

Análisis Comparativo de Colonia de Hormigas vs. un Enfoque Combinado Cuello de Botella Móvil/Búsqueda Tabú en la Minimización de la Tardanza Ponderada Total en Sistemas de Manufactura Tipo Taller

Oscar Buitrago Suescun* Rodrigo Britto Agudelo⁺ Gonzalo Mejía Delgadillo[±].

Resumen

Programar la producción en talleres (*job shops*) es un problema que consiste en secuenciar las diferentes operaciones de n trabajos a procesar en m máquinas, con el fin de minimizar alguna función objetivo. Este es un problema NP-Hard en el sentido fuerte. En el presente trabajo se proponen dos alternativas novedosas y prometedoras para solucionar dicho problema con el objetivo de minimizar la tardanza ponderada total. La primera es una implementación de la meta-heurística de colonia de hormigas y la segunda un procedimiento compuesto que utiliza la heurística del Cuello de Botella Móvil combinada con un algoritmo de Búsqueda Tabú con. En el caso de la Colonia de Hormigas se determinan por diseño de experimentos y superficies de respuesta valores recomendables para los diferentes parámetros del algoritmo. En el procedimiento híbrido Cuello de Botella-Búsqueda Tabú se proponen mejoras con respecto a la secuenciación de los sub-problemas y el criterio de selección de la máquina crítica en el algoritmo de cuello de botella. En cuanto a la Búsqueda Tabú se implementó una lista tabú dinámica escalonada que solo toma valores de 8 y 16. Se comparó el desempeño de los algoritmos propuestos mediante 20 problemas de la literatura. Los resultados obtenidos son bastante competitivos en cuanto a calidad de la solución y tiempo computacional.

Palabras Clave: *Meta-heurísticas, programación de taller, Colonia de Hormigas, Búsqueda Tabú, Cuello de botella Movil.*

Abstract

The job shop scheduling problem seeks to sequence the different operations of n jobs to be processed in the m machines minimizing an objective function. It is a NP Strongly Hard problem. In this paper, two novels and promising approaches to solve the JSSP with total weighted tardiness as the objective function are proposed. The first one is an implementation of the meta-heuristic Ant Colony and the second a compound procedure which integrates the Shifting Bottleneck heuristic with the Tabu Search algorithm. Recommendable values for the parameters of the Ant colony, using design experiments and surface responses, were found. Improvements related to the sub-problems sequence and the criteria used for selecting the critical machine in the Shifting Bottleneck were proposed in the hybrid procedure SBTS. A dynamic tabu list which only takes values of 8 and 16 was implemented in the Tabu Search. The performance of the proposed algorithm was compared using 20 instances found in the literature. The obtained results are very competitive regarding the quality of the solution and computational time.

Key words: *Meta-heuristics, job shop scheduling, ant colony, tabu search, shifting bottleneck.*

* Universidad de Boyacá, Cra 2 Este No. 64-169 Tunja, Colombia, oscarbuitrago@uniboyaca.edu.co

⁺ Universidad de Los Andes, Carrera 1 N° 18^a- 10 Bogota, Colombia, ro-britt@uniandes.edu.co

[±] Universidad de Los Andes, Carrera 1 N° 18^a- 10 Bogota, Colombia, gmejia@uniandes.edu.co

1 Introducción

La programación de la producción es un área de gran importancia tanto desde el punto de vista investigativo como del práctico, puesto que permite modelar y buscar soluciones a diversos problemas que son verdaderos retos matemáticos y que a la vez son de interés para las industrias que en un mundo globalizado buscan ventajas competitivas para superar a sus adversarios.

En las empresas manufactureras la forma de programar la producción permite la utilización eficiente de sus recursos, aumentar su productividad y lograr la satisfacción de sus clientes mediante el cumplimiento en los compromisos de entrega de pedidos. Se puede decir que es inconveniente terminar un trabajo antes de tiempo debido a los costos de mantener inventario, así como entregarlo tarde, debido a las sanciones y pérdida de imagen para la compañía [6]. Programar la producción parece un problema sencillo, sin embargo dista mucho de serlo, ha sido abordado por muchos investigadores bajo diferentes enfoques y esta clasificado como NP-Hard según Graham et. al [17].

Por otra parte, un ambiente de trabajo que es común en el estudio de la programación de la producción y que se presenta con frecuencia en nuestro medio es el de tipo taller, este consiste en un conjunto de máquinas en las que se debe procesar un determinado número de trabajos. En esta clase de compañías es más importante cumplir con los tiempos de entrega que un objetivo que tenga que ver con la utilización de las máquinas. Elaborar una programación inadecuada puede dar como resultado entregas retrasadas, pérdida de competitividad e imagen y el riesgo de perder los clientes.

Dentro de los procedimientos de solución al problema de programación de la producción, se encuentran las meta-heurísticas que son técnicas que proporcionan soluciones buenas en tiempos razonables (obtener la óptima resulta prácticamente imposible aún en tamaño de problemas moderados) y son aplicables a diversas situaciones de optimización combinatoria. Las más conocidas son los algoritmos genéticos, el recocido simulado (*Simulating Annealing*), la búsqueda tabú (*Tabú Search*) y a partir de los años 90 la colonia de hormigas (*Ants Colony Optimization*). Esta última se empleo en esta investigación porque es más nueva que las otras, es prometedora, no se encuentran reportadas muchas aplicaciones a problemas de producción y ninguna a la solución del problema descrito. La combinación de cuello de botella y búsqueda tabú se propuso porque los dos procedimientos han arrojado buenos resultados en problemas similares al tratado [18, 26].

Después de adaptar las meta-heurísticas en mención al problema específico, se codificaron en lenguaje C y con el programa obtenido se estudiaron problemas reportados en la literatura especializada para luego realizar las comparaciones respectivas.

2 El Problema de la Programación de la Producción en Talleres (*Job Shop Scheduling Problem*)

Un taller es un ambiente de producción en el que los trabajos a realizar deben ser procesados mediante múltiples operaciones que se realizan en diferentes máquinas. Para cada trabajo, el orden que debe seguir a través de las máquinas para ser procesado se conoce como ruta. Debido a que los talleres están diseñados para permitir la flexibilidad, en ellos usualmente se producen pequeñas

cantidades de una gran variedad de artículos, incluyendo productos que son individualizados para cada cliente.

En un problema de tamaño $m \times n$, hay m máquinas y n trabajos cada uno de los cuales tiene una secuencia de operaciones que indica el orden en que se procesan con un tiempo de proceso p_{ij} , donde el subíndice i denota el trabajo y el j la máquina. Los trabajos tienen un tiempo de inicio r_i antes del cual, ningún procesamiento del trabajo i puede llevarse a cabo y cada máquina tener un tiempo de alistamiento u_j antes del cual, no se puede realizar ninguna operación en ella. También existen un tiempo de entrega para cada trabajo, d_j , y una medida, w_j , de que tan importante es este comparado con los otros.

Para calcular las medidas de desempeño se debe considerar el tiempo en el que se programa el inicio de cada operación, S_{ij} , y el tiempo de terminación de cada trabajo, C_i , con ellos se pueden realizar los cálculos necesarios para obtener la tardanza T_i del trabajo i , la cual se define como $T_i = \max(C_i - d_i, 0)$. La suma ponderada de las tardanzas se conoce como tardanza ponderada total $\sum w_i T_i$.

Además, cuando se realizan las operaciones en las máquinas, cada máquina solo puede procesar una operación al mismo tiempo y solo una operación de cada trabajo puede ser procesada en cada máquina y sin permitir interrupciones. La solución se expresa mediante una programación que es la descripción de los tiempos en que deben ser procesadas cada una de las operaciones de modo que se satisfagan las restricciones.

2.1 Formulación Matemática del Problema

El modelo de programación entera para la minimización de la tardanza ponderada se muestra a continuación [29]:

$$\min \sum_{j=1}^n w_j T_j \quad (1)$$

Sujeto a

$$C_j - S_{ij} \geq p_{ij} \quad \text{para } i = 1 \dots m; j = 1 \dots, n \quad (2)$$

$$T_j - C_j \geq -d_j \quad \text{para } j = 1 \dots n \quad (3)$$

$$T_j \geq 0 \quad \text{para } j = 1 \dots n \quad (4)$$

$$S_{ij} \geq r_j \quad \text{para } i = 1 \dots m; j = 1 \dots, n \quad (5)$$

$$S_{kj} - S_{ij} \geq p_{ij} \quad \text{para todos } (k, j) \text{ que sigue a } (i, j) \quad (6)$$

$$S_{ij} - S_{ij} \geq p_{il} \quad \text{o} \quad S_{ij} - S_{ij} \geq p_{il} \quad \text{para todos } (i, l) \text{ que sigue a } (i, j) \quad \text{para } i = 1 \dots m \quad (7)$$

restricciones (3) y (4) aseguran que el T_j es igual a $(C_j - d_j)^+$. El conjunto de restricciones (5) hace que los tiempos de inicio de todas las operaciones sean más grandes o iguales que los tiempos de disponibilidad de todos los trabajos correspondientes. El conjunto de restricciones (6) contiene la

secuencia de las restricciones conectivas y el conjunto de restricciones (7) contiene las restricciones disyuntivas.

2.2 Número Máximo de Soluciones y Complejidad del Problema

Este cálculo se hace considerando que existen m permutaciones de los n trabajos. Cada permutación proporciona la secuencia de procesamiento de los trabajos en una máquina en particular. Existen $n!$ diferentes permutaciones de los n trabajos y son m permutaciones distintas, por lo tanto el número de máximo de soluciones posibles (si todas son factibles) está dado por $(n!)^m$. Considere el caso de un problema de 10×10 (10 trabajos que se deben procesar en 10 máquinas), el número de posibles soluciones es $3.95 \text{ EXP } 65$ si todas las soluciones fuesen factibles. Este número es $6.56 \text{ EXP } 41$ veces más grande que el número de Avogadro tan común en la química.

El problema ha sido clasificado como NP-Hard [17] es decir no se ha encontrado un algoritmo polinómico para solucionarlo, por lo que el tiempo para encontrar una solución crece exponencialmente con respecto al tamaño del problema, esto implica que para problemas pequeños se pueda encontrar la solución óptima en poco tiempo pero para problemas grandes y aún de tamaño moderado (situaciones reales) el consumo de tiempo computacional sea muy grande.

Estos problemas son por lo general muy difíciles de resolver. Por ejemplo, el problema MT 10 que está compuesto de 10 máquinas y 10 trabajos se publicó en 1963 y su solución óptima se encontró solo en 1988 por Adams et al [1].

2.3 Representación del Problema por Medio de Grafos

Para una mejor visualización del problema en ocasiones es conveniente recurrir a un grafo disyuntivo, $G(N,A,B)$ [27] como el mostrado en la Fig. 1. En este los nodos N representan las operaciones (i,j) que se deben realizar sobre los n trabajos (una operación i,j significa el proceso del trabajo j en la máquina i), los arcos conjuntivos A representan las rutas que debe seguir cada trabajo, si un arco $(i,j) \rightarrow (k,j)$ es parte de A , el trabajo j tiene que ser procesado en la máquina i antes de ser procesado en la máquina k y tiene una longitud $|(i,j) \rightarrow (k,j)| = p_{ij}$. B es el conjunto de arcos disyuntivos, los cuales conectan (por medio de dos arcos con direcciones opuestas) las operaciones que pertenecen a diferentes trabajos pero se realizan en la misma máquina.

Existen dos tipos de nodos especiales; el fuente, U , del cual salen las primeras operaciones en la secuencia de cada uno de los n trabajos, y los nodos terminales, V_i , a los cuales concurren las operaciones finales de los trabajos. Cuando el objetivo es la tardanza ponderada total se debe contar con n nodos terminales V_1, \dots, V_n , uno por cada trabajo. La longitud del camino más largo desde el nodo fuente U hasta el nodo terminal V_i representa el tiempo de terminación del trabajo i .

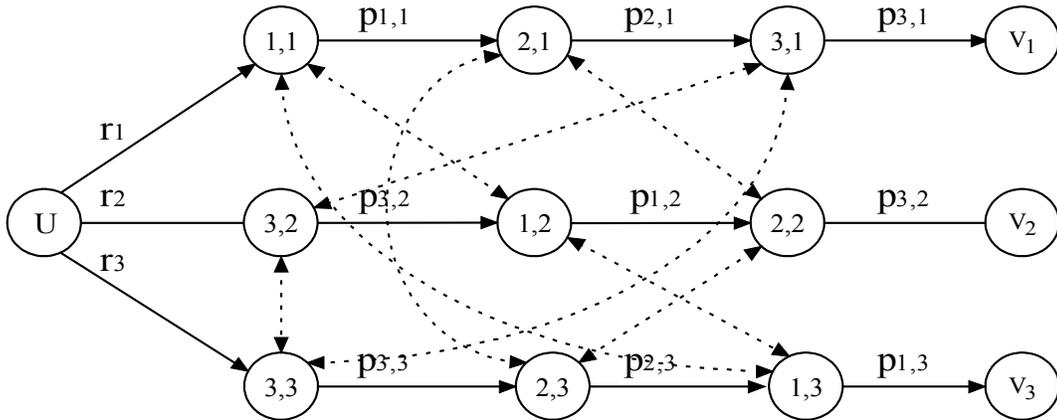


Fig. 1. Representación en grafo de un JSSP minimizando tardanza ponderada total en el caso de tres trabajos y tres máquinas [27]

3 Metaheurística de Colonia de Hormigas (CH)

Esta metaheurística fue propuesta por Dorigo [11] para la solución de problemas combinatorios tales como el del agente viajero (TSP) [12] y el de asignación cuadrática (QAP) [8]. Se basa en la observación de hormigas reales, ya que estos insectos tienen características de comportamiento que han llamado la atención de varios investigadores. Uno de estos aspectos es la capacidad que tiene la colonia para encontrar la ruta más corta entre el hormiguero y las fuentes de alimento.

Mientras hacen sus recorridos las hormigas depositan en la tierra una sustancia química llamada feromona, formando en el camino un rastro de dicha sustancia. Las hormigas pueden percibir la feromona y cuando tienen que escoger una ruta se deciden por aquella que tenga el rastro más fuerte. El rastro de feromonas también les permite encontrar el camino de regreso al hormiguero (ó a la fuente de alimento) y transmitir esta información a otras hormigas.

4 Búsqueda Tabú (BT)

El nombre y la metodología fueron introducidos oficialmente por Fred Glover en 1989 y es una técnica para encontrar soluciones cercanas al óptimo en problemas de optimización combinatoria. Está basada en principios generales de Inteligencia Artificial (IA), "La filosofía de la BT, es la de manejar y explotar una colección de principios para resolver problemas de manera inteligente. Uno de los elementos fundamentales de la BT es el uso de la memoria flexible. Desde el punto de vista de BT, la memoria flexible envuelve el proceso dual de crear y explotar estructuras para tomar ventaja mediante la combinación de actividades de adquisición, evaluación y mejoramiento de la información de manera histórica" [16].

En términos generales el método de BT se describe de la siguiente forma: se basa en un procedimiento iterativo "método de búsqueda local" para encontrar dentro de un conjunto finito X de soluciones factibles una solución s perteneciente a X , la cual minimice una función objetivo f . Para evitar que la búsqueda quede atrapada en ciclos, el procedimiento cuenta con una estructura, lista tabú L de longitud l (fija ó variable) que almacena los últimos movimientos realizados con el

fin de prevenir que se visiten nuevamente soluciones encontradas en las últimas l iteraciones. La diferencia con las técnicas de mejoramiento local radica en el hecho de que se pueden realizar movimientos desde la solución s hacia una peor s^* .

BT se fundamenta en tres puntos principales [21]. Primero está el uso de estructuras de memoria basadas en atributos diseñados para permitir criterios de evaluación e información de búsqueda histórica, luego está un mecanismo asociado de control, mediante el empleo de estructuras de memoria, basado en la interacción entre las condiciones que restringen y liberan al proceso de búsqueda (envuelto en las restricciones tabú y el criterio de aspiración) y el tercero es la incorporación de funciones de memoria de diferentes lapsos de tiempo, desde término corto hasta de término largo, para implantar estrategias que refuercen la combinación de movimientos y las características de solución que históricamente se han encontrado buenas, mientras que las estrategias de diversificación manejan la búsqueda dentro de nuevas regiones.

La memoria a corto plazo almacena por lo general atributos de soluciones recientemente visitadas, y su objetivo es explorar a fondo una región dada del espacio de soluciones. En ocasiones se utilizan estrategias de listas de candidatos para restringir el número de soluciones examinadas en una iteración dada o para mantener un carácter agresivo en la búsqueda.

Es importante considerar que los métodos basados en búsqueda local requieren la exploración de un gran número de soluciones en poco tiempo, por ello es crítico reducir al mínimo el esfuerzo computacional de las operaciones que se realizan a menudo. En ese sentido, la memoria a corto plazo de BT está basada en atributos en lugar de ser explícita; esto es, en lugar de almacenar las soluciones completas (como ocurre en los procedimientos de búsqueda exhaustiva) se almacenan únicamente algunas características de éstas.

Los pasos en un algoritmo de Búsqueda Tabú son los siguientes:

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

Selección de una secuencia inicial factible utilizando alguna heurística

$S_0 = S_1$

Paso 2. Elección del vecindario.

(Esquemas de vecinos)

Definir de la lista tabú (su longitud) y el criterio de aspiración.

Seleccionar una secuencia candidata S_c de la vecindario de S_k de acuerdo a la función objetivo.

Si el movimiento $S_k \rightarrow S_c$ es prohibido o es tabú.

Hacer $S_{k+1} = S_k$ y vaya al paso 3.

Si el movimiento $S_k \rightarrow S_c$ es permitido.

Hacer $S_{k+1} = S_c$

Si $G(S_c) < G(S_0)$

Vaya al paso tres.

Actualizar la lista tabú y el criterio de aspiración.

Paso 3.

$k = k+1$

Si k es igual al criterio de finalización, deténgase, de lo contrario, vaya al paso 2.

5 Algoritmo Cuello de Botella Móvil

Un algoritmo que ha resultado eficiente en la solución de JSSP es el cuello de botella móvil y se describe a continuación [27]:

Sea M el conjunto de las m máquinas y Mo el conjunto de máquinas que ya han sido programadas:

Paso 1.

Hacer $Mo = \emptyset$, siendo G el grafo con todos los arcos conjuntivos, seleccionar la máquina i con mayor valor $\sum w_j T_j$ donde T_j es la tardanza del trabajo j y agregarla a Mo .

Paso 2. (Selección del Cuello de Botella)

Para cada una de las máquinas del conjunto $M-Mo$, formular el problema de minimización.

En la i -ésima iteración: Seleccionar la máquina con mayor valor de $\sum w_j T_j$, donde T_j es la tardanza del trabajo j y agregarla a Mo .

Paso 3.

Seleccionar las operaciones (i,j) de la máquina.

Definir las fechas de entrega d_{ij}^k (fecha de entrega de trabajo k dada la operación (i,j))

$$d_{ij}^k = \begin{cases} \max(C_k, d_k) - L((i,j), V_k) + p_{ij} & \text{si } L((i,j), V_k) \text{ existe,} \\ \infty & \text{de lo contrario.} \end{cases} \quad (8)$$

Si no hay conexión entre el nodo ij y el nodo V_k entonces $d_{ij}^k = \infty$

C_k = el tiempo de terminación del trabajo j es igual $L(U, V_k)$.

Definir r_{ij} como la mayor distancia desde el nodo inicial hasta el nodo ij , donde $t = \min(r_{ij})$

Calcular la prioridad de la operación I_{ij} como:

$$I_{ij}(t) = \sum_{k=1}^n \frac{w_k}{p_{ij}} \exp \left(- \frac{(d_{ij}^k - p_{ij} + (r_{ij} - t))^t}{K \bar{p}} \right) \quad (9)$$

K = valor de escalamiento que depende de la naturaleza del problema.

\bar{p} = parte entera del promedio de los tiempos de proceso de los trabajos en la respectiva máquina.

Insertar los arcos disyuntivos correspondientes en G y agregar la máquina k (la seleccionada en Mo).

Paso 4 (Resecuenciación de máquinas).

Para cada una de las máquinas i que pertenecen a $Mo - \{k\}$, borrar los arcos disyuntivos de G y formular el problema de una sola máquina, encontrar la secuencia que minimice T_j e insertar los arcos disyuntivos correspondientes en G .

Paso 5.

Si $Mo = M$ terminar, de lo contrario volver al paso 2.

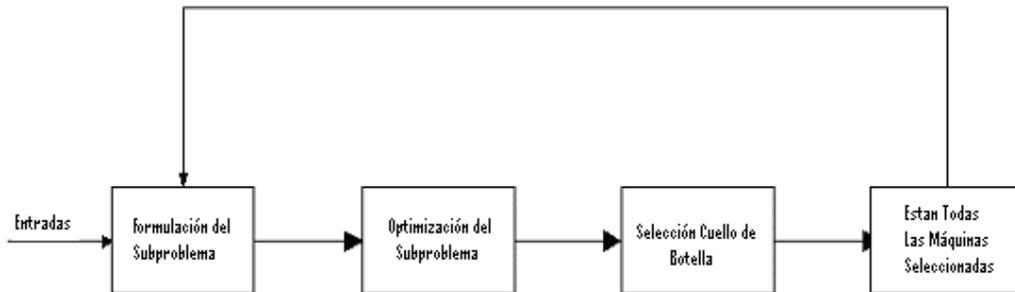


Fig. 2. Diagrama de flujo de la heurística Cuello de Botella Móvil

6 Enfoques Propuestos para la Solución del Problema

Para la solución al problema planteado, en la presente investigación se proponen dos alternativas; un algoritmo de colonia de hormigas y una heurística que combina la búsqueda tabú con el cuello de botella móvil. La elección de la colonia de hormigas se basa en que es un algoritmo de tipo constructivo y del que no se encuentran reportes en cuanto a su aplicación al JSSP con objetivo de minimizar la tardanza ponderada total. Por su parte el enfoque combinado de búsqueda tabú y cuello de botella móvil se fundamenta en que son dos heurísticas que han reportado resultados satisfactorios en problemas de tipo taller y al igual que la colonia de hormigas no se encuentran registros bibliográficos de una aplicación similar cuando el objetivo es minimizar la tardanza ponderada total.

6.1 Algoritmo de Colonia de Hormigas (CH) Propuesto para el Problema de Programación de la Producción en Talleres (Job Shop Scheduling Problem)

Se diseñó un algoritmo en el que la colonia de hormigas sólo construye programaciones activas ya que este tipo de soluciones forma el subconjunto más pequeño que contiene el óptimo.

Debido a la analogía en que se basa CH, puede resultar conveniente visualizar el problema que se está solucionando mediante la representación gráfica ilustrada en la Fig. 1, ya que el algoritmo lo que hace es construir buenas rutas que conecten el nodo inicial con los nodos terminales. Por esto cada una de las operaciones i,j se le asignó un número correspondiente al nodo que la representa, por lo tanto dos operaciones unidas por un arco conjuntivo (continuo) se pueden visualizar como un segmento del camino creado por una hormiga y al igual que en la naturaleza, si muchas hormigas siguen el mismo camino ó escogen el mismo par de operaciones consecutivas incentivarán a las que vienen atrás a escoger el mismo camino.

Inicialmente cada par de operaciones (nodos) es inicializado con una pequeña cantidad de feromonas, es decir se hace $\tau_{ij}(0) = \tau_o$, donde τ_o es un valor inicial pequeño y positivo escogido ó mediante la expresión

$$\tau_o = (1/(n * TPTcs)) \quad (10)$$

con n representando el número de trabajos y $TPTcs$ el valor de la función objetivo de una solución encontrada de cualquier forma.

Con el fin de disminuir el número de búsquedas que realiza una hormiga en cada iteración, se elabora una lista de candidatos que cumplan con ciertas características. Se crea un conjunto Ω en el que se incluyen las operaciones que hacen que la secuencia que esta construyéndose sea activa y se escoge entre ellos de acuerdo con el rastro de feromonas y un factor de visibilidad f_v . Con este factor se pretende que la búsqueda no se realice a ciegas y se base solo en la percepción del rastro de feromona sino que tenga en cuenta las características del candidato.

Para una secuencia parcial de operaciones una hormiga escoge la siguiente (el nodo al que se va a ir) teniendo en cuenta la intensidad del rastro de feromonas $\tau_{ij}(t)$ y el factor de visibilidad.

Para ello se fija el valor de un parámetro q_o , en el intervalo $(0, 1)$, y en cada iteración se genera un número aleatorio q en el mismo intervalo. Si $q_o \geq q$, la siguiente operación de la secuencia se escoge del conjunto Ω considerando la siguiente regla:

$$\arg \max_{l \in Tabu_k} \left\{ [\tau_{il}(t)]^\alpha [1/fv_{il}^1]^\beta [1/fv_{il}^2]^\varphi \right\} \quad (11)$$

El subíndice l indican que la operación pertenece a Ω y α , β y φ representan los exponentes del rastro de feromonas y de los factores de visibilidad. Según las recomendaciones de Bauer et al [5] y Gagne et al [14] estos exponentes toman valores entre 1 y 4.

Si $q_o < q$, entonces la siguiente operación en la secuencia se elige aleatoriamente del conjunto Ω .

Para evitar que algunas operaciones (nodos) se hagan demasiado atractivas y conlleven a caer en un óptimo local de forma prematura, se realiza una "evaporación" de feromonas mediante una actualización local del rastro de feromonas. Esta realiza una disminución del nivel de feromonas para un par de operaciones adyacentes τ_{ij} , de esta manera se desanima a la siguiente hormiga elegir el mismo par de nodos consecutivos durante la iteración.

Se evalúan las m secuencias halladas durante la iteración y si el valor de la función objetivo en alguna de ellas es mejor que el que se tiene hasta el momento, entonces se realiza la actualización del mejor valor encontrado para la función objetivo $TPT+$ y para la secuencia correspondiente $S+$. Después de evaluadas todas las soluciones, se hace una actualización global de la intensidad del rastro de feromonas. Al contrario de la actualización local, en este paso se incrementa el nivel de feromonas para cada par de operaciones ó nodos pertenecientes a la mejor secuencia de las m construidas durante la iteración. La actualización se realiza de acuerdo a la calidad de dicha secuencia, para cada par de nodos adyacentes i,j que pertenezcan a esta se incrementa el valor de τ_{ij} .

$$\tau_{ij}(t+1) = \rho_g \tau_{ij}(t) + (1-\rho_g) \Delta \tau_{ij}(t) \quad \text{donde } \Delta \tau_{ij}(t) = 1/TPT^* \quad (12)$$

La magnitud del incremento del rastro de feromonas está dado por ρ_g , este parámetro toma valores en el intervalo $[0,1]$. TPT^* es el valor de la función objetivo correspondiente a la mejor secuencia de la iteración, dicha secuencia se representa por S^* .

El factor de visibilidad que se utilizó en el presente trabajo está definido como:

$$fv_{ij} = r_j + p_{i,j} - d_j \quad (13)$$

Se pretende dar prioridad a las operaciones que estén disponibles más rápido para ser programadas y que deben ser entregadas con mayor premura.

El presente es el esquema del algoritmo (adaptación y mejora del propuesto por Gagne [14] para el problema de una sola máquina).

Inicialización del rastro de feromonas.

Para cada par de operaciones (i,j)

Iniciar $\tau_{ij}(0) = \tau_o$

Fin

Sea S^+ la mejor secuencia hallada hasta el momento y TPT^+ el valor de la función objetivo.

Ciclo principal.

Para $t = 1$ hasta $t =$ ultima iteración

Asignación de la operación inicial.

Para $k = 1$ hasta $k = m$, haga

Asignar la operación inicial para la hormiga k y almacenarla en la lista $Tabu_k$

Fin

Construcción de secuencias por cada hormiga.

Sea S^* la mejor secuencia encontrada en la iteración actual y TPT^* su valor.

Para $i = 1$ a $m \times n$

Para $k = 1$ a m

Crear el conjunto Ω

Escoger la siguiente operación no perteneciente a $Tabu_k$)

si $q \leq q_o$, se escoge entre las operaciones del conjunto Ω de acuerdo con

$$j = \arg \max_{k \in Tabu_k} \left\{ [\tau_{ij}(t)]^\alpha \left[1 / fv_{ij}^1 \right]^\beta \left[1 / fv_{ij}^2 \right]^\varphi \right\} \quad (14)$$

si $q > q_o$ se escoge aleatoriamente una operación del conjunto Ω

Almacenar esta información en $Tabu$ y actualizar Ω

Realizar la actualización local del nivel del rastro de feromonas para el

par (i,j) de operaciones escogidas:

$$\tau_{ij}(t) = \rho_i \tau_{ij}(t) + (1 - \rho_i) \Delta \tau_{ij} \quad \text{donde } \Delta \tau_{ij} = \tau_o$$

Fin

Fin

Evaluación de soluciones.

Para $k = 1$ a m

Calcular $TPT_k(t)$ para la secuencia $S_k(t)$ construida por la hormiga k .

Si se encuentra una solución mejor que la actual entonces actualizar S^+ y TPT^+

Fin.

Actualización global del rastro de feromonas.

Para cada par de operaciones adyacentes (i,j) que pertenecen a S^*

Actualizar el rastro de feromonas de acuerdo a:

$$\tau_{i,j}(t+1) = \rho_g \tau_{i,j}(t) + (1 - \rho_g) \Delta \tau_{i,j}(t) \text{ donde } \Delta \tau_{i,j}(t) = 1 / TPT^*$$

Fin.

Limpiar todas las listas Tabú_k

Fin

6.2 Propuesta de una Heurística Compuesta: Búsqueda Tabú con Cuello de Botella Móvil (BTCBM)

El algoritmo está compuesto por tres módulos fundamentales, en el primero se implementa la heurística cuello de botella móvil con el objetivo de generar la solución inicial, el segundo realiza una búsqueda local basado en la búsqueda tabú y el tercero re-optimiza localmente la secuencia de cada máquina cuando una solución mejor es generada por la búsqueda tabú. La integración de los tres módulos y el diagrama de flujo del algoritmo descrito son mostrados en la Fig. 3.

Fig. 3. Módulos principales para el algoritmo implementado

A continuación se describen las mejoras propuestas al cuello de botella móvil y a la búsqueda tabú.

6.2.1 Mejoras Propuestas al Algoritmo Cuello de Botella

❖ Secuenciación de los sub-problemas en Cuello de Botella

Para secuenciar los sub-problemas de una sola máquina se propusieron reglas de despacho que habían mostrado buenos resultados en el problema de tardanza total en una máquina. Para definir las reglas de despachos se utilizará la siguiente notación:

- t tiempo en que se toma la decisión de programar
- k la k ésima operación del trabajo j que esta esperando para ser procesada
- g_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_{kj} tiempo de entrega de la k ésima operación del trabajo j

Las reglas que se implementaron fueron:

- *Modified Operation Due Date (MOD)*: El tiempo de entrega de la operación modificada se define como:

$$d'_{kj} = \max \left\{ d_{kj}, t + \sum_{i=k}^{g_j} p_{ij} \right\} \quad (15)$$

Para establecer los tiempos de entrega de las operaciones se utiliza la regla de trabajo total (TWK, por sus iniciales en inglés):

$$d_{kj} = d_{k-1,j} + d_j \cdot p_{kj} / \sum_{i=1}^{g_j} p_{ij}; \quad d_{0j} = 0 \quad (16)$$

- *Critical Ratio + Shortest Processing Time (CR+SPT)*: combinación de las reglas Tiempo de Proceso más Corto (SPT, por sus iniciales en inglés) con la Relación Crítica (CR, por sus iniciales en inglés), el tiempo de entrega de la k-ésima operación del trabajo j se define así:

$$d_{kj} = \max\{t + \beta(t) \cdot p_{kj}, t + p_{kj}\} \quad (17) \text{ donde}$$

$$\beta(t) = a_j(t) / \sum_{i=k}^{g_j} p_{ij} \quad y \quad a_j(t) = d_j - t \quad (18)$$

En Armentano y Scrich [3], se aplicaron las anteriores reglas de despacho como solución inicial a la heurística Búsqueda Tabú para minimizar la tardanza total en ambientes de manufactura tipo taller; la mejor solución se emplea como punto de partida para aplicar la heurística Búsqueda Tabú.

❖ Selección de la máquina cuello de botella

Para la selección de la máquina crítica ó cuello de botella se aplico el criterio de tardanza implementado por Pinedo y Singer [28]. Este consiste en que después de secuenciar cada una de las máquinas se calcula el aumento de la tardanza ponderada total que se obtiene al programarlas; la máquina que más aumente la tardanza ponderada total es la que se elige para programar; si ocurre un empate entre las máquinas se inclina por la que ocasione un mayor retardo.

6.2.2 Descripción y Aportes a la Búsqueda Tabú aplicada

❖ Elementos memoria corta

Los elementos de memoria corta aplicados al problema son los siguientes:

- **Movimiento:** inversión de un solo arco disyuntivo en el camino crítico de un trabajo dado.
- **Vecindario:** Un buen esquema de vecinos se obtiene utilizando la información inicial para obtener la ruta crítica para cada trabajo. Con la información del camino crítico se pretende encontrar nuevas soluciones invirtiendo los arcos que pertenecen a los caminos críticos de los trabajos. Van Laarhoven et al. [31] demostraron que la inversión de un arco en el camino crítico desde el nodo inicial al nodo final no genera ciclos en el grafo; consecuentemente, una inversión en el camino crítico de cualquier trabajo tampoco genera ciclos. Se debe recalcar que solamente los arcos disyuntivos pueden ser invertidos. Se establecieron dos vecindarios E_1 y E_2 para solucionar el problema propuesto:

E_1 : todas las soluciones obtenidas a través de la inversión de los arcos en todas las rutas críticas.

E_2 : invierte solo los arcos de los trabajos que presentan tardanza.

- **Lista tabú:** Almacena un conjunto de arcos que fueron invertidos para evitar caer en ciclos o mínimos locales. Por ejemplo, si el arco (2,5) es invertido a (5,2), entonces (2,5) se almacena en la lista tabú durante un cierto número de iteraciones.

- Criterio de aspiración: Se utilizan dos criterios de aspiración por objetivo y por defecto:
Aspiración por Objetivo: Una aspiración de movimiento se satisface, permitiendo que un movimiento x sea un candidato para seleccionarse si $F(x) <$ mejor costo.
Aspiración por defecto: Si todos los movimientos posibles son clasificados como tabú, entonces se selecciona el movimiento más antiguo en la lista tabú.

❖ Tamaño de lista tabú

Se implementó un tamaño de lista tabú dinámica siguiendo la idea de Taillard [32] que sugiere un tamaño de lista tabú que depende de la naturaleza del problema, para problemas “difíciles” con $n = m$ el tamaño es igual a $(n+m)/2$. Consecuentemente propone calcular L como una función de n y m . Por lo tanto, el tamaño verdadero de la lista es aleatorio uniformemente escogido entre $L_{\min} = \lfloor .8L \rfloor$ y $L_{\max} = \lfloor 1.2L \rfloor$ y se cambia después de cierto número de iteraciones ligeramente más grande que el tamaño máximo. A diferencia de la propuesta de [32], el tamaño de la lista dinámica implementada es escalonado y toma valores entre 8 y 16. Comienza con 8 y se mantiene con ese valor hasta que no exista un mejoramiento en la función objetivo durante 15 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 objetiva para volver a 8. El tamaño de lista 8 permite intensificar en la región y el tamaño 16 permite que la heurística recorra otras regiones.

7 Metodología

Inicialmente se estudiaron las características del JSSP, de la colonia de hormigas, la búsqueda tabú y el cuello de botella móvil, basándose en los resultados de una amplia revisión bibliográfica. Posteriormente se diseñaron los dos algoritmos tendientes a solucionar el problema, colonia de hormigas, **CH**, y búsqueda tabú combinada con cuello de botella móvil, **BTCBM**, los cuales ya han sido ampliamente descritos.

Para ajustar el valor de los parámetros de CH se realizó un diseño simplex de experimentos, considerando los siguientes factores; exponente del rastro de feromona (α), exponente del factor de visibilidad (β), el parámetro de evaporación de feromona (ρ_i), parámetro de incremento de feromona (ρ_g) y el número aleatorio con que se decide que regla aplicar para la selección de las operaciones (q_o).

Adicionalmente, se emplearon las meta-heurísticas Recocido Simulado, **R.S**, y Búsqueda Tabú, **B.T**, de las que dispone el software LISA.

Para medir la eficiencia de los algoritmos diseñados y realizar las correspondientes comparaciones, se buscaron en la literatura problemas a los que se les conoce la solución óptima. Se seleccionaron ocho de tamaño 10x10 que se encuentran en la OR-Library (<http://mscmga.ms.ic.ac.uk/info.html>); ABZ5 y ABZ6 de Adams, Balas y Zawack, LA 16 – LA 20 de Lawrence y el MT10 de Ruth y Thompson. En la determinación de los tiempos de entrega, d_j , de los trabajos se utilizó la regla de trabajo total (TWK) en según la cual $d_j = K P_j$, siendo P_j la suma de los tiempos de las operaciones del trabajo j y K un parámetro. Por cada uno de los problemas considerados se obtuvieron dos versiones, una asignando los tiempos de entrega con $K = 1.3$ y otra con $K = 1.5$.

Todos los problemas se solucionaron con las metaheurísticas búsqueda tabú y recocido simulado disponibles en el software LISA, luego se corrieron en el algoritmo de colonia de hormigas codificado en lenguaje C y en el algoritmo BTCBM también codificado en C. Se utilizó un computador Athlon 900 MHz con 256 Megas de memoria RAM.

Para medir el desempeño de la heurísticas y poder compararlos con otros enfoques se calculó un indicador al que se le llamo **relación de calidad**, el cual está dado por la relación entre la solución alcanzada y el valor óptimo. Por lo tanto, si este indicador vale uno quiere decir que el procedimiento evaluado logró encontrar la solución óptima del problema. Los resultados obtenidos se tabularon para su posterior comparación y análisis.

$$\text{Relación de calidad} = \frac{\text{valor solución alcanzada}}{\text{valor solución óptima}} \quad (19)$$

8 Análisis y Discusión de Resultados

8.1 Diseño de Experimentos

En principio se optó por aplicar un diseño 2^k factorial con el fin de ajustar una superficie de respuesta lineal, explorar hasta encontrar la región del óptimo y luego aplicar un diseño de orden superior con el fin de determinar el valor de los parámetros del algoritmo. Se llegó al siguiente ajuste:

$$TPT = 3191 + 119\alpha - 161\beta + 1086\rho_l - 956\rho_g + 989q_o \quad (20)$$

Donde TPT es el valor de la tardanza ponderada total.

Al realizar el análisis de varianza se obtuvo un valor P de 0.894, indicando que el ajuste lineal no es adecuado y como consecuencia el método del descenso más pronunciado que se pretendió aplicar no es factible.

Una alternativa apropiada es aplicar el diseño simplex ya que permite acercarse al óptimo de manera eficiente. Se procedió tomando como valor del paso $\Delta q_o = 0.5$, obteniendo los siguientes valores para los parámetros del algoritmo de colonia de hormigas. Con estos se trabajó el algoritmo finalmente implementado.

Tabla 1. Valores de los parámetros obtenidos con el diseño simples

α	β	ρ_l	ρ_g	q_o
1.50	1.50	0.425	0.425	0.50

8.2 Resultados Obtenidos con Colonia de Hormigas (CH), Búsqueda Tabú/Cuello de Botella Móvil (BTCBM), Búsqueda Tabú (BT) y Recocido Simulado (RS)

Para la obtención de resultados se utilizaron los algoritmos diseñados en el siguiente trabajo, **CH** y **BTCBM**¹, así como la meta-heurísticas Búsqueda Tabú y Recocido Simulado disponibles en el software LISA, en los dos últimos casos se crearon 5.000 soluciones y el tiempo promedio de ejecución es de diez segundos.

En cuanto a colonia de hormigas, las corridas se hicieron con 15 iteraciones y 10 hormigas lo que da 150 soluciones creadas, el tiempo promedio de cada procedimiento es de 130 segundos. Para iniciar el procedimiento se empleo un valor de tao inicial igual a 0.001.

Una forma objetiva de analizar los resultados obtenidos es calcular y comparar su distancia relación con respecto al valor óptimo, Para ello se empleo el indicador dado por la ecuación (19), la relación de calidad para las soluciones alcanzadas.

Tabla 2. Relación de calidad (valor solución / valor óptimo) en problemas con factor $K = 1.3$.

PROBLEMA	Óptimo	C.H	B.T	R.S	BTCBM ₁	BTCBM ₂
ABZ 5	1405	3.87	1.76	1.76	1.87	1.12
ABZ 6	436	2.00	1.90	2.20	1.50	1.24
MT 10	1368	6.39	3.11	5.56	1.53	1.31
LA 16	1170	5.05	1.52	2.95	1.33	1.13
LA 17	900	6.40	3.28	5.42	1.44	1.44
LA 18	929	5.83	3.02	6.98	1.06	1.01
LA 19	949	3.38	3.52	4.32	1.17	1.32
LA 20	809	4.40	6.41	4.40	1.50	1.18
Promedio	-----	4.7	3.1	4.2	1.42	1.22

Con ninguno de los procedimientos se lograron soluciones óptimas, sin embargo las dos versiones de búsqueda tabú/cuello de botella móvil reportan resultados que son notoriamente superiores a los de los otros procedimientos. Además, aplicando BTCBM₂ se obtiene la mejor solución en el 87.5 % de los casos y la versión BTCBM₁ obtiene la mejor solución en el porcentaje restante. Es decir que el procedimiento híbrido nunca fue superado por los otros enfoques.

El procedimiento de recocido simulado muestra un mejor desempeño promedio que la colonia de hormigas, sin embargo en la mitad de los problemas ésta obtiene mejores resultados que el recocido. No obstante el mejor comportamiento entre las meta-heurísticas puras lo reporta la búsqueda tabú.

Los resultados obtenidos con el procedimiento BTCBM₂ son de alta calidad, ya que aunque no alcanzó la solución óptima en algún problema, los valores de la relación de calidad en ningún caso son mayores a 1.44.

¹ C.H es Colonia de Hormigas, B.T Búsqueda Tabú, R.S Recocido Simulado. BTCBM₁ Búsqueda Tabú con Cuello de Botella Móvil secuenciado con la regla MOD y . BTCBM₂ Búsqueda Tabú con Cuello de Botella Móvil secuenciado con la regla CR + SPT

Tabla 3. Relación de calidad (valor solución / valor óptimo) en problemas con factor $K = 1.5$

PROBLEMA	Óptimo	C.H	B. T	R. S	BTCBM ₁	BTCBM ₂
ABZ 5	69	34.8	2.78	42.2	1.82	1.58
ABZ 6	0	4192 / 0 ²	317 / 0	2878 / 0	1*	1 *
MT 10	394	5.3	5.3	7.1	2.04	1.74
LA 16	166	27.5	10.7	20.8	1.56	1.23
LA 17	260	13.3	6.46	9.1	1.51	1.09
LA 18	34	98.4	77.8	38.5	2.18	2.44
LA 19	76	134	58.1	261	1	1.81
LA 20	0	4141 / 0	3354 / 0	4885 / 0	14 / 0	10 / 0

En el conjunto de problemas con factor $K = 1.5$, la heurística búsqueda tabú/cuello de botella móvil secuenciada mediante la regla CR + SPT, sigue arrojando los mejores resultados. El 62.5 % de los casos es la mejor solución, en el 12.5% de ocasiones es igualada por BTCBM₁ y en el 25% es superada por esta última. En esta oportunidad se alcanzó la solución óptima en tres de los problemas. Comparando con los otros procedimientos, nuevamente los híbridos dan resultados ampliamente mejores.

Es bastante notorio que para los procedimientos **BTCBM₁** y **BTCBM₂** la calidad de los resultados no decrece de forma significativa (el factor de calidad promedio es 1.68 para el primero y 1.65 para el segundo), mientras que para **C.H**, **R.S** y **B.T** es evidente que la calidad de las soluciones decreció (el valor del factor de calidad aumento drásticamente) con respecto a los problemas generados con parámetro $k = 1.3$. Por lo anterior se puede decir que el comportamiento de la heurísticas híbridas propuestas es más robusto que el de colonia de hormigas, recocido simulado y búsqueda tabú, ya que estas últimas ven afectado su desempeño con la forma de asignar los tiempos de entrega de los trabajos, en tanto que para **BTCBM₁** y **BTCBM₂** este efecto es apenas notorio.

Entre las metaheurísticas, Búsqueda Tabú consigue los mejores resultados, mientras que la Colonia de Hormigas y el Recocido Simulado no muestran diferencias significativas en cuanto a la proporción de veces que uno supera al otro. Sin embargo la calidad de las respuestas logradas en ninguno de los tres casos es comparable con los procedimientos híbridos.

En cuanto a Colonia de Hormigas se puede decir que aunque solo se generan 150 soluciones, su comportamiento es competitivo con respecto a Recocido Simulado y Búsqueda Tabú aún cuando con estos se están generando 5.000 soluciones.

² División por cero (valor óptimo en ese caso). C.H es Colonia de Hormigas, B.T Búsqueda Tabú, R.S Recocido Simulado. BTCBM₁ Búsqueda Tabú con Cuello de Botella Móvil secuenciado con la regla MOD y . BTCBM₂ Búsqueda Tabú con Cuello de Botella Móvil secuenciado con la regla CR + SPT

* Simboliza que se alcanzó el óptimo y este vale cero.

9 Conclusiones e Investigaciones Futuras

De los procedimientos probados, el mejor desempeño se alcanza con la heurística compuesta **BTCBM₂**, aunque el desempeño de **BTCBM₁** es bastante competitivo.

Los resultados obtenidos con los procedimientos híbridos son de alta calidad ya que siempre están cerca de la solución óptima y en ocasiones la alcanzan.

Los algoritmos probados de recocido simulado, búsqueda tabú y colonia de hormigas, no resultan competitivos al compararlos con las propuestas híbridas de búsqueda tabú/cuello de botella móvil.

Al aumentar el valor del factor de escala de las fechas de entrega K empleado para la generación de los problemas, se vuelve más difícil encontrar la solución óptima para las metaheurísticas Búsqueda Tabú, Recocido Simulado y Colonia de Hormigas. Esto se evidencia al calcular el factor de calidad, (a menos valor mejor solución) el cual es mayor para los problemas generados con $k = 1.5$ que para los generados con $K = 1.3$. Los procedimientos **BTCBM₁** y **BTCBM₂** no se ven afectados por el incremento del valor K .

Los resultados para la Colonia de Hormigas son alentadores en el sentido que con solo 150 soluciones generadas es competitiva frente a la Búsqueda Tabú y superior a Recocido Simulado, aún cuando estas generan 5.000 soluciones. Máxime si se tiene en cuenta el algoritmo diseñado C.H no incluye pasos remejoramiento local para las soluciones construidas, es decir que estas son fruto exclusivo de la cooperación entre los agentes (hormigas).

El algoritmo de Colonia de Hormigas considerado no incluye pasos de mejoramiento local para las soluciones obtenidas. Se propone para investigaciones futuras incluir estas etapas con el fin de incrementar la calidad de los resultados. Este aspecto es especialmente importante debido a que la colonia de hormigas al ser un procedimiento lento, debe proporcionar muy buenas soluciones en pocas iteraciones.

Referencias

- [1] J. Adams; E. Balas y D. Zawack. The shifting bottleneck producer for job shop scheduling, *Management Science* 34 391 – 401. 1988.
- [2] D. Applegate and W. Cook. A Computational Study of the Job-Shop Scheduling Problem, *ORSA Journal on Computing*, Spring, 3(2), 149-156. 1991.
- [3] V.A Armentano and C.R. Scrich. Tabú Search for minimizing total tardiness in a job shop, *International Journal Production Economics* 63 131 – 140 2000.
- [4] K.R. Baker. Sequencing Rules and Due Date assignments in a Job Shop. *Management Science* 30, 1093-1104. 1984.
- [5] A.Bauer et al. Minimizing total tardiness on a single machine using Ant Colony Optimization. *Cejor* 8: 125-141. 2000.
- [6] T.C. Cheng, E; and M.C Gupta. Survey of Scheduling Research Involving Due Date determination Decisions, *European Journal of Operational Research* 47, 271-292. 1989.

- [7] A. Colomi et al. Ant system applied for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science [JORBEL]*, 34: 39- 53, 1994.
- [8] A. Colomi; M. Dorigo and V. Maniezzo. *Ants System applied to the Quadratic Assignment Problem*. Technical Report No. 94- 28, IRIDIA, Brussels, Belgium. 1994.
- [9] G. M. Cox and W. Cochran. *Diseños experimentales*. Jhon Wiley & Sons. 1965.
- [10] JI. Deneubourg J.M. Pasteels and JC. Verhaeghe. Probabilistic behavior in ants: a strategy of errors? *J Theor Biol* **105**: 259-271, 1983.
- [11] M. Dorigo. *Optimization, Learning and Natural Algorithms [in Italian]*. PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, IT, 1992.
- [12] M.Dorigo and L.M. Gambardella. Ant colonies for the traveling salesman problem. *BioSystems*, 43: 73-81, 1997.
- [13] M. L. Fisher et al. Surrogate Duality Relaxation for Job-Shop Scheduling, *Discrete Applied Mathematics*, 5(1), 65-75. 1983.
- [14] C.Gagne ; M Gravel & W I. Pric. Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence dependent setup times. *Journal of the Operational Research Society*. **53**: 895 – 906. 2002.
- [15] L. M. Gambardella; E. Taillard and G. Agazzi. Ant colonies for vehicle routing problems. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*. McGraw- Hill, 1999.
- [16] F. Glover & M. Laguna. *Tabú Search*. Kluwer Academic Publishers.1997.
- [17] R. Graham et al. Optimization and approximation in deterministic sequencing and scheduling ; A survey, *Annals of Discrete Mathematics* 5 287 – 326. 1979.
- [18] D. J. Hootom; P. B. Luh and K. R. Pattipati. Practical Approach to Job-Shop Scheduling Problems, *IEEE Trans Rob Autom*, Feb, 9 (1), 1-13. 1993.
- [19] A. S. Jain and S. Meeran, A State-of-the-Art Review of Job-Shop Scheduling Techniques, *Journal of Heuristics*. 1998.
- [20] S. Kirkpatrick; C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, *Science*, 220(4598), 13 May, 671-680. 1983.
- [21] M.Laguna; J. W. Barnes and F. W. Glover. Tabú Search Methods for a Single Machine Scheduling Problem, *Journal of Intelligent Manufacturing*, vol 2, 63-74. 1991.
- [22] J.K Lenstra; AHG Rinnoy Kan and P. Bruckner. Complexity of machine scheduling problems. In: Hammer PL, Johnson EL, Korte BH, Nemhauser GL (eds) *Studies in Integer Programming*, *Annals of Discrete Mathematics* 1. North- Holland, Amsterdam, pp 343- 362, 1977.

- [23] N. Metropolis et al. Equation of State Calculations by Fast Computing Machines, *The Journal of Chemical Physics*, 21(6), June, 1087-1092. 1953.
- [24] D. C. Montgomery. *Design and Analysis of Experiments*, United States. Jhon Wiley & Sons. 1997.
- [25] E. Nowicki, and C. Smutnicki. A fast tabu search algorithm for the job shop problem, *Management Science* 42, 797 – 813. 1996.
- [26] F. Pezzella; E. Merelli. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*.
- [27] M. Pinedo and X. Chao. *Operations Scheduling With Applications in Manufacturing And Service*. Boston: Irwin/McGraw-Hill. 1999.
- [28] M. Pinedo and M. Singer. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop., *Naval Research Logistics*; 46:1–17. 1999.
- [29] E. Silver, et al. *Inventory Management Production Planning and Scheduling*.
- [30] M. Singer and M. Pinedo. A Computational Study of Branch and Bounds Techniques for Minimizing the Total Weighted Tardiness in Job Shops, *IIE Transactions* 30, 109.118. 1998.
- [31] O. Van Laarhoven; E. Aarts and J. Lenstra. Job shop scheduling by simulated annealing, *Operations Research* 40 113- 125. 1992.
- [32] É. Taillard. Parallel Taboo Search Techniques for the Job-Shop Scheduling Problem. *ORSA Journal on Computing*, 16(2), 108-117. 1994.