PROGRAMACIÓN DE LA PRODUCCIÓN EN SISTEMAS DE MANUFACTURA TIPO TALLER CON EL ALGORITMO COMBINADO CUELLO DE BOTELLA MÓVIL Y BÚSQUEDA TABÚ*

Production Programming in Manufacturing Systems (Workshop Type) with a Combined Mobile Bottleneck and Taboo Search Algorithm

*Rodrigo Alberto Britto Agudelo***

*Gonzalo Mejía Delgadillo****

*Juan Pablo Caballero Villalobos*****

Resumen: la programación de la producción en manufacturas tipo taller (*job shop*) encuentra muchas aplicaciones en sistemas reales de producción, como empresas metalmecánicas, de impresión, de textiles y otras más. En general, en estos sistemas de manufactura, el objetivo principal es entregar los trabajos a tiempo. En esta investigación se propone un enfoque híbrido que utiliza la heurística del *cuello de botella móvil* (CBM) o *shifting bottleneck* y la *búsqueda tabú* (BT) con el objetivo de minimizar la tardanza ponderada total. La heurística CBM provee una solución inicial factible que sucesivamente es mejorada por el método de BT. Adicionalmente, en este trabajo se realizaron varias mejoras sobre los

* *Fecha de recepción: 14 de junio de 2007. Fecha de aceptación para publicación***:** *10 de septiembre de 2007. Este artículo se deriva del proyecto de investigación* Programación de Job Shop a través del uso de Tabu Search sobre las operaciones de ruta crítica del grafo disyuntivo, *realizado en el marco de la Maestría en Ingeniería Industrial de la Universidad de los Andes.*

** *Ingeniero químico, Universidad Nacional de Colombia. Magíster en Ingeniería Industrial, Universidad de los Andes, Colombia. Profesor, Facultad de Administración, Universidad de los Andes, Colombia. Correo electrónico: rab@adm.uniandes.edu.co*

*** *Ingeniero mecánico, Universidad de los Andes, Colombia. Ph. D. in Industrial Engineering, Lehigh University, Estados Unidos. Profesor asociado, Departamento de Ingeniería Industrial, Universidad de los Andes, Colombia. Correo electrónico: gmejia@uniandes.edu.co*

**** *Ingeniero Industrial, Pontificia Universidad Javeriana, Colombia. Magíster en Ingeniería Industrial, Universidad de los Andes, Colombia. Profesor asistente, Departamento de Procesos Productivos, Pontificia Universidad Javeriana, Colombia. Correo electrónico: juan.caballero@javeriana.edu.co*

algoritmos clásicos CBT y BT, como nuevos criterios para la escogencia de las máquinas críticas o cuello de botella y novedosas estrategias de diversificación e intensificación. El desempeño de la heurística propuesta (denominada CBBT) se evaluó con 17 problemas clásicos de la literatura sobre el tema. La heurística implementada muestra resultados muy competitivos comparados con otros enfoques encontrados en la literatura tanto en la calidad de las soluciones como en el tiempo computacional.

Palabras clave: sistemas flexibles de manufactura, programación de la producción, algoritmos.

Abstract: Job Shop Scheduling has many applications in real production systems such as metal machining, printing, and textiles, among others. Commonly, in these manufacturing systems the main objective is the delivery of the jobs on time. In this research we present a hybrid approach that uses the Shifting Bottleneck (SB) and Tabu Search (TS) heuristics with the purpose of minimizing the Total Weighted Tardiness. The Shifting Bottleneck algorithm provides a feasible initial solution which is iteratively improved by the TS method. Additionally, several improvements were performed on the classical algorithms SB and TS such as new criteria for the selection of the critical machines and a number of innovative strategies of diversification and intensification. The performance of the proposed heuristic algorithm denominated CBBT was evaluated with 17 classical problems found in the literature. The implemented heuristic algorithm shows very competitive results compared with other approaches found in literature both in quality of the solutions and computational time.

Key words: Flexible manufacturing systems, production scheduling, algorithms.

INTRODUCCIÓN

Programar la producción constituye una de las tareas más difíciles que enfrentan las empresas que cuentan con sistemas tipo taller (*job shop*). Este problema ha sido ampliamente estudiado en la literatura sobre el tema usando como función objetivo la terminación de todos los trabajos (*makespan*). Sin embargo, dado que la mayoría de estos sistemas opera bajo pedido (*make-to-order*), cumplir con las fechas de entrega de las órdenes representa probablemente un objetivo más importante (Baker y Hayya, 1982). Un programa de producción que no puede cumplir con los tiempos de entrega puede dar como resultado bajos índices de servicio e insatisfacción de los clientes.

El problema que se estudia en este trabajo consiste en programar un conjunto de trabajos en un taller de máquinas con el objetivo de minimizar la *tardanza ponderada total* definida como:

$$\sum_{j \in J} w_j T_j \qquad (1)$$

Donde:

w_j: importancia o prioridad del trabajo $j \in J$ (conjunto de trabajos)

T_j: $\max (C_j - d_j, 0)$ con C_j y d_j, que representan respectivamente el tiempo de finalización y la fecha de entrega del trabajo j.

El problema de programación tipo taller (JSSP, por su sigla en inglés) ha sido estudiado por diferentes autores y es un problema no polinomial difícil (NP-Hard) (Grahman *et al.*, 1979). Muchos métodos han sido propuestos para resolverlo. Entre estos se cuentan métodos exactos como algoritmos de ramificación y acotación y métodos heurísticos de búsqueda local. Entre los enfoques de ramificación y acotación están los desarrollados por Carlier y Pison (1989), que permitieron comprobar soluciones óptimas para problemas de 10×10 (10 trabajos y 10 máquinas).

Para problemas más grandes se han utilizado recientemente heurísticas basadas en descomposición, en especial el algoritmo del *cuello de botella móvil*, desarrollado por Adams, Balas y Zawack (1988) y en búsquedas locales como son *búsqueda tabú* (Glover y Laguna, 1997), *enfriamiento simulado* (Van Laarhoven *et al.*, 1992; Wang y Wu, 2000) y *algoritmos genéticos* (Ponnambalam, Aravinda y Sreenivasa, 2001).

Para problemas cuya función objetivo es minimizar la tardanza ponderada total la literatura sobre el tema es escasa. Con frecuencia, para este problema se han utilizado reglas de despacho, aunque su desempeño no es muy confiable (Anderson y Nyirenda, 1990; Vepsalainen y Morton, 1987). La heurística cuello de botella móvil fue adaptada para el objetivo de tardanza ponderada total por Pinedo y Singer (1999). Otra heurística para el problema de *job shop* con objetivo de la tardanza ponderada total es la heurística desarrollada por Asano y Ohta (2002).

1. Marco teórico

1.1 Cuello de botella móvil

El cuello de botella móvil (CBM) o *shifting bottleneck*, desarrollado por Adams, Balas y Zawack (1989), es una de las heurísticas para programación de talleres más exitosas (Pinedo y Singer, 1999). Su idea principal es secuenciar una a una las máquinas del taller según su grado de criticidad. Este último es calculado de acuerdo con la secuenciación de las máquinas que afecta la función objetivo. El algoritmo inicia con la determinación del cuello de botella inicial y establece una secuencia de trabajos óptima para esa máquina. Una vez se ha hecho esta operación, se imponen nuevas restricciones que afectan la secuenciación de las máquinas restantes. Se determina y se secuencia la nueva máquina cuello de botella entre aquellas no secuenciadas y el proceso se repite hasta que no haya máquinas sin secuenciar.

1.2 Algoritmos de búsqueda local

En forma general, una heurística local es un procedimiento que parte de una solución inicial (s_0) que pertenece a un conjunto finito de soluciones (S) y procede iterativamente a modificar esa solución s_0 para encontrar nuevas soluciones (s'). Una solución s' se obtiene a partir de otra solución s mediante una modificación parcial definida de s, a

lo cual se denomina un movimiento. El algoritmo se detiene cuando se cumple con un criterio de terminación que típicamente es un número predefinido de iteraciones. Allí se recupera la mejor solución $s^* \in S$.

Entre las heurísticas de búsqueda local, el enfoque de la búsqueda tabú (BT) se ha destacado por su eficiencia y calidad de sus soluciones (Glover y Laguna, 1997). Un algoritmo de búsqueda tabú inicia con una solución factible s_0. A continuación se examina el vecindario de la solución actual y se escoge el vecino cuya función objetivo sea la mejor. En ocasiones —definidas por el programador— se puede escoger un vecino con el cual se obtenga un valor peor de la solución objetivo solamente si esto contribuye a escapar de mínimo locales.

Muchos autores (Pezzella y Merelli, 2000; Nowicki y Smutnicki, 1996) han observado que tanto la escogencia de una buena solución inicial como el establecimiento del vecindario son los aspectos más importantes del desempeño del algoritmo en términos de la calidad de la solución y del tiempo computacional. Por ejemplo, en Armentano y Scrich (2000) se usaron varias reglas de despacho como solución inicial a la heurística BT en ambientes *job shop*, y se inicia la búsqueda con la solución que produjo el mejor resultado. En este trabajo se propone una nueva heurística compuesta, basada en la heurística CBM y el enfoque de BT.

2. Formulación del problema

2.1 Taller (*job shop*)

El taller es un sistema de manufactura de bajo volumen de producción y gran variedad de productos. En la literatura, este sistema está conformado por un conjunto de *n* trabajos y *m* máquinas. Los trabajos deben programarse buscando minimizar alguna función objetivo: por ejemplo, flujo máximo (*makespan*), tardanza media, retardo medio, tardanza máxima, etc. Los supuestos típicos para este tipo de problema son:

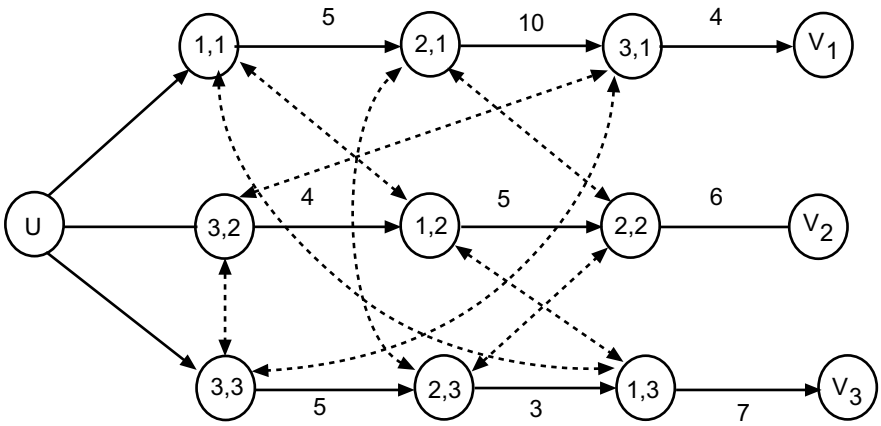
- Cada trabajo está compuesto por distintas operaciones que deben ser ejecutadas en un orden predefinido.
- No se permiten las interrupciones.
- Cada trabajo consiste de *m* operaciones distintas, una operación en cada máquina (no recirculación).
- Los tiempos de procesos son conocidos e independientes de la secuencia.
- Las máquinas no tienen fallas y siempre están disponibles durante el periodo de programación.
- Todos los trabajos están disponibles en tiempo cero (0) y ningún trabajo arriba después al sistema.

2.2 Grafo disyuntivo

Una de las representaciones más conocidas del problema de programación en un sistema tipo taller es el grafo disyuntivo $G(N,A,B)$, donde N es el conjunto de nodos; A, el conjunto de arcos conjuntivos, y B, el conjunto de arcos disyuntivos (Pezzella y Merelli, 2000). El nodo (i,j) representa la operación del trabajo j en la máquina i. Los arcos conjuntivos A representan las rutas que debe seguir cada trabajo: si un arco $(i,j) \rightarrow (l,j)$ es parte del conjunto A, el trabajo j debe ser procesado en la máquina i antes de ser procesado en la máquina l.

Cada arco $(i,j) \rightarrow (l,j)$ tiene una longitud $|(i,j) \rightarrow (l,j)| = p_{ij}$, que representa el tiempo de la operación (i,j). B es el conjunto de arcos disyuntivos, los cuales conectan (por medio de dos arcos con direcciones opuestas) las operaciones que se realizan en la misma máquina. Además, existen un nodo fuente, U, del cual emanan las primeras operaciones de cada uno de trabajos, y n nodos terminales $V_1 \ldots, V_n$, (Figura 1). La longitud del camino más largo desde la fuente U hasta el nodo terminal V_k representa el tiempo de terminación del trabajo k.

Figura 1. Grafo disyuntivo para un problema de programación tipo taller con n=3, m=3 con el objetivo de minimizar la tardanza ponderada



Fuente: presentación propia de los autores.

Una solución factible es una selección de arcos disyuntivos del conjunto B, que cumple que (i) dicha selección contiene un arco de cada par $(i,j) \leftrightarrow (k,j)$, y (ii) el grafo resultante $G:=(N, A, \sigma(B))$ es acíclico.

Sea $L(v,v')$ la longitud del camino crítico (ruta más larga) desde el nodo v al nodo v' en el gráfico $G:=(N,A, \sigma(B))$ —si no existe ningún camino, entonces $L(v,v')$ no está definido—. El tiempo de terminación C_j del trabajo j es igual a $L(U,V_j)$. Los valores de $L(U,v)$, al ser v un nodo arbitrario, son calculados utilizando algún algoritmo de ruta más larga. El algoritmo usado en este proyecto fue el algoritmo DAG (*Direct Acyclic Graph*). El desempeño de la heurística CBM depende en gran medida de la implementación de un algoritmo eficiente de ruta más larga.

La Tabla 1 señala un ejemplo de los trabajos y la ruta de proceso de un sistema tipo taller; en la Figura 1 se muestra el grafo disyuntivo correspondiente al problema de minimizar la tardanza ponderada total en un sistema tipo *job shop*.

Tabla 1. Grafo disyuntivo

	Secuencia de máquina	Tiempos de proceso
1	1,2,3	p_{11}=5, p_{21}=10, p_{31}=4
2	3,1,2	P_{32}=5, p_{12}=5, p_{22}=6
3	3,2,1	p_{13}=5, p_{23}=3, p_{13}=7

Fuente: presentación propia de los autores.

2.3 Heurística cuello de botella móvil

La heurística CBM se basa en la idea empírica de que el desempeño de un sistema depende del ritmo de la estación con mayor utilización. Descompone el problema tipo taller en un número *m* de subproblemas de una sola máquina y la secuenciación de trabajos se hace iterativamente en el recurso más crítico. Los principales pasos de una iteración en el algoritmo CBM son:

- Identificación y secuenciación en cada iteración de la máquina cuello de botella. Esto es, se resuelven *m–i* problemas de una sola máquina, donde *i* es el número de máquinas ya secuenciadas.
- En este caso, la máquina cuello de botella es la que causa un mayor aumento en la tardanza ponderada total.
- Reoptimización de la secuencia local para las máquinas ya secuenciadas. Esto es resolver de nuevo los problemas de una sola máquina en cada una de dichas máquinas para intentar una mejora en la función objetivo.

2.4 Algoritmo cuello de botella móvil

Sea M el conjunto de las m máquinas y M_0 el conjunto de máquinas que ya han sido secuenciadas:

Paso 1. G es el grafo disyuntivo del problema. Haga $M_0=\varnothing$.

Paso 2. Seleccione la máquina cuello de botella.

Para cada una de las operaciones (i,j) de la máquina. i entre las no secuenciadas (conjunto $M–M_0$), se debe definir las fechas de entrega d^k_{ij}. Esta fecha de entrega corresponde al máximo tiempo permitido de terminación de la operación (i,j) sin retrasar la terminación del trabajo k.

$$d^k_{ij} = \begin{cases} \max \ (C_k, d_k) - L((i,j), V_k) + p_{ij} & \text{Si el camino } (i,j), V_k \ \text{ existe} \\ \infty & \text{de lo contrario} \end{cases}$$

$L((i,j),V_k)$ es la longitud de la ruta más larga entre el nodo (i,j) y el nodo V_k., C_k es el tiempo de terminación del trabajo k y es igual a $L(U,V_k)$.

Nota: si no hay conexión entre el nodo (i,j) y el nodo V_k, entonces $d^k_{ij} = \infty$

Se debe resolver el problema de una máquina con la tardanza ponderada total como función objetivo para cada una de las máquinas del conjunto $M–M_0$. Para la secuenciación de cada máquina puede utilizarse alguna regla de despacho tal como la regla ACT (*aparent cost tardiness*) o algoritmos exactos de ramificación y acotación (*Branch and Bound*). La regla ACT se caracteriza por escoger en el instante t el trabajo cuyo índice de prioridad $I_{ij}(t)$ sea mayor. Dicho índice se define como:

$$I_{ij}(t) = \sum_{k=1}^{n} \frac{w_k}{p_{ij}} \exp\left(- \frac{\left(dij^k - p_{ij} + \left(r_{ij} - t\right)\right)^+}{K\,\bar{p}} \right)$$

Donde:

r_{ij} es la fecha de disponibilidad de la operación i,j y corresponde a la distancia más larga desde el nodo inicial hasta el nodo (i,j) y t=min (r_{ij}).

K es el valor de escalamiento que depende de la naturaleza del problema.

$\bar{p}$ es el promedio de los tiempos de proceso de los trabajos remanentes en la respectiva máquina.

La constante K se determina experimentalmente.

Seleccione la máquina i^*, cuya secuenciación resulte en el *mayor* valor de la función objetivo, y agréguela a M_0.

Inserte los arcos disyuntivos correspondientes de la secuencia hallada de la máquina i^* en G.

Paso 3 (Resecuenciación de máquinas).

Para cada una de las máquinas i de $M_0–\{i^*\}$, elimine sus arcos disyuntivos en G y formule el problema de una sola máquina. Encuentre de nuevo la secuencia que minimice la tardanza ponderada total e inserte los arcos disyuntivos correspondientes en G.

Paso 4. Si $M_0=M$, termine; de lo contrario, vuelva al Paso 2.

2.5 Modificaciones realizadas al algoritmo del cuello de botella móvil

2.5.1 Escogencia de la máquina cuello de botella móvil

Un aspecto importante en el desempeño del algoritmo del CBM es la escogencia de la máquina crítica en cada iteración. En esta investigación se usaron dos criterios para esta asignación: el primer criterio fue im-

plementado por Pinedo y Singer (1999) y el segundo fue desarrollado por Pinedo y Chao (1999). A continuación se describen los dos criterios:

1. Función objetivo: después de secuenciar cada una de las máquinas disponibles se elige como máquina crítica aquella que cause un mayor aumento en la función objetivo (tardanza ponderada total) (Pinedo y Singer, 1999); si ocurre un empate entre dos o más máquinas se escoge la que ocasione un mayor retardo en el tiempo total (*makespan*). Este caso fue el descrito en la sección 2.4.
2. Criticidad: suponga que en una iteración del CBM se seleccionan los arcos disyuntivos de la máquina *i*. Con esta selección se puede calcular en el grafo disyuntivo resultante el nuevo tiempo de finalización del *k*-ésimo trabajo que se denota aquí como C'''_k. Si se define C'_k como el tiempo de terminación del *k*-ésimo trabajo antes de la secuenciación de la máquina *i* se tiene que $C''_k \geq C'_k$. Esto porque al agregar arcos disyuntivos se imponen nuevas restricciones de recurso que podrían retrasar la terminación de los trabajos. La contribución del trabajo *k* en la "criticidad" de la máquina *i* se puede calcular de la siguiente forma: si $C''_k \geq d_k$ hay un aumento en la función objetivo y la contribución del trabajo *k* para la medida la criticidad de la máquina *i* se toma como $w_k (C''_k - C'_k)$ (Pinedo y Singer, 1999). Sin embargo, si $C''_k \leq d_k$, la penalización causada por la terminación del trabajo *k* es más difícil de estimar (no hay aumento en la función objetivo). Esta penalización debe ser una función que involucre C''_k, C'_k, y d_k. Una de las funciones sugeridas (Pinedo y Singer, 1999) y que arrojó resultados positivos fue:

$$w_k \left(C_k'' - C_k'\right) \exp\left(-\frac{\max\left(0, d_k - C_k''\right)}{K}\right)$$

donde *K* es el parámetro de escalamiento. Cuando se suma sobre todos los trabajos se tiene:

$$\sum_{k=1}^{n} w_k \left(C_k'' - C_k'\right) \exp\left(-\frac{\max\left(0, d_k - C_k''\right)}{K}\right)$$

La máquina que arroje una mayor de medida de criticidad se inserta en el conjunto M_0. En la presente investigación las máquinas se secuenciaron utilizando ambos criterios, a diferencia de Pinedo y Singer (1999), quienes sólo consideraron el criterio de la función objetivo.

2.5.2 Restricciones de precedencia retrasadas (*delayed precedence constraints*, DPC)

La secuenciación de las operaciones en una máquina dada puede resultar en ciclos en el grafo disyuntivo, los cuales a su vez resultan en programaciones no factibles. Para ello se ha formulado un tipo especial de restricciones de precedencia que evitan los ciclos en el grafo. A estas restricciones se les ha denominado de restricciones de precedencias

retrasadas (DPC, por su sigla en inglés) y deben ser calculadas en cada iteración del CBM.

Las restricciones de precedencia pueden ser impuestas por la secuencia de las operaciones en la máquina que ya ha sido programada en iteraciones anteriores. Si dos operaciones están sujetas a una restricción de precedencia —una operación (i,j) debe preceder a otra (i,k)—, no sólo se deben procesar en un orden dado, sino que también durante el proceso de las dos operaciones tiene que transcurrir cierta cantidad de tiempo mínimo (retraso). Se sugiere consultar a Pinedo y Chao (1999) para ejemplos y discusión sobre el tema.

En este enfoque se propone un método para obtener estas DPC. Después de obtener los índices de prioridad con la regla ACT para una máquina, se verifica si existe conexión entre los nodos de los trabajos que se van a secuenciar en la máquina dada. Se calcula la distancia entre los nodos, esto es, la longitud de las DPC y se empieza a secuenciar según las prioridades de la regla ACT. Si existe una DPC que obligue a un trabajo de alta prioridad a suceder a otro de menor prioridad en la secuencia de la máquina, se secuencia primero el de menor prioridad, siempre y cuando la máquina en cuestión esté libre (que el último trabajo de la secuencia parcial tenga una fecha de terminación menor o igual a la fecha de disponibilidad del trabajo de menor prioridad); de lo contrario, se secuencia el trabajo de mayor prioridad y se calculan sus DPC. Lo que se busca es programar los trabajos de menor prioridad en los espacios vacíos y retrasar en lo mínimo posible los trabajos de mayor prioridad, debido a las DPC.

3. Búsqueda tabú

3.1 Generalidades

La BT es un algoritmo metaheurístico que fue propuesto y desarrollado por Glover (Glover y Laguna, 1997). En la actualidad se aplica en una variedad de problemas que incluyen telecomunicaciones, planeación y programación de la producción, análisis financiero, logística, manufactura flexible, manejo de los desperdicios, etc.

En términos generales, el método de BT es un procedimiento iterativo en el cual se parte de una solución inicial y mediante una serie de movimientos (operaciones que transforman una solución s en otra solución s') se intenta encontrar una solución cercana al óptimo (*near-optimal solution*). Si existe un movimiento que transforme s en s', se dice que s' es vecino de s. El vecindario de una solución s es entonces el conjunto de todos los vecinos que pueden generarse a partir de s'.

Si no se toman en cuenta ciertas precauciones que dependen del método y de la naturaleza del problema, es posible que la búsqueda quede atrapada en ciclos y regrese a soluciones ya visitadas. Para evitar esta situación, el procedimiento cuenta con una lista tabú de longitud l (fija o variable) que almacena los últimos movimientos realizados y

que son prohibidos durante el tiempo en que éstos permanezcan en la lista. Este tiempo de permanencia es justamente la longitud de la lista tabú. Asimismo, los movimientos entran y salen de la lista con disciplina *first in-first out* (FIFO).

La BT procede, en una iteración dada, a generar todo el vecindario de la solución actual y selecciona el vecino con mejor valor de la función objetivo, siempre y cuando dicho vecino no sea generado por un movimiento que esté en la lista tabú. En ocasiones, el criterio de la lista tabú puede omitirse si el vecino generado es el mejor hallado hasta ese momento. Éste se denomina el *criterio de aspiración*, que domina al criterio impuesto por la lista tabú.

En adición a lo anterior, la BT puede tener otros elementos de memoria que sirven para intensificar o diversificar la búsqueda. La lista tabú es un ejemplo de un elemento de memoria corta (o de corto plazo). Existen también elementos de memoria larga (o de largo plazo), cuyo objetivo principal es explorar nuevas regiones del espacio de soluciones. En la sección 4 se describen los elementos de memoria corta y larga usados en este algoritmo.

3.2 Algoritmo

Los pasos en un algoritmo de BT son los siguientes:

Paso 1. Hacer k^*=1 $S_{k=1}$

- Seleccionar una secuencia inicial factible utilizando alguna heurística, en este caso la de CBM.
- Hacer S_0=S_1

Paso 2. Elegir el vecindario (esquemas de vecinos).

- Definir de la lista tabú (su longitud) y el criterio de aspiración.
- Seleccionar una secuencia candidata S_c del vecindario de S_k, de acuerdo con la función objetivo.
- Si el movimiento $S_k \rightarrow S_c$ es tabú, hacer S_{k+1}=S_k e ir al paso 3.
- Si el movimiento $S_k \rightarrow S_c$ es permitido, hacer S_{k+1}=S_c
- Si $G(S_c)<G(S_0)$, ir al paso tres.
- Actualizar la lista tabú y el criterio de aspiración.

Paso 3.

- Incrementar k en 1. Si se cumple con el criterio de finalización, detenerse; de lo contrario, ir al paso 2.

Algunos de los criterios de finalización típicos son:

- No se encuentran soluciones factibles en el vecindario elegido en la siguiente iteración.

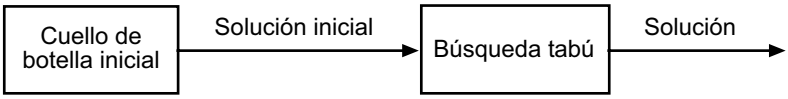
* *k se usa aquí como contador del algoritmo a diferencia de la sección previa donde se usó como índice de trabajos.*

- El número de iteraciones desde el último mejoramiento de la mejor solución es mayor que un número específico.
- El número de iteraciones es mayor que el máximo número de iteraciones permitidas.
- El valor óptimo de la función objetivo fue obtenido (si se conoce).

4. EL ALGORITMO HÍBRIDO CUELLO DE BOTELLA-BÚSQUEDA TABÚ

En esta sección se describe con detalle el algoritmo propuesto. Está compuesto por dos módulos fundamentales: en el primero se implementa la heurística CBM con el objetivo de generar la solución inicial, en el segundo se realiza una búsqueda local basada en la BT y en el tercero se reoptimiza localmente la secuencia de cada máquina cuando una solución mejor es generada por la BT. La integración de los dos módulos se muestra en la Figura 2.

Figura 2. Módulos principales para el algoritmo implementado



Fuente: presentación propia de los autores.

4.1 SOLUCIÓN INICIAL

La solución inicial se halló con el algoritmo del CBM para tardanza ponderada total.

4.2 MOVIMIENTOS

Un vecino se genera a través de un movimiento que consiste en la inversión de un arco disyuntivo en el camino crítico (ruta más larga) de un bloque dado. Un bloque es un camino en la ruta crítica de arcos disyuntivos.

4.3 ELEMENTOS DE MEMORIA CORTA

En la presente aplicación se utilizaron los siguientes elementos de memoria corta: vecindario, lista tabú y criterio de aspiración.

- Vecindario: se obtuvo la ruta crítica para cada trabajo, basada en la información inicial. Con este camino se buscar crear nuevas soluciones mediante la inversión de los arcos que pertenecen a éste. Van Laarhoven, Aarts y Lenstra (1992) demostraron que no se generan ciclos al invertir un arco en el camino crítico desde el nodo inicial al nodo final; por lo tanto, invertir arcos en el camino crítico de cualquier trabajo tampoco generará ciclos. Se destaca que solamente se pueden invertir los arcos disyuntivos.

A continuación se describen los dos vecindarios, E_1 y E_2, que se diseñaron a partir de las consideraciones anteriores:

- ✓ E_1: inversión de los arcos en todas las rutas criticas.
- ✓ E_2: se invierten sólo arcos de los trabajos que aumentan el valor la función objetivo.

- Lista tabú: sirve para almacenar el conjunto de los arcos que se invirtieron. Su propósito es evitar que se caiga en óptimos locales o que se formen ciclos.
- Criterio de aspiración: se utilizaron dos criterios de aspiración:
  - ✓ Aspiración por objetivo: si *F(x)*<mejor costo, entonces se satisface la aspiración del movimiento y esto le permite a *x* ser un candidato.
  - ✓ Aspiración por defecto: si todos los movimientos posibles son tabú, se selecciona el movimiento más antiguo en la lista tabú.

4.4 Tamaño de lista tabú

En este algoritmo se implementó un tamaño de lista tabú dinámica a partir de la idea de Taillard (1994). Ésta sugiere un tamaño de lista tabú que depende de la naturaleza del problema, el tamaño se fija en $(n+m)/2$ (cuando m y n son iguales). Para valores diferentes se propone calcular L como una función de n y m, razón por la cual el tamaño verdadero de la lista es aleatoriamente escogido entre una distribución uniforme entre L_{min}=[,8L] y L_{min}=[1,2L] y se cambia después de cierto número de iteraciones.

A diferencia de la propuesta de Taillard (1994), el tamaño de la lista dinámica que se implementó en este trabajo es escalonado y toma valores entre 8 y 16. Se empieza con 8 y se mantiene ese valor hasta que no exista un mejoramiento en la función objetivo durante quince iteraciones seguidas (que corresponde al 1% de las iteraciones que se utilizaron para verificar el desempeño promedio del algoritmo), después pasa a 16 hasta que se logre un mejoramiento en la función objetivo para luego volver a un tamaño de 8. El tamaño de lista, L, fijado en 8 permite intensificar en la región, y el tamaño 16, que la heurística se diversifique.

4.5 Elementos de memoria larga: diversificación

Después de ejecutar la memoria corta se aplica nuevamente el CBM utilizando las otras reglas de despacho. Se comienza con MDD (*modified due date*), seguido de MODD (*modified operation due date*) y CR+SPT (*critical ratio+shortest processing time*) y ACT con diversos factores de escalamiento. Esto con el fin de que el algoritmo no quede atrapado en regiones planas de soluciones y pueda recorrer otras regiones donde puede encontrarse el óptimo. Las definiciones de las reglas de despacho definidas se muestran a continuación.

Para definir dichas reglas de despachos se utilizará la siguiente notación:

- t: tiempo en que se toma la decisión de programar.
- l: la l-ésima operación del trabajo j.
- o_j: número total de operaciones del trabajo j.
- p_{ij}: tiempo de proceso de la operación i del trabajo j.
- d_j: tiempo de entrega del trabajo j.
- d_{lj}: tiempo de entrega de la l-ésima operación del trabajo j.

Las reglas que se implementaron fueron:

- MDD. El tiempo de entrega modificado para el trabajo j es definido como:

$$d'_j = \max \left(d_j,\ t + \sum_{i=l}^{o_j} p_{ij}\right)$$

- MODD. El tiempo de entrega de la operación modificada se define como:

$$d'_{lj} = \max \left(d_{l-1\,j}, t + \frac{p_{lj}}{\sum_{i=l}^{o_j} p_{ij}}\right),\ d_{0\,j} = 0$$

- CR+SPT. Esta regla selecciona la operación con menor valor de d'_{lj} que es la seudofecha de entrega de la l-ésima operación del trabajo j. Dicha seudofecha de entrega se define como: $d'_{lj} = \max(\beta(t),0)$, donde:

$$\beta((t) = \frac{d_j - t}{\sum_{i=l}^{o_j} p_{ij}},$$

5. Experimentos computacionales

5.1 Problemas solucionados con el algoritmo propuesto

Se consideraron 17 problemas de 10×10 que se encuentran en la OR-Library (http://people.brunel.ac.uk/~mastjjb/jeb/info.html) para medir el desempeño del algoritmo propuesto. Estos problemas fueron elegidos dado que han sido extensivamente estudiados en la literatura sobre el tema y, por lo tanto, se tienen tanto las soluciones óptimas como diversos algoritmos con los cuales establecer parámetros de comparación. Los problemas LA16 a LA20 son instancias clásicas de 10×10. Las instancias LA21-LA24 son originalmente problemas de tamaño 15×10, en los que se eliminaron los últimos cinco trabajos para reducirlos a un tamaño de 10×10. Los problemas ORB1-ORB5 son también instancias de 10×10.

Debido a que estos problemas fueron formulados para minimizar el flujo máximo, se debió agregar una prioridad y una fecha de entrega para cada trabajo, tal y como lo propusieron Pinedo y Singer (1999), y así poder comparar los resultados. En la práctica se observa muchas veces que el 20% de los clientes son muy importantes, el 60% de ellos tiene una importancia promedio y el restante 20% son de menor im-

portancia. Por lo tanto, se establecieron las siguientes ponderaciones para los problemas 10×10:

$$w_1 = w_2 = 4\ w_j = 2 \text{ para } j = 2,3,...8\ w_9 = w_{10} = 1$$

Para determinar los tiempos de entrega de los trabajos se utilizó la regla de trabajo total (TWK, por su sigla en inglés), donde los tiempos de entrega $d_j=\gamma P_j$, donde P_j es la suma de los tiempos de las operaciones del trabajo j y γ es un factor de escala que se tomó como 1,5, tal y como sugieren Baker y Hayya (1982).

5.2 Nomenclatura utilizada en las tablas

- Óptimo. Valor óptimo de la función objetivo, obtenido para problemas de 10×10, utilizado el algoritmo de ramificación y acotación desarrollado Pinedo y Singer (1999).
- PTB: *priority threshold backtracking*. Heurística desarrollada por Pinedo y Singer (1999).
- CBBT. Heurística compuesta implementada en este trabajo, conformada por heurística CBM con la que se obtiene la solución inicial y la BT. Como solución inicial se escoge la solución que se obtiene con el CBM y se utilizó en la BT la lista dinámica explicada en aportes BT se itera durante 60 segundos.
- $CBBT_{ML}$. Es la misma heurística CBBT implementada que incorpora la estrategia de diversificación explicada en un tiempo de iteración máximo de 300 segundos.
- *CB(10,2)*. Heurística CBM implementada por Pinedo y Singer (1999) que tiene dos parámetros, *subproblem backtracking aperture* α=10 y *machine backtracking aperture* β=2.
- *CB(10,3)*. Lo mismo que CB(10,2), con α=10 y β=3.
- BT_1. búsqueda tabú implementada en el *software* LISA®, que toma la solución inicial con la regla WSPT (*weighted shortest processing time*).
- BT_2. Búsqueda tabú implementada en LISA® con las mismas característica de la BT_1, pero su solución inicial es generada con EDD (*earliest due date*).
- EF_1. Enfriamiento simulado implementado en LISA ® con solución inicial generada por la regla WSPT.
- EF_2. Enfriamiento simulado implementado en LISA con las mismas características de EF_1, pero su solución inicial es generada con EDD.

Nota. En las heurísticas BT1-2 y EF1-2 se puede ajustar el tiempo computacional de forma tal que se pueda realizar una comparación justa del desempeño de los algoritmos. En este caso se corrió primero la heurística CBBT y se anotó su tiempo computacional, que fue el límite de tiempo usado en las heurísticas mencionadas.

En la Tabla 2 se muestran los resultados obtenidos para las heurísticas descritas y para el CBBT. En la Tabla 3, la desviación relativa a la solución óptima.

Tabla 2.Valor de la función objetivo (tardanza ponderada total). Factor TWK (γ=1,5)

PG	BT_1	BT_2	EF_1	EF_2	PTB	CB (10, 2)	CB(10,3)	CBBT	$CBBT_{ML}$	Óptimo
MT10	427	564	394	466	1.039	394	394	466	444	394
ABZ5	114	114	316	114	109	109	77	114	109	69
ABZ6	0	0	0	0	0	0	0	0	0	0
LA16	377	168	468	344	168	178	178	280	166	166
LA17	260	260	463	803	353	260	260	284	284	260
LA 18	34	367	188	354	329	83	34	42	42	34
LA 19	199	21	284	428	73	76	21	64	21	21
LA20	201	1	298	465	282	0	0	7	1	0
LA21	16	0	53	16	62	16	0	16	4	0
LA22	441	371	592	485	427	196	196	368	196	196
LA23	2	2	516	288	396	2	2	32	2	2
LA24	94	94	106	88	216	82	82	106	106	82
ORB1	1.662	1.760	2.499	2.577	1.845	1.538	1.196	1.565	1.490	1.098
ORB2	488	391	736	774	672	324	292	442	352	292
ORB3	1.575	2.039	1.640	2.039	1.648	1.073	967	978	978	918
ORB4	723	358	1.089	506	876	358	517	690	396	358
ORB5	620	798	994	763	1.190	524	524	591	572	405

Fuente: presentación propia de los autores.

Tabla 3.Valor del error relativo con respecto al valor óptimo

PG	BT_1 (%)	BT_2 (%)	EF_1 (%)	EF_2 (%)	PTB (%)	CB (10,2) (%)	CB (10,3) (%)	CBBT (%)	$CBBT_{ML}$ (%)	Óptimo
MT10	8,4	43,1	0,0	18,3	163,7	0,0	0,0	18,3	12,7	394
ABZ5	65,2	65,2	358,0	65,2	58,0	58,0	11,6	65,2	58,0	69
ABZ6	*	*	*	*	*	*	*	*	*	0
LA16	127,1	1,2	181,9	107,2	1,2	7,2	7,2	68,7	0,0	166
LA17	0,0	0,0	78,1	208,8	35,8	0,0	0,0	9,2	9,2	260
LA 18	0,0	979,4	452,9	941,2	867,6	144,1	0,0	23,5	23,5	34
LA 19	847,6	0,0	1252,4	1938,1	247,6	261,9	0,0	204,8	0,0	21
LA20	*	*	*	*	*	*	*	*	*	0
LA21	*	*	*	*	*	*	*	*	*	0
LA22	125,0	89,3	202,0	147,4	117,9	0,0	0,0	87,8	0,0	196
LA23	0,0	0,0	25700,0	14300,0	19700,0	0,0	0,0	1500,0	0,0	2
LA24	14,6	14,6	29,3	7,3	163,4	0,0	0,0	29,3	29,3	82
ORB1	51,4	60,3	127,6	134,7	68,0	40,1	8,9	42,5	35,7	1098
ORB2	67,1	33,9	152,1	165,1	130,1	11,0	0,0	51,4	20,5	292
ORB3	71,6	122,1	78,6	122,1	79,5	16,9	5,3	6,5	6,5	918
ORB4	102,0	0,0	204,2	41,3	144,7	0,0	44,4	92,7	10,6	358
ORB5	53,1	97,0	145,4	88,4	193,8	29,4	29,4	45,9	41,2	405
Promedio	109,5	107,6	2068,8	1306,1	1569,4	40,6	7,6	160,4	17,7	

Fuente: presentación propia de los autores.

6. Análisis de resultados

6.1 Calibración de parámetros

Previo al análisis de resultados computacionales se resumen en esta sección los criterios de elección de los parámetros más relevantes.

6.1.1 Criterios de elección del cuello de botella móvil

La decisión de escoger la máquina CBM es relevante, sobre todo en las primeras iteraciones del algoritmo, cuando se han programado pocas máquinas. Los criterios de escogencia *tardanza* y *criticidad* arrojaron los mismos resultados. Para la secuenciación de los problemas de una sola máquina se obtuvieron los mejores resultados con la regla ACT. Los resultados mostrados se obtuvieron usando la regla ACT como regla de secuenciación de los subproblemas de una máquina y el criterio de tardanza.

6.1.2 Búsqueda tabú

En las pruebas se verificó que la solución inicial no es tan importante en la BT como en otras heurísticas de búsqueda local. Sin embargo, se obtuvieron los mejores resultados a partir de la mejor solución inicial del CBM, que ocurre cuando se secuencian las máquinas utilizando la regla ACT.

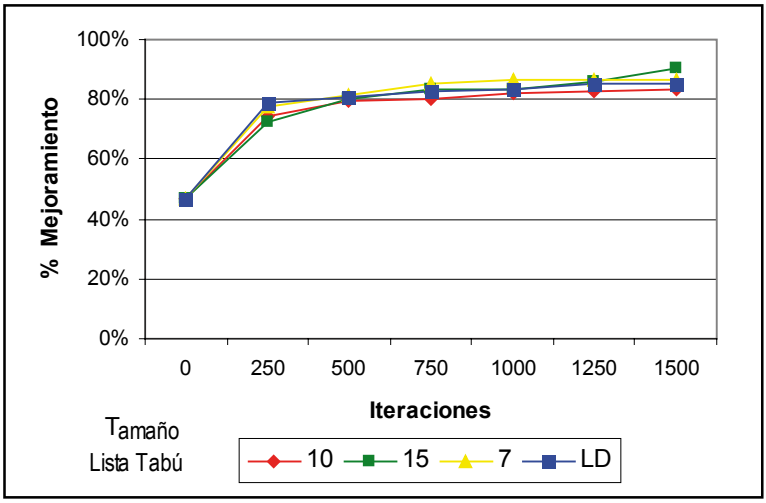
- *Vecindario*. La solución inicial en promedio para las primeras 250 iteraciones obtuvo un mejoramiento del 56% con el vecindario E_1 y de un 54,5% con el vecindario E_2. Al alcanzar las 1.500 iteraciones, el mejoramiento fue del 77% con el vecindario E_1, y del 80% con el vecindario E_2. El inconveniente de utilizar un solo tipo de vecindario es que la solución queda atrapada en pocas iteraciones en óptimos locales cuando se parte de una buena solución inicial. Esto es más evidente con el vecindario E_2, por ser éste más reducido.
- *Tamaño de lista tabú*. Para todas las listas tabú se mejora la solución inicial rápidamente, y en las primeras 250 iteraciones se obtiene más del 60% de mejoramiento. Sin embargo, después de 750 iteraciones se da un mejoramiento muy pequeño. El vecindario E_2, con tamaño de lista 15, es el que mejor desempeño tiene; sin embargo, para el vecindario E_1 la lista tabú de tamaño 10 es la que mejor desempeño arroja con un error relativo cercano al 65% después de 1.500 iteraciones. Le sigue de lista tabú LD de tamaño dinámico con error relativo cercano al 72% en las mismas iteraciones. Para efectos de estudio, en la Tabla 2 se muestran los resultados de CBBT, que varían el tamaño y el tipo de lista (estática y dinámica) contra el desempeño promedio de varias reglas de despacho.

La Figura 3 permite observar que desde la solución inicial la heurística CBBT da mejor resultado que las reglas de despacho (47% mejor, en promedio). Después de 250 iteraciones, CBBT con respecto a las

reglas de despacho obtiene un 75% de mejoramiento promedio para los diferentes tamaños de listas tabú, hasta lograr un mejoramiento del 86% en 1.500 iteraciones.

Como se observa en la Figura 3, el desempeño de la heurística CBBT mejora a una tasa elevada hasta cerca de las 250 iteraciones. Posteriormente también existe un mejoramiento, pero a una velocidad menor. Además, en ambas comparaciones la calidad de las soluciones obtenidas mediante CBBT no se ve afectada notoriamente por la elección del tamaño de la lista tabú.

Figura 3. Mejoramiento promedio de la heurística CBBT con respecto al desempeño promedio de las reglas de despacho (WSPT, EDD Y ATC), vecindario E_1. Criterio de escogencia de la máquina cuello de botella tardanza



Fuente: presentación propia de los autores.

6.2 Indicadores de desempeño comparativo entre las heurísticas

Los problemas fueron corridos en un computador Athlon 900 MHz con 256 MB de memoria RAM. Para medir el desempeño de la heurística y poder comparar con otros enfoques se utilizó el error relativo para cada instancia. El error relativo (ER) se define como:

$$ER = \frac{TWT - óptimo}{óptimo}$$

Donde TWT es el valor obtenido de la tardanza ponderada total por las diferentes heurísticas.

6.3 Comparación de resultados

Se compararon los resultados arrojados por CBBT con los que se obtuvieron por otros métodos de solución, que van desde diversas reglas de despacho —la heurística de Asano y Ohta (2002)— hasta las metaheurísticas BT y enfriamiento simulado.

6.3.1 CBTT frente a PTB, CB, BT y EF

Los resultados anotados en las tablas 1 y 2 muestran que los tiempos computacionales para las instancias de 10×10 fueron de aproximadamente 20 segundos. Puede concluirse que la heurística CBBT presenta resultados muy competitivos tanto en la calidad de la solución como en tiempo computacional. Se puede notar que en las instancias más grandes y en los difíciles problemas ORB es donde el algoritmo encuentra las mayores diferencias con respecto a la solución óptima.

Por ejemplo, la heurística $CBBT_{ML}$ arrojó mejores resultados promedio que las heurísticas BT (BT_1, BT_2) y enfriamiento simulado (EF_1, EF_2) en el mismo tiempo de ejecución en la misma máquina (este es determinado por el usuario) y usando el mismo vecindario.

Igualmente, mejoraron los resultados obtenidos con respecto a la heurística PTB. Con respecto a las heurísticas CB(10,2) y CB(10,3), la heurística $CBBT_{ML}$ arroja resultados en promedio bastante similares. La heurística CB(10,3) obtiene en promedio ligeramente mejores resultados que $CBBT_{ML}$, pero dada la cercanía de los resultados y la gran variación en los resultados obtenidos en cuanto a función objetivo no es concluyente decir que CB(10,3) supera completamente a $CBBT_{ML}$.

Los mejores resultados de la heurística CB, puede intuirse, son debidos a la secuenciación de cada una de las máquinas en la cual se utiliza ramificación y acotación, en comparación con la simple ATC, que utiliza CBBT. Esto es a expensas de mayores tiempos computacionales. La heurística compuesta CBBT presenta un comportamiento robusto, porque permite obtener buenas programaciones en pocas iteraciones (250 es donde se encuentra el 76% del mejoramiento) desde distintas soluciones iniciales.

7. Conclusiones

En este artículo se presentó una heurística denominada CBBT para programar la producción en sistemas de manufactura en ambientes tipo taller. La heurística CBBT combina el algoritmo del CBM con la técnica de la BT. Además de usar el CBM como solución inicial, se proponen una serie de mejoras al algoritmo original, que incluye las restricciones de precedencia retrasadas y la selección por criticidad de la máquina cuello de botella. La BT incorpora elementos de memoria corta (lista tabú escalonada) y de memoria larga.

Se implementó un CBM simple que usa un método original de secuencia de una máquina con *delayed precedence constraints* (DPC), con lo

cual se garantiza obtener soluciones factibles. Además, se utilizaron los criterios tardanza y criticidad de escogencia de la máquina CBM. El criterio de selección de la máquina que se va a secuenciar no afectó significativamente el desempeño del algoritmo.

La reoptimización es una etapa fundamental en el algoritmo de CBM, porque produce un mejoramiento promedio significativo (45,2%) con respecto al algoritmo sin reoptimización. En futuros trabajos es necesario considerar otros enfoques de reoptimización que utilizan *backtracking*, como se describe en Pinedo y Singer (1999), para mejorar la eficiencia de la heurística CBM.

La BT mejoró de forma significativa en la gran mayoría de los casos los resultados de la solución inicial obtenida por el CBM. El tamaño de la lista tabú y la elección del vecindario son los elementos de memoria corta más importantes para el desempeño de la heurística tabú. La lista de tamaño 15 arrojó mejores resultados, seguida de la lista dinámica, que se implementó para ejecutar todos los problemas.

El vecindario que mejores resultados arrojó en promedio fue el vecindario E_2. Sin embargo, para hacer más eficiente la heurística es importante considerar la combinación de los dos vecindarios implementados. Una alternativa es iniciar con el vecindario E_2, que es más reducido, y se cambia al vecindario E_1 cuando no se encuentre una mejor solución en un cierto número de iteraciones.

8. Recomendaciones

En trabajos futuros es importante considerar otros métodos de secuenciar los subproblemas de una máquina en el algoritmo CBM de forma eficiente. En la heurística implementada por Pinedo y Singer (1999) los resultados cambian dependiendo de cuántos árboles de enumeración considere el algoritmo. En los últimos años se han desarrollado heurísticas que se pueden implementar para resolver el problema de minimizar la tardanza ponderada total en una máquina. También es importante aplicar las reglas de dominancia desarrolladas por Akturk y Ozdemir (2001), ya que permiten establecer para dos trabajos adyacentes (i,j) si i debe preceder a j o j debe preceder a i en un tiempo determinado (*breakpoint*).

Se recomienda incorporar el enfoque de reencadenamiento de trayectorias en la implementación de la BT, porque permite lograr una interacción entre las estrategias de intensificación y de diversificación, con el propósito de mejorar el desempeño del algoritmo. La utilización de otros vecindarios también es asunto de investigaciones futuras, así como lo es la generación parcial de vecindarios.

Al igual que lo ocurrido con los algoritmos que se han probado en manufactura tipo taller, se recomienda para futuros estudios adaptar esta heurística compuesta para otros ambientes de manufactura, como manufactura flexible, línea de flujo (*flowshop*), proyectos con recursos restringidos (*resource constrained projects*) y otros.

Otro aspecto sería comprobar el desempeño de la heurística CBBT en problemas reales, como se hizo en el artículo de Mason, Flower y Carlyle (2002), que aplica la heurística del CBM a un ambiente de manufactura tipo taller complejo (*flexible job shop with setups*).

Referencias

ADAMS, J.; BALAS, E. and ZAWACK, D. The Shifting Bottleneck Producer for Job Shop Scheduling. *Management Science*, 1988, (34): 391-401.

AKTURK, M. S. and OZDEMIR, D. A New Dominance Rule to Minimize Total Weighted Tardiness with Unequal Release Dates. *European Journal of Operational Research*, 2001, 135 (2): 394-412.

ANDERSON, E. J. and NYIRENDA, J. C. Two New Rules to Minimize Tardiness in a Job Shop. *International Journal of Production Research*, 1990, 28 (12): 2277-2292.

ARMENTANO, V. A. and SCRICH, C. R. Taboo Search for minimizing total Tardiness in a Job Shop. *International Journal Production Economics*, 2000, (63): 131-140.

ASANO, M. and OHTA, H. A Heuristic for Job Shop Scheduling to minimize total weighted Tardiness. *Computers and Industrial Engineering*, 2002, 42 (11): 137-147.

BAKER, K. R. and HAYYA, J. C. Priority Dispatching with Operation due Dates. *Journal of Operations Management*, 1982 (2): 167-175.

BARNES, J. W. and LAGUNA, M. A Tabu Search Experience in Production Scheduling. Annals of Operations Research, 1993 (41): 141-156.

CARLIER, J. and PISON, E. An Algorithm for Solving the Job-Shop Problem. *Managing Science*, 1989, 35: 164-176.

GLOVER, F. and LAGUNA, M. *Tabú Search*. Amsterdam: Kluwer Academic Publishers, 1997.

GRAHAM, R. L. *et al*. Optimization and Approximation in Deterministic Sequencing and Scheduling: A survey. *Annals of Discrete Mathematics*, 1979 (5): 287-326.

HOLSENBACK, J. E. *et al.* An Improved Heuristic for the Single-Machine, Weighted-Tardiness Problem. *Omega*, 1999, 27 (4): 485-495.

HOLTSCLAW, H. H. and UZSOY, R. Machine Criticidadity Measures and Subproblem Solution Procedures in Shifting Bottleneck Methods: A Computational Study. *Journal of the Operational Research Society*, 1996 (47): 666-677.

MASON, S. J., FLOWER, J. W. and CARLYLE, W. M. A Modified Shifting Bottleneck Heuristic for Minimizing Total Weighted Tardiness in Complex Job Shops. *Journal of Scheduling*, 2002 (5): 247-262.

NOWICKI, E. and SMUTNICKI, C. A Fast Taboo Search Algorithm for the Job Shop Problem. *Management Science*, 1996 (42): 797-813.

PEZZELLA, F. and MERELLI, E. A Taboo Search Method guided by Shifting Bottleneck for the Job Shop Scheduling Problem. *European Journal of Operational Research*, 2000, 120 (2): 297-310.

PINEDO M. and CHAO, X. *Operations Scheduling with Applications in Manufacturing and Service*. Boston: Irwin/McGraw-Hill, 1999.

PINEDO, M. and SINGER, M. A Shifting Bottleneck Heuristic for Minimizing the Total Weighted Tardiness in a Job Shop. *Naval Research Logistics*, 1999, 46 (1): 1-17.

PONNAMBALAM, S. G.; ARAVINDA, P. and SREENIVASA, R. P. Comparative Evaluation of Genetic Algorithms for Job-Shop Scheduling. *Production Planning and Control*, 2001, 12 (6): 560-574.

TAILLARD, É. Parallel Taboo Search Techniques for the Job-Shop Scheduling Problem. *ORSA Journal on Computing*, 1994, 16 (2): 108-117.

VAN LAARHOVEN, O.; AARTS, E. and LENSTRA, J. Job Shop Scheduling by Simulated Annealing. *Operations Research*, 1992 (40): 113-125.

VEPSALAINEN, V. and MORTON, T. Priority Rules for Job Shops Weighted Tardiness Cost. *Management Science*, 1987, 33 (8): 1035-1047.

WANG, T. Y. and WU, K. B. A Revised Simulated Annealing Algorithm for Obtaining the Minimum Total Tardiness in Job Shop Scheduling Problems. *International Journal of Systems Science*, 2000, 31 (4): 537-542.