Algoritmos *GRASP* para solucionar el problema *Blocking Flow Shop*

Joaquín Bautista, Alberto Cano, Rocío Alfaro y Cristina Batalla

Universitat Politècnica de Catalunya.
Avenida Diagonal 647, 7 planta. 08028 Barcelona, Spain
{joaquin.bautista,alberto.cano-perez,rocio.alfaro,
cristina.batalla}@upc.edu
http://www.prothius.com

Resumen Se resuelve el *Blocking Flow Shop Problem* (*BFSP*), consiste en determinar una secuencia de piezas procesadas por una línea de máquinas sin espacios de almacenamiento entre ellas. Para ello, se implementan 14 algoritmos heurísticos *Greedy Randomized Adaptive Search Procedure* (*GRASP*) aplicados a las 60 primeras instancias de Taillard. Los resultados obtenidos con los 7 procedimientos que incorporan la heurística de Nawaz son los más competitivos cuando se comparan con otros los de la literatura.

Keywords: GRASP; Algoritmos de secuenciación; secuencias; Blocking Flow Shop.

1. Introducción

El *Flow Shop Scheduling Problem* (*FSP*) es uno de los problemas que ha recibido más atención en los últimos 50 años por parte de profesionales e investigadores debido a la gran variedad de contextos productivos que permite modelar. En el *FSP*, un conjunto I de n de trabajos o piezas debe ser procesado en un conjunto K de m de máquinas. Todos los trabajos deben pasar por todas las máquinas en el mismo orden, empezando en la máquina 1 y finalizando en la máquina m. Cada trabajo, $i \in I$, requiere un tiempo de proceso, $p_{i,k} > 0$, en cada una de las máquinas, $k \in K$. El objetivo es encontrar una secuencia de proceso de los trabajos que optimice un criterio de eficiencia.

En la versión más popular del problema, conocida como *Permutation Flow Shop Problem* (*PFSP*), el espacio de almacenamiento entre dos fases consecutivas del proceso, donde las piezas pueden esperar hasta que puedan ser procesados por la siguiente máquina, es ilimitado. Sin embargo, existen sistemas productivos, en los sectores químico, farmacéutico, modelado de plástico, electrónica, metalurgía y alimentación, en los que existen líneas de producción en las que el espacio de almacenamiento está limitado. Si se asume que no existe espacios de almacenamiento entre dos fases consecutivas del proceso, entonces se produce un gran cambio estructural en el comportamiento del sistema, debido a que una pieza no puede abandonar la máquina que lo está procesando hasta que la siguiente

máquina esté libre. En este caso, el trabajo debe permanecer en la máquina previa, bloqueando la máquina e impidiendo a ésta realizar otros trabajos.

Esta variante se conoce como *Blocking Flow Shop Problem* (*BFSP*) y es la que consideramos en este documento, teniendo como objetivo minimizar el instante de finalización de todos trabajos en el taller (*makespan*, C_{max}). Haciendo uso de la notación propuesta por Graham [1], el problema considerado se conoce como $Fm|block|C_{max}$ (y el *PFSP* como $Fm|prmu|C_{max}$).

Dada la naturaleza *NP-difícil* del problema, su resolución emplea heurísticas, debido a su capacidad de encontrar soluciones de calidad en un tiempo reducido. En la literatura se encuentra el procedimiento de Nawaz, Enscore y Ham (*NEH*) [2], y más recientemente, se encuentra Ronconi [3], que propone dos variantes de éste, y Bautista [4] donde se propone un algoritmo basado en *programación dinámica acotada* (*BDP*) y una lista actualizada de las mejores soluciones para las instancias de Taillard.

Para este trabajo se ha diseñado un procedimiento *GRASP extendido* (*GRASP-x*) que admite diversas variantes en función de los valores asignados a tres parámetros. Entre dichas variantes se encuentran las heurísticas *Greedy* constructivas con posterior optimización local, los procedimientos *Multistart*, los algoritmos *GRASP clásicos* [5,6] y los algoritmos *GRASP* en los que la probabilidad de selección de los candidatos se hace depender de la aptitud de éstos, la cual puede medirse a través de una función de calidad dependiente de cada candidato. Una extensa recopilación de trabajos sobre aplicaciones se puede encontrar en [7,8].

Nuestra propuesta contiene: (1) el diseño e implementación de cotas parciales para el problema; (2) siete algoritmos basados en el procedimiento *GRASP* que actúan como maestros para dirigir la exploración en el espacio de búsqueda; y (3) una experiencia computacional, con ejemplares de la literatura, para comparar los resultados obtenidos mediante los procedimientos implementados con los mejores resultados encontrados en la literatura.

Este trabajo se organiza de la siguiente forma: la sección 2 presenta la descripción del problema; la sección 3 contiene el diseño de cotas parciales y globales para el problema; en la sección 4 se describe el procedimiento *GRASP-x* adaptado para resolver el *BFSP*; la sección 5 se centra en la descripción y comparación de resultados de la experiencia computacional realizada, que explota siete algoritmos (*derivados del procedimiento GRASP-x tras fijar siete conjuntos de valores a los tres parámetros*) sobre ejemplares de la literatura; finalmente, la sección 6 muestra algunas conclusiones sobre el presente trabajo.

2. Descripción del problema

A partir del instante 0 se deben procesar n trabajos, en el mismo orden, en m máquinas. Los tiempos de proceso para cada operación se denominan $p_{i,k}$, donde k denota una máquina y i un trabajo. Los tiempos de proceso están fijados y son positivos. La función objetivo considerada es la minimización del *makespan* (C_{max}), que es el instante en que el taller finaliza todas sus operaciones.

En un programa factible asociado a la permutación π, definimos $s_{k,t}$ como el instante de inicio del proceso destinado en la máquina k al trabajo que ocupa la posición t y $e_{k,t}$ como el instante en que el trabajo que ocupa la posición t abandona la máquina k. El problema $Fm|prmu|C_{max}$ puede ser formalizado como sigue:

$$s_{k,t} + p_{[t],k} \leqslant e_{k,t} \qquad k = 1, ..., m; t = 1, ..., n \tag{1}$$

$$s_{k,t} \geqslant e_{k,t-1} \qquad k = 1, ..., m; t = 1, ..., n \tag{2}$$

$$s_{k,t} \geqslant e_{k-1,t} \qquad k = 1, ..., m; t = 1, ..., n \tag{3}$$

$$C_{max} = e_{m,n} \tag{4}$$

Siendo $p_{[t],k}$ el tiempo de proceso en la máquina k de la pieza que ocupa la *t-ésima* posición en la secuencia π. Además $e_{k,0} = 0 \quad \forall k$ y $e_{0,t} = 0 \quad \forall t$ son las condiciones iniciales.

El programa obtenido es semi-activo si la restricción (1) se escribe como $s_{k,t} + p_{[t],k} = e_{k,t}$ y las restricciones (2) y (3) se resumen como $s_{k,t} = max\{e_{k,t-1}, e_{k-1,t}\}$.

Cuando no existen espacios de almacenamiento entre etapas, caso del problema $Fm|block|C_{max}$, si el trabajo i finaliza su operación en la máquina k y la próxima máquina, $k+1$, se encuentra todavía ocupada con el trabajo anterior, el trabajo completado i ha de permanecer en la máquina k, bloqueándola. Esta condición requiere una restricción adicional (5) en la formulación del problema.

$$e_{k,t} \geqslant e_{k+1,t-1} \qquad k = 1, ..., m; t = 1, ..., n \tag{5}$$

siendo necesario añadir la condición inicial $e_{m+1,t} = 0$, $t = 1, ..., n$.

En el caso $Fm|block|C_{max}$ el programa obtenido es semi-activo si la restricción (1) y (5) se resumen como en (6):

$$e_{k,t} = max\{s_{k,t} + p_{[t],k}, e_{k+1,t-1}\} \qquad k = 1, ..., m; t = 1, ..., n \tag{6}$$

Consecuentemente, el problema $Fm|prmu|C_{max}$ puede ser visto como una relajación del problema $Fm|block|C_{max}$.

3. Acotando los valores de las secuencias

Asumamos que se ha construido una subsecuencia $\pi(t) = \{\pi_1, \pi_2, ..., \pi_t\}$ de t trabajos.

Supongamos ademas que disponemos de la información $x_i(t)$ y $e_k(t)$; donde $x_i(t)$ adopta el valor 1 si el trabajo i está presente en $\pi(t)$ y $e_k(t)$ representa el instante en que queda libre la máquina k tras procesar todos los trabajos contenidos en $\pi(t)$. El esquema de acotación empleado se muestra en la figura 1.

Para completar todos los trabajos, nos faltará añadir a la secuencia $\pi(t)$ un segmento conteniendo los $n - t$ trabajos pendientes no contenidos en $\pi(t)$. Si

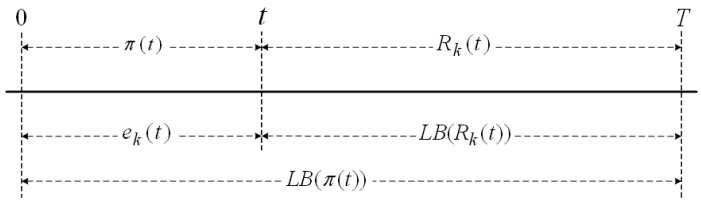

Figura 1. Esquema de acotación.

$R_k(t)$ es una secuencia de los trabajos pendientes en la máquina $k \in K$, es fácil obtener una cota inferior, $LB(R_k(t))$, para dicha secuencia. En efecto:

$$LB(R_k(t)) = \sum_{i \notin \pi(t)} p_{i,k} + \underset{i \in \pi(t)}{min} \left\{ \sum_{\kappa=k+1}^{m} p_{i,\kappa} \right\} \tag{7}$$

Y, por consiguiente, una cota global de C_{max} asociada a la secuencia $\pi(t)$ será:

$$LB(\pi(t)) = \underset{k \in K}{max} \{e_k(t) + LB(R_k(t))\} \tag{8}$$

4. Algoritmos *GRASP* para el *BFSP*

4.1. Preliminares

La complejidad del *BFSP* y el interés de obtener soluciones para ejemplares del problema con dimensiones industriales, hace recomendable el uso de procedimientos heurísticos capaces de ofrecer soluciones aceptables con bajo esfuerzo de computación.

La metaheurística *GRASP* [5,6] es de tipo multi-arranque y está provista de 2 fases en cada iteración: (1) un procedimiento *Greedy* que sirve para construir una solución aceptable sin que sea preciso alcanzar el óptimo global; y (2) una segunda fase para obtener un óptimo local dentro de un vecindario y teniendo como punto de partida la solución que resulta al aplicar el procedimiento *Greedy* de la fase_1. Obviamente, la solución ofrecida por el *GRASP* es la mejor entre las obtenidas en el conjunto de iteraciones.

Para la primera fase *Greedy* es importante definir un buen procedimiento que pueda ofrecer soluciones aceptables y una diversidad suficiente de soluciones que permitan explorar diferentes regiones en el espacio de soluciones. Para garantizar dicha diversidad se emplea el azar, de manera que el siguiente elemento a añadir a una solución parcial se sortea entre los elementos de una lista restringida de candidatos (*RCL*); dicha lista contiene los candidatos que presentan los mejores valores en relación a una función (*una cota para* C_{max}*, por ejemplo*) diseñada para la selección.

Para resolver un problema de optimización mediante un procedimiento *GRASP* es preciso definir los siguientes elementos: (1) el proceso aleatorio empleado en

la selección entre candidatos y el procedimiento *Greedy*; (2) el vecindario de una solución y, lógicamente, el procedimiento para explorar dicho vecindario; y (3) el criterio de finalización del algoritmo, normalmente vinculado a un máximo número de iteraciones o al tiempo de ejecución.

En la figura 2 se presenta un esquema general del algoritmo *GRASP*.

1. Inicialización
2. Mientras tiempo $CPU \leqslant CPU_{max}$
   2.1. Solución ⟵ Fase_constructiva (*Semilla*)
   2.2. Solución ⟵ Mejora_local (*Solución*)
   2.3. Actualizar_solución (*Solución, Mejor_solución*)
3. Salida: *Mejor_solución*

Figura 2. Esquema general de la metaheurística *GRASP*.

En algunos casos, al procedimiento *GRASP*, se le puede añadir un post-proceso que permita combinar las soluciones generadas [9].

4.2. Fase *Greedy* de construcción de una solución

El procedimiento implementado para esta fase del algoritmo (*ver figura 3*) construye progresivamente una secuencia seleccionando, en cada etapa asociada con el instante, $t = 1, ..., T = n$, un elemento candidato a partir de una lista restringida de éstos (*sea RCL*). En efecto, llegados a la etapa t, en la que se dispone de una secuencia (*solución*) parcial $\pi(t)$, para cada trabajo i no procesado aún, se determina el índice f_i ($\forall i : x_i = 0$) a partir del valor de la cota $LB(\pi(t) \cup \{i\})$; tras ello, la lista de candidatos se construye en 2 pasos:

1. En el primero, a partir del parámetro $Z \in [0, 1]$ denominado impedancia, se seleccionan todos los trabajos con un valor de cota no superior a $1/Z$ veces el valor de la menor de ellas (*cota correspondiente al mejor candidato*).
2. En el segundo paso, se seleccionan como máximo los L mejores candidatos (*ordenados por cota, de menor a mayor*) incluyendo en la lista, claro está, los trabajos empatados en cota con el *L-ésimo* candidato.

Nótese que gracias a la impedancia se pueden descartar soluciones peores a las de un valor de referencia, dejando en *RCL* menos soluciones que L. Además, la formulación propuesta en la figura 3 comprende como casos particulares los siguientes procedimientos: (1) los algoritmos *GRASP* tradicionales con tratamiento de empates incorporado, pues basta hacer $Z \to 0$, $f_0 \to \infty$, $\eta = 1$ y fijar un valor de L menor que el número de candidatos; (2) *Multistart*, haciendo $Z \to 0$, $f_0 \to \infty$, $\eta = 1$ y fijar un valor de L igual al número de candidatos; y (3) las heurísticas *Greedy* con tratamiento de empates con $Z = 1$; si $Z \to 0$ y L es suficientemente grande ($L = |I| = n$) todos los elementos compatibles

son candidatos y toda solución del *BFSP* tiene una probabilidad no nula de ser generada por el procedimiento.

---

0. Inicialización

Leer: $n = T$, I, K, $p_{i,k}(\forall i, \forall k)$, Z, L, f_0, η

Hacer:

$t = 0$

$\pi(t) = \{\varnothing\}$ siendo $\pi(t) = \{\pi_1, \pi_2, ..., \pi_t\}$ la secuencia parcial de los t trabajos procesados.

$x_i = 0$ $\forall i \in I$, siendo $x_i = 1$ si el trabajo i está contenido en $\pi(t)$ y 0 en caso contrario.

1. Cálculo del índice f

$\forall i \in I$: $x_i = 0$, determinar:

$f_i = LB(\pi(t) \cup \{i\})$
siendo $LB(\pi(t) \cup \{i\})$ el valor de la cota de C_{max} asociado a la secuencia parcial $\pi(t) \cup \{i\}$.

2. Creación de la lista de candidatos *RCL*

Sea $f^* = \text{mín}_{x_i=0}\{f_i\}$;

$RCL(f) = \{i \in I : (x_i = 0) \wedge (f_i \leq \text{mín}\{f_0, f^*/Z\}\}$

donde $Z \in [0, 1]$ es la impedancia sobre el conjunto de elementos compatibles y f_0 es la elasticidad aditiva que que se puede corresponder con el valor de una solución de referencia afectada por la impedancia:
- Si $|RCL(f)| \leq L \Rightarrow RCL = RCL(f)$
- Si $|RCL(f)| > L$:

Sea $i_L \in RCL(f)$ el trabajo que ocupa la *L-ésima* posición en la lista $RCL(f)$ ordenada no-decrecientemente respecto al índice f. Hacer:

$$RCL = \{i \in RCL(f) : f_i \leq f_{i_L}\}$$

3. Selección del trabajo a secuenciar

$\forall i \in RCL$, determinar:

$g_i = \frac{(f_0 - f_i)^\eta}{\sum_{j \in RCL}(f_0 - f_j)^\eta}$, donde η es la elasticidad potencial.

Seleccionar, por sorteo, con probabilidades g_i $\forall i \in RCL$ un trabajo; sea i^* el resultado de esta selección.

4. Actualización

$x_{i^*} \leftarrow x_{i^*} + 1$; $t \leftarrow t + 1$; $\pi_t = i^*$

5. Finalización

Si $t < T = n$, ir a paso-1.

Si no, finalizar.

Figura 3. Fase constructiva *GRASP* para el *BFSP*.

4.3. Fase de mejora local

Se propone una mejora local exhaustiva tipo 2-intercambio entre dos elementos de la secuencia en curso de mejora. La exploración del vecindario es determinista y se realiza de izquierda a derecha.

A partir de una secuencia en curso, $\pi_c(T)$, con valor $C_{max}(\pi_c(T))$, se genera una secuencia vecina, $\pi_v(T)$, mediante un 2-intercambio tentativo entre los elementos que ocupan las posiciones t y t' de la secuencia en curso. Si $C_{max}(\pi_c(T)) > C_{max}(\pi_v(T))$, el intercambio se consolida, $\pi_v(T)$ se convierte en la nueva secuencia en curso y se reinicia el proceso de intercambios. En caso contrario, se prosigue con la generación de una nueva secuencia tentativa, una secuencia vecina, mediante otro 2-intercambio tentativo. El procedimiento finaliza cuando ninguna solución vecina tiene un valor de C_{max} mejor que el de la solución en curso.

4.4. Fase de mejora a través de *NEH*

Además, a la secuencia obtenida en la fase de mejora anterior, cuando se ha alcanzado un óptimo local, se puede aplicar el procedimiento *NEH* consistente en seleccionar al azar un número fijo de elementos de la secuencia y extraerlos de la misma; posteriormente, los elementos extraídos se van insertando de uno en uno en la posición de la secuencia que genera menor C_{max}.

5. Experiencia computacional

Se ha realizado una experiencia computacional con las 6 primeros sets de ejemplares de Taillard [10]. Cada set contiene 10 ejemplares con el mismo número de trabajos (n) y mismo número de máquinas (m). En los 6 sets, el número de trabajos varía entre 20 (set 1) y 50 (set 6) y el número de máquinas entre 5 (set 1) y 20 (set 6). Un valor actualizado de las mejores soluciones para estos ejemplares se puede encontrar en [4].

Para obtener las soluciones de la fase constructiva se emplean 7 algoritmos derivados del procedimiento general *GRASP-x* para el que se han fijado los valores de los parámetros Z, L, f_0 y η (*ver tabla* 1) siendo f_G el valor de referencia, para cada ejemplar, ofrecido por el procedimiento *greedy (G)* con tratamiento de empates. Los algoritmos resultantes tras asignar valores a los 4 parámetros son: (G) una heurística *Greedy* constructiva; (M) un procedimiento *Multistart* tradicional; (*GR-01/*$0{,}5 * |I|$) un algoritmo *GRASP* tradicional con una lista *RCL* limitada a $|I|/2$ candidatos (50 % del número de trabajos); (*GR-5/*$0{,}5 * |I|$) un algoritmo *GRASP* con lista limitada a $|I|/2$ candidatos y con probabilidades de selección de éstos levemente dependientes de su aptitud; (*GR-8/*$0{,}5 * |I|$) un algoritmo *GRASP* con alta dependencia entre las probabilidades de selección de los candidatos y su aptitud y con lista restringida a $|I|/2$ candidatos; y dos

algoritmos *Multistart*, (*GR-5/*$|I|$) y (*GR-8/*$|I|$) con probabilidades de selección de los candidatos idénticas a (*GR-5/*$0{,}5*|I|$) y (*GR-8/*$0{,}5*|I|$), respectivamente.

Tabla 1. Algoritmos derivados de *GRASP-x*. Características.

Alg.	Z	L	f_0	η
G	1	1	∞	1
M	0.01	$\|I\|$	f_G/Z	1
GR-01/$0{,}5*\|I\|$	0.01	$0{,}5*\|I\|$	f_G/Z	1
GR-5/$0{,}5*\|I\|$	0.5	$0{,}5*\|I\|$	f_G/Z	1
GR-8/$0{,}5*\|I\|$	0.8	$0{,}5*\|I\|$	f_G/Z	1
GR-5/$\|I\|$	0.5	$\|I\|$	f_G/Z	1
GR-8/$\|I\|$	0.8	$\|I\|$	f_G/Z	1

Cada uno de los 60 ejemplares se ha resuelto, primero, empleando las 7 variantes de la fase constructiva del algoritmo *GRASP* y, posteriormente, se han aplicado, a cada solución obtenida en la primera fase, dos procedimientos de mejora local. El primer procedimiento de mejora local (*LS* en el texto) es un 2-intercambio exhaustivo, consolidando el intercambio cuando se produce mejora. El segundo procedimiento (*LS+NEH* en el texto) consiste en un bucle que aplica consecutivamente el primer procedimiento de mejora local mencionado, hasta alcanzar un óptimo local, y seguidamente la heurística *NEH* extrayendo e insertando 5 trabajos de la secuencia. El tiempo máximo de *CPU* concedido a la mejora de un ejemplar es de 10 *s*.

Los procedimientos *GRASP* han sido programados en *gcc v. 4.2.1*, en un ordenador Macintosh iMac con un procesador Intel Core i7, 2.93 Ghz. y 8 Gb de memoria RAM, usando MAC OS X 10.6.8 como sistema operativo. Ni la implementación ni el compilador hacen uso de *threads* ni de otra forma de código paralelo, y, por tanto, el ordenador actúa como un único procesador a 2.93 GHz.

En las tablas 2 y 3 se muestran los resultados más significativos del experimento, empleando los procedimientos de mejora *LS* y *LS+NEH*, respectivamente, tras la fase constructiva del *GRASP*. En la tabla 2 se recoge: (1) el número de óptimos alcanzado (*#opt*) por cada uno de los 7 algoritmos sobre los 60 ejemplares del *BFSP*; (2) el promedio de la *desviación porcentual relativa* (RPD=(*solución - óptimo*)/*óptimo*)×100), para los ejemplares y cada algoritmo, tanto para la fase_1 *Greedy* constructiva del *GRASP* ($\overline{RPD_1}$) como para el proceso completo ($\overline{RPD_2}$) que incluye el procedimiento de mejora local (*LS* o *LS+NEH*); (3) el tiempo medio de *CPU*, por ejemplar, requerido por una iteración de cada uno de los 7 algoritmos *GRASP* ($\overline{CPU}$); y (4) el promedio para RPD para cada uno de los sets.

A la vista de las tablas 2 y 3 observamos (en promedio para los 60 ejemplares) los siguientes hechos: (1) las soluciones obtenidas en la fase constructiva del *GRASP*, se alejan de las mejores soluciones conocidas aproximadamente entre el 13 % con el algoritmo *Greedy* determinista (G) y casi el 23 % con los algo-

Tabla 2. *#opt*, $\overline{RPD_1}$, $\overline{RPD_2}$, $\overline{CPU}$ y $\overline{RPD_2}$ para cada uno de los sets en el caso de *LS* para cada uno de los 7 algoritmos

	#opt	$\overline{RPD_1}$	$\overline{RPD_2}$	$\overline{CPU}$	S1	S2	S3	S4	S5	S6
G	0	13.01	4.01	0.28	3.67	3.85	3.84	3.31	4.07	5.31
M	0	22.75	3.22	0.38	2.79	2.37	2.41	3.89	4.04	3.82
GR-01/$0{,}5 * \|I\|$	0	20.64	3.06	0.40	2.42	2.56	1.80	3.55	4.09	3.94
GR-5/$0{,}5 * \|I\|$	0	20.46	2.92	0.40	2.13	2.40	1.81	3.62	3.76	3.81
GR-8/$0{,}5 * \|I\|$	0	20.76	3.01	0.41	2.45	2.11	2.25	3.62	3.87	3.76
GR-5/$\|I\|$	0	22.43	3.26	0.37	2.58	2.43	2.00	4.17	4.26	4.10
GR-8/$\|I\|$	0	22.38	3.16	0.36	2.50	2.62	1.85	3.80	4.14	4.03

Tabla 3. *#opt*, $\overline{RPD_1}$, $\overline{RPD_2}$, $\overline{CPU}$ y $\overline{RPD_2}$ para cada uno de los sets en el caso de *LS+NEH* para cada uno de los 7 algoritmos

	#opt	$\overline{RPD_1}$	$\overline{RPD_2}$	$\overline{CPU}$	S1	S2	S3	S4	S5	S6
G	19	13.01	0.69	10.00	0.07	0.12	0.16	0.95	1.27	1.56
M	25	22.75	0.70	10.00	0.13	0.00	0.03	1.10	1.37	1.57
GR-01/$0{,}5 * \|I\|$	20	20.64	0.64	10.00	0.14	0.13	0.07	0.85	1.25	1.43
GR-5/$0{,}5 * \|I\|$	24	20.46	0.68	10.00	0.10	0.11	0.00	1.04	1.29	1.53
GR-8/$0{,}5 * \|I\|$	28	20.76	0.66	10.00	0.00	0.00	0.02	1.12	1.32	1.53
GR-5/$\|I\|$	25	22.43	0.67	10.00	0.02	0.03	0.02	1.01	1.31	1.64
GR-8/$\|I\|$	25	22.38	0.71	10.00	0.04	0.12	0.02	1.20	1.35	1.52

ritmos *Multistart* (M) y *GRASP* con $L = |I|$ ($GR - 5/|I|$ y $GR - 8/|I|$); (2) *LS* desciende, desde la solución ofrecida por la fase constructiva del algoritmo *Greedy* determinista (G), a óptimos locales con peor valor que los alcanzados por el resto de procedimientos que incorporan azar; (3) atendiendo a todos los procedimientos, *LS* mejora del 20.35 % al 3.23 % la distancia a las mejores soluciones conocidas; (4) *LS+NEH* reduce dichas distancias al 0.68 %; y (5) en todos los procedimientos con mejora *LS*, se alcanza el primer óptimo local en 0.37 s.

Además, en los procedimientos con mejora *LS*, no se alcanza ninguna mejor solución conocida para los 60 ejemplares. En cambio, usando todos los procedimientos con mejora local *LS+NEH* se alcanzan 29 sobre 60. Finalmente, el procedimiento $GR - 8/0{,}5 * |I|$ alcanza 28 óptimos, incluyendo los del set 1 y 2.

6. Conclusiones

Los procedimientos *GRASP* implementados se muestran competitivos respecto a los existentes en la literatura. Concretamente el procedimiento $GR - 8/0{,}5 * |I|$ alcanza todas las mejores soluciones conocidas para los sets 1 y 2 de Taillard. El procedimiento *NEH* reduce las distancias al 0.68 % respecto a

las mejores soluciones conocidas, mientras que su exclusión de los procedimientos producen una reducción al 3.23 %; sin optimización local los procedimientos constructivos dejan esta distancia al 20.35 %.

Agradecimientos. Los autores agradecen la colaboración prestada por *Nissan Spanish Industrial Operations* (NSIO), la Cátedra Nissan UPC y al Gobierno Español por la financiación parcial de este trabajo a través del proyecto PROTHIUS-III: DPI2010-16759, incluyendo fondos FEDER.

Referencias

1. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan,A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics, vol. .5, 287-326 (1979).
2. Nawaz, M., Enscore, Jr E.E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, vol. 11 (1), 91-95 (1983).
3. Ronconi, D. P.: A note on constructive heuristics for the flowshop problem with blocking. International Journal of Production Economics, vol. 87 (1), 39-48 (2004).
4. Bautista, J., Cano, A., Companys, R., Ribas, I.: Solving the $Fm|block|C_{max}$ problem using Bounded Dynamic Programming. Engineering Applications of Artificial Intelligence, vol. 25 (6), 12351245 (2012).
5. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. of Global Optimization, vol. 6, 109-133 (1995).
6. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. Handbook of Metaheuristics, F. Glover and G. Kochenberger, eds., Kluwer Academic Publishers, 219-249 (2003).
7. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP-Part I: Algorithms. International Transactions in Operational Research, vol. 16, 1-24 (2009).
8. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP-Part II: Applications. International Transactions in Operational Research, vol. 16, 131-172 (2009).
9. Laguna, M., Martí, R.: GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization. INFORMS Journal on Computing, vol. 11(1), 44-52 (1999).
10. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research, vol. 64(2), 278-285 (1993) .