Revista EIA, ISSN 1794-1237 Número 13, p. 39-51. Julio 2010
Escuela de Ingeniería de Antioquia, Medellín (Colombia)

UN ALGORITMO GENÉTICO HÍBRIDO Y UN ENFRIAMIENTO SIMULADO PARA SOLUCIONAR EL PROBLEMA DE PROGRAMACIÓN DE PEDIDOS *JOB SHOP*

José David Meisel*
Liliana Katherine Prado**

RESUMEN

La programación de pedidos para el problema de producción Job Shop (JSP), catalogado como NP-Hard, ha constituido un reto para la comunidad científica, debido a que alcanzar una solución óptima a este problema se dificulta en la medida que crece en número de máquinas y trabajos. Numerosas técnicas, entre ellas las metaheurísticas, se han empleado para su solución, sin embargo, su eficiencia, en cuanto a tiempo computacional, no ha sido muy satisfactoria. Por lo anterior y para contribuir a la solución de este problema, se planteó el uso de un enfriamiento simulado propuesto (ESP) y de un algoritmo genético mejorado (AGM). Para el AGM se implementó una estrategia de enfriamiento simulado en la fase de mutación, que permite al algoritmo intensificar y diversificar las soluciones al mismo tiempo, con el fin de que no converja prematuramente a un óptimo local. Los resultados mostraron que los algoritmos propuestos arrojan buenos resultados, con desviaciones alrededor de los mejores valores encontrados que no superan el 5 % para los problemas más complejos.

PALABRAS CLAVE: *Job Shop;* algoritmo genético; enfriamiento simulado; administración de operaciones; optimización combinatoria.

A HYBRID GENETIC ALGORITHM AND A SIMULATED ANNEALING FOR SOLVING THE JOB SHOP SCHEDULING PROBLEM

ABSTRACT

Job Shop Scheduling Problem (JSP), classified as NP-Hard, has been a challenge for the scientific community because achieving an optimal solution to this problem is complicated as it grows in number of machines and jobs. Numerous techniques, including metaheuristics, have been used for its solution; however, the efficiency of

* Ingeniero Industrial, Universidad de Ibagué; Magíster en Ingeniería Industrial, Universidad de los Andes. Docente e Investigador, Grupo de Investigación GINNOVA, Universidad de Ibagué. Ibagué, Colombia. Jose.meisel@unibague.edu.co

** Ingeniera Industrial, Universidad de Ibagué. Ibagué, Colombia. liloprado@hotmail.com

Artículo recibido 7-III-2010. Aprobado 26-V-2010
Discusión abierta hasta diciembre de 2010

the techniques, in terms of computational time, has not been very satisfactory. Because of this and for contributing to the solution of this problem, a simulated annealing (SA) and an improved genetic algorithm (IGA) have been proposed. The latter, by implementing a strategy of simulated annealing in the mutation phase, allows the algorithm to enhance and diversify the solutions at the same time, in order not to converge prematurely to a local optimum. The results showed that the proposed algorithms yield good results with deviations around the best values found not exceeding 5 % for more complex problems.

KEY WORDS: Job Shop; genetic algorithm; simulated annealing; operations management; combinatorial optimization.

UM ALGORITMO GENÉTICO HÍBRIDO E UM ESFRIAMENTO SIMULADO PARA SOLUCIONAR O PROBLEMA DE PROGRAMAÇÃO DE PEDIDOS JOB SHOP

RESUMO

A programação de pedidos para o problema de produção Job Shop (JSP), catalogado como NP-Hard, tem constituído um desafio para a comunidade científica, devido a que alcançar uma solução ótima a este problema se dificulta na medida em que cresce em número de máquinas e trabalhos. Numerosas técnicas, entre elas as metaheurísticas, foram empregadas para sua solução, no entanto, sua eficiência, em quanto a tempo computacional, não há sido muito satisfatória. Pelo anterior e para contribuir à solução deste problema, propôs-se o uso de um esfriamento simulado proposto (ESP) e de um algoritmo genético melhorado (AGM). Para o AGM se implementou uma estratégia de esfriamento simulado na fase de mutação, que permite ao algoritmo intensificar e diversificar as soluções ao mesmo tempo, com o objetivo de que não convirja prematuramente a um ótimo local. Os resultados mostraram que os algoritmos propostos arrojam bons resultados, com desvios ao redor dos melhores valores achados que não superam 5 % para os problemas mais complexos.

PALAVRAS-CÓDIGO: Job Shop; algoritmo genético; esfriamento simulado; administração de operações; otimização combinatória.

1. INTRODUCCIÓN

La investigación que recoge este artículo tiene como objetivo solucionar el problema de programación de pedidos *Job Shop* (JSP), que trata de determinar el orden de emisión y el tiempo de un conjunto de trabajos sobre máquinas relevantes sujeto a restricciones de procesamiento, en un esfuerzo por mejorar la eficiencia en la producción y reducir el tiempo de procesamiento, de tal manera que se incremente la utilidad tanto como sea posible (Zhou, Feng y Han, 2001).

El problema de programación *Job Shop* es un problema NP-Hard, por lo cual, a medida que empieza a crecer en número de variables y número de restricciones, el tiempo de corrida del algoritmo se incrementa a una tasa exponencial (Garey, Johnson y Sethi, 1976). Por ejemplo, una instancia bastante simple con un tamaño pequeño, como 10 trabajos y 10 máquinas, es difícil de resolver con una aproximación bastante exacta. De hecho, la solución óptima no es tan necesaria en muchas situaciones prácticas, porque el tiempo computacional y el costo para una solución óptima son usualmente tan largos que apenas si se puede permitirlos, en cambio una buena solución subóptima puede ser bien aceptada (Zhou, Feng y Han, 2001).

El JSP es un problema que, a pesar de ser clásico en la literatura, aún es relevante de resolver, debido a su complejidad y al bajo desempeño que han mostrado métodos de solución exactos en su resolución. Por esta razón, muchos investigadores han diseñado diferentes heurísticas y metaheurísticas para solucionarlo; en particular durante los últimos

años la investigación sobre este problema se ha incrementado (Zang y Wu, 2010).

La teoría de la programación de pedidos ha sido objeto de análisis en la literatura científica, con técnicas que van desde las reglas de despacho (Krajewski y Ritzman, 2000), pasando por algoritmos de bifurcación y acotación (*branch and bound*) hasta heurísticas basadas en cuellos de botella, redes neuronales, redes de Hopfield, inteligencia artificial, métodos de búsqueda local y metaheurísticas. Éstas últimas se han difundido ampliamente en diversas publicaciones como una opción eficiente de solución, por ello son consideradas en esta investigación a partir de la comparación de los mejores resultados obtenidos (Fisher y Thompson, 1963; Lawrence, 1984).

Alrededor del año 2004, los investigadores empezaron a orientarse hacia la aplicación de metaheurísticas híbridas para resolver el JSP, por ejemplo, el trabajo realizado por Azizi y Zolfaghari (2004), autores que mostraron que los resultados analíticos conocidos sobre la convergencia del enfriamiento simulado no soportan su aplicación al JSP. Ellos presentaron una nueva propuesta que usa una pequeña población de corridas del enfriamiento simulado en el marco de un algoritmo genético. Las características nuevas fueron un control adaptable de temperaturas que permitía "recalentamiento" del enfriamiento; usaron un esquema de enfriamiento adaptable que ajustaba la temperatura dinámicamente, basado en el perfil de la ruta de búsqueda.

De igual manera, Gonçalves, Mendes y Resende (2005) presentaron un algoritmo genético híbrido para el JSP. La representación del cromosoma del problema estaba basada en llaves aleatorias. Los programas fueron construidos usando una regla de preferencia definida por el algoritmo genético y construidos usando un procedimiento que genera programas activos parametrizados. Después que un programa se obtiene, una heurística de búsqueda local se aplica para mejorar la solución.

Finalmente, otro trabajo para destacar es la investigación realizada por Vilcot y Billaut (2008), quienes trataron el problema de *Job Shop* general con múltiples restricciones, procedentes de la industria gráfica. Su objetivo fue la minimización de dos criterios, el *makespan* y el *retraso máximo*, centrados en encontrar una aproximación de la frontera de Pareto; propusieron un algoritmo genético rápido y elitista basado en NSGA-II para solucionar el problema. La población inicial de este algoritmo es generada al azar o parcialmente generada usando un algoritmo de búsqueda de tabú, que minimiza una combinación lineal de ambos criterios.

En este artículo se propone el uso de un algoritmo genético mejorado (AGM) y un enfriamiento simulado propuesto (ESP). El artículo está organizado en 9 secciones. En la sección 2 se presenta la formulación matemática del JSP, en las secciones 3, 4 y 5 se presentan los dos algoritmos propuestos para solucionar el JSP. Las ganancias en la eficiencia del AGM se logran en 5 pasos: 1) generación de la población inicial mediante la heurística G&T; 2) selección de la subpoblación; 3) selección de los padres por torneo; 4) cruce empleando *Re-insert*; 5) mutación, que se realiza aplicando una estrategia de enfriamiento simulado en la que se explora la vecindad mediante una estrategia *Re-insert*. En cuanto al ESP, se utiliza la misma representación y generación del cromosoma utilizada por el AGM, pero la estrategia de vecindad utilizada es *Swap*. En la sección 6 se presenta el esquema de validación de los algoritmos propuestos, en las secciones 7 y 8 se reportan los resultados computacionales y se presenta su análisis y las conclusiones son presentadas en la sección 9.

2. FORMULACIÓN DEL PROBLEMA

Matemáticamente, el problema de secuenciación de pedidos *Job Shop* (JSP) puede describirse como sigue (Aiex, Binato y Resende, 2003). Dado un conjunto de $\mathcal{M}$ máquinas (donde se define el tamaño de $\mathcal{M}$ por $|\mathcal{M}|$) y un conjunto de trabajos

$\mathcal{J}$ (donde el tamaño de $\mathcal{J}$ se denota por $|\mathcal{J}|$), sea $\sigma_1^j \prec \sigma_2^j \prec \cdots \sigma_{|\mathcal{M}|}^j$ el conjunto ordenado de $|\mathcal{M}|$ operaciones del trabajo $\mathcal{J}$ donde $\sigma_k^j \prec \sigma_{k+1}^j$ indica que la operación σ_{k+1}^j puede ser procesada sólo después de que haya sido completada la operación σ_k^j. Sea $\mathcal{O}$ el conjunto de operaciones. Cada operación σ_k^j se define por dos parámetros: $\mathcal{M}_k^j$ es la máquina en la cual σ_k^j es procesada y $p_k^j = p(\sigma_k^j)$ es el tiempo de procesamiento de la operación σ_k^j. Definiendo $t(\sigma_k^j)$ como el tiempo de comienzo de la k-ésima operación $\sigma_k^j \in \mathcal{O}$, el JSP puede formularse como sigue:

$Minimizar\ C_{max}$

$Sujeto\ a$:

$$C_{max} \geq t(\sigma_k^j) + p(\sigma_k^j)\ \forall \sigma_k^j \in \mathcal{O},$$

$$t(\sigma_k^j) \geq t(\sigma_l^j) + p(\sigma_l^j)\ \forall \sigma_l^j \prec \sigma_k^j$$

$$t(\sigma_k^j) \geq t(\sigma_l^i) + p(\sigma_l^i)\ \vee$$

$$t(\sigma_l^i) \geq t(\sigma_k^j) + p(\sigma_k^j)\ \forall \sigma_l^j, \sigma_k^j \in \mathcal{O}\ tal\ que\ \mathcal{M}_{\sigma_l^i} = \mathcal{M}_{\sigma_k^j}$$

$$t(\sigma_k^j) \geq 0\ \forall \sigma_k^j \in \mathcal{O}$$

donde C_{max} es el máximo tiempo de terminación de los trabajos (*makespan*) para ser minimizado.

Una solución factible del JSP puede ser construida por una permutación de $\mathcal{J}$ sobre cada una de las máquinas en $\mathcal{M}$, teniendo en cuenta las siguientes restricciones:

- De precedencia, es decir, el orden de las operaciones sobre las máquinas es preespecificado, las operaciones de un trabajo dado tienen que ser procesadas en un orden dado.
- Cada operación usa una de las $|\mathcal{M}|$ máquinas por una duración fija.
- Cada máquina puede procesar una operación a la vez, y en cuanto una operación inicia el procesamiento sobre una máquina dada, debe completar el procesamiento sobre esa máquina sin interrupción.
- Ni la fecha de liberación ni la fecha de entrega son especificados.
- Un trabajo no visita la misma máquina dos veces.
- No hay restricciones de precedencia entre las operaciones de diferentes trabajos.

3. METODOLOGÍA AGM

El algoritmo genético es una técnica de optimización combinatoria que hace parte de la familia de las técnicas evolutivas propuestas en la década de los cincuenta. Se caracteriza por realizar en su estructura la reproducción, variaciones aleatorias, promoción de la competencia y selección de individuos de una población dada. Desde su aparición, los algoritmos genéticos han sido objeto de numerosas modificaciones: cuando la función objetivo es de minimización, cuando aparecen configuraciones "infactibles", formas de codificación, tipos de selección, tipos de recombinación, propuestas alternativas de selección, entre otros (Gallego, Escobar y Romero, 2006).

En el AGM se realiza una propuesta híbrida, en donde la etapa de mutación se lleva a cabo mediante una estrategia de enfriamiento simulado, que se detallará en la sección 3.6. El seudocódigo del AGM aplicado al JSP se muestra en la figura 1.

```
Inicio
For i=1 to experimentos
   Generación_población_inicial_G&T ()
   For ii=1 to generaciones
        Selección_por_torneo ()
        Cruce_Re_insert ()
        Mutación_Enfriamiento_Simulado ()
        Reemplazo de la sub-población ()
   Next ii
Next i
```

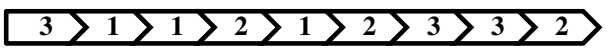
Figura 1. Seudocódigo del AGM aplicado al JSP

3.1 Codificación del cromosoma

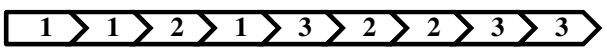
Cada cromosoma tiene información para solucionar el problema. El método de codificación que se utilizó para solucionar el problema de

programación es la permutación (Bierwirth, Mattfeld y Kopfer, 1996; Park, Choi y Kim, 2003; Watanabe, Ida y Gen, 2005). Se empleó una representación basada en las operaciones que usa una permutación con $\mathcal{M}$ repeticiones del número de trabajos, por lo tanto, cada trabajo ocurre $\mathcal{M}$ veces en la permutación, tan frecuentemente como operaciones haya asociadas con él. Este método de codificación puede crear un programa activo.

El cromosoma muestra el orden del número de trabajos. Si el número de trabajos es $|\mathcal{J}|$ y el número de máquinas es $|\mathcal{M}|$, el cromosoma consta de $|\mathcal{J}| * |\mathcal{M}|$ genes. Escaneando la permutación de izquierda a derecha, la *k*-ésima ocurrencia de un número de trabajo se refiere a la *k*-ésima operación en su secuencia tecnológica. Una permutación con repetición de números de trabajos expresa el orden en el cual se programan las operaciones de los trabajos. Suponga el cromosoma dado en la figura 2, para un problema de 3 trabajos x 3 máquinas, cada trabajo consta de 3 operaciones y, por lo tanto, se repite tres veces. El tercer gen del cromosoma en este ejemplo es 1, esto implica que es la segunda operación del trabajo 1, puesto que el trabajo 1 ya ha sido repetido dos veces. Si el número de trabajos es repetido como el número de operaciones, el cromosoma es siempre factible.



a. Secuencia de trabajos para programar



b. Índice de ocurrencia del trabajo

Figura 2. Codificación del cromosoma

3.2 Generación de la población inicial

Para generar la población inicial se empleó el algoritmo G&T (Giffler y Thompson, 1960; Park, Choi y Kim, 2003). El cromosoma es creado grabando el número del trabajo de la operación seleccionada acorde al orden de selección. Los programas de los individuos de la población inicial varían de acuerdo con los tres procedimientos para obtener programas (activo, no retraso, activo).

3.3 Elección de subpoblación

La selección de una subpoblación es una propuesta novedosa que busca generar buenos resultados en algoritmos genéticos aplicados al JSP. En problemas de programación de pedidos, como el *Flow Shop*, la aplicación de esta estrategia ha demostrado generar buenos resultados, debido a que puede producir una gran diversificación en el transcurso del algoritmo, evitando que éste caiga en óptimos locales y no alcance soluciones de buena calidad (Toro, Restrepo y Granada, 2006). Dicho procedimiento se hace de manera aleatoria sobre la población inicial y se guardan las posiciones que la subpoblación de cromosomas tiene respecto a la población. El número de cromosomas en la subpoblación constituye un parámetro por optimizar del AGM.

3.4 Selección por torneo

Consistió en escoger aleatoriamente un número de cromosomas j de la subpoblación sin "reemplazamiento". Luego se escoge el mejor individuo de ese grupo. Este proceso se realiza n veces, donde n es el tamaño de la subpoblación. El tamaño del grupo de torneo es fijo y es otro de los parámetros para validar en el AGM. La principal característica que hace atractiva la estrategia de selección por torneo es su rapidez en esfuerzo computacional. Cada juego en el torneo requiere la selección aleatoria de un número constante de individuos de la subpoblación.

3.5 Estrategia de recombinación o cruce

Para la selección de los individuos que se cruzan se utilizó una probabilidad de cruce y, mediante un generador de número aleatorios entre 0 y 1, se escoge cuáles individuos van a ser cruzados.

Para el cruce se utilizó la estrategia *Re-insert*, basada en el trabajo de Duncan (1995). La estrategia de *Re-insert* consiste en seleccionar dos posiciones i e $i + 1$ en el cromosoma e introducir un trabajo k, seleccionado aleatoriamente diferente a las dos posiciones seleccionadas al comienzo, dentro de i e $i + 1$. Esta estrategia de cruce mantiene la factibilidad del cromosoma, debido a que al introducir un trabajo entre otros dos, todos los trabajos se siguen repitiendo el mismo número de veces (ver figura 3).

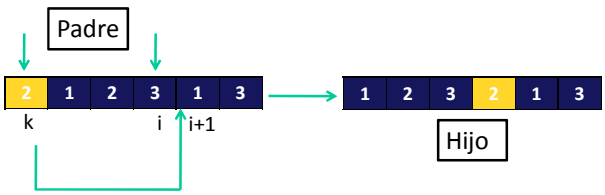


Figura 3. Estrategia de cruce *Re-insert*

3.6 Estrategia de mutación

La fase de mutación era considerada como un operador genético secundario en los algoritmos genéticos. Sin embargo, las investigaciones más recientes le están dando una importancia mucho mayor, en especial en aplicaciones a problemas reales de gran tamaño (Gallego, Escobar y Romero, 2006).

A partir de los aportes metodológicos propuestos en la literatura (Wang y Zheng, 2001), para la mutación del AGM se utilizó una estrategia basada en el enfriamiento simulado. La implementación de esta estrategia consiste en seleccionar un gen para mutar con respecto a una probabilidad de mutación. Si un gen fue seleccionado para mutar, se procede a utilizar una estrategia de enfriamiento simulado que aplica *Re-insert* como estrategia de vecindad.

Todas las movidas de intercambio creadas por las N iteraciones del algoritmo se evalúan para la estrategia de vecindad. Cada vecino generado en las N iteraciones es comparado contra la solución actual; si es mejor, actualiza la solución actual y la compara contra la mejor solución encontrada hasta el momento y la actualiza si la mejora. Si el mejor vecino no es mejor que la solución actual, entonces con una probabilidad de enfriamiento dada, acepta soluciones que no son tan buenas para diversificar la búsqueda.

3.7. Estrategia de modificación de la población

La estrategia de modificación de la población es sencilla, como quiera que cada vez que se toma la subpoblación se guarda la posición de cada cromosoma, entonces se compara el cromosoma inicial de la población con el cromosoma de la subpoblación después del proceso de evolución, y si mejora, éste se reemplaza así como su función objetivo, y si no, se deja igual en la población.

4. METODOLOGÍA DEL ESP

El ESP es una metaheurística que está inspirada en la idea del enfriamiento de los metales. Se basa en una técnica de búsqueda aleatoria regulada por el seguimiento de un parámetro de control denominado temperatura, donde es posible obtener soluciones aproximadas a problemas combinatorios de optimización. El ESP se compone de dos fases: una fase de exploración en la que, a temperaturas altas, el recocido simulado acepta soluciones malas con altas probabilidades, para salir del óptimo local; y una fase de explotación, en la que a temperaturas bajas, el recocido simulado explota las soluciones encontradas en la fase de altas temperaturas y explora localmente para encontrar el óptimo (Gallego, Escobar y Romero, 2006).

El ESP utiliza la misma representación y generación del cromosoma utilizadas por el algoritmo genético. La estrategia de vecindad usada es un intercambio tipo *Swap* entre dos trabajos diferentes. Se evalúan N movidas de intercambio, cada vecino generado en las N iteraciones se compara contra la solución actual; si es mejor, actualiza la solución actual y la compara contra la mejor solución encontrada hasta el momento, y actualiza ésta, si la mejora. Si el mejor vecino no es mejor que la solución actual,

entonces con una probabilidad de enfriamiento, acepta soluciones que no son tan buenas para diversificar la búsqueda.

5. BÚSQUEDA LOCAL

El AGM y el ESP emplean una heurística de búsqueda local, con el propósito de mejorar las soluciones encontradas. En el AGM la búsqueda local se le aplica al mejor valor encontrado en cada generación, solo si el mejor valor encontrado en la generación es mejor que el mejor valor encontrado actual en las generaciones corridas. En otras palabras, la búsqueda local se aplica sólo si se encuentra un mejor valor que el que se tenía hasta el momento.

La estrategia funciona aplicando una movida tipo *Swap*, intercambiando dos trabajos diferentes. La búsqueda aplica a todas las posibles movidas tipo *Swap* que se puedan hacer a partir del mejor valor encontrado, de forma que escoge el mejor vecino de todos los posibles intercambios y lo compara con el mejor valor encontrado. Si es mejor, actualiza el mejor valor encontrado y realiza una nueva búsqueda, pero ahora sobre el nuevo mejor valor encontrado.

Para el ESP, la estrategia de búsqueda local es similar a la utilizada por el AGM, con la modificación de que sólo se aplica si el mejor vecino mejora la mejor solución actual, es decir, se le aplica al mejor valor encontrado cada vez que éste se actualice.

6. VALIDACIÓN DE LOS ALGORITMOS PROPUESTOS

Para validar el AGM se diseñó un experimento para cada una de las instancias evaluadas (experimento factorial de 2x2). Debido a que el AGM tiene varios parámetros por ajustar, se realizaron una serie de pruebas para encontrar los parámetros óptimos para cada problema teniendo en cuenta la media, la desviación y el tiempo computacional. Cada experimento contó con 30 repeticiones, en las que se analizó la influencia de dos variables independientes en los resultados de los algoritmos. En el diseño experimental, primero se buscaron los mejores valores de las variables independientes probabilidad de cruce y probabilidad de mutación; y luego de encontrar los valores óptimos, se cruzaron las variables de la estrategia de selección y el tamaño de la subpoblación. Por último, se realizó el mismo procedimiento para encontrar el número óptimo de generaciones y el tamaño de la población, lo anterior con el propósito de encontrar los parámetros óptimos del problema para cada instancia.

Igualmente, para validar el ESP se analizaron los parámetros de la temperatura inicial y final, buscando que a altas temperaturas el enfriamiento simulado se diversificara, aceptando soluciones malas con altas probabilidades; y que a bajas temperaturas explotara esas soluciones encontradas a altas temperaturas, aceptando con una baja probabilidad pocas soluciones malas.

Las probabilidades encontradas para altas temperaturas fueron del orden de 0,95 a 0,99 y a medida que iba bajando la temperatura la probabilidad disminuía, hasta llegar, por ejemplo, al orden de 6,98072E-05. Una vez que estas temperaturas cumplían con el criterio del ESP, eran aceptadas para continuar con el análisis.

7. RESULTADOS COMPUTACIONALES

Los resultados de la aplicación de los algoritmos AGM y el ESP para cada una de las instancias corridas del JSP se observan en la tabla 1. Para evaluar la efectividad de ambos métodos se consideraron las instancias más corridas en la literatura (Fisher y Thompson, 1963; Lawrence, 1984), que presentan diferentes niveles de complejidad, y los resultados se compararon con diversos algoritmos propuestos para solucionar el JSP (ver tabla 2). Los algoritmos se implementaron en la macro de Visual Basic de Excel y las pruebas fueron corridas sobre un computador con procesador Intel Pentium Dual-Core de 2,4 GHz + 4 GB de RAM.

Para la comparación de los dos métodos propuestos se utilizó una prueba de hipótesis de diferencia de medias, varianza conocida con muestras independientes y grandes (mayor de 25 elementos), con una hipótesis alternativa unilateral izquierda para cada una de las instancias corridas por los dos algoritmos.

$$H_0: \mu_{ES} - \mu_{AGM} = 0$$
$$H_1: \mu_{ES} - \mu_{AGM} < 0$$

Se empleó la ecuación (1) para el cálculo del estadístico de prueba Z_{cal}, el cual frente al estadístico tabulado Z_{tab}, encontrado en las tablas de la distribución normal estándar para pruebas de una cola, arroja los resultados expuestos en la tabla 3.

$$Z_{cal} = Z_{cal} = \frac{\overline{x}_{ES} - \overline{x}_{AGM}}{\sqrt{\frac{\delta_{ES}^2}{n_{ES}} + \frac{\delta_{AGM}^2}{n_{AGM}}}} \quad (1)$$

8. ANÁLISIS DE RESULTADOS

Del diseño de experimentos realizado para encontrar los parámetros óptimos del algoritmo genético mejorado, se puede analizar que el AGM no requiere grandes tamaños de población (P<100) para encontrar buenos resultados. El tamaño de la subpoblación y el tamaño del torneo varían de acuerdo con la complejidad del problema; sin embargo, se puede analizar que para el 65 % de las instancias corridas el tamaño del torneo no debe ser mayor de 5 individuos. Un aspecto interesante para resaltar se puede observar en la probabilidad de cruce, donde casi el 80 % de las instancias corridas alcanzaron los mejores resultados con una alta tasa de cruce, esto nos indica que la estrategia de cruce Re-insert es crucial para el desempeño del AGM; caso contrario se observa con la probabilidad de mutación, que no muestra ninguna tendencia significativa, pero sí evidencia su importancia para encontrar los mejores resultados. También se puede observar que el AGM requiere un gran número de generaciones para alcanzar las mejores soluciones, lo que afecta directamente el desempeño del algoritmo en cuanto al consumo de tiempo computacional (ver tabla 1).

Por otro lado, cuando se empleó la heurística de búsqueda local, que se realiza al final de los dos algoritmos propuestos, no genera diferencias significativas en el desempeño de ellos, en términos de calidad de las soluciones, aunque sí está generando un mayor consumo de tiempo computacional. Sólo en 2 de las 14 instancias corridas por el AGM y ESP la heurística contribuyó al mejoramiento de la calidad de las configuraciones obtenidas (ver tabla 1).

Otro aspecto importante para resaltar es el buen desempeño del AGM y ESP: cada instancia de problema fue corrida 30 veces, presentándose una baja desviación estándar con respecto al *makespan* promedio para las 30 corridas. Esto nos indica que los algoritmos propuestos tienen una alta confiabilidad para alcanzar buenos resultados. En particular se puede destacar que, para las instancias de baja y media complejidad (FT06, LA01, LA06, LA09, LA11, LA14, LA31), los algoritmos propuestos siempre alcanzan los mejores valores reportados en la literatura, y para las instancias de alta complejidad (FT10, FT20, LA16, LA21, LA26, LA27, LA36) los resultados alcanzados (*makespan* promedio de las 30 corridas) por los algoritmos propuestos presentan una desviación estándar máxima del 5 % con respecto a los mejores valores reportados hasta el momento en la literatura (ver tablas 1 y 2).

Al analizar los resultados de la tabla 2 se puede observar que el AGM presenta poca eficiencia computacional para instancias de problemas de alta complejidad; los problemas que involucran 15 o más trabajos con 10 máquinas (LA21, LA26 y LA 31) pueden llegar a tomar tiempos de ejecución de más de 20 minutos. Este algoritmo es más sensible frente al crecimiento del número de máquinas, es decir, genera mejores soluciones en la medida que el número de máquinas es menor.

Tabla 1. Parámetros óptimos y resultados del AGM y del ESP para cada instancia

Instancia	Algoritmo genético mejorado												Enfriamiento simulado propuesto							
	Parámetros óptimos							Resultados					Parámetros			Resultados				
	P	*Sub*	*Torneo*	*Pc*	*Pm*	*G*	*BL*	μ	*s*	*MS*	*DE*	*Tp*	*TI*	*E*	*BL*	μ	*s*	*MS*	*DE*	*Tp*
FT06	30	24	3	0,75	0,10	2000	No	55,6	0,9	55	0,00	18,0	10.000	0,99	Sí	55	0,00	55	0,00	11,6
FT10	50	50	5	0,75	0,10	1000	No	989,5	18,9	945	1,59	45,6	10.000	0,99	No	976,8	17,3	950	2,11	33,6
FT20	50	50	10	0,75	0,70	2000	No	1210,1	21	1179	1,19	484,1	10.000	0,99	Sí	1209,1	25,2	1178	1,10	53,9
LA01	100	20	3	0,75	0,20	2000	No	666	0,0	666	0,00	60,7	10.000	0,99	No	666	0,0	666	0,00	18,0
LA06	30	15	5	0,75	0,70	500	No	926	0,0	926	0,00	31,1	10.000	0,90	No	926	0,0	926	0,00	31,0
LA09	30	30	5	0,75	0,10	100	No	951	0,0	951	0,00	2,5	10.000	0,90	No	951	0,0	951	0,00	2,9
LA11	100	80	5	0,75	0,50	100	No	1222	0,0	1222	0,00	30,8	100	0,90	No	1222	0,0	1222	0,00	2,2
LA14	30	6	3	0,75	0,10	100	No	1292	0,0	1292	0,00	2,1	10.000	0,90	No	1292	0,0	1292	0,00	4,4
LA16	100	20	10	0,75	0,50	2000	No	991,7	13,9	959	1,46	162,5	10.000	0,99	No	980,9	11,6	946	0,11	33,3
LA21	100	100	10	0,50	0,70	2000	Sí	1112,5	18,7	1079	3,06	1194,3	10.000	0,90	Sí	1106,5	20	1068	2,06	60,3
LA26	100	100	10	0,75	0,70	2000	Sí	1253,7	18,3	1219	0,08	2212,6	100.000	0,99	No	1268,3	20,4	1221	0,25	99,9
LA27	100	50	5	0,75	0,10	2000	No	1326,5	27,5	1284	3,82	299,1	1.000.000	0,99	No	1317	17,6	1286	3,97	155,3
LA31	100	100	3	0,50	0,50	2000	No	1785,7	7,0	1784	0,00	2101,4	100.000	0,99	No	1784	0,0	1784	0,00	179,0
LA36	100	100	10	0,50	0,20	2000	No	1383,9	33,1	1323	4,16	581,4	100.000	0,99	No	1358,4	21,8	1316	3,65	97,2

P: Tamaño de población
Sub: Tamaño de la subpoblación
Pc: Probabilidad de cruce
Pm: probabilidad de mutación
G: Generaciones
BL: Búsqueda local
μ: valor promedio del makespan para 30 corridas
s: Desviación estándar con respecto al makespan para 30 corridas
MS: Mejor valor objetivo alcanzado
DE: Desviación con respecto al valor óptimo conocido hasta ahora (%)
Tp: Tiempo promedio de corrida (segundos)
TI: Temperatura inicial
E: Factor de enfriamiento

Tabla 2. Resultados experimentales

Instancia			Genetic algorithms								GRASP		Tabu search	Hybrid genetic and simulate annealing		Memetic algorithm		Ant colony optimization combined with taboo search				AGM		ESP	
			Della Croce, Tadei y Volta (1995)		Dorndorf y Pesch (1995)			Gon-çalves y Beirão (1999)	Gonçalves, Mendes y Resende (2005)		Binato *et al.* (2002)	Aiex, Binato y Resende (2003)	Nowicki y Smutnicki (1996)	Wang y Zheng (2001)		Yang *et al.* (2008)		Huang y Liao (2008)							
					P-GA (1995)	SBGA (1995)	SBGA (1995)											ACOFT-MWR		ACOFT-TR					
	nxm	*MS***	*MS*	*Tp*	*MS*	*MS*	*MS*	*MS*	*MS*	*Tp*	*MS*	*MS*	*MS*	*MS*	*Tp*	*MS*	*Tp*	*MS*	*Tp*	*MS*	*Tp*	*MS*	*Tp*	*MS*	*Tp*
FT06	6x6	55	55	223				55	55	13	55	55	55	55	2,1	55	0,1					55	18	55	11,6
FT10	10x10	930	946	628	960			936	930	292	938	930	930	930	44,2	930	81,6	930	64,6	930	118,2	945	45,6	950	33,6
FT20	20x5	1165	1178	675	1249			1177	1165	204	1169	1165	1165	1165	90,2	1165	134,8	1165	46	1165	22,3	1179	484,1	1178	53,9
LA01	10x5	666	666	282	666	666		666	666	37	666	666	666	666	5,72	666	0,2					666	60,7	666	18
LA06	15x5	926	926	473	926	926		926	926	99	926	926	926	926	12,3	926	0,1					926	31,1	926	31
LA09	15x5	951			951	951		951	951	94	951	951	951			951	0,2					951	2,5	951	2,9
LA11	20x5	1222	1222	717	1222	1222		1222	1222	197	1222	1222	1222	1222	25,9	1222	0,2					1222	30,8	1222	2,2
LA14	20x5	1292			1292	1291		1292	1292	187	1292	1292	1292			1292	0,1					1292	2,1	1292	4,4
LA16	10x10	945	979	637	1008	961	961	977	945	232	945	945	945	945	29,4	945	78					959	162,5	946	33,3
LA21	15x10	1046	1097	1062	1139	1074	1074	1047	1046	602	1091	1057	1047	1046	184,5							1079	1194,3	1068	60,3
LA26	20x10	1218	1231	1542	1278	1219	1218	1218	1218	1388	1271	1218	1218	1218	417,9							1219	2212,6	1221	99,9
LA27	20x10	1235			1378	1272	1269	1260	1256	1251	1320	1269	1236					1235	385,6	1235	1035,6	1284	299,1	1286	155,3
LA31	30x10	1784	1784	2762				1784	1784	3745	1784	1784	1784	1784	546,2							1784	2101,4	1784	179
LA36	15x15	1268	1305	1880	1373	1317	1317	1305	1279	1826	1334	1287	1268	1268	348,1			1268	36,6	1268	123,5	1323	581,4	1316	97,2

n: Número de trabajos; *m*: Número de máquinas

*MS***: Valor óptimo conocido en la literatura hasta ahora

MS: Mejor valor objetivo alcanzado

Tp: Tiempo promedio de corrida (segundos)

Caso contrario se presenta con el ESP, el cual es un algoritmo con una alta eficiencia computacional. Al comparar el ESP con otros algoritmos propuestos en la literatura y con el AGM, se puede observar que el ESP es eficiente en el consumo de tiempo computacional; para las instancias de mayor complejidad alcanza buenos resultados en menos de 3 minutos. Es importante aclarar que para estos tiempos de procesamiento tan cortos, el hardware del computador donde se corrieron los algoritmos deja de ser una variable significativa para comparar la eficiencia del algoritmo con otros algoritmos recientes.

Por último, se compararon los resultados obtenidos por el AGM y por el ESP para cada una de las instancias corridas. Los resultados de las pruebas de hipótesis realizadas evidenciaron que el 28,57 % de las veces (FT06, FT10, LA16, LA 36) Z_{cal} fue menor que el Z_{tab}, es decir, el ESP encontró mejores resultados que el AGM. Para un 64,28 % de los casos (FT20, LA01, LA06, LA09, LA11, LA14, LA21, LA27, LA 31) el AGM y el ESP presentaron los mismos resultados con un nivel de confianza del 5 %, sin embargo, el ESP empleó un menor tiempo computacional que el AGM. Solo 7,14 % de las veces (LA26), el AGM encontró mejores resultados que el EFP. Es importante analizar que para 8 de las instancias probadas con los algoritmos propuestos, el ESP convergió más rápido y el *makespan* estuvo más cerca del mejor valor conocido hasta ahora en la literatura (ver tabla 3). Los resultados evidencian que el ESP obtiene mejores resultados tanto en tiempo computacional como en la calidad de sus respuestas.

Tabla 3. Cálculo de los estadísticos de prueba para cada uno de las instancias de prueba

Instancia	MS**	ESP		AGM		Nivel de confianza: 95 %	
		μ	S	μ	S	Z_{cal}	Z_{tab}
FT06	55	55,00	0,00	55,60	0,93	-3,53	-1,64
FT10	930	976,80	17,25	989,50	18,93	-2,72	-1,64
FT20	1165	1.209,13	25,20	1210,10	20,97	-0,16	-1,64
LA01	666	666,00	0,00	666,00	0,00	0,00	-1,64
LA06	926	926,00	0,00	926,00	0,00	0,00	-1,64
LA09	951	951,00	0,00	951,00	0,00	0,00	-1,64
LA11	1222	1.222,00	0,00	1222,00	0,00	0,00	-1,64
LA14	1292	1.292,00	0,00	1292,00	0,00	0,00	-1,64
LA16	945	980,87	11,62	991,73	13,92	-3,28	-1,64
LA21	1046	1.106,53	19,95	1112,53	18,68	-1,20	-1,64
LA26	1218	1.268,27	20,35	1253,73	18,25	+2,91	-1,64
LA27	1235	1.317,03	17,62	1326,50	27,53	-1,59	-1,64
LA31	1784	1784,00	0,00	1785,73	7,03	-1,35	-1,64
LA36	1268	1.358,43	21,76	1383,87	33,05	-3,52	-1,64

*MS***: Valor óptimo conocido en la literatura hasta ahora

μ: valor promedio del *makespan* para 30 corridas

S: Desviación con respecto a la media (en %)

9. CONCLUSIONES

Se concluye que los dos algoritmos propuestos (AGM y ESP) son opciones eficientes para solucionar el JSP, encontrando óptimos globales. Son algoritmos que obtienen buenos resultados para las instancias de problemas evaluadas, aunque se demuestra que el ESP encuentra mejores configuraciones que el AGM, en términos de la calidad de las configuraciones obtenidas y el tiempo computacional requerido.

Más allá de esto, se considera que los algoritmos analizados son efectivos, en la medida en que:

- Arrojan buenos resultados y encuentran fácilmente las configuraciones óptimas de sistemas de baja y media complejidad matemática. Para problemas más complejos, ambos obtienen resultados con unas desviaciones promedio menores del 3 % de los mejores valores reportados.
- Los algoritmos propuestos son bastante confiables, para sistemas de baja y media complejidad encuentran el 100 % de las veces el mejor valor reportado (el valor medio del *makespan* para 30 corridas es igual al mejor valor reportado en la literatura), y para los sistemas de alta complejidad presentan desviaciones estándar muy bajas con relación al valor medio del *makespan* para 30 corridas.
- Tienen en cuenta la aplicación de heurísticas en la generación de la población inicial para converger más rápido al óptimo.
- El ESP es eficiente en el consumo de tiempo computacional para converger al óptimo, en comparación con los demás algoritmos propuestos en la literatura y el AGM.
- Los algoritmos son capaces de generar muchas soluciones de calidad para el caso de múltiples óptimos.
- La metodología implementada es una buena alternativa de solución para problemas de naturaleza combinatoria, por la facilidad de adaptación.

REFERENCIAS

Aiex, R. M.; Binato, S. and Resende M. G. C. (2003). "Parallel GRASP with path-relinking for job shop scheduling". *Parallel Computing* 29, pp. 393-430.

Azizi, Nader and Zolfaghari, Saeed (2004). "Adaptive temperature control for simulated annealing: a comparative study". *Computers & Operations Research*, vol. 31, pp. 2439-2451.

Bierwirth, C.; Mattfeld D. and Kopfer H. (1996). "Proceedings of parallel problem solving from Nature IV". *Springer*, pp. 310-318.

Binato, S.; Hery, W. J.; Loewenstern, D. M. and Resende, M. G. C. A GRASP for job shop scheduling. In: Ribeiro, C. C., Hansen, P. (eds.). *Essays and Surveys in Metaheuristics*. Kluwer Academic Publishers, 2002.

Della Croce, Federico; Tadei, Roberto and Volta, Giuseppe (1995). "A genetic algorithm for the job shop problem". *Computers & Operations Research,* vol. 22, No. 1, pp. 15-24.

Dorndorf, U. and Pesch, E. (1995). "Evolution based learning in a job shop environment". *Computers and Operations Research*, vol. 22, No. 1 (January), pp. 25-40.

Duncan, Tim. *Neighbourhood search and the vehicle routing problem*. Artificial Intelligence Applications Institute, 1995.

Fisher, H. and Thompson G. L. "Probabilistic learning combinations of local job-shop scheduling rules". In: Muth J. F and Thompson, G. L. *Industrial scheduling* (eds.). Englewood Cliffs, NJ: Prentice Hall,1963, pp. 225-251.

Gallego, Ramón Alfonso; Escobar, Antonio Hernando y Romero, Rubén Augusto. *Técnicas de optimizacion combinatorial*. Pereira: Textos Universitarios, 2006.

Garey, E. L.; Johnson, D. S. and Sethi, R. (1976) "The complexity of flow shop and job shop scheduling". *Mathematics of Operations Research*, vol. 1, No. 2 (May), pp. 117-129.

Giffler, B. and Thompson G. L. (1960). "Algorithms for solving production-scheduling problems". *Operations Research*, vol. 8, No. 4 (Jul.-Aug.), pp. 487-503.

Gonçalves, José Fernando; Mendes, Jorge José and Resende, Mauricio G. C. (2005). "A hybrid genetic algorithm for the job shop problem". *European Journal of Operational Research*, vol. 167 (November.), pp. 77-95.

Gonçalves, J. F. e Beirão, N. C. (1999). "Um algoritmo genético baseado em chaves aleatórias para sequenciamento de operações". *Revista Associação Portu-*

guesa de Desenvolvimento e Investigação Operacional, vol. 19, pp. 123-137.

Huang, Kuo-Ling and Liao, Ching-Jong (2008). "Ant colony optimization combined with taboo search for the job shop scheduling problem". *Computers & Operations Research*, vol. 35, pp. 1030-1046.

Krajewski, L. J. and Ritzman L. P. *Administración de operaciones: estrategias y análisis*. 5 ed. México: Pearson Educación, 2000. 892 p.

Lawrence, S. "*Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques",* Graduate School of Industrial Administration, Pittsburgh, PA. Carnegie Mellon University, 1984.

Nowicki, E. and Smutnicki, C. (1996). "A fast taboo search algorithm for the job-shop problem". *Management Science*, vol. 42 (6), pp. 797-813.

Park, Byung Joo; Choi, Hyung Rim and Kim, Hyun Soo (2003). "A hybrid genetic algorithm for the job shop scheduling problems". *Computers & Industrial Engineering*, vol. 45, pp. 597-613.

Toro, Eliana; Restrepo, Yov y Granada, Mauricio. (2006). Algoritmo genético modificado aplicado al problema de secuenciacion de tareas en sistemas de produccion lineal flow shop. *Scientia et Technica*, año 7, vol. 30 (mayo), pp. 285-290.

Vilcot, Geoffrey and Billaut, Jean-Charles (2008). "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem". *European Journal of Operational Research*, vol. 190, pp. 398-411.

Wang, Ling and Zheng, Da-Zhong (2001). "An efective hybrid optimization strategy for job-shop scheduling problems". *Computers & Operations Research*, vol. 28, No. 6 (May), pp. 585-596.

Watanabe, Masato; Ida Kenichi and Gen Mitsuo (2005). "A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem". *Computers & Industrial Engineering*, vol. 48, pp.743-752.

Yang, Jin-hui; Sun, Liang; Lee, Heow Pueh; Qian, Yun and Liang, Yan-chun (2008). "Clonal selection based memetic algorithm for job shop scheduling problems" *Journal of Bionic Engineering,* vol. 5, No. 2 (June), pp. 111-119.

Zhang, Rui and Wu, Cheng (2010). "A hybrid immune simulated annealing algorithm for the job shop scheduling problem". *Applied Soft Computing*, vol. 10, No. 1 (January), pp. 79-89.

Zhou, Hong; Feng, Yuncheng and Han, Limin (2001). "The hybrid heuristic genetic algorithm for job shop scheduling". *Computers & Industrial Engineering*, vol. 40, pp. 191-200.