Entre Ciencia e Ingeniería, vol. 12, no. 24, julio-diciembre de 2018, página 105 - 115 DOI: http://dx.doi.org/10.31908/19098367.3820
ISSN 1909-8367 (Impreso), ISSN 2539-4169 (En línea)

Optimización del makespan en el problema de Job Shop Flexible con restricciones de transporte usando Algoritmos Genéticos[1]

Makespan optimization in the flexible Job Shop problem with transportation constraints using Genetic Algorithms

Otimização da Makespan no problema flexível da Job Shop com restrição de transporte usando Algoritmos Genéticos

T. A. Castillo, C. E. Díaz B, J. D. Gómez , E. A. Orduz y M. L. Niño

Recibido: marzo 27 de 2018 - Aceptado: junio 29 de 2018

***Resumen*— En el presente artículo se aborda el problema Flexible Job Shop Scheduling (FJSSP) con restricciones de transporte, con el objetivo de minimizar el *makespan* realizando la secuenciación y asignación de máquinas. Se llevó a cabo una revisión bibliográfica para orientar la metodología a utilizar, y a partir de allí, se decidió abordar el problema con un algoritmo genético, validando su efectividad a través de la comparación de los resultados obtenidos con distintas instancias propuestas en la literatura. Los resultados obtenidos muestran que el algoritmo genético propuesto es eficiente en las diferentes configuraciones del Job Shop clásico probadas, y para el Job Shop flexible con restricciones de transportes se presentan soluciones muy aproximadas a las mejores encontradas hasta el día de hoy.**

***Palabras clave*— Algoritmo genético, AG, Job Shop, Job Shop Flexible, Job Shop Flexible Restricciones de transporte, Metaheurísticas, Makespan, Restricciones de transporte.**

***Abstract*— We solved the Flexible Job Shop Scheduling Problem (FJSSP) with transportation constraints in order to minimize the *makespan* by sequencing and assigning machines. A bibliographic review was made in order to guide the methodology to be used. From there, we approached the problem with a genetic algorithm. We validated its effectiveness by comparing the results obtained with different instances proposed in the literature. The results obtained show that the proposed genetic algorithm is efficient in the different classic Job Shop configurations tested. The algorithm is able to find very approximate solutions to the best found to date for the Flexible Job Shop Scheduling Problem with transportation constraints.**

***Keywords*— Job Shop, Flexible Job Shop, Flexible Job Shop with transportation constraints, Transportation constraints, Genetic Algorithm, GA, Metaheuristics, Makespan.**

***Resumo*— Resolvemos o Problema de Agendamento de Oficina de Trabalho Flexível (FJSSP) com restrições de transporte para minimizar o makespan por sequenciamento e atribuição de máquinas. Uma revisão bibliográfica foi realizada para orientar a metodologia a ser utilizada. A partir daí, abordamos o problema com um algoritmo genético. Nós validamos sua eficácia comparando os resultados obtidos com diferentes instâncias propostas na literatura. Os resultados obtidos mostram que o algoritmo genético proposto é eficiente nas diferentes configurações clássicas de Job Shop testadas. O algoritmo é capaz de encontrar soluções muito aproximadas**

[1]Producto derivado del proyecto de grado en investigación "Minimización del makespan en el problema de Job Shop Flexible con restricciones de transporte utilizando Algoritmo Genético". Presentado por el Grupo de Investigación ÓPALO, de la Universidad Industrial de Santander.

T. A. Castillo - Jaimes, Universidad Industrial de Santander, Bucaramanga, Colombia; email: tatiana.castillo@correo.uis.edu.co

C. E. Díaz Bohórquez, Universidad Industrial de Santander, Bucaramanga, Colombia; email: cediazbo@uis.edu.co

J. D. Gómez Moreno, Avianca, Bogotá, Colombia; email: gomezjuandavid3@gmail.com

E. A. Orduz González, IBM, Bogotá, Colombia; email: orduzg.edwin@gmail.com

M. L. Niño López, Universidad Industrial de Santander, Bucaramanga, Colombia; email: myleni@uis.edu.co

Como citar este artículo: Castillo, T. A., Díaz, C. E., Gómez, J. D., Orduz, E. A. y Niño, M. L. Optimización del makespan en el problema Job Shop Flexible con restricciones de transporte usando algoritmos genéticos, Entre Ciencia e Ingeniería, vol 12, no. 24, pp. 105-115, julio-diciembre 2018.
DOI: http://dx.doi.org/10.31908/19098367.3820

para o melhor encontrado até o momento para o Problema de Agendamento de Oficina de Trabalho Flexível com restrições de transporte.

***Palabras chave*— Job Shop, flexível Job Shop, flexível Job Shop com restrições de transporte, restrições de transporte, algoritmo genético, GA, Metaheuristics, Makespan.**

I. Introducción

En la actualidad, la optimización de los recursos productivos juega un papel importante en la planeación de la producción dentro de las organizaciones. Una adecuada programación dentro del área de producción que involucre la asignación de máquinas y la secuenciación de operaciones dentro de las mismas, es un proceso valioso dentro de los sistemas de producción. Por tal razón, [1] [4], entre otros, han unido esfuerzos para estudiar un problema que describe este tipo de situaciones, el Job Shop flexible.

El presente artículo analiza el problema de la programación, asignación y secuenciación de máquinas para la ejecución de múltiples trabajos, conocido como Job Shop flexible (FJSSP por sus siglas en inglés), incluyéndole recursos y operaciones de transporte. El FJSSP es un problema de optimización combinatoria de tipo *NP-Hard* [5].

De acuerdo con lo anterior, se utiliza la metaheurística de algoritmo genético propuesta por [6], un método establecido como una adaptación algorítmica de la evolución biológica a la inteligencia artificial.

II. Trabajos relacionados

No se confirma con seguridad quién propuso el JSSP, pero [7] [9] declaran a S.M. Johnson como pionero en el estudio de problemas de *Scheduling* a través de su trabajo [10], donde describe un problema sobre Flow Shop resuelto por medio de un algoritmo propuesto y elaborado por él mismo.

Según [11], el Job Shop flexible se aborda por primera vez en [12]. Allí se propone un algoritmo polinomial para resolver un FJSSP con 2 trabajos. A partir de ahí se han desarrollado otros estudios sobre el tema. En [13] se concluye que la complejidad del Job Shop flexible es mayor a la del Job Shop debido a la necesidad de asignar operaciones a las diferentes máquinas y, de igual forma, mayor que la complejidad del Flow Shop flexible tratado en [14] donde todos los trabajos tienen el mismo orden de procesamiento.

Posteriormente, FJSSP fue abordado con ciertas modificaciones que lo aproximan más a la realidad empresarial:

- En [15] se estudia un algoritmo híbrido usando optimización por enjambre de partículas (PSO) y recocido simulado (SA) para optimizar la programación de un Job Shop flexible con objetivos múltiples.
- En [16], se utiliza la planificación jerárquica de la producción con el fin de encontrar buenas soluciones al problema de planificación y control de la producción en un taller de tipo FJSSP.
- En [17] se estudia el problema de restricciones no fijas para la programación de un Job Shop flexible (FJSSP-*nfa*), el cual se considera una variación más real del Job Shop flexible clásico, puesto que asume cada máquina sujeta a un número arbitrario de tareas asociadas a mantenimiento preventivo (PM). Por esta razón, proponen un algoritmo genético híbrido (hGA) en el que los operadores de cruce y de mutación son implementados en el espacio del fenotipo con el fin de incrementar su capacidad de heredar.
- En [18] se propone un algoritmo híbrido entre el PSO y la búsqueda tabú (TS), para solucionar un problema de objetivos múltiples del Job Shop flexible. El PSO diseñado muestra una alta eficiencia al combinar búsquedas globales y locales.
- Un algoritmo genético mejorado (IGA) para el problema de programación del Job Shop flexible Distribuido (DFJSP) se presenta en [19], donde los trabajos son procesados por un sistema de varias unidades de manufactura flexible (FMUs).
- El problema de Job Shop flexible con restricciones de transporte y tiempos acotados de procesamiento (*Bounded processing times*) con dos objetivos, es resuelto por medio de un algoritmo genético con búsqueda tabú (GATS) por [20]. Allí buscan minimizar el *makespan* y el almacenamiento (manejado en términos de tiempo de espera de cada trabajo).
- La optimización de la programación del Job Shop flexible de objetivos múltiples la examina [21] a través de una metaheurística híbrida entre PSO (basada en prioridades) y búsqueda local.
- El Job Shop flexible con tiempos de alistamiento (SDST-FJSP) se estudia en [22], donde presentan un algoritmo memético, con una estructura de vecindario (*fast neighborhood structure*) compuesta por dos movimientos (*reversal crititcal arcs* y *machines assignment*) para hacer estimaciones más rápidas.
- Un algoritmo micro artificial de colonia de abejas (MMABC) para solucionar un Job Shop flexible multiobjetivo con restricciones de transporte lo presentan [23], quienes proponen un operador de cruce (*Crossover operator*) para mejorar los resultados y así optimizar el *makespan* y el tiempo de procesamiento de cada operación, donde este último varía en cada una de las máquinas.
- En [24] se propone una solución basada en un planteamiento por niveles (*assigning and sequencing sub-problems*) para solucionar un problema de programación de Job Shop flexible con tiempos de transporte y tiempos de configuración dependientes de la secuencia (*sequence-dependent setup times*).

III. Planteamiento del problema

En concordancia con lo dicho por [25] y lo revisado en la literatura para este trabajo, el *makespan* es el objetivo a optimizar más utilizado, pues equivale a minimizar tiempos muertos o maximizar la utilización de las máquinas. Por ello, este es el objetivo a tener en cuenta para el modelo del presente artículo. Al realizar modificaciones al modelo matemático presentado por [20], el problema se puede

modelar de la siguiente manera:

A. Supuestos del modelo

Dentro de las principales consideraciones a tener en cuenta para el presente modelo se encuentran:

- Los tiempos de operación son conocidos con certidumbre.
- Todos los *Jobs* están disponibles para ser procesados en el instante t=0.
- No existen relaciones de precedencia entre trabajos.
- Todas las máquinas se encuentran listas para operar en t=0.
- No se permite la interrupción de los trabajos en las máquinas.
- Todos los *Jobs* tienen la misma prioridad dentro del sistema.
- No se consideran tiempos de alistamiento.
- Se permite la recirculación dentro de la programación, es decir, un trabajo puede visitar 2 veces una misma máquina.
- Existe una restricción de precedencia entre las operaciones de un *Job*.
- Existe una restricción de interferencia para una misma máquina en cualquier instante de tiempo.
- Se considera un *Job* terminado cuando finalicen todas sus operaciones.
- El tiempo de transporte será el mismo para desplazamientos vacíos o con carga.

B. Notación

Los conjuntos a manejar a lo largo del modelo matemático son:

$i,i' \in [1,n]$, trabajos
$j,j' \in [1,O_i]$, operaciones
$k,l \in [1,m]$,, máquinas
$h \in [1,r]$, recursos de transporte

Los parámetros a tener en cuenta en el modelo matemático son:

J_i, Trabajo i
O_i, Número de operaciones para el trabajo J_i
OP_{ij}, j-esima operación del trabajo J_i
P_{ijk}, tiempo de procesamiento para cada OP_{ij} en la máquina k
C_i, Tiempo de terminación del trabajo J_i
M_k, Máquina k
MP_{ij}, Conjunto de máquinas que pueden procesar la operación OP_{ij}
R_h, Recurso de transporte h
TP_{ij}, Actividad de transporte entre OP_{ij} y OP_{ij+1}
TR_{ij}, Conjunto total de recursos de transporte que pueden realizar TP_{ij}
σ_{klh}, Tiempo de transporte vacío entre la máquina M_k y M_l utilizando el recurso h
τ_{klh}, Tiempo de transporte con carga entre la máquina M_k y M_l utilizando el recurso h
Q, Número muy grande

Las siguientes son las variables necesarias para el modelo:

t_{ij}, Fecha de inicio de la OP_{ij} donde $i \in [1,n]$ y $j \in [1,O_i]$
v_{ij}, Fecha de inicio para TP_{ij} donde $i \in [1,n]$ y $j \in [1,O_i - 1]$
PJ_{ijk}, es 1 si la OP_{ij} es realizada por la máquina M_k donde $M_k \in MP_{ij}$
$YP_{ij\,i'j'k}$, es 1 si la OP_{ij} es realizada antes de la $OP_{i'j'}$ en la máquina k (dado que $PJ_{ijk} = 1$ y $PJ_{i'j'k} = 1$)
TJ_{ijh}, es 1 si el TP_{ij} es realizado por el recurso R_h
$YT_{ij\,i'j'h}$, es 1 si el TP_{ij} es realizado antes de $TP_{i'j'}$ (dado que $TJ_{ijh} = 1$ y $TJ_{i'j'h} = 1$)

C. Modelo matemático

Función Objetivo:

$$\min C_{max} = max_{i \in [1,n]} \{C_i\} \tag{1}$$

Donde

$$C_i = t_{iO_i} + \sum_{k \in MP_{iO_i}} P_{iO_i k} \tag{2}$$

Sujeto a:

$$t_{ij} + \sum_{k \in MP_{ij}} P_{ijk} * PJ_{ijk} \leq t_{ij'} \quad \forall\, i \in [1,n], j, j' \in [1,O_i] \tag{3}$$

$$\sum_{k \in MP_{ij}} PJ_{ijk} = 1 \quad \forall\, i \in [1,n], j \in [1,O_i] \tag{4}$$

$$\sum_{h \in [1,r]} TJ_{ijh} = 1 \quad \forall\, i \in [1,n], j \in [1,O_i - 1] \tag{5}$$

$$t_{ij} + P_{ijk} \leq t_{i'j'} + \left(1 - YP_{iji'j'k}\right) * Q \quad \forall\, i,i' \in [1,n], j,j' \in [1,O_i], k \in MP_{ij} \cap MP_{i'j'} \tag{6}$$

$$PJ_{ijk} * PJ_{i'j'k} * \left(t_{i'j'} + P_{i'j'k}\right) \leq t_{ij} + YP_{iji'j'k} * Q \quad \forall\, i,i' \in [1,n], j,j' \in [1,O_i], k \in MP_{ij} \cap MP_{i'j'} \tag{7}$$

$$v_{ij} + \sigma_{klh} + \tau_{klh} \leq v_{i'j'} + \left(1 - YT_{iji'j'h}\right) * Q \quad \forall\, i,i' \in [1,n], j \in [1,O_i - 1], j' \in [1,O_{i'} - 1], k,l \in [1,m], h \in TR_{ij} \cap TR_{i'j'} \tag{8}$$

$$TJ_{ijh} * TJ_{i'j'h} * \left(v_{i'j'} + \tau_{klh} + \sigma_{lkh}\right) \leq v_{ij} + YT_{iji'j'h} * Q \quad \forall\, i,i' \in [1,n], j \in [1,O_i - 1], j' \in [1,O_{i'} - 1], k,l \in [1,m], h \in TR_{ij} \cap TR_{i'j'} \tag{9}$$

$$t_{ij} + P_{ijk} + \tau_{kk'h} \leq t_{ij+1} \quad \forall\, i \in [1,n], j \in [1,O_i - 1], k \in MP_{ij}, k' \in MP_{ij+1}, h \in [1,r] \tag{10}$$

$$t_{ij} + \sum_{k \in MP_{ij}} P_{ijk} * PJ_{ijk} \leq v_{ij} \quad \forall\, i \in [1,n], j \in [1,O_i] \tag{11}$$

Donde la Ecuación (1) representa la función objetivo del modelo de optimización: minimizar el *makespan*; mientras que la Ecuación (2) representa la fecha de terminación de cada trabajo. La restricciones del modelo están representadas por las Ecuaciones de la (3) a la (10), donde (3) el tiempo de inicio de una operación subsiguiente a otra deber ser mayor o igual al tiempo de inicio de la operación anterior más su tiempo de duración. La Ecuación (4) obliga a cada trabajo a ser realizado solo por una máquina, al igual que cada transporte debe ser realizado por un único recurso, como se muestra en la Ecuación (5). Las Ecuaciones (6) y (7) tratan la capacidad de las maquinas con el fin de que no se procese más de una operación al tiempo en una misma máquina, donde (6) exige que el tiempo de inicio de una actividad en una determinada máquina debe ser mayor que el tiempo de inicio de la actividad predecesora en dicha máquina, más su tiempo de procesamiento; mientras que (7) determina la continuidad de dos operaciones de dos trabajos que se realizan en la misma máquina.

Las Ecuaciones (8) y (9) relacionan la capacidad de los recursos de transporte con el fin de que no se realice más de un transporte por un mismo recurso en el mismo periodo de tiempo, donde en (8) el tiempo de inicio del transporte utilizando un determinado recurso para un trabajo entre dos operaciones realizadas en máquinas diferentes, debe ser mayor que el tiempo de inicio del transporte predecesor utilizando el mismo recurso más su tiempo de transporte; mientras que (9) permite conocer la continuidad del transporte de dos trabajos para dos operaciones que se realizan en máquinas diferentes usando un mismo recurso. Por su parte, en la Ecuación (10) se establece que un recurso de transporte debe tener tiempo suficiente para mover un trabajo entre dos operaciones sucesivas. Finalmente, la Ecuación (11) establece que no se puede iniciar el transporte entre dos operaciones sucesivas de un trabajo sin que haya terminado la primera operación.

IV. Metodología

Para resolver el problema de *Job Shop* Flexible con restricciones de transporte propuesto en esta investigación, se descompuso la metodología en tres sub-etapas: 1) tipo de codificación, 2) asignación de operaciones y, 3) asignación de transporte. Posteriormente, se usa el algoritmo genético como técnica de optimización.

A. Codificación

El tipo de codificación representa así mismo lo que en términos del algoritmo genético es el cromosoma. Con base en los diferentes tipos de representaciones genéticas para el FJSSP, en esta investigación se usa la representación basada en operaciones codificando un *Schedule* por medio de una secuencia de operaciones a partir de números enteros, los cuales se repiten tantas veces como cantidad de operaciones haya en los trabajos.

Como se observa en la Fig. 1, a medida que se avanza en el cromosoma de izquierda a derecha, se determinan las operaciones de cada *Job*. De esta manera siempre se respeta la restricción de precedencia entre operaciones asegurando la factibilidad del cromosoma.

A continuación se realiza la asignación de las máquinas a cada una de las operaciones, lo cual se hace de manera aleatoria, dejando la optimización al algoritmo evolutivo. Esta información se almacena en otro cromosoma; ambos cromosomas representan una solución factible al problema. Cabe aclarar que la información de asignación está ligada al cromosoma de secuenciación, por ende, al ejecutarse cualquier procedimiento de optimización, la información de asignación será transmitida en el algoritmo genético a cromosomas hijos.

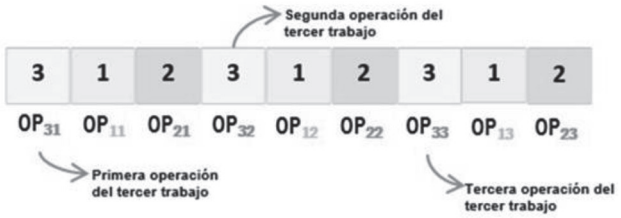

Fig. 1. Codificación del cromosoma.

Finalmente, el cromosoma se modifica adicionando el doble de operaciones a las que se tenían inicialmente. Estas representan los transportes entre las máquinas y su traslado inicial de bodega. El cromosoma quedará como se muestra en la Fig. 2.

La asignación de operaciones de transporte se realiza por medio de evaluaciones parciales de la carga de cada recurso de transporte, de manera que exista un equilibrio entre los mismos.

B. Algoritmo propuesto

Los Algoritmos Genéticos (AGs) trabajan con una población de individuos donde cada uno representa una solución factible a un problema dado. Cada individuo (cromosoma) está compuesto por genes (información) cuyo valor está relacionado con la bondad de la solución y representa lo que en la naturaleza corresponde al grado de efectividad de un organismo para competir por unos determinados recursos. Los AGs funcionan de manera iterativa y cada iteración es llamada una generación. A medida que las iteraciones aumentan, se supone que los cromosomas tendrán mejores genes de sus padres (generación anterior), por lo tanto, la calidad promedio de las soluciones será siempre mejor que la anterior generación. Para ello hace uso de operadores genéticos:

- Generación de la población inicial: en la presente investigación la población inicial se generó de forma aleatoria.
- Selección: en esta etapa se realizó una *selección por torneo* con reemplazo donde la probabilidad de escoger una solución para el torneo era proporcional a la función *fitness*.
- Cruce: de acuerdo con [26], el cruce basado en dos puntos es el que mejores resultados arroja en comparación con 3 o más puntos, por ende, es el aplicado en la presente

investigación (ver Fig. 3), donde adicionalmente se tiene en cuenta la probabilidad de que exista el cruce.

- Mutación: dado que su objetivo es diversificación de la población cambiando de forma arbitraria algunos de los genes, se escogió el operador *swap*, en el cual se seleccionan de manera aleatoria dos genes para ser intercambiados uno por el otro. Sin embargo, la probabilidad de mutación controla el porcentaje en el que se produce esta diversificación en la población.
- Reparación: gracias al tipo de codificación del cromosoma propuesto para esta investigación, no es necesario ejecutar el operador de Reparación.

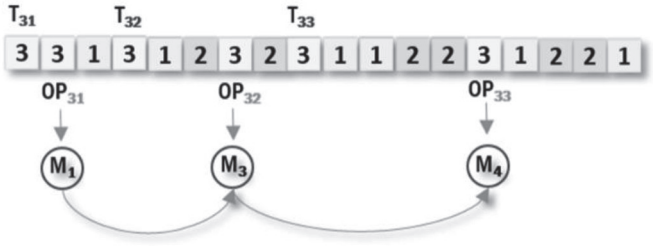

Fig. 2. Asignación de transporte de operaciones entre máquinas.

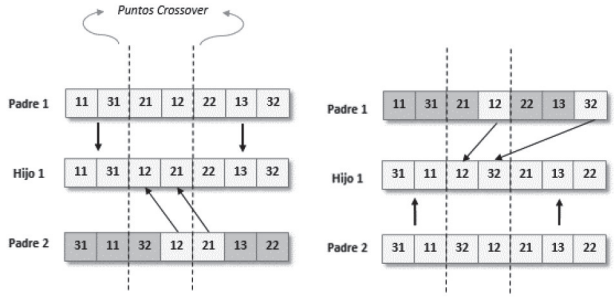

Fig. 3. Cruce de dos puntos.

Todo el ciclo de evolución (todos los operadores) se repiten hasta que se cumpla algún criterio de parada, como el número de iteraciones o la diferencia de avance en la solución menor a una tolerancia establecida previamente. En este caso, el criterio de parada elegido fue el número de generaciones.

Por otro lado, algunos parámetros de entrada necesarios para que se ejecute el AG diseñado son: tamaño de la población, número de iteraciones, probabilidad de cruce y probabilidad de mutación. En la Fig. 4 se puede ver el pseudocódigo del algoritmo programado.

V. Resultados experimentales

Con el fin de validar el desempeño de la codificación propuesta en el lenguaje de programación Matlab®, se realizó la comparación de los resultados versus las instancias en diferentes etapas que representan la descomposición por partes del *Job Shop* Flexible con restricciones de transporte. Estas etapas son:

- Problemas de tipo *Job Shop*.
- Problemas tipo *Job Shop* Flexible.

```
INICIAR /*Algoritmo Genetico*/
  Generar poblacion inicial aleatoria
  Calcular la función fitness de cada individuo
  Insertar Población; Tot_Generaciones; Prob_Cruce; Prob_Mutacion
  HACER
    Generaciones=1
  MIENTRAS Terminar=FALSE HACER
  INICIAR /*Nueva generación*/
    POR Tamaño_población/2 HACER
    INICIAR /*Ciclo*/
      Calcular probabilidad de selección a cada individuo según fitness
      Seleccionar dos individuos aleatorios para el torneo
      El mejor individuo es Padre1
      Seleccionar dos individuos aleatorios para el torneo
      El mejor individuo es Padre2
      Generar Aleatorio para operador de cruce
      SI Aleatorio≤Prob_Cruce ENTONCES
        Cruzar Padre1 y Padre2
      SI NO
        Hijo1=Padre1
        Hijo2=Padre2
      Generar Aleatorio para operador de mutacion Hijo1
      SI Aleatorio≤Prob_Mutacion ENTONCES
        Mutar Hijo1
      Generar Aleatorio para operador de mutacion Hijo2
      SI Aleatorio≤Prob_Mutacion ENTONCES
        Mutar Hijo2
      Calcular fitness de los dos descendientes
      Insertar  los dos descendientes en la nueva generación
      Generaciones=Generaciones+1
    FIN
    SI Generaciones=Tot_Generaciones ENTONCES
      Terminar=TRUE
  FIN
FIN
```

Fig. 4. Pseudocódigo del Algoritmo Genético.

- Problemas tipo *Job Shop* con restricciones de transporte.
- Problemas tipo *Job Shop* Flexible con restricciones de transporte.

El algoritmo fue ejecutado en una computadora con procesador Intel core i7, 2 GHz y 8 Gb de memoria RAM para todas las instancias y para la ejecución y desarrollo del diseño de experimentos.

A. Resultados de la validación para el problema Job Shop

Por medio de las instancias desarrolladas por Fisher & Thomson [27] y Lawrence [28], se confrontan los resultados obtenidos y se verifica la efectividad de la codificación propuesta para los problemas de tipo *Job Shop*. Se toman 10 de estas instancias. Lo resultados se muestran en la Tabla I, y de acuerdo con ellos se puede concluir que el Algoritmo genético diseñado con restricción de transporte encuentra muy buenas soluciones al problema del *Job Shop* clásico.

B. Resultados de la validación para el problema Job Shop Flexible

Para este apartado se tomaron las instancias presentadas por Fattahi [29], Paulli [30] y Hurink [31], se realizó la validación de la codificación para problemas de tipo Job Shop flexible, en donde las operaciones pueden ser realizadas por más de una máquina. De igual forma, se tomó una muestra aleatoria de las instancias a resolver para validar la efectividad del algoritmo en estos problemas. Los resultados se muestran en la Tabla II.

Tabla I
Resultados instancias de job shop

Instancia	Jobs	Máq.	C_{max} obtenido	Mejor C_{MAX}	% Error
Ft06	6	6	55	55	0,00%
Ft10	10	10	978	937	4,38%
Ft20	20	5	1217	1165	4,46%
La01	10	5	666	666	0,00%
La06	15	5	926	926	0,00%
La09	15	5	951	951	0,00%
La14	20	5	1292	1292	0,00%
La17	10	10	801	784	2,17%
La21	15	10	1079	1046	3,15%
La38	15	15	1254	1196	4,85%

Tabla II
Resultados instancias de job shop flexible

Instancia	Jobs	Máq.	C_{max} obtenido	Mejor C_{MAX}	% Error
Fattahi01	2	2	107	107	0,00%
Fattahi04	3	2	355	355	0,00%
Fattahi06	3	3	320	320	0,00%
Fattahi10	4	5	516	516	0,00%
Fattahi20	12	8	1237	1208	2,40%
Paulli 1	10	5	2616	2568	1,87%
Paulli 5	10	5	2279	2243	1,60%
Hurink01	6	6	47	47	0,00%
Hurink06	10	5	493	477	3,35%
Hurink10	15	5	766	750	2,13%
Hurink14	20	5	1096	1072	2,24%
Hurink23	10	10	810	757	7,00%

Según estos resultados, se puede concluir que el código es efectivo para problemas de tipo Job Shop flexible, observando que para instancias de complejidad media y baja el margen de diferencia con el mejor resultado encontrado no supera 3,5%, donde en 5 de estos casos alcanza dicho valor. Adicionalmente, se muestra que, para instancias de gran complejidad, solo se presenta un margen de error máximo del 7% en la instancia de "Hurink23" de tamaño 10x10.

C. Resultados de la validación para el problema Job Shop con restricciones de transporte

Posteriormente, a través de instancias propuestas por Deroussi [32] que describen una combinación de *jobsets* y distribuciones de planta o *layouts*, se realizó la validación del algoritmo para instancias de tipo *Job Shop* con restricciones de transporte.

De acuerdo con los resultados encontrados, para tres instancias se coincidió con la mejor solución encontrada hasta el momento. Además, en otros tres casos, se obtiene una mejor solución que la hallada por TSAG [33] (Ver Tabla III).

Adicionalmente, se puede observar que se presenta un margen de error menor al 11,6%, donde los más atípicos se presentan en las configuraciones de *layout* número 3. Esta configuración, a diferencia de las demás, muestra un rango de valores en tiempos de transporte bastante amplio, lo cual genera dependencia del sentido de desplazamiento y aumenta la complejidad del problema. Para los demás casos se observa un error promedio del 5,62%.

Tabla III
Resultados instancias de job shop con restricciones de transporte

Instancia	C_{max} obtenido	Algoritmo SALS	Algoritmo TSAG	Mejor C_{MAX}	% Error
EX11	96	96	96	96	**0,00%**
EX12	86	82	82	82	4,88%
EX13	89	84	84	84	5,95%
EX21	106	100	104	100	6,00%
EX22	84	76	76	76	10,53%
EX23	95	86	86	86	10,47%
EX31	108	99	99	99	9,09%
EX32	92	85	85	85	8,24%
EX33	96	86	86	86	11,63%
EX41	117	112	116	112	4,46%
EX42	93	87	91	87	6,90%
EX43	98	89	94	89	10,11%
EX51	87	87	88	87	0,00%
EX52	74	69	69	69	7,25%
EX53	79	74	74	74	6,76%
EX61	124	118	120	118	5,08%
EX62	109	98	98	98	11,22%
EX63	112	103	103	103	8,74%
EX71	111	111	115	111	0,00%
EX72	82	79	85	79	3,80%
EX73	90	83	93	83	8,43%
EX81	163	161	161	161	1,24%
EX82	159	151	151	151	5,30%
EX83	166	153	153	153	8,50%
EX91	123	116	116	115	6,96%
EX92	111	102	102	102	8,82%
EX93	114	105	105	105	8,57%
EX101	153	147	148	147	4,08%
EX102	147	135	135	135	8,89%
EX103	153	138	141	138	10,87%

D. Resultados de la validación para el problema Job Shop Flexible con restricciones de transporte

Finalmente, se realizó la validación del algoritmo por medio de problemas de tipo *Job Shop* flexible con restricciones de transporte. Para esto, se modificaron las instancias de Deroussi [32] con el fin de obtener las de tipo *Job Shop* flexible con flexibilidad parcial. Así, al procesar dichas instancias se puede ver una disminución en el valor del *makespan*.

Las instancias fueron nombradas de la siguiente manera: EXF11, que hace referencia a la instancia original del autor, indicando que fueron modificadas para poder ser evaluadas en problemas de tipo flexible, donde la parte numérica

relaciona el *Jobset* con la configuración del *layout* para los tiempos de transporte. Estas instancias modificadas pueden encontrarse en el Github *https://github.com/INDUSTRIALOPALO/FJSSP-t*, disponible para su consulta y prueba por demás algoritmos.

Tabla IV
Resultados instancias de job shop flexible con restricciones de transporte

Instancia	C_{MAX} obtenido	Mejor C_{MAX} sin flexibilizar	% Error	C_{MAX} obtenido sin flexibilizar	% Error
EXF12	76	82	7,89%	86	13,16%
EXF22	70	76	8,57%	84	20,00%
EXF32	80	85	6,25%	92	15,00%
EXF33	81	86	6,17%	98	20,99%
EXF43	85	89	4,71%	98	15,29%
EXF62	79	98	24,05%	109	37,97%
EXF82	116	151	30,17%	159	36,07%
EXF83	114	152	33,30%	166	45,61%
EXF81	118	161	36,44%	163	38,13%
EXF91	101	116	14,85%	123	21,78%

De los resultados mostrados en la Tabla IV se puede concluir que, al flexibilizar las instancias, el algoritmo es capaz de realizar mejores asignaciones para obtener un menor valor de *makespan*. Si bien el porcentaje de mejora dependerá en gran parte del tipo de flexibilidad que se tenga en el problema (parcial o total), se observa que, para la flexibilización del problema de forma aleatoria, se obtuvo una mejor solución con una mejora del 4,71% como mínimo.

VI. Diseño de experimentos

Para poder evaluar el algoritmo y su comportamiento respecto a los 4 factores que indicen en su eficiencia para solución de problemas tipo *Job shop* flexible, se realizó un diseño de experimentos factorial 2^{k-1}, teniendo en cuenta como variable respuesta el *makespan*. Se tomaron 4 factores y dos niveles para cada uno de ellos, con 5 réplicas por cada uno de los tratamientos. Además, de acuerdo con [34], cada escenario debe ajustarse y realizar las ejecuciones del algoritmo de manera aleatoria después de haber definido el diseño factorial.

A. Instancias a testear

Teniendo en cuenta que el algoritmo propuesto fue desarrollado para solucionar problemas de tipo *Job shop* flexible con restricciones de transporte, los diseños de experimentos a ejecutar se basaron en las instancias propuestas por [30] y se seleccionaron 7 de las 32 posibles. Las instancias usadas para realizar el experimento fueron: EX11, EX31, EX32, EX43, EX62, EX82 y EX93.

B. Factores y niveles

Los factores analizados son los que presentan mayor influencia sobre la variable respuesta. En la Tabla V se aprecian dichos factores y los niveles para cada uno, mientras que en la Tabla VI se presentan los niveles que se probaron en cada uno de los 8 tratamientos del diseño experimental fraccionado.

C. Resultados

Para todos los diseños factoriales realizados se validaron los distintos supuestos de normalidad, homocedasticidad e independencia por medio del software estadístico Minitab 17, así como las diferentes pruebas estadísticas necesarias. En las Fig. 5 a Fig. 11 se aprecian los diagramas de efectos estandarizados para cada una de las instancias probadas. Además, en las Tabla VII a Tabla XIII se presentan los respectivos análisis de varianza.

Tabla V
Factores y niveles para el experimento

Factores	Niveles	
	Bajo	Alto
Número de generaciones	100	900
Tamaño de población	10	100
Probabilidad de cruce	0,5	0,9
Probabilidad de mutación	0,01	0,1

Tabla VI
Definición de los niveles para el análisis de varianza

Prob. Mutación		Prob. Cruce		Tam. Población		Num. Generac.	
-	+	-	+	-	+	-	+
0,01	**0,1**	**0,5**	**0,9**	**10**	**100**	**100**	**900**
x		x		x		x	
	x	x		x			x
x			x	x			x
	x		x	x		x	
x		x			x		x
	x	x			x	x	
x			x		x	x	
	x		x		x		x

Tabla VII
Anova: problema ex11

Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Prob. Mutación	1	9,025	9,025	0,84	0,365
Prob. Cruce	1	21,025	21,025	1,96	0,17
Tam. Pob.	1	164,025	164,025	15,31	0,0001
Generaciones	1	3,025	3,025	0,28	0,598
Error	35	374,875	10,711	14,14	
Total	**39**	**571,975**			

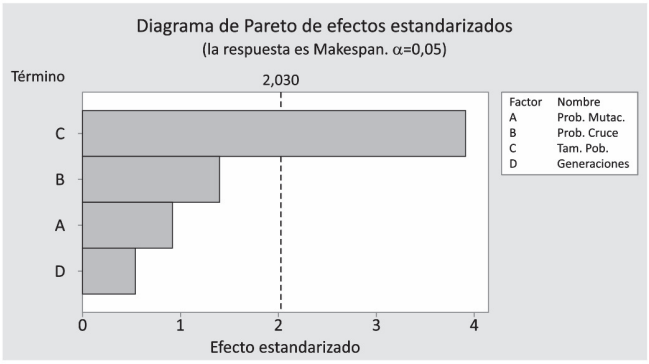

Fig. 5. Diagrama de Pareto de Efectos Estandarizados: problema EX11

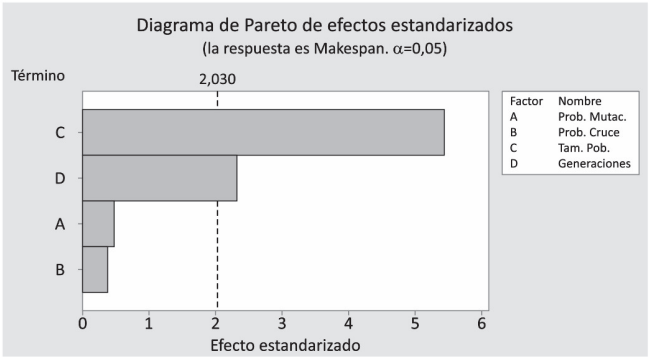

Fig. 6. Diagrama de Pareto de Efectos Estandarizados: problema EX31

Tabla VIII
Anova: problema ex31

Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Prob. Mutación	1	2,025	2,025	0,24	0,630
Prob. Cruce	1	1,225	1,225	0,14	0,708
Tam. Pob.	1	255,025	255,025	29,71	0,0001
Generaciones	1	46,225	46,225	5,38	0,026
Error	35	300,475	8,585		
Total	39	604,975			

Fig. 9. Diagrama de Pareto de Efectos Estandarizados: problema EX62

Tabla IX
Anova: problema ex32

Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Prob. Mutación	1	25,6	25,6	5,23	0,028
Prob. Cruce	1	8,1	8,1	1,65	0,207
Tam. Pob.	1	313,6	313,6	64,07	0,0001
Generaciones	1	62,5	62,5	12,77	0,001
Error	35	171,3	4,894		
Total	39	581,1			

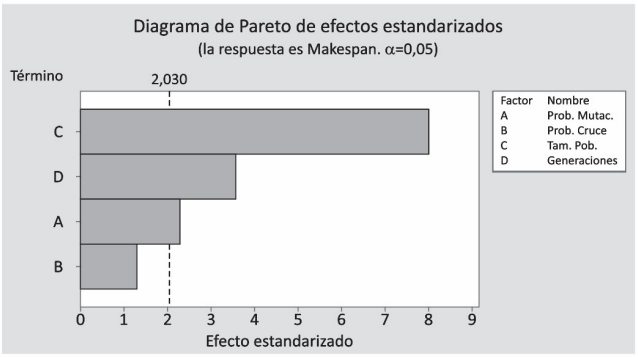

Fig. 7. Diagrama de Pareto de Efectos Estandarizados: problema EX32

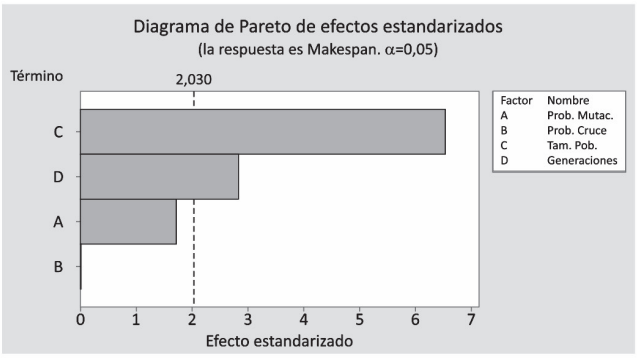

Fig. 8. Diagrama de Pareto de Efectos Estandarizados: problema EX43

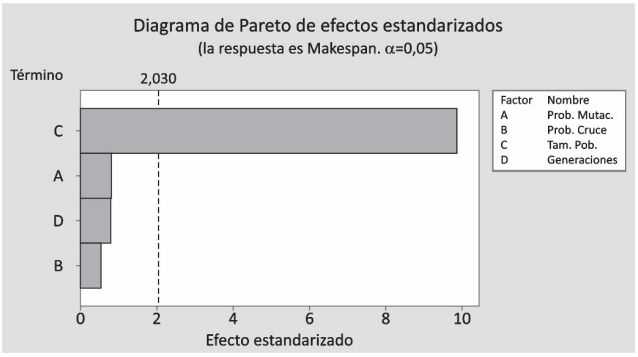

Fig. 9. Diagrama de Pareto de Efectos Estandarizados: problema EXG2

Tabla X
Anova: problema ex43

Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Prob. Mutación	1	19,6	19,6	2,97	0,093
Prob. Cruce	1	0	0	0	1
Tam. Pob.	1	280,9	280,9	42,63	0,0001
Generaciones	1	52,9	52,9	8,03	0,008
Error	35	230,6	6,589		
Total	39	584			

Tabla XI

Anova: problema ex62

Fuente	GL	SC Ajust	MC Ajust	Val-or F	Valor p
Prob. Mutación	1	0,9	0,9	0,61	0,441
Prob. Cruce	1	0,4	0,4	0,27	0,606
Tam. Pob.	1	144,4	144,4	97,57	0,0001
Generaciones	1	0,9	0,9	0,61	0,441
Error	35	51,8	1,48		
Total	39	198,4			

Tabla XII

Anova: problema ex82

Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Prob. Mutación	1	2,025	2,025	0,19	0,663
Prob. Cruce	1	2,025	2,025	0,19	0,663
Tam. Pob.	1	60,025	60,025	5,73	0,022
Generaciones	1	60,025	60,025	5,73	0,022
Error	35	366,875	10,482	-	-
Total	39	490,975	-	-	-

Tabla XIII

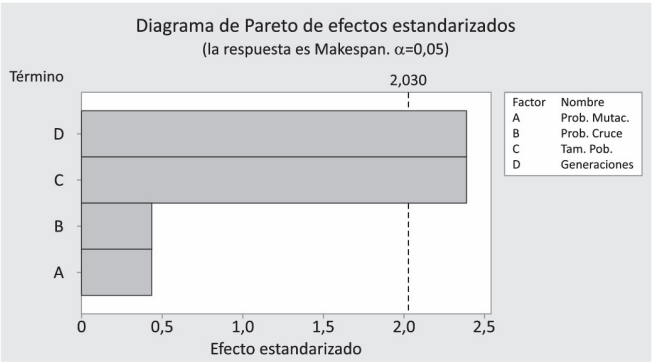

Fig. 10. Diagrama de Pareto de Efectos Estandarizados: problema EX82

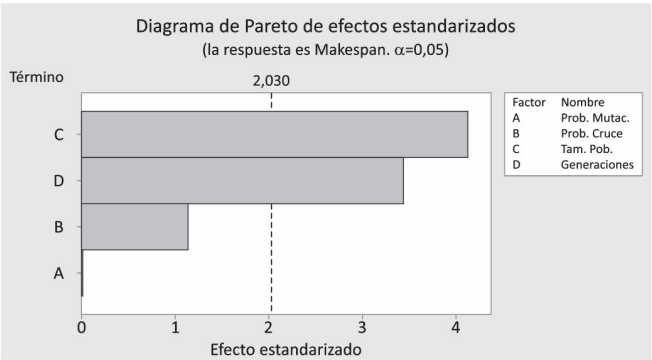

Fig. 11. Diagrama de Pareto de Efectos Estandarizados: problema EX93

Anova: problema ex93

Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Prob. Mutación	1	0	0	0	1
Prob. Cruce	1	2,5	2,5	1,32	0,259
Tam. Pob.	1	32,4	32,4	17,05	0,0001
Generaciones	1	22,5	22,5	11,84	0,002
Error	35	66,5	1,9		
Total	39	123,9			

De lo obtenido a través de los distintos experimentos factoriales elaborados, se concluye que en el 100% de los casos el tamaño de la población es significativo para poder obtener mejores resultados de la variable respuesta. Adicionalmente, en el 72% de los casos el número de generaciones también fue un factor determinante. Estos dos hechos son consistentes con la utilización de un algoritmo evolutivo.

VII. Conclusiones

A partir de los resultados obtenidos y la validación hecha de manera progresiva en las instancias encontradas de programación de operaciones (*job shop*, *job shop* flexible, entre otros), se puede concluir que el algoritmo genético programado es una herramienta efectiva a la hora de abarcar problemas de programación de operaciones con limitaciones de transporte. Cabe mencionar que estos, los AGs, no son la única forma de abordarlos, tal y como se expone en [35].

De acuerdo con lo observado en el diseño de experimentos, más del 70% de los casos analizados demuestran la significancia del número de generaciones, lo cual podría tomarse como un resultado esperado al tener en cuenta que se está trabajando con un algoritmo evolutivo.

Igualmente, se determinó que en el 100% de los casos el tamaño de población fue un factor significativo en función de la minimización del *makespan*. Esto hace ver que aun cuando los factores de probabilidad de cruce y mutación no fueron representativos en la mayoría de casos, es necesario contar con una cantidad amplia de cromosomas (soluciones) para que operen estos elementos del algoritmo y obtener mejores resultados a medida que se avance en el número de iteraciones. El hecho de contar con un tamaño de población elevado, puede verse reflejado en una mayor exploración del espacio de solución.

Finalmente, se corroboró la importancia en la selección de buenos valores para los parámetros de entrada del algoritmo genérico, especialmente en el tamaño de la población y el número de generaciones. Por tanto, se insta a analizarlos a mayor profundidad con el fin de establecer valores coherentes que permitan encontrar muy buenas soluciones sin castigar la utilización de los recursos (tiempo y espacio computacional).

Referencias

[1] Zhang, G., Shao, X., Li, P. and Gao, L., "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem", *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1309-1318, 2009. doi: 10.1016/j.cie.2008.07.021

[2] Tang, J., Zhang, G., Lin, B. and Zhang, B., "A hybrid algorithm for flexible job-shop scheduling problem", *Procedia Engineering*, vol. 15, pp. 3678-3683, 2011. doi: 10.1016/j.proeng.2011.08.689

[3] Fernández, M. A., "Soluciones metaheurísticas al 'Job-shop scheduling problem with sequence-dependent setup times'", Ph.D. dissertation, directed by M. del C. Rodríguez Vela, Universidad de Oviedo, España, Jul. 2011

[4] Rossi, A., "Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships", *International Journal of Production Economics*, vol. 153, pp. 253-267, 2014. doi: 0.1016/j.ijpe.2014.03.006

[5] Garey, M. R. y D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, San Francisco, LA: Freeman, 1979.

[6] John Henry Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, Cambridge, MA: MIT press, 1992.

[7] Jain, A. S. and S. Meeran, "Deterministic job-shop scheduling: Past, present and future", *European journal of operational research*, vol. 113, no. 2, pp. 390-434, 1999. doi: 10.1016/S0377-2217(98)00113-1

[8] Graham, R. L., Lawler, E. L., Lenstra, J. K. and Kan, A. R., "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of discrete mathematics*, vol. 5, pp. 287-326, 1979. doi: 10.1016/S0167-5060(08)70356-X

[9] Conway, R. W., Maxwell, W. L. and Miller, L. W., *Theory of scheduling*. Courier Corporation, 2012.

[10] Johnson, S. M., "Optimal two and three stage production schedules with setup times included", *Naval research logistics quarterly*, vol. 1, no. 1, pp. 61-68, 1954. doi: 10.1002/nav.3800010110

[11] Najid, N. M., Dauzere-Pérès, S. y Zaidat, A., "A modified simulated annealing method for flexible job shop scheduling problem", *IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, p. 6, 2002. doi: 10.1109/ICSMC.2002.1176334

[12] Brucker, P. and Schlie, R., "Job-shop scheduling with multi-purpose machines", *Computing*, vol. 45, no. 4, pp. 369-375, 1990.

[13] Jansen, k., Mastrolilli, M. and Solis-Oba, R., "Approximation algorithms for flexible job shop problems" In: *LATIN 2000: Theoretical Informatics.* Springer Berlin Heidelberg, pp. 68-77, 2000.

[14] Jiménez, A. P., Muñoz, C. A. and Toro, E. M.. "Solución del Problema de Flow Shop Flexible Aplicando el Algoritmo Genético de Chu-Beasley", *Entre Ciencia e Ingeniería*, vol. 7, no. 13, pp. 34-40, 2013.

[15] Xia, W. and Wu, Z., "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems", *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 409-425, 2005. doi: 10.1016/j.cie.2005.01.018

[16] Osorio, J. C., Motoa, T. G., "Planificación jerárquica de la producción en un job shop flexible", *Revista Facultad de Ingeniería Universidad de Antioquia*, no. 44, pp. 158-171, 2008.

[17] Gao, J., Gen, M. and Sun, L., "Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm", *Journal of Intelligent Manufacturing*, vol. 17, no. 4, pp. 493-507, 2006. doi: 10.1007/s10845-005-0021-x

[18] Zhang, G., Shao, X., Li, P. and Gao, L., "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem", *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1309-1318, 2009. doi: 10.1016/j.cie.2008.07.021

[19] De Giovanni, L. and Pezzella, F., "An improved genetic algorithm for the distributed and flexible job-shop scheduling problem", *European journal of operational research*, vol. 200, no. 2, pp. 395-408, 2010. doi: 10.1016/j.ejor.2009.01.008

[20] Zhang, Q., Manier, H. and Manier, M. A., "A genetic algorithm with Tabu Search procedure for flexible Job Shop Scheduling with transportation constraints and bounded processing times", *Computers & Operations Research*, vol. 39, no. 7, pp. 1713-1723, 2012. doi: 10.1016/j.cor.2011.10.007

[21] Moslehi, G. and Mahnam, M., "A Pareto approach to multi-objective flexible Job-Shop Scheduling problem using Particle Swarm Optimization and local search", *International Journal of Production Economics*, vol. 129, no. 1, pp. 14-22, 2011. doi: 10.1016/j.ijpe.2010.08.004

[22] González, M. A., Rodríguez Vela, C. and Varela, R., "An efficient memetic algorithm for the flexible job shop with setup times", In: *Twenty-Third International Conference on Automated Planning and Scheduling*, Febrero, 2013.

[23] Liu, Z., Ma, S., Shi, Y. and Teng, H., "Solving multi-objective Flexible Job Shop Scheduling with transportation constraints using a micro artificial bee colony algorithm", *In: Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, IEEE, pp. 427-432, Junio, 2013. doi: 10.1109/CSCWD.2013.6581001

[24] Rossi, A., "Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships" *International Journal of Production Economics*, vol. 153, pp. 253-267, 2014. doi: 10.1016/j.ijpe.2014.03.006

[25] Gallego, G., "*Linear control policies for scheduling a single facility after an initial disruption*", Informe técnico No. 770, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, enero, 1988.

[26] DE JONG, K. A., "*An analysis of the behaviour of a class of genetic adaptive systems*". Tesis doctoral, Universidad de Michigan, Ann Arbor, MI, USA, 1975.

[27] Fisher, H. and Thompson, G. L., "Probabilistic learning combinations of local job-shop scheduling rules", *Industrial scheduling*, vol. 3, pp. 225-251, 1963.

[28] Lawrence, S., "*Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement)*". Tesis doctoral, Carnegie-Mellon University, Graduate School of Industrial Administration, Pittsburgh, PA, 1984.

[29] Fattahi, P., Mehrabad, M. Saidi, J. y Fariborz., "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems", *Journal of Intelligent Manufacturing*, vol. 18, no 3, pp. 331-342, 2007. doi: 10.1007/s10845-007-0026-8

[30] Dauzère-Pérès, S. and Paulli, J., "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search", *Annals of Operations Research*, vol. 70, p. 281-306, 1997. doi: 10.1023/A:1018930406487

[31] Hurink, J. and Knust, S., "Makespan minimization for flow-shop problems with transportation times and a single robot", *Discrete Applied Mathematics*, vol. 112, no 1, pp. 199-216, 2001. doi: 10.1016/S0166-218X(00)00316-4

[32] Deroussi, L. and Norre, S., Simultaneous scheduling of machines and vehicles for the flexible job shop problem. En: *International conference on metaheuristics and nature inspired computing.* 2010.

[33] Zhang, Q., Manier, H. and Manier, M.-A., "A genetic algorithm with Tabu Search procedure for flexible job shop scheduling with transportation constraints and bounded processing times", *Computers & Operations Research*, vol. 39, no 7, p. 1713-1723, 2012. doi: 10.1016/j.cor.2011.10.007

[34] Box, G. E. P., Stuart Hunter, J. and Hunter, W. G., *Statistics for Experimenters: Design, Innovation, and Discovery*. Wiley-Interscience: New York, NY, USA, vol. 2, 2005.

[35] Castrillón, O. D., Giraldo J. A. y Sarache, W. A., "Solución de un problema Job Shop con un agente inteligente", *Ingeniería y Ciencia*, vol. 5, no. 10, pp. 75–92, 2009.

Tatiana Andrea Castillo Jaimes. Nació en Bucaramanga, Colombia. Se graduó en la Universidad Industrial de Santander como Ingeniera Industrial y Magister en Ingeniería Industrial. Ejerció profesionalmente en la Comercializadora Arte Laser de Bucaramanga y como docente en la Universidad Industrial de Santander y Universidad de Investigación y Desarrollo. Es investigadora afiliada al Grupo de Investigación en Optimización y Organización de Sistemas Productivos y Logísticos, OPALO. Entre sus campos de interés están la optimización de la cadena de suministros, los modelos de optimización y los algoritmos heurísticos y metaheurísticos.

Carlos Eduardo Díaz Bohórquez nació en Socorro, Colombia. Se graduó como Ingeniero Industrial y Especialista en Evaluación y Gerencia de Proyectos en la Universidad Industrial de Santander. Además, obtuvo el título de Magister en Ingeniera Industrial en la Universidad de los Andes. Ha ejercido como docente en la Universidad de los Andes, la

Universidad Pontificia Bolivariana Seccional Bucaramanga, la Universidad Cooperativa de Colombia y, actualmente en, la Universidad Industrial de Santander donde se desempeña como docente investigador del Grupo de Investigación en Optimización y Organización de Sistemas Productivos y Logísticos, OPALO. Entre sus campos de interés están la optimización de la programación de la producción, los modelos matemáticos y los algoritmos heurísticos y metaheurísticos.

Juan David Gómez Moreno nació en Bucaramanga, Colombia. Se graduó como Ingeniero Industrial en la Universidad Industrial de Santander. Se desempeñó como estudiante investigador en el Grupo de Investigación en Optimización y Organización de Sistemas Productivos y Logísticos, OPALO. Además, trabajó como Analista de Contratos en Oxy Colombia y posteriormente como Comprador Sr de Quala Colombia. Hoy día ejerce como Coordinador de Dotación Corporativa en Avianca, Colombia.

Edwin Alfredo Orduz González nació en Bucaramanga, Colombia. Se graduó como Ingeniero Industrial en la Universidad Industrial de Santander. Se desempeñó como estudiante investigador en el Grupo de Investigación en Optimización y Organización de Sistemas Productivos y Logísticos, OPALO. Laborando para IBM Colombia como Sales & Staff Professional y Sales Specialist.

Myriam Leonor Niño López nació en Bucaramanga, Colombia. Se graduó como Ingeniera Industrial y especialista en Gerencia de la Producción en Mejoramiento Continuo en la Universidad Industrial de Santander. También obtuvo el título de Magister en Administración en el Instituto Tecnológico y de Estudios Superiores de Monterrey, y el Doctorado en Dirección y Organización de Empresas en la Universidad Politécnica de Cataluña. Ejerció profesionalmente en la Universidad Industrial de Santander como Profesional Administrativo. Actualmente se desempeña como docente investigadora adscrita al Grupo de Investigación en Optimización y Organización de Sistemas Productivos y Logísticos, OPALO, en la Universidad Industrial de Santander. Entre sus campos de interés están la Ingeniería de Producción y la Gestión de las organizaciones.