3rd International Conference on Industrial Engineering and Industrial Management
XIII Congreso de Ingeniería de Organización
Barcelona-Terrassa, September 2nd-4th 2009

Programación de la producción en talleres de flujo con máquinas sin interrupción. Algoritmos y aplicaciones*

Carlos Fernández-Martínez, Rubén Ruiz, Eva Vallada

Grupo de Sistemas de Optimización Aplicada. Instituto Tecnológico de Informática de Valencia. Universidad Politécnica de Valencia. Camino de Vera s/n, 46022, Valencia. {cfernandez, rruiz, evallada}@iti.upv.es.

Palabras clave: Flowshop, No-idle, Iterated Greedy

1. Introducción

El presente trabajo trata los problemas de programación de la producción en talleres de flujo (*flowshop*) bajo la restricción de que las máquinas no pueden parar (*no-idle)* con el objetivo de minimizar el tiempo máximo de finalización o *makespan* ($C_{máx}$). En este sentido se presentan dos entornos diferentes: *no-idle* puro y *no-idle* mixto. En el primero de ellos todas las máquinas presentan la restricción *no-idle* y, en el segundo, coexisten máquinas de tipo *no-idle* con máquinas regulares. En la siguiente figura se puede observar como afectan estas restricciones al taller. Se puede ver como, entre otras cosas, el primer trabajo asignado a cada máquina normalmente tiene que retrasarse de manera que se pueda procesar junto con los demás sin interrupción. Se trata pues de una generalización del problema del taller de flujo que resulta ser mucho más complicada de tratar.

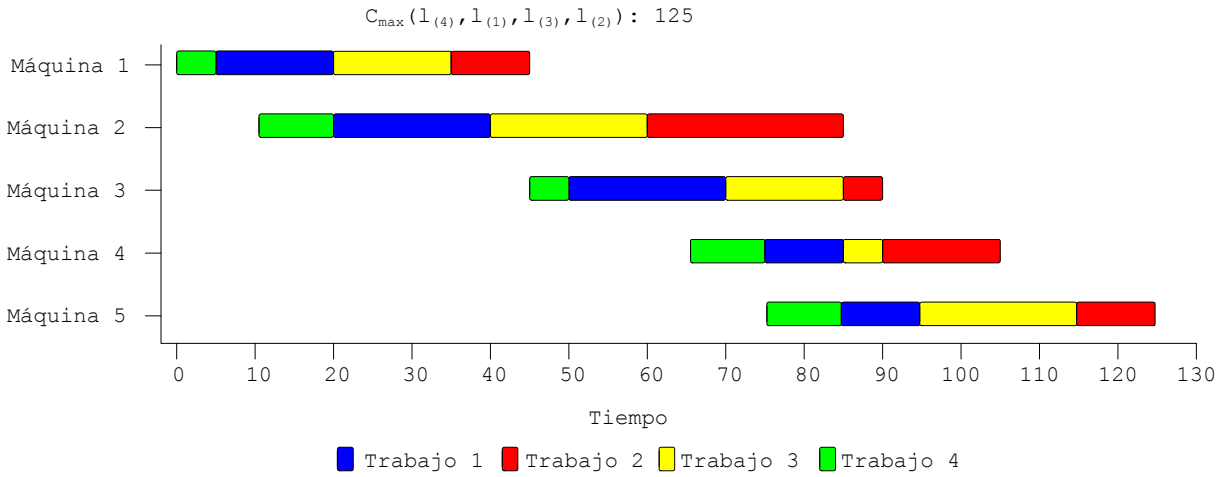

Figura 1. Ejemplo de taller de flujo con la restricción de *no-idle* en todas las máquinas.

En particular, este problema ha sido poco tratado en la literatura cuando hablamos de un entorno puro y, que nosotros sepamos, no se ha estudiado todavía cuando el entorno es mixto. Por esto último, una de las novedades presentadas en este trabajo es el desarrollo de una formulación para el cálculo del *makespan* en un entorno mixto con un coste de solo $O(nm)$.

* Este trabajo está parcialmente subvencionado por el Ministerio de Ciencia e Innovación, bajo los proyectos "OACS - Optimización Avanzada de la Cadena de Suministro" y "SMPA - Secuenciación Multiobjetivo Paralela Avanzada: Avances Teóricos y Prácticos" con referencias IAP-020100-2008-11 y DPI2008-03511/DPI, respectivamente.

Otra aportación del trabajo consiste en una amplia revisión bibliográfica sobre el conjunto de heurísticas y metaheurísticas existentes hasta el momento en la literatura para el problema dado en el entorno puro (Ruiz, et al., 2009). Además proponemos modificaciones de algunas de ellas, así como la adaptación de algunas de las heurísticas y metaheurísticas para el problema en el entorno mixto.

Por último, hemos realizado un análisis exhaustivo para comparar las heurísticas y metaheurísticas en cada uno de los dos escenarios presentados. Para poder llevar a cabo las pruebas hemos generado un conjunto de instancias representativo que está a disposición en la web http://soa.iti.upv.es.

2. Cálculo del *makespan* en un entorno mixto

En este apartado presentamos una de las contribuciones del presente trabajo, el cálculo del *makespan* para un entorno mixto. Una de las razones para llevar a cabo un estudio sobre este problema es que nos parece una situación mucho más realista que el problema que plantea el entorno puro en el que todas las máquinas son del tipo *no-idle*. En base a la experiencia que se dispone a nivel empresarial, sabemos de la existencia de entornos en los que existen algunas máquinas del tipo *no-idle*. Un ejemplo lo podemos encontrar en el sector cerámico en el que existen unos determinados hornos que por razones técnicas o económicas no permiten tiempos ociosos entre trabajos consecutivos.

Siguiendo el ejemplo de la figura 1, si asumimos que las máquinas con número impar son del tipo *no-idle* y las de numeración par del tipo *idle* o regulares tenemos en la figura 2 un diagrama de Gantt mostrando cómo afecta esta nueva situación al *makespan*.

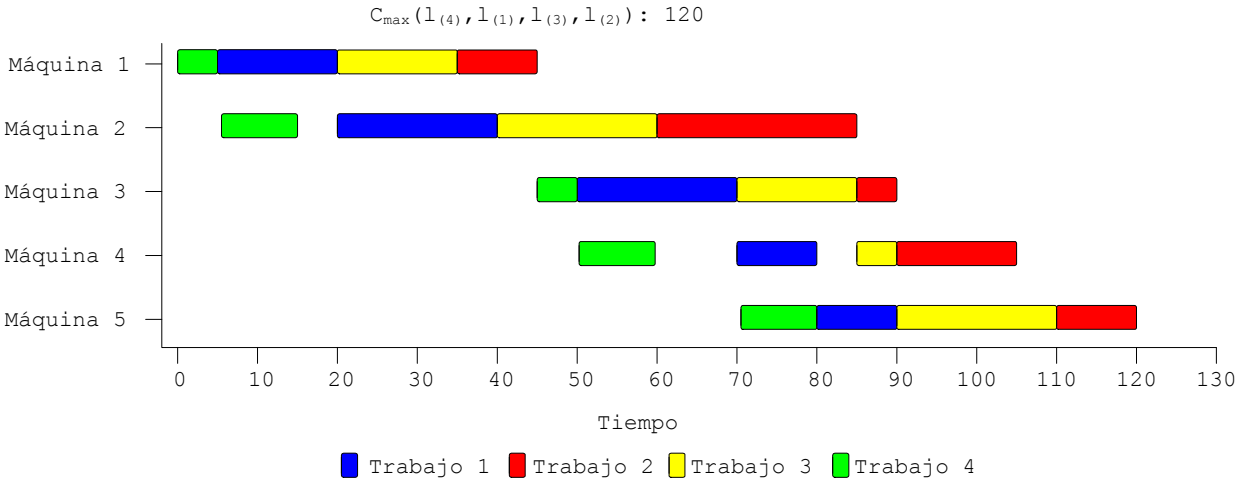

Figura 2. Ejemplo de taller de flujo con la restricción de *no-idle* solo en algunas máquinas (entorno mixto).

Parece evidente, que una de las primeras conclusiones a las que se puede llegar es que $C_{máx_{PURO}} \leq C_{máx_{MIXTO}}$. A continuación presentamos la formulación correspondiente a la fórmula para el cálculo del *makespan* de entorno mixto.

Dado un conjunto de M máquinas y N trabajos:

π representa la secuencia de permutación en la que todos los trabajos siguen el mismo orden de entrada en las máquinas, secuencia para la cual se quiere calcular el valor del *makespan*.

$p_{i,j}, \forall i \in M, \forall j \in N$ representa el tiempo de proceso del trabajo en j en la máquina i.

$O_{i,j}s$ y $O_{i,j}f$ los instantes de tiempo de inicio y finalización respectivamente $\forall i \in M, \forall j \in N$.

$S_i, \forall i \in M$ representa el instante o tiempo de inicio de cada máquina. Se asume $S_1 = 0$.

$I_i, \forall i \in M$ representa un valor booleano. Un valor de 0 (o *false*) en la máquina i representa que es *idle* o regular y un valor de 1 (o *true*) representa que la máquina i debe cumplir la restricción *no-idle*.

Con todo esto ya estamos en disposición de presentar de forma pseudo algorítmica la manera de calcular el *makespan* en un entorno mixto en $O(nm)$:

$$i, r = \{2, \ldots, M\}$$

Si $I_r = false$ entonces

$$S_i = O_{i-1,\pi_{(1)}} f \tag{1}$$

Si $I_r = true$ entonces

Si $I_{r-1} = true$ entonces

$$S_i = S_{i-1} + \max_{1 \le h \le n} \left\{ \sum_{j=1}^{h} p_{i-1,\pi_{(j)}} - \sum_{j=1}^{h-1} p_{i,\pi_{(j)}} \right\} \tag{2}$$

Si $I_{r-1} = false$ entonces

$$S_i = S_{i-1} + \max_{1 \le h \le n} \left\{ \sum_{j=1}^{h} \left(p_{i-1,\pi_{(j)}} + O_{i-1,\pi_{(j)}} s \right) - \left(\sum_{j=1}^{h-1} p_{i,\pi_{(j)}} + O_{i-1,\pi_{(j)}} f \right) \right\} \tag{3}$$

Aunque pueda parecer una notación que indique un cálculo costoso del *makespan*, se calcula en $O(nm)$, aunque con unas constantes importantes. Dicho cálculo también dependerá del tipo de la máquina:

Si $I_m = true$ entonces $C_{max} = S_m + \sum_{j=1}^{n} p_{m,j}$ (4)

Si $I_m = false$ entonces $C_{max} = S_m + \sum_{j=1}^{n} p_{m,j} + \sum_{j=1}^{n-1} max\left(0, O_{i-1,\pi_{(j+1)}} f - O_{i,\pi_{(j)}} f\right)$ (5)

En definitiva, el cálculo del $C_{máx}$ en el caso en que la última máquina es *no-idle* se reduce a sumarle al instante de comienzo de dicha máquina los tiempos de proceso de todos los trabajos en esa misma máquina. En el caso en que la última máquina es *idle*, además, le suma los posibles espacios en los que no se procesan trabajos desde el primer trabajo hasta el último.

3. Evaluación

Este esta sección se comenta el conjunto de heurísticas y metaheurísticas, el banco de pruebas y los experimentos computacionales llevados a cabo.

3.1. Heurísticas y metaheurísticas

Uno de los aspectos en los que se ha puesto mucho énfasis, ha sido el de reimplementar el actual estado del arte en lo que respecta al taller de flujo con la restricción *no-idle*. Sin esta revisión y reimplementación no podríamos asegurar mejores resultados tras nuestros experimentos computacionales. A modo de resumen, se han estudiado 7 heurísticas y 4 metaheurísticas:

La primera heurística es la NEH propuesta por Nawaz et al. (1983) con las aceleraciones propuestas por Pan y Wang (2008a,b) que tiene una complejidad de $O(n^2m)$. La segunda será la versión sin aceleraciones NEH_{na} que tiene una complejidad de $O(n^3m)$.

La tercera heurística estudiada es la propuesta por Saadani et al. (2005) que tiene una complejidad de $O(n^3)$. La denotaremos como SGM.

La cuarta heurística será la denominada KK y que fue propuesta por Kalczynski y Kamburowski (2005). Tiene una complejidad computacional de $O(n^3m)$.

En este punto se presentan 2 heurísticas y una metaheurística que están basados en la NEH y que han sido propuestos recientemente por Rad et al. (2009).

FRB3 es una extensión del método NEH. Tras insertar un trabajo en una posición, la que minimice el makespan, todos los trabajos ya insertados en anteriores iteraciones reinsertan en todas las posibles posiciones de la secuencia parcial.

FRB4 es una simplificación de FRB3. Después de insertar un trabajo j, sólo los trabajos desde la posición $\pm k$ contando desde j son insertados y reinsertados. Si asumimos que $k \ll n$ tenemos que la complejidad es de $O(n^2m)$.

FRB5 es una metaheurística y una extensión de FRB3. Básicamente, después de insertar un trabajo j, se hace una búsqueda local completa basada en la inserción por vecindad hasta un óptimo local.

La sexta (GH_BM) y séptima (GH_BM2) heurísticas propuestas están basadas en el artículo de Baraz y Mosheiov (2008). La primera de ellas se ha programado basándose fidedignamente en lo propuesto en el artículo. La segunda de ellas se basa en la primera, pero presenta modificaciones críticas y que, como veremos, modifican enormemente su eficiencia y eficacia.

Las dos metaheurísticas de Pan y Wang (2008a,b) que se basan en un *Discrete Differential Evolution* (DDE) y un algoritmo *Discrete Particle Swarm* (HDPSO).

Hemos querido dejar para el final el *Iterated Greedy* (IGLS) presentado por Ruiz y Stützle (2007) por ser actualmente uno de los métodos que presenta los mejores resultados en comparación con los métodos existentes.

Para mayor detalle sobre alguno de los métodos se remite a cada uno de los artículos o al análisis en conjunto llevado a cabo en Ruiz, Vallada, Fernández-Martínez (2009).

3.2. Banco de pruebas

No existe en la literatura un conjunto de instancias completo y común para este problema. En el caso de entorno puro las instancias de otros problemas pueden ser de utilidad, pero, para el entorno mixto se hace necesario definir instancias que recojan distintas situaciones.

Por todo esto proponemos un conjunto completo de instancias. Se han generado 7 subconjuntos de instancias. Cada uno de ellos está compuesto por todas las posibles combinaciones de $n = \{50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}, |n| = 10$ y de $m = \{10, 20, 30, 40, 50\}, |m| = 5$. Además se han generado 5 réplicas por cada combinación. Esto hace un total de 250 instancias para cada uno de los subgrupos. Los tiempos de proceso $p_{i,j}$ se generan aleatoriamente de acuerdo a una distribución uniforme en el rango $[1, 99]$ como es habitual en la literatura. En la siguiente tabla mostramos como quedan los grupos:

Grupo	Distribución máquinas *no-idle*
1	La primera mitad de las máquinas son siempre *no-idle*.
2	La segunda mitad de las máquinas son siempre *no-idle*.
3	Este conjunto tiene las máquinas con la restricción *no-idle* alternadas (*stripe*).
4	De manera aleatoria, tiene un 25% de máquinas de tipo *no-idle*.
5	De manera aleatoria, tiene un 50% de máquinas de tipo *no-idle*.
6	De manera aleatoria, tiene un 75% de máquinas de tipo *no-idle*.
7	El 100% de las máquinas es de tipo *no-idle*.

Tabla 1. Distribución de los grupos de instancias.

3.3. Pruebas

Una vez definido el conjunto de instancias, de metaheurísticas y heurísticas, establecemos las dos partes en que hemos dividido las pruebas. La totalidad de los algoritmos propuestos han sido codificados en Delphi 2007 y lanzados en 12 ordenadores Intel Core 2 Duo E6600 a 2,4Ghz y 2GB de memoria RAM. Sólo se aprovecha un Core de cada uno de ellos.

La variable respuesta que usaremos será el Índice de Desviación Relativa (IDR) sobre el mejor resultado para cada instancia:

$$\text{Índice de Desviación Relativa (IDR)} = \frac{Heu_{sol} - Best_{sol}}{Best_{sol}} \times 100 \quad (6)$$

Donde Heu_{sol} es la solución dada por cualquiera de las heurísticas probadas y $Best_{sol}$ es la mejor solución conocida para esa instancia.

Es importante remarcar que las 4 metaheurísticas (HDPSO, DDE, FRB5 e IGLS) son estocásticas y por lo tanto llevaremos a cabo 5 ejecuciones diferentes de cada instancia. Además, estos métodos, salvo FRB5, tienen un criterio de parada. Estudios previos establecen un criterio de parada basado en el tiempo transcurrido de CPU. Este tiempo transcurrido es medido con extremada precisión y cuidado dentro de cada método para que se produzca la

parada en cuanto se alcance dicho tiempo. Además, la forma de establecerlo se hace a partir de una función: $n \cdot (m/2) \cdot t$ milisegundos. Como se puede ver, tiene en cuenta el número de máquinas y el de trabajos en orden creciente. Es decir, a mayor cantidad de máquinas y/o trabajos, mayor tiempo de proceso para el método. Esta fórmula además minimiza el efecto del tamaño de la instancia en los resultados y en el análisis estadístico. Por último, para poder comprobar el efecto del tiempo de CPU, los tres métodos con criterio de parada, (HDPSO, DDE e IGLS) se prueban con 3 valores distintos de $t, t = \{10,20,30\}$. Como se puede ver, es sencillo calcular el tiempo que se les da a estos métodos cuando se dispone de una instancia, puesto que se conocen los parámetros n,m y t. La forma que tendremos para denotar el parámetro t en los métodos será, por ejemplo, HDPSO^{10} que se refiere al método HDPSO con $t = 10$.

Las pruebas se dividen en dos partes bien diferenciadas:

Se prueban las instancias del grupo 7 en las 7 heurísticas y 4 metaheurísticas que se han detallado anteriormente. En cada una de las heurísticas se calcula una sola vez cada instancia y en cada metaheurística se calcula 5 veces. Además 3 metaheurísticas se calculan para 3 valores distintos de t. Con todo esto, el número de instancias resueltas es de $(7 * 250) + (10 * 250) * 5 = 15.000$.

Se lanzan las instancias de todos los grupos menos el 7 para 2 heurísticas y 2 metaheurísticas. En cada una de las heurísticas se calcula una sola vez cada instancia y en cada metaheurística se calcula 5 veces. Además las 2 metaheurísticas se calculan para $t = 10$. Con todo esto, el número de instancias resueltas es de $6 * (2 * 250) + 6 * (2 * 250) * 5 = 18.000$.

El siguiente esquema resume las pruebas llevadas a cabo:

$$\left.\begin{array}{l} \text{Heurísticas} \left\{ \begin{array}{l} NEH \\ NEH_{na} \\ SGM \\ KK \\ GH\_BM \\ GH\_BM2 \\ FRB3 \\ FRB4 \end{array} \right. \\ \text{Metaheurísticas} \left\{ \begin{array}{l} HDSPO \\ DDELS \\ FRB5 \\ IG_{LS} \end{array} \right. \end{array} \right\} Grupo_7 \qquad \left.\begin{array}{l} \text{Heurísticas} \left\{ \begin{array}{l} KK \\ GH\_BM2 \end{array} \right. \\ \text{Metaheurísticas} \left\{ \begin{array}{l} HDSPO^{30} \\ IG_{LS}^{30} \end{array} \right. \end{array} \right\} Grupo_i, i = \{1,\ldots,6\}$$

3.4. Resultados

En primer lugar comentamos los resultados del entorno puro, en el que todas las máquinas son del tipo *no-idle*. En la siguiente tabla mostramos los datos relativos a las pruebas sobre las heurísticas:

	NEH	NEH_{na}	SGM	KK	GH_BM	GH_BM2	FRB3	$FRB4_4$	$FRB4_{12}$
IDR medio	6,12	6,12	22,61	2,35	4,59	2,82	1,75	3,24	2,61
Tiempo medio (s)	0,077	5,834	0,114	41,447	21,551	0,229	19,190	0,680	1,596

Tabla 2. IDR medio y tiempo de CPU medio para las heurísticas en un entorno *no-idle* puro.

Como podemos apreciar, en cuanto al IDR medio destaca FRB3 entre todos los métodos, siendo KK y $FRB4_{12}$ heurísticos bastante competitivos. En cuanto al tiempo de CPU, NEH y SGM presentan valores muy buenos. También se puede apreciar que es complicado obtener los mejores valores de IDR y tiempo para un mismo método. Por ello, si tenemos que dar una solución compromiso sería GH_BM2. Es importante destacar la gran mejora que hemos obtenido comparando GH_BM y GH_BM2, el primero de ellos recordamos que es la versión propuesta por los Baraz y Mosheiov (2008) mientras que el segundo es una modificación del original.

La siguiente tabla muestra los resultados para las metaheurísticas.

	$HDPSO^{10}$	$HDPSO^{20}$	$HDPSO^{30}$	DDE_{LS}^{10}	DDE_{LS}^{20}	DDE_{LS}^{30}
IDR medio	0,78	0,57	0,48	2,65	2,66	2,65
Tiempo medio (s)	41,29	82,54	123,79	41,25	82,50	123,75

	IG_{LS}^{10}	IG_{LS}^{20}	IG_{LS}^{30}	FRB5
IDR medio	0,65	0,43	0,34	1,36
Tiempo medio (s)	41,25	82,50	123,75	54,64

Tabla 3. IDR medio y tiempo medio para las metaheurísticas en un entorno *no-idle* puro.

En lo que respecta a las metaheurísticas, podemos observar como IG_{LS} y HDPSO son claramente mejores que los otros métodos y, además, llama la atención que HDPSO no mejora los resultados de IG_{LS} cuando IG_{LS} se usa como una subrutina de búsqueda local en HDPSO.

Se analiza ahora si las diferencias en las medias anteriormente descritas son estadísticamente significativas. Para poder hacerlo llevaremos a cabo un Diseño de Experimentos (DOE), Montgomery (2005). Por motivos de espacio no podemos extendernos, así que mostramos tan solo las conclusiones del estudio. En concreto y a modo de ejemplo, podemos plantearnos la duda de si son estadísticamente significativas las diferencias presentadas por IG_{LS} y HDPSO para los valores de *t* estudiados. En la figura 3 podemos ver el resultado del análisis:

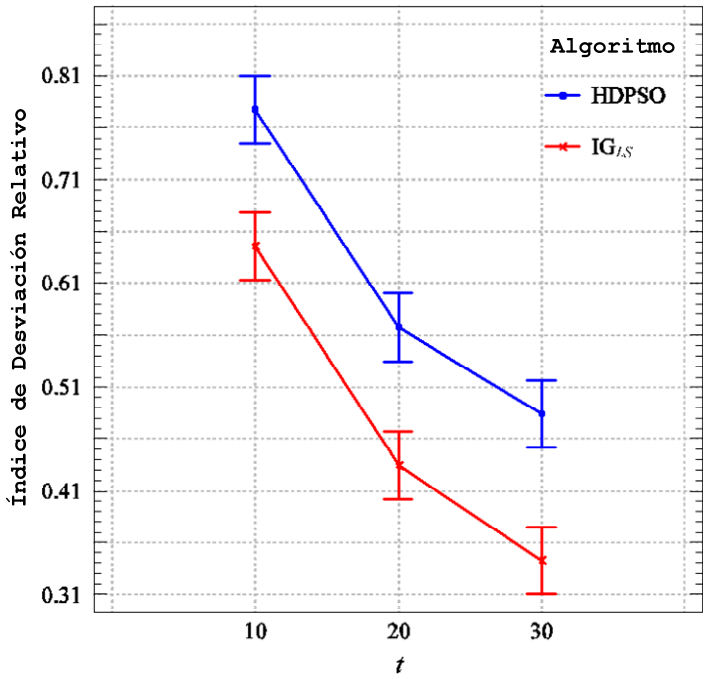

Figura 3. Gráfico de interacción del factor t para IGLS y HDPSO con intervalos de confianza de Tukey al 95%

Podemos por tanto concluir que sí existen diferencias estadísticamente significativas entre IG_{LS} y HDPSO para los valores de t estudiados.

La manera de proceder para el análisis de los datos del entorno mixto es similar al llevado a cabo en el puro.

	Grupo 1		Grupo 2		Grupo 3	
	$HDPSO^{30}$	**IG_{LS}^{30}**	**$HDPSO^{30}$**	**IG_{LS}^{30}**	**$HDPSO^{30}$**	**IG_{LS}^{30}**
IDR medio	0,45	0,43	0,46	0,42	0,52	0,53
Tiempo medio (seg)	123,75	123,75	123,75	123,75	123,75	123,75

	Grupo 4		Grupo 5		Grupo 6	
	$HDPSO^{30}$	**IG_{LS}^{30}**	**$HDPSO^{30}$**	**IG_{LS}^{30}**	**$HDPSO^{30}$**	**IG_{LS}^{30}**
IDR medio	0,47	0,48	0,53	0,54	0,53	0,52
Tiempo medio (seg)	123,75	123,75	123,75	123,75	123,75	123,75

	Grupo 1		Grupo 2		Grupo 3	
	GH_BM2	**KK**	**GH_BM2**	**KK**	**GH_BM2**	**KK**
IDR medio	1,83	6,71	1,80	6,61	2,00	5,81
Tiempo medio (seg)	23,85	41,47	24,22	41,44	26,33	41,46

	Grupo 4		Grupo 5		Grupo 6	
	GH_BM2	**KK**	**GH_BM2**	**KK**	**GH_BM2**	**KK**
IDR medio	1,99	10,16	2,01	6,63	1,94	3,70
Tiempo medio (seg)	23,75	41,44	24,88	41,46	25,34	41,45

Tabla 4. IDR medio y tiempo medio para los métodos en un entorno mixto.

A partir de esta tabla podemos observar como se hace harto complicado extraer alguna conclusión debido a la gran cantidad de datos. No obstante, observamos como el heurístico GH_BM2 es mejor que KK. Por el contrario, en las metaheurísticas se hace imposible dar una clara opinión. Como en el caso anterior, recurrimos al análisis estadístico para resolver la duda sobre si existen diferencias estadísticamente significativas entre $HDPSO^{30}$ y IG_{LS}^{30}. La figura 4 nos indica que la respuesta es no, al contrario que ocurría en el entorno puro.

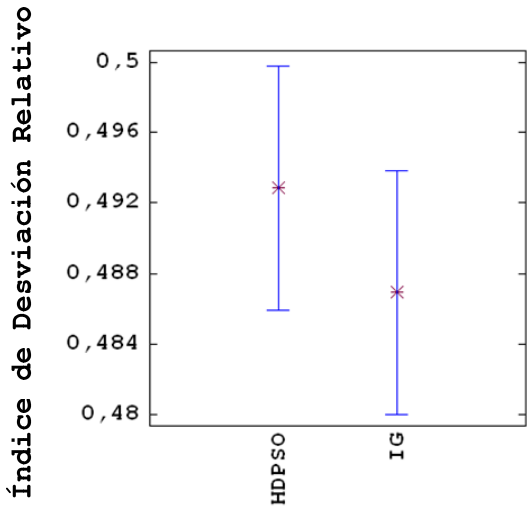

Figura 4. Intervalos de confianza de Tukey al 95% del promedio del IDR para las metaheurísticas en el entorno mixto.

4. Conclusiones

En el presente trabajo se ha presentado un amplio estudio sobre el problema del taller de flujo bajo la restricción *no-idle* y con el objetivo de minimizar el *makespan*. Para ello se ha llevado a cabo una completa revisión bibliográfica y una codificación de todos los métodos existentes.

Como novedades se ha propuesto un entorno más realista al que hemos llamado entorno mixto en el que conviven máquinas regulares y *no-idle*. Además, se contribuye con una formulación para el cálculo del *makespan* en $O(nm)$. También se han readaptado algunos métodos para el entorno mixto y se han propuesto mejoras para ambos entornos. Por último, se aporta un gran conjunto de instancias para que sirvan de referencia en futuros estudios así como un análisis estadístico de un conjunto amplio de pruebas. Los resultados indican que los métodos readaptados por nosotros son los que mejores resultados proporcionan.

Referencias

Baraz, D. and Mosheiov, G. (2008). A note on a greedy heuristic for flow-shop makespan minimization with no machine idle-time. European Journal of Operational Research, 184(2):810–813.

Kalczynski, P. J. and Kamburowski, J. (2005). A heuristic for minimizing the *makespan* in *no-idle* permutation flow shops. *Computers & Industrial Engineering*, 49(1):146–154.

Montgomery, D. (2005). *Design and Analysis of Experiments*. John Wiley & Sons, New York, sixth edition.

Nawaz, M., Enscore, Jr E. E., y Ham, I. (1983). A Heuristic Algorithm for the *m*-Machine, *n*-Job Flow-shop Sequencing Problem. *OMEGA, The International Journal of Management Science*, 11(1):91-95.

Rad, S. F., Ruiz, R., and Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. OMEGA, the International Journal of Management Science, 37:331–345.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165:479–494.

Ruiz, R., Maroto, C., and Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. OMEGA, *the International Journal of Management Science*, 34:461–476.

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

Ruiz, R. and Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with *makespan* and weighted tardiness objectives. *European Journal of Operational Research*, 187(3):1143–1159.

Ruiz, R., Vallada, E. y Fernández-Martínez, C. Scheduling in flowshops with no-idle machines. Informe técnico DEIOAC-2008-1. Departamento de Estadística e Investigación Operativa Aplicadas y Calidad. Universidad Politécnica de Valencia.

Saadani, N. E. H., Guinet, A., and Moalla, M. (2005). A travelling salesman approach to solve the F/no−idle/Cmax problem. *European Journal of Operational Research,* 161(1):11–20.

Pan, Q.-K. and Wang, L. (2008a). *No-idle* permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. *In press at International Journal of Advanced Manufacturing Technology*.

Pan, Q.-K. and Wang, L. (2008b). A novel differential evolution algorithm for *no-idle* permutation flow-shop scheduling problems. *European Journal of Industrial Engineering*, 2(3):279–297.