Programación de la Producción en un Sistema de Job Shop Flexible para Minimizar el Retraso Máximo

Luz Angela Bermudez
Universidad del Norte, Barranquilla, Colombia, luzabermudez7@gmail.com

Lourdes Rico Pomares
Universidad del Norte, Barranquilla, Colombia, lourdes.rico.pomares@gmail.com

Miguel Rojas-Santiago
Universidad del Norte, Barranquilla, Colombia, miguelrojas@uninorte.edu.co

ABSTRACT

This study was conducted in a workshop that specializes in engines and power transmission systems repair for heavy-duty trucks utilized in the coal mining industry. The workshop is equipped with batch processing machines, single and parallel capacitated machines for prewashing, disassembly, washing and evaluation operations. Given a set of jobs, their process routes and sizes, the objective is to schedule the jobs, such that the L_{max} is minimized. The workshop scheduling problem is very similar to the job shop problem. To solve it, a Simulated Annealing (SA) heuristic is proposed and compared to an Ant Colony Optimization (ACO) heuristic in terms of solution quality and run time.

Keywords: Flexible Job Shop, Scheduling, Simulated Annealing, Ant Colony Optimization

RESUMEN

Este estudio fue llevado a cabo en un taller especializado en reparar motores y sistemas de transmisión de grandes camiones utilizados en la industria minera. El taller está equipado con máquinas que solo pueden realizar un trabajo al tiempo (capacidad unitaria) y máquinas que pueden procesar por lotes (BPM), las cuales están organizadas para trabajar en forma individual y en paralelo para las operaciones de prelavado, desarme, lavado y evaluación. Dado un grupo de trabajos, sus secuencias de reparación y tamaños, el objetivo es programarlos de tal manera que el retraso máximo (L_{max}) sea minimizado. El problema estudiado es muy similar al problema del Job Shop. Para solucionarlo se propone una heuristica de Recocido Simulado(SA), la cual es comparada con la heuristica de la Colonia de Hormigas (ACO) en cuanto a la calidad de la solución y al tiempo de corrida.

Palabras claves: Job Shop Flexible, Programación, Recocido Simulado, Colonia de Hormigas

1. INTRODUCCIÓN

El problema de programación de un sistema Job Shop (JS) es uno de los más estudiados en la literatura(Jain & Meeran, 1998), y son conocidos como NP-hard, puesto que para ciertas instancias no se puede encontrar una solución óptima (Garey, Johnson, & Sethi, 1996). Mientras que el problema del JS consiste en *n* trabajos y *m* máquinas, donde cada trabajo debe ser procesado en cada máquina con una secuencia predefinida y cada máquina puede procesar un solo trabajo a la vez, el Job Shop Flexible (FJS) muestra de mejor manera la realidad de los procesos dado que considera que una operación puede ser procesada en más de una máquina. Esta extensión del JS involucra dos tareas principales: la asignación de un trabajo a una máquina y el secuenciamiento de operaciones en cada máquina (Joc Cing Tay & Nhu Binh Ho, 2008).

Este artículo estudia el proceso productivo de una empresa dedicada a la reparación de equipos para los sectores de la construcción y minería, entre otros; especialmente la reparación de motores y sistemas de transmisión de camiones. La empresa cuenta con dos talleres, dedicados uno al desarme y evaluación de cada equipo y otro al ensamble de estos.

La función objetivo del estudio consiste en minimizar el retraso máximo de un grupo de órdenes de trabajo, con el fin de brindar una mejor atención al cliente y cumplir al máximo con los tiempos de entrega establecidos. Para lograr esto, en el presente trabajo se aplica la heurística de Recocido Simulado.

El artículo está organizado de la siguiente manera: en la Sección 2 se realiza un resumen del estado del arte del problema abordado, el cuál se describe en la Sección 3. Por su parte, en la Sección 4 se detalla el procedimiento utilizado para llevar a cabo los procedimientos computacionales desarrollados en la Sección 5. Por último, en la Sección 6 se presentan las conclusiones

2. ESTADO DEL ARTE

La mayoría de las soluciones propuestas por diversos autores a los problemas de programación de producción para un sistema Job Shop Flexible (FJS) se basan en distintos enfoques de heurísticas y meta-heurísticas. Un gran número de artículos publicados plantean procedimientos híbridos, tomando lo mejor de cada heurística con el fin de lograr el resultado más cercano al óptimo del problema planteado. Un ejemplo es el algoritmo híbrido evolucionario propuesto por Zobolas, Tarantilis, e Ioannou (2009). Esta metodología comprende los siguientes tres pasos: i) Generación de diversas soluciones iniciales usando el algoritmo DE (Differential Evolution –based algorithm), el cual ha sido aplicado exitosamente en varios problemas de optimización y programación (Nearchou, 2006); ii) Intensificación usando la Búsqueda Local, por medio del método VNS (Variable Neighborhood Search), probado para problemas de programación, incluso como parte de un método híbrido (Hindu, Fleszar, & Charamlambous, 2003); y iii) Diversificación y escape de un óptimo local con GA (Genetic Algorithm), el cual ha demostrado ser efectivo para una variedad de problemas de programación con optimización combinatoria (Dorndorf & Pesch, 1993). Los resultados obtenidos tanto con el método formulado por Zobolas et al. como los obtenidos con el de Búsqueda Controlada por Recocido Simulado (Matsuo, Suh, & Sullivan, 1998) son mejores que los de otros algoritmos de su tipo.

Otros autores han formulado metodologías híbridas como es el caso de Tsung-Che Chiang y Hsiao-Jou Lin (2013), quienes abarcaron el problema de JSF multi-objetivo evolucionario para minimizar el makespan, la carga de trabajo total y la carga máxima de trabajo. El problema es resuelto por medio de método de Pareto. Estos autores proponen un algoritmo multi-objetivo evolucionario, utilizando operadores genéticos efectivos que mantienen la diversidad de la población de manera cuidadosa. Por su parte, en otro trabajo (Nascimento, 1993) se busca implementar el algoritmo AGT de Giffler y Thompson (1960) para compararlo con el ACS propuesto por Chang y Sullivan (1990), el cual es una generalización del AGT. Debido a que estos algoritmos por sí solos no funcionan para problemas complejos dado que fueron diseñados para problemas de Job Shop, el autor utilizó el enfoque "branch-and-bound" para mejorar su efectividad, puesto que asegura que el espacio de búsqueda de la solución sea el adecuado para encontrar el óptimo.

Por otro lado, diferentes autores (Kacem, Hammadi, & Borne, 2002) plantean combinar las decisiones de asignación y programación en niveles de producción iguales. Ellos presentan dos métodos: Approach by Localization (AL) y Controlled Genetic Algorithm (CGA). Por otra parte, Rajeev Agrawal y Pattanaik (2012) proponen resolver un problema de FJS, en el cual existen máquinas que se alternan para procesar el mismo trabajo, utilizando el Algoritmo Genético multi-objetivo (GA). En otro trabajo de los mismos autores (Kacem, Hammadi, & Borne, 2002) se generaron un grupo de cinco estudios de casos, los cuales sirvieron de base para investigar sobre FJS multi-objetivo. En ese trabajo ellos propone un enfoque de Pareto basado en la hibridación de la lógica difusa (FL) y los algoritmos evolutivos (AE), para resolver problemas de programación de FJS. Otros autores como Nasr Al-Hinai y ElMekkawy (2011) abarcan el problema de encontrar una solución robusta y estable a problemas de FJS con máquinas que presentan fallas aleatorias, mediante un Algoritmo Híbrido Genético –HGA. Por su parte, Vinod y Sridharan (2009) estudiaron experimentalmente reglas de decisión para la programación de Job Shop dinámicos utilizando un modelo basado en simulación discreta. Ellos consideraron un

sistema de Job Shop parcialmente flexible, compuesto de seis máquinas de las cuales tres trabajan en paralelo. Mientras que Caballero & Mejía (2010) aplican Redes de Petri y Algoritmos Genéticos para minimizar la tardanza ponderada en un FJS. Otras propuestas, como la de Mati, Rezg y Xiaolan Xie (2011), utilizan una Heurística Miope para resolver problemas con más de dos trabajos; mientras que Durán, Rojas y Daza (2011) proponen un algoritmo genético secuencial para resolver el problema del JSF y Li Li y Keqi Wang (2009) presentan la heurística de la Colonia de Hormigas para un problema de programación FJS multi-objetivo. La búsqueda Tabú también ha sido utilizada en problemas FJS (Logendran y Sonthinen, 1997); por ejemplo, Fattahi, Saidi Meharab, y Jolai (2007) proponen una combinación de este método con Recocido Simulado. Por otro lado, Baykasoglu, Owen y Gindy (1999) utilizan un algoritmo basado en la búsqueda Tabú, con el fin de encontrar la mejor solución a un problema de optimización multi-objetivo.

En resumen, el problema de Flexible Job Shop ha sido investigado ampliamente por diferentes académicos en las últimas dos décadas. Cabe resaltar que, en su mayoría, las investigaciones enfocan su función objetivo en el makespan, la carga de trabajo, y la carga máxima de trabajo.

3. DESCRIPCIÓN DEL PROBLEMA

El problema que se estudia en la presente investigación es: Minimizar el máximo retraso de un número *n* de trabajos, dados sus respectivos tiempos de procesamiento, fechas de entregas, tamaños y ruta del proceso de reparación de cada uno de ellos en un número de bahías o máquinas *m* agrupadas en *M* estaciones. El proceso inicia con el recibo de la parte o componente a reparar. Luego, se lleva a la estación de prelavado donde se limpia y luego se pasa a la sección de desarme, la cual cuenta con tres (3) bahías para motores y dos (2) bahías para sistemas de transmisión. En esta sección, el equipo se desarma y las piezas se envían a la estación de lavado, la cual cuenta con dos máquinas en paralelo que tienen capacidad de trabajar por lotes. Luego, las piezas desarmadas se evalúan en otra bahía para ser enviadas a otro taller dentro de la misma empresa, quien llevará a cabo la respectiva reparación.

Debido a la complejidad del problema, el alcance de este estudio sólo cubre las tres primeras estaciones, que corresponden a las operaciones de prelavado, desarme y evaluación. Sin embargo, la empresa estudiada tiene más estaciones, las cuales se describen a continuación: M1 (prelavado), M2 (desarme de motores), M3 (desarme de los sistemas de transmisión de potencia), M4 (lavadora), M5 (evaluación), M6 (armado de motores), M7 (armado de sistemas de transmisión de potencia), y M8 (dinamómetro). La planta está dividida en tres áreas: la zona de desarme y evaluación (ver Figura 1), la zona de armado y la zona de pruebas.

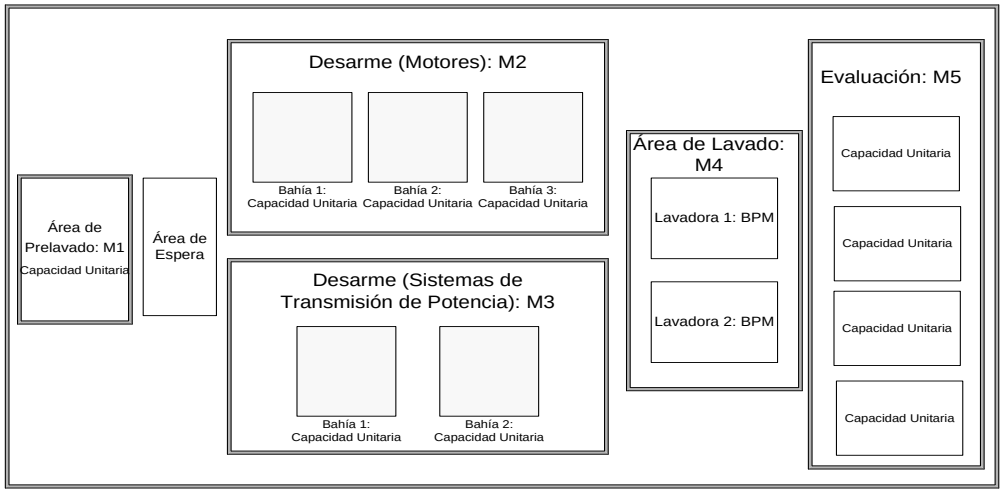

Figura 1: Distribución en planta del taller dedicado al Desarme y Evaluación.

Para solucionar el problema propuesto, se han hecho algunas suposiciones:

• El proceso de reparación comienza con la recepción de la pieza o componente a reparar.

• El primer paso del proceso es la operación de prelavado, que se realiza en una máquina que sólo puede realizar un trabajo al tiempo. En esta estación (M1), la parte se limpia con el fin de ser llevado a la sección de desarme.

• El segundo paso consiste en desarmar los motores y sistemas de transmisión, dónde cada una de estas piezas tiene su ruta de proceso, tamaño y tiempos de procesamiento. La sección de desarme cuenta con 3 bahías (M2) para motores y 2 bahías (M3) para los sistemas de transmisión. Todas esas bahías solo pueden procesar un trabajo a la vez.

• El tercer paso se realiza en la estación de lavado, la cual cuenta con dos máquinas que pueden limpiar simultáneamente un lote de máximo D piezas, sin exceder su capacidad S; donde S se refiere al total de volumen o tamaño que puede lavar al tiempo.

• El último paso se realiza en la estación de evaluación y puede ser realizado en cualquiera de las cuatro bahías asignadas para esta operación. Las bahías trabajan en paralelo y solo pueden procesar un trabajo a la vez.

• Algunos componentes requieren ser lavados una segunda vez (esto es, se devuelven al tercer paso), después de ser inspeccionados por ensayos no destructivos (END).

• El proceso termina cuando las partes están armadas y probadas, despues que el cliente autorizara su reparación.

• En este paper sólo se programan las primeras tres estaciones (M1 , M2 y M3), de tal forma que el problema considerado en esta investigación se puede denotar por *FJm||Cmax*.

• En una estación con máquinas en paralelo, el primero que está en la cola de los trabajos pendientes se programa o asigna a la primera máquina que esté disponible.

• Todas las máquinas están disponibles en el tiempo 0.

• Los tiempos de llegada de los trabajos son iguales a 0.

• Todos los trabajos se deben terminar una vez son comenzados.

• No se considera que las maquinas tengan averías.

• La recirculación de trabajos no está permitida entre las estaciones M1 y M3.

El problema objeto de estudio se puede modelar como un grafo dirigido con N nodos, donde cada nodo (*i*, *j*) representa una operación del trabajo *j* en una máquina que pertenece a la estación *i* y cada cuadrado representa una estación con varias máquinas trabajando en paralelo. Una representación gráfica de este problema se muestra en la Figura 2 para una instancia con siete trabajos (ver Tabla 1), que van a ser procesados en sólo tres estaciones; con una máquina en M1 y dos y tres máquinas en las estaciones M2 y M3, respectivamente. En dicha tabla, la columna (3) muestra las operaciones requeridas para cada trabajo, la columna (4) presenta el tiempo de procesamiento p_{ij} del trabajo *j* en la máquina *i*, y la columna (5) establece la fecha de entrega d_j de cada trabajo *j*.

Tabla 1: Datos de un problema con siete trabajos

Parte	Trabajo	Secuencia de procesamiento	Tiempo de procesamiento	Fecha de entrega
Motor	1	M1, M2	p_{11}=48, p_{21}=159	290
	2	M1, M2	p_{12}=41, p_{22}=155	274
	3	M1, M2	p_{13}=42, p_{23}=120	227
	4	M1, M2	p_{14}=47, p_{24}=124	239
Sistema de Transmisión	5	M1, M3	p_{15}=36, p_{35}=160	274
	6	M1, M3	p_{16}=36, p_{36}=156	269
	7	M1, M3	p_{17}=33, p_{37}=149	255

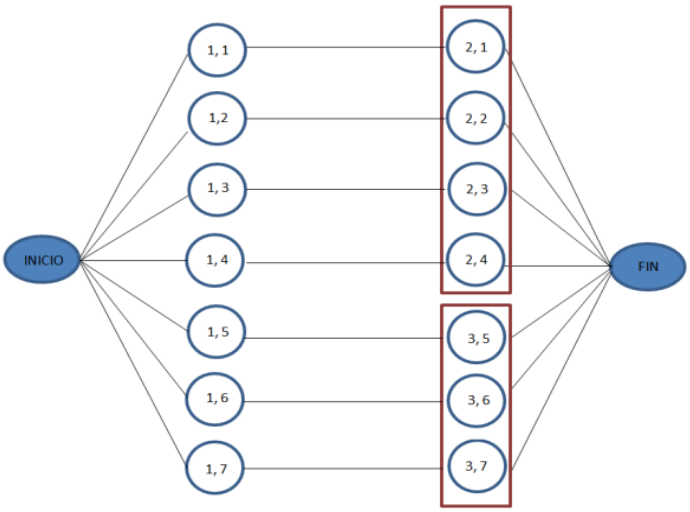

Figura 2: Representación gráfica de un problema de siete trabajos.

4. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

La heurística de Recocido Simulado fue introducida por primera vez por Kirkpatrick, Gelatt y Vecchi (1983), quienes hicieron una analogía del proceso físico del tratamiento térmico aplicado a los metales para resolver grandes problemas de optimización combinatoria. Este proceso consiste en aumentar la temperatura hasta un valor máximo antes que el sólido empiece a fundirse, para luego disminuirla gradualmente hasta que el material alcance un estado de estructura cristalina perfecta. En Recocido Simulado, Brooks y Morgan (1995) consideran un valor inicial alto de temperatura T_0, el cual va decreciendo hasta alcanzar el equilibrio térmico a medida que van pasando las iteraciones, y en el que la probabilidad de que el sistema esté en cierto estado con cierta energía (E), es dada por la distribución de Boltzman.

El pseudo código de la solución propuesta se detalla a continuación:

MenorOpt ← Número grande (En esta variable se guarda el mejor de todos los resultados)

K← Número máximo de iteraciones

T← Temperatura

α←0.95

S←Cmax de la secuencia encontrada con la regla SPT.

for iter=0 hasta k **do**

<<Genera otra secuencia a partir de la inicial y se calcula su Cmax (S´)>>

if S' < S **then**

S ← S'

else

if $e^{\frac{s-s'}{T}}$ > random [0,1) **then**

S ← S'

end if

end if

```
        T ← α*T
        if S < MenorOpt then
        MenorOpt ← S
        end if

end for
```

5. RESULTADOS

La heurística de Recocido Simulado fue implementada en Matlab, mientras que ACO se realizó en C++. Los experimentos se llevaron a cabo en una PC con un procesador de 2 GHz y 2 GB de memoria RAM. Para el estudio se consideraron instancias con 25, 50 y 75 trabajos generados de manera aleatoria. Para evaluar la eficacia de SA, los resultados fueron comparados contra ACO, una heurística propuesta por Colorni, Dorigo y Maniezzo (1996). Este método se inspira en el comportamiento que siguen las hormigas para encontrar el camino más corto entre una fuente de comida y el hormiguero.

Iniacialmente, para cada heuristica se realizaron experimentos para establecer cuales combinaciones de parámetros eran las más adecuadas para hallar la mejor solución. Inicialmente, los parámetros para SA fueron: Número total de iteraciones = 1000, 5000 y 10000; temperatura inicial= 500, 750 y 1000; y α= 0.8, 0.9, 0.95 y 0.99. Para ACO se analizaron los siguientes: Hormigas =1, 5 y 10 por trabajo; Q =1, 100 y 10000; α= 0, 0.5, 1, 2 y 5; β=0, 1, 2 y 5; ρ= 0.3, 0.5, 0.7, 0.9 y 0.9999.

Para desarrollar la heuristica SA, el primer paso es generar una solución inicial, la cual para este estudio se obtuvo con la regla SPT(Shortest Processing Time). En el caso de la instancia de la Tabla 1, la solución inicial Lmax=595 fue luego mejorada usando el algoritmo de SA hasta llegar a Lmax=593. En todo caso, para cada una de las instancias se registró la mejor solución y se calculó la diferencia porcentual entre el Lmax dado por SA y el Lmax de ACO utilizando la ecuación (1).

$$\%Dif = \frac{C_{Lmax}^{ACO} - C_{Lmax}^{SA}}{C_{Lmax}^{ACO}} \times 100\% \quad (1)$$

Los resultados de las instancias consideradas en este artículo se muestran en la Tabla 2, cuya cuarta columna presenta el porcentaje calculado con la ecuación (1). Por su parte, la Figura 3 muestra los valores de Lmax obtenidos con SA y ACO, mientras que la Tabla 3 y la Figura 4 presentan el tiempo promedio que gastaron esas heuristicas. Para todas las instancias, el resultado de Lmax obtenido con SA es mejor que el de ACO. Además, SA encuentra la solución en menor tiempo que ACO.

6. CONCLUSIONES

El problema de FJS con máquinas que procesan un solo trabajo al tiempo y otras de procesamiento por lotes, algunas operando en paralelo, fue abordado utilizando heurísticas basadas en SA y en ACO. Sinembargo, dado que es un problema NP-hard, sólo tres estaciones fueron consideradas en este paper, con el fin de simplificar su implementación; el resto de las estaciones deben ser incluídas en estudios futuros. En resumen, se observa que la calidad del Lmax y el tiempo promedio computacional requerido para correr la heurística SA es mucho menor que el obtenido por ACO, lo cual hace que el Recocido Simulado sea una metodología más atractiva para aquellos que quieran resolver este problema.

Tabla 2: Lmax Obtenido con SA y ACO.

Código de Instancia	Lmax		
	SA	ACO	% Dif.
25j_1	1087	1396	-28.43%
25j_2	1095	1396	-27.49%
25j_3	1073	1455	-35.60%
25j_4	1083	1428	-31.86%
50j_1	1908	2246	-17.71%
50j_2	1909	2211	-15.82%
50j_3	1898	2242	-18.12%
50j_4	1898	2232	-17.60%
75j_1	2972	3348	-12.65%
75j_2	2966	3333	-12.37%
75j_3	2959	3395	-14.73%
75j_4	2937	3389	-15.39%

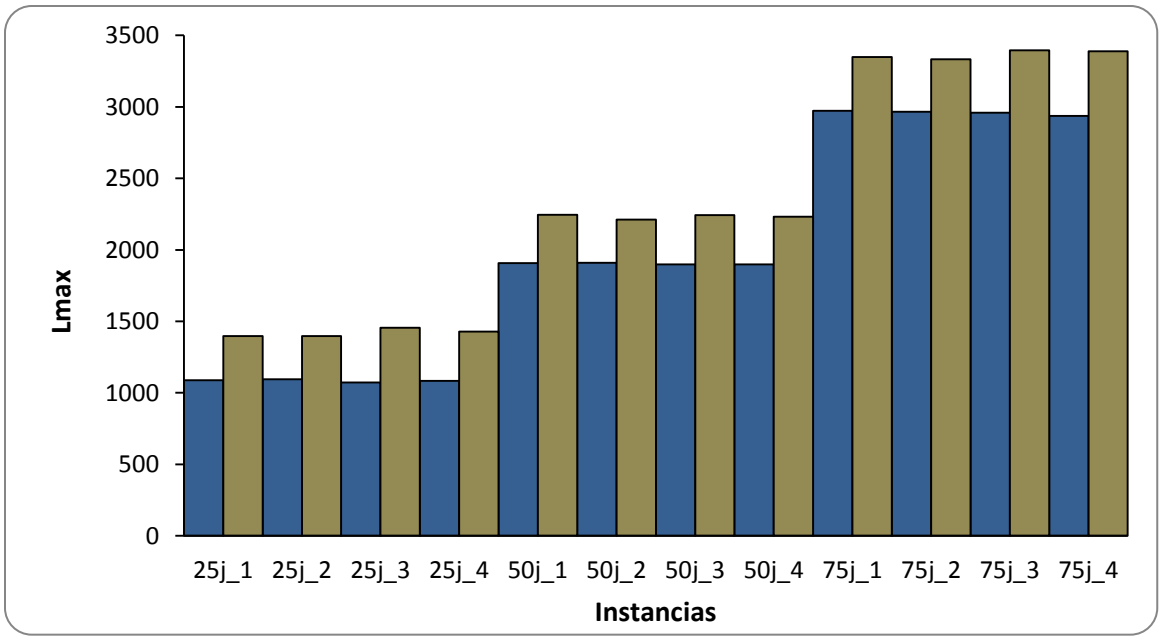

Figura 3: Comparación de Lmax entre SA y ACO.

Tabla 3: Comparación de tiempo promedio de corrida entre SA y ACO

Código de la Instancia	Tiempo de corrida (sec.)	
	SA	ACO
25j_1	0.23	8.65
25j_2	0.26	8.57
25j_3	0.51	8.25
25j_4	0.53	8.85
50j_1	0.45	92.92
50j_2	0.64	92.49
50j_3	0.97	78.46
50j_4	0.71	90.73
75j_1	0.72	379.7
75j_2	0.74	384.22
75j_3	1.38	382.78
75j_4	1.36	384.01

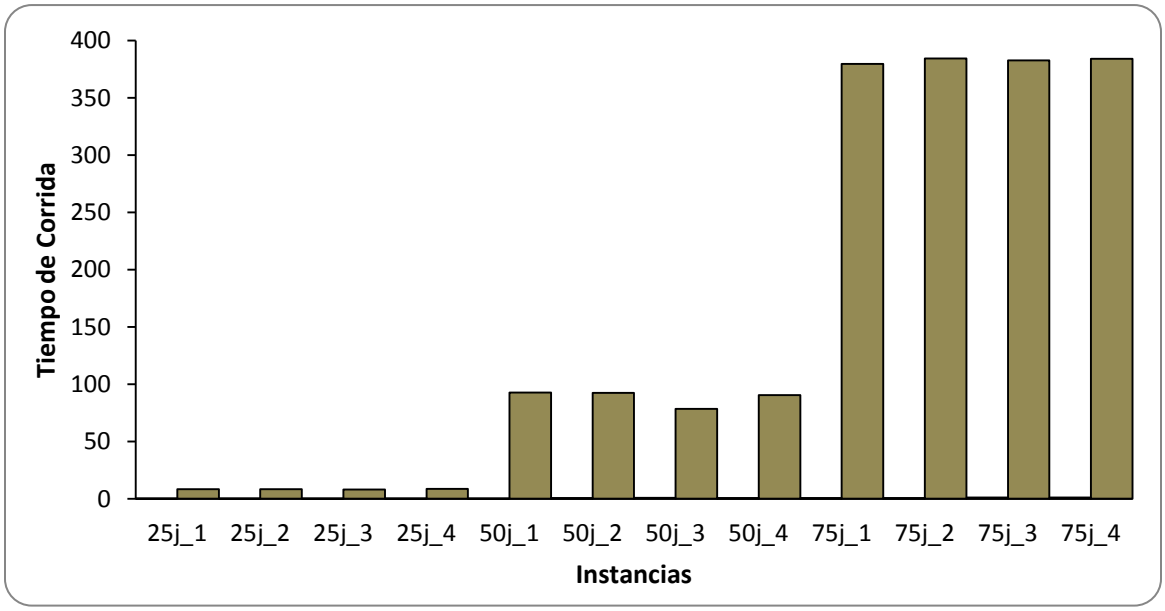

Figura 4: Comparación del tiempo promedio de corrida entre SA y ACO

BIBLIOGRAFÍA

Arias, J., & Gonzalez, J. (2008). *Ordenamiento Óptimo Usando el Algoritmo Recocido Simulado y Almacenamiento Compacto Aplicados a la Solución de sistemas de Ecuaciones Lineales.* Pereira: Programa de Ingeniería Eléctrica, Universidad Tecnológica de Pereira.

Baykasoglu, A., Owen, S., & Gindy, N. (1999). A taboo search based approach to find the pareto optimal set in multiple objective optimization. *Engineering Optimization, 31(6)* , 731-478.

Brooks, S., & Morgan , B. (1995). Optimization Using Simulated Annealing. *Journal of the Royal Statistical Society. Series D (The Statistician) , Vol. 44, No. 2* , 241-257.

Caballero, J., & Mejía, G. (2010). Redes de Petri y algoritmos genéticos, una propuesta para la programación de sistemas de manufactura flexible. *Ingenieria y Universidad, 10(1).*

Chang, Y., & Sullivan, R. (1990). Schedule generation in a dynamic job shop. *Int. J. Prod. Res. 28* , 65-74.

Colorni, A. D. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part-B, Vol. 26, No.1.* , pp. 1-13.

Dorndorf, U., & Pesch, E. (1993). Combining genetic and local search for solving the job shop scheduling problem. In: Maros I (ed). . *Symposium on Applied Mathematical Programming and Modelling APMOD93* (pp. 142-149). Budapest, Hungary: Akaprint.

Durán, R., Rojas, L., & Daza, V. (2011). Un algoritmo genético para el problema de job shop Flexible/A genetic algorithm for the flexible job shop problem. *Ingeniare : Revista Chilena De Ingenieria, 19(1)* , 53-61.

Fattahi, P., Saidi Meharab, M., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Jounal of Intelligent Manufacturing. Vol. 8, Issue 3* , 331-342.

Garey, M., Johnson, D., & Sethi, R. (1996). The complexity of flow shop and job-shop scheduling. *Mathematics of Operations Research, 1* , 117-129.

Giffler, B., & Thompson, G. L. (1960). Algorithms for solving production-scheduling problems. *Opns Res. 8* , 487-503.

Hindu, K., Fleszar, K., & Charamlambous, C. (2003). *An effective heuristic for the CLSP with set-up times.* J Opl Res Soc 54: 490-498.

Jain, A., & Meeran, S. (1998). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research* , 390-434.

Joc Cing Tay, & Nhu Binh Ho. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering, Volume 54, Issue 3,* , 453-473.

Kacem, I., Hammadi, S., & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.32, no.1* , 1-13.

Kacem, I., Hammadi, S., & Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation, 60* , 245-276.

Kirkpatrick, S. (1984). "Optimization by simulated annealing: quantitative studies". *Journal of Statistical Physics 34:* , 5-6.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by Simulated Annealing. *Science, New Series, Vol. 220, No. 4598* , 671-680.

Lambrechts, O., Demeulemeester, E., & Herroelen, W. (2008). A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics 111* , 493-508.

Li Li, & Keqi Wang. (2009). An improved ant colony algorithm for multi-objective flexible job shop scheduling problem. *Automation and Logistics, 2009. ICAL '09. IEEE International Conference on , vol., no.,* , 697,702,5,7.

Logendran, R., & Sonthinen, A. (1997). A Tabu Search-Based Approach for Scheduling Job-Shop Type Flexible Manufacturing Systems. *The Journal of the Operational Research Society , Vol. 48, No. 3* , 264-277.

Mati, Y., Rezg, N., & Xiaolan Xie. (2011). An integrated greedy heuristic for a flexible job shop scheduling problem. *Systems, Man, and Cybernetics IEEE International Conference on , vol.4* , 2534,2539.

Matsuo, H., Suh, C., & Sullivan, R. (1998). *A controlled search simulated annealing method for the general job-shop scheduling problem.* Graduate School of Business, University of Texas at Austin.

Nascimento, M. A. (1993). Giffler and Thompson's Algorithm for Job Shop Scheduling is Still Good for Flexible Manufacturing Systems. *The Journal of the Operational Research Society , Vol. 44, No. 5* , 521-524.

Nasr Al-Hinai, & ElMekkawy, T. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics, Volum000e 132, Issue 2* , 279-291.

Nearchou, A. (2006). *A differential evolution approach for the common due date early/tardy job scheduling problem.* Comput Opl Res 35: 1329-1343.

Rajeev Agrawal, L., & Pattanaik, S. (2012). Scheduling of a flexible job-shop using a multi-objective genetic algorithm. *Journal of Advances in Management Research, Vol. 9 Iss: 2* , 178-188.

Surico, M., Kaymak, U., Naso, D., & Dekker, R. (2006). Hybrid Meta-Heuristic for Robust Scheduling. *ERIM Report Series Reference No.ERS-2006-018-LIS.*

Torres Delgado, J. F., & Vélez Gallego, M. C. (2009). Algoritmo de recocido simulado para la descomposición robusta del horizonte de tiempo en problemas de planeación de producción. *Red Ingeniería y Ciencia* , 18.

Tsung-Che Chiang, & Hsiao-Jou Lin. (2013). A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. *International Journal of Production Economics, Vol 141, Issue1* , 87-98.

Vinod, V., & Sridharan, R. (2009). Development and analysis of scheduling decision rules for a dynamic flexible job shop production system. *International Journal of Business Performance Management* , 43-71.

Zobolas, G., Tarantilis, C., & Ioannou, G. (2009). A Hybrid Evolutionary Algorithm for the Job Shop Scheduling Problem. *The Journal of the Operational Research Society , Vol. 60, No. 2* , 221-235.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.