IMPLEMENTACIÓN DE UN HÍBRIDO ENTRE GRASP Y TABÚ SEARCH PARA LA SOLUCIÓN DEL PROBLEMA DE PROGRAMACIÓN DE LA PRODUCCIÓN EN UN AMBIENTE JOB SHOP PARA LA MINIMIZACIÓN DE LA TARDANZA TOTAL PONDERADA

JUAN SEBASTIÁN CABALLERO

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA INDUSTRIAL

Bogotá, D. C.
2014

IMPLEMENTACIÓN DE UN HÍBRIDO ENTRE GRASP Y TABÚ SEARCH PARA LA SOLUCIÓN DEL PROBLEMA DE PROGRAMACIÓN DE LA PRODUCCIÓN EN UN AMBIENTE JOB SHOP PARA LA MINIMIZACIÓN DE LA TARDANZA TOTAL PONDERADA

JUAN SEBASTIÁN CABALLERO

Trabajo de grado para optar al título de ingeniero industrial

Asesor
ELIANA MARÌA GONZALEZ NEIRA
Ingeniera Industrial

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA INDUSTRIAL
Bogotá, D. C.
2014

Tabla de contenido

Índice de Ilustraciones

Índice de Tablas

1. Introducción

El problema de programación de la producción para ambientes de trabajo Job Shop, es un tema muy investigado en los últimos años, debido a su gran nivel de aplicabilidad e importancia en la industria manufacturera. Un gran número de algoritmos de construcción y meta heurísticas, han sido utilizados para resolver éste problema, tomando diversos objetivos tales como la minimización del makespan, la minimización de la tardanza total, la minimización de la tardanza total ponderada, entre otros.

El presente trabajo resuelve un Job shop para la minimización de la tardanza total ponderada ya que ésta es una medida de desempeño que tiene en cuenta no solo el nivel de cumplimiento de los clientes sino la importancia de los mismos. Como método de solución se propone un algoritmo híbrido entre la metodología GRASP la cual no ha sido muy estudiada para la solución de éste problema (y es de gran ayuda para la construcción inicial de una solución), y la búsqueda tabú (con la cual se han obtenido muy buenos resultados para Job Shop) para la fase de búsqueda local del algoritmo. Los resultados obtenidos se comparan con el algoritmo de búsqueda local genética propuesto por (Essafi, Mati, & Dauzère-Pérès, 2008).

Éste documento presenta inicialmente el planteamiento de problema y la justificación del mismo, seguido por una explicación del problema, la meta heurística realizada, y los antecedentes relacionados con investigación del problema y métodos de solución propuestos para éste. Posteriormente se plantean los objetivos y alcance del documento, junto con el desarrollo, análisis de resultados del mismo, y finalmente algunas recomendaciones para futuros trabajos.

2. Glosario

Lateness: El lapso de tiempo comprendido entre la terminación de un trabajo con el tiempo deseado de terminación (Mellor, 1966)

Restricciones de precedencia (Prec): Este término implica que una o más operaciones deben ser realizadas antes de que un trabajo pueda inicializar su procesamiento en una máquina en particular (Metta, 2008)

Derecho preferente (Preemption prmp): Término usado para indicar que la operación O_{ij} del trabajo j puede ser interrumpida en cualquier momento durante su procesamiento en la máquina i. (Metta, 2008)

Tiempo de compleción (C_j): Tiempo en que el trabajo j es procesado completamente. (Metta, 2008)

Makespan (C_{max}): Tiempo que representa la terminación de todos los trabajos en el sistema.

Tardiness ($T_j = max\{C_j - d_j\}$): Indica la eficiencia en que el sistema enfrenta las fechas de entrega acordadas con el cliente. Es definida como el valor máximo entre 0 y la diferencia entre el tiempo de compleción y el due date. Si un trabajo se termina después de su due date, este es tardío. (Metta, 2008)

Lower Bound (LB): Mientras todas las operaciones no hayan sido programadas, el LB es un estimado del makespan, correspondiente a un valor menor o igual al óptimo (Jain & Meeran, 1999)

Upper Bound (UB): Es una cota superior del verdadero valor óptimo de la función objetivo.

SDST: Tiempos de Setup dependientes de la secuencia

JSSP (Job shop Scheduling Problem): Se refiere al problema de programación en un ambiente Job shop

FIFO: "Primero en entrar, primero en salir", y se refiere a la norma usada en la lista tabú en donde al momento en que se llene y se deba colocar un nuevo elemento, el primero en entrar a la lista, será el primero en salir para darle entrada al nuevo elemento.

Release time (r_j): Tiempo de preparación para el inicio de procesamiento de un trabajo. Se puede dar por llegada de material prima, por disposiciones de gerencia, entre otros.

3. Planteamiento del problema y Justificación

La globalización y la competencia del mercado en los últimos años ha creado en las empresas la necesidad de mejorar sus procesos, costos, calidad y servicio al cliente de manera paralela, sin que el mejoramiento de uno de estos factores afecte otro. Esto lleva a los empresarios a buscar metodologías que integren un análisis cuantitativo con los aspectos cualitativos, generando así un complemento entre al análisis cualitativo y el cuantitativo.

Uno de los puntos fundamentales para abarcar estos requisitos del mercado y de los clientes es tener una producción que se adapte a la calidad requerida y a los tiempos de entrega prometidos al cliente, y a la vez, que optimice el uso de maquinaria, mano de obra y materiales para minimizar así costos e inventarios innecesarios. Para esto es necesario realizar una programación de la producción que lleve a la compañía al mayor nivel de competitividad.

Según (Groover, 2007) existen tres tipos de producción asociados a la manufactura discreta (maneja todos los aspectos de la producción, compras, inventario, y así sucesivamente):

- Producción Job Shop (Bajos Volúmenes de producción)
- Producción por lotes (Lotes de productos)
- Producción en masa

Uno de los modelos de la programación es el modelo Job shop, donde los trabajos requieren una secuencia dada (particular para cada trabajo o tipo de trabajo) para su procesamiento como por ejemplo, fábricas de pintura, impresión, carpinterías, fábricas de reparación de autos, fábricas de procesamiento de metal, entre muchos otros. Éste se basa principalmente en la programación de un grupo finito de trabajos en un grupo finito de máquinas, teniendo en cuenta que cada máquina puede procesar solo un trabajo a la vez y cada uno de los trabajos tiene una secuencia específica en el conjunto de todas las máquinas (Ivan Lazar, 2012).
(Beenhakker, 1963) Elaboró una lista de objetivos para el JSSP, los cuales generan una ventaja competitiva en el mercado, entre los cuales se muestran los más relevantes a continuación:

- Mínimo uso de las instalaciones
- Mínimo Inventario de producto en proceso
- Mínimos costos de set up de las instalaciones
- Estabilidad de la mano de obra (que ésta trabaje al mismo ritmo siempre)
- Adherencia a la fecha de entrega prometida
- Máxima rata de producción
- Mínimo uso de materiales

- Adherencia a prioridades de trabajos (según deseado)
- Sensibilidad a cambios bruscos de producción
- No dependencia de procesos poco confiables
- Capacidad de reserva para órdenes especiales
- Óptimo en programación de transporte en planta
- Costos mínimos de distribución
- Costos totales esperados mínimos
- Utilización máxima de mano de obra
- Asignación óptima de trabajos
- Mínimo inventario de producto terminado
- Mínima obsolescencia y daño de productos
- Makespan más pequeño para ciertos productos
- Riesgo mínimo de pérdidas excesivas

Aunque sería ideal que todos y cada uno de estos objetivos sean optimizados, esto en términos matemáticos es muy complicado, y no hay evidencia de la eficiencia en el proceso de programación teniendo en cuenta tantos objetivos simultáneamente. Por tanto, la programación es realizada para optimizar objetivos específicos que envuelven parte de los mencionados. Estos objetivos son determinados con respecto a la demanda del mercado, las necesidades del interesado, las demandas de la compañía y la satisfacción del cliente (Madivada & Rao, 2012). Existen dos objetivos principales para la programación de la producción: el makespan y los costos de entregas tardías.

La mayoría de literatura concerniente a la programación de la producción en un ambiente Job Shop se centra en la minimización del Makespan como objetivo principal (Ivan Lazar, 2012). La minimización del makespan trae muchas ventajas a la hora de realizar una programación de la producción, pero la mayoría de éstas van de la mano con la compañía, y dejan a un lado las necesidades de los clientes. Hoy en día, la concentración en los factores externos tales como el medio ambiente, la responsabilidad social y los clientes finales, es de vital importancia para las organizaciones. El cumplimiento con los requerimientos del cliente, tales como la calidad del producto, la entrega rápida, los bajos costos, entre otros, es necesario para mantenerse competitivo en el mercado.
Por esto en el presente trabajo, se buscará como objetivo principal, el cumplimiento en las fechas de entrega (due dates), poniendo al cliente como prioridad. Adicionalmente para estar más acorde con las tendencias actuales del mercado, en donde algunos clientes son de mayor importancia que otros, ya sea debido al volumen histórico en pedidos, o al nombre del cliente (que puede garantizar publicidad misma para la compañía), a preferencias de la gerencia, entre muchas otras razones, se contemplará una priorización de dichos trabajos, dándole un valor cualitativo a la búsqueda de la programación óptima.

Las técnicas utilizadas durante más de 50 años para la resolución de dicho problema son muy extensas y van desde simples reglas de despacho, hasta complejas meta-heurísticas

que resuelven problemas de gran magnitud en tiempos relativamente cortos. En la Ilustración 1 se muestran los diferentes métodos de solución aplicados en los últimos años (2008-2013) para resolver dicho problema, según la revisión literaria realizada, mostrando a los algoritmos genéticos y a PSO como los métodos más frecuentes en la literatura En esta investigación de la literatura de los últimos años (Anexo 1) se observa también que aproximadamente un 50% de los artículos consultados utilizan técnicas híbridas para resolver el problema, debido a que estas han demostrado tener un alto grado de eficiencia y eficacia a la hora de generar resultados para el JSSP.

Algunos métodos tales como Filter and fan F&F (Duarte, Rego, & Gamboa, 2009), Ant Bee Colony Optimization ABC (colonia de hormigas) (Zhang, ShijiSong, & ChengWub, 2013), Greedy randomized adaptive search procedures GRASP (Chassaing, y otros, 2013), demuestran en la literatura una gran eficiencia en la resolución del problema, aunque su estudio no ha sido muy frecuente en los últimos años. La metodología GRASP, utilizada con una frecuencia media en la resolución de problemas como Flow shop (Davoudpour & Ashrafi, 2009), ya estudiada anteriormente en Job shop (Aiex, 2003) (Binato, Hery, & Resende, A Grasp for Job Shop Scheduling, 2002), muestra un muy buen comportamiento en términos de optimalidad y rapidez computacional, por tanto se debería realizar más investigación para este método de solución, en Job shop, y compararlo con los métodos más utilizados en este tema.
Teniendo en cuenta el poderío de los métodos híbridos en la investigación para la resolución de JSSP, es fundamental seguir realizando investigación teniendo en cuenta esta técnica debido a que se demuestra que envuelve los métodos de aproximación que más se acercan al óptimo y que requieren menos tiempo de computación.
La búsqueda Tabú (TS) ha sido mostrada en la literatura como una de las metodologías más eficaces en Job shop y en muchos otros problemas de Programación de la producción, mostrando superioridad frente otros algoritmos tales como recocido simulado, algoritmos genéticos y redes neurales (Madivada & Rao, 2012) y demuestra grandes capacidades en lo que respecta a la búsqueda local (Meeran & Morshed, 2012).

Por todas estas razones es interesante resolver JSSP con una metodología poco frecuentada en la literatura como GRASP, realizando un híbrido con TS que prueba ser un método muy eficiente para complementar a la Técnica GRASP. No se evidencia en la literatura consultada la resolución del problema Job shop, con esta metodología híbrida, por tanto, **¿Podría un Híbrido entre GRASP y TS mejorar la calidad de las soluciones en un problema** $J|r_i|\sum w_i T_i$ (Graham, Lawler, Lenstra, & Kan, 1979) **en comparación con el método de búsqueda local genética GLS propuesto por** (Essafi, Mati, & Dauzère-Pérès, 2008) **y contra los mejores valores conocidos para las intancias benchmark** (Singer & Pinedo, 1998)**?**

El desempeño del algoritmo híbrido que se propondrá, como ya se expuso, será comparado con el híbrido entre algoritmos genéticos y búsqueda local GLS (Búsqueda local genética),

usando las instancias propuestas para el JSSP por (Singer & Pinedo, 1998), en donde se añaden a las instancias clásicas para el makespan, los due dates y pesos a los trabajos.

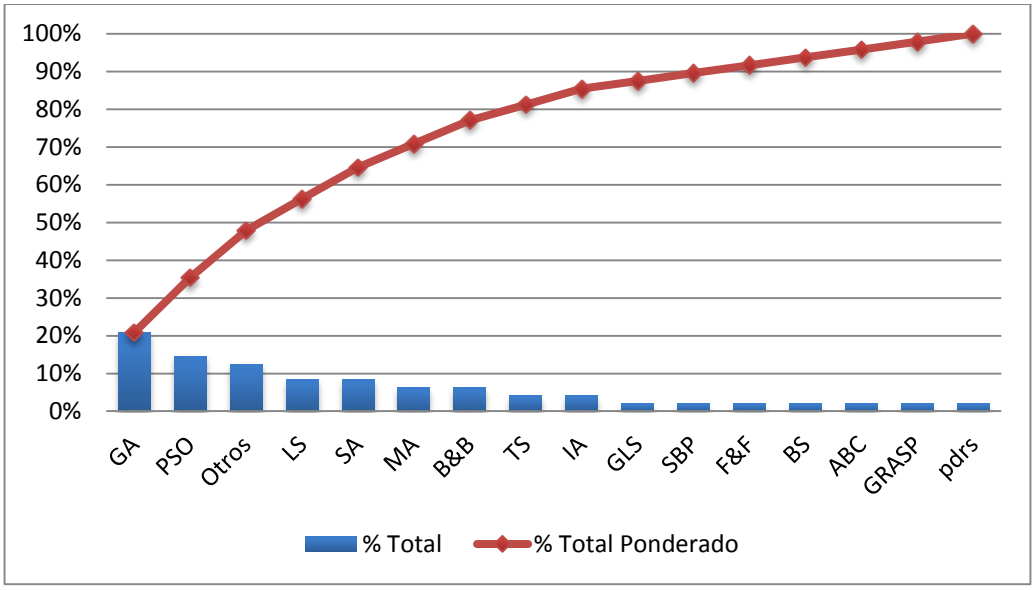

Ilustración 1. Técnicas usadas en JSSP años 2008-2013 (Fuente: Realizada por el autor)

Donde:
GA: Genetic Algorithms (Algoritmos genéticos)
GLS: Genetic Local Search (Búsqueda local genética)
MA: Memetic Algorithms
B&B: Algoritmos Branch and Bound
LS: Local Search (Búsqueda local)
F&F: Filter and Fan
PSO: Particle swarm optimization
F&F: Filter and Fan
TS: Tabú search (Búsqueda Tabú)
BS: Beam Search
SA: Simulated annealing (Recocido simulado)
IA: Inmune algorithm
SBP: Shifting Bottleneck Procedure (Heurística basada en el cuello de botella)
ABC: Artificial Bee Colony Algorithm (Colonia de abejas)
GRASP: Greedy Randomized adaptive search Procedure
Pdrs: Priority dispatching rules (Reglas de despacho)
AO: Ant Optimization (Colonia de hormigas)

4. Marco teórico

4.1. Job Shop Scheduling Problem (JSSP)

El problema de programación de tareas en un JSSP consiste en un grupo J de n trabajos $\boldsymbol{J_1, J_2 \ldots\ldots J_n}$ que deben ser procesados en un grupo M de m diferentes máquinas $\boldsymbol{M_1, M_2 \ldots\ldots M_m}$.
Cada trabajo J_j consiste en una secuencia de μ_j operaciones $O_{ij}\ (i = 1, \ldots, \mu_j)$ que deben ser procesadas de manera $\boldsymbol{O_{1j}, O_{2j}, \ldots, O_{nj}}$. La operación O_{ij} tiene un tiempo de procesamiento p_{ij}. El número de operaciones total dados todos los trabajos J, es determinado por $\boldsymbol{N = \sum_{j=1}^{n} \mu_j}$. Para J trabajos y M máquinas, en casos generales habrán $(\boldsymbol{j}!)^{\boldsymbol{M}}$ secuencias.

Las restricciones conjuntivas (de precedencia) son (ver ecuación 1)

$$\boldsymbol{\tau_{i(j+1)} - \tau_{ij} \geq p_{ij}}\ , \forall\left(\boldsymbol{O_{ij}; O_{i(j+1)}}\right) \in\ \boldsymbol{A_i} \qquad \textbf{(1)}$$

Las restricciones disyuntivas (de capacidad) son (ver ecuación 2):

$$\boldsymbol{\tau_{ij} - \tau_{kl} \geq p_{kl}}\ \textbf{Ó}\ \boldsymbol{\tau_{kl} - \tau_{ij} \geq p_{ij}}\ , \forall\left(\boldsymbol{O_{ij}; O_{kl}}\right) \in \boldsymbol{E_x} \qquad \textbf{(2)}$$

Dados,

$\boldsymbol{\tau}$ **=** Tiempo de inicio de una operación
$\boldsymbol{A_i}$ **=** Pares ordenados de operaciones restringidas por relaciones de precedencia para cada trabajo J_i
$\boldsymbol{E_x}$ **=** Todos los pares de operaciones a ser realizadas en la máquina

En la Ilustración 2 se presenta la gráfica disyuntiva, la cual es una de las representaciones más comunes de JSSP (Balas, 1969), allí cada nodo representa cada una de las operaciones O_{ij}, y se representan las restricciones conjuntivas con líneas de trazado completo, y las restricciones disyuntivas con líneas de trazado punteado. El “Source” y “el Sink” son nodos adicionales que representan el inicio y la terminación de todas las operaciones respectivamente. El método más utilizado en la literatura para la representación de la programación de la producción, es el diagrama de Gantt (Ilustración 3) el cual muestra al detalle la duración de cada una de las operaciones, y el orden de trabajos y máquinas que representan la programación óptima.

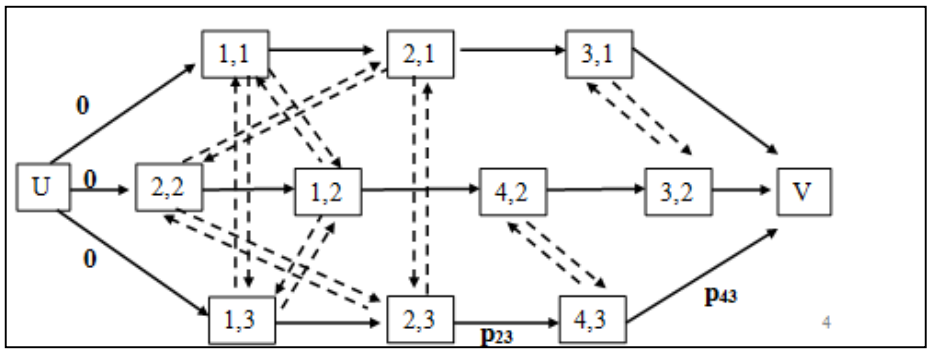

Ilustración 2. Gráfica Disyuntiva (Pinedo, 2012)

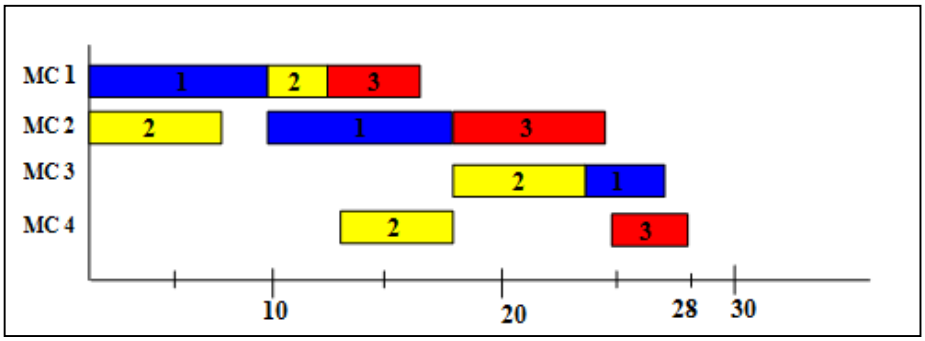

Ilustración 3. Diagrama de Gantt (Pinedo, 2012)

Las suposiciones principales de JSSP consideradas en el presente trabajo son:

- Ninguna máquina puede procesar más de una operación al tiempo
- Cada operación O_{ij} debe ser completada sin interrupciones
- Todos los trabajos deben ser completados (no hay cancelaciones)
- El desarrollo de cada operación tiene un tiempo finito, y cada una de éstas debe ser completada en su totalidad, antes de iniciar la otra que la sucede
- Los intervalos de tiempo de procesamiento son independientes del orden en que las operaciones son desarrolladas
- Es permitido el inventario de producto en proceso
- Las máquinas nunca se dañan y la mano de obra se mantiene constante (no hay cansancio)
- El ruteo de los trabajos dado debe ser llevado a cabo sin cambios (no se permite ruteo alternativo)
- No hay grupos de máquinas iguales
- Cada trabajo debe ser secuenciado en cada máquina en un orden predefinido por una secuencia de operaciones
- Los tiempos de procesamiento para cada operación son conocidos

4.2. GRASP

La metodología GRASP (Greedy randomized adaptive search procedures) es un proceso iterativo, en donde cada iteración GRASP consiste en dos fases: Una fase constructiva y una fase de búsqueda local (Feo & Resende, 1995).
Según (Binato, Hery, & Resende, 2002), en la fase constructiva, se crea una solución factible, elemento por elemento. Se genera una lista de candidatos llamada RCL (Restricted candidate list), en donde, por cada iteración de la fase constructiva, la operación a ser añadida a la programación es determinada por la elección de una de las operaciones en esta lista de manera aleatoria, o con una función de probabilidad.
No existe garantía de que en la fase constructiva se encuentre una solución óptima con respecto al vecindario adoptado, se realiza una fase de búsqueda local que mejorará la solución encontrada previamente.
En la búsqueda local el mecanismo de generación resalta un vecindario para cada una de las configuraciones. El objetivo de estas estrategias es realizar una perturbación de la configuración a través de una sucesión de vecinos para así dirigir la búsqueda a una solución satisfactoria (Meeran & Sheik, 1998). La búsqueda local posee algunas limitaciones, como por ejemplo, que dependen de múltiples corridas del algoritmo, para así poder obtener resultados significativos (cercanos al óptimo), y un buen grupo de parámetros deben ser cuidadosamente seleccionados para obtener buenas soluciones. Adicionalmente muchos métodos de búsqueda local son dependientes de las soluciones iniciales, por tanto una mala inicialización puede conllevar a programaciones de mala calidad, o tiempos de cómputo excesivos.

4.3. Búsqueda Tabú (TS)

Es un método de aproximación iterativa el cual ha sido generador de muy buenas formulaciones para JSSP. TS es una técnica simple que inteligentemente guía un proceso de búsqueda, lejos de soluciones que aparentemente duplican o copian soluciones previamente obtenidas. (Meeran & Sheik, 1998).
Consiste en llegar a un óptimo local mediante el archivo del histórico de búsqueda en su memoria. Este prohíbe movimientos en el vecindario dados ciertos atributos, con el fin de guiar el procedimiento de búsqueda lejos de soluciones que duplican o se asemejan a soluciones previamente conseguidas.
La idea básica de la búsqueda tabú (Glover, 1990) es explorar todas las programaciones posibles mediante una secuencia de movimientos. El movimiento de una programación a otra es hecho mediante la evaluación de todos los candidatos y la elección del mejor. Algunos movimientos son clasificados como Tabú (prohibidos) debido a que atrapan la búsqueda en un óptimo local, o bien, llevan a ciclos sin fin en la búsqueda. Estos movimientos son colocados en una lista tabú, que es construida por medio del histórico de movimientos usados durante la búsqueda. Otra característica de TS es el congelamiento

de la búsqueda en una memoria a corto plazo que provee un “olvido estratégico” de la misma.

4.4. Genetic local search (GLS)

Los algoritmos genéticos han sido ampliamente estudiados en los últimos años, para resolver el problema de programación de la producción en un ambiente Job shop (Ilustración 1). El algoritmo genético comienza con una población generada aleatoriamente llamada cromosomas, los cuales representan la solución del problema. Estos son evaluados para la función objetivo y elegidos teniendo en cuenta el valor fitness. Muchos procedimientos de selección están basados en el valor fitness de los individuos de la generación actual. Para mejorar esta generación, se utilizan operadores llamados crossover y mutación. (Madivada & Rao, 2012)

GLS es una técnica de búsqueda local de dos niveles en donde se usa como solución inicial una solución dada por algoritmos genéticos. La búsqueda local dirige los hijos al punto óptimo local más cercano el cual es operado en la siguiente generación por los operadores tradicionales de recombinación genética (Essafi, Mati, & Dauzère-Pérès, 2008)

5. Antecedentes

La investigación en teorías de programación de la producción en ambientes Job Shop ha evolucionado en los pasados 50 años y ha estado sujeta a extensa literatura relevante que contiene técnicas muy innovadoras que van desde simples reglas de despacho, hasta sofisticados algoritmos paralelos de branch and bound, y heurísticas basadas en cuello de botella (A.S.Gromicho, Hoorn, Saldanha-da-Gama, & Timmer, 2012). Se han usado métodos exactos, que encuentran el óptimo global en problemas pequeños, pero que en problemas largos no se desempeñan como se espera (en términos de tiempos computacionales) por su complejidad. Y se han desarrollado métodos de aproximación, que si bien, muchas veces no encuentran el óptimo global, encuentran óptimos locales muy cercanos al global en un tiempo muy reducido. Y en los últimos años, se ha venido desarrollando métodos híbridos que resultan de la combinación de dos o más técnicas. En la Ilustración 4 y la Ilustración 5, basadas en los estudios realizados por (Madivada & Rao, 2012) (Jain & Meeran, 1999) se muestra un resumen de los métodos principales estudiados en la literatura para resolver este problema, en donde se muestran dos grandes técnicas las cuales son las técnicas de optimización (exactas) y las técnicas de aproximación.

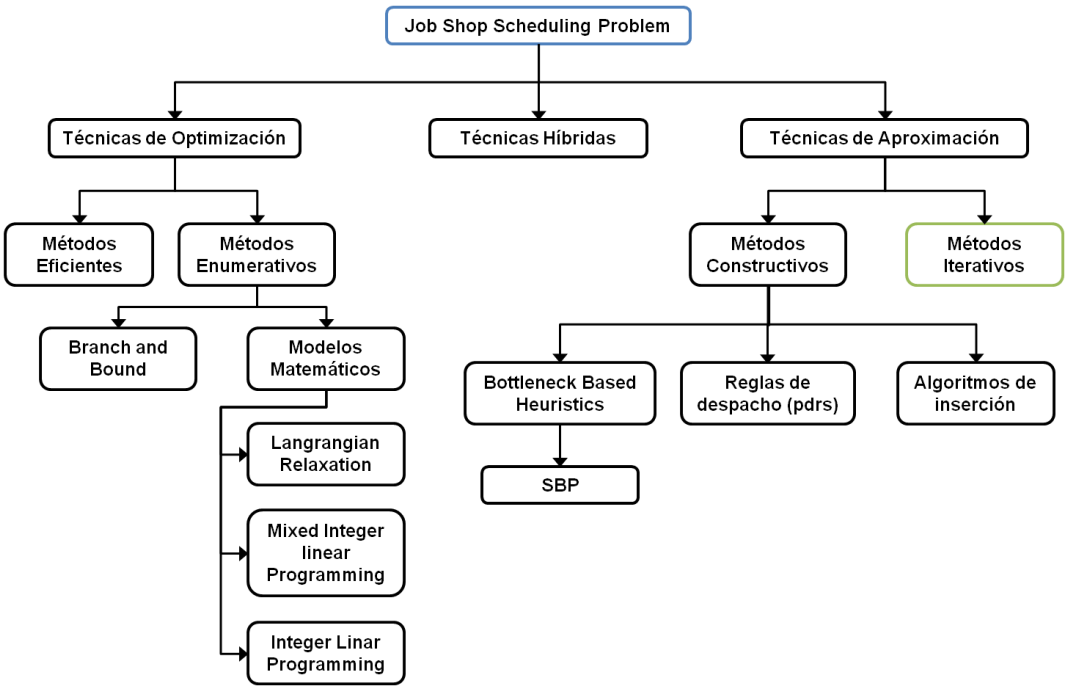

Ilustración 4. JSSP Antecedentes (Fuente: Realizada por el autor)

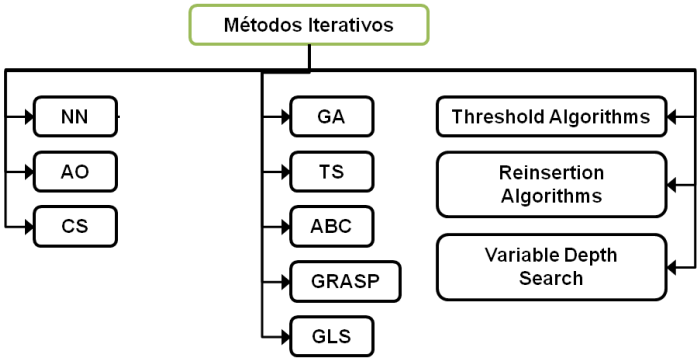

Ilustración 5. JSSP Antecedentes - Métodos Iterativos (Fuente: Realizada por el autor)

Los **métodos exactos** se caracterizan por buscar solución al problema de manera exacta, sin importar la longitud de la solución o la dificultad para la misma. Dos grandes métodos se derivan de estas técnicas exactas, los métodos eficientes, y los métodos enumerativos. Un algoritmo eficiente resuelve un problema dado con un requerimiento que crece de forma polinomial con respecto al tamaño de los datos de entrada. Estos métodos construyen una solución óptima siguiendo reglas exactas que al final dan paso al orden de procesamiento deseado. Los primeros trabajos en este tipo de métodos se pueden observar en (Johnson, 1953) el cual resuelve un Flow shop de 2 y 3 máquinas con un algoritmo eficiente. Se pueden encontrar en la literatura algunos otros avances en métodos eficientes, ya que estos fueron pioneros para la investigación de la optimización del makespan en la programación de la producción. (Meeran & Sheik, 1998) Resalta en su investigación que los métodos eficientes no pueden solucionar JSSP donde m ≥ 3 y $n \geq 3$.

Por otro lado, entre los métodos enumerativos se pueden encontrar los algoritmos Branch and bound (B&B), los cuales usan un árbol dinámico que representa el espacio de solución para todas las secuencias posibles. La búsqueda comienza desde la raíz y una selección completa es alcanzada hasta que el nivel más bajo (“la hoja”) sea evaluado.
Algunos de los trabajos iniciales en esta área fueron realizados por (Brooks & White, 1965). (Brucker, Jurisch, & Sievers, 1992) realizaron un algoritmo Branch and Bound para minimizar el makespan, para un problema 10x10 el cual tuvo mucho éxito ya que no requirió de mucho esfuerzo computacional. Aunque la literatura presenta más información con respecto a mejoras al algoritmo Branch and Bound para JSSP, los métodos exactos presentan tiempos de cómputo muy altos, por lo que no es práctico solucionarlos de manera exacta. Entre otros métodos enumerativos se encuentran los modelos matemáticos, los cuales son muy útiles para la resolución de problemas de programación con N < 250. Entre los modelos matemáticos más utilizados están: (i) Mixed integer linear programming (MILP) (Manne, 1960). (ii) relajación langrangiana (LR) (Fisher, 1973) y (iii) Métodos de descomposición.

La otra gran rama de métodos de solución de la programación de la producción son los **métodos de aproximación**. Si un algoritmo para JSSP va a correr en un tiempo polinomial, entonces sólo puede garantizar una solución que es un porcentaje ρ de la solución óptima. Estos algoritmos se conocen como algoritmos de aproximación (Jain & Meeran, 1999). Estos garantizan una solución óptima en términos de velocidad y pueden ser utilizados para la resolución de problemas más amplios, lo que los hace más adecuados para las empresas ya que se pueden realizar cambios, inclusiones o exclusiones al programa, y éste puede dar un resultado cercano al óptimo rápidamente.
Las técnicas de aproximación tienen dos grandes ramas, (i) Métodos constructivos y (ii) Métodos iterativos.

Entre los métodos constructivos se encuentran las heurísticas basadas en el cuello de botella y las reglas de despacho. El Shifting Bottleneck Procedure (SBP) basado en el cuello de botella, según (Demirkol, E. Mehta, & Uzsoy, 1997) se caracteriza por las siguientes tareas: (i) Identificación de sub problemas, (ii) Elección del cuello de botella, (iii) Solución del sub problema, y (iii) Re optimización de la programación.

Por otro lado, las reglas de despacho constituyen uno de los métodos más investigados para resolver JSSP. Estas técnicas asignan una prioridad a todas las operaciones libres para ser secuenciadas y posteriormente se elige la operación con prioridad más alta. Son muy fáciles de implementar bajo uso computacional. El mayor problema se presenta en el momento de la búsqueda de minimización de makespan, ya que ninguna regla muestra gran superioridad frente a las otras (Jain & Meeran, 1999).
Según (Holthaus & Rajendran, 1997) las reglas de despacho pueden ser clasificadas en (i) Reglas basadas en el tiempo de procesamiento, (ii) Reglas basadas en el due date, (iii) reglas combinadas, y (iiii) Reglas que no se basan ni en due dates ni en tiempo de procesamiento. Una de las reglas más estudiadas en la literatura para la programación de la producción es Shortest Processing time (SPT) (Sels, Gheysen, & Vanhoucke, 2012) y (Jayamohan & Rajendran, 2004) presentan análisis de las reglas de despacho más aplicadas para el JSSP, y se realizan comparaciones entre ellas para determinar su efectividad a la hora de la programación.

La otra gran rama concerniente a los métodos de aproximación son los métodos iterativos, una parte de estos son basados en la inteligencia artificial. Éste es un sub-campo de las ciencias computacionales que busca la integración de la biología y la computación. Sus orígenes fundamentales están ligados a análisis biológicos y los usos de principios de la naturaleza para encontrar soluciones a los problemas. Los métodos de aproximación más acertados son los basados en la búsqueda local, ya que estos se acercan al óptimo global en un gran porcentaje, y el tiempo de cómputo no es muy excesivo. Gran parte de la literatura está ligada a la comparación de las distintas técnicas de aproximación para determinar cuál es mejor y en qué casos.

Entre los métodos de aproximación usados para la resolución de JSSP, se encuentran:

- las redes neurales (NN) las cuales se basan en la estructura cerebral de entidades vivientes. En esta técnica la información en proceso es llevada por medio de una red masiva de unidades paralelas de procesamiento (Meeran & Sheik, 1998);
- los algoritmos genéticos, basados en la evolución natural, en donde la calidad de los organismos crece hasta llegar al nivel más compatible con el medio ambiente; Colonia de abejas (ABC) (Zhang, ShijiSong, & ChengWub, 2013);
- colonia de Hormigas (AO);
- recocido simulado (SA) (Rui Zhang, 2011);
- búsqueda Tabú (TS) (Li, Pan, & Liang, 2010), entre otros.

Para más detalles de la historia previa a los años recientes para resolver el problema de programación de la producción en un ambiente Job shop, las revisiones de (Jain & Meeran, 1999) (Meeran & Sheik, 1998) (Subramaniam & Raheja, 2002) (Mellor, 1966) muestran un vasto recorrido histórico que resalta la superioridad de TS y otras técnicas de aproximación en estos años.

Recientemente (Ivan Lazar, 2012) (Madivada & Rao, 2012) presentan revisiones de la literatura de los últimos años para π_j donde demuestran que los algoritmos genéticos es una de las técnicas más revisadas e investigadas. Se presenta un gran aumento en los últimos años en la utilización de meta-heurísticas y métodos híbridos para resolver problemas de JSSP. Para complementar la revisión realizada por los estudios mencionados, se realiza una búsqueda de literatura concerniente al JSSP, la cual se encuentra resumida en el Anexo 1. En ésta se puede evidenciar que la minimización del makespan ya no es prioritaria a la hora de realizar investigación, ya que se apunta al realismo un poco más que a la teoría. Adicionalmente la investigación en meta-heurísticas predomina, y los métodos eficientes ya no son un blanco de estudio primario. En la Ilustración 1 se muestra la frecuencia de estudio de distintas técnicas de solución del problema de programación de la producción Job Shop, y se evidencia que los algoritmos genéticos y PSO, son los métodos más frecuentados en la literatura. Cabe resaltar también que se han empezado a investigar otros métodos de solución distintos a los más comunes y conocidos. Por ejemplo un híbrido basado en PMBGA (probabilistic model-building genetic algorithm), y un algoritmo basado en la perturbación de parámetros (PP) (Zhang & Wu, 2012), el uso de la técnica Filter and Fan (Duarte, Rego, & Gamboa, 2009), entre otros.

6. Objetivos y Alcance

6.1. Objetivo General

Proponer un método para minimizar la tardanza total ponderada en la programación de la producción de un ambiente Job Shop $J|r_i|\sum w_i T_i$ a través del uso de la meta-heurística GRASP híbrida con búsqueda tabú como método de búsqueda local

6.1.1. Objetivos Específicos

- Proponer un algoritmo Híbrido, basado en la meta-heurística GRASP, con TS como búsqueda local, para resolver el problema $J|r_i|\sum w_i T_i$
- Realizar un diseño experimental para determinar una configuración adecuada de parámetros, referente al método híbrido propuesto para la solución de $J|r_i|\sum w_i T_i$
- Realizar pruebas del algoritmo con instancias generadas y comparar el desempeño de la meta-heurística híbrida propuesta contra la búsqueda local genética propuesta por (Essafi, Mati, & Dauzère-Pérès, 2008) y los mejores valores conocidos para instancias benchmark del problema

6.2. Alcance

Este proyecto pretende explorar el uso de la metodología GRASP utilizando en la fase de búsqueda local TS, como método para resolver el problema de la minimización de la tardanza total ponderada en un ambiente Job Shop.
Este proyecto pretende comparar los resultados contra la búsqueda local genética, que recrea un híbrido entre un algoritmo genético y la búsqueda local.

7. Metodología

Objetivo Específico	Actividades a realizar	Resultados esperados
Proponer un algoritmo Híbrido, basado en la meta-heurística GRASP, con TS como búsqueda local, para resolver el problema $J\|r_i\|\sum w_i T_i$	Estudiar a fondo la implementación de algoritmos basados en la metodología GRASP, para ver como realizar su implementación al problema	Algoritmo Híbrido entre GRASP y TS como búsqueda local para la solución del JSSP para la minimización de la tardanza total ponderada
	Estudiar la implementación de TS como búsqueda local en GRASP, para definir su implementación en el problema	
	Realizar el pseudocódigo	
	Implementar el algoritmo en el lenguaje de programación JAVA usando Netbeans IDE 8.0	
Realizar un diseño experimental para determinar una configuración adecuada de parámetros, referente al método híbrido propuesto para la solución de $J\|r_i\|\sum w_i T_i$	Realizar un diseño experimental para determinar la configuración de parámetros adecuada.	Configuración de parámetros para el método híbrido entre GRASP y TS para la solución de $J\|r_i\|\sum w_i T_i$
	Realizar la configuración de parámetros de el Híbrido entre GRASP y TS	
Realizar pruebas del algoritmo con instancias generadas y comparar el desempeño de la meta-heurística GRASP-TABÚ contra la búsqueda local genética propuesta por (Essafi, Mati, & Dauzère-Pérès, 2008) y los mejores valores conocidos para instancias benchmark del problema	Ejecutar el algoritmo según instancias dadas por (Singer & Pinedo, 1998). Hacerlo tantas veces sea necesario	Solución del problema (Programación) con el algoritmo Híbrido
	Comparar los resultados obtenidos con los resultados de GLS para el problema, dados por (Essafi, Mati, & Dauzère-Pérès, 2008)	Comparación estadística de resultados, determinación de la mejor metodología entre las comparadas

Tabla 1. Metodología

8. Desarrollo de la solución propuesta

8.1. Notación

- **n**

  Número de trabajos. Para las instancias benchmark seleccionadas, el número de trabajos es de 10.

- **m**

  Número de máquinas. Para las instancias benchmark seleccionadas, el número de máquinas es de 10.

- **N**

  Número de operaciones totales.

- r_j

  Fecha de lanzamiento (reléase date) del trabajo j dado j = $\{1,2,3 \dots n\}$

- f

  Factor de ajuste (Tightness factor) de la fecha de entrega del trabajo j, para las instancias seleccionadas. En las instancias benchmark, se presentan 3 valores para el factor de ajuste, los cuales son 1.3, 1.5, y 1.6 respectivamente.

- d_j

  Fecha de entrega (due date) del trabajo j dado j = $\{1,2,3 \dots n\}$. Para las instancias benchmark seleccionadas, el due date depende de la fecha de lanzamiento del trabajo j y el factor de ajuste (f), la fórmula para la determinación de la fecha de entrega para cada uno de los trabajos j de las instancias de (Singer & Pinedo, 1998) es:

$$d_j = r_j + \left\lfloor f * \sum_{i=1}^{10} p_{ij} \right\rfloor \quad (3)$$

- w_j

  Peso del trabajo j dado j = $\{1,2,3 \dots n\}$. En las instancias seleccionadas de presentan 3 valores para los pesos de los trabajos, los cuales son 4.0 para los trabajos de alta prioridad, 2.0 para los trabajos de mediana prioridad, y 1.0 para los trabajos de baja prioridad.

- O_{ij}

  Indica el orden de la secuencia en que debe ser procesado el trabajo j en la máquina i

- p_{ij}

  Tiempo de procesamiento del trabajo j en la máquina i

8.2. Modelo General

El híbrido entre GRASP y búsqueda tabú propuesto (Ilustración 6), se focaliza en la construcción de soluciones (Fase constructiva), que faciliten la búsqueda local posterior por medio de la generación de vecindarios que apunten al óptimo buscado. Esto se realiza mediante estrategias como la evaluación respecto a una función de utilidad lineal que se enfoca en la asignación de operaciones teniendo en cuenta la tardanza de los mismos, y el nivel de aleatoriedad controlado por el factor greedy el cual será explicado más adelante.

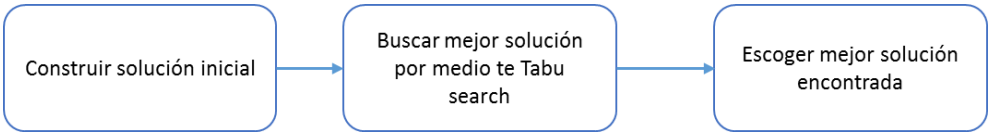

Ilustración 6. Pasos algoritmo propuesto (Fuente: realizada por el autor)

Para el GRASP, tal como fue establecido anteriormente, se utiliza como búsqueda local, la búsqueda tabú, la cual se encarga del mejoramiento de la solución entregada por la fase constructiva del algoritmo, y utiliza estrategias de memoria para evitar el estancamiento en óptimos locales (Ver pseudocódigo Algoritmo 1). Para el algoritmo propuesto, se establece como criterio de parada, el número de iteraciones de GRASP, el cual se fija en 500 iteraciones, lo cual garantiza la búsqueda de un buen resultado teniendo en cuenta los tiempos computacionales de procesamiento.

Algoritmo 1

Programa <GRASP-TABÚ>

Para <i=0> **desde** <i> **hasta** CantIteracionesGRASP> **inc** <1>

    Construir solución inicial

    Mientras <# iteraciones TABU **<** # iteraciones TABU criterio de parada>

        Iterar según estrategia de vecindario

        Si <el valor de la nueva tardanza total ponderada **<** Valor inicial>

Reemplazar solución – Establecer nueva tardanza óptima

Si no Continuar con solución inicial

Fin Si

Fin Mientras

Fin Para

Imprimir mejor solución encontrada dentro de las iteraciones

Fin <GRASP-TABÚ>

A continuación, se presenta una descripción más a fondo de cada una de las fases del algoritmo GRASP-TABÚ, y las estrategias propuestas dentro del algoritmo para inclinarse hacia la búsqueda de óptimos globales para los problemas de las instancias benchmark utilizadas (Singer & Pinedo, 1998)

8.2.1. Fase constructiva

GRASP dirige la mayor parte de sus esfuerzos a construir soluciones de alta calidad que son posteriormente procesadas con el fin de conseguir aún mejores soluciones (Gonzalez, 2005)

El mecanismo de construcción considerado, construye una solución incorporando un elemento (operación) a la vez, así en cada paso de este proceso, se genera una solución parcial, por medio de la asignación inicial de todas las operaciones para cada uno de los trabajos.

Para la fase constructiva (Ilustración 7) de este problema, inicialmente se genera una lista primaria de candidatos, según restricciones de precedencia, y disponibilidad de operaciones teniendo en cuenta el espacio de tiempo en que se encuentre la iteración. Se tiene en cuenta el Release time y la secuencia de operaciones para cada uno de los trabajos, así como los tiempos de procesamiento en cada una de las máquinas (evalúa si el trabajo está siendo realizado o si la máquina requerida están siendo utilizada)

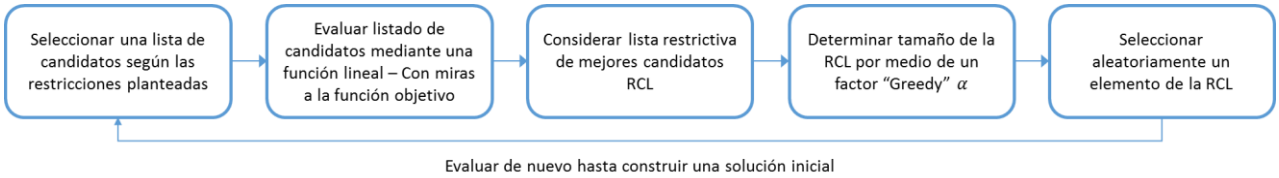

Ilustración 7. Pasos para la fase constructiva de GRASP (Binato S. , Hery, Loerstern, & Resende, 2009)

Posteriormente se evalúa cada uno de estos candidatos O_{ij} (Operación del trabajo j en la máquina i) con una función preestablecida, la cual tiene como objetivo la minimización de la tardanza de cada uno de los trabajos J_n. La función de utilidad determinada para garantizar la búsqueda acertada inclinándose hacia la minimización de la tardanza total ponderada, es la siguiente (Ver Ecuación 4):

$$\boldsymbol{G_{ij} = (d_j - (p_{ij} + T_j)) \times (1 - (w_j / \sum_{j=1}^{n} w_j))} \qquad \textbf{(4)}$$

Se ordenan los candidatos O_{ij} por su $\boldsymbol{G_{ij}}$ respectivamente de menor a mayor teniendo en cuenta los valores negativos (en caso que aplique), por medio de un "algoritmo de ordenación por el método de la burbuja".

Éste método consiste en una serie de intercambios de elementos adyacentes por medio de la comparación entre los mismos, colocando en le parte "superior" el menor valor comparado. Se realiza dicha comparación e intercambio para todos y cada uno de los valores candidatos (ver pseudocódigo Algoritmo 2).

Algoritmo 2

```
Subprograma <Algoritmo burbuja>
Para <posición i del arreglo> desde <1> hasta <Número de candidatos> inc <1>
Variable <int> <j>
j=i+1
        Para <posición j del arreglo> desde <2> hasta <Número de candidatos> inc <1>
                Si <el valor de i+j es menor que el valor de i>
                        Cambiar de posición
                Si no   Continuar con posiciones iniciales
                Fin Si
        Fin para
Fin para
Fin <Algoritmo burbuja>
```

Una vez realizado el ordenamiento, se realiza el cálculo del tamaño de la RCL teniendo en cuenta un factor "greedy" $\boldsymbol{\alpha}$, el valor del candidato O_{ij} mejor posicionado (aquel que tiene el valor mínimo de G_{ij}, el cual se nombrará con Min), y el valor del candidatoO_{ij} peor posicionado (El que tiene el máximo valor de G_{ij}, el cual se nombrará con Max). Se almacenan los candidatos ordenados, en la lista restrictiva de candidatos definida (tamaño) por la ecuación 5

$$\boldsymbol{Tamaño\ RCL} = \begin{matrix} \boldsymbol{Desde} & Min \\ \boldsymbol{Hasta} & (Max - Min) \times \alpha \end{matrix} \qquad \textbf{(5)}$$

Se elige aleatoriamente uno de los candidatos almacenados en la RCL se asigna, y se almacena la operación seleccionada, la cual pasa a ser parte de la solución (durante el tiempo de procesamiento de dicha operación O_{ij}, la máquina i y el trabajo j serán restringidos y no se podrán asignar para las próximas iteraciones). Se repetirá esta operación, hasta que todas las operaciones sean asignadas en su totalidad y se tenga una solución completa (todas las operaciones O_{ij} debidamente programadas).

El presente algoritmo (fase constructiva) será regido por instantes de tiempo absoluto {0, 1, 2,….t}, los cuales tendrán un papel fundamental a la hora de la elección de candidatos posibles para añadir a la solución. Se presentan las siguientes características:

- En cada instante de tiempo se buscarán los posibles candidatos a ser asignados. No se asignarán aquellas operaciones en las que el trabajo esté siendo realizado (Valor mayor a cero), que el trabajo esté en su reléase time, o que la máquina requerida por el trabajo esté siendo utilizada (Tiempo mayor a cero)
- Cada vez que se asigna una operación, se guardará el valor del tiempo de procesamiento al trabajo (j) y a la máquina a utilizar (i) respectivamente, con el objetivo de restringir su asignación, hasta que se complete su tiempo de procesamiento, para no asignarlos como candidatos hasta que se completen, así en cada unidad de tiempo que pasa, se buscan los trabajos previamente asignados y las máquinas utilizadas, y se les resta una unidad hasta que lleguen a cero (Se complete la operación) y el trabajo y la máquina se puedan volver a asignar
- Para la elección aleatoria de los candidatos en la RCL, se eligen aleatoriamente todos los candidatos posibles (Que no se restrinjan uno al otro) ya que en algunas ocasiones se puede asignar más de una operación en un instante de tiempo. Estos se eligen uno por uno, evaluando si el tiempo de la máquina está en 0 (para garantizar que cuando se elija un opcionado aleatoriamente de la RCL, si hay uno o más trabajos en la RCL como opcionados para trabajar con la misma máquina, éstos no puedan ser añadidos y deban ser evaluados en la siguiente iteración de la fase constructiva)
- El tiempo solo avanza, cuando no haya ningún candidato posible para asignar en el instante de tiempo en que se encuentre la evaluación
- El valor del factor greedy α se determina como un parámetro, y su valor es determinado en el **numeral 8.3.3** del presente documento
- La tardanza total ponderada $\sum w_i T_i$ se calcula cuando ya todas las operaciones hayan sido asignadas

A continuación se presenta el pseudocódigo para la fase constructiva del GRASP propuesto en el presente trabajo (Ver pseudocódigo Algoritmo 3)

Algoritmo 3

Programa <FaseConstructiva>

Recolección de datos // Se recogen los datos de un archivo plano, y se solicita α, el número de iteraciones de Grasp y de la fase iterativa (Tabú) y el tamaño de la lista tabú

Int Operaciones completadas = 0 // Número de operaciones completadas

Int TiempoAbsoluto = 0 // Instante de tiempo

List<Integer> tiemposTrabajos **= new** ArrayList<> // Contiene el tiempo que va a estar ocupado cada trabajo

List<Integer> tiemposMaquinas = **new** ArrayList<> // Contiene el tiempo que va a estar ocupada cada máquina

Para <i=0> **desde** <i> **hasta** <número de máquinas**> inc** <1>

List<Integer> MaquinasVSTrabajos = **new** ArrayList<> // Contiene la distribución de trabajos para cada una de las máquinas

Fin Para

Mientras <OperacionesCompletadas **<=** NumOperaciones>

Para <i=0> **desde** <i> **hasta** <número de trabajos **> inc** <1>

Si < Tiempo trabajo (i) **>** 0 > // Decremento de tiempos de trabajos y máquinas que estén asignados

Tiempo de trabajo (i) = Tiempo de trabajo (i) **-** 1

Tiempo de m que realiza el trabajo (i) = Tiempo de m que realiza el trabajo (i) - 1

Si < Tiempo trabajo (i) **==** 0 > // Aumento de Op completadas, y asignación de la siguiente Op del trabajo (i)

OperaciónActual del trabajo (i) = OperaciónActual del trabajo (i) + 1

OperacionesCompletadas = OperacionesCompletadas + 1

Fin Si

Si < OperaciónActual del trabajo (i) **=** NumeroOperaciones del trabajo (i) && TiempoEmpleado del trabajo (i) == 0 >

TiempoEmpleado del trabajo (i) **=** TiempoAbsoluto // Almacenar tiempo total de cada trabajo

Fin Si

Fin Si

Fin para

Mientras < True >

Int contador = 0

List<Dupla>opcionados = **New** ArrayList<> // En donde se agregarán los opcionados (Dupla de trabajo vs máquina)

Para <i=0> **desde** <i> **hasta** <número de trabajos > **inc** <1>

Si < OperaciónActual del trabajo (i) **!=** NumeroOperaciones del trabajo (i) > // Revisa si el trabajo (i) ya está terminado

Si < TiempoEmpleado del trabajo (i) = 0 **>** // Revisa si el trabajo (i) ya está terminado

Int MaquinaSolicitada por el trabajo (i) = **Buscar** MaquinaSolicitada por el trabajo (i)

Si < RJ del trabajo (i) **<=** TiempoAbsoluto **>** // Revisa si el trabajo (i) ya cumplió con su reléase time

Si < Tiempo de trabajo (i) **==** 0 **>** // Revisa que el trabajo no se esté realizando actualmente

Si < Tiempo MaquinaSolicitada por el trabajo (i)

== 0 **>** // Revisa que la máquina que necesita el trabajo (i) no se esté utilizando actualmente

Añadir dupla a opcionados (Trabajo, Máquina)

Si No // El trabajo no es un candidato

Contador = contador + 1

Fin Si

Si No // El trabajo no es un candidato

Contador = contador + 1

Fin Si

Si No // El trabajo no es un candidato

Contador = contador + 1

Fin Si

Si No // El trabajo no es un candidato

Contador = contador + 1

Fin Si

Fin Para

Si < Contador **>=** Número de trabajos >

Break (Terminar ciclo While)

Fin Si

Para <i=0> **desde** <i> **hasta** <número de opcionados > **inc** <1> //Ordenación Algoritmo Burbuja

Double actual = <<Aplicar fórmula>> // Se aplica fórmula para el candidato. Se usa variable tipo Double debido a que es un número real

Para <j=i+1> **desde** <j> **hasta** <número de opcionados > **inc** <1> //Ordenación Algoritmo Burbuja

Double siguiente = <<Aplicar fórmula>> // Se aplica fórmula para el candidato

Si < actual **>** siguiente >

Cambiar de posición

Fin Si

Fin Para

Fin Para

Double mínimo **=** Buscar valor mínimo en lista de candidatos // Valor mínimo - determinar tamaño de la RCL

Double máximo **=** Buscar valor máximo en lista de candidatos // Valor máximo – Determinar tamaño de la RCL

Double Rango = ((α*(máximo – mínimo)) + Mínimo) // Evaluar fórmula con el factor greedy

Para <i=0> **desde** <i> **hasta** <número de opcionados > **inc** <1>

Si < Valor fórmula opcionado (i) **<=** Rango >

Añadir opcionado a RCL // Se añaden opcionados a la lista restrictiva de candidatos cuando estos son menores al rango previamente establecido. Es decir se añaden valores entre el valor mínimo, y el rango.

Fin Si

Fin Para

Para <i=0> **desde** <i> **hasta** <número de opcionados de la RCL > **inc** <1>

<Elegir aleatoriamente un opcionado de la RCL>

Si < TiemposMaquinas del opcionado (i) == 0 > //Si el opcionado se puede añadir

Añadir opcionado a la solución

Asignar tiempo de procesamiento al TiemposMaquinas del opcionado (i) // Asigna tiempo de procesamiento para evitar asignar máquina mientras esté trabajando

Asignar el tiempo de procesamiento a TiemposTrabajos del opcionado (i) // Asigna tiempo de procesamiento para evitar asignar trabajo mientras se esté procesando

Contador = Contador + 1

Añadir opcionado en MaquinasVSTrabajos // Añadir trabajo asignado a la lista de la maquina asignada

Fin Si

Fin Para

Fin Mientras

Tiempo Absoluto= TiempoAbsoluto + 1 // Se avanza el instante de tiempo

Fin Mientras

Double TardanzaTotalPonderada = 0

Calcular tardanza total ponderada

Fin <FaseConstructiva>

8.2.1.1. Ejemplo

Para describir el paso a paso del funcionamiento de la fase constructiva, se toma como ejemplo un problema 3x3 tomado de (Kutanoglu & Wu, 1997) con los siguientes datos:

TRABAJO	Peso	Release date	Due Date	$Secuencia\ M_j$ (P_{ij})
J_1	$w_1 = 4$	$r_1 = 0$	$d_1 = 10$	1 (3), 2 (1), 3 (6)
J_2	$w_2 = 6$	$r_2 = 0$	$d_2 = 10$	3 (3), 1(7), 2 (1)
J_3	$w_3 = 2$	$r_3 = 0$	$d_3 = 12$	1 (2), 3 (4), 2(4)

Tabla 2. Datos ejemplo aplicación fase constructiva

Adicionalmente se utilizará como factor "greedy" $\alpha = 0{,}2$

TIEMPO	DESCRIPCIÓN
0	• **Buscar opcionados:** O_{11}, O_{32}, O_{13} • **Se evalúan opcionados:** $G_{ij} = (d_j - (p_{ij} + \tau_j)) \times (1 - (W_j / \sum_{j=1}^{n} W_j))$ G_{11}= 4,66 G_{32}= 3,5 G_{13}= 8,33 • **Se establece rango de la RCL** $\text{Tamaño RCL} = \begin{matrix} \text{Desde} \\ \text{Hasta} \end{matrix} \begin{matrix} Min \\ (Max - Min) \times \alpha \end{matrix}$ $RCL = \begin{matrix} Desde \\ Hasta \end{matrix} \begin{matrix} 3{,}5 \\ 4{,}46 \end{matrix}$ • **Opcionados que entran en RCL:** O_{32} Asignar O_{32} --- $P_{32} = 3$ • **Buscar opcionados:** O_{11}, O_{13} • **Se evalúan opcionados:** G_{11}= 4,66 G_{13}= 8,33 • **Se establece rango de la RCL** $RCL = \begin{matrix} Desde \\ Hasta \end{matrix} \begin{matrix} 4{,}66 \\ 5{,}4 \end{matrix}$ • **Opcionados que entran en RCL:** O_{11} Asignar O_{11} --- $P_{11} = 3$ • **Buscar opcionados:** No se presentan opcionados

1	• **Buscar opcionados:** No se presentan opcionados
2	• **Buscar opcionados:** No se presentan opcionados
3	• **Buscar opcionados:** O_{21}, O_{12}, O_{13} • **Se evalúan opcionados:** G_{21}= 4 G_{12}= 0 G_{13}= 5,83 • **Se establece rango de la RCL** $RCL = \begin{matrix} Desde & 0 \\ Hasta & 1{,}16 \end{matrix}$ • **Opcionados que entran en RCL:** O_{12} Asignar O_{12} --- $P_{12} = 7$ • **Buscar opcionados:** O_{21} • **Se evalúan opcionados:** G_{21}= 4 • **Se establece rango de la RCL** $RCL = \begin{matrix} Desde & 4 \\ Hasta & 4 \end{matrix}$ • **Opcionados que entran en RCL:** O_{21} Asignar O_{21} --- $P_{21} = 1$ • **Buscar opcionados:** No se presentan opcionados
4	• **Buscar opcionados:** O_{31} • **Se evalúan opcionados:** G_{31}= 0 • **Se establece rango de la RCL** $RCL = \begin{matrix} Desde & 0 \\ Hasta & 0 \end{matrix}$ • **Opcionados que entran en RCL:** O_{31} Asignar O_{31} --- $P_{31} = 6$ • **Buscar opcionados:** No se presentan opcionados
5	• **Buscar opcionados:** No se presentan opcionados
6	• **Buscar opcionados:** No se presentan opcionados
7	• **Buscar opcionados:** No se presentan opcionados
8	• **Buscar opcionados:** No se presentan opcionados
9	• **Buscar opcionados:** No se presentan opcionados
10	• **FINALIZA TRABAJO 1 --- TIEMPO TOTAL DE J_1 = 10** • **Buscar opcionados:**

	O_{22}, O_{13} • **Se evalúan opcionados:** G_{22}= -0,5 G_{13}= 0 • **Se establece rango de la RCL** $RCL = \begin{matrix} Desde & -0{,}5 \\ Hasta & -0{,}4 \end{matrix}$ • **Opcionados que entran en RCL:** O_{22} Asignar O_{22} --- $P_{22} = 1$ • **Buscar opcionados:** O_{13} • **Se evalúan opcionados:** G_{13}= 0 • **Se establece rango de la RCL** $RCL = \begin{matrix} Desde & 0 \\ Hasta & 0 \end{matrix}$ • **Opcionados que entran en RCL:** O_{13} Asignar O_{13} --- $P_{13} = 2$ • **Buscar opcionados:** No se presentan opcionados
11	• **FINALIZA TRABAJO 2 --- TIEMPO TOTAL DE J_2 = 11** • **Buscar opcionados:** No se presentan opcionados
12	• **Buscar opcionados:** O_{33} • **Se evalúan opcionados:** G_{33}= -3,33 • **Se establece rango de la RCL** $RCL = \begin{matrix} Desde & -3{,}33 \\ Hasta & -3{,}33 \end{matrix}$ • **Opcionados que entran en RCL:** O_{33} Asignar O_{33} --- $P_{33} = 4$ • **Buscar opcionados:** • No se presentan opcionados
13	• **Buscar opcionados:** No se presentan opcionados
14	• **Buscar opcionados:** No se presentan opcionados
15	• **Buscar opcionados:** • No se presentan opcionados
16	• **Buscar opcionados:** O_{23} • **Se evalúan opcionados:** G_{33}= -6,66 • **Se establece rango de la RCL** $RCL = \begin{matrix} Desde & -6{,}66 \\ Hasta & -6{,}66 \end{matrix}$ • **Opcionados que entran en RCL:**

	O_{23} Asignar O_{33} --- $P_{23} = 4$ • **Buscar opcionados:** No se presentan opcionados
17	• **Buscar opcionados:** No se presentan opcionados
18	• **Buscar opcionados:** • No se presentan opcionados
19	• **Buscar opcionados:** • No se presentan opcionados
20	• **FINALIZA TRABAJO 3 --- TIEMPO TOTAL DE J_3 = 20** • **FINALIZAN TODOS LOS TRABAJOS** • **Calcular Tardanza total ponderada** $$\sum w_i T_i = w_1 T_1 + w_2 T_2 + w_3 T_3$$ $$\sum w_i T_i = 22$$ • Determinar secuencia de máquinas (MaquinasVSTrabajos) ver Tabla 4**.** Se Inicia fase iterativa (Tabu search) con secuencia de máquinas definida

Tabla 3. Descripción funcionamiento algoritmo fase constructiva

8.2.2. Fase de búsqueda local

Para la fase de búsqueda local del GRASP propuesto, se utiliza la búsqueda Tabú como método para encontrar una mejor solución al problema a la hallada en la fase de construcción.

La búsqueda tabú utiliza una estrategia basada en el uso de estructuras de memoria (Acero & Torres, 1992) para así poder salir de los óptimos locales que se encuentran, y dirigir la estrategia a la búsqueda del óptimo global. Estas estructuras de memoria se determinan con la lista tabú, la cual contiene los movimientos tabú (prohibidos) durante un periodo de tiempo determinado.

Para la búsqueda tabú propuesta en el presente trabajo, se utiliza la matriz construida en la fase previa de GRASP "MaquinasVSTrabajos" (Tabla 4). Ésta permite realizar la evaluación de la tardanza total ponderada con un método similar al utilizado en la fase constructiva del problema, mediante la asignación en instantes de tiempo, pero con el uso de la matriz de máquinas trabajos para determinar la secuencia según los requerimientos de las máquinas.

M_1	J_1	J_2	J_3
M_2	J_1	J_2	J_3
M_3	J_2	J_1	J_3

Tabla 4. Ejemplo distribución de trabajos en las máquinas

A continuación se presenta el pseudocódigo para la evaluación de la tardanza total ponderada en la fase iterativa (Ver pseudocódigo Algoritmo 4)

Algoritmo 4

Subprograma <EvaluarTardanza>

Mientras <OperacionesCompletadas **<=** NumOperaciones>
List<Dupla>opcionados = **New** ArrayList<> // En donde se agregarán los opcionados (Dupla de trabajo vs máquina)
Para <i=0> **desde** <i> **hasta** <número de trabajos > **inc** <1>
Si < OperaciónActual del trabajo (i)**!=** NumeroOperaciones del trabajo (i) > // Revisa si el trabajo (i) ya está terminado
Si < RJ del trabajo (i) **<=** TiempoAbsoluto **>** // Revisa si el trabajo (i) ya cumplió con su reléase time
Buscar operación actual del trabajo (i)
Buscar máquina requerida por el trabajo (i)
Si < Tiempo trabajo (i) **==** 0 > // Revisa si el trabajo no está siendo realizado
Si < Tiempo máquina requerida por trabajo (i) **==** 0 > // Revisa si la máquina requerida no está siendo utilizada
Si < El tiempo de procesamiento de la operación a asignar **==** 0 > // Revisa si alguna operación tiene tiempo de procesamiento = a cero, para que se asigne de inmediato
Asignar operación
Operación actual del trabajo (i)= Operación actual del trabajo (i) + 1
OperacionesCompletadas ++
Sino Si <La máquina requerida por el trabajo (i) se encuentra disponible por nivel de dependencias en matriz "MaquinasVSTrabajos">
Asignar Operación
Asignar tiempo de procesamiento al TiemposMaquinas del opcionado (i) // Asigna tiempo de procesamiento para evitar asignar máquina mientras esté trabajando
Asignar el tiempo de procesamiento a TiemposTrabajos del opcionado (i) // Asigna tiempo de procesamiento para evitar asignar trabajo mientras se esté procesando
Fin Si
Fin Si
Fin Si

Fin Si

Fin Si

Si < No hay opcionados && No hay máquinas activas**>** // Casos en que no sea posible una asignación

Devolver Máx_Value // Se devuelve un valor infinito a la tardanza, para que ésta no sea factible.

Fin Si

Tiempo absoluto ++

Para <i=0> **desde** <i> **hasta** <número de trabajos **> inc** <1>

Si < Tiempo trabajo (i) **>** 0 > // Decremento de tiempos de trabajos y máquinas que estén asignados

Tiempo de trabajo (i) = Tiempo de trabajo (i) **-** 1

Tiempo de m que realiza el trabajo (i) = Tiempo de m que realiza el trabajo (i) - 1

Si < Tiempo trabajo (i) **==** 0 > // Aumento de Op completadas, y asignación de la siguiente Op del trabajo (i)

OperaciónActual del trabajo (i) = OperaciónActual del trabajo (i) + 1

OperacionesCompletadas = OperacionesCompletadas + 1

Aumentar precedencia en matriz de máquina utilizada

Fin Si

Si < OperaciónActual del trabajo (i) **=** NumeroOperaciones del trabajo (i) && TiempoEmpleado del trabajo (i) == 0 >

TiempoEmpleado del trabajo (i) **=** TiempoAbsoluto // Almacenar tiempo total de cada trabajo

Fin Si

Fin Si

Fin para

Fin Mientras

Double TardanzaTotalPonderada = 0

Calcular tardanza total ponderada

Fin <EvaluarTardanza>

Para la búsqueda tabú propuesta éste trabajo (Ilustración **8**) se presenta una estrategia de vecindad, basada en el cambio de secuencia de trabajos sobre una máquina de manera aleatoria.

Inicialmente se recibe la secuencia generada por la fase constructiva (Matriz MaquinasVSTrabajos) y la tardanza de la misma. Se manejan 2 resultados en el algoritmo, los cuales se encuentran en la estructura interna del tabú “Tardanza temporal” (secuencia sobre la cual siempre se realizan los cambios aleatorios), y en la estructura externa y final del tabú “TardanzaOptima” (la cual representa la mejor solución encontrada).

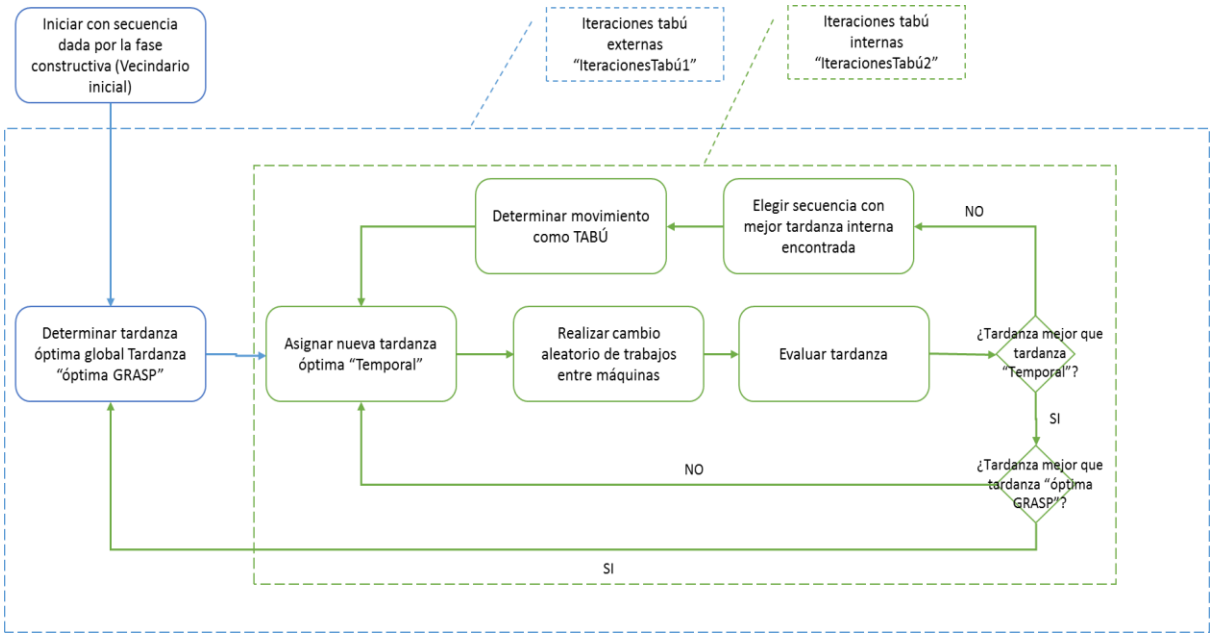

Ilustración 8. Pasos para búsqueda local de Grasp (Fuente: Realizada por el autor)

Como se puede observar en la Ilustración 8 el algoritmo tabú propuesto, realiza un cambio aleatorio en el orden de los trabajos contenidos en una máquina y evalúa la tardanza de esa nueva secuencia. En caso que ésta sea mejor que la tardanza temporal, descrita anteriormente, éste la reemplaza automáticamente y rompe las iteraciones internas (Iteraciones Tabú 2). En caso contrario, si se cumplen las iteraciones internas establecidas, se elige el opcionado que presenta la tardanza "menos peor" evaluada durante las iteraciones, como método para salir del óptimo local encontrado, y encontrar un nuevo vecindario que ofrezca mejores soluciones al problema dado. Luego, se instala dicho cambio en la lista tabú, la cual funciona con la regla FIFO (según el tamaño establecido – cuando se llena, el siguiente elemento se añade, eliminando el primero en entrar), para evitar volver al óptimo local no deseado.

En caso de encontrar una mejor tardanza durante las iteraciones internas, y reemplazarla con la tardanza temporal, ésta se compara con la TardanzaOptima para así, reemplazarla en caso que ésta sea mejor. En este caso las iteraciones externas del programa (Iteraciones tabú 1), se reinician (Hertz & Widmer, 1996) (Salhi, 2002), lo cual direcciona la búsqueda del óptimo global, o de un valor cercano a éste.

En el Algoritmo 5 se presenta el pseudocódigo referente a la búsqueda tabú propuesta.

Algoritmo 5

Programa <Búsqueda tabú>

Int tardanza óptima = Tardanza fase constructiva // La solución óptima se inicializa con la que se encuentra en la fase constructiva

SolucionOptima.setCombinacion (MaquinasVSTrabajos)

Para <a=0> **desde** <a> **hasta** < CantIteracionesTabú1> **inc** <1>

Si <Es la primera iteración**>**

SolucionTemporal **=** SolucionOptima // Se guarda la solución óptima en una "temporal" sobre la cual se realizan las iteraciones

Fin Si

Para <b=0> **desde** <b> **hasta** < CantIteracionesTabú2> **inc** <1>

Int maquina **= 0**

Int Trabajo1 = 0

Int Trabajo2 = 0

Elegir máquina aleatoria

Elegir trabajo 1 aleatorio

Elegir trabajo 2 aleatorio // Tal que sea diferente del trabajo 1

Si <Combinación aleatoria elegida no se encuentra en lista tabú>

Realizar cambio de orden de trabajos en la máquina sobre SolucionTemporal

Subprograma <EvaluarTardanza> // Evaluar tardanza con nueva asignación

Si <el valor de la nueva tardanza total ponderada **<** SolucionTemporal>

SolucionTemporal = NuevaTardanza // Reemplazar solución y asignación

Romper ciclo iteraciones Tabu 2

Si No **Si** <La nueva combinación no es factible>

b - - // No contar como iteración Tabú 2

Si No **Si <**Es la primera iteración**>**

GuardarSobreTabu = NuevaTardanza // Guardar nueva tardanza encontrada, con su asignación MaquinasVSTrabajos

Si No **Si <** Nueva tardanza **<** GuardarSobreTabu **> //** Comparar tardanza guardada en iteración anterior (b-1) con la nueva asignación y dejar la mejor para encontrar tardanza "menos peor"

GuardarSobreTabu = NuevaTardanza // Guardar nueva tardanza encontrada, con su asignación MaquinasVSTrabajos

Fin Si

Fin Si

Fin Si

Fin Si

Fin Si

Fin Para

Si <Se encuentra mejor solución que la solución temporal durante las iteraciones internas>

Si < SolucionTemporal **<** SolucionOptima **>** // En el caso que se encuentre una solución mejor a la mejor solución conocida en el programa hasta el momento

SolucionOptima = SolucionTemporal // Reemplazar con nueva solución y asignación encontrada

a = 0 // Se reinician iteraciones externas Tabú1

Fin Si

Sí No

SolucionTemporal = GuardarSobreTabu // Se reemplaza solución temporal por la solución "menos peor" encontrada durante las iteraciones internas, para buscar salir del óptimo local encontrado

Guardar cambio en Lista tabú // Se guarda el cambio realizado en lista tabú, para evitar volver a dicho óptimo local; Se guarda una tripleta máquina, Trabajo 1 y Trabajo 2

Fin Si

Fin Para

Guardar SolucionOptima // mejor solución encontrada dentro de las iteraciones, para posterior comparación durante las corridas del algoritmo

Fin <Búsqueda tabú>

Para el algoritmo de búsqueda local, se establecen las iteraciones internas (Tabú 2), las iteraciones externas (Tabú 1) y el tamaño de la lista tabú, como parámetros cuyo valor es definido en el diseño de experimentos realizado en el **numeral 8.3.2**.

8.3. Determinación de mejores parámetros para el algoritmo

8.3.1. Parámetros definidos

Como se enunció anteriormente, los parámetros definidos, que se estudiarán a través de diseños experimentales, con el objetivo de determinar la mejor combinación de factores para que se genere un valor igual o cercano al mejor valor conocido para cada una de las instancias. Estos parámetros son:

- Factor "greedy" α
- Número de iteraciones de la búsqueda tabú
- Número de iteraciones de Grasp
- Tamaño de la lista tabú

8.3.2. Diseño de experimentos – Fase de búsqueda local (búsqueda tabú)

Se realiza un diseño de experimentos factorial para determinar la mejor combinación de parámetros para la búsqueda local (Tabú), realizando la evaluación según el índice de mejora de la tardanza total ponderada respecto a la que arroja la fase constructiva.

$$\boldsymbol{Indice\ mejora} = \frac{Tardanza\ fase\ constructiva - Tardanza\ búsqueda\ tabú}{Tardanza\ fase\ constructiva} \quad \textbf{(6)}$$

Se realizan corridas del algoritmo para las instancias de problemas ORB con f: 1.3 los cuales de entre todas las instancias, son las que presentan las tardanzas más altas en su valor óptimo, tal como se lleva a cabo en (Bongarala, 2000)

Los factores considerados son:

- **Iteraciones tabú externas – Tabú 1:** Iteraciones de la búsqueda tabú, que se presentan cada vez que se presenta una nueva solución. En caso que ésta solución sea mejor que la óptima conocida por el algoritmo, se reiniciarán dichas iteraciones (Criterio de parada: Se cumplen dichas iteraciones sin mejorar la solución óptima conocida por el algoritmo). Para éste factor se consideran 3 niveles:

    - 250 (Bajo nivel de búsqueda, bajo tiempo de procesamiento computacional)
    - 500 (Nivel medio de búsqueda, tiempo medio de procesamiento computacional)
    - 1000 (Alto nivel de búsqueda, alto tiempo de procesamiento computacional)

- **Iteraciones tabú internas – Tabú 2:** Iteraciones de la búsqueda tabú, en donde se realizan los cambios aleatorios de la estrategia de vecindad propuesta .El criterio de parada consiste, en el rompimiento de las iteraciones en caso que se encuentre una mejor solución a la tardanza temporal, o en caso que no se encuentre, el criterio de parada será cuando se cumpla el total de iteraciones utilizadas. Para éste factor se consideran 3 niveles (Ver Tabla 5). Estos fueron elegidos debido a que el total de búsquedas internas posibles se totaliza en 450 (suponiendo que ninguna combinación se repite) dado:

  $$\boldsymbol{Total\ combinaciones} = m *_n C_2 \quad \textbf{(7)}$$

  Los niveles del factor a evaluar son elegidos teniendo en cuenta:

    - Nivel de búsqueda: Búsqueda interna de mejores valores, dado la cantidad de movimientos realizados. Entre más alto sea el nivel de búsqueda, aumenta la probabilidad de encontrar un mejor valor de tardanza.
    - Nivel de aleatoriedad: Entre más alto es el nivel de aleatoriedad se da el supuesto que es más fácil que salga del óptimo local en caso que no se encuentren mejores soluciones, debido a que el opcionado “menos peor” elegido será un valor algo alejado del óptimo local.
    - Tiempo de procesamiento computacional: Entre más iteraciones, el tiempo de procesamiento computacional será mayor.

Número de Iteraciones	Nivel de búsqueda	Nivel de aleatoriedad	Tiempo de procesamiento computacional
100	Bajo	Alto	Bajo
200	Medio	Medio	Medio
300	Alto	Bajo	Alto

Tabla 5. Niveles del factor iteraciones tabú internas

- **Lista Tabú:** Éste hace referencia al tamaño de la lista tabú, para la prohibición de movimientos que retornen a óptimos locales. Para este factor se consideran 3 niveles, tomados de autores que determinan dichos tamaños como eficientes durante la implementación de búsqueda tabú para la solución de Job Shop:

    - 7 (Acero & Torres, 1992) para la solución de un problema de programación de tareas en línea flexible de manufactura, y (Bongarala, 2000)
    - $\sqrt{N}$ (Schmidt, 2001): Siendo N el número total de operaciones de la instancia, que para el caso de las instancias a evaluar siempre es 100, por tanto se toma este valor como 10.
    - 30 (Hurink, Jurish, & Thole, 1994)

Posterior a las corridas del algoritmo respectivas para cada uno de los tratamientos, se realiza el diseño de experimentos utilizando la herramienta SPSS. Inicialmente se realiza el ANOVA para determinar los efectos significativos en el mejoramiento dado por la búsqueda tabú respecto a los resultados de la tardanza ponderada arrojados por la fase de construcción (Ver Ilustración 12)

Factores inter-sujetos

		N
TL	7	90
	10	90
	30	90
TABU1	250	90
	500	90
	1000	90
TABU2	100	90
	200	90
	300	90

Ilustración 9. Factores - Niveles Búsqueda tabú

Se realiza la prueba contraste de Levene, para verificar la homogeneidad de varianzas en los tratamientos (Ver Ilustración 10), en la cual se determina que las varianzas no son homogéneas (Significancia menor a 0,05).

Contraste de Levene sobre la igualdad de las varianzas error[a]

Variable dependiente:MEJORA

F	gl1	gl2	Sig.
4,770	26	243	,000

Contrasta la hipótesis nula de que la varianza error de la variable dependiente es igual a lo largo de todos los grupos.

a. Diseño: Intersección + ListaTabu + IteraExternas + IteraInternas + ListaTabu * IteraExternas + ListaTabu * IteraInternas + IteraExternas * IteraInternas + ListaTabu * IteraExternas * IteraInternas

Ilustración 10. Prueba de homogeneidad de varianzas - Búsqueda local

Posteriormente se realiza la prueba de kolmogorov-Smirnov para determinar la normalidad de los datos (Ver Ilustración 11), en donde se encuentra que la distribución de los datos no corresponde a la distribución normal.

Prueba de Kolmogorov-Smirnov para una muestra

		Residuo para IndiceMejora
N		270
Parámetros normales[a,b]	Media	,0000
	Desviación típica	,24941
Diferencias más extremas	Absoluta	,098
	Positiva	,098
	Negativa	-,077
Z de Kolmogorov-Smirnov		1,618
Sig. asintót. (bilateral)		,011

a. La distribución de contraste es la Normal.
b. Se han calculado a partir de los datos.

Ilustración 11. Prueba de Kolmogorov-Smirnov Búsqueda local

Pruebas de los efectos inter-sujetos

Variable dependiente:MEJORA

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	9,000[a]	26	,346	5,027	,000
Intersección	48,543	1	48,543	704,952	,000
ListaTabu	2,884	2	1,442	20,939	,000
IteraExternas	2,188	2	1,094	15,889	,000
IteraInternas	,832	2	,416	6,044	,003
ListaTabu * IteraExternas	1,983	4	,496	7,200	,000
ListaTabu * IteraInternas	,484	4	,121	1,756	,138
IteraExternas * IteraInternas	,290	4	,072	1,052	,381
ListaTabu * IteraExternas * IteraInternas	,339	8	,042	,616	,765
Error	16,733	243	,069		
Total	74,276	270			
Total corregida	25,733	269			

a. R cuadrado = ,350 (R cuadrado corregida = ,280)

Ilustración 12. ANOVA Búsqueda local

Dado que no se cumplen los supuestos de homogeneidad de varianzas ni normalidad de los residuos, es necesario hacer la prueba no paramétrica de Tamhane para determinar los mejores niveles de cada factor, y comprobar la validez de los resultados del ANOVA presentado en la Ilustración 12.

Para el tamaño de la lista tabú (Ver Ilustración 13), se encuentra que el tamaño 30 es significativamente diferente de los tamaños 7 y 10 ($\sqrt{N}$), y la diferencia de medias es positiva, por lo cual éste nivel (30) se considera como el más adecuado. Por otro lado, en lo que respecta a la cantidad de iteraciones externas “Tabu1” (Ver Ilustración 14), las 500 y 1000 iteraciones son significativamente diferentes de 250 iteraciones, y su diferencia de medias es mayor por tanto dichas iteraciones son mejores; Los niveles 500 y 1000 iteraciones no son significativamente diferentes uno de otro por tanto se definen 500 iteraciones como el nivel más adecuado para el algoritmo, teniendo en cuenta la reducción en tiempos de procesamiento que esto supone (Comparado con 1000 iteraciones Tabu1). Finalmente se realiza el análisis de la prueba de Tamhane para el factor referente a las iteraciones internas “Tabu2” (Ver Ilustración 15) el cual arroja como resultado al nivel de 200 iteraciones como el más adecuado para el mejoramiento de la solución de la fase constructiva, con la búsqueda local propuesta.

Comparaciones múltiples

MEJORA
Tamhane

(I) TL	(J) TL	Diferencia de medias (I-J)	Error típico	Sig.	Intervalo de confianza al 95%	
					Límite inferior	Límite superior
7	10	,00938	,02359	,971	-,0475	,0662
	30	-,21439*	,05072	,000	-,3374	-,0914
10	7	-,00938	,02359	,971	-,0662	,0475
	30	-,22377*	,05076	,000	-,3468	-,1007
30	7	,21439*	,05072	,000	,0914	,3374
	10	,22377*	,05076	,000	,1007	,3468

*. La diferencia de medias es significativa al nivel 0.05.

Ilustración 13. Prueba de Tamhane - Lista tabú

Comparaciones múltiples

MEJORA
Tamhane

(I) TABU1	(J) TABU1	Diferencia de medias (I-J)	Error típico	Sig.	Intervalo de confianza al 95%	
					Límite inferior	Límite superior
250	500	-,14008*	,03865	,001	-,2337	-,0465
	1000	-,21753*	,04100	,000	-,3168	-,1182
500	250	,14008*	,03865	,001	,0465	,2337
	1000	-,07746	,05200	,360	-,2028	,0479
1000	250	,21753*	,04100	,000	,1182	,3168
	500	,07746	,05200	,360	-,0479	,2028

*. La diferencia de medias es significativa al nivel 0.05.

Ilustración 14. Prueba de Tamhane - Iteraciones externas Tabú1

Comparaciones múltiples

MEJORA
Tamhane

(I) TABU2	(J) TABU2	Diferencia de medias (I-J)	Error típico	Sig.	Intervalo de confianza al 95%	
					Límite inferior	Límite superior
100	200	-,13501*	,04667	,013	-,2476	-,0224
	300	-,08170	,04160	,146	-,1820	,0186
200	100	,13501*	,04667	,013	,0224	,2476
	300	,05331	,04805	,609	-,0626	,1692
300	100	,08170	,04160	,146	-,0186	,1820
	200	-,05331	,04805	,609	-,1692	,0626

*. La diferencia de medias es significativa al nivel 0.05.

Ilustración 15. Prueba de Tamhane - Iteraciones internas Tabú2

Como se puede observar en las pruebas de Tamhane realizadas, se verifican los resultados del ANOVA en relación a que los tres factores tienen un efecto significativo en el porcentaje de mejoramiento de las tardanzas ponderadas.

Teniendo en cuenta el diseño de experimentos realizado, se determinan como mejores valores para los factores:

- Lista tabú: 7
- Iteraciones Externas – Tabú1: 500
- Iteraciones Internas – Tabú2: 200

Posterior a las pruebas realizadas, se realiza la prueba de Tamhane para ver la mejor combinación de los parámetros lista tabú e iteraciones externas, la cual se muestran en el ANOVA (Ilustración 12) como significativamente diferentes, y el resultado es igual al previamente descrito, dando como mejores combinaciones (i) Lista tabú 30, iteraciones externas 1000, y (ii) Lista tabú 30, iteraciones externas 500, y ambos tratamientos no son significativamente diferentes uno del otro, por tanto se toma el (ii).

8.3.3. Diseño de experimentos – Algoritmo propuesto GRASP

Se realiza un diseño de experimentos para determinar cuál es el mejor valor del parámetro greedy del GRASP de entre tres valores seleccionados. Para ello se utiliza como variable de respuesta el Relative Deviation Index (RDI) (Vallada, Ruiz, & Minella, 2008) que arroja el híbrido GRASP-Tabu search.

$$\boldsymbol{RDI} = \frac{Tardanza\ GRASP - Tardanza\ óptima}{Peor\ tardanza - Tardanza\ óptima} \qquad \textbf{(8)}$$

Esta variable se determina dado que para algunas instancias la tardanza óptima es cero, por lo cual el RPD (Relative percentage index) que compara la diferencia porcentual entre la solución obtenida y la solución óptima, respecto a la solución óptima, no se puede analizar, ya que se producen divisiones sobre 0. En el **numeral 8.3.3.1** se presenta la manera en que se determinaron los peores valores conocidos de las tardanzas para cada una de las instancias, para así poder realizar el experimento.
Se realizan corridas del algoritmo con las mismas instancias con las que se corrió el primer diseño de experimentos del presente documento (Ver numeral 8.3.2), y se halla el RDI para cada una de éstas (5 corridas por tratamiento).
El factor considerado es:

- **Factor greedy α:** Factor de la fase constructiva que determina el tamaño de la RCL dada, y por tanto determina el nivel de aleatoriedad y diversidad del resultado final. Entre más alto es el factor greedy ($0 < \alpha < 1$) (Binato S. , Hery, Loerstern, & Resende, 2009), pueden entrar más opcionados a la RCL lo cual genera mayor aleatoriedad en la construcción, lo cual puede llevar a un mejor vecindario, o por el contrario a una muy mala solución que no pueda ser mejorada eficazmente por la búsqueda local. Los factores greedy evaluados son; (i) 0.1, (ii) 0.2 y (iii) 0.3, dado que se busca llegar a un valor óptimo que permita a la búsqueda tabú hacer su trabajo sin dificultad.

Posterior a las corridas del algoritmo respectivas se realiza el ANOVA en SPSS para determinar si el factor greedy tiene alguna influencia en el RDI.

Factores inter-sujetos

		N
FactorGreedy	,10	10
	,20	10
	,30	10

Ilustración 16. Factores - Niveles Grasp

Se realiza la prueba contraste de Levene, para verificar la homogeneidad de varianzas en los tratamientos (Ver Ilustración 17), en la cual se determina que las varianzas son homogéneas (Significancia mayor a 0,05).

Contraste de Levene sobre la igualdad de las varianzas error[a]

Variable dependiente:IndiceVariación

F	gl1	gl2	Sig.
1,326	2	27	,282

Contrasta la hipótesis nula de que la varianza error de la variable dependiente es igual a lo largo de todos los grupos.

a. Diseño: Intersección + FactorGreedy

Ilustración 17. Prueba de homogeneidad de varianzas - Grasp

Posteriormente se realiza la prueba de kolmogorov-Smirnov para determinar la normalidad de los datos (Ver Ilustración 18), en donde se encuentra que la distribución de los residuos corresponde a la distribución normal.

Prueba de Kolmogorov-Smirnov para una muestra

		Residuo para IndiceVariación
N		30
Parámetros normales[a,b]	Media	,0000
	Desviación típica	,04499
Diferencias más extremas	Absoluta	,107
	Positiva	,104
	Negativa	-,107
Z de Kolmogorov-Smirnov		,584
Sig. asintót. (bilateral)		,884

a. La distribución de contraste es la Normal.
b. Se han calculado a partir de los datos.

Ilustración 18. . Prueba de Kolmogorov-Smirnov Grasp

Ya comprobados los supuestos de homogeneidad de varianzas y de normalidad de los residuos, se puede observar en el ANOVA (Ver Ilustración 19), que para los tres niveles seleccionados de dicho parámetro, no se presenta un efecto significativo sobre el RDI. Esto en parte debe verse influenciado porque las instancias aunque sean del mismo tamaño, tienen características diferentes. En todo caso las comparaciones a ser realizadas contra

el algoritmo GLS presentado por (Essafi, Mati, & Dauzère-Pérès, 2008) se harán con los resultados dados por el valor de $\alpha = 0{,}1$ como valor del parámetro, teniendo en cuenta que descriptivamente entre los tres seleccionados, tiene el valor promedio de RDI (Ver Tabla 6)

Pruebas de los efectos inter-sujetos

Variable dependiente:IndiceVariación

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	,003[a]	2	,002	,773	,472
Intersección	,462	1	,462	212,594	,000
FactorGreedy	,003	2	,002	,773	,472
Error	,059	27	,002		
Total	,524	30			
Total corregida	,062	29			

a. R cuadrado = ,054 (R cuadrado corregida = -,016)

Ilustración 19. ANOVA Grasp

	0,1	***0,2***	***0,3***
Media	0,053	0,069	0,061
Error típico	0,009	0,015	0,009
Mediana	0,060	0,073	0,058
Desviación estándar	0,028	0,047	0,029
Varianza de la muestra	0,001	0,002	0,001
Mínimo	0,009	0,006	0,024
Máximo	0,095	0,165	0,112
Cuenta	10	10	10

Tabla 6. Estadística descriptiva factor α

8.3.3.1. Determinación peores tardanzas

El valor de la peor tardanza conocida para cada una de las instancias benchmark seleccionadas, no se encuentra consignado en la literatura referente a las mismas; por tanto, con el fin de determinar el valor del RDI y realizar un correcto análisis del desempeño del algoritmo propuesto, se realiza un código para determinar dichos valores (Peor tardanza conocida) para cada una de las instancias.

El código realizado consiste en la búsqueda del peor candidato a ser asignado teniendo en cuenta la misma fórmula utilizada en la fase constructiva, para determinar las operaciones a asignar, con la diferencia que en este caso se elige y asigna automáticamente la operación que presenta el peor valor en la función (Ver Ilustración 20). En el Algoritmo 6 se presenta el pseudocódigo para la búsqueda de la peor solución conocida

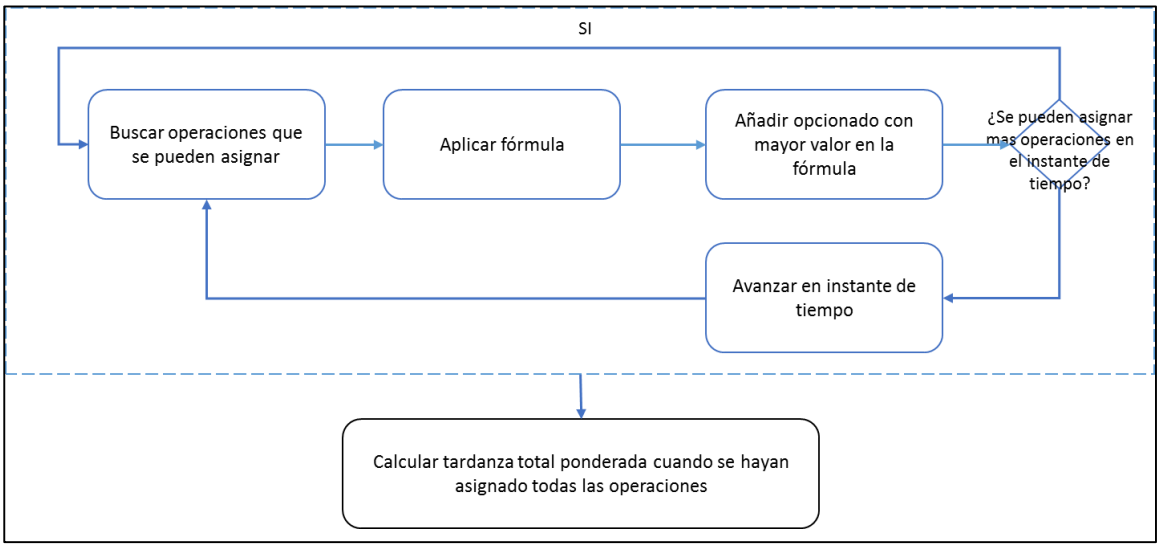

Ilustración 20. Proceso búsqueda de peor solución (Fuente: Realizada por el autor)

Algoritmo 6

Programa <PeorTardanza>

Recolección de datos // Se recogen los datos de un archivo plano, y se solicita α, el número de iteraciones de Grasp y de la fase iterativa (Tabú) y el tamaño de la lista tabú

Mientras <OperacionesCompletadas **<=** NumOperaciones>

Para <i=0> **desde** <i> **hasta** <número de trabajos **> inc** <1>

Si < Tiempo trabajo (i) **>** 0 > // Decremento de tiempos de trabajos y máquinas que estén asignados

Decremento de tiempos de trabajos y máquinas que estén asignados

Si < Tiempo trabajo (i) **==** 0 > // Aumento de Op completadas, y asignación de la siguiente Op del trabajo (i)

Aumento de Op completadas, y asignación de la siguiente Op del trabajo (i)

Fin Si

Si < OperaciónActual del trabajo (i) **=** NumeroOperaciones del trabajo (i) && TiempoEmpleado del trabajo (i) == 0 >

TiempoEmpleado del trabajo (i) **=** TiempoAbsoluto // Almacenar tiempo total de cada trabajo

Fin Si

Fin Si

Fin para

Mientras < True >

Para <i=0> **desde** <i> **hasta** <número de trabajos > **inc** <1>

Buscar opcionados que se puedan asignar

Fin Para

Si < Contador **>=** Número de trabajos >

Break (Terminar ciclo While)

Fin Si

Para <i=0> **desde** <i> **hasta** <número de opcionados > **inc** <1> //Ordenación Algoritmo Burbuja

Double actual = <<Aplicar fórmula>> // Se aplica fórmula para el candidato. Se usa variable tipo Double debido a que es un número real

Elegir opcionado que presenta mayor valor en la formula

Fin Para

Añadir opcionado a la solución

Asignar tiempo de procesamiento al TiemposMaquinas del opcionado (i) // Asigna tiempo de procesamiento para evitar asignar máquina mientras esté trabajando

Asignar el tiempo de procesamiento a TiemposTrabajos del opcionado (i) // Asigna tiempo de procesamiento para evitar asignar trabajo mientras se esté procesando

Añadir opcionado en MaquinasVSTrabajos // Añadir trabajo asignado a la lista de la maquina asignada

Fin Mientras

Tiempo Absoluto= TiempoAbsoluto + 1 // Se avanza el instante de tiempo

Fin Mientras

Double TardanzaTotalPonderada = 0

Calcular tardanza total ponderada

Fin <PeorTardanza>

9. Análisis de resultados

El algoritmo propuesto, fue corrido 5 veces para cada una de las instancias, con los parámetros previamente definidos (ver Tabla 7) en un computador con procesador Intel® CORE™ i5-4200U CPU 1.60GHz 2.30GHz con memoria RAM de 4,00 GB (3,72 GB utilizable). Para cada una de las instancias, se determina el promedio de las tardanzas ponderadas obtenidas (Para analizar la efectividad general del algoritmo por instancia), la mejor solución obtenida, y el RDI para cada uno de estos resultados, así como el porcentaje de soluciones óptimas obtenidas por instancia.

Número iteraciones GRASP	500
Número de iteraciones Externas Tabú	500
Número de iteraciones Internas Tabú	200
Factor α	0.1
Tamaño lista Tabú	30
Estrategia de vecindad	Intercambio de trabajos aleatorios en una máquina aleatoria por iteración

Tabla 7. Resumen parámetros definidos

Los tiempos computacionales corresponden a un promedio de 241 minutos por corrida para cada una de las instancias benchmark

	TOTAL (Seg)	Total (min)
Tiempo promedio	14501	241,7
Desv Estandar	5936	98,9

Tabla 8. Resumen tiempos computacionales

Los resultados obtenidos tras realizar la implementación de el algoritmo propuesto en las instancias benchmark seleccionadas es muy bueno, debido a que en promedio el RDI para la mejor solución obtenida por instancia es 1,65%, y para el promedio de las 5 corridas por instancia es de 2,64%, lo cual muestra que los resultados obtenidos, si bien, no corresponden todos al mejor valor conocido, entregan una muy buena secuencia para programar la producción y reducir la tardanza total ponderada a un valor muy cercano al óptimo y lejano a la peor programación conocida posible.

Adicionalmente se encuentran 21 óptimos conocidos de las 66 instancias corridas, lo cual corresponde al 32% de las mismas, teniendo en cuenta que el resto de las soluciones, no se encuentran muy alejadas del mismo. El peor RDI encontrado para las instancias seleccionadas es de 7,79% correspondiente al problema ORB1_1.6, y en lo que respecta

al promedio de las 5 corridas, el peor índice encontrado es de 13,329% correspondiente al ORB1_1.3.

En las Tabla 9, Tabla 10 y Tabla 11, se presentan los resultados del algoritmo propuesto, y la comparación con el GLS de (Essafi, Mati, & Dauzère-Pérès, 2008) . Los valores que se presentan como un asterisco (*) representan valores en que se encontró el mejor valor conocido, y los que presentan el valor junto con el asterisco, representan que se encontró un mejor valor al valor previo conocido (antes de desarrollar el algoritmo). Para éste último caso, el algoritmo GLS presenta 11 nuevos mejores valores, 2 de los cuales son encontrados también por el algoritmo GRASP propuesto.

Si bien, los resultados no son tan buenos como el GLS, en el algoritmo híbrido propuesto, se presentan muy buenos resultados en donde el 91% de los mismos presentan un valor de gap (RDI) menor al 5%, teniendo en cuenta los mejores resultados obtenidos para cada una de las instancias, y solo el para el 6% de las instancias se presenta un gap mayor al 10% (Valor máximo de 13,3%)

F= 1,3										
Instancias	Mejor conocido (Optimo)	Peor Sol.	GLS			HIBRIDO ENTRE GRASP Y TABÚ				
			Media	Mejor	% Opt	Media	Mejor	% Opt	RDI - Media	RDI - Mejor
ABZ5	1403	4712	*	*	100%	1489	1438	0%	2,611%	1,058%
ABZ6	436	2213	*	*	100%	529	499	0%	5,222%	3,545%
LA16	1169	5627	1175	**1169***	90%	1237	**1169***	20%	1,521%	0%
LA17	899	4095	*	**899***	100%	1090	912	0%	5,970%	0,407%
LA18	929	4183	933	*	60%	952	936	0%	0,713%	0,215%
LA19	948	3599	949	*	80%	1074	956	0%	4,745%	0,302%
LA20	805	4799	805	**805***	100%	884	855	0%	1,988%	1,252%
LA21	463	4362	*	**463***	100%	477	**463***	60%	0,359%	0,000%
LA22	1064	3617	1087	**1064***	10%	1212	1180	0%	5,797%	4,544%
LA23	835	2398	865	**835***	20%	916	873	0%	5,157%	2,431%
LA24	835	4236	*	*	100%	839	*	80%	0,129%	0%
MT10	1363	7774	1372	**1363***	90%	1628	1524	0%	4,140%	2,511%
ORB1	2568	7398	2651	2570	0%	3212	2929	0%	13,329%	7,474%
ORB2	1408	3492	1444	**1408***	20%	1544	1503	0%	6,516%	4,559%
ORB3	2111	9766	2170	**2111***	40%	2531	2431	0%	5,490%	4,180%
ORB4	1623	7026	1643	*	70%	1711	1653	0%	1,629%	0,555%
ORB5	1593	5076	1659	*	10%	1832	1790	0%	6,848%	5,656%
ORB6	1790	10514	*	**1790***	100%	1907	1806	0%	1,338%	0,183%
ORB7	590	1868	592	*	90%	609	*	80%	1,455%	0%
ORB8	2429	7053	2522	2439	0%	2772	2754	0%	7,412%	7,029%
ORB9	1316	8793	*	*	100%	1558	1430	0%	3,240%	1,525%
ORB10	1679	5127	1718	*	50%	2059	1946	0%	11,027%	7,744%

Tabla 9. Resultados algoritmo propuesto - comparación para instancias con f=1,3

F= 1,5										
Instancias	Mejor conocido (Optimo)	Peor Sol.	GLS			HIBRIDO ENTRE GRASP Y TABÚ				
			Media	Mejor	% Opt	Media	Mejor	% Opt	RDI - Media	RDI - Mejor
ABZ5	69	2794	*	*	100%	115	114	0%	1,695%	1,651%
ABZ6	(0)	1688	*	*	100%	*	*	100%	0%	0%
LA16	166	4179	*	*	100%	172	*	60%	0,150%	0%
LA17	260	2635	*	*	100%	*	*	100%	0,000%	0%
LA18	34	2730	*	*	100%	94	79	0%	2,218%	1,669%
LA19	21	1867	*	*	100%	43	27	0%	1,213%	0,325%
LA20	(0)	3094	*	*	100%	2	1	0%	0,058%	0,032%
LA21	(0)	2974	*	*	100%	9	5	0%	0,296%	0,168%
LA22	196	1961	*	*	100%	218	*	60%	1,258%	0%
LA23	2	1583	*	*	100%	*	*	100%	0%	0%
LA24	82	2803	86	*	30%	88	86	0%	0,206%	0,147%
MT10	394	5926	*	*	100%	485	475	0%	1,645%	1,464%
ORB1	1098	5311	1159	*	60%	1544	1403	0%	10,586%	7,239%
ORB2	292	2237	*	*	100%	381	344	0%	4,586%	2,674%
ORB3	918	7548	943	*	40%	1042	1004	0%	1,867%	1,297%
ORB4	358	5240	394	*	80%	544	522	0%	3,818%	3,359%
ORB5	405	3346	*	*	100%	560	524	0%	5,257%	4,046%
ORB6	426	8322	440	*	50%	540	518	0%	1,446%	1,165%
ORB7	50	988	55	*	80%	56	54	0%	0,682%	0,426%
ORB8	1023	5202	1059	*	70%	1227	1213	0%	4,877%	4,547%
ORB9	297	6860	311	*	70%	353	330	0%	0,847%	0,503%
ORB10	346	3590	400	*	40%	542	442	0%	6,030%	2,959%

Tabla 10. Resultados algoritmo propuesto - comparación para instancias con f=1,5

F= 1,6										
Instancias	Mejor conocido (Optimo)	Peor Sol.	GLS			HIBRIDO ENTRE GRASP Y TABÚ				
			Media	Mejor	% Opt	Media	Mejor	% Opt	RDI - Media	RDI - Mejor
ABZ5	(0)	1978	*	*	100%	*	*	100%	0%	0%
ABZ6	(0)	1138	*	*	100%	*	*	100%	0%	0%
LA16	(0)	3547	*	*	100%	*	*	100%	0%	0%
LA17	65	2104	*	*	100%	*	*	100%	0%	0%
LA18	(0)	2126	*	*	100%	*	*	100%	0%	0%
LA19	(0)	1343	*	*	100%	*	*	100%	0%	0%
LA20	(0)	2417	*	*	100%	*	*	100%	0%	0%
LA21	(0)	2456	*	*	100%	*	*	100%	0%	0%
LA22	(0)	1492	*	*	100%	*	*	100%	0%	0%
LA23	(0)	1019	*	*	100%	*	*	100%	0%	0%
LA24	(0)	2089	*	*	100%	*	*	100%	0%	0%
MT10	141	6246	162	*	10%	181	176	0%	0,652%	0,573%
ORB1	566	4516	688	604	0%	997	874	0%	10,911%	7,797%
ORB2	44	1860	*	*	100%	60	52	0%	0,881%	0,441%
ORB3	422	6661	514	*	10%	625	615	0%	3,247%	3,093%
ORB4	66	4428	78	*	80%	105	84	0%	0,899%	0,413%
ORB5	163	2818	181	181	0%	229	211	0%	2,501%	1,808%
ORB6	28	7344	*	**28***	100%	59	38	0%	0,424%	0,137%
ORB7	(0)	678	*	*	100%	0	*	100%	0%	0%
ORB8	621	4511	669	646	0%	772	684	0%	3,892%	1,620%
ORB9	66	6044	83	*	70%	120	84	0%	0,897%	0,301%
ORB10	76	3017	142	84	0%	204	195	0%	4,366%	4,046%

Tabla 11. Resultados algoritmo propuesto - comparación para instancias con f=1,6

10. Conclusiones y recomendaciones

Un algoritmo híbrido entre la meta heurística GRASP y la búsqueda tabú como búsqueda local, se propone para resolver el problema de programación de la producción en un ambiente Job Shop para la minimización de la tardanza total ponderada. El trabajo se enfoca en la determinación de parámetros óptimos para el algoritmo propuesto para resolver las instancias seleccionadas mediante la realización de un diseño de experimentos.

De acuerdo a los resultados obtenidos tras la implementación del algoritmo propuesto en las instancias benchmark seleccionadas, se concluye que aunque no mejora la calidad de las soluciones obtenidas por la búsqueda local genética propuesta por (Essafi, Mati, & Dauzère-Pérès, 2008), éste representa un método eficiente (en términos de resultados) como meta heurística para la minimización de la tardanza total ponderada de acuerdo al bajo RDI presentado en la mayor parte de las instancias.

Se propone realizar estudios que mejoren el algoritmo híbrido propuesto, tales como el uso de diferentes estrategias de vecindario en la búsqueda local, la determinación de una función de utilidad diferente (más especializada) para la fase constructiva, el uso de estrategias para evitar el estancamiento en óptimos locales tales como la modificación del tamaño de la lista tabú a medida que se encuentran buenas o malas soluciones (Schmidt, 2001), y el uso de métodos más eficientes en la programación que produzcan una reducción significativa en los tiempos de procesamiento. Se recomienda aplicar el algoritmo propuesto para otras funciones objetivo del JSSP tales como la minimización del makespan, para examinar sus resultados y eficiencia del mismo.

Título	Autor	Año	Complejidad					F. Objetivo	Método de Solución	Perspectivas de Investigación
			prmp	SDST	N° Máquinas	N° Trabajos	Otros			
A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem	Imen Essafia,*, Yazid Matib,1, Stéphane Dauzère-Pérèsc	2008	No	No	10	10	Release Dates conocidos Due Dates Conocidos	Minimizar Tardanza total ponderada	Genetic Local Search	° Usar una medida de distancia entre los individuos obtenidos por la búsqueda local para así descartar individuos muy similares. Usar algún tipo de probabilidad para la aplicación de la búsqueda local y seleccionar los individuos a ser mejorados por el operador de la búsqueda local
A genetic algorithm for the Flexible Job-shop Scheduling Problem	F. Pezzellaa, G. Morganti, G. Ciaschetti	2008	No	No	Benchmark problems (m)	Benchmark problems (n)	Flexible Job Shop	Minimizar Makespan	Algoritmos Genéticos	Mejorar el operador de mutación mediante la elección de una operación crítica, en vez de cualquier operación, en la máquina mas crítica.
A memetic algorithm for the job-shop with time-lags	Anthony Caumond Philippe Lacomme, Nikolay Tchernev	2008	No	No	Benchmark problems (m)	Benchmark problems (n)	Retrasos de Tiempo (Time Lags)	Minimizar Makespan	Memetic Algorithm (MA)	Uso de algoritmos exactos para problemas de larga escala, tiempo de transporte para as máquinas
Clonal Selection Based Memetic Algorithm for Job Shop Scheduling Problems	Jin-hui Yang, Liang Sun, Heow Pueh Lee, Yun Qian, Yan-chun Liang	2008	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	Memetic Algorithm (MA) con SA como búsqueda local	Aplicar el algoritmo a problemas de manufactura mas prácticos e integrados , como la inclusión de llegadas dinámicas, daños en las máquinas, entre otros factores

Anexo 1. Antecedentes JSSP

Título	Autor	Año	Complejidad					F. Objetivo	Método de Solución	Perspectivas de Investigación
			prmp	SDST	N° Máquinas	N° Trabajos	Otros			
Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound	Christian Artiguesa, MichelGendreau, Louis-MartinRousseau, AdrienVergnaud	2009	No	No	m	n	Adicional al problema Job Shop se propone un problema involucrando a los empleados (Programación de Timetable de los empleados)	Multiobjetivo. Minimizar Makespan y minimizar costo total de los empleados	Hibrido entre B&B y búsqueda local	N.A.
A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times	Davide Anghinolfi, Massimo Paolucci	2009	No	Si	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Tardanza total ponderada	Discrete Particle Swarm Optimization (DPSO)	N.A.
A filter and Fan approach to the job shop scheduling problem: A preliminary study	Renato Duarte, Cesar Rego, Dorabela Gamboa	2009	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	Híbrido entre SBP y método "Filter and Fan" (F&F)	La búsqueda tabú puede ser de gran ayuda para diversificar la búsqueda y para identificar buenos movimientos potenciales determinados por el uso de la memoria adaptiva
An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem	Guohui Zhang, Xinyu Shao, Peigen Li, Liang Gao	2009	No	No	10	10, 15	Flexible Job Shop	Multiobjetivo. Minimizar Makespan, Minimizar el tiempo máximo de trabajo en las máquinas, y minimizar la carga de trabajo total de todas las máquinas	Híbrido entre PSO y TS	Aplicar PSO en otras extensiones del problema

Anexo 1. Antecedentes JSSP (Continuación 1)

Título	Autor	Año	Complejidad					F. Objetivo	Método de Solución	Perspectivas de Investigación
			pr mp	SDST	N° Máquinas	N° Trabajos	Otros			
A multi-modal immune algorithm for the job-shop scheduling problem	Guan-Chun Luh, Chung-Huei Chueh	2009	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	Inmune algorithm (IA)	N.A.
A hybrid immune simulated annealing algorithm for the job shop scheduling problem	Rui Zhang, Cheng Wu	2010	No	No	m	n	N.A	Minimizar Tardanza total ponderada	Híbrido basado en algoritmo Inmune (IA) y Recocido Simulado (SA)	Desarrollar otros mecanismos basados en inmunidad para combinación con otras métodos de búsqueda local
An agent-based parallel approach for the job shop scheduling problem with genetic algorithms	Leila Asadzadeh, Kamran Zamanifar	2010	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	Algoritmos Genéticos Paralelos	Mejorar el rendimiento de el método y aplicarlo a problemas Similares
An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem	L. De Giovanni, F. Pezzella	2010	No	No	Benchmark problems (m)	Benchmark problems (n)	Flexible Job Shop	Minimizar Makespan	Algoritmos Genéticos	Definición de un mejor LB para el problema
An effective heuristic for flexible job-shop scheduling problem with maintenance activities	Shijin Wanga, Jianbo Yu	2010	No	No	Benchmark problems (m)	Benchmark problems (n)	Flexible Job Shop, Mantenimiento preventivo	Multiobjetivo. Minimizar Makespan, Minimizar el tiempo máximo de trabajo en las máquinas, y minimizar la carga de trabajo total de todas las máquinas	Beam Search	Estudiar el problema en donde las actividades de mantenimiento sean dadas como un input del problema, no como un resultado de la operación de las máquinas
A multi-objective PSO for job-shop scheduling problems	D.Y. Sha, Hsing-Hung Lin b	2010	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Multiobjetivo. Minimizar makespan, minimizar tardanza total, y minimizar el tiempo de inactividad total	Particle Swarm Optimization (PSO)	Temas relacionados a la optimización por pareto, como por ejemplo estrategias de mantenimiento de la solución y medición de la gestión

Anexo 1. Antecedentes JSSP (Continuación 2)

Título	Autor	Año	Complejidad					F. Objetivo	Método de Solución	Perspectivas de Investigación
			prmp	SDST	N° Máquinas	N° Trabajos	Otros			
A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardinessobjective	Rui Zhang, Cheng Wu	2011	No	No	m	n	N.A	Minimizar Tardanza total ponderada	Recocido Simulado Basado en propiedades de Bloques	Aplicar el vecindario basado en Bloques, en otros algoritmos de búsqueda local
Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem	E. Moradi, S.M.T. Fatemi Ghomi, M. Zandieh	2011	No	No	Benchmark problems (m)	Benchmark problems (n)	Flexible Job Shop, Mantenimiento preventivo	Multiobjetivo. Minimizar Makespan, y minimizar falta de disponibilidad para el mantenimiento de las máquinas	NSGA (non dominated sort genetic algorithm), y NRGA (non ranking genetic algorithm).	Aplicar otras políticas de mantenimiento al problema
An adaptive annealing genetic algorithm for the job-shop planning and scheduling problem	Min Liu, Zhi-jiang Sun, Jun-wei Yan, Jing-song Kang	2011	No	No	10	30, 50	Lotes pequeños de producción	Minimizar Makespan	Híbrido entre GA y SA	Determinar la mejor combinación de parámetros
Identifying and exploiting commonalities for the job-shop scheduling problem	Marnix Kammer,Marjan van den Akker,Han Hoogeveen	2011	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	Recocido Simulado SA	N.A.
A general approach for optimizing regular criteria in the job-shop scheduling problem	Yazid Mati, Stèphane Dauzère-Pérès, Chams Lahlou	2011	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Tardanza total ponderada	Búsqueda local	Aplicar el método a Flexible Job Shop
An efficient memetic algorithm for solving the job shop scheduling problem	Liang Gao, Guohui Zhang , Liping Zhang, Xinyu Li	2011	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	Memetic Algorithm (MA)	Usar MA para resolver otros problemas de manufactura tales como flexible Job shop

Anexo 1. Antecedentes JSSP (Continuación 3)

Título	Autor	Año	Complejidad					F. Objetivo	Método de Solución	Perspectivas de Investigación
			prmp	SDST	N° Máquinas	N° Trabajos	Otros			
Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm	Rubiyah Yusof, Marzuki Khalid, Gan Teck Hui, Syafawati Md Yusof, Mohd Fauzi Othman	2011	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	GA Híbrido. Asynchronous Colony GA y Autonomous Inmigration GA	Se propone investigar mas en tres áreas principales: Escabilidad, robustez y Adaptabilidad
A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem	R. Tavakkoli-Moghaddam, M. Azarkish, A. Sadeghnejad-Barkousaraie	2011	No	Si	m	n	Tiempos de alistamiento dependientes de secuencia	Multiobjetivo. Minimizar tiempo de flujo total ponderado y minimizar penalidades totales por tardanza y terminar los trabajos muy temprano (Earliness)	Híbrido entre PSO y VNS (Variable Neighbourhood Search)	N.A.
A Hybrid Algorithm for Flexible Job-shop Scheduling Problem	Jianchao Tanga, Guoji Zhanga, Binbin Lina, Bixi Zhang	2011	No	No	Benchmark problems (m)	Benchmark problems (n)	Flexible Job shop	Minimizar Makespan	Híbrido entre PSO y GA	N.A.
A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search	Ghasem Moslehi,Mehdi Mahnam	2011	No	No	m	n	Flexible Job shop	Multiobjetivo. Minimizar makespan y Minimizar el tiempo máximo de trabajo en las máquinas	Híbrido entre PSO y Búsqueda local	N.A.
An effective TPA-based algorithm for job-shop scheduling problem	Ye Li, Yan Chen	2011	No	No	6 a 10	6 a 20	N.A.	Minimizar Makespan	Algoritmo de proceso de equipo TPA	N.A.

Anexo1. Antecedentes JSSP (Continuación 4)

Título	Autor	Año	Complejidad					F. Objetivo	Método de Solución	Perspectivas de Investigación
			prmp	SDST	N° Máquinas	N° Trabajos	Otros			
A hybrid local search algorithm for scheduling real-world job shops with batch-wise pending due dates	Rui Zhang, Cheng Wu	2012	No	No	m	n*(N° lotes)	Existencia de Due dates pendientes, y producción por lotes	Minimizar Tardanza total ponderada	Híbrido basado en PMBGA (probabilistic model-building genetic algorithm), y algoritmo basado en la perturbación de parámetros (PP)	Incorporar mas factores de la vida real al problema, por ejemplo el efecto de eventos aleatorios en la línea de producción
A branch and bound algorithm for the cyclic job-shop problem with transportation	Peter Brucker, Edmund K. Burke, Sven Groenmeyer	2012	No	No	5 a 15	5 a 15	Job Shop Cíclico, Tiempos de transporte	Minimizar Tiempo de ciclo	B & B	Cambiar la estrategia de búsqueda a una mezcla balanceada entre encontrar soluciones factibles y mejores LB
Solving the job-shop scheduling problem optimally by dynamic programming	Joaquim A.S.Gromicho, JelkeJ.vanHoorn, Francisco Saldanha-da-Gama, GerritT.Timmer	2012	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	Dinamic Programming (DP)	N.A.
Modified Genetic Algorithm for Flexible Job-Shop Scheduling Problems	Wannaporn Teekeng, Arit Thammano	2012	No	No	Benchmark problems (m)	Benchmark problems (n)	Flexible Job Shop	Minimizar Makespan	Algoritmos Genéticos	Trabajar en método en multiobjetivo
Minimizing makespan for scheduling stochastic job shop with random breakdown	De-ming Lei	2012	No	No	Benchmark problems (m)	Benchmark problems (n)	Job shop Estocástico, con daños en las máquinas aleatorios	Minimizar Makespan	Algoritmos Genéticos	N.A.

Anexo1. Antecedentes JSSP (Continuación 5)

Título	Autor	Año	Complejidad					F. Objetivo	Método de Solución	Perspectivas de Investigación
			prmp	SDST	N° Máquinas	N° Trabajos	Otros			
A two-stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem	Rui Zhang, Shiji Song, Cheng Wu	2012	No	No	m	n	Job Shop estocástico, con tiempos de procesamiento variables	Minimizar Tardanza total ponderada	PSO	Considerar otro tipo de aleatoriedad, por ejemplo, el desconocimiento de rutas de procesamiento de ciertos trabajos
Inventory based two-objective job shop scheduling model and its hybrid genetic algorithm	Ren Qing-Dao-er-ji, Yuping Wang, Xiaoli Wang	2013	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A	Multiobjetivo. Minimizar Makespan, y minimizar capacidad de inventario	GA Híbrido	N.A.
An optimal method for the preemptive job shop scheduling problem	Abbas Ebadi, Ghasem Moslehi	2013	Si	No	Benchmark problems (m)	Benchmark problems (n)	N.A	Minimizar Makespan	B & B	El desarrollo de otras reglas de Priorización, LB, y otras técnicas, pueden mejorar el poder del algoritmo
A hybrid artificial bee colony algorithm for the job shop scheduling problem	Rui Zhang, Cheng Wu, Shiji Song	2013	No	No	m	n	N.A	Minimizar Tardanza total ponderada	Artificial Bee colony algorithm (ABC)	Considerar otros sistemas de codificación y mecanismos de búsqueda local para el algoritmo ABC
A hybrid Differential Evolution—Tabu Search algorithm for the solution of Job-Shop Scheduling Problems	Rui Zhang, ShijiSong, ChengWu	2013	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A	Minimizar Tardanza total ponderada	Hibrido entre Evolución diferencial (DE), Y Búsqueda tabú (TS)	Adaptar el procedimiento interno de DE para algoritmos basados en la permutación, tal y como se propone en Evolución diferencial Geométrica

Anexo1. Antecedentes JSSP (Continuación 6)

Título	Autor	Año	Complejidad					F. Objetivo	Método de Solución	Perspectivas de Investigación
			prmp	SDST	N° Máquinas	N° Trabajos	Otros			
A hybrid genetic algorithm for the job shop scheduling problem with practical considerations for manufacturing costs: Investigations motivated by vehicle production	Rui Zhang, Pei-ChannChang, ChengWu	2013	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A	Multiobjetivo. Minimizar Tardanza total ponderada, la cercanía en tipo de modelo para la secuencia de la máquina K1, y la cercanía en tipo de color para la secuencia de máquina K2	Hibrido entre GA y un LS	Investigar objetivos mas prácticos y mas aplicados a la industria
A GRASP x ELS approach for the job-shop with a web service paradigm packaging	Maxime Chassaing, Jonathan Fontanel, Philippe Lacomme, Libo Ren, Nikolay Tchernev, Pierre Villechenon	2013	No	No	Benchmark problems (m)	Benchmark problems (n)	N.A.	Minimizar Makespan	GRASP con una potente búsqueda local	N.A.
A flexible dispatching rule for minimizing tardiness in job shop scheduling	Binchao Chen,Timothy I. Matis	2013	No	No	6	4	N.A.	Minimizar Tardanza total ponderada	Regla de despacho llamada Weight Biased Modified Rule	Aplicar otras reglas entre algunas de las características del sistema

Anexo1. Antecedentes JSSP (Continuación 7)

Bibliografía

A.S.Gromicho, J., Hoorn, J. J., Saldanha-da-Gama, F., & Timmer, G. T. (2012). Solving the job-shop scheduling problem optimally. *Computers & Operations Research*, 2968–2977.

Acero, R. B., & Torres, J. F. (1992). Aplicación de una heurística de búsqueda tabú en un problema de programación de tareas en linea flexible de manufactura.

Adams, J., Balas, E, & Zawack, D. (1988). *The Shifting Bottleneck Procedure for Job-Shop Scheduling, Management Science.*

Aiex, R. (2003). Parallel GRASP with path-relinking for job shop scheduling. *Parallel computing in numerical optimization*, 393–430.

Anghinolfi, D., & Paolucci, M. (2009). A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 73–85.

Artigues, C., Gendreau, M., Rousseau, L.-M., & Vergnaud, A. (2009). Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research*, 2330--2340.

Asadzadeh, L., & Zamanifar, K. (2010). An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematical and Computer Modelling*, 1957-1965.

Balas, E. (1969). Machine Scheduling via Disjunctive Graphs: An Implicit Enumeration Algorithm. *Operations Research*, 941-957.

Beenhakker, H. L. (1963). *Development of alternate criteria for optimality in the machine sequencing problem.* Thesis, Purdue University.

Binato, S., Hery, W. J., & Resende, M. G. (2002). A Grasp for Job Shop Scheduling. *Operations Research/Computer Science Interfaces Series, 15*, 59-79.

Binato, S., Hery, W. J., Loerstern, D., & Resende, M. (2009). A GRASP for job shop shceduling. *Encyclopedia of optimization*.

Binato, S., Hery, W., Loewenstern, D. M., & Resende, M. G. (2000). A GRASP For Job Shop Scheduling . *Essays and Surveys on Metaheuristics*.

Bongarala, S. R. (2000). *Minimization of Weighted Tardinness in Job Shops Using Shifting Bottleneck and Tabu Search Procedures.* Montreal.

Brooks, G. H., & White, C. R. (1965). An algorithm for finding optimal or near optimal solutions to the production scheduling problem. *Journal of Industrial Engineering*.

Brucker, P., EdmundK.Burke, & SvenGroenemeyer. (2012). A branch and bound algorithm for the cyclic job-shop problem. *Computers & Operations Research*, 3200–3214.

Brucker, P., Jurisch, B., & Sievers, B. (1992). A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*.

Carlier, J., Péridy, L., Pinson, E., & Rivreau, D. (s.f.). Elimination Rules for Job Shop Scheduling Problem: Overview and extensions. En L. P. Jacques Carlier, *Classical Scheduling problems.*

Caumond, A., Lacomme, P., & Tchernev, N. (2008). A memetic algorithm for the job-shop with time-lags. *Computers & Operations Research*, 2331 – 2356.

Chassaing, M., Fontanel, J., Lacomme, P., Ren, L., Tchernev, N., & Villechenon, P. (2013). A GRASP x ELS approach for the job-shop with a web service paradigm packaging. *Expert Systems with Applications*.

Chen, B., & Matis, T. I. (2013). A flexible dispatching rule for minimizing tardiness in job shop scheduling. *Int. J.ProductionEconomics*.

Cheng, T. E. (1987). Integration of priority dispatching and due-date assignment in a job shop. En *Systems Science* (págs. 1813-1825).

Cheung. (1994). *Artificial Neural Networks for Intelligent Manufacturing.* Dagli, C.H.

Davoudpour, H., & Ashrafi, M. (2009). Solving multi-objective SDST flexible flow shop using GRASP algorithm. *The International Journal of Advanced Manufacturing Technology*, 737-747.

Demirkol, E. Mehta, S., & Uzsoy, R. (1997). A computacional study of shifting bottleneck procedures for shop scheduling problems. *Journal of Heuristics*, 111-137.

Duarte, R., Rego, C., & Gamboa, D. (2009). A filter and fan approach to the job shop scheduling.

Ebadi, A., & Moslehi, G. (2013). An optimal method for the preemptive job shop scheduling problem. *Computers & Operations Research*, 1314–1327.

Essafi, I., Mati, Y., & Dauzère-Pérès, S. (2008). A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. *Computers & Operations Research*, 2599 – 2616.

Feo, T. A., & Resende, M. G. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 109-133.

Fisher, M. L. (1973). Optimal Solution of Scheduling Problems using Lagrange Multipliers: Part II.

Gao, L., Zhang, G., Zhang, L., & Li, X. (2011). An efficient memetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering*, 699–705.

Giffler, B., & Thompson, G. L. (1960). Algorithms for Solving Production Scheduling Problems. En *Operations Resarch* (págs. 487-503).

Giovanni, L. D., & Pezzella, F. (2010). An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem. *European Journal of Operational Research*, 395–408.

Glover, F. (1990). Tabú Search - Part II. *ORSA Journal on computing, 2*.

Gonzalez, F. P. (Diciembre de 2005). Una metodología de solución basada en la metaheuríastica GRASP para el problema de diseño de red con incertidumbre.

Graham, R. L., Lawler, E. L., Lenstra, J., & Kan, A. H. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. En Graham, *Discrete Optimization II* (págs. 287–326). Vancouver: P.L. Hammer; E.L. Johnson; B.H. Korte.

Groover, M. P. (2007). *Automation production systems and computer-integrated Manufacturing* (3 ed.).

Hertz, A., & Widmer, M. (1996). An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. *Discrete applied Mathematics*, 319-345.

Holthaus, O., & Rajendran, C. (1997). New dispatching rules for scheduling in a job shop — An experimental study. *The International Journal of Advanced Manufacturing Technology*, 148-153.

Hurink, J., Jurish, B., & Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum*, 205-215.

Ivan Lazar, M. (2012). *A review on solving the job shop scheduling problem: Recent development and trends.* Technical University of Kosice with a seat in Presov, Faculty of manufacturing Technologies, Presov.

Jain, A., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of operational Research*.

Jayamohan, M., & Rajendran, C. (2004). Development and analysis of cost-based dispatching rules for job shop scheduling. *European Journal of Operational Research*, 307–321.

Johnson, S. M. (1953). Optimal Two and three stage production schedules with setup times Included.

Kammer, M., Akker, M. v., & Hoogeveen, H. (2011). Identifying and exploiting commonalities for the job-shop scheduling problem. *Computers & Operations Research*, 1556–1561.

Kanet, J. J., & Hayya, J. C. (1982). Priority dispatching with operation due dates in a job shop. *Journal of Operations Management, 2*, 167–175.

Kutanoglu, E., & Wu, S. D. (2 de Diciembre de 1997). On combinatorial auction and lagrangean relaxation for distributed resource scheduling.

Lei, D.-m. (2012). Minimizing makespan for scheduling stochastic job shop with random breakdown. *Applied Mathematics and Computation*.

Li, J.-q., Pan, Q.-k., & Liang, Y.-C. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 647–662.

Li, Y., & Chen, Y. (2011). An effective TPA-based algorithm for job-shop scheduling problem. *Expert Systems with Applications*, 2913–2918.

Liu, M., Sun, Z.-j., Yan, J.-w., & Kang, J.-s. (2011). An adaptive annealing genetic algorithm for the job-shop planning and scheduling problem. *Expert Systems with Applications*, 9248–9255.

Luh, G.-C., Chueh, & Chung-Huei. (2009). A multi-modal immune algorithm for the job-shop scheduling problem. *Information Sciences*, 1516–1532.

Madivada, H., & Rao, C. (2012). A review on non traditional algorithms for job shop scheduling. *International journal of production technology and management (IJPTM)*.

Manne, A. S. (1960). *On the Job-Shop Scheduling Problem* (Vol. 8).

Mati, Y., Dauzère-Pérès, S., & Lahlou, C. (2011). A general approach for optimizing regular criteria in the job-shop scheduling problem. *European Journal of Operational Research*, 33–42.

Meeran, & Sheik, A. S. (1998). *A state of the art review of Job Shop shedulig techniques.* University of Dundee, Department of applied physics, electronic and mechanical engineering, Dundee, Scotland.

Meeran, A. S.-S. (1998). *A state of the art review of Job Shop shedulig techniques.* University of Dundee, Department of applied physics, electronic and mechanical engineering, Dundee, Scotland.

Meeran, S., & Morshed, M. S. (2012). A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study. *Journal of Intelligent Manufacturing*, 1063-1078.

Mellor, P. (1966). A Review of Job Shop Scheduling. *The OR Society*.

Metta, H. (2008). *Adaptive, Multi-objective Job Shop scheduling Using Genetic algorithms.* University of Kentucky.

Moradi, E., Ghomi, S. F., & Zandieh, M. (2011). Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem. *Expert Systems with Applications*, 7169–7178.

Moslehi, G., & Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *Int. J.Production Economics*, 14–22.

Panwalkar, S., & Iskander, W. (1976). A survey of Scheduling Rules. *Operations Resarch*.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research*, 3202 – 3212.

Pinedo, M. L. (2012). *Scheduling Theory, algorithms and systems* (4 ed.). New York: Springer.

Qing-dao-er-ji, R., Wang, Y., & Wang, X. (2013). Inventory based two-objective job shop scheduling model and its hybrid genetic algorithm. *Applied Soft Computing*, 1400–1406.

Rui Zhang, C. W. (2011). A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research*, 854–867.

Salhi, S. (2002). Defining tabu list size and aspiration criterion within tabu search methods. *Computers & Operations Resarch*, 67-86.

Schmidt, K. (2001). *Using Tabu Search to Solve the Job Shop Scheduling Problem with Sequence Dependent Setup Times.*

Sels, V., Gheysen, N., & Vanhoucke, M. (2012). A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. *International Journal of Production Research*, 4255-4270.

Sha, D., & Lin, H.-H. (2010). A multi-objective PSO for job-shop scheduling problems. *Expert Systems with Applications*, 1065–1070.

Singer, M., & Pinedo, M. (1998). A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. *IIE TRANSACTIONS*, 109-118.

Singer, M., & Pinedo, M. (1998). A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. *IIE Transactions*, 109-188.

Subramaniam, V., & Raheja, A. S. (2002). Reactive Recovery of Job Shop Schedules - A Review. *The international journal of advanced manufacturing technology*.

Tang, J., Zhang, G., Lin, B., & Zhang, B. (2011). A Hybrid Algorithm for Flexible Job-shop Scheduling Problem. *Procedia Engineering*, 3678 – 3683.

Tavakkoli-Moghaddam, R., Azarkish, M., & Sadeghnejad-Barkousaraie, A. (2011). A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem. *Expert Systems with Applications*, 10812–10821.

Teekeng, W., & Thammano, A. (2012). Modified Genetic Algorithm for Flexible Job-Shop Scheduling Problems. *Procedia Computer Science*, 122 – 128.

Vallada, E., Ruiz, R., & Minella, G. (2008). Minimizing total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and methaheuristics. *Computers & Operations research*, 1350 - 1373.

W.S. Gere, J. (1962). *A heuristic approach to job shop scheduling.* Thesis, Carnegie Institute of Technology.

Wang, S., & Yu, J. (2010). An effective heuristic for flexible job-shop scheduling problem with maintenance activities. *Computers & Industrial Engineering*, 436–447.

Yang, J.-h., Sun, L., Lee, H. P., Qian, Y., & Liang, Y.-c. (2008). Clonal Selection Based Memetic Algorithm for Job Shop Scheduling Problems. *Journal of Bionic Engineering*, 111-119.

Yusof, R., Khalid, M., Hui, G. T., Yusof, S. M., & Othman, M. F. (2011). Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm. *Applied Soft Computing*, 5782–5792.

Zhang, G., Shao, X., Li, P., & Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 1309–1318.

Zhang, R., & Wu, C. (2010). A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing*, 79–89.

Zhang, R., & Wu, C. (2011). A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research*, 854–867.

Zhang, R., & Wu, C. (2012). A hybrid local search algorithm for scheduling real-world job shops with batch-wise pending due dates. *Engineering Applications of Artificial Intelligence*, 209–221.

Zhang, R., Pei-ChannChang, & ChengWu. (2013). A hybrid genetic algorithm for the job shop scheduling problem with practical considerations for manufacturing costs: Investigations motivated by vehicle production. *Int. J.Production Economics*, 38–52.

Zhang, R., ShijiSong, & ChengWub. (2013). A hybrid artificial bee colony algorithm for the job shop scheduling problem. *Int. J.Production Economics*, 167–178.

Zhang, R., Song, S., & Wu, C. (2012). A two-stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem. *Knowledge-Based Systems*, 393–406.