4th International Conference on Industrial Engineering and Industrial Management
XIV Congreso de Ingeniería de Organización
Donostia- San Sebastián , September 8th -10th 2010

Programación de la producción en máquinas en paralelo con lotes de transferencia: un caso de estudio.

Gema Calleja[1], Rafael Pastor[1]

[1] Institute of Industrial and Control Engineering (IOC), Universitat Politècnica de Catalunya (UPC).
gema.calleja@upc.edu, rafael.pastor@upc.edu

Resumen

En este trabajo se presenta un algoritmo de dispatching para resolver un problema real de asignación y secuenciación de operaciones de una empresa del sector automoción, en un ambiente de Flexible Job-Shop Scheduling Problem (FJSSP) con lotes de transferencia, con el objetivo de minimizar el retraso medio de los pedidos. A partir de los datos del plan de entregas, el algoritmo propuesto genera como resultado en cada máquina, la secuencia de operaciones necesaria para cumplir el plan y su temporización correspondiente. La experiencia computacional muestra que el algoritmo mejora sensiblemente el método utilizado en la empresa, permitiendo una mayor competitividad en los plazos de entrega.

Palabras clave: secuenciación, máquinas en paralelo, algoritmo de *dispatching*, lotes de transferencia.

1. Introducción

El problema presentado en este trabajo corresponde al caso de una industria del sector automoción. La empresa se halla en una situación crítica, debida principalmente a los continuos retrasos en las entregas, y desea obtener un método para programar adecuadamente los pedidos en las máquinas, de manera que se minimice el retraso medio de las entregas.

La producción se realiza en una planta de estampación y soldadura, integrada por 23 personas y 16 máquinas, de las cuales 10 son prensas y el resto son equipos de soldadura. Las piezas producidas son componentes metálicos para el automóvil. Los pedidos son semanales y la demanda es variable; finalmente su único cliente habitualmente solicita pedidos de unos 18 tipos de piezas. La empresa factura unos 5 millones de euros anuales.

A continuación se describe la configuración del sistema productivo. El lunes de cada semana la empresa recibe el pedido de su cliente que es un plan de entregas de *z* tipos de pieza, con demandas y fechas de entregas diferentes (aunque siempre, dentro de la semana siguiente a la que se recibe el pedido). La fabricación de cada pieza se compone de varias operaciones que han de realizarse siguiendo un orden de precedencias establecido. Cada máquina puede procesar una única operación a la vez y una operación puede ser procesada en una entre varias de las máquinas. Para favorecer el avance de la producción, existen lotes de transferencia de piezas entre máquinas de dos operaciones consecutivas. Esta característica implica que la segunda máquina no espera a que la primera complete todas las unidades de una pieza, sino que una vez se ha fabricado una cierta cantidad de unidades en la primera máquina (el lote de transferencia), éstas se transfieren a la máquina siguiente para que trabaje en paralelo.

El objetivo es minimizar el retraso medio de las piezas, siendo el retraso la diferencia positiva entre la finalización de fabricación de una pieza y su fecha de entrega.

2. Estado del arte

El problema del taller mecánico clásico (*Job-Shop Scheduling Problem*, *JSSP*) es uno de los problemas de optimización combinatoria más populares y ha sido objeto de numerosos estudios para su resolución. En el JSSP se debe obtener la secuencia de operaciones en cada máquina y su temporización correspondiente, con el fin de optimizar un índice de eficiencia determinado. Como extensión de este modelo se ha desarrollado el problema del taller mecánico flexible (*Flexible Job-Shop Problem FJSSP*), que incorpora la posibilidad de considerar un conjunto de máquinas para realizar una operación determinada. Es más complejo que el JSSP puesto que se añade la necesidad de realizar la asignación de las operaciones a las máquinas.

Ambos problemas son de tipo *NP-Hard*, de los más difíciles de resolver. Esta complejidad ha motivado la investigación y desarrollo de métodos heurísticos y metaheurísticos para intentar resolverlos (por ejemplo, recocido simulado, búsqueda tabú, algoritmos genéticos, optimización por colonias de hormigas, redes neuronales y algoritmos evolutivos).

Existen diversas versiones sobre el origen del JSSP. Roy y Sussman (1964) fueron los primeros en proponer la representación mediante el grafo disyuntivo, y Balas (1969) aplicó por primera vez un procedimiento enumerativo basado en este grafo. Sin embargo existen trabajos anteriores: Giffer y Thompson (1960) propusieron un algoritmo de *dispatching* de reglas de prioridad, Jackson (1956) generalizó el algoritmo del *flow-shop* de Johnson de 1954 al del JSSP y Akers y Friedman (1955) aplicaron un modelo de álgebra booleana para representar secuencias de procesamiento. Aunque no está claro a quién se le debe atribuir el primer planteamiento del JSSP, se considera que el libro "*Industrial Scheduling*" de Muth y Thompson (1963) es la base de la mayoría de investigaciones posteriores.

Más recientemente, se han desarrollado métodos exactos para la resolución del JSSP, como los publicados por Applegate y Cook (1991) y Williamson et al (1997), quienes resolvieron con éxito pequeños ejemplares. En la actualidad, problemas de dimensión 15x15 siguen siendo intratables mediante métodos exactos. Para resolver tales problemas se han desarrollado numerosos métodos heurísticos y metaheurísticos, entre los que cabe mencionar las contribuciones en algoritmos genéticos de Oliveira (2000), el desarrollo de un método GRASP con recirculación de Aiex et al (2003) y el método de optimización híbrida de Wang y Zheng (2001).

En el entorno del FJSSP Brucker y Schile (1990) fueron los primeros en formular el problema. Gomes et al (2005) presentaron un modelo de programación lineal entera que resuelve de forma óptima casos realistas de FJSSP, con grupos de máquinas homogéneos, tiempos de preparación negligibles y recirculación de operaciones. Sin embargo, igual que en el caso del JSSP, los métodos exactos no suelen ser efectivos para ejemplares de grandes dimensiones del problema, por lo que se ha dedicado un gran esfuerzo en desarrollar heurísticas y metaheurísticas para resolverlo. Kim et al (2002) publicaron un método basado en algoritmo de recocido simulado, y Gao et al (2006) propusieron un método híbrido de algoritmo genético y búsqueda local para el FJSSP considerando la secuenciación de las operaciones de mantenimiento preventivo. Más tarde, Ravetti et al (2007) desarrollaron un modelo matemático basado en la metaheurística GRASP para máquinas en paralelo y tiempos de preparación dependientes de la secuencia. Un trabajo en esta misma área es el publicado por Allaverdi et al (2008), en el que se indican las diferentes clases de problemas FJSSP y las técnicas empleadas en su resolución. Finalmente, otra contribución reciente es la propuesta de Xing et al (2009), quienes presentan un modelo de simulación para resolver el FJSSP con múltiples objetivos.

3. La programación de la producción en la empresa

La programación de la producción en la empresa se realiza de forma manual e intuitiva, siguiendo los pasos básicos de un hipotético y sencillo algoritmo de *dispatching*. De todas formas, no se llega a generar un *schedule* (programa) con la programación prevista de las actividades productivas, ya que la selección de operaciones y su asignación a las máquinas se improvisa en tiempo real, según las necesidades y la urgencia del momento. Las ineficiencias que presenta la programación de la producción provocan continuos retrasos y la insatisfacción de su cliente.

4. Definición e hipótesis del problema a resolver

El caso real expuesto corresponde a un problema de taller mecánico flexible $n/m/G/T_{med}$ con máquinas no homogéneas en paralelo, es decir, no todas las operaciones pueden ser realizadas en cualquier máquina, sino sólo en un conjunto predeterminado.

Se consideran n piezas, formadas por varias operaciones que deben ser fabricadas según un orden establecido, y m máquinas en las que pueden ser procesadas. Las rutas de las piezas siguen un flujo de tipo general a través de las máquinas (*General Job-Shop*, *G*). Las máquinas no pueden procesar más de una operación a la vez y no se permiten interrupciones en la máquina una vez iniciada la operación. La flexibilidad de las máquinas es parcial: no todas las operaciones son realizables en todas las máquinas disponibles. Se conocen los tiempos de proceso $p_{k,i,j}$ para la operación k de la pieza i en la máquina j, así como las fechas de entrega d_i de cada pieza (pedido) i. Los tiempos de preparación son constantes e independientes de la secuencia, por lo que se añaden al tiempo de proceso de las operaciones.

El objetivo es encontrar una asignación y secuencia de las operaciones en las máquinas que minimice el retraso medio (T_{med}) de las mismas, según la siguiente función (1):

$$T_{med} = \frac{\sum_{i=1}^{n} \max(0, C_i - d_i)}{n}, \quad (1)$$

donde $\boldsymbol{C_i}$ es el tiempo de finalización de la pieza $\boldsymbol{i}$ y $\boldsymbol{d_i}$ su fecha de entrega.

Con estos datos debe establecerse un programa, es decir: i) la asignación de operaciones a las máquinas, ii) la secuencia de operaciones en cada máquina, y iii) los tiempos de inicio y finalización de cada operación.

La Figura 1 muestra un esquema simplificado de la configuración del problema. El *input* lo constituye el plan de entregas, que agrega los datos del pedido del cliente y los retrasos pendientes, y los datos relativos a operaciones y máquinas. Una vez introducida esta información, el algoritmo se puede lanzar en modo determinista (genera una única solución) o en modo aleatorizado (genera tantas soluciones como repeticiones indique el usuario). La solución más satisfactoria proporciona el programa de asignación de operaciones a máquinas y su secuenciación temporizada. El programa de fabricación se ejecuta durante la semana actual, y las piezas fabricadas se entregan en la semana siguiente, en las fechas solicitadas.

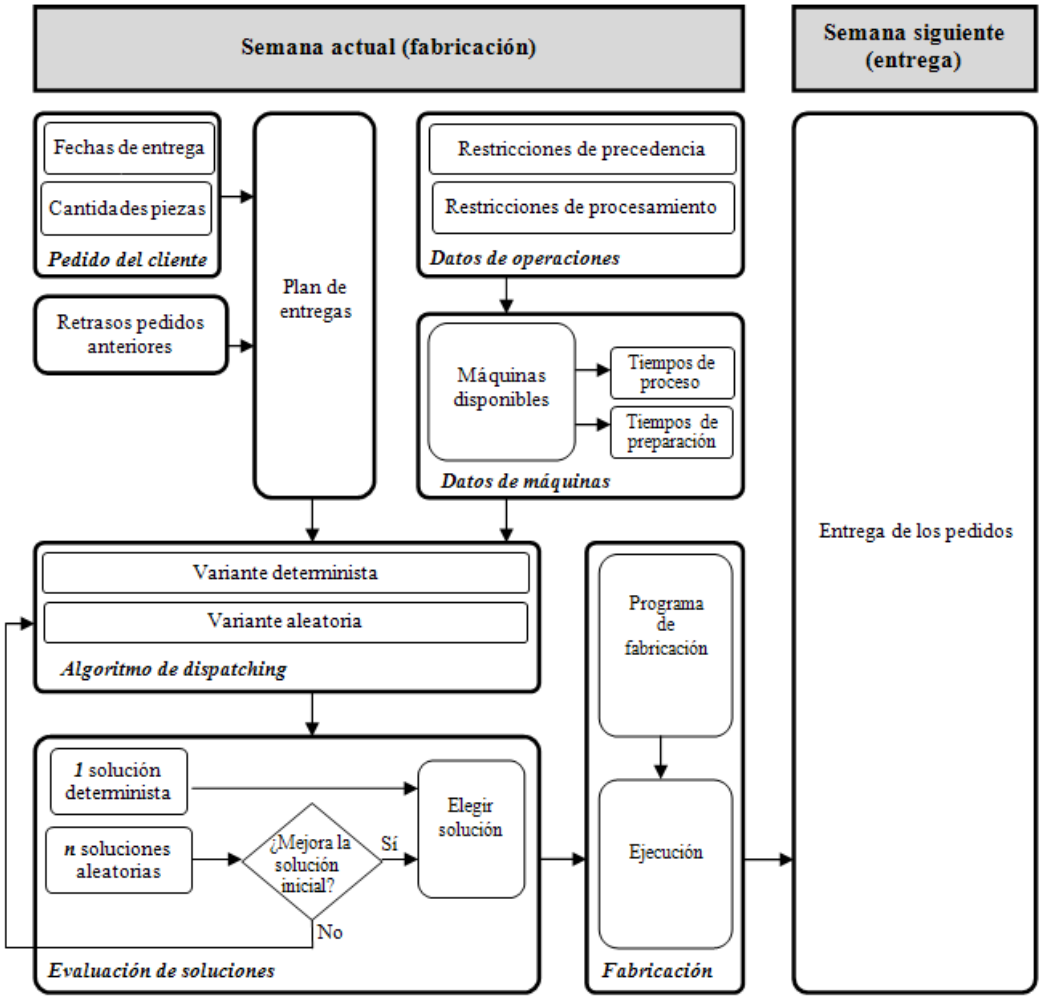

Figura 1. Configuración del problema.

5. Esquema del algoritmo de *dispatching*

Para resolver el problema planteado se ha diseñado un algoritmo de *dispatching* que se describe a continuación. En el apartado 5.1 se indica la nomenclatura utilizada y en el apartado 5.2 se presenta un esquema básico de su funcionamiento.

5.1. Nomenclatura

- D_i: demanda de la pieza *i*.
- $r_{k,i}$: instante de disponibilidad de la operación *k* de la pieza *i* (k, i).
- fp_j: instante más temprano para empezar una nueva operación en la máquina *j*.
- $rp_{k,i,j}$: (*release date* o *ready date*) instante más temprano para empezar la operación (*k, i*) en la máquina *j*.
- $t_{inicio\ (k,\ i)}$: instante de inicio de fabricación la operación (*k, i*) programada.
- $t_{final\ (k,\ i)}$: instante final de fabricación la operación (*k, i*) programada.

5.2. Procedimiento heurístico

La heurística propuesta consiste en un algoritmo de *dispatching*; se trata de un método constructivo (se parte e una solución inicial vacía y se van añadiendo elementos siguiendo

ciertos criterios para obtener la solución final) y directo (una vez se programa una operación no se reconsidera ni modifica en los pasos siguientes).

En cada iteración existen tres conjuntos de operaciones: elegibles (E), programadas (P) y el resto (N). En un problema *n*/*m* inicialmente se sitúan en E las n primeras operaciones de las piezas sin operación precedente, y en N las operaciones restantes; P está vacío. Al programar la operación de una pieza, la operación pasa de E a P automáticamente y la operación siguiente de dicha pieza pasa de N a E (salvo si se trata de la última operación de la pieza). En un instante cualquiera, los subconjuntos N, E y P se hallan en un estado determinado. Cuando el algoritmo termina, todas las operaciones están en P.

El algoritmo debe proporcionar la operación elegida, la máquina asignada y los instantes de tiempo de inicio y fin de fabricación. En cada iteración, la programación se realiza en dos fases. En primer lugar se elige la máquina o máquinas con menor instante de disponibilidad para iniciar una nueva operación, y a continuación se escoge la operación a programar entre las operaciones candidatas. Así, se necesitan dos criterios de selección: de máquina y de operación.

5.2.1 Reglas de elección de máquina

La elección de máquina se realiza según las reglas indicadas en la Figura 2.

Regla 1: Seleccionar la máquina j que esté disponible más pronto. Es decir, aquella con menor fp_j: $fp_{min} = \min \{ fp_j \}$.

Regla 2: Utilizar las reglas de elección de operaciones y elegir la máquina cuya operación sea la más prioritaria.

Regla 3: Elegir la máquina más rápida para realizar la operación de la *Regla 2.*

Regla 4: Elegir cualquiera al azar.

Figura 2. Reglas de elección de máquina

5.2.2 Reglas de elección de operación

Se selecciona la operación correspondiente a la pieza más prioritaria según las reglas mostradas en la Figura 3.

Regla 1: Mayor unidades en retraso acumulado desde el pedido anterior.

Regla 2: Más días de retraso acumulados desde el pedido anterior.

Regla 3: Fecha de entrega más urgente.

Regla 4: Mayor duración de proceso pendiente.

Regla 5: Mayor demanda mensual media.

Regla 6: Elegir cualquiera al azar.

Figura 3. Reglas de elección de operación

5.2.3 Reglas de elección de operación

En el diseño del procedimiento heurístico se han integrado reglas de robustez con el fin de generar un programa más robusto (eficiente) frente a imprevistos que puedan ocurrir durante su ejecución.

Un ejemplo lo constituye la regla 1 de elección de máquina (apartado 5.2.1), que evita posponer innecesariamente el procesamiento de las operaciones. Las operaciones empiezan lo antes posible, reduciendo así el impacto de posibles futuros retrasos provocados por algún contratiempo que altere el ritmo de producción.

Otras reglas de robustez consideradas consisten en priorizar los trabajos menos flexibles (regla 4 de elección de operaciones) y en mantener alimentadas las máquinas cuello de botella (programando lotes de transferencia entre máquinas de manera que se mantenga un inventario suficiente de piezas frente a la segunda máquina). De esta manera se evita que el cuello de botella permanezca ocioso por falta de suministro, ya que puede que posteriormente no sea posible recuperar el tiempo perdido.

5.2.4 Pseudocódigo

En la Figura 4 se muestra el pseudocódigo del algoritmo desarrollado

Paso 0 (inicialización).

- Situar en E las primeras operaciones de las piezas, con sus valores $r_{1,i}$ correspondientes.
- Para cada máquina j, determinar el valor fp_j.
- Determinar fp_{min} (y la máquina asociada).

Paso 1 (elección de la máquina).

- Si $fp_{min} = \infty$, se han programado todas las operaciones. En caso contrario, elegir la máquina según fp_{min} (Regla 1). En caso de empate utilizar las Reglas 2 y 3 de elección de máquina.

Paso 2 (elección de la operación).

- Si existe una sola operación candidata tomarla. Sino, aplicar criterio de selección de operaciones.

Paso 3 (actualización).

- Programar la operación (*k, i*) elegida determinando sus instantes de inicio ($t_{inicio\ (k,i)}$) y de fin ($t_{final\ (k,i)}$).
- Pasar la operación elegida al conjunto P y guardarla en la secuencia con sus instantes de inicio y fin. Si no es la última operación de la pieza *i*, pasar la operación siguiente de *i* de N a E.
- Actualizar los valores de disponibilidad de operación siguiente $r_{(k+1,i)}$ y los de la máquina asociada.
- Calcular $rp_{k,i,j'}$= max (*rk, i, fj'*) y determinar fp_{min} = min ($rp_{k,+1,i}$)
- Volver al paso 1.

Paso 4 (función objetivo).

- Calcular el retraso medio según la fórmula (1).

Figura 4. Pseudocódigo del algoritmo de *dispatching*

6. Experiencia computacional

La aplicación genera, de forma predeterminada, una única secuencia (solución) según el procedimiento descrito anteriormente. Con el fin de explorar un mayor número de soluciones y mejorar la solución inicial, se ha añadido la posibilidad de ejecutar el procedimiento en modo aleatorizado: el usuario pondera las reglas de prioridad de elección de las operaciones, introduciendo un valor de probabilidad para cada una de las reglas, e indica el número de secuencias que desea obtener.

Se han considerado 10 ejemplares correspondientes a 10 pedidos reales del cliente. Para cada caso, en primer lugar se ejecuta el algoritmo en modo predeterminado y después se ejecuta en modo aleatorizado con 1000 repeticiones. La mejor solución se compara con la obtenida en la

empresa. Los resultados (Tabla 1) muestran que el algoritmo aleatorizado mejora en todos los casos los valores de retraso medio obtenidos en la empresa en un 22,93% en promedio.

Tabla 1. Resultados obtenidos.

Nº Prueba		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**% Mejora media**
Retraso medio (h)	**Algoritmo**	6,95	4,91	10,69	6,62	11,41	10,63	14,91	7,98	16,55	10,61	**22,93**
	Empresa	8,07	5,88	12,24	8,37	16,82	13,5	18,5	12,97	22,56	14,63	
% Mejora		13,88	16,50	12,66	20,91	32,16	21,26	19,41	38,47	26,64	27,48	

Todos los test se han realizado en un Pentium IV con 896 MB RAM y un procesador de 3.0 GHz. En la Tabla 2 se muestra los tiempos medios, en segundos, necesarios para resolver un ejemplar.

Tabla 2. Tiempos medios, en segundos, de resolución de un ejemplar.

Método		**Nº Repeticiones**	**Tiempo (s)**
Algoritmo	**Directo**	1	0,03
	Aleatorio	1	0,03
		500	17,59
		1.000	34,98
Empresa		1	18.000,00

7. Conclusiones

En este trabajo se presenta un procedimiento de programación de la producción para resolver un caso real de asignación y secuenciación de operaciones de una planta del sector automoción, en la que se requiere minimizar el retraso medio en las entregas al cliente. El problema se ha identificado como un caso de taller mecánico con máquinas en paralelo y el retraso medio como función objetivo. El procedimiento diseñado es un algoritmo de *dispatching* implementado informáticamente, en el que se han planteado dos variantes de ejecución: por un lado, un método directo, que proporciona una solución determinista inicial, y por otro, un método aleatorizado en el que se permite al usuario ponderar probabilísticamente los criterios de prioridad, con el fin de intentar mejorar la solución inicial.

A partir de los datos de máquinas y piezas iniciales, la aplicación proporciona la secuencia de fabricación de las operaciones con su retraso medio asociado, las máquinas a las que deben asignarse y los tiempos de inicio y final de fabricación. Para afrontar posibles reprogramaciones que deban realizarse para adaptar el programa generado a imprevistos, se ha tratado que el programa sea robusto introduciendo reglas de robustez en el propio algoritmo.

Los resultados obtenidos muestran que el procedimiento propuesto mejora sensiblemente el retraso medio en comparación con el método utilizado por la empresa, del orden de 23% en modo aleatorizado. Además, se obtienen otros beneficios secundarios, entre los que destaca el gran ahorro en el tiempo de ejecución y en los tiempos muertos de máquina, la mejora en la información para la toma de decisiones del planificador y, en definitiva, una mayor competitividad en los plazos de entrega.

Referencias

Aiex, R.M., Binato S., Resende M.G.C. (2003). Parallel GRASP with Path-Relinking for Job Shop Scheduling. Parallel Computing, 29. pp. 393-430.

Akers, S.B., Friedman J., (1955). A Non-Numerical Approach to Production Scheduling Problems, Operations research, 3 pp. 422-429.

Allahverdi, A., Cheng, T.C.E. and Kovalyov, M.Y. (2008). A Survey of Scheduling Problems with Setup Times or Costs. European Journal of Operational Research 187, pp. 985-1032.

Applegate, D., Cook, W. (1991). A Computational Study of the Job-Shop Scheduling Problem, ORSA Journal on Computing, Vol.3, No.2, pp. 149-156.

Balas, E. (1969). Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm, Operations Research, Vol.17, No.6. (Nov. - Dec.), pp. 941-957.

Brucker, P., Schile, R. (1990). Job-Shop Scheduling with Multi-Purpose Machines, Computing 45 (4), pp. 369–375.

Gao, J., Gen, M., Sun, L. (2006). Scheduling Jobs and Maintenance in Flexible Job shop with a Hybrid Genetic Algorithm, Journal of Intelligent Manufacturing, Vol.17, No.4, pp. 503-512.

Giffer, B., Thompson, G.L. (1960). Algorithms for Solving Production Scheduling Problems, Operations Research, Vol.8, pp. 487-503.

Gomes, M.C., Barbosa-Povoa, A.P, Novais A.Q. (2005). Optimal scheduling for flexible-job shop operation. International of Production Research, Vol.11, No.43, pp. 2323–2353.

Jackson, J.R., (1956). An extension of Johnson's Result on Job Lot Scheduling, Naval Research Logistics Quaterly, Vol.3, pp. 201-203.

Kim, D.W. Kim, K.H. Jang, W., Chen, F.F. (2002). Unrelated Parallel Machine Scheduling with Setup Times Using Simulated Annealing. Robotics and Computer Integrated Manufacturing Vol.18, pp. 223-231.

Muth, J.F., Thompson G.L. (1963). Industrial Scheduling, Prentice-Hall, Englewood Cliffs, N.J.

Oliveira, J.A.V. (2000). Aplicação de Modelos e Algoritmos de Investigação Operacional ao Planeamento de Operações em Armazéns, Ph.D. Thesis, Universidade do Minho, Portugal.

Ravetti, M.G. Mateus, G.R. Rocha P.L. and Pardalos, P.M. (2007). A Scheduling Problem with Unrelated Parallel Machines and Sequence Dependent Setups. International Journal of Operational Research Vol.2, pp. 380-399.

Roy, B, Sussman, B. (1964). Les Problemes d'Ordonnancement avec Constraintes Disjonctives. SEMA, Note D.S., No 9, Paris.

Xing, L., Chen, Y., Yang, K. (2009). Multi-objective Flexible Job-Shop Scheduling Problem: Design and Evaluation by Simulation Modeling. Applied Soft Computing, Vol.9 No.1, pp. 362-376.

Wang, L., Zheng, D. (2001). An Effective Hybrid Optimisation Strategy for Job-Shop Scheduling Problems, Computers & Operations Research, Vol. 28, pp. 585-596.

Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevastjanov, S., Schmoys, D.B. (1997). Short Shop Schedules, Operations Research, Vol.45 (2), pp. 288-294.