Un algoritmo genético para el problema de *Job Shop Flexible*

A genetic algorithm for the Flexible Job Shop problem

Rosa Medina Durán[1] Lorena Pradenas Rojas[1] Víctor Parada Daza[2]

Recibido 21 de diciembre de 2009, aceptado 10 de enero de 2011
Received: December 21, 2009 Accepted: January 10, 2011

RESUMEN

En este estudio se propone e implementa computacionalmente un algoritmo genético secuencial para resolver el problema del *Job Shop Flexible* (existente en la Gestión de Operaciones), el cual es parte de la familia de los problemas de programación de tareas o trabajos (*Scheduling*) en un taller que funciona a pedido. Surge como una generalización del problema del *Job Shop* y permite optimizar el uso de los recursos (máquinas) con mayor flexibilidad, ya que cada máquina puede realizar más de una operación. Este problema ha sido estudiado por numerosos autores, los que han propuesto diversos modelos matemáticos y enfoques heurísticos. Debido a la naturaleza combinatoria, los métodos exactos que resuelven modelos matemáticos encuentran soluciones sólo para instancias pequeñas o simples del problema mencionado. Los resultados muestran la efectividad del algoritmo propuesto para entregar buenas soluciones en tiempos computacionales razonables en más de 130 instancias encontradas en la literatura.

Palabras clave: Problema *Job Shop Flexible*, algoritmos genéticos, programación de trabajos, optimización combinatoria, gestión de operaciones.

ABSTRACT

This study proposes and computationally implements a sequential genetic algorithm to solve the Flexible Job Shop problem (found in Operations Management), which is part of the family of job or task scheduling problems in a shop that works on demand. It is a generalization of the Job Shop problem, and allows optimizing the use of resources (machines) in the shop, with greater flexibility, since each machine can perform more than one operation. This problem has been studied by many authors, who have proposed various mathematical models and heuristic approaches. Due to the combinatorial nature of the problem, the exact methods that solve the mathematical models are often solutions for small and simple instances of the problem. The results show the effectiveness of the proposed algorithm to provide good solutions in reasonable computational times in over 130 instances found in the literatura.

Keywords: Flexible Job Shop problem, genetic algorithms, scheduling, combinatorial optimization, operations management.

INTRODUCCIÓN

En la programación de la producción de las industrias manufactureras se debe decidir sobre la asignación de los recursos a las tareas o trabajos para optimizar, uno o más objetivos en el corto plazo. Para apoyar estas decisiones tradicionalmente se utiliza el modelo de *Job Shop*, el cual considera un conjunto de máquinas y un conjunto de trabajos compuestos por una secuencia ordenada de operaciones que se deben procesar en las máquinas con el objetivo de minimizar, entre otros, el tiempo de completación de la última operación, makespan.

[1] Departamento de Ingeniería Industrial. Facultad de Ingeniería. Universidad de Concepción. Casilla 160-C. Correo 3. Concepción, Chile. E-mail: rosmedina@udec.cl; lpradena@udec.cl
[2] Departamento de Ingeniería Informática. Facultad de Ingeniería. Universidad de Santiago de Chile. Avda. Ecuador 3659. Santiago, Chile. E-mail: victor.parada@usach.cl

El problema de *Job Shop Flexible* es una generalización del problema de *Job Shop*. Considera que las operaciones pueden ser procesadas por un grupo de máquinas, razón por la cual también se requiere decidir sobre cuáles de las máquinas procesan cada operación.

Por ejemplo, consideremos un taller metal-mecánico, en el que se reciben dos trabajos compuestos por una secuencia fija de tres operaciones a realizar en tres máquinas. El primero requiere las operaciones de perforar, ranurar y pulir, mientras el segundo trabajo requiere cortar, limar y perforar. El taller dispone de un torno, que puede realizar las operaciones de perforar, pulir y cortar; una fresadora que puede perforar, ranurar, pulir y limar; y un taladro que puede perforar, ranurar y cortar. Los tiempos de las operaciones varían en las distintas máquinas, entonces, es necesario decidir cuál máquina realiza cada operación y en qué orden, con el objetivo de minimizar el tiempo total de completación de todos los trabajos.

Este problema pertenece a la clase de problemas denominados de NP-Difíciles, para los cuales aún no se disponen de algoritmos de resolución eficientes que los ejecuten en tiempos computacionales razonables, sobre todo para instancias grandes y/o complejas.

Haciendo un breve recuento de los enfoques que intentan resolver el problema, tenemos que Brandimarte [1] propone un enfoque jerárquico para la resolución del problema, considerando un subproblema de asignación y un subproblema de secuenciamiento. El primero lo resuelve como un problema de ruteo, mientras que el segundo corresponde al problema de *Job Shop*. Para probar el algoritmo, este investigador usa quince instancias del *Job Shop Flexible*. Mesghouni, Hammadi y Borne [11] en cambio, resuelven el problema utilizando algoritmos genéticos, consideran un enfoque integrado, representando la solución mediante matrices. Por su parte, Kacem, Hammadi y Borne [10] proponen un enfoque por localización, que permite encontrar buenas soluciones, minimizando el *makespan* y la carga de las máquinas. Este último enfoque controla la evolución de un algoritmo genético.

Ho y Tay [6] y en sus distintas publicaciones estudian el problema del *Job Shop Flexible*, proponiendo reglas de despacho que utilizan como solución inicial para una nueva metodología denominada GENACE [8], en la cual las poblaciones son influenciadas por reglas heurísticas y en cada generación se guía el espacio de búsqueda mediante esquemas. Otros autores también estudian las representaciones de las soluciones [13], para encontrar la más eficaz en términos de tiempo de cómputo y de alcanzar el mejor *makespan*. Este enfoque es generalizado en una publicación posterior en Ho, Tay y Lai [7], donde se propone una arquitectura basada en algoritmos evolutivos, un aprendizaje con teoría de esquemas y un generador de poblaciones mediante reglas de despacho compuestas.

Fattahi, Meharab y Jolai [5] proponen un modelo matemático para el *Job Shop Flexible* y para probarlo generan diez instancias pequeñas y diez instancias medianas/grandes, aunque sólo logran resolver las instancias pequeñas. Utilizando también el enfoque jerárquico, combinan las metaheurísticas de *Tabu Search* y *Simulated Annealing*, con las que concluyen que el algoritmo que resuelve el subproblema de asignación con *Tabu Search* y el subproblema de secuenciamiento con *Simulated Annealing* presenta el mejor desempeño.

Zhang y Gen [15] proponen un algoritmo genético basado en escenarios múltiples, donde cada escenario corresponde a una operación y cada máquina factible a un estado. Pezzella, Morganti y Ciaschetti [12] también proponen un algoritmo genético, pero utilizan el enfoque de localización propuesto por Kacem, Hammadi y Borne [10]. Con ello establecen una mutación inteligente, que consiste en seleccionar una operación entre aquellas máquinas con más carga de trabajo y reasignar dicha carga a la máquina que tenga menos. Por último, la publicación más reciente corresponde a Yazdani, Zandieh y Amiri [14], quienes proponen un enfoque en paralelo de vecindario variable (VNS), con seis vecindarios, minimizando también el *makespan*.

Todo lo anterior nos lleva a plantear que el objetivo del presente estudio es proponer e implementar computacionalmente un nuevo algoritmo genético para resolver el problema de *Job Shop Flexible*, y luego comparar los resultados con otros obtenidos de la literatura en términos de desempeño computacional y calidad de solución.

DEFINICIÓN DEL PROBLEMA

El problema de *Job Shop Flexible* se define como:

"Dado un sistema con un conjunto de *m* máquinas, $M=\{M_1,...,M_m\}$, y un conjunto de *n* trabajos independientes, $J = \{J_1,...,J_n\}$. Considerando que cada trabajo está compuesto por una secuencia de operaciones $O_{\{j,1\}},O_{\{j,2\}},...,O_{\{j,h\}},...,O_{\{j,hj\}}$, cada una de las cuales debe ser procesada en una máquina. Se dispone de los tiempos de proceso $p_{\{i,j,h\}}$ de cada operación $O_{\{j,h\}}$ en cada máquina factible $M_i \in M_{\{j,h\}} \subseteq M$, el problema de *Job Shop Flexible* requiere minimizar el tiempo de completación de la última operación: el *makespan*".

En la definición, $O_{\{j,h\}}$ corresponde a la *h*-ésima operación del trabajo J_j y h_j indica el número de operaciones que requiere el trabajo J_j. Además, $M_{\{j,h\}} \subseteq M$ es el conjunto de todas las máquinas en el problema que pueden procesar la operación $O_{\{j,h\}}$. Cuando todas las operaciones pueden ser procesadas por todas las máquinas, $M_{\{j,h\}} = M$, la flexibilidad del problema es total (T-FJSP). Por el contrario, cuando al menos una operación $O_{\{j,h\}}$ no puede ser procesada en todas las máquinas, es decir $M_{\{j,h\}} \subset M$, la flexibilidad del problema es parcial (P-FJSP).

Se conoce el tiempo de proceso de cada operación en cada una de las máquinas donde puede ser procesada. Se denota como $p_{\{i,j,h\}}$ el tiempo de proceso de la operación $O_{\{j,h\}}$ en la máquina $M_i \in M_{\{j,h\}}$. No se permite interrumpir las operaciones cuando han iniciado su proceso, como tampoco se permite que las máquinas ejecuten más de una operación simultáneamente. Además, se asume que todos los trabajos y máquinas están disponibles en el instante (tiempo) cero.

Para encontrar el tiempo de completación de la última operación (*makespan*) es necesario asignar las operaciones a las máquinas y secuenciar dichas operaciones, por lo cual la función objetivo del problema se puede expresar como:

$$\min C_{max} = \min \{\max_{Jj \in J} C_{Jj}\} \qquad (1)$$

ALGORITMO PROPUESTO

Los Algoritmos Genéticos son métodos metaheurísticos de optimización estocástica inspirados en la evolución natural de las especies y propuestos por Holland [9]. Establecen una analogía entre el conjunto de soluciones del problema y el conjunto de individuos de una población natural. Así como las poblaciones de individuos evolucionan en cada generación, el conjunto de soluciones mejora en cada iteración.

Para explicar con mayor detalle el algoritmo genético propuesto en este estudio se utiliza un ejemplo que corresponde a una instancia de P-FJSP [3], que contiene dos trabajos y tres máquinas (2×3), como el presentado en la Introducción. Para ejecutar cada trabajo se requiere realizar tres operaciones, por lo que el problema total consta de seis operaciones cuyos tiempos de proceso se muestran en la Tabla 1, donde las "X" indican que la máquina no puede procesar esa operación.

Tabla 1. Ejemplo *Job Shop Flexible* parcial.

	M_1	M_2	M_3
$O_{1,1}$	1	2	1
$O_{1,2}$	X	1	1
$O_{1,3}$	4	3	X
$O_{2,1}$	5	X	2
$O_{2,2}$	X	2	X
$O_{2,3}$	7	5	3

Es fundamental utilizar una codificación de las soluciones que represente las características del problema y respete las restricciones. Para la representación de la solución y para una mejor comprensión y tratamiento de la solución, en este estudio se propone que cada solución del problema utilice tres vectores. El primero de ellos representa una solución para el subproblema de secuenciamiento; el segundo indica la operación a la cual corresponde cada celda y el tercero determina una solución para el subproblema de asignación.

El cromosoma de secuenciamiento es un vector de tamaño igual al número de operaciones en el problema, similar a la representación propuesta en [8]. En cada celda aparece el número de un trabajo; el orden en que se encuentran define cuál es la prioridad de proceso de los trabajos. Cada trabajo

j aparece h_j veces en el vector, una vez por cada operación. Se requiere, además, una función que no permita que la repetición de los trabajos exceda el número de sus operaciones. De este modo se indica la precedencia de las operaciones entre distintos trabajos y esta se respeta cuando pertenecen al mismo trabajo. En el ejemplo, una solución para el problema de secuenciamiento se muestra en la Figura 1, los números de cada celda corresponden a un determinado trabajo, los cuales aparecen repetidos tres veces, ya que cada trabajo tiene tres operaciones señaladas bajo cada celda.

1º	2º	3º	4º	5º	6º
2	1	1	2	2	1
$O_{2,1}$	$O_{1,1}$	$O_{1,2}$	$O_{2,2}$	$O_{2,3}$	$O_{1,3}$

Figura 1. Ejemplo de cromosoma de secuenciamiento.

Por su parte el cromosoma de operaciones es un vector similar al anterior, pero en cada celda se indica el número de la operación para hacer la correspondencia entre los trabajos del cromosoma de secuenciamiento y sus operaciones. Para el ejemplo, los números de las operaciones se muestran a la izquierda en la Figura 2 y, a la derecha, se muestra el cromosoma de operaciones que se obtiene para el cromosoma de secuenciamiento de la Figura 1.

$O_{1,1}$	1
$O_{1,2}$	2
$O_{1,3}$	3
$O_{2,1}$	4
$O_{2,2}$	5
$O_{2,3}$	6

1º	2º	3º	4º	5º	6º
4	1	2	5	6	3
$O_{2,1}$	$O_{1,1}$	$O_{1,2}$	$O_{2,2}$	$O_{2,3}$	$O_{1,3}$

Figura 2. Número de las operaciones y ejemplo de cromosoma de operaciones.

Por otro lado, el cromosoma de asignación también es un vector de las mismas dimensiones que los anteriores, pero cada celda contiene el índice de la máquina en la cual se procesa la operación que corresponde según los cromosomas anteriores. Esta representación es parecida a la propuesta en [3], aunque difiere en que la secuencia de las operaciones a las cuales es asignado depende del cromosoma del subproblema de operaciones. En el ejemplo (Figura 3) cada celda está asociada con la operación indicada en la parte superior, de acuerdo al cromosoma de operaciones. Estas máquinas corresponden sólo a aquellas que pueden procesar la operación, es decir, en nuestro ejemplo no puede ir un número diferente de dos en la cuarta celda. De este modo, el cromosoma de la figura asigna la máquina tres (M_3) a la operación $O_{\{2,1\}}$, la máquina uno (M_1) a la operación $O_{\{1,1\}}$, la máquina dos (M_2) a la operación $O_{\{1,2\}}$, etc.

$O_{2,1}$	$O_{1,1}$	$O_{1,2}$	$O_{2,2}$	$O_{2,3}$	$O_{1,3}$
3	1	2	2	3	2

Figura 3. Ejemplo de cromosoma de asignación.

Para evaluar la calidad de un individuo se utiliza la función de *fitness*. En este problema, el objetivo es la minimización del *makespan*; es decir, individuos con menor *makespan* tienen un mejor *fitness*; por lo cual, la función de *fitness* adoptada corresponde al inverso del *makespan*.

Los operadores genéticos más utilizados son el *crossover* y la mutación. El *crossover* corresponde al cruzamiento de los individuos de una población y el consiguiente intercambio de información genética en los individuos generados. La mutación, por otra parte, permite introducir la aleatoriedad que puede dar lugar a diferentes soluciones y permiten explorar distintas zonas de la región factible evitando óptimos locales o convergencia prematura.

El operador de *crossover* se aplica a los cromosomas de secuenciamiento y de asignación, entre cromosomas del mismo tipo. Dadas dos soluciones para aplicar el operador, se aplica a ambos cromosomas (con 20% de probabilidad); con 40% se aplica sólo al cromosoma de secuenciamiento y también con 40% se aplica sólo al cromosoma de asignación. Se utiliza el *crossover* de dos puntos, el cual selecciona dos posiciones aleatorias e intercambia los genes intermedios dando origen a dos soluciones nuevas. Una función verifica que los cromosomas obtenidos sean válidos para el subproblema correspondiente, si no es así, reasigna valores válidos. Los cromosomas de secuenciamiento pueden no ser factibles, ya que podría incrementarse el número de veces que aparece un trabajo en la solución; en este caso, se cambian los trabajos que se encuentran en exceso por aquellos en los que faltan operaciones. Los cromosomas de asignación pueden no ser válidos, ya que las máquinas podrían no ser factibles para

las operaciones de los nuevos cromosomas; en este caso se cambia la máquina a una de las máquinas factibles seleccionada en forma aleatoria.

Se proponen tres operadores de mutación, de los cuales se aplica uno a cada individuo seleccionado para la mutación. El operador de mutación de asignación al azar, aplicado con 20% de probabilidad, selecciona una posición del cromosoma de asignación y cambia la máquina que ha sido asignada por otra máquina factible. El operador de asignación inteligente, también con 20% de probabilidad, es similar al anterior pero la nueva máquina asignada se escoge entre aquellas que tienen menor carga. El tercer operador modifica el cromosoma de secuenciamiento, tiene 60% de probabilidad; dado un cromosoma de secuenciamiento, escoge dos posiciones aleatoriamente y cambia su valor. Una función asegura que los cromosomas de operaciones y de asignación asociados continúen siendo válidos para el nuevo cromosoma de secuenciamiento.

La solución inicial del algoritmo genético se obtiene de modo aleatorio. Para generar un cromosoma de secuenciamiento se escoge una posición cualquiera y se le asigna la primera operación; en la celda a la derecha de ésta, se asigna la segunda operación y así sucesivamente se completan todas las celdas; cuando se llega a la última, se vuelve al inicio del vector. Una vez obtenida la población de secuenciamiento queda determinada la población de operaciones. Por cada cromosoma se debe generar un cromosoma de asignación, para lo cual, por cada celda, se escoge una máquina aleatoriamente. Se verifica que la máquina pueda procesar la operación, de lo contrario se escoge una nueva máquina para ser asignada.

Los parámetros del algoritmo genético son el tamaño de la población, el número de generaciones o iteraciones, la probabilidad de *crossover* y la mutación. Estos deben ser determinados por el usuario antes de su ejecución.

RESULTADOS

El algoritmo se implementa en Lenguaje C y los experimentos se realizan en un MacBook, con un procesador Intel Core 2 Duo de 2.4Ghz, 2GB de memoria RAM y sistema operativo Mac OS X 10.5.8.

Para la experimentación se utilizan algunas instancias de la literatura, disponibles en internet o en las publicaciones, mientras que otras han sido solicitadas a los propios autores. Es preciso señalar que los elementos que caracterizan a una instancia son: el número de máquinas, el número de trabajo y el número de operaciones totales.

El desempeño de cualquier metaheurística depende fuertemente de los parámetros que la controlan. Por esta razón, es muy importante hacer un análisis de cuáles son los mejores óptimos o intervalos de valores para estos parámetros, con la finalidad de encontrar los mejores resultados. Se usa un diseño experimental [4], para lo cual se seleccionan al azar ocho instancias de prueba (08a.fjs, 01a.fjs, 04a.fjs, Mk04.fjs, 11a.fjs, orb8.fjs, car5.fjs, mt20.fjs). Por cada parámetro se elige un valor inicial, un rango y un paso de incremento. Estos valores se muestran en la Tabla 2; fueron determinados de acuerdo a estudios preliminares de comportamiento del algoritmo propuesto.

Por ejemplo, para el parámetro; "Tamaño de la Población", el algoritmo fue ejecutado para cada una de las ocho instancias de prueba y para cada valor de "Tamaño de Población" en el intervalo recomendado (1.000, 1.500, 2.000, 2.500, 3.000, 3.500, 4.000, 4.500 y 5.000), manteniendo constantes todos los otros parámetros. De igual forma se realizó el experimento para los otros parámetros restantes. De los resultados anteriores, los mejores parámetros en términos de *makespan* para el algoritmo propuesto son:

- Tamaño de la Población : 4.500
- Número de Generaciones : 3.000
- Probabilidad de *Crossover* : 0,75
- Probabilidad de Mutación : 0,30

En Anexo A (Tabla 4) se encuentran los resultados alcanzados en cada una de las 137 instancias de la literatura utilizadas, se dispone de: número de máquinas (N° máq.), número de trabajos (N° trab.), número de operaciones totales (N° Ope.), tiempos de cómputo (Tiempo, en segundos) y el mejor valor de *makespan* obtenido. El tiempo medio de cómputo es 43,56 minutos. La convergencia del valor promedio de *makespan* por cada generación se muestra en el gráfico de la Figura 4. La línea violeta (superior) corresponde al *makespan* promedio de cada

generación, mientras que la línea negra (inferior) corresponde al mejor *makespan* de cada generación. Se observa que ambas secuencias se estabilizan luego de 1.500 generaciones, aproximadamente, en un valor cercano a 1.500.

DISCUSIÓN

La Tabla 3 compara los resultados obtenidos con el algoritmo propuesto respecto a otros algoritmos de la literatura. La primera columna indica el nombre del grupo de instancias; la segunda, el número de instancias en el grupo; la tercera, entrega el promedio de los mínimos *makespan* informados en la literatura ($\overline{\min C_{\max}}$), mientras que la cuarta columna entrega los promedios de nuestros resultados, ($\overline{C_{\max}}$); la quinta columna indica el tiempo promedio de cómputo de nuestro algoritmo. Lamentablemente no se dispone de los tiempos de cómputo de los algoritmos de la literatura; la última columna indica la diferencia porcentual promedio entre la solución obtenida

Tabla 2. Valor inicial, rango y paso de incremento para la parametrización.

Parámetro	**Valor inicial**	**Rango**	**Paso de incremento**
Tamaño de la Población	3.000	[1.000,5.000]	500
Número de Generaciones	2.000	[1.000,3.000]	500
Probabilidad de *Crossover*	0,70	[0,60, 0,80]	0,05
Probabilidad de Mutación	0,20	[0,1, 0,3]	0,05

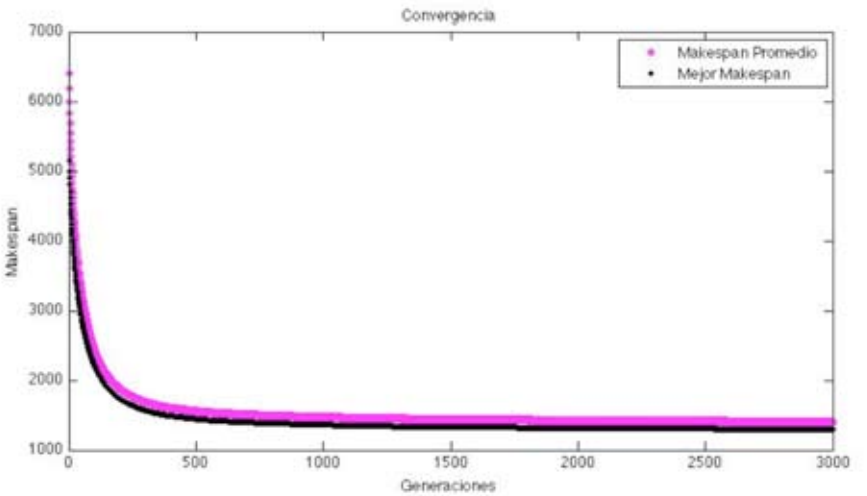

Figura 4. Convergencia de la media de los valores promedio de *makespan* y mejor *makespan* por cada generación.

Tabla 3. Comparación instancias de la literatura.

Grupo de instancias	**Nº instancias**	$\overline{\min C_{\max}}$	$\overline{C_{\max}}$	**Tiempo (s)**	$\overline{Dif}$
Fattahi [5]	20	490,00	501,85	104,85	1,17%
Brandimarte [1]	10	172,80	193,10	2304,74	16,44%
Chan [2]	10	781,30	655,30	555,84	–16,76%
Mesghouni [11]	1	7	10	199,38	42,86%
Kacem [10]	1	14	14	174,72	0,00%

con nuestro algoritmo y la mejor de la literatura calculada de acuerdo a la fórmula:

$$Dif = (C_{max} - minC_{max}) / minC_{max} \, x100\% \quad (2)$$

Se observa un buen desempeño del algoritmo. En las instancias de Chan[2] y Kacem[10] se superan o alcanzan las mejores soluciones conocidas, mientras en los otros grupos nuestra solución es cercana a la mejor solución conocida, sin superar el 50% de diferencia, en media. Respecto a los tiempos de cómputo, podemos ver que no exceden de una hora.

CONCLUSIONES

Esta investigación propone un algoritmo genético para resolver el problema de *Job Shop Flexible*. Mediante un diseño de experimentos se determinan los parámetros del algoritmo. Se resuelven instancias de la literatura, en un tiempo razonable para la programación de la producción en un taller que funciona a pedido.

Dado que las soluciones convergen, podrían ser mejoradas diversificando la población inicial o los operadores genéticos. También sería interesante paralelizar el algoritmo, lo cual permite disminuir los tiempos de cómputo y diversificar las soluciones.

AGRADECIMIENTOS

Este estudio fue apoyado por el proyecto ALFA Nº II-0457-FA-FCD-FI-FC, de la Comunidad Europea, coordinado desde Concepción (Chile), con la participación de investigadores de: Chile, Uruguay, Brasil, Cuba, Bélgica, Francia e Italia y por el proyecto DIUC-208.97011-1, de la Universidad de Concepción, Chile.

REFERENCIAS

[1] P. Brandimarte. "Routing and scheduling in a flexible job shop by tabu search". Annals of Operations Research. Vol. 41, Issue 1-4, pp. 157-183. 1993.

[2] F.T.S. Chan, T.C. Wong and L.Y. Chan. "Flexible job-shop scheduling problem under resource constraints". International Journal of Production Research. Vol. 44, Issue 11, pp. 2071-2089. 2006.

[3] H. Chen, J. Ihlow and C. Lehman. "A genetic algorithm for flexible job-shop scheduling". IEEE International Conference on Robotics and Automation. Vol. 2, pp. 1120-1125. 1999.

[4] S. Coy, B. Golden, G. Runger and E. Wasil. "Using experimental design to find effective parameter setting for heuristics". Journal of Heuristics. Vol. 7, pp. 77-97. 2001.

[5] P. Fattahi, M. Saidi Meharab and F. Jolai. "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems". Jounal of Intelligent Manufacturing. Vol. 8, Issue 3, pp. 331-342. 2007.

[6] N.B. Ho and J.C. Tay. "Evolving dispatching rules for solving the flexible job-shop problem". IEEE Congress on Evolutionary Computation. Vol. 3, pp. 2848-2855. September, 2005.

[7] N.B. Ho, J.C. Tay and E.M.-K. Lai. "An effective architecture for learning and evolving flexible job-shop schedules". European Journal of Operational Research. Vol. 179, Issue 2, pp. 316-333. June, 2007.

[8] N.B. Ho and J.C. Tay. "GENACE: An efficient cultural algorithm for solving the flexible job-shop problem". Congress on Evolutionary Computation, pp. 1759-1766. 2004.

[9] J.H. Holland. "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence". The MIT Press, 228 p. April, 1992.

[10] I. Kacem, S. Hammadi and P. Borne. "Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems". IEEE Transactions on Systems, Man, and Cybernetics. Vol. 32, Issue 1, pp. 1-13. February, 2002.

[11] K. Mesghouni, S. Hammadi and P. Borne. "Evolution Programs for job-shop scheduling". IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation. Vol. 1, pp. 720-725. October, 1997.

[12] F. Pezzella, G. Morganti and G. Ciaschetti. "A genetic algorithm for the flexible job-shop scheduling problem". Computers and

Operations Research. Vol. 35, Issue 10, pp. 3202-3212. October, 2008.

[13] J.C. Tay and D. Wibowo. "An effective chromosome representation for evolving flexible job shop schedules". Lecture Notes in Computer Sciences. Vol. 3103/2004, pp. 210-221. 2004.

[14] M. Yazdani, M. Zandieh and M. Amiri. "Flexible job-shop scheduling with parallel variable neighborhood search algorithm". Expert Systems with Applications: An International Journal. Vol. 37, Issue 1, pp. 678-687. January, 2010.

[15] M. Gen, J. Gao and L. Lin. "Multistaged-based genetic algorithm for flexible job-shop scheduling problem". Complexity International. Vol. 11. 2005. URL: http://www.complexity.org.au/ci/vol11/zhang01/

ANEXO A. TABLA DE RESULTADOS

La primera columna de la Tabla 4 tiene el número y nombre de la instancia; la segunda, el número de máquinas (N° máq.); la tercera, el número de trabajos (N° trab.); la cuarta, el total de operaciones (N° Ope.); la quinta, el tiempo de cómputo del algoritmo propuesto y, la última, el mejor *makespan* encontrado. La instancia orb7.fjs no fue ejecutada por problemas en el archivo de lectura.

Tabla 4. Resultados del algoritmo propuesto.

	Instancia	N° máq.	N° trab.	N° Ope.	Tiempo (s)	*Makespan*
1	SFJS1	2	2	4	16.59	66
2	SFJS2	2	2	4	16.83	107
3	SFJS3	2	3	6	25.99	221
4	SFJS4	2	3	6	25.91	355
5	SFJS5	2	3	6	26.32	119
6	SFJS6	3	3	9	38.74	320
7	SFJS7	5	3	9	39.03	397
8	SFJS8	4	3	9	38.21	253
9	SFJS9	3	3	9	37.62	210
10	SFJS10	5	4	12	56.72	516
11	MFJS1	6	5	15	72.33	468
12	MFJS2	7	5	15	74.19	448
13	MFJS3	7	6	18	94.62	466
14	MFJS4	7	7	21	116.95	554
15	MFJS5	7	7	21	115.68	514
16	MFJS6	7	8	24	142.00	634
17	MFJS7	7	8	32	205.45	949
18	MFJS8	8	9	36	245.77	921
19	MFJS9	8	11	44	330.55	1198
20	MFJS10	8	12	48	379.89	1321
21	mt10c1.fjs	11	10	100	1098.71	937
22	mt10cc.fjs	12	10	100	1102.47	919
23	mt10x.fjs	11	10	100	1098.29	973
24	mt10xx.fjs	12	10	100	1098.39	949
25	mt10xxx.fjs	13	10	100	1105.30	1000
26	mt10xy.fjs	12	10	100	1106.97	991
27	mt10xyz.fjs	13	10	100	1118.60	938
28	setb4c9.fjs	11	15	150	2211.28	964
29	setb4cc.fjs	12	15	150	2213.55	970
30	setb4x.fjs	11	15	150	2205.68	1000
31	setb4xx.fjs	12	15	150	2210.54	1000
32	setb4xxx.fjs	13	15	150	2220.25	992
33	setb4xy.fjs	12	15	150	2215.34	956
34	setb4xyz.fjs	13	15	150	2229.33	989
35	seti5c12.fjs	16	15	225	4385.25	1292
36	seti5cc.fjs	17	15	225	4399.57	1288
37	seti5x.fjs	16	15	225	4379.84	1305
38	seti5xx.fjs	17	15	225	4406.06	1311
39	seti5xxx.fjs	18	15	225	4400.99	1342
40	seti5xy.fjs	17	15	225	4394.36	1324
41	seti5xyz.fjs	18	15	225	4407.96	1272
42	Mk01.fjs	6	10	55	442.71	42
43	Mk02.fjs	6	10	58	483.81	28
44	Mk03.fjs	8	15	150	2216.48	207
45	Mk04.fjs	8	15	90	1005.73	67
46	Mk05.fjs	4	15	106	1222.74	176
47	Mk06.fjs	15	10	150	2123.51	96
48	Mk07.fjs	5	20	100	1227.39	150
49	Mk08.fjs	10	20	225	4429.62	523
50	Mk09.fjs	10	20	240	4942.58	339
51	Mk10.fjs	15	20	240	4952.85	303
52	01a.fjs	5	10	196	3178.18	2636
53	02a.fjs	5	10	196	3170.87	2413
54	03a.fjs	5	10	196	3173.12	2436
55	04a.fjs	5	10	196	3177.62	2658
56	05a.fjs	5	10	196	3178.31	2383
57	06a.fjs	5	10	196	3167.98	2306
58	07a.fjs	8	15	293	6716.15	2689
59	08a.fjs	8	15	293	6779.00	2485
60	09a.fjs	8	15	293	6722.33	2595

	Instancia	N° máq.	N° trab.	N° Ope.	Tiempo (s)	*Makespan*
61	10a.fjs	8	15	293	6784.16	2579
62	11a.fjs	8	15	293	6756.39	2421
63	12a.fjs	8	15	293	6719.88	2764
64	13a.fjs	10	20	387	11370.99	2832
65	14a.fjs	10	20	387	11493.35	2964
66	15a.fjs	10	20	387	11471.91	2554
67	16a.fjs	10	20	387	11354.93	2714
68	17a.fjs	10	20	387	11386.15	3047
69	18a.fjs	10	20	387	11528.85	2713
70	abz5.fjs	10	10	100	1094.23	964
71	abz6.fjs	10	10	100	1100.77	742
72	abz7.fjs	15	20	300	7352.85	736
73	abz8.fjs	15	20	300	7355.80	758
74	abz9.fjs	15	20	300	7356.65	758
75	car1.fjs	5	11	55	437.82	5035
76	car2.fjs	4	13	52	414.34	5937
77	car3.fjs	5	12	60	502.92	5624
78	car4.fjs	4	14	56	468.74	6518
79	car5.fjs	6	10	60	492.76	5112
80	car6.fjs	9	8	72	639.06	5486
81	car7.fjs	7	7	49	350.47	4281
82	car8.fjs	8	8	64	535.52	4637
83	la01.fjs	5	10	50	376.19	577
84	la02.fjs	5	10	50	377.75	530
85	la03.fjs	5	10	50	379.34	487
86	la04.fjs	5	10	50	378.75	511
87	la05.fjs	5	10	50	378.46	463
88	la06.fjs	5	15	75	734.78	805
89	la07.fjs	5	15	75	731.35	754
90	la08.fjs	5	15	75	730.85	766
91	la09.fjs	5	15	75	734.96	855
92	la10.fjs	5	15	75	735.95	805
93	la11.fjs	5	20	100	1189.74	1072
94	la12.fjs	5	20	100	1195.23	937
95	la13.fjs	5	20	100	1196.12	1040
96	la14.fjs	5	20	100	1196.08	1072
97	la15.fjs	5	20	100	1195.38	1093
98	la16.fjs	10	10	100	1095.78	717
99	la17.fjs	10	10	100	1095.38	646

	Instancia	N° máq.	N° trab.	N° Ope.	Tiempo (s)	*Makespan*
100	la18.fjs	10	10	100	1096.24	663
101	la19.fjs	10	10	100	1096.51	644
102	la20.fjs	10	10	100	1123.39	756
103	la21.fjs	10	15	150	2193.78	904
104	la22.fjs	10	15	150	2190.96	810
105	la23.fjs	10	15	150	2204.08	917
106	la24.fjs	10	15	150	2246.09	870
107	la25.fjs	10	15	150	2252.43	817
108	la26.fjs	10	20	200	3781.87	1109
109	la27.fjs	10	20	200	3737.50	1185
110	la28.fjs	10	20	200	3659.54	1190
111	la29.fjs	10	20	200	3655.36	1103
112	la30.fjs	10	20	200	3664.66	1157
113	la31.fjs	10	30	300	7673.38	1676
114	la32.fjs	10	30	300	7667.95	1866
115	la33.fjs	10	30	300	7683.07	1615
116	la34.fjs	10	30	300	7667.43	1630
117	la35.fjs	10	30	300	7707.73	1690
118	la36.fjs	15	15	225	4368.32	1178
119	la37.fjs	15	15	225	4360.37	1367
120	la38.fjs	15	15	225	4366.05	1084
121	la39.fjs	15	15	225	4359.40	1199
122	la40.fjs	15	15	225	4571.56	1112
123	mt06.fjs	6	6	36	235.94	47
124	mt10.fjs	10	10	100	1131.62	655
125	mt20.fjs	5	20	100	1253.36	1023
126	orb1.fjs	10	10	100	1130.54	708
127	orb2.fjs	10	10	100	1108.51	681
128	orb3.fjs	10	10	100	1112.15	648
129	orb4.fjs	10	10	100	1115.34	753
130	orb5.fjs	10	10	100	1098.15	638
131	orb6.fjs	10	10	100	1096.29	715
132	orb7.fjs	10	10	100		
133	orb8.fjs	10	10	100	1109.25	573
134	orb9.fjs	10	10	100	1122.88	662
135	orb10.fjs	10	10	100	1108.79	681
136	Mesghouni1997	10	10	30	199.38	10
137	Kacem2002	8	8	27	174.72	14