Secuenciación en una máquina de la vida real con tiempos de preparación dependientes de la secuencia

Mayra D'Armas[1], Ramón Companys[2]

[1] Dpto. de Ingeniería Industrial. Universidad Nacional Experimental Politécnica "Antonio José de Sucre". Urb. Villa Asia, Final Calle China 1209, Puerto Ordaz, Venezuela. mdarmas@bqto.unexpo.edu.ve
[2] Dpto. de Organización de Empresas. Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Cataluña. Av. Diagonal 647 7º 08028 Barcelona, España. ramon.companys@upc.edu

Resumen

Este trabajo considera el problema de la programación de la producción en una máquina de la vida real, donde los tiempos de preparación están separados de los tiempos de procesamiento, y son dependientes de la secuencia, con el objetivo de minimizar el retraso total. El objetivo fue desarrollar y evaluar el comportamiento de las metaheurísticas Recocido Simulado, Búsqueda Tabú, GRASP y Algoritmos Genéticos para el problema planteado, mediante una investigación no experimental del tipo exploratoria y evaluativa. Los algoritmos propuestos se codificaron en lenguaje Visual Basic 6.0. La experimentación computacional se realizó con una colección de datos reales de una empresa venezolana del sector metal. Los datos que se tomaron correspondieron a las fechas de vencimientos de los pedidos, los tiempos de preparación de la máquina, los tiempos de procesamiento de los pedidos, la cantidad de pedidos y las familias de los pedidos. Los resultados computacionales revelan que la Búsqueda Tabú es un procedimiento que puede proporcionar buenas soluciones para el problema específico estudiado cuando se considera como objetivo el retraso total con tiempos de preparación dependientes de la secuencia.

Palabras clave: secuenciación, una máquina, metaheurísticas, tiempos de preparación

1. Introducción

En este trabajo se presenta un estudio llevado a cabo en una empresa real venezolana dedicada a la fabricación de productos de acero, cuyo proceso productivo es reducible a una máquina, con tiempos de preparación que dependen de la secuencia de producción de los pedidos, con una cartera de productos agrupados por familias que se diferencian entre sí por las propiedades mecánicas, y cuya fabricación es bajo pedido con fechas de vencimiento. El propósito de este trabajo es desarrollar metaheurísticas que den resultados de calidad en un tiempo razonable, que determinen la secuencia de operaciones en una máquina con el objeto de cumplir los plazos de entrega, o en su defecto minimizar el retraso total. La empresa estudiada produce alambrón, que es un producto de acero de sección transversal circular y superficie lisa, que se obtiene por laminación en caliente de palanquillas. El producto se fabrica con un total de dieciocho familias de productos diferentes que tienen un diámetro que varía desde 5,5 mm hasta 12,5 mm. El número de pedidos fabricados en cada uno de los ejemplares estudiados varía entre ocho y dieciséis. Los tiempos de procesamiento de los pedidos varían desde dos hasta ciento noventa y ocho horas, y los tiempos de preparación, que son dependientes de la secuencia varían desde una hasta cuatro horas.

2. Notación usada

n número de pedidos

i índice de los pedidos; $i \in I = \{1,2,3,\ldots,n\}$

p_i tiempo de operación del pedido *i*

g_i familia a la que pertenece el pedido *i*

ST_{hi} tiempo de preparación del pedido *i* cuando *h* ha sido el pedido anterior de la secuencia

d_i instante comprometido de salida del pedido *i*

r_i instante de entrada del pedido *i* en el taller

c_i instante en que el pedido sale del taller

w_i tiempo de espera del pedido *i*

L_i diferencia entre el instante de salida real y el previsto

T_i retraso

3. Formulación del problema

Se tiene un conjunto de *n* pedidos que deben ser secuenciados en una máquina. Se asume que la máquina puede procesar un solo pedido a la vez y que está disponible en el instante cero. Los pedidos están disponibles al inicio del proceso y tienen un tiempo d_i límite de entrega. Se asume que los pedidos están clasificadas en g_i familias y que el tiempo de preparación ST_{hi} se produce cuando se pasa de una familia a otra. El tiempo de preparación de la máquina depende de la familia del pedido a ser procesado y la familia del pedido que lo precede. Para cada pedido *i* se conoce el tiempo de operación p_i; el tiempo de entrega comprometida d_i, la familia g_i a la que pertenece y los tiempos de preparación *ST* de la máquina al pasar de una familia a otra.

Según la secuencia escogida, para cada pedido *i* el instante en que sale del taller c_i, viene dado por (1):

$$c_i = r_i + w_i + p_i + ST_{hi} \quad (1)$$

y su retraso T_i viene dado por (2):

$$T_i = \max\{0, c_i - di\} \quad (2)$$

El objetivo (3) es encontrar una secuencia de los pedidos que minimice la suma del retraso de los pedidos:

$$[\text{MIN}]\ Z = \sum T_i \quad (3)$$

4. Soluciones iniciales

Se aplicaron cuatro reglas que permitieron generar soluciones iniciales, las dos primeras estáticas y las dos últimas dinámicas: (1) EDD: earliest due date; (2) SST-EDD: shortest setup time–earliest due date; (3) Índice CR_1: Cociente; y (4) Índice CR_2: Suma ponderada. (D'Armas y Companys, 2005)

– La regla EDD, comúnmente usada, que consiste en ordenar los pedidos de acuerdo con la fecha de vencimiento se adaptó a la consideración de tiempos de preparación dependientes de la secuencia.

– La regla SST-EDD consiste en ordenar los pedidos por familias de acuerdo con el tiempo de preparación más corto cuando se cambia de una familia a otra (para favorecer un tiempo mínimo de cambios entre familias), y, además, secuenciar los pedidos entre familia por orden creciente de fechas de vencimiento.

– La regla CR_1 consiste en calcular el índice de prioridad por medio de la fecha de vencimiento dividida por la suma entre el tiempo de preparación de la familia del pedido *i* (dependiente de la familia del pedido anterior *h*) y el tiempo de operación del pedido *i* (4). Los pedidos que tienen menor CR_1 se asignan primero.

$$CR_1 = d_i/(ST_{hi}+p_i) \quad (4)$$

– La regla CR_2 pondera la fecha de vencimiento, el tiempo de preparación de la familia del pedido *i* y el tiempo de operación del pedido *i* (5) asignando primero los pedidos que tienen índices más pequeños.

$$CR_2 = 0{,}2d_i+0{,}8(p_i+ST_{hi}) \quad (5)$$

5. Heurísticas de mejoras

Se desarrollaron y evaluaron las metaheurísticas: Recocido Simulado, Búsqueda Tabú, GRASP y Algoritmos Genéticos. (D'Armas, 2006)

5.1. Recocido Simulado (SA)

SA fue desarrollado por Kirkpatrick et al (1983) y se le conoce como Recocido Simulado dado la analogía entre la simulación del recocido de sólidos y el problema de la resolución de los grandes problemas de optimización combinatoria. Para el problema estudiado, la regla de generación de la probabilidad de aceptación de la solución vecina que es peor que la solución en curso sigue la distribución Boltzmann (6):

$$\pi(X_{vec}) = e^{-\frac{f(X_{vec})-f(X_{cur})}{T}} \quad (6)$$

Para la implementación del SA se tomaron las soluciones iniciales EDD, SST-EDD, CR_1 y CR_2, seleccionándose la mejor de las cuatro soluciones mejoradas. La generación de los vecinos se realizó al azar. Para ello se generan, aleatoriamente, dos piezas a permutar en la secuencia dando lugar a una nueva secuencia vecina. La secuencia vecina se diferencia de la secuencia madre en el orden de realización de dos pedidos. Se tomó una Temperatura inicial T= 0,4 y una tasa de enfriamiento r= 0,95. Como condición para finalizar la ejecución del algoritmo, se consideró un número máximo de iteraciones o cuando no se produce mejora de la solución tras un número concreto de iteraciones.

5.2. Búsqueda Tabú (TS)

TS es un procedimiento metaheurístico introducido y desarrollado en su forma actual por Fred Glover (1989). Para la implementación del TS se tomaron las soluciones iniciales EDD, SST-EDD, CR_1 y CR_2, seleccionándose la mejor de las cuatro soluciones mejoradas. La generación de los vecinos se realizó mediante la exploración de todo el entorno de la solución en curso, es decir, dada una secuencia se generan todos los vecinos mediante la permutación entre dos

pedidos, haciéndose todos los cambios posibles de dos posiciones (que podrán ser contiguas o no). Por consiguiente, los vecinos se generan de modo secuencial, comenzando por la permutación de los dos primeros pedidos y terminando con la permutación de los dos últimos.

Los mejores vecinos de la solución en curso se guardan en una lista cuya dimensión es proporcional a la dimensión del vecindario, teniendo una capacidad de almacenamiento del 15% de las vecinas generadas. El nivel de aspiración selecciona aquellas secuencias de pedidos cuyo valor de ΣT sea inferior al de la mejor solución hallada hasta el momento. Primero se comprueba si la solución vecina actual satisface el nivel de aspiración, si es así, la solución vecina sustituye a la solución en curso y a la mejor. En caso contrario, si dicha solución vecina es igual o peor que la mejor, entonces se compara con los atributos de la lista tabú. El algoritmo finaliza cuando se alcanza un número máximo de iteraciones o cuando no se produce mejora de la solución tras un número concreto de iteraciones.

5.3. GRASP

GRASP fue desarrollado originalmente por Feo y Resende (1989). Cada iteración en GRASP consta de dos pasos: la fase de construcción y el procedimiento de búsqueda local, en el primero se construye una solución tentativa, que luego es mejorada mediante un procedimiento de intercambio hasta que se llega a un óptimo local.

En la primera Fase, se construyó una solución inicial a partir de Índice Crítico CR_2, secuenciándose los pedidos en orden no decreciente de su índice. Se toma el menor valor CR_2, y se determina un valor de referencia mediante un incremento del veinte por ciento del menor valor. Se elige al azar uno de los pedidos cuyo índice sea menor que el valor de referencia, y se asigna como primer pedido a procesar. Se recalculan los índices CR_2 y se ordenan nuevamente los pedidos, desde la segunda posición, de acuerdo con un orden no decreciente de los índices recalculados.

En la segunda Fase, el procedimiento de mejora aplica una variante del Algoritmo No Exhaustivo de Descenso (ANED), a partir de la solución en curso se generan y evalúan en cierto orden sus vecinos, si uno de ellos es mejor que la solución en curso se toma como nueva solución en curso (sin terminar la generación de los vecinos de la solución primitiva) y se prosigue aplicando el procedimiento a los vecinos de la nueva solución en curso; cuando se han generado todos los vecinos de una determinada solución en curso sin que ninguno sea mejor el procedimiento se da por terminado. En la variante del ANED cada iteración recorre el conjunto de soluciones de forma distinta pudiendo acceder a vecindarios no explorados. Para ello se define un nuevo vector de posiciones que permite codificar los punteros de posición que se usan durante la exploración del vecindario.

5.4. Algoritmos Genéticos (GA)

GA fue desarrollado por Holland (1975) y se distingue de los anteriores por el hecho de que en cada iteración se tiene un conjunto de soluciones y no una única solución en curso.

En este trabajo la dimensión de la población se tomó igual al número de pedidos de tal manera que la población tiene una relación directamente proporcional con la dimensión del problema a tratar. En cuanto a su composición, la población inicial se dividió en dos partes iguales, las cuales se diferencian entre sí por el esquema de confección de los cromosomas que conforman dicha población. La mitad de la población se forma secuenciando todos los pedidos, en orden creciente, de acuerdo con la fecha de vencimiento (EDD). Posteriormente, se generan aleatoriamente, dos genes a permutar en la secuencia dando lugar a una nueva secuencia. La

otra mitad, se forma secuenciando todas las operaciones de acuerdo con el Índice Crítico CR_2, en orden no decreciente de su índice. Posteriormente, se generan aleatoriamente, dos genes a permutar en la secuencia dando lugar a una nueva secuencia.

El proceso de selección se realizó mediante la combinación de las mejores secuencias y de forma aleatoria, es decir un 50% de las secuencias a cruzar lo conforman las mejores soluciones de la población actual y el otro 50% se elige aleatoriamente. Las soluciones progenitoras del grupo seleccionado se emparejaron por orden de bondad de la solución, la primera solución se empareja con la segunda, la tercera con la cuarta, etc. dentro de la lista de soluciones elegidas ordenadas de acuerdo con el valor de ΣT. El procedimiento de cruce que se aplicó fue el Cruce PMX (de Partially-Matched Crossover).

Para el problema, la probabilidad de mutación se fijó en un treinta por ciento, el procedimiento genera un número aleatorio entre 0 y 1, si este valor es inferior al parámetro de probabilidad fijado (0,3) se realiza la mutación en la secuencia. En el supuesto que lo haga, la mutación consiste en intercambiar un gen, seleccionado al azar, con su consecutivo en la secuencia. En el proceso de regeneración se seleccionaron las mejores secuencias de las dos poblaciones de soluciones, progenitoras y descendientes, por lo que la nueva población generada para la siguiente iteración está formada por la mitad del total de las soluciones (padres + descendientes).

6. Utilización de los datos reales

La recolección de los datos se realizó a partir de los informes y reportes mensuales del área de programación y control de la producción. Se estudiaron diecisietes meses. La información procesada contempla: (a) los tipos de productos a fabricarse, en cada pedido, y sus respectivas cantidades requeridas expresadas en toneladas; (b) la familia a la que pertenece cada pedido; (c) el tiempo de entrega de cada pedido en horas; (d) los tiempos de preparación de la máquina para pasar de una familia de productos a otra, expresado en horas; (e) el tiempo unitario de producción de cada familia de productos, expresado en horas/ toneladas; y (f) el tiempo de operación de cada pedido (en horas). En cuanto a la preparación de la máquina, para pasar de una familia de productos a otra, se requiere cambiar los cilindros y anillos laminadores del tren de laminación, lo que origina unos tiempos de preparación que dependen tanto de la familia a ser fabricada como de la inmediatamente precedente.

7. Experiencia Computacional

Las metaheurísticas fueron codificadas en lenguaje Visual Basic 6.0, y la experiencia computacional se realizó en un ordenador personal Pentium IV de 3,00 GHz y 1,00 GB de Ram. La validación de los algoritmos se realizó con una colección de mil ejemplares del Laboratorio de Organización Industrial del Departamento de Organización de Empresas de la UPC, con capacidad para procesar hasta 25 pedidos y 6 familias. Posteriormente se realizaron las corridas con los datos reales. Los resultados de los retrasos totales, expresados en horas, para cada uno de los ejemplares reales estudiados se muestran a continuación en la Tabla 1. Se ha indicado para cada ejemplar el número de pedidos *n* y de familias *g*, señalándose con un asterisco la mejor solución, es decir el menor retraso total.

Tabla 1. Caso real. Retraso total expresado en horas.

Ejemplar	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
n	16	15	8	12	16	14	17	15	16	15	11	15	15	14	15	8	13
g	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
SA	265	647	192	587	507	622	931	632	1343	407	0	309	1202	34	0*	0*	255
GRASP	269	639	194	588	483	610	857*	617	1245	332	12	305	1152	38	0*	0*	218
TS	17*	563*	0*	499*	341*	574*	857*	613*	919*	149*	0*	48*	854*	20*	0*	0*	17*
GA	274	649	192	597	489	638	918	639	1295	422	15	307	1161	33	3	7	270

El promedio de estos resultados se puede apreciar en la Figura 1, pudiéndose deducir que el menor retraso medio lo arroja TS y que los mayores retrasos los arrojan SA y GA.

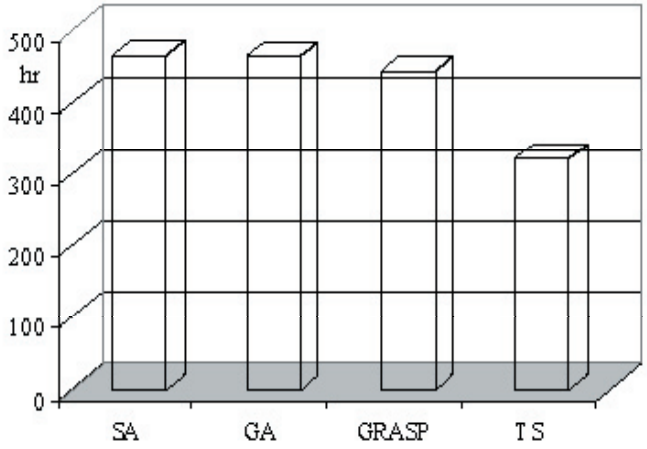

Figura 1. Caso real. Retraso total promedio (horas)

El 100% de las mejores soluciones se obtuvo con TS, seguido por GRASP con un 17,8% y SA con un 11,8%. Siendo GA el procedimiento menos adecuado para resolver el problema estudiado, debido a que en ninguno de los casos logró alcanzar los mejores resultados. Ver Figura 2.

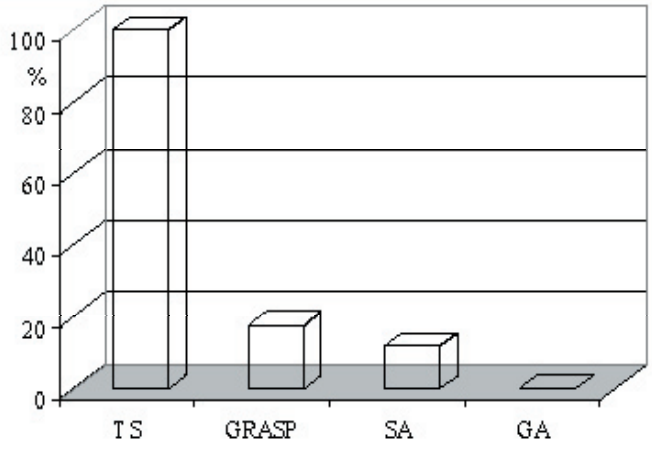

Figura 2. Caso real. Porcentaje de mejores resultados.

Las metaheurísticas desarrolladas en este trabajo se juzgaron por su eficacia (el esfuerzo realizado para obtener la solución) y su calidad (la diferencia entre una solución heurística y la óptima). Para juzgar la eficacia, se computó el tiempo promedio para resolver un ejemplar. Para juzgar la calidad, se compararon las soluciones obtenidas mediante la aplicación de cada uno

de los procedimientos.

Debido a que en el problema estudiado no se tiene la solución óptima, la comparación se llevó a cabo determinando el porcentaje de error relativo, es decir la diferencia entre la solución heurística y la mejor solución según lo propuesto por Rajendran y Ziegler (2003), Allahverdi y Aldowaisan (2000), Yang, Kreipl y Pinedo (2000), Raghu y Rajendran (1995), mediante la expresión (7):

% error relativo= [100. (Heurística- Mejor Solución)/Mejor Solución] (7)

Los porcentajes de error relativo de las soluciones obtenidas en cada uno de los ejemplares estudiados, se presentan en la Tabla 2. De esta tabla se puede deducir que el procedimiento que arroja mejores resultados es TS, ya que tiene porcentajes de error relativo muy bajos. Además, se observa que % de error relativo es muy elevado para SA, GRASP y GA.

Tabla 2. Caso real. Porcentaje de error relativo.

	SA	GRASP	TS	GA
1	1458,8	1482,4	0,0	1511,8
2	14,9	13,5	0,0	15,3
3	19200,0	19400,0	0,0	19200,0
4	17,6	17,8	0,0	19,6
5	48,7	41,6	0,0	43,4
6	8,4	6,3	0,0	11,1
7	8,6	0,0	0,0	7,1
8	3,1	0,7	0,0	4,2
9	46,1	35,5	0,0	40,9
10	173,2	122,8	0,0	183,2
11	0,0	1200,0	0,0	1500,0
12	543,8	535,4	0,0	539,6
13	40,7	34,9	0,0	35,9
14	70,0	90,0	0,0	65,0
15	0,0	0,0	0,0	300,0
16	0,0	0,0	0,0	700,0
17	1400,0	1182,4	0,0	1488,2

En la Figura 3 se muestran gráficamente los tiempos promedios de ejecución, expresados en segundos, necesarios para resolver un ejemplar usando cada uno de los procedimientos. Puede observarse que los tiempos promedios para resolver un ejemplar no llegan a un (1) segundo. Siendo SA la que consume mayor cantidad de tiempo.

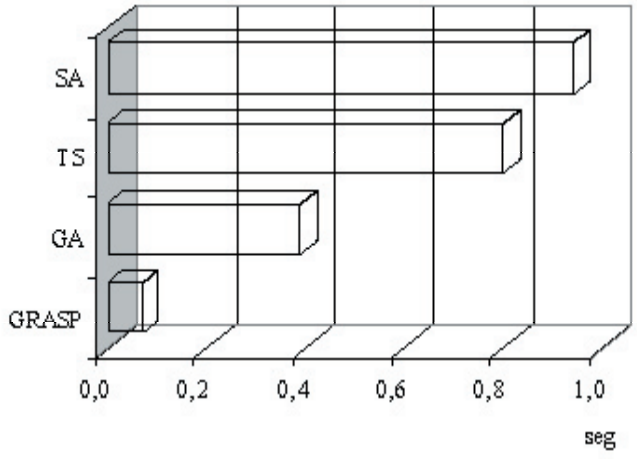

Figura 3. Caso real. Tiempo promedio de ejecución (segundos)

8. Conclusiones

A partir de los resultados obtenidos se puede concluir que TS es un procedimiento que puede proporcionar buenas soluciones para el problema específico estudiado cuando se considera como objetivo el retraso total con tiempos de preparación dependientes de la secuencia, ya que necesita poco tiempo de procesamiento y logra las mejores soluciones. GRASP puede tener una ventaja sobre TS para problemas con gran cantidad de pedidos *n* debido a que necesita menos tiempo para encontrar una buena solución. GA es la que arroja los mayores retrasos, en vista de esto no es aconsejable su aplicación para el ambiente industrial estudiado. Teniendo en cuenta la complejidad del problema estudiado, se concluye que los resultados obtenidos son satisfactorios. Resaltando la adaptabilidad de las metaheurísticas al problema de una máquina con tiempos de preparación dependientes de la secuencia de los pedidos.

La empresa encontró los resultados muy prometedores y está considerando la utilización de los procedimientos propuestos en su sistema de programación. La investigación presentada en este trabajo puede ser tomada como una base para posibles extensiones enfocadas a considerar otro tipo de variaciones del problema estudiado, incorporando elementos que en ocasiones también aparecen en determinados ambientes industriales reales: introducir restricciones de costos de penalización, modelar el problema teniendo en cuenta la incertidumbre o priorizar las órdenes de producción de acuerdo con la importancia o pesos relativos. En cuanto a las metaheurísticas, se sugiere continuar investigando sobre el impacto de los diferentes elementos que forman los procedimientos propuestos con el fin de medir sus rendimientos, particularmente si se incrementa el número de pedidos *n*. Para el caso de TS sería interesante investigar los efectos de diferentes estrategias de diversificación y del tamaño de la lista tabú; en cuanto a GA sería recomendable probar otros métodos de cruces y de mutación.

Referencias

Allahverdi A.; Aldowaisan T. (2000). No-wait and separate setup three-machine flowshop with total completion time criterion. International Transactions in Operational Research, Vol. 7, pp. 245-264

D' Armas M.; Companys R. (2005). Procedimientos de exploración de entornos para la programación de operaciones en una máquina con tiempos de preparación. Documento Interno de Trabajo. Departamento de Organización de Empresas. UPC.

D' Armas M. (2006). Programación de operaciones en una máquina con tiempos de preparación dependientes de la secuencia. Aplicación a la industria metalmecánica. Tesis Doctoral. Departamento de Organización de Empresas. UPC.

Feo T.; Resende. (1989). A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters. Vol. 8, pp. 67-71

Glover F. (1989). Tabu Search, Part I. ORSA Journal on Computing. Vol. 1, pp. 190-206.

Holland J. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press.

Kirkpatrick S.; Gelatt CD.; Vecchi MP. (1983). Optimization by simulated annealing, Science, Vol. 220, pp. 661-680.

Raghu T.S.; Rajendran C. (1995). Due-date setting methodologies based on simulated annealing-an experimental study in a real-life job shop. International Journal of Production Research, Vol. 33, No. 9, pp. 2535-2554

Rajendran C.; Ziegler H. (2003). Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times, European Journal of Operational Research, Vol. 149, No. 3, pp. 513-522

Yang Y.; Kreipl S.; Pinedo M. (2000). Heuristics for minimizing total weighted tardiness in flexible flow shops. Journal of Scheduling, Vol. 3, No. 2, pp. 89-108