Original

PROGRAMACIÓN DE LA PRODUCCIÓN EN UN AMBIENTE FLOW SHOP CON LA APLICACIÓN DE ALGORITMOS GENÉTICOS

Programming of the production under flow shop enviroment with the aplication of Genetic Algorithms

Lic. Celia Maliuska García-Pérez, Universidad de Granma. cgarciap@udg.co.cu , Cuba.
Dr. C. José Eduardo Márquez-Delgado, Universidad de Granma, cgarciap@udg.co.cu, Cuba.
MSc. Carlos Rafael Herrera-Márquez, Empresa de Acumuladores "XX Aniversario", cgarciap@udg.co.cu, Cuba.

Recibido: 19/05/2017 – Aceptado: 09/06/2017

RESUMEN

En este trabajo se estudia la programación de la producción en un ambiente *flow shop* que tiene lugar en talleres de mecanizado de piezas. Para dar respuesta a un problema que pertenece al área de planificación, hemos acudido al empleo de la metaheurística Algoritmos Genéticos, como método de solución que ofrece la inteligencia artificial. En la programación de tareas se precisa encontrar un orden adecuado de ejecución que minimice el tiempo total de trabajo de las máquinas o *makespan*. Este problema, considerado de difícil solución, es típico de la optimización combinatoria y se presenta en talleres con tecnología de mecanizado, donde existen máquinas-herramientas convencionales y se fabrican diferentes tipos de piezas que tienen en común una misma ruta tecnológica (orden del proceso). Se resolvieron diversos problemas, considerando distintos tamaños de lista de trabajos y máquinas y se desarrollaron cálculos que permiten conocer el valor del *makespan*. Se probó con problemas clásicos publicados por otros autores, obteniéndose resultados satisfactorios en cuanto a la calidad de las soluciones encontradas y el tiempo de cómputo.

Palabras clave: Algoritmos Genéticos, *flow shop, makespan*, programación de la producción.

ABSTRACT

In this work flow studies the programming of the production in an environment itself shop that has place in workshops of mechanization of pieces. In order to give answer to a problem that belongs to the area of planning, we have attended the job of her meta-heuristic Genetic

Algorithms, like method of solution that offers artificial intelligence. In the programming of tasks he needs to find him an order made suitable of execution that minimizes total time of work of the machines or makespan. This problem, considered of difficult solution, is typical of the combinatorial optimization and he shows up in workshops with technology of mechanization, where there are conventional machine tools and they fabricate different kinds of pieces that have in common a same technological route (order of the process).

Various problems, considering several sizes of list of works and machines got worked out and calculations that they enable developed to know the makespan's value themselves. Proofs with classical problems published by other authors came true, and we got out satisfactory results as to quality of the found solutions and the computer time.

Key words: Genetic Algorithms, flow shop, makespan, programming of the production.

INTRODUCCIÓN

El desarrollo actual de los ordenadores, y la aparición de nuevas técnicas de simulación y optimización heurística que aprovechan plenamente las disponibilidades de cálculo intensivo que estos 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*). Varios investigadores de esta temática han desarrollado múltiples algoritmos para resolver este problema, debido a su complejidad en instancias grandes [6], [7], 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 (AG) [8], [9], Búsqueda Tabú [10], [11], Recocido Simulado [12], Colonia de Hormigas [13], Enjambre de Partículas [14], entre otras.

Los problemas de *scheduling* en la práctica poseen estructuras más complejas, aunque en situaciones reales pueden ser relevantes diferentes restricciones. Existen variantes o casos especiales de gran interés dentro de los problemas de planificación general (*General Shop Scheduling, GSS*) estos son: el *Flow Shop Scheduling* (FSS), el *Job Shop Scheduling* (*JSS*) y el *Open Shop Scheduling* (*OSS*).

Los dos primeros tienen en común la existencia de relaciones de precedencia en las tareas u operaciones (ruta tecnológica en la construcción de piezas) de los trabajos, mientras que en el último no existen relaciones de precedencias entre las operaciones, lo cual hace que el orden de ejecución sea indiferente.

El *FSS* constituye un caso particular, en el cual el orden de ejecución de las operaciones es el mismo para todos los trabajos, no siendo así en el caso del *JSS*, donde cada trabajo puede seguir su propio orden. En este trabajo se resuelve la primera variante, ya que se pone de manifiesto en talleres con tecnología de mecanizado tanto para la fabricación de nuevos productos, como para la fabricación de piezas de repuestos.

Materiales y métodos

Complejidad del problema

El problema de asignar cargas de trabajo a máquinas se cataloga como "problema no polinomial completo" (*NP-Hard*), pues se trata de unos de los problemas de optimización combinatoria más difíciles de resolver [15]. Una característica común a la mayoría de los problemas estudiados por la optimización combinatoria, es que suelen ser relativamente "fáciles" de plantear pero mucho más difíciles de modelar y, consecuentemente, mucho más difíciles de resolver.

La complejidad del problema de secuenciar trabajos del tipo *job shop* radica en la cantidad abrumadora de posibles soluciones. Debido a que las operaciones a ser procesadas en una máquina forman la secuencia de operaciones para esa máquina, el plan de trabajo está formado por n secuencias de operaciones para cada máquina.

Puesto que cada secuencia de operaciones puede ser permutada independientemente de la de otra máquina, el número total de posibles soluciones para el *JSSP* es $n!^{m}$, donde n denota el número de trabajos y m el número de máquinas. Este problema no solo es del tipo *NP-Hard*, sino que de entre los que pertenecen a esta tipología, es uno de los más difíciles de resolver, de forma que no hay hasta el momento algoritmos determinísticos que lo resuelvan en forma eficiente (polinomial). Problemas de solo 10 trabajos y 10 máquinas han podido resolverse solo después de un período de 25 años [16].

1. 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 a resolver asume las siguientes restricciones:

- Hay solo una máquina de cada tipo, no existiendo varias máquinas para realizar una misma operación.
- Una máquina puede procesar solamente un trabajo en un solo instante.
- 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 sola 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.

1.1 *Representación del problema*

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 [17] y la representación con Redes de Petri. Para el desarrollo de este trabajo se utilizó una representación basada en un grafo disyuntivo dirigido, teniendo en cuenta 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.

En este trabajo para la representación del *JSSP* se utilizó un grafo disyuntivo. 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 $u \neq v$.

El objetivo perseguido en este problema es encontrar alguna planificación factible que optimice el camino máximo, siendo este el más usado en la literatura, es decir, la minimización del *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.

Makespan: es el tiempo mínimo para completar todos los trabajos. Esta versión del problema es conocida en la literatura como $J||C_{máx}$ y se obtiene de la forma:

$$C_{máx} = máx_{j \in 1...n} C_j \quad (1)$$

1.2 Solución con Algoritmos Genéticos

Los algoritmos genéticos son métodos adaptativos, generalmente utilizados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual y en el principio de supervivencia del más apto. Golberg [18], los define como algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural, que combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado.

Para alcanzar la solución a un problema se parte de un conjunto inicial de individuos, llamado población, generado de una manera aleatoria. Cada uno de estos individuos representa una posible solución al problema. A grandes rasgos un AG consiste de una población de soluciones codificadas de forma similar a cromosomas. Cada uno de estos cromosomas tendrá asociado un ajuste (*fitness*), valor de bondad, que cuantifica su validez como solución al problema. En función de este valor se le darán más o menos oportunidades de reproducción. Además, con cierta probabilidad se realizarán mutaciones a estos cromosomas. Estos algoritmos han demostrado ser un método global y robusto de búsqueda de soluciones de problemas, unido a esto, la aplicación más común de estos ha sido la solución de problemas de optimización donde se ha comprobado que son muy eficientes y confiables.

Si bien no se garantiza que el AG 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.

Varias han sido las formas de codificar las soluciones para el problema de la secuenciación de trabajos usando Algoritmos Genéticos. Para el desarrollo de esta investigación se utilizó la secuencia de operaciones sin partición (secuencia de trabajos) por resultar suficiente para obtener valores óptimos de planificaciones para el objetivo de minimizar el *makespan*, además por ser la representación que más aprovecha el conocimiento del problema a modelar, lo que se traduce en la no ocurrencia de intentos por violentar el orden del proceso o ruta tecnológica de la fabricación. Por último, es la que utiliza menos memoria para almacenar un cromosoma, quedando determinada la longitud de este por la cantidad de trabajos n .

La siguiente figura (Figura 1), muestra un cromosoma conformado por una secuencia de caracteres que representa los distintos trabajos (piezas), y donde coinciden genes y alelos. Con esta representación se le dio solución una serie de instancias de problemas (desde 20 trabajos en 5 máquinas hasta 500 en 20) de tipo *FSS*. Una parte de los resultados se ofrecen en la sección de “resultados y discusión” de este trabajo.

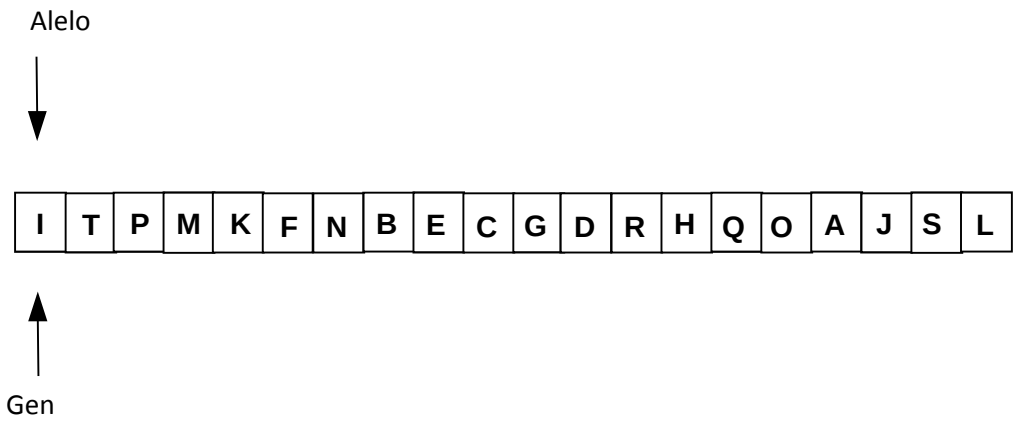

Figura 1. Representación de un cromosoma para un problema tipo *flow shop* de 20 trabajos.

Resultados y Discusión

Debido a 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 ejemplos (*benchmark problems*) de uso común entre los investigadores, lo cual facilita la comparación de distintos métodos de resolución.

La tabla 1 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 [19], que sirven para comparar los resultados obtenidos con la solución ofrecida al problema en este trabajo. Particularmente, se seleccionaron las 20 primeras instancias de problemas conformados por 20 trabajos en 5 máquinas (20 x 5). Como se puede apreciar, en algunos casos (destacados en negritas) se alcanzó o se obtuvo un valor inferior al límite o cota superior (*upper bound*). La figura 2 muestra un gráfico de Gantt con la representación de la solución obtenida para la instancia (ta005).

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

- tamaño de población: 100
- número de generaciones (iteraciones): 500
- método de selección: ranking
- método de cruzamiento: 1 punto de cruce
- factor de cruzamiento: 0.7
- factor de mutación: 0.01
- elitismo: habilitado

Instancias de problemas	Tamaño N x M	Resultados de Taillard		Resultados obtenidos con AGS
		Lower bound	Upper bound	
ta001	20 x 5	1232	**1278**	1278
ta002	20 x 5	1290	**1359**	1359
ta003	20 x 5	1073	**1081**	1081
ta004	20 x 5	1268	**1293**	1293
ta005	20 x 5	1198	**1236**	1235
ta006	20 x 5	1180	**1195**	1195
ta007	20 x 5	1226	**1239**	1239
ta008	20 x 5	1170	**1206**	1206
ta009	20 x 5	1206	**1230**	1230

ta010	20 x 5	1082	**1108**	1108
ta011	20 x 10	1448	1582	1586
ta012	20 x 10	1479	1659	1664
ta013	20 x 10	1407	1496	1509
ta014	20 x 10	1308	1378	1380
ta015	20 x 10	1325	1419	1422
ta016	20 x 10	1290	1397	1401
ta017	20 x 10	1388	**1484**	1484
ta018	20 x 10	1363	**1538**	1401
ta019	20 x 10	1472	1593	1596
ta020	20 x 10	1356	**1591**	1591

Tabla 1. Resultados obtenidos para problemas de tipo flow shop. Las instancias corresponden a Eric Taillard. (Taillard's benchmark results).

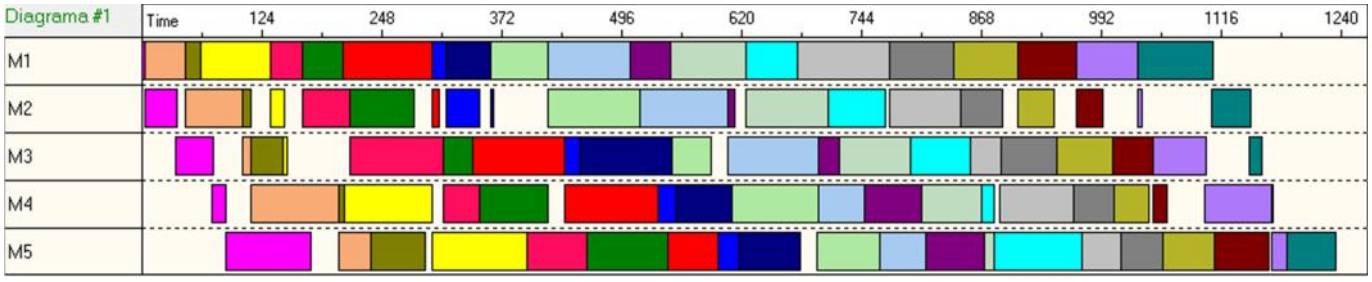

Figura 2. Representación en forma de gráfico de Gantt de un plan de trabajo tipo flow shop. Corresponde a la solución de la instancia de problema (ta005). Valor alcanzado de $C_{máx} = 1235$.

Finalmente, derivado de la investigación, se realizó una aplicación computacional, la cual ha sido utilizada para mostrar los resultados alcanzados en este trabajo. Para los experimentos y ejecuciones se utilizó un ordenador con 1024 MB de RAM y un microprocesador Intel Pentium a 3.0 GH. El código fue escrito en el lenguaje de programación C++ y se compiló para la plataforma Windows.

CONCLUSIONES

En esta investigación se le da solución al conocido problema del taller mecánico, en una de sus formas típicas de presentarse en un proceso tecnológico de maquinado, es decir, la variante *flow shop*.

Se realizó la modelación del problema para minimizar un único objetivo: el tiempo en que todos los trabajos son terminados en el proceso (*makespan*), lográndose representar adecuadamente en función de la metaheurística aplicada, en particular, Algoritmos Genéticos.

Todos los algoritmos desarrollados en general tiene un buen desempeño, y la solución propuesta muestra resultados de alta calidad, y se adapta apropiadamente al tipo de problema resuelto.

REFERENCIAS BIBLIOGÁFICAS

1. Pinedo M. L. (2008). Scheduling. Theory, Algorithms, and Systems. Third edition. New York Springer, pp. 151-172. ISBN: 978-0-387-78934-7.
2. Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G. y Wewlarz J. (2007). “Handbook on Scheduling from Theory to applications”. Berlin, Springer-Verlag,. pp. 57-70. ISBN: 978-3-540-28046-0.
3. Carlier, J. and Pinson, E. (1989). "An Algorithm for solving the job-shop problem." Management Science. vol. 35(2), pp. 164-176. ISSN: 0025-1909.
4. Brucker P., Jurisch B, Sievers B. (1994). “A branch and bound algorithm for the Job-Shop Scheduling Problems”. Discrete Applied Mathematics, vol. 49, pp. 107-127, ISSN: 0166-218X.
5. Wenqi, H. and. Aihua, Y (2004). "An improved shifting bottleneck procedure for the job shop scheduling problem." Computers & Operations Research, vol. 31, pp. 2093-2110,. ISSN: 0305-0548.
6. Brucker, P. (2008). “Scheduling Algorithms”. Fifth edition. Berlin, Springer-Verlag, pp. 37-60. ISBN: 978-3-540-69515-8.

7. Applegate, D. and W. Cook. (1991). "A computational study of the job-shop scheduling problem." ORSA Journal on computing, vol. 3(2), pp. 149-156,. ISSN: 0899-1499.
8. Lestan, Z., M. Brezocnik, B. Buschmeister, S. M. Brezovnik, J. Balic. (2009). "Solving the Job-Shop Scheduling Problem with a Simple Genetic Algorithm." Int J Simul Model, vol. 8(4), pp. 197-205. ISSN: 1726-4529.
9. Watanabe, M., K. Ida, G. Mitsuo. (2005). "A genetic algorithm with modified crossover operator and search area adaptation for the job shop scheduling problem." Computers & Industrial Engineering, vol. 48, pp. 743-752. ISSN: 0360-8352.
10. Vilcot, G. and J.-C. Billaut. (2008). "A Tabu search and genetic algorithm for solving a bicriteria general job shop scheduling problem." European Journal of Operational Research, vol. 190, pp. 398-411. ISSN: 0377-2217.
11. Zhang, C. Y., L. PeiGen, R. YunQing, G. ZaiLin. (2008). "A very fast TS/SA algorithm for the job shop scheduling problem." Computers & Operations Research, vol. 35, pp. 282-294. ISSN: 0305-0548.
12. Zhang, R. and C. Wu. (2010). "A hybrid immune simulated annealing algorithm for the job shop scheuling problem." Applied Soft Computing, vol. 10 pp. 79-89, ISSN: 1568-4946.
13. Xing L., Chen Y., Wang P., Zhao Q. Xiong J. (2010). “A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems”. Applied Soft Computing, vol. 10, pp. 888-896, ISSN: 1568-4946.
14. Toro, E. M., Y. S. Restrepo, M. Granada. (2006). "Adaptación de la técnica de Particle Swarm al problema de secuenciamiento de tareas." Scientia et Technica. año. XII (32), pp. 307-312, ISSN: 0122-1701.
15. Garey M. R., D. S. Johnson, R. Sethi. (1976). “Complexity of Flow Shop and Job Shop Scheduling”. Mathematics of Operations Research, vol. 1(2), pp. 117-119.