ALGORITMO GENÉTICO PARA EL PROBLEMA DE JOB SHOP FLEXIBLE

GENETIC ALGORITHM FOR THE FLEXIBLE JOB SHOP PROBLEM

Rosa Medina Durán[1] **Lorena Pradenas Rojas**[2] **Víctor Parada Daza**[3]

RESUMEN

El problema de Job Shop Flexible es parte de la familia de los problemas de scheduling. Nace como una generalización del problema de Job Shop, para optimizar el uso de los recursos en sistemas de producción con una mayor flexibilidad, por ejemplo la capacidad de una máquina de realizar más de un tipo de operación. Este problema ha sido estudiado por numerosos autores, que han propuesto diversos modelos matemáticos y enfoques heurísticos. Debido a la naturaleza combinatorial de este problema, los métodos exactos que resuelven los modelos matemáticos logran resolver sólo instancias pequeñas. Entre los métodos heurísticos propuestos, las metaheurísticas de búsqueda local han presentado un mejor desempeño. En este estudio se presenta un algoritmo genético secuencial para resolver el problema del Job Shop Flexible. El algoritmo propuesto se prueba utilizando las instancias de la literatura. Los resultados muestran la efectividad del algoritmo para encontrar buenas soluciones al problema.

Palabras clave: Job Shop Flexible, Algoritmos Genéticos, Scheduling.

ABSTRACT

The Flexible Job Shop Problem is part of the family of scheduling problems. It extends the job shop problem in order to optimize the use of the resources in flexible production systems, for example those with machines that can process more than one type of operations. This problem has been studied for many authors, proposing mathematical models and heuristics approaches. Due to its combinatorial complexity, the exact methods that solve the mathematical models only solve small instances. Among the heuristics approaches, the metaheuristics of local search have demonstrated a better performance. The proposed algorithm is tested using literature instances. The results show that the algorithm is effective for finding good solutions for the problem.

Keywords: Flexible Job Shop Problem, Genetic Algorithms, Scheduling.

INTRODUCCIÓN

En las industrias de manufactura, la planeación de la producción debe decidir la asignación de los recursos a las tareas para optimizar uno o más objetivos. 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, por ejemplo, el tiempo de completación de la última operación, makespan.

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; por lo cual, también se requiere decidir cual de las máquinas procesa cada operación.

Brandimarte [1], propone un enfoque jerárquico para la resolución del problema, separandolo en el subproblema de asignación y en el subproblema de secuenciamiento. El primer suproblema lo resuelve como un problema de ruteo, mientras que el segundo corresponde al problema de Job Shop. Para probar el algoritmo propone quince instancias del Job Shop Flexible. Mesghouni [9] corresponde a la primera publicación que resuelve el problema utilizando

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

algoritmos genéticos. Utilizan un enfoque integrado, representando la solución mediante matrices. Kacem [8] propone un enfoque por localización que permite encontrar buenas soluciones, minimiza el makespan y la carga de las máquinas. Este enfoque controla la evolución de un algoritmo genético.
Ho *et al.* en sus distintas publicaciones, estudian el problema del Job Shop Flexible, proponiendo reglas de despacho [5] que utilizan como solución inicial para una nueva metodología denominada GENACE [7], en la cual las poblaciones son influenciadas por reglas heurísticas y en cada generación se guía el espacio de búsqueda mediante esquemas. También, estudian las representaciones de la solución [11], para encontrar la más eficaz en términos de tiempo de cómputo y de mejor makespan. Este enfoque es generalizado en una publicación posterior [6], que propone una arquitectura basada en algoritmos evolutivos, aprendizaje con teoría de esquemas y generador de poblaciones mediante reglas de despacho compuestas. Fattahi [4] propone un modelo matemático, para probarlo generan diez instancias pequeñas y diez instancias medianas/grandes, pero logran resolver sólo las instancias pequeñas. También utilizando el enfoque jerárquico, combinan las metaheurísticas de Tabu Search y Simulated Annealing, 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 [13] propone un algoritmo genético basado en escenarios múltiples, dónde cada escenario corresponde a una operación y cada máquina factible a un estado. Pezzella [10] también propone un algoritmo genético; se utiliza el enfoque de localización propuesto por Kacem [8]. Proponen la mutación inteligente, que consiste en seleccionar una operación entre aquellas de la máquina con más carga de trabajo y reasignarla a la máquina con menos carga de trabajo. La publicación más reciente corresponde a Yazdani [12], se proponen un enfoque en paralelo de Variable Neighborhood Search con seis vecindarios para resolver el problema con el objetivo de minimizar el makespan.

En la próxima sección se define el problema, luego en la sección Algoritmo Propuesto se describe el algoritmo genético propuesto y en la sección Experimentos y Resultados se muestran los experimentos realizados y resultados obtenidos; finalmente, en la sección Conclusiones se presentan las conclusiones junto con sugerencias e ideas para trabajos futuros.

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, $J = \{J_1,...,J_n\}$. Cada trabajo, compuesto por una secuencia de operaciones $O_{\{j,1\}},O_{\{j,2\}},...,O_{\{j,h\}},...,O_{\{j,hj\}}$, cada una a procesar en una máquina. Y dados 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, 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 sistema 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 sistema 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).

Por cada operación, se conoce el tiempo de proceso 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 está permitido interrumpir las operaciones cuando se ha iniciado su proceso, como tampoco que las máquinas ejecuten más de una operación simultáneamente. Además, se asume que todos los trabajos y máquinas están disponibles en el tiempo cero.

Para encontrar el tiempo de completación de la última operación, makespan, es necesario asignar las operaciones a las máquinas y secuenciar las operaciones en las máquinas. Cabe destacar que el makespan corresponde al tiempo de completación, C_{Jj}, del último trabajo en el sistema, por lo cual, la función objetivo del problema se puede expresar como:

$$\min C_{max} = \min \{\max_{Jj \in J} C_{Jj}\}$$

ALGORITMO PROPUESTO

Los Algoritmos Genéticos son un método metaheurístico de optimización estocástica inspirado en la evolución natural de las especies, propuesto por Holland, 1975. 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 dar mayor claridad al algoritmo propuesto, se utiliza un pequeño ejemplo que corresponde a una instancia de FJSP-P [2] con dos trabajos y tres máquinas, 2×3, cada trabajo con tres operaciones, por lo cual el problema consta de seis operaciones. Los 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, cada solución del problema requiere tres vectores. El primero de ellos representa una solución para el subproblema de secuenciamiento, el segundo indica la operación a la cual corresponde cada casilla 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. En cada casilla aparece el número de un trabajo; el orden en que se encuentran define cual 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 se respeta la precedencia de la operaciones pertenecientes al mismo trabajo. En el ejemplo, una solución para el problema de secuenciamiento se muestra en la Figura 1. Los números de cada casilla corresponden a un número de trabajo, los cuales aparecen repetidos tres veces, ya que cada trabajo tiene tres operaciones, señaladas bajo cada casilla.

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

El cromosoma de operaciones es un vector similar al anterior pero en cada casilla 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 anterior.

$O_{1,1}$	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

El cromosoma de asignación también es un vector de las mismas dimensiones que los anteriores, pero cada casilla indica el índice de la máquina en la cual se procesa la operación que corresponde según los cromosomas anteriores. En el ejemplo, Figura 3, cada casilla está asociada con la operación indicada en la parte superior, de acuerdo al cromosoma de operaciones visto anteriormente. 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 casilla. 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 diferentes 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, con 20% de probabilidad, se aplica a ambos cromosomas; con 40% se aplica sólo al cromosoma de secuenciamiento y 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 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.

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 valida 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 aleatoria. Para generar un cromosoma de secuenciamiento, se escoge una posición aleatoria y se le asigna la primera operación; en la casilla a la derecha de esta, se asigna la segunda operación, así sucesivamente se completan todas las casillas; cuando se llega a la última casilla, 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 casilla, 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.

EXPERIMENTOS Y 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 instancias de la literatura, que se encuentran disponibles en internet o en las publicaciones, mientras que otras han sido solicitadas a los propios autores. 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 cuales son los valores óptimos (o intervalos de valores) para estos parámetros, con la finalidad de encontrar los mejores resultados. Se usa un diseño experimental [3], para lo cual se seleccionan 8 instancias (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.

Tabla 2: Valor inicial, Rango y Paso de Incremento para Parametrización

Parámetro	Valor inicial	Rango	Paso de Incremento
Tamaño de la Población	3000	[1000,5000]	500
Número de Generaciones	2000	[1000,3000]	500
Probabilidad de *Crossover*	0.70	[0.60,0.80]	0.05
Probabilidad de Mutación	0.20	[0.1,0.3]	0.05

De los resultados anteriores se concluye que el mejor grupo de parámetros en términos de makespan para el algoritmo son:

- Tamaño de la Población : 4500
- Número de Generaciones : 3000
- Probabilidad de *Crossover* : 0.75
- Probabilidad de Mutación : 0.30

Para cada instancia el número de máquinas, el número de trabajos, el número de operaciones totales, los tiempos de cómputo y makespan obtenidos se encuentran en el anexo A. 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 linea roja corresponde al makespan promedio de cada generación, mientras que la linea azul corresponde al mejor makespan de cada generación. Se observa que ambas secuencias se estabilizan luego de 1500 generaciones, aproximadamente, en un valor cercano a 1500.

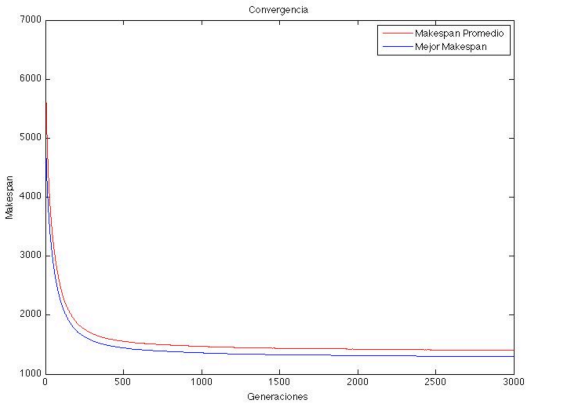

Figura 4:Gráfico Makespan v/s Generaciones

CONCLUSIONES

En esta investigación se propone un algoritmo genético para resolver el problema de Job Shop Flexible. El algoritmo propuesto es explicado en detalle e implementado en lenguaje C. Mediante un diseño de experimentos se determinan los parámetros del algoritmo. Se resuelven las instancias de la literatura, en un tiempo razonable para la etapa de planeación de la producción.

Como trabajo futuro, se sugiere comparar el desempeño del algoritmo con los resultados de la literatura. También sería interesante paralelizar el algoritmo, lo cual permite disminuir los tiempos de cómputo y mejorar las soluciones.

AGRADECIMENTOS

Este trabajo es parcialmente apoyado por el proyecto ALFA Nº II-0457-FA-FCD-FI-FC y por el proyecto DIUC-208.97011-1.

REFERENCIAS

[1] P. Brandimarte. “Routing and scheduling in a flexible job shop by tabu search”. Annals of Operations Research, Vol. 41 Num. 3 pp. 157-183. Septiembre 1993.

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

[3] S. Coy, B. Golden, G. Runger, E. Wasil. “Using experimental design to find effective parameter setting for heuristics”. Journal of Heuristics, Vol. 7 Issue 1 pp 77-97. Enero 2001.

[4] P. Fattahi, M. Meharab, F. Jolai. “Mathematical modeling and heuristic approaches to flexible job shop scheduling problems”. Jounal of Intelligent Manufacturing, Vol. 8 Num. 3 pp 331-342. Junio 2007.

[5] N. Ho, J. Tay. “Evolving dispatching rules for solving the flexible job-shop problem”. En: IEEE Congress on Evolutionary Computation. Vol. 3 pp 2848-2855 2005.

[6] N. Ho, J. Tay, E. Lai. “An effective architecture for learning and evolving flexible job-shop schedules”. European Journal of Operational Research, Vol. 179 Issue 2 pp 316-333. Junio 2007.

[7] N. Ho, J. Tay. “GENACE: An effective cultural algorithm for solving the flexible job-shop problem”. En: Lecture notes in IEEE, pp 1758-1766. 2004.

[8] I. Kacem, S. Hammadi, P. Borne. “Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems”. En: IEEE Transaction on Systems, Man, and Cybernetics, Vol. 32 Num. 1 pp 1-13. Febrero 2002.

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

[10] F. Pezzella, G. Morganti, G. Ciaschetti. "A genetic algorithm for the flexible job-shop scheduling problem". Computers & Operations Research, Vol. 35 Issue 10 pp 3202-3212. Octubre 2008.

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

[12] M. Yazdani, M. Zandieh, M. Amiri. "Flexible job-shop scheduling with parallel variable neighborhood search algorithm". Expert Systems with Applications, Vol. 37 Issue 1 pp 678-687. Enero 2010.

[13] H. Zhang, M. Gen. "Multistaged-based genetic algorithm for flexible job-shop scheduling problem". Complexity International, Volume 11, Paper ID: zhang01, URL: http://www.complexity.org.au/vol11/zhang01/

ANEXO A: RESULTADOS

En la Tabla 3 se presentan los resultados para las instancias del problema. La primera columna indica el nombre de la instancia, la segunda el número de máquinas, la tercera el número de trabajos, la cuarta el total de operaciones, la quinta el tiempo de cómputo del algoritmo propuesto y la última columna, el mejor makespan encontrado por el algoritmo. La instancia orb7.fjs no pudo ser resuelta debido a problemas en el archivo de lectura.

Tabla 3: Resultados Algoritmo

Nombre Instancia		Número de máquinas	Número de trabajos	Número de Operaciones	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

Continúa en la página siguiente

Tabla 3 Continuación de la página anterior

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
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

Continúa en la página siguiente

Tabla 3 Continuación de la página anterior						
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
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
Continúa en la página siguiente						

Tabla 3 Continuación de la página anterior						
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