REPARACIÓN Y TECNOLOGÍA MECÁNICA
REPAIRING AND MECHANICAL TECHNOLOGICAL

ARTÍCULO ORIGINAL

Algoritmo Genético aplicado al problema de programación en procesos tecnológicos de maquinado con ambiente Flow Shop

Genetic algorithm applied to the scheduling problem in technological processes of having schemed with ambient Flow Shop

José Eduardo Márquez Delgado[1], Ricardo Lorenzo Ávila Rondón[2], Miguel Ángel Gómez-Elvira González[3] y Carlos Rafael Herrera Márquez[4]

RESUMEN. Debido a las limitaciones de las técnicas de optimización convencionales, en el siguiente trabajo se presenta una metaheurística basada en un algoritmo genético (AG), para resolver problemas de programación de tipo flow shop, con el objetivo de minimizar el tiempo de finalización de todos los trabajos, más conocido como makespan. Este problema, considerado de difícil solución, es típico de la optimización combinatoria y se presenta en talleres con tecnología de maquinado, donde existen máquinas-herramientas convencionales y se fabrican diferentes tipos de piezas que tienen en común una misma ruta tecnológica (orden del proceso). La solución propuesta 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 empleado.

Palabras clave: algoritmo genético, flow shop, programación, makespan.

ABSTRACT. Due to the limitations of the conventional techniques of optimization, in the following work a metaheuristic one is presented based on a genetic algorithm (GA), to solve scheduling problems of type flow shop, with the objective of minimizing the time of culmination of all the works, good known as makespan. This problem, considered to be a difficult solution is typical in combinatory optimization and it is presented in workshops with technology of having schemed, where conventional machine-tools exist and different types of pieces are manufactured that have in common oneself technological route (process order). The proposed solution it was proven with classic problems published by other authors, obtaining you satisfactory results as for the quality of the opposing solutions and the time of used computation.

Keywords: genetic algorithm, flow shop, scheduling, makespan.

INTRODUCCIÓN

La planificación y control de la producción se reconoce como un problema complejo en el mundo empresarial, y en particular la secuenciación (scheduling), realiza un rol muy importante dentro del proceso de toma de decisiones en una organización y es usada en muchas áreas de la misma, tales como: producción, distribución, transporte, en procesos de información y comunicación, etc. 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 como también se le conocen,

Recibido 10/12/10, aprobado 30/01/12, trabajo 27/12, artículo original.

[1] Prof., Departamento de Informática. Facultad de Ciencias Técnicas, Universidad de Granma (UDG), Carretera Bayamo–Manzanillo km. 17 ½, Granma, CP 85100, Cuba, E-✉: jmarquezd@udg.co.cu

[2] Prof., Centro de Estudios CAD/CAM,. Facultad de Ingeniería, Universidad de Holguín (UHo), Ave. XX Aniversario Holguín. CP: 80900, Cuba.

[3] Prof., Escuela Técnica Superior de Ingenieros (E.T.S.I.) Agrónomos, Universidad Politécnica de Madrid (UPM), España.

[4] Ing., Empresa de Acumuladores "XX Aniversario", Zona Industrial. Manzanillo, CP 87510 Granma, Cuba.

suministrando un creciente arsenal de métodos y algoritmos cuyo uso se extiende paulatinamente sustituyendo a las antiguas reglas y algoritmos usados tradicionalmente (Pinedo, 2008; Brucker *et al.,* 1994; Giffler y Thompson, 1960; González *et al*., 2006; Yamada y Nakano, 1997).

Las industrias que pertencen a la rama de construcción de maquinarias en nuestro país, que cuentan con talleres de maquinado para la fabricación de piezas, adolecen de métodos eficaces que, convertidos en herramientas, contribuyan a mejorar el proceso de planificación de la producción. En uno de los estudios realizados en este sentido en el Centro de Estudios de Diseño y Fabricación Asistidos por Computadoras *(CAD/CAM)* de la Universidad de Holguín, su autor Rondón (2008), hace 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), usando como enfoque para su solución una Red Neuronal Artificial (RNA). Esta temática ha sido abordada por varios investigadores, los cuales han desarrollado múltiples algoritmos para resolver este problema, pero debido a su complejidad en instancias grandes (Brucker, 2006), no resulta posible contar con un método totalmente determinista para su solución general. De ahí, que en los últimos años se han desarrollado y aplicado diversas metaheurísticas, tales como: Algoritmos Genéticos de Yamada y Nakano (1997), Búsqueda Tabú de Zhang *et al.* (2008), Recocido Simulado de Yamada y Nakano (1996), Colonia de Hormigas de Xing *et al.* (2010), entre otras. Los problemas de scheduling en la práctica poseen estructuras más complejas, pero 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. Sin embargo, 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 maquinado tanto para la fabricación de nuevos productos, como para la fabricación de piezas de repuestos.

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 (Garey *et al.,* 1976). 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 esta formado por *n* secuencias de operaciones para cada máquina. Puesto que cada secuencia de operaciones puede ser permutada independientemente de la secuencia de operaciones 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 (Schutten, 1998).

MÉTODOS

Descripción del problema de scheduling

El *JSSP* requiere planificar un conjunto de N trabajos $\{J_1,\ldots, J_N\}$ sobre un conjunto de *M* máquinas $\{R_1,\ldots, R_N\}$. Cada trabajo J_i consiste en una serie de operaciones $\{\theta_{i1},\ldots, \theta_{iM}\}$ 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 $st_{\theta i1}$, 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.

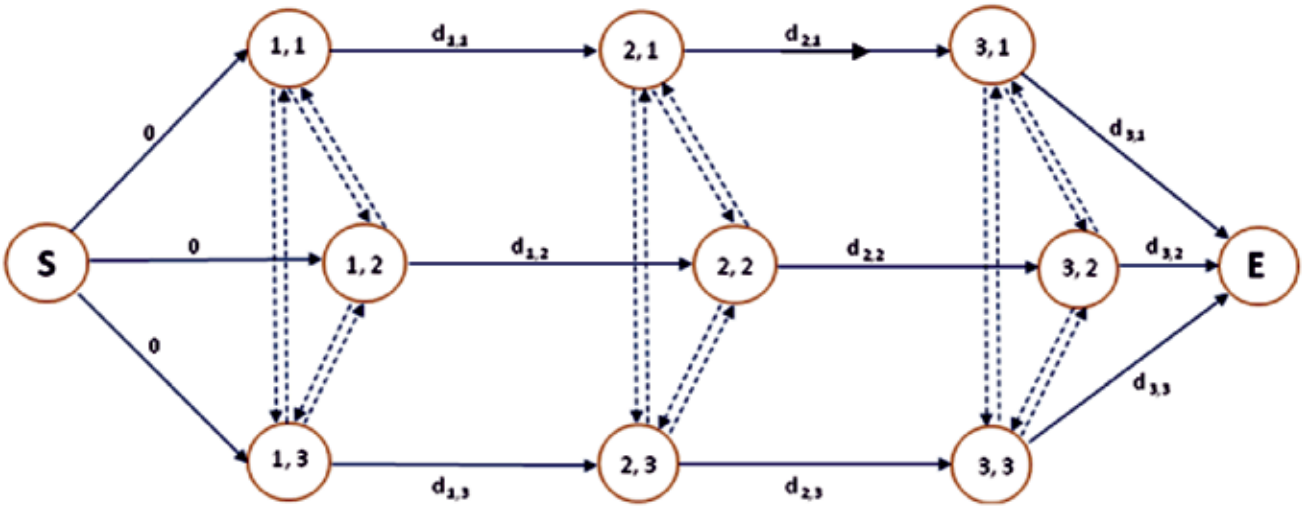
El problema a resolver asume las siguiente restricciones:

- Hay solo una máquina de cada tipo, no existiendo varias máquinas para realizar una 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 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, etc.

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 de Yamada y Nakano (1997), 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, lo cual no le quita el mérito ni la fortaleza a la representación basada en redes de Petri.

Un grafo (Figura 1) 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 $u, v \in V$, tal que $u \neq v$. 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; los nodos S y E son el primero y el último nodo de todos los trabajos respectivamente. 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 más largo entre S y E también conocido como camino máximo o camino crítico *(critical path)*.



Operación en la máquina i, para el trabajo j.

Tiempo de procesamiento de la máquina i para el trabajo j.

Arco conjuntivo (secuencia tecnológica). Restricciones tecnológicas.

Arco disyuntivo (pares de operaciones en la misma máquina). Restricciones de capacidad.

FIGURA 1. Ejemplo de un grafo disyuntivo dirigido para un problema tipo FSS de 3 trabajos en 3 máquinas (3 x 3).

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.

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:

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

Solución a problemas de scheduling 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 (1989), 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 alcan-

zar 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 algoritmo genético (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 oprtunidades 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. En la solución al *Flow Shop Scheduling* en esta investigación, se utilizó un Algoritmo Genético Simple (Simple Genetic Algorithm, SAG).

Estructura utilizada para construir el cromosoma

Varias han sido las formas de codificar las soluciones para el problema de la secuenciación de trabajos usando Algoritmos Genéticos. En la siguiente tabla (Tabla 1) se hace una clasificación de la bibliografía atendiendo al tipo de representación empleada.

TABLA 1. Tipos 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 el método propuesto por Holsapple (1993) para la variante *FSS*, debido a varios factores, en primer lugar, de todas las representaciones utilizadas, según en la literatura, es la que utiliza menos memoria para almacenar un cromosoma quedando determinada la longitud de este por la cantidad de trabajos (*n*), en segundo lugar es la representación que más aprovecha el conocimiento del problema a modelar, lo que se traduce en pocos intentos por violentar el orden del proceso o ruta tecnología de la fabricación y además, por resultar suficiente para obtener valores óptimos de planificaciones para el objetivo de minimizar el *makespan*. A continuación, en la Figura 2, se 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 de 20 trabajos en 5 máquinas (20 x 5) de tipo *FSS,* cuyos resultados se ofrecen en la sección de "resultados y discusión" de este trabajo.

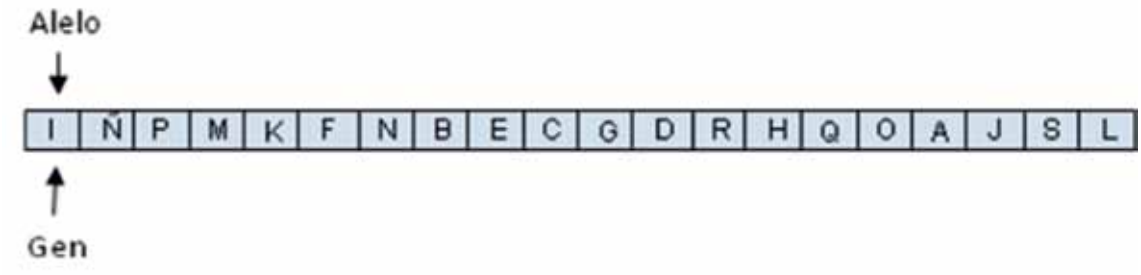


FIGURA 2. Representación de un cromosoma para un problema tipo *flow shop* de 20 trabajos.

Espacios de búsqueda de soluciones

El espacio total de planificaciones factibles es demasiado grande y contiene además muchas soluciones que no son realmente interesantes, ya que a partir de ellas es posible obtener otras mejores o iguales. En la búsqueda de soluciones para problemas de scheduling se consideran tres tipos de planificaciones más reducidas, nombradas: semiactivas, activas y sin retardo (densas). Una planificación es semiactiva cuando para adelantar la ejecución de una tarea (operación) es preciso cambiar el orden relativo de ejecución de al menos dos operaciones. Una planificación es activa si no es posible adelantar la ejecución de una operación sin retardar la ejecución de al menos otra. Una planificación es sin retardo si no existe una máquina inactiva en algún intervalo de tiempo, encontrándose un trabajo listo para ser procesado en dicha máquina.

La Figura 3 muestra la jerarquía de las clases de planes de trabajo. Es fácil comprobar que una planificación sin retardo es activa, y que una planificación activa, es también semiactiva. Además, la intersección del conjunto de planificaciones óptimas

con el conjunto de las planificaciones activas, es siempre no vacío, mientras que en general, esto mismo no se puede garantizar en el caso de las planificaciones sin retardo y semiactivas. En consecuencia, si queremos tener la seguridad de que el espacio de búsqueda contiene al menos una solución óptima debemos restringir la búsqueda, como mucho, al espacio de planificaciones activas. En la presente investigación, se reconoce como región o zona donde es posible encontrar los valores óptimos de solución a problemas de *scheduling,* al espacio de búsqueda de planificaciones activas y dependiendo de la instancia del problema, a los espacios de búsqueda de planificaciones semiactivas y sin retardo. Es decir, no siempre se encuentran valores óptimos en estos dos últimos espacios de planificaciones.

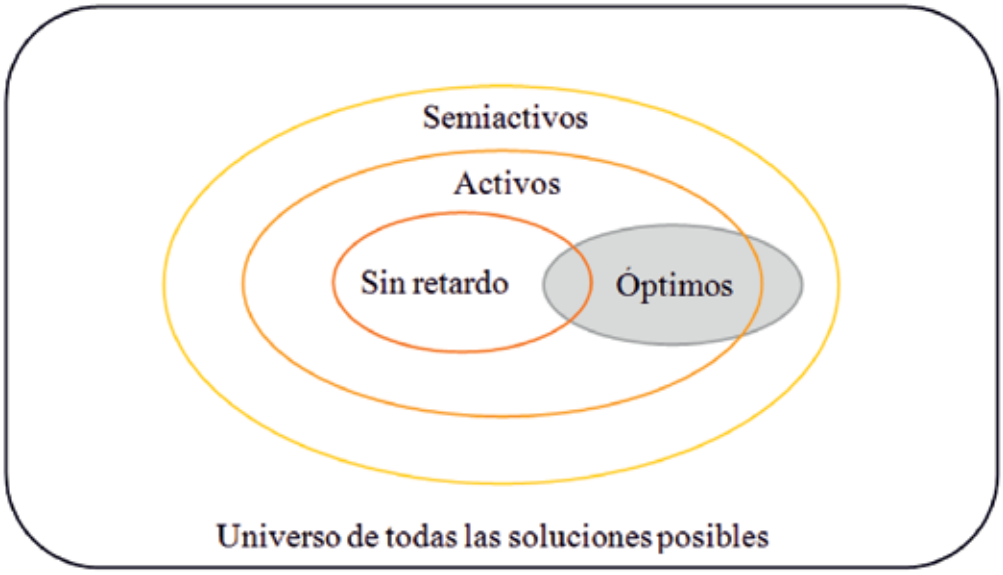


Figura 3. Jerarquía de las clases de planes de trabajo.

RESULTADOS Y DISCUSIÓN

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 ejemplos *(benchmark)* de uso común entre los investigadores, lo cual facilita la comparación de distintos métodos de resolución. La Tabla 2 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 Taillard (1993), que sirven para comparar los resultados obtenidos con la solución ofrecida al problema en este trabajo. Particularmente, se seleccionaron 10 instancias de problemas conformados por 20 trabajos en 5 máquinas (20 x 5). Como se puede apreciar, en todos los casos se alcanzó el límite o cota superior *(upper bound)*. Para la instancia de problema (ta_20_5_05) se obtuvo un resultado en el rango ofrecido entre la cota inferior *(lower bound)* y la superior. La Figura 4 muestra un gráfico de Gantt con la representación de la solución obtenida para la primera instancia (ta_20_5_01).

TABLA 2. Resultados obtenidos para problemas de tipo *flow shop*. Las instancias corresponden a Eric Taillard. *(Taillard's benchmark results)*

Instancias de problemas	Resultados de Taillard		Resultados obtenidos con SGA
	Lower bound	Upper bound	
ta_20_5_01	1232	1278	1278
ta_20_5_02	1290	1359	1359
ta_20_5_03	1073	1081	1081
ta_20_5_04	1268	1293	1293
ta_20_5_05	1198	1236	1235
ta_20_5_06	1180	1195	1195
ta_20_5_07	1226	1239	1239
ta_20_5_08	1170	1206	1206
ta_20_5_09	1206	1230	1230
ta_20_5_10	1082	1108	1108

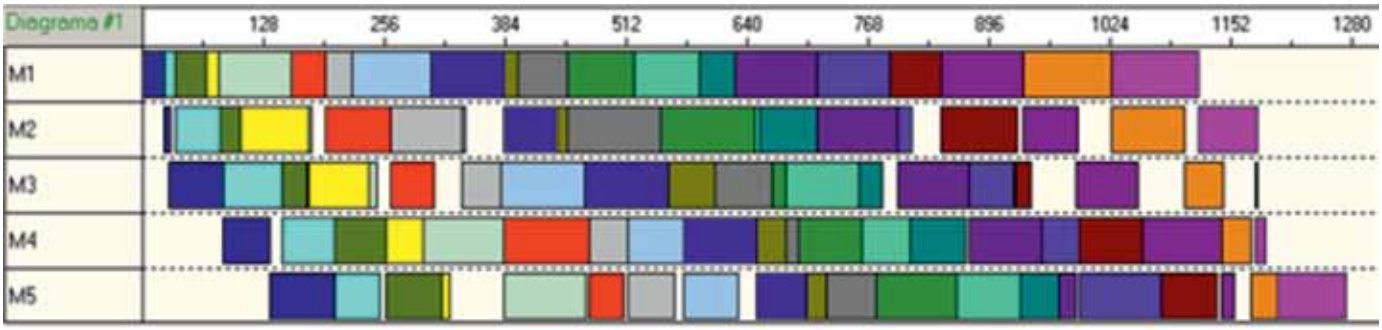


FIGURA 4. Representación en forma de gráfico de Gantt de un plan de trabajo tipo *flow shop.* Corresponde a la solución de la primera instancia de problemas (ta_20_5_01). Valor de cota superior de $C_{máx}$=1278.

Finalmente, derivado de la investigación, se realizó una aplicación (programa para computadora), la cual ha sido utilizada para mostrar los resultados alcanzados en este trabajo. Para los experimentos y ejecuciones se utilizó un ordenador con 512 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.* En este caso 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 este caso Algoritmos Genéticos. Todos los algoritmos desarrollados en general tienen un buen desempeño, y la solución propuesta muestra resultados de alta calidad, y se adapta apropiadamente al tipo de problema resuelto en esta investigación.

REFERENCIAS BIBLIOGRÁFICAS

1. ÁVILA RONDÓN, R. L.; S, CARVALHO & I. HERNÁNDEZ: *IFIP International Federation for Information Processing,* pp.231-238, Volume 266, Innovation in Manufacturing Networks; ed. A. Azevedo; (Boston: Springer), USA, 2008.
2. BRUCKER, P., B. JURISCH & B. SIEVERS: "A branch and bound algorithm for the Job-Shop Scheduling Problems", *Discrete Applied Mathematics*, 49:107-127, 1994.
3. BRUCKER, P.& S. KNUST: *Complex Scheduling,* Springer, USA, 2006.
4. GAREY, M. R.; S. JOHNSON & R. SETHI: "Cpmplexity of Flow Shop and Job Shop Scheduling", *Mathematics of Operations Research*, 1(2): 117-119, 1976.
5. GIFFLER, B & G. THOMPSON: "Algorithm for solving production scheduling problems", *Operation Research*; 8: 487-503, 1960.
6. GOLBERG, D. E.: *Genetic Algorithm in Search*, Optimization and Machine Learning, Addison-Wesley, Publishing Company, Inc, USA, 1989.
7. GONZÁLEZ, M. A., R. VELA & R. VARELA: *Scheduling with memetics algotithms over the spaces of semi-active and active schedules*, Lecture notes in Artificial Intelligence, 4029:370-379, DOI: 10.1007/11785231_40, Springer-Verlag Berling Heidelberg, 2006.
8. HOLSAPPLE, C.; V. JACOB; R. PAKATH & J. ZAVERI: "A Genetics-Based Hybrid Scheduler for Generating Static Schedules in Flexible Manufacturing Contexts". *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4): 953-971, 1993.
9. PINEDO, L. M.: *Scheduling, Theory, Algorithms, and Systems*, Springer, third edition, Springer-New York, 2008.
10. SCHUTTEN, J. M. J.: "Practical job shop scheduling". *Annals of Operations Research*, 83: 1998.
11. TAILLARD, E.: "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, 64: 278-85, 1993.
12. XING, L., Y. CHEN; P. WANG; Q. ZHAO; & J. XIONG: "A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems", *Applied Soft Computing*, 10: 888-896, 2010.
13. YAMADA, T & R. NAKANO: *Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search,* Kluwer academic publishers, MA, USA, 1996.
14. YAMADA, T & R. NAKANO: Genetic Algorithms for Job-Shop Scheduling Problems, Proceedings of Modern Heuristic for decision support, In: UNICOM Seminar; 18-19 March; London, 1997,
15. ZHANG, C. Y.; P. LI; Y. RAO & Z. GUAN: "A very fast TS/SA algorithms for job-shop scheduling problems", *Computers and Operations Research*, 35: 282-294, 2008.