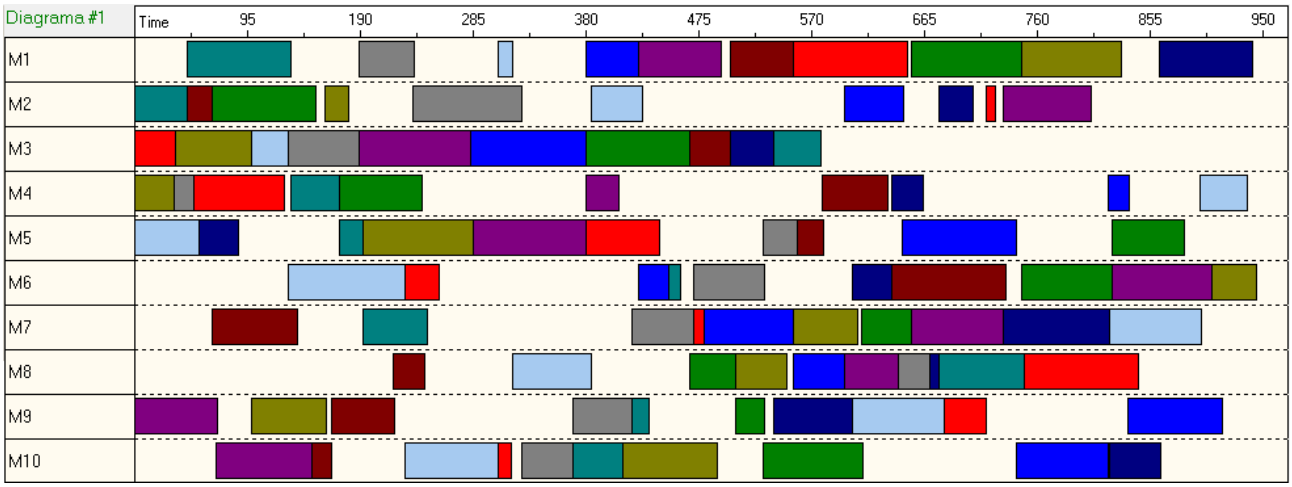

Fig. 5. Representación en forma de gráfico de Gantt de un plan de trabajo tipo *job shop*. Corresponde a la solución de la instancia de problema la16. Valor óptimo de $C_{máx} = 945$.

En las diferentes ejecuciones realizadas para la variante *JSS*, que utiliza un cromosoma de dimensión ($N \times M$), el algoritmo genético fue regulado con los siguiente parámetros:

- tamaño de población: 250
- número de generaciones: 5000
- método de selección: ranking
- método de cruzamiento: 2 puntos de cruce
- factor de cruzamiento: 0.8
- factor de mutación: 0.01
- elitismo: habilitado

No forma parte de estos resultados un estudio estadístico realizado, con el objetivo de revelar el mejor rendimiento alcanzado con esta metaheurística para distintas instancias de problema en cada variante.

Discusión

Los resultados obtenidos demuestran la acertada representación realizada de la estructura del cromosoma para cada variante de problema estudiado. Es así que, resulta importante reconocer frente a que variante de problema de secuenciamiento o scheduling nos encontramos como tecnólogos, al realizar la programación de las máquinas en un taller de maquinado, para asignarle la representación adecuada y no incurrir por un lado, en la generación de una excesiva dimensión del espacio total de soluciones factibles, donde buscar valores óptimos al objetivo u objetivos planteados, y por otro lado, no reducir esta dimensión, lo cual privaría de representar el conjunto de soluciones factibles en las cuales encontrar dichos valores óptimos. Hay que tener en cuenta que este es un problema, que genera una explosión combinatoria y todo intento por resolverlo de manera óptima, debe partir en primer lugar, de una adecuada representación, de acuerdo con el método de

solución empleado. En este caso, la representación cromosómica es de vital importancia para el funcionamiento eficiente del algoritmo genético.

Conclusiones

1- En un taller de maquinado están presentes las variantes *JSS* y *FSS*, así como el caso particular *PFSS*. Su identificación correcta fue determinante para la posterior aplicación de algoritmos genéticos, metaheurística utilizada en esta investigación.

2- El cálculo aplicado a la forma de obtener las secuencias para cada variante resultó correcto, y fue la base para la representación de la estructura y dimensión del cromosoma utilizado con el algoritmo genético en cada variante, lo cual a su vez permitió cumplir con las resctricciones del problema y generar el universo completo de soluciones factibles, donde fue posible encontrar luego valores óptimos a diversas instancias de problema.

3- La representación del problema mediante un grafo disyuntivo dirigido, resultó apropiada por ser una estructura de datos con los elementos suficientes para darle solución a las variantes del problema, identificadas en un taller de maquinado.

4- La aplicación de algoritmos genéticos demostró su efectividad como método de búsqueda de soluciones en problemas de naturaleza combinatoria, como es el caso del *scheduling*.

5- Se obtuvieron resultados comparables con los reportados por otros autores, incluso valores óptimos, para el objetivo de minimizar el camino máximo*,* con la utilización de instancias de problema clásicos, lo cual permite avalar la calidad de las soluciones obtenidas.

Referencias

1. Pinedo M. L. *Scheduling. Theory, Algorithms, and Systems*. Third edition. New York: Springer, 2008, 671 p. ISBN 978-0-387-78934-7.
2. Blazewicz, J., K. H. Ecker, *et al*. *Handbook on Scheduling from Theory to applications*. Berlin: Springer-Verlag. 2007. p. 57-70, 647 p. ISBN 978-3-540-28046-0.
3. Carlier, J. y Pinson, E. "An Algorithm for solving the job-shop problem." *Management Science.* 1989*.* vol. 35, nº. 2: p. 164-176. ISSN 0025-1909.
4. Brucker P., Jurisch, B. y Sievers, B. “A branch and bound algorithm for the Job-Shop Scheduling Problems”. *Discrete Applied Mathematics*, 1994, vol. 49, p. 107-127. ISSN 0166-218X.
5. Wenqi, H. y Aihua, Y. "An improved shifting bottleneck procedure for the job shop scheduling problem." *Computers & Operations Research*. 2004, vol. 31, p. 2093-2110. ISSN 0305-0548.
6. Ávila Rondón, R. y da Silva Carbalho, L. A. y Infantes Hernández, G. “Neural Network Modeling and Simulation of the Scheduling”. *Innovation in Manufacturing Networks*. 2008. New York: Springer ISBN 978-0-387-09491-5.
7. Brucker, P. *Scheduling Algorithms*. Fifth edition. Berlin: Springer-Verlag, 2008, p. 37-60, 371 p. ISBN 978-3-540-69515-8.
8. Applegate, D. y Cook, W. "A computational study of the job-shop scheduling problem." *ORSA Journal on computing*. 1991. vol. 3, nº. 2, p. 149-156. ISSN 0899-1499.
9. Lestan, Z., Brezocnik, M. *et al.* "Solving the Job-Shop Scheduling Problem with a Simple Genetic Algorithm." *Int J Simul Model*. 2009. vol. 8, nº. 4, p 197-205. ISSN 1726-4529. DOI: 10.2507/IJSIMM08(4)2.138.
10. Watanabe, M., Ida, K. *et al.* "A genetic algorithm with modified crossover operator and search area adaptation for the job shop scheduling problem." *Computers & Industrial Engineering*. 2005, vol. 48, p. 743-752. ISSN 0360-8352. DOI 10.1016/j.cie.2004.12.008.
11. Vilcot, G. y Billaut, J.-C. "A Tabu search and genetic algorithm for solving a bicriteria general job shop scheduling problem." *European Journal of Operational Research.* 2008, vol. 190, p. 398-411. ISSN 0377-2217. DOI 10.1016/j.ejor. 2007.06.039
12. Zhang, C. Y., Li, P. *et al*. "A very fast TS/SA algorithm for the job shop scheduling problem." *Computers & Operations Research*. 2008, vol. 35, p. 282-294. ISSN 0305-0548. DOI 10.1016/j.cor.2006.02.024.
13. Zhang, R. y Wu, C. "A hybrid immune simulated annealing algorithm for the job shop scheuling problem." *Applied Soft Computing*. 2010, vol. 10, p. 79-89. ISSN 1568-4946. DOI 10.1016/j.asoc.2009.06.008.

14. Xing L., Chen Y., Wang P. *et al*. “A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems”. *Applied Soft Computing*. 2010, vol. 10, p. 888-896. ISSN 1568-4946. DOI 10.1016/j.asoc.2009.10.006.

15. Toro, E. M., Restrepo, Y. S. *et al*. "Adaptación de la técnica de Particle Swarm al problema de secuenciamiento de tareas". *Scientia et Technica*. 2006. vol. XII, nº. 32, p. 307-312. ISSN 0122-1701.

16. Blazewicz, J., Lenstra, J. K. *et al*. "Scheduling subject to resource constraints: clasification and complexity." *Discrete Applied Mathematics*. 1983. vol. 5, p. 11-24. ISSN 0166-218X.

17. Beasley, J. E. "Or-library: Distributing Test Problems by Electronic Mail." *Journal of the Operational Research Society.* 1990. vol. 41, nº. 11, p. 1069-1072. ISSN 0160-5682. [Consultado el: 15 de diciembre de 2011]. Disponible en: http://www.jstor.org/discover/10.2307/2582903?uid=3737824&uid=2129&uid=2&uid=70&uid=4&sid=21100692505181 .

18. Desrochers, A. A. y Al-Jaar, R. *Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis*. Piscataway, NJ: IEEE Press, 1995. 348 p. ISBN 0879422955.

19. Yamada, T. Studies on Metaheuristics for Jobshop and Flowshop Scheduling Problems. Kyoto, Japan: Department of Applied Mathematics and Physics. Kyoto University. 2003. 120 p. [Consultado el: 14 de mayo de 2010]. Disponible en: http://www.kecl.ntt.co.jp/as/members/yamada/YamadaThesis.pdf

20. Pan, J. C.-H. y Huang, H.-C. "A hybrid genetic algorithm for no-wait job shop scheduling problems." *Expert Systems with Applications*. 2009, vol. 36, p. 5800-5806. ISSN 0957-4174. DOI 10.1016/j.eswa.2008. 07.005.

21. Kuczapski, A. M., Micea, M. V. *et al*. "Eficient generation of near optimal initial populations to enhance Genetic Algorithms for Job-Shop Scheduluing." *Information Technology and Control.* 2010. vol. 39, nº. 1 p. 32-37. ISSN 1392-124X.

22. Gao, J., Sun, L. *et al.* "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems." *Computers & Operations Research*. 2008, vol. 35, p. 2892-2907. ISSN 0305-0548. DOI 10.1016/j.cor.2007.01.001.

23. López, S., Sánchez P., *et al*. "Secuenciación mediante metaheurísticos". En: *VIII Congreso de Ingeniería de organización*, Leganés, 9 y 10 de septiembre de 2004. [Consultado el: 7 de septiembre de 2009]. Disponible en: http://www.iit.upcomillas.es/docs/IIT-04-024A.pdf ISBN 84-688-7879-0.