Ingeniería Industrial/ISSN 1815-5936/Vol. XXXIII/No. 1/enero-abril/2012/p. 50-59

Artículo original
INVESTIGACIÓN DE OPERACIONES

TÉCNICAS EVOLUTIVAS EN PROBLEMAS MULTI-OBJETIVOS EN EL PROCESO DE PLANIFICACIÓN DE LA PRODUCCIÓN / *EVOLUTIONARY TECHNIQUES FOR MULTI-OBJECTIVE PROBLEMS IN PRODUCTION PLANNING*

Mariano Frutos-Alazard [I], ***Fernando Tohmé-Hauptmann*** [II]

[I] *Departamento de Ingeniería, Universidad Nacional del Sur. Bahía Blanca, Argentina.*
E-mail: mfrutos@uns.edu.ar
[II] *Departamento de Economía, Universidad Nacional del Sur. Bahía Blanca, Argentina.*
E-mail: ftohme@criba.edu.ar

Recibido: 14/04/2011
Aprobado: 06/12/2011

Resumen / *Abstract*
La planificación, en el ámbito productivo, se encarga de diseñar, coordinar, administrar y controlar todas las operaciones que se hallan presentes en la explotación de los sistemas productivos. En este marco de trabajo, aparecen numerosos Problemas de Optimización Multi-objetivo (MOPs). Éstos constan de varias funciones que suelen ser complejas y evaluarlas puede ser muy costoso. La optimización multi-objetivo es la disciplina que trata de encontrar las soluciones, denominadas Pareto óptimas, a este tipo de problemas. La compleja resolución de los MOPs es debida a las dimensiones propias del problema, al carácter combinatorio de los algoritmos y a la naturaleza de los objetivos, los cuales están vinculados a la eficiencia del sistema. En las últimas décadas muchos MOPs vinculados a la producción han sido tratados con éxito con técnicas de resolución basadas en Algoritmos Genéticos. En este trabajo se evalúa a NSGAII (*Non-dominated Sorting Genetic Algorithm* II), SPEAII (*Strength Pareto Evolutionary Algorithm* II) y a sus antecesores, NSGA y SPEA, en el proceso de planificación de la producción no estandarizada. Luego de la experiencia realizada, el algoritmo NSGAII mostró mayor eficiencia.

Planning in production environments takes care of designing, coordinating, managing and controlling all the operations existing in the use of productive systems. There are, in the framework analyzed within this work, several relevant Multi-Objective Optimization Problems (MOPs). They consist of several functions which tend to be complex and expensive to evaluate. Multi-objective optimization is the discipline developed to provide solutions, called Pareto optimal, for the simultaneous optimization of those functions. The costs of solving MOPs is due to the dimension of the problems, the combinatorial nature of the algorithms and the kind of objectives represented, linked to the efficiency of the system.. In the last decades several production-related MOPs have been handled successfully by means of Genetic Algorithms. Here we will evaluate the performance of some particular genetic-based algorithms like NSGAII (Non-dominated Sorting Genetic Algorithm II), SPEAII (Strength Pareto Evolutionary Algorithm II) and their predecessors, NSGA and SPEA, in the process of planning non-standardized production activities. After the experiment was carried out, the NSGAII algorithm proved to be more efficient.

Palabras clave */ Key words*
Algoritmo Memético Multi-objetivo, Configuración Productiva tipo Job-Shop, Frontera de Pareto, Optimización Multi-objetivo

Multi-objective Memetic Algorithm, Job-shop production environment, Pareto frontier, Multi-objective optimization

I. INTRODUCCIÓN

La programación de la producción en un ambiente productivo tipo *Job-Shop* implica asignar y dimensionar adecuadamente los recursos involucrados en el mismo [1; 2; 3; 4]. Para ello, es necesario contar con procedimientos eficientes que permitan optimizar decisiones en este ámbito. El problema de secuenciación de operaciones en un entorno *Job-Shop* es, por lo general, muy difícil de resolver. El mismo ha sido clasificado como *NP-Hard*, es decir, no se ha encontrado un algoritmo polinómico para resolverlo, por lo que el tiempo para encontrar una solución crece exponencialmente con respecto al tamaño del problema [5; 6]. Se han dado diversas propuestas sobre cómo plantear el problema, para posteriormente darle solución a través de una gran variedad de algoritmos [7; 8; 9; 10; 11]. La mayoría de estos problemas involucran la optimización simultánea de 2 ó más objetivos, los cuales, generalmente se encuentran en conflicto entre sí [12; 13]. Este tipo de problemas se denominan multi-objetivo y, por su naturaleza, suelen tener múltiples soluciones. Se incluyen en este artículo, algunos conceptos fundamentales sobre optimización multi-objetivo. En estas definiciones asumimos, sin pérdida de generalidad, la minimización de todos los objetivos.

A. Problema de Optimización Multi-objetivo: es encontrar un vector $\vec{x}^* = [x_1^*,...,x_n^*]^T$ que satisfaga las q restricciones de desigualdad $g_i(\vec{x}) \geq 0, i = 1,...,q$, las p restricciones de igualdad $h_i(\vec{x}) = 0, i = 1,..., p$, y que minimice la función vector $\vec{f}(\vec{x}) = [f_1(\vec{x}),...,f_k(\vec{x})]^T$; donde $\vec{x} = [x_1,...,x_n]^T$, es el vector de variables de decisión. El conjunto de todos los valores que satisfacen las restricciones define la región de soluciones factibles Ω y cualquier punto en $\vec{x} \in \Omega$ es una solución factible.

B. Optimalidad de Pareto: un punto $\vec{x}^* \in \Omega$ es un óptimo de Pareto si para cada $\vec{x} \in \Omega$, hay al menos una i para el cual $f_i(\vec{x}^*) \leq f_i(\vec{x})$. Esta definición dice que $\vec{x}^*$ es un óptimo de Pareto, si no existe ningún vector factible $\vec{x}$ que mejore algún objetivo sin causar simultáneamente un empeoramiento en, al menos, otro objetivo.

C. Dominancia de Pareto: un vector $\vec{u} = [u_1,...,u_n]^T$ se dice que domina a otro vector $\vec{v} = [v_1,...,v_n]^T$ (representado por $\vec{u} \prec \vec{v}$) si y sólo si $\vec{u}$ es parcialmente menor que $\vec{v}$; es decir, $\forall i \in \{1,...,k\}$, $u_i \leq v_i \wedge \exists i \in \{1,...,k\} : u_i < v_i$.

D. Conjunto Óptimo de Pareto: para un Problema de Optimización Multi-objetivo dado $\vec{f}(\vec{x})$, el Conjunto óptimo de Pareto se define como $P^* = \{\vec{x} \in \Omega \mid \neg \exists \vec{x}' \in \Omega, \vec{f}(\vec{x}') \prec \vec{f}(\vec{x})\}$.

E. Frontera de Pareto: Para un Problema de Optimización Multi-objetivo dado $\vec{f}(\vec{x})$ y su Conjunto Óptimo de Pareto P^*, la Frontera de Pareto se define como $FP^* = \{\vec{f}(\vec{x}), \vec{x} \in P^*\}$.

Obtener la frontera de Pareto es la meta principal de la optimización multi-objetivo. De todas maneras, dado que esta frontera puede contener un gran número de puntos, una buena solución debe contener un número limitado de ellos, localizados lo más cerca posible de la frontera de Pareto exacta, y estar uniformemente distribuidos a lo largo de la misma, para que sean de la mayor utilidad posible al experto que interpreta las soluciones.

Debido a sus diversas ventajas, el uso de algoritmos evolutivos se ha vuelto muy popular en optimización multi-objetivo [14; 15]. Entre los diversos algoritmos evolutivos que se han utilizado, uno de los más interesantes en su desarrollo y aplicación es el Algoritmo Genético (GA, *Genetic Algorithm*) [16; 17; 18]. La alta tasa de convergencia de éstos implica un costo en problemas multi-objetivo, ya que se pierde diversidad de soluciones, lo cual se refleja en fronteras de Pareto pobremente distribuidas. Sin embargo, si se complementa al algoritmo con un método de búsqueda local eficiente, es posible diseñar un algoritmo multi-objetivo que requiera un número muy bajo de evaluaciones de la función de aptitud. A este tipo de algoritmo combinado se lo llama Algoritmo Memético Multi-objetivo [19].

El resto del artículo está organizado de la siguiente manera. En la sección II se proporciona una descripción general del algoritmo propuesto. En la sección III se enuncian los algoritmos a comparar, se presentan las experiencias y se exponen los resultados obtenidos con cada uno de los procedimientos. Además, en este punto, se establece el procedimiento de comparación. Finalmente, en la sección IV, se presentan las conclusiones y algunos posibles trabajos futuros.

II. MÉTODOS

El problema de secuenciación de operaciones en un entorno *Job-Shop* (JSSP, *Job-Shop Scheduling Problem*) consiste en organizar la ejecución de n trabajos a ser realizados en m máquinas. Es decir, se tiene un conjunto finito J de trabajos

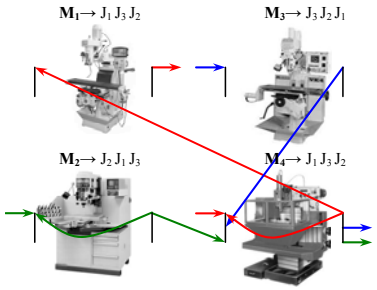
$\{J_j\}_{j=1}^{n}$, los cuales deben ser procesados por un conjunto finito *M* de máquinas $\{M_k\}_{k=1}^{m}$. Al procesamiento de una operación de un trabajo J_j en una máquina M_k se la denomina operación O_{jk}^{i}, donde *i* indica el orden en que deben ser realizadas el conjunto de operaciones de un mismo trabajo J_j. La operación O_{jk}^{i} requiere el uso de una máquina M_k durante un tiempo ininterrumpido τ_{jk}^{i}, conocido como tiempo de procesamiento.

En base a lo mencionado, el algoritmo propuesto opera en base a un cromosoma que representa la secuenciación de las operaciones O_{jk}^{i} a realizarse en cada una de las máquinas M_k. La codificación del mismo se presenta como una lista ordenada de números enteros, los cuales representan el orden en que los trabajos serán realizados en cada máquina [20], como se muestra en la Tabla 1. Es decir, se designa con valores entre *0* y *n!-1* la secuencia de trabajos en cada máquina. Para *m = 4* y *n = 3*, *0→$J_1J_2J_3$, 1→$J_1J_3J_2$, 2→$J_2J_1J_3$, 3→$J_2J_3J_1$, 4→$J_3J_1J_2$ y 5→$J_3J_2J_1$.*

TABLA 1
Proceso de codificación del cromosoma

Cromosoma / Codificación					
M_k	**Codificación**	**Secuencia**			
M_1	1	$J_1J_3J_2$			
M_2	2	$J_2J_1J_3$	**J_1(→)**	**J_2(→)**	J_3(→)
M_3	5	$J_3J_2J_1$			
M_4	1	$J_1J_3J_2$			

0→$J_1J_2J_3$, 1→$J_1J_3J_2$, 2→$J_2J_1J_3$, 3→$J_2J_3J_1$, 4→$J_3J_1J_2$ y 5→$J_3J_2J_1$



La población inicial es conformada por individuos cuyo cromosoma es constituido por genes seleccionados de manera aleatoria. Una vez establecido el cromosoma, se procede a la decodificación y evaluación de los individuos en términos del *Makespan* (tiempo en terminar todos los trabajos) [21] (Ecuación 1) y el *Delay* (tiempo de retardo del trabajo que termina con mayor retraso con respecto a una fecha consignada) [22] (Ecuación 2).

$$O_1 \rightarrow C_{máx} = máx\,(C_j) \tag{1}$$

$$O_2 \rightarrow T_{máx} = máx\,(T_j) \tag{2}$$

Donde C_j se corresponde con la fecha de culminación del trabajo J_j y T_j es el retraso del trabajo J_j con respecto a una fecha consignada. Poder evaluar los cromosomas en término de estos dos objetivos, permite determinar la criticidad de cada una de las máquinas que intervienen en el proceso productivo. El algoritmo opera con operadores genéticos básicos, gracias al diseño establecido para el cromosoma. Luego de que los individuos fueron afectados por los operadores genéticos y antes de que éstos formen parte de la nueva población, se les aplica a ellos un operador de mejora. Este operador se diseñó siguiendo el desarrollo de la meta-heurística *Simulated Annealing* (SA) [23]. Ésta es una técnica que presenta una búsqueda local guiada para progresar desde una solución a otra de mejor calidad. SA introduce una estrategia de aceptación que permite explorar temporalmente áreas desfavorables del espacio de búsqueda. Cabe mencionar que el esquema de la solución va cambiando con la alteración aleatoria del valor de los genes que conforman cada cromosoma. Se diferencia este procedimiento de otros al incorporar direcciones de búsqueda con criterios de densidad. Después de que los operadores genéticos y de mejora han sido aplicados, y la población ha sido decodificada y evaluada, el algoritmo continúa de acuerdo a NSGAII (*Non-Dominated Sorting Genetic Algorithm II*) [24]. NSGAII utiliza una estrategia elitista junto con un mecanismo explícito de diversidad. En la Figura 1 se presenta el seudo-código del algoritmo y en la Figura 2 su esquema general.

1. *Generar población inicial (P_0) de tamaño N*
2. *Decodificar y evaluar $O_1(x)$ y $O_2(x)$ a todo individuo $x \in P_0$*
3. *Asignar valores r_i y d_i a todo individuo $x \in P_0$*
4. *Seleccionar Padres de P_0*
5. *Q_0 = Cruzar (P_0)*
6. *Q'_0 = Mutar (Q_0)*
7. *Q''_0 = Búsqueda Local (Q'_0)*
8. ***for** $i = 0$ a $(G - 1)$ **do***
9. *Decodificar y evaluar $O_1(x)$ y $O_2(x)$ en todo individuo $x \in Q''_i$*
10. *Asignar valores r_i y d_i a todo individuo $x \in Q''_i$*
11. *Seleccionar de $P_i \cup Q''_i$ los N individuos mejores y eliminar el resto*
12. *Crear la siguiente generación P_{i+1}*
13. *Seleccionar Padres de P_{i+1}*
14. *Q_{i+1} = Cruzar (P_{i+1})*
15. *Q'_{i+1} = Mutar (Q_{i+1})*
16. *Q''_{i+1} = Búsqueda Local (Q'_{i+1})*
17. ***end for***
18. ***end***

Figura 1 *Seudo-código del algoritmo*

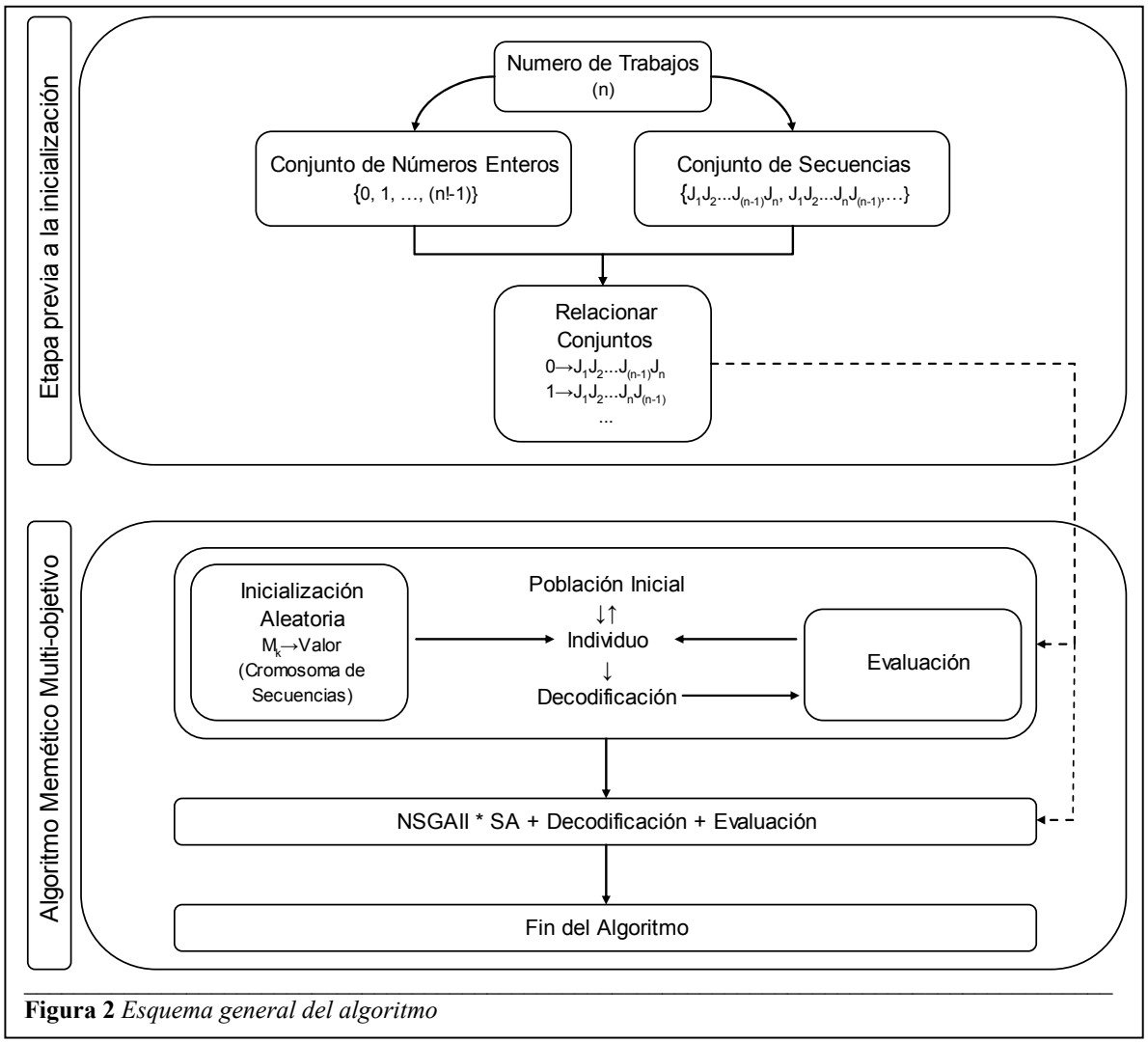


Figura 2 *Esquema general del algoritmo*

III. RESULTADOS

Se presentan, en este punto, las características principales de los 3 algoritmos evolutivos multi-objetivos a ser comparados con NSGAII. Los algoritmos son los siguientes: NSGA (*Non-dominated Sorting Genetic Algorithm*) [25], SPEA (*Strength Pareto Evolutionary algorithm*) [26] y SPEAII (*Strength Pareto Evolutionary algorithm II*) [27]. El algoritmo NSGA clasifica los individuos en varias capas o frentes. Esta clasificación consiste en agrupar a todos los individuos no dominados en un frente, con un valor de *fitness* igual para todos los individuos. Este valor es proporcional al tamaño de la población, para así proporcionar un potencial reproductivo igual para todos los individuos de este frente. Entonces, el grupo de individuos clasificados es ignorado y otro frente de individuos es considerado. El proceso continúa hasta que se clasifican a todos los individuos en la población. Puesto que los individuos en el primer frente tienen el valor de *fitness* mayor, consiguen siempre más atención que el resto de la población. SPEA, es un algoritmo que utiliza un archivo que contiene las soluciones no dominadas encontradas. En cada generación, se copian los individuos no dominados a ese archivo y se borran de éste las soluciones dominadas. Para cada individuo en el sistema externo, se computa un valor de fuerza o *strength* proporcional al número de las soluciones a las cuales cada individuo domina. En SPEA, el *fitness* de cada miembro de la población actual se computa según las fuerzas de todas las soluciones no dominadas externas que la dominen. SPEAII tiene diferencias importantes con respecto a su precursor. Este algoritmo incorpora una estrategia fina de asignación del *fitness* que considera, para cada individuo, el número de los individuos que lo dominan y el número de los individuos por los cuales es dominado. Además, utiliza la técnica del "vecino más cercano" para la valoración de la densidad, dirigiendo la búsqueda en forma más eficiente.

En un análisis preliminar de estos algoritmos, se verificó que el proceso de mejoramiento de soluciones tiende a estabilizarse en torno a la generación *200*, por lo que se definió el número de generaciones igual a *500*, dando así un margen para un mejoramiento tardío de la solución. Los parámetros y las características del equipamiento utilizado durante los ensayos fueron: tamaño de la población: *200*, número de generaciones: *500*, tipo de cruce: *Uniform*, probabilidad de cruce: *0.90*, tipo de mutación: *Two-swap*, probabilidad de mutación: *0.01*, tipo de búsqueda local: *Simulated Annealing*, probabilidad de búsqueda local: *0.01*, CPU: *3.00* GHZ y RAM: *1.00 GB*. Los problemas resueltos fueron: la01, la02, la03, la04, la05, la06, la07, la08, la09 y la10 [28]. Estos problemas son los más populares en cuanto al JSSP se refiere. Son variados en tamaño y complejidad. Se muestran los resultados obtenidos con los algoritmos presentados en el punto anterior para un análisis multi-objetivo haciendo *Makespan* vs. *Delay*. Los resultados se analizaron de la siguiente manera [23]. Se corrió cada algoritmo 10 veces. Se obtuvo para cada algoritmo los conjuntos de soluciones no dominadas: $P_1, P_2, ..., P_{10}$. Se creó una superpoblación $P_T = P_1 \cup P_2 \cup ... \cup P_{10}$ para cada uno de los algoritmos. De cada superpoblación se extrajeron las soluciones no dominadas, formando así el frente de Pareto de cada algoritmo: Y_{NSGA}, Y_{SPEA}, Y_{NSGAII} y Y_{SPEAII}. Es decir, se agrupan las soluciones de todas las réplicas y se eliminan las dominadas con el fin de estipular una frontera de Pareto para cada algoritmo. Los frentes de Pareto desarrollados por cada algoritmo se muestran de la Figura 3 a la Figura 7.

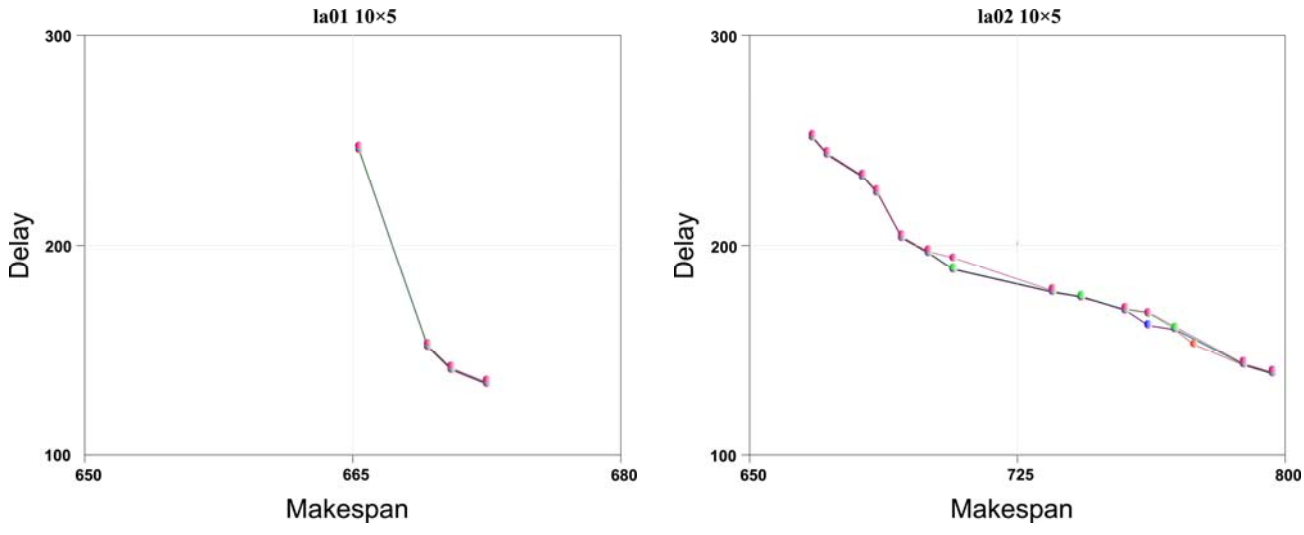


Figura 3 *Frentes de Pareto para la01 y la02.* Y_{NSGA}, Y_{SPEA}, Y_{NSGAII} *e* Y_{SPEAII}

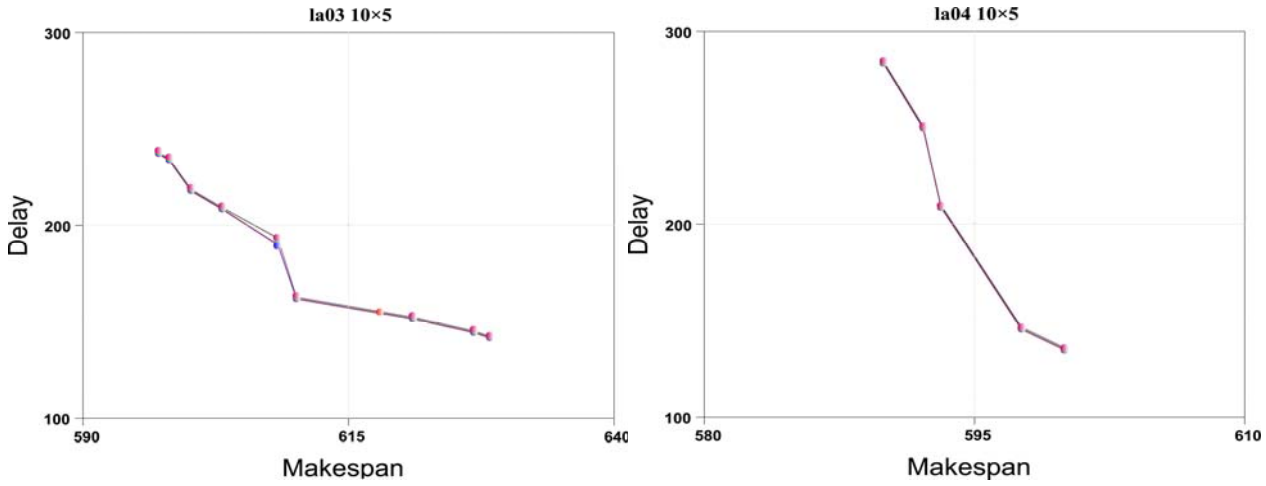


Figura 4 *Frentes de Pareto para la03 y la04.* Y_{NSGA}', Y_{SPEA}', Y_{NSGAII}' e Y_{SPEAII}'

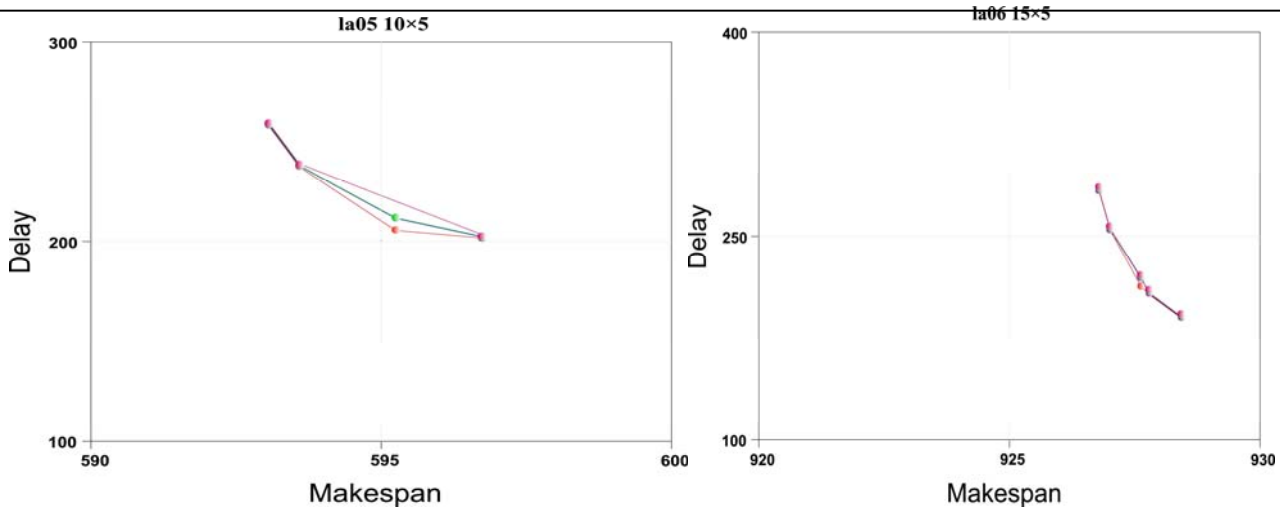


Figura 5 *Frentes de Pareto para la05 y la06.* Y_{NSGA}', Y_{SPEA}', Y_{NSGAII}' e Y_{SPEAII}'

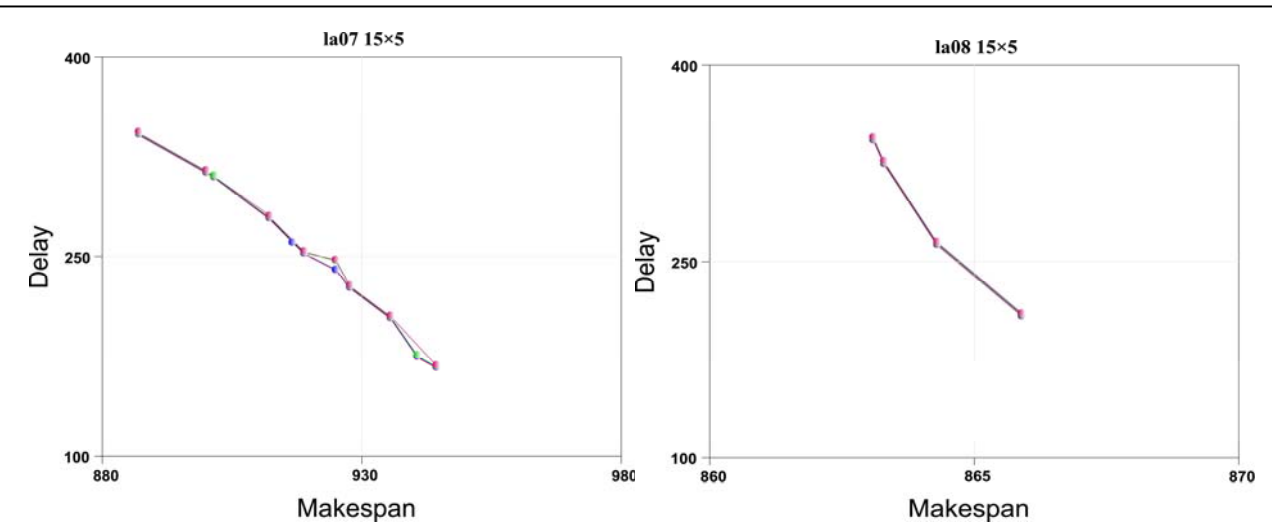


Figura 6 *Frentes de Pareto para la07 y la08.* Y_{NSGA}', Y_{SPEA}', Y_{NSGAII}' e Y_{SPEAII}'

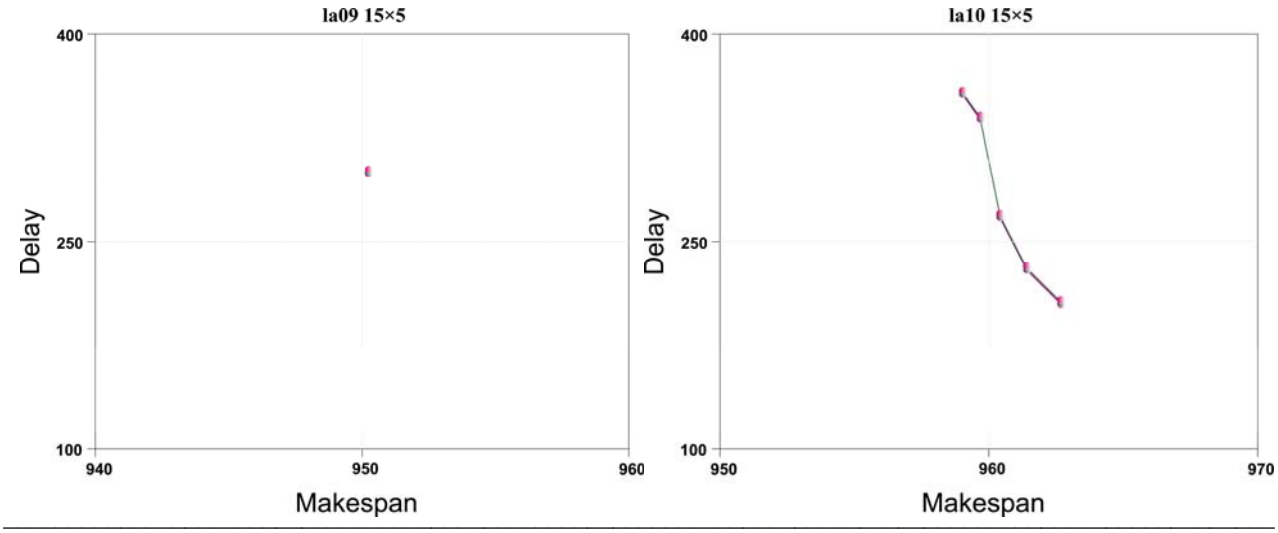


Figura 7 *Frentes de Pareto para la09 y la10.* Y_{NSGA}, Y_{SPEA}, Y_{NSGAII} *e* Y_{SPEAII}

IV. DISCUSIÓN

Para obtener una aproximación del frente Pareto óptimo $\hat{Y}_{true}$, se toma al conjunto de soluciones $Y_{NSGA} \cup Y_{SPEA} \cup Y_{NSGAII} \cup Y_{SPEAII}$, y se eliminan de él las soluciones dominadas. Para la comparación se tendrá en cuenta: la cantidad de soluciones de cada algoritmo que se encuentran en $\hat{Y}_{true}$ ($\in \hat{Y}_{true}$), la cantidad de soluciones que son dominadas por $\hat{Y}_{true}$ ($\hat{Y}_{true} \prec$), el número de soluciones encontradas por cada algoritmo ($\hat{Y}_{alg}$) y el porcentaje de soluciones en $\hat{Y}_{true}$ ($\% \in \hat{Y}_{true}$). En los problemas la01 (Tabla 2(a)), la04 (Tabla 3(b)), la08 (Tabla 5(b)), la09 (Tabla 6(a)) y la10 (Tabla 6(b)), se puede apreciar que NSGA, SPEA, NSGAII y SPEAII, poseen el 100% de sus soluciones en $\hat{Y}_{true}$. No existe dominancia de un algoritmo hacia otro. En el problema la02 (Tabla 2(b)), se observa que NSGAII posee el 100% de sus soluciones en $\hat{Y}_{true}$. Si bien SPEAII tiene el 100% de efectividad, no tiene a todas las soluciones de $\hat{Y}_{true}$. SPEA alcanza un 92,8% de efectividad, ya que de un total de 14 soluciones, solo 1 de ellas es dominada por $\hat{Y}_{true}$. Se observa que NSGAII y SPEAII dominan a NSGA en 2 soluciones y a SPEA en 1 solución. Además, SPEA domina a NSGA en 1 solución. En el problema la03 (Tabla 3(a)), se observa que NSGAII posee el 100% de sus soluciones en $\hat{Y}_{true}$. Si bien SPEAII tiene el 100% de efectividad, no tiene a todas las soluciones de $\hat{Y}_{true}$. Se observa que NSGAII y SPEAII dominan a NSGA y a SPEA en 1 solución.

En el problema la05 (Tabla 4(a)), se aprecia que NSGAII posee el 100% de sus soluciones en $\hat{Y}_{true}$. Si bien NSGA tiene el 100% de efectividad, no tiene a todas las soluciones de $\hat{Y}_{true}$. Se observa que NSGAII domina a SPEA y a SPEAII en 1 solución. En el problema la06 (Tabla 4(b)), se observa que solo NSGAII posee el 100% de sus soluciones en $\hat{Y}_{true}$. Se aprecia que NSGAII domina a NSGA, a SPEA y a SPEAII en 1 solución. En el problema la07 (Tabla 5(a)), se observa que SPEAII posee el 100% de sus soluciones en $\hat{Y}_{true}$. Si bien NSGAII tiene el 100% de efectividad, no tiene todas las soluciones de $\hat{Y}_{true}$. Se observa que NSGAII y SPEAII dominan a NSGA y a SPEA en 1 solución.

TABLA 2
Comparación de las soluciones con $\hat{Y}_{true}$ (*la01* y *la02*)

Problemas la01 (10x5) y la02 (10x5) / Makespan vs. Delay

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	4	0	4	100
Y_{SPEA}	4	0	4	100
Y_{NSGAII}	4	0	4	100
Y_{SPEAII}	4	0	4	100

(a)

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	10	2	12	83,3
Y_{SPEA}	13	1	14	92,8
Y_{NSGAII}	15	0	15	100
Y_{SPEAII}	14	0	14	100

(b)

TABLA 3
Comparación de las soluciones con $\hat{Y}_{true}$ (*la03* y *la04*)

Problemas la03 (10x5) y la04 (10x5) / Makespan vs. Delay

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	8	1	9	88,8
Y_{SPEA}	8	1	9	88,8
Y_{NSGAII}	10	0	10	100
Y_{SPEAII}	9	0	9	100

(a)

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	5	0	5	100
Y_{SPEA}	5	0	5	100
Y_{NSGAII}	5	0	5	100
Y_{SPEAII}	5	0	5	100

(b)

TABLA 4
Comparación de las soluciones con $\hat{Y}_{true}$ (*la05* y *la06*)

Problemas la05 (10x5) y la06 (15x5) / Makespan vs. Delay

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	3	0	3	100
Y_{SPEA}	3	1	4	75
Y_{NSGAII}	4	0	4	100
Y_{SPEAII}	3	1	4	75

(a)

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	4	1	5	80
Y_{SPEA}	4	1	5	80
Y_{NSGAII}	5	0	5	100
Y_{SPEAII}	4	1	5	80

(b)

TABLA 5
Comparación de las soluciones con $\hat{Y}_{true}$ (*la07* y *la08*)

Problemas la07 (10x5) y la08 (15x5) / Makespan vs. Delay

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	7	1	8	87,5
Y_{SPEA}	9	1	10	90
Y_{NSGAII}	10	0	10	100
Y_{SPEAII}	11	0	11	100

(a)

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	4	0	4	100
Y_{SPEA}	4	0	4	100
Y_{NSGAII}	4	0	4	100
Y_{SPEAII}	4	0	4	100

(b)

TABLA 6
Comparación de las soluciones con $\hat{Y}_{true}$ (*la09* y *la10*).

Problemas la09 (10x5) y la10 (15x5) / Makespan vs. Delay

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	1	0	1	100
Y_{SPEA}	1	0	1	100
Y_{NSGAII}	1	0	1	100
Y_{SPEAII}	1	0	1	100

(a)

	$\in \hat{Y}_{true}$	$\hat{Y}_{true} \prec$	$\hat{Y}_{alg}$	$\% \in \hat{Y}_{true}$
Y_{NSGA}	5	0	5	100
Y_{SPEA}	5	0	5	100
Y_{NSGAII}	5	0	5	100
Y_{SPEAII}	5	0	5	100

(b)

V. CONCLUSIONES

Se comparó al algoritmo NSGAII con otros algoritmos evolutivos multi-objetivo. Los casos abordados fueron problemas de secuenciación de operaciones encontrados en la literatura especializada. Se recuerda que este tipo de problema es uno de los *NP-Hard* más estudiado en la literatura y su tratamiento es un tema de investigación actual. Se pudo observar que, en la mayoría de los problemas, la calidad de las soluciones obtenidas con NSGAII respecto a las alcanzadas con SPEAII, fue igual o superior. NSGAII superó a NSGA y a SPEA. Se puede concluir que el algoritmo propuesto tiene muy buen desempeño, siendo una buena opción para resolver este problema. Como parte de trabajos futuros, se podrían estudiar planteamientos multi-objetivo distintos de este problema. Se está experimentando con otras técnicas eficientes de búsqueda local para lograr una exploración más agresiva. Finalmente, resultaría de interés evaluar este procedimiento en otros tipos de problemas y así verificar si el esquema propuesto realmente produce ahorros importantes en el procesamiento, sin sacrificar de manera significativa la convergencia. Sin lugar a dudas, queda mucho trabajo que se puede realizar para tener un algoritmo más eficiente y robusto.

VI. REFERENCIAS

1. ADAMS, J.; BALAS, E.; ZAWACK, D., «The Shifting Bottleneck Procedure for job shop scheduling» *Management Science*, 1998, vol. 34, pp. 391-401, ISSN 0025-1909.
2. DE GIOVANNI, L.; PEZZELLA F., «An improved genetic algorithm for the distributed and flexible job-shop scheduling problem» *European Journal of Operational Research*, 2010, vol. 200, no. 2, pp. 395-408, ISSN 0377-2217.
3. PARK, B. J.; CHOI H. R.; KIM, H. S. A., «Hybrid genetic algorithm for the job shop scheduling problems» *Computers and Industrial Engineering*, 2003, vol. 45, no. 4, pp. 597-613, ISSN 0360-8352.
4. TSAI, C. F.; LIN, F. C., «A new hybrid heuristic technique for solving job-shop scheduling problem», en *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. Second IEEE International Workshop* Lviv, Ucrania, 2003, ISBN 80-967402-6-1.
5. PAPADIMITRIOU, C. H., *Computational complexity*, USA, Addison Wesley, 1994, ISBN 0-201-53082-1.
6. ULLMAN, J. D., «Np-complete scheduling problems» *Journal of Computer and System sciences*, 1975, vol. 10, pp. 384-393, ISSN 0022-0000.
7. DELLA CROCE, F.; GROSSO, A.; SALASSA, F., «A matheuristic approach for the two-machine total completion time flow-shop problem» *Annals of Operations Research*, 2011, July, ISSN 0254-5330.
8. HEINONEN, J.; PETTERSSON, F., «Hybrid ant colony optimization and visibility studies applied to a job shop scheduling problem» *Applied Mathematics and Computation*, 2007, vol. 187, no.2, pp. 989-998, ISSN 0096-3003.
9. LIN, Y.; PFUND, M.; FOWLER, J., «Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems» *Computers & Operations Research*, 2011, vol. 38, no. 6, pp. 901-916, ISSN 0305-0548.
10. MERKLE, D.; MIDDENDORF, M., «A new approach to solve permutation scheduling problems with ant colony optimization», en *Applications of Evolutionary Computing. EvoWorkshops* Italia, 2001, vol. 2037, pp. 484-494. ISBN 3-540-41920-9.
11. WU, C. G.; XING, X. L.; LEE, H. P.; ZHOU, C. G.; LIANG, Y. C. , «Genetic Algorithm Application on the Job Shop Scheduling Problem», en *Machine Learning and Cybernetics* Shanghai, Proceedings of 2004 International Conference, 2004, vol. 4, pp. 2102- 2106. ISBN 0-780-38403-2.
12. FRUTOS, M.; TOHMÉ, F., «Desarrollo de un procedimiento genético diseñado para programar la producción en un sistema de manufactura tipo job-shop», en *VI Congreso Español sobre Meta-heurísticas, Algoritmos Evolutivos y Bioinspirados* Málaga, 2009, ISBN 974-84-691-6813-4.
13. STEINER, G.; ZHANG, R., «Minimizing the weighted number of tardy jobs with due date assignment and capacity-constrained deliveries» *Annals of Operations Research*, 2011, vol. 191, no. 1, pp. 171-181, ISSN 0254-5330.
14. COELLO, C. A.; VAN VELDHUIZEN, D. A., LAMONT, G. B., *Evolutionary Algorithms for Solving Multi-Objective Problems*, New York, Kluwer Academic Publishers, 2002, ISBN 0-306-46762-3.
15. CORTÉS, D.; COELLO, C. A.; CORTÉS, N. C., «Use of an Artificial Immune System for Job Shop Scheduling», en *Second International Conference on Artificial Immune Systems* Edinburgh (Scotland), Springer-Verlag. Lecture Notes in Computer Science, 2003, vol. 2787, pp. 1-10. ISBN 3-540-40766-9.
16. CHAO-HSIEN, J.; HAN-CHIANG, H., «A hybrid genetic algorithm for no-wait job-shop cheduling problems» *Expert Systems with Applications*, 2009, vol. 36, no. 3, pp. 5800-5806, ISSN 0957-4174.
17. CHINYAO, L.; YULING, Y., «Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated» *Robotics and Computer-Integrated Manufacturing*, 2009, vol. 25, no. 2, pp. 314-322, ISSN 0736-5845.
18. GOLDBERG, D. E., «Genetic Algorithms in Search», *Optimization and Machine Learning*, Massachusetts, Addison Wesley, 1989, ISBN 0-201-15767-5.

19. ISHIBUCHI, H.; YOSHIDA, T., MURATA, T., «Balance Between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flow-shop Scheduling» *IEEE Transactions on Evolutionary Computation*, 2003, vol. 7, no. 2, pp. 204-223, ISSN 1089-778X.
20. FRUTOS, M.; OLIVERA, A. C., TOHMÉ, F., «A Memetic Algorithm Based on a NSGAII Scheme for the Flexible Job-Shop Scheduling Problem» *Annals of Operations Research*, 2010, vol. 181, pp. 745-765, ISSN 0254-5330.
21. CHENG, C. C.; SMITH, S. F., «Applying constraint satisfaction techniques to job shop scheduling» *Annals of Operations Research*, 1997, vol. 70, pp. 327-357, ISSN 0254-5330.
22. ARMENTANO, V. A.; SCRICH, C. R., «Tabu search for minimizing total tardiness in a job shop» *Int. J. Production Economics*, 2000, vol. 63, pp. 131-140, ISSN 0925-5273.
23. DOWSLAND, K. A., *Simulated Annealing, Modern Heuristic Techniques for Combinatorial Problems*, Oxford, C. R. Reeves. Blackwell Scientific Pub, 1993, ISBN 1-55860-260-7.
24. DEB, K., PRATAP, A., AGARWAL, S.; MEYARIVAN, T., «A Fast and Elitist Multi-objective Genetic Algorithm: NSGAII» *IEEE Transactions on Evolutionary Computation*, 2002, vol. 6, no. 2, pp. 182-197, ISSN 1089-778X.
25. SRINIVAS, N., «Multiobjetive optimization using nondominated sorting in genetic algorithms», [master thesis], Kuanpur (India), Indian Institute of Technology, 1994.
26. ZITZLER, E.; THIELE, L., «Multiobjective Evolutionary Algorithms: A comparative Case Study and the Strength Pareto Approach» *IEEE Transactions on Evolutionary Computation*, 1999, vol. 3, no. 4, pp. 257-271, ISSN 1089-778X.
27. ZITZLER, E.; LAUMANNS, M.; THIELE, L., «SPEAII: Improving the strength pareto evolutionary algorithm for multiobjective optimization», *Evolutionary Methods for Design. Optimisations and Control*, Athens (Greece), Giannakoglou, Tsahalis, Periaux, Papailiou, and Fogarty, 2002, pp. 19-26, ISBN 84-89925-97-6.
28. BEASLEY, J. E., «OR-Library: Distributing Test Problems by Electronic Mail» *Journal of the Operations Research Society*, 1990, vol. 41, no. 11, pp. 1069-1072, ISSN 0160-5682.