UN ALGORITMO HÍBRIDO GENÉTICO PARA EL PROBLEMA JOB SHOP FLEXIBLE CON TIEMPOS DE ALISTAMIENTO DEPENDIENTES DE LA SECUENCIA (FJSP-SDST)

A SIMULATED ANNEALING GENETIC ALGORITHM FOR FLEXIBLE JOB SHOP PROBLEM WITH SEQUENCE-DEPENDENT SETUP TIMES (FJSP-SDST)

Carlos Eduardo Díaz Bohórquez[1]
Lina Mayerly Lozano Suarez[2]
Fabián Alexander Torres Cárdenas[3]

Resumen

La programación de producción es uno de los temas más críticos en los sistemas de fabricación y ha sido ampliamente estudiado en la literatura. Los problemas de programación del taller (JSSP) se encuentran entre los problemas combinatorios más estudiados, donde un conjunto de trabajos deben procesarse en un conjunto de máquinas específicas. Cada trabajo consta de un conjunto específico de operaciones, que deben procesarse de acuerdo con un orden determinado. El problema del taller de trabajo flexible (FJSP) es una generalización del JSSP clásico donde cada operación puede ser procesada por más de una máquina, el problema FJSP es considerado un problema de tipo Np-hard en el que se cubren dos dificultades: el problema de asignación de máquinas y la secuenciación de operaciones. En este documento se aborda el problema FJSP con tiempos de alistamiento dependientes de la secuencia (FJSP-SDST) en el que se busca minimizar el Makespan. Proponemos un algoritmo hibrido basado en un algoritmo genético (GA) y recocido simulado (SA) para resolver este problema. Para evaluar el rendimiento de nuestro algoritmo, comparamos nuestros resultados con otros métodos existentes en la literatura. Resolviendo los escenarios con resultados cercanos a los encontrados en la literatura en tiempos computacionales razonables.

Palabras clave: Programación de Operaciones, Problema Job Shop Flexible, Tiempos de Alistamiento Dependientes de la Secuencia, Metaheurísticas, Algoritmo Genético, Recocido Simulado.

Abstract

Production scheduling is one of the most critical issues in manufacturing systems and has been widely studied in the literature. The job shop scheduling problems (JSSP) are among the most studied combinatorial problems, where a set of jobs must be processed in a set of

Recibido: 15 de marzo de 2020 /Evaluación: 20 de abril de 2020 / Aprobado: 12 de junio de 2020

[1] Magister en Ingeniería Industrial. Docente de la Universidad Industrial de Santander. Email: cediazbo@uis.edu.co. ORCID: https://orcid.org/0000-0002-0365-833X
[2] Magister en Ingeniería Industrial. Docente de la Universidad Industrial de Santander. Email: llozano7@udi.edu.co. ORCID: https://orcid.org/0000-0002-5945-3480
[3] Magister en Ingeniería Industrial. Docente de la Universidad Industrial de Santander. Email: ftorres4@udi.edu.co. ORCID: https://orcid.org/0000-0003-1781-5380

specific machines. Each job consists of a specific set of operations, which must be processed according to a specific order. Flexible job shop problem (FJSP) it is a generalization of the classic JSSP where each operation can be processed by more than one machine. The FJSP problem is considered an Np-hard type problem in which two difficulties are covered: the allocation problem and sequencing of operations. This document addresses the FJSP problem with sequence-dependent setup times (FJSP-SDST) where Makespan is minimized. We propose a hybrid algorithm based on a Genetic Algorithm (GA) and Simulated Annealing (SA) to solve this problem. To evaluate the performance of our algorithm, we compare our results with other methods in the literature. Solving the scenarios with results close to those found in the literature in reasonable computational times.

Keywords: Operations Scheduling**,** Flexible Job Shop Problem, Sequence-dependent Setup Times, Metaheuristics, Genetic algorithm, Simulated Annealing.

Introducción

Actualmente las organizaciones necesitan ser competitivas y para esto, es necesario una eficiente gestión de operaciones que permita la optimización de los recursos disponibles y la satisfacción del cliente. El problema de programación de taller Job shop es uno de los problemas de programación de operaciones de optimización combinatoria más difíciles y es un ambiente común en los sistemas de manufactura reales. El Job shop consiste de n trabajos a ser procesados en m máquinas no relacionadas. Cada trabajo consta de un conjunto específico de operaciones, que deben procesarse de acuerdo con un orden determinado. Dada la importancia del problema se ha decidido tratar una generalización del Job Shop, el Job Shop Flexible; este se diferencia del mencionado anteriormente, donde cada operación puede ser procesada por un conjunto de máquinas y tiene un tiempo de procesamiento que depende de la máquina utilizada. El Job Shop Flexible (FJSP) incluye dos subproblemas: 1) asignación de máquinas: seleccionar una máquina de un conjunto de máquinas candidatas para las operaciones y en consecuencia, un tiempo de procesamiento que depende de la máquina utilizada y 2) la secuenciación: la programación de las operaciones en las máquinas asignadas para obtener una solución con el menor Makespan, es decir, el tiempo de finalización de la última operación programada.

El Job Shop Flexible (FJSP) ha sido un tema de interés por los investigadores desde el trabajo de Brucker y Schlie (1990) , ellos desarrollaron un algoritmo de tiempo polinomial que se basa en un método gráfico para resolver el FJPS con dos trabajos. Desde esa fecha hasta la actualidad se han utilizado diferentes métodos de solución como: ramificación y acotamiento, Programación Entera, Heurísticas, Metaheurísticas y técnicas hibridas. Las principales metaheurísticas son: Búsqueda Tabú (TS) (Brandimarte, 1993); (Hurink, Jurisch, & Thole, 1994); (Saidi-Mehrabad & Fattahi, 2007), Algoritmo Genético (GA) (Kacem, Hammadi, & Borne, 2002); (Pezzella, Morganti, & Ciaschetti, 2008); (Zhang, Gao, & Shi, 2011); (Teekeng & Thammano, 2012); Recocido Simulado (SA) (Najid, Dauzere-Peres, & Zaidat, 2002); (Cruz-Chávez, Martínez-Rangel, & Cruz-Rosales, 2017),Optimización por Enjambre de Partículas (PSO) (Teekeng, Thammano, Unkaw, & Kiatwuthiamorn, 2016); (Nouiri, Bekrar, Jemai, Niar, & Ammari, 2018); (Kato, Aranha, & Tsunaki, 2018), Variable neighborhood search (VNS) (Amiri, Zandieh, Yazdani, & Bagheri, 2010).

Si bien, han trabajado ampliamente el FJSP, considerar los tiempos de alistamiento dependientes de la secuencia (SDST) ha generado interés desde la comunidad académica.

Estos tiempos dependen del trabajo en sí y el trabajo anterior que se ejecutó en la misma máquina. Esta característica es importante en muchos entornos de programación reales como fabricación de productos químicos, impresión, farmacéuticos y de automóviles (Shen, Dauzère-pérès, & Neufeld, 2018). Las características incorporadas permiten que el SDST-FJSP se aproxime mejor a aplicaciones del mundo real que el JSP porque es común observar que exista más de un recurso disponible para realizar una tarea y teniendo en cuenta el tiempo necesario para el alistamiento del recurso. Este tipo de problema "es considerado NP-Hard, incorporándole la flexibilidad aumenta en gran medida la complejidad del problema, porque se requiere un nivel adicional de decisión, es decir, la selección de máquinas en las que debería realizarse el trabajo" (Brandimarte, 1993).

Pocos autores han abordado el SDST-FJSP. Choi y Choi (2002) plantearon un modelo de programación entera mixta y lo solucionaron por medio de un método de búsqueda local basado en varias reglas de despacho de tipo codicioso. Luego, (Imanipour, 2006) planteó un modelo de programación entera mixta no lineal y lo abordó mediante un Algoritmo Híbrido entre Búsqueda Tabú y Búsqueda de Vecindario Codiciosa (GNS). Saidi-Mehrabad y Fattahi (2007) utilizaron ambas metaheurísticas: Recocido simulado y Búsqueda tabú para los dos subproblemas del FJSP: Asignación y secuenciación. Bagheri y Zandieh (2011) emplearon Búsqueda de Vecindario Variable (VND). El algoritmo hibrido genético basado en Algoritmo Genético y Búsqueda Tabú propuesto en González, Vela, y Varela (2013), un Algoritmo Genético propuesto en Azzouz et al. (2016). Recientemente, Shen, Dauzère-pérès y Neufeld (2018) desarrollaron un algoritmo de Búsqueda Tabú con función específica de vecindario y estructura de diversificación, que permite mover operaciones no críticas de la solución.

En los últimos años la utilización de diferentes técnicas Metaheurísticas está tomando relevancia para la formulación de métodos híbridos como alternativa de solución del problema SDST.FJSP, como el de Azzouz, Ennigrou, y Ben Said, (2017) desarrollaron una metaheurística hibrida basada en Algoritmo Genético y Búsqueda de Vecindario Variable, un Algoritmo Híbrido Auto-Adaptativo (SAHA) que integra el Algoritmo genético (GA) y la Búsqueda Local Iterativa (ILS) propuesto en Azzouz et al. (2017). En el presente estudio, se propone un Algoritmo Genético Recocido Simulado (AHG), su estructura principal consiste en un Algoritmo Genético (GA) y se complementa usando el Recocido Simulado (SA) para generar parte de la población inicial.

Metodología

Primero, se presenta la descripción del problema y modelo matemático que lo representa y luego el Algoritmo Híbrido Genético.

El problema Job Shop Flexible, tiene un conjunto de máquinas M, M={M1,…,Mk}. Cada trabajo i consiste en una secuencia de operaciones ni. Cada operación i de un trabajo j, Oi,j la cual puede ejecutada en un conjunto de máquinas dado Mi,j, es decir por un número determinado de máquinas disponibles que pueden realizar cada operación Oi,j. El problema es definir la asignación de las máquinas a cada operación y la secuenciación en el que se determine el tiempo de inicio y finalización de cada operación.

Las condiciones y restricciones del problema son las siguientes:

- Las máquinas son independientes una de otra.

- Los trabajos son independientes de uno a otro. No hay restricciones de precedencia entre operaciones de diferentes trabajos. No obstante, existen restricciones de precedencia entre las operaciones del mismo trabajo.
- Todos los trabajos y máquinas están disponibles en el tiempo cero.
- En un momento dado, una máquina sólo puede ejecutar una operación.
- No se permite la anticipación. Es decir, cada operación debe ser ejecutada sin interrupción, una vez que se inicia.
- Los tiempos de configuración dependen de la secuencia de trabajos. Cuando una de las operaciones de un trabajo t se procesa antes que una de las del trabajo i ($t \neq i$) en la máquina Mk, el tiempo de configuración dependiente de la secuencia es S t, i, k> 0

El FJSP se clasifica en FJSP parcial y FJSP total. En el FJSP total todas las operaciones se pueden ejecutar en todas las máquinas. Por otro lado, en el FJSP parcial, al menos una operación no puede ser procesada en todas las máquinas. Varios investigadores como Kacem (2002) afirman que el FJSP parcial es más complejo que el FJSP total. En este artículo se considera el FJSP parcial. Para ilustrar el problema a resolver, se considera una instancia de 3 trabajos y 3 máquinas. En la Tabla 1 se muestran los tiempos de procesamiento y en la

Tabla 2 los tiempos de alistamiento.

Tabla 1 Tiempos de procesamiento

Trabajo	**Operación**	**M1**	**M2**	**M3**
	O11	4	-	5
1	O12	-	3	4
	O13	6	5	-
	O21	3	-	4
2	O22	4	5	-
	O23	-	4	7
	O31	5	3	-
3	O32	-	4	-
	O33	4	5	3

Tabla 2 Tiempos de alistamiento dependientes de la secuencia

	Máquina 1			Máquina 2			Máquina 3		
	T1	T2	T3	T1	T2	T3	T1	T2	T3
Dummy	3	2	1	2	4	1	4	3	2
T1	0	1	3	0	2	4	0	2	3
T2	1	0	2	4	0	3	2	0	4
T3	1	3	0	2	1	0	3	2	0

El algoritmo propuesto está basado en la hibridación entre dos Metaheurísticas ampliamente utilizadas en la solución de problemas de optimización combinatoria en programación de operaciones como: Algoritmo Genético (GA) y Recocido simulado (SA). La elección de estos algoritmos se justifica de la siguiente manera: utilizamos el GA como estructura principal en el que se aprovecha la capacidad de exploración y el Recocido Simulado por su capacidad de búsqueda local generando buenas soluciones, las cuales serán parte de la población inicial (ver **¡Error! No se encuentra el origen de la referencia.**). El SA tiene una estructura de vecindario de operaciones adyacentes y su temperatura inicial se determina en función de la solución inicial. En el GA, la selección se realiza por torneo, en el cruce se utilizaron dos operadores: el primero interviene la secuencia de las operaciones, operador Precedence Preserving Order-based (POX) propuesto por Lee, Yamakawa, y Lee, (1998) y el segundo para la asignación de las máquinas, operador de cruce de dos puntos. Por último, para la mutación se utiliza el operador Precedence Preserving Shift (PPS) utilizado por (He, Weng, & Fujimura, 2017), que modifica la secuencia de las operaciones respetando las restricciones de precedencia.

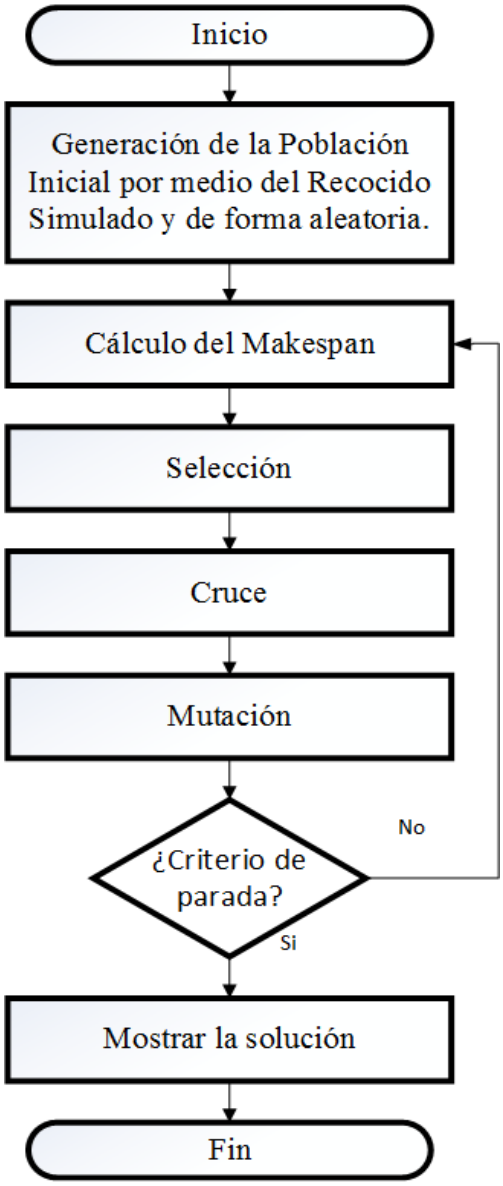

Figura 1 Diagrama de Flujo del Algoritmo Híbrido Genético.

Fuente autores

Algoritmo Genético: Este algoritmo se ha aplicado en diferentes problemas de Optimización combinatoria arrojando buenos resultados en comparación con otros métodos. El GA fue propuesto por Holland (1992) basado en la selección natural.

Representación de la solución

La manera de representar los individuos influye en la eficiencia del algoritmo. Se utilizó la representación de dos vectores propuesta por Gen, Gao, y Lin (2009). El primer vector representa la secuencia de operación en las máquinas, este utiliza la representación basada en operaciones, cubre todo el espacio de la solución y cualquier permutación de las

operaciones puede corresponder a un programa factible. El segundo vector representa la asignación de las máquinas, asigna la máquina dependiendo del conjunto de máquinas disponibles para realizar cada operación. En la Figura 2 se ilustra la representación de una solución para el ejemplo dado en la sección 2 y en la Figura 3 el Diagrama de Gantt correspondiente.

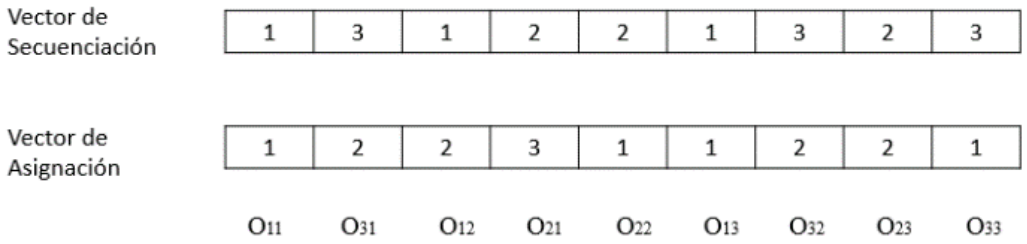

Figura 2 Representación de la solución

Fuente: Autores

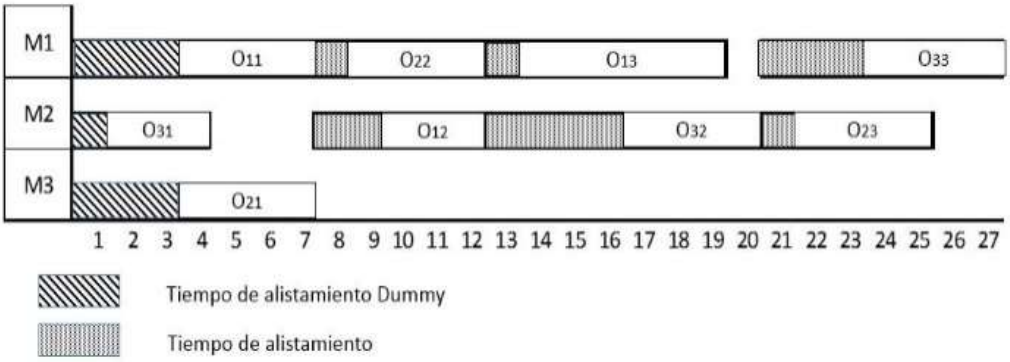

Figura 3. Diagrama de Gantt de la solución

Fuente: Autores

Población inicial

La población inicial influye de manera importante en el rendimiento del algoritmo genético. Usualmente la población inicial se genera de manera aleatoria, con el objetivo de generar buenas soluciones desde el inicio se propone el siguiente método: 20% usando el Recocido Simulado y 80% de forma aleatoria, basado en experimentos previos en el que se observó la mejora del Algoritmo Genético con diferentes porcentajes de población generados por el Recocido Simulado, siendo el del 20% el que proporcionaba mejorar en la calidad de las soluciones.

Selección

Para seleccionar los mejores individuos de la población se utilizó el criterio de selección por torneo de tamaño n. Aleatoriamente se escogen n individuos de la población, los cuales participan en un torneo de tamaño k en el que compiten, evaluando qué solución

es mejor a otra basándose en el fitness de cada uno, en este caso es el Makespan, y el ganador tendrá la opción de reproducirse, es decir, según el número de torneos ganados por un individuo será el número de descendientes. Esto se realiza hasta que el número de descendientes sea igual al tamaño de la población.

Operadores Genéticos

Para este tipo de problema se distinguen dos tipos de operadores. Los operadores de asignación y los operadores de secuenciación. Los operadores de asignación solo cambian la propiedad de asignación de los cromosomas, es decir la secuenciación de las operaciones se conserva en la descendencia. Por otro lado, los operadores de secuenciación sólo cambian la secuencia de las operaciones en los cromosomas de la descendencia, es decir, la asignación de operaciones a máquinas se conserva, estos operadores deben respetar las restricciones de precedencia.

Operadores de cruce

Para generar nuevas descendencias en la población se utilizaron 2 operadores. Para la asignación, el operador de dos puntos y para la secuenciación, el Operador POX.

El operador de cruce de dos puntos consiste en copiar los genes del primer padre que están entre los dos puntos de cruce (a y b) y se rellenan los faltantes con los genes del segundo padre Ver Figura 4.

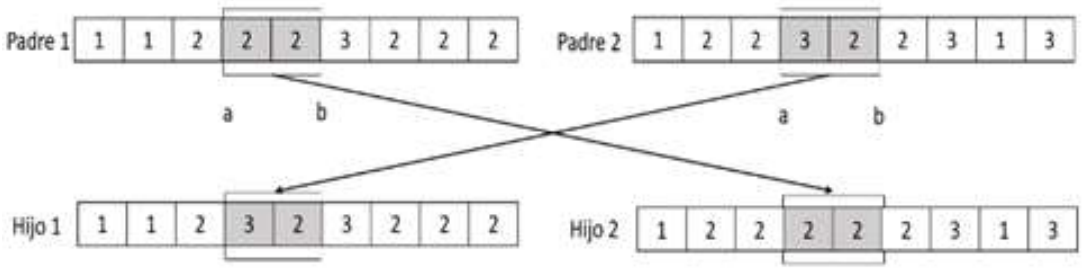

Figura 4 Operador de cruce de dos puntos

Fuente: Autores

El operador POX selecciona a los dos padres. Luego, determina para cada padre el conjunto de trabajos que serán heredados a su correspondiente hijo. Para el ejemplo mostrado en la Figura 6 el subconjunto de trabajos del padre 1 es {3,1} y el del padre 2 es {2}.) Los trabajos del padre 1, toman la misma posición en el hijo 1, igualmente para el trabajo del padre 2, toma la misma posición en el hijo2. Después se elimina del padre 1 los trabajos obtenidos del padre 2 y se coloca el resultado en el primer hijo según el mismo orden de aparición y viceversa para el segundo hijo.

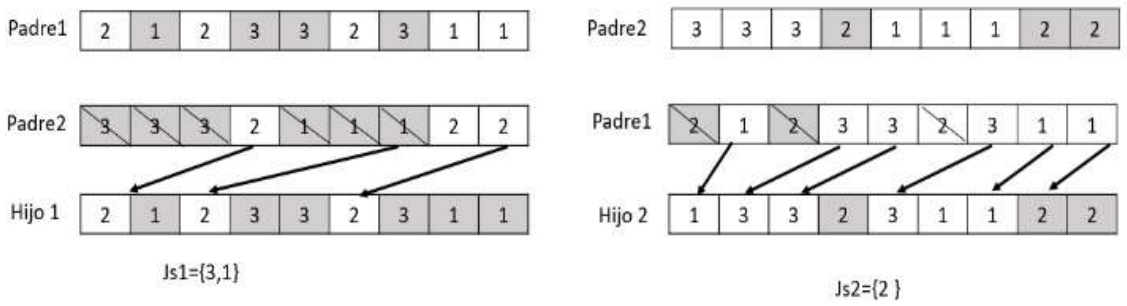

Figura 5. Operador POX

Fuente: Autores

Operadores de mutación

Con el fin de obtener una mayor diversidad en el espacio de soluciones se utilizó el operador PPS. Consiste en seleccionar un gen del vector de secuenciación de manera aleatoria, luego el gen se desplaza a otra posición, mientras que la operación que este gen representa en su trabajo no cambia. En la Figura 6 se selecciona aleatoriamente el tercer 3, representa la tercera operación del trabajo 3, por lo tanto, solo se puede desplazar entre el rango del segundo 3 y el tercer 3, los cuales representan la segunda y tercera operación del trabajo 3.

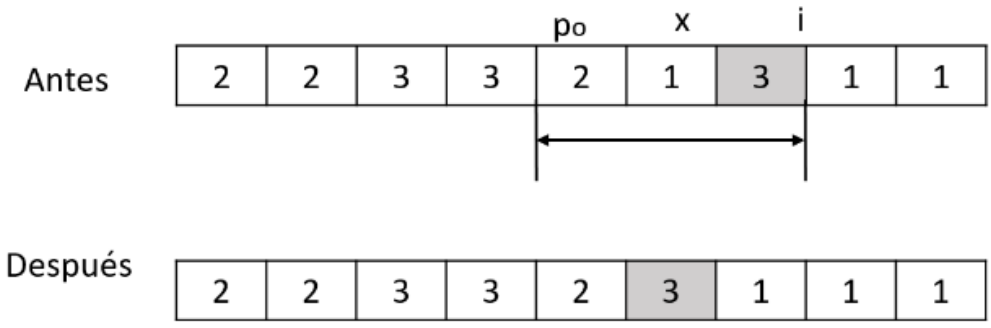

Figura 6. Operador PPS

Fuente: Autores

Recocido Simulado

La solución inicial se genera de manera aleatoria. La estructura de vecindario utilizada se denomina: Estructura de vecindario de secuenciación entre operaciones adyacentes. Los vecinos de una solución se generan intercambiando las posiciones de dos operaciones adyacentes, es decir, dos operaciones que pertenecen a la misma máquina, respetando las restricciones de precedencia.

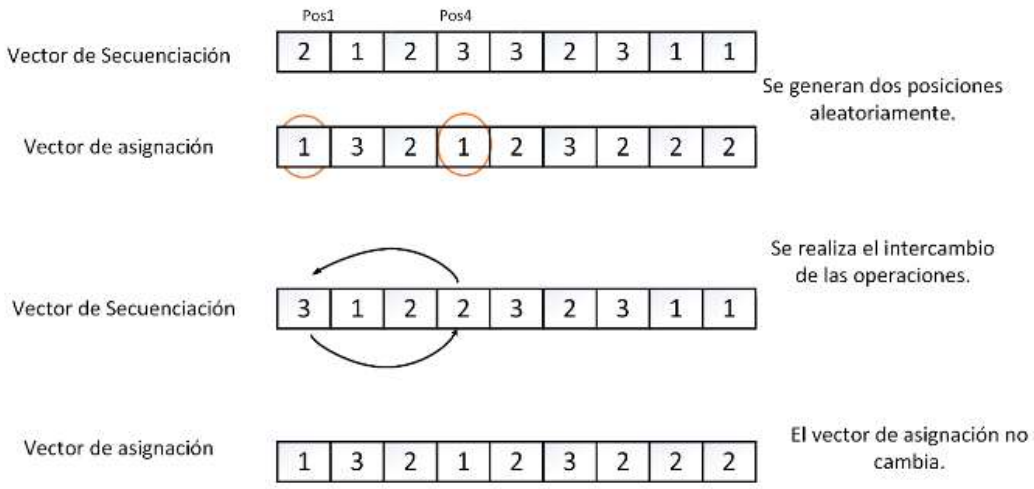

Figura 7. Estructura de vecindario de secuenciación entre operaciones adyacentes.

Fuente: Autores

Los parámetros del mecanismo de enfriamiento se calcularon de la siguiente manera:

Temperatura inicial: Se establece para que exista una probabilidad de 0.9 de aceptar soluciones que desmejoren la solución inicial hasta en un 20% del valor inicial como lo propuso (Bula, 2004). Ver ecuación 1.

$$Ti = e^{-\frac{\alpha x f o}{To}} \geq 0.9 \; para \; valores \; de \; \alpha \leq 0.2 \qquad (1)$$

Función de decremento: Se utilizó un modelo exponencial para realizar el enfriamiento, con un α de 0.5 el cual permite observar mejores soluciones ajustándose de mejor manera al criterio de parada. Ver ecuación 2.

$$Tk + 1 = Tk.\alpha \qquad (2)$$

Número de vecindario: Se estableció el vecindario tamaño 10.

Resultados

En esta sección se evalúa el rendimiento del algoritmo propuesto (SAGA), comparando los resultados con los encontrados en la literatura. El algoritmo se desarrolló en Matlab ® versión 2018b en un procesador Intel(R) Core (TM) i7-8700 CPU@ 3.20GHz 3.19GHz, Memoria (RAM) 8,00 GB. Se utilizaron 20 instancias propuestas por Bagheri et al. Los parámetros utilizados del Algoritmo Recocico Simulado Genético para cada clase de instancia se presentan en la Tabla 3 En cada uno de los problemas fue evaluado el algoritmo 10 veces y se seleccionó la mejor solución. Los nombres de las instancias están dados en la columna Instance. La segunda columna nxnixm contiene el tamaño de la instancia, número de trabajos (n), número de operaciones de cada trabajo (ni), número de máquinas (m). En las

siguientes 8 columnas se muestras los parámetros del AHG, distinguidos en Algoritmo Genético y Recocido Simulado.

Tabla 3 Parámetros SA

		Parameters SA		Parameters GA					
Instance	nxnixm	α	Vec	Psa	Pal	Pob	Pc	Pm	Gen
AEB01-AEB05	10x5x5	0.5	10	0.2	0.8	100	0.8	0.02	100
AEB06-AEB10	15x5x8	0.5	10	0.2	0.8	100	0.8	0.02	100
AEB11-AEB15	10x10x5	0.5	10	0.2	0.8	100	0.8	0.02	100
AEB16-AEB20	15x10x10	0.5	10	0.2	0.8	100	0.8	0.02	50

Los resultados computacionales son presentados en la tabla 4. Los nombres de las instancias están dados en la columna Instance, el tamaño en la columna tamaño y en el resto de las columnas se muestra el error relativo entre algoritmo propuesto SAGA con otros algoritmos como: Búsqueda Local Iterativa (ILS), Búsqueda de vecindario Variable (VNS), Algoritmo Genético (GA), Algoritmo Híbrido Genético (HGA) basado en GA y VNS, Algoritmo híbrido SAHA basado en Algoritmo Genético (GA) y la Búsqueda Local Iterativa (ILS) con dos versiones: SAHA(EE) y SAHA(AS), el último utiliza una estrategia auto adaptativa basada en: (1) la especificidad actual del espacio de búsqueda, (2) los resultados anteriores de algoritmos ya aplicados (GA e ILS) y (3) sus ajustes de los parámetros asociados.

Tabla 4 Comparación Algoritmo propuesto (SAGA) vs Algoritmos de la literatura

Instance	Size	ILS	VNS	GA	HGA	SAHA (EE)	SAHA (AS)
AEB01		-0.236	-0.208	-0.1306	0.106	0.049	0.131
AEB02		-0.254	-0.148	-0.101	0.089	0.011	0.104
AEB03	(1) 10x5x5	-0.262	-0.222	-0.053	0.103	0.038	0.116
AEB04		-0.208	-0.107	0.015	0.108	0.090	0.153
AEB05		-0.210	-0.183	-0.048	0.021	0.011	0.016
AEB06		-0.142	-0.037	0.022	0.127	0.173	0.180
AEB07		-0.259	-0.046	0.107	0.092	0.125	0.193
AEB08	(2) 15x5x8	-0.329	-0.131	0.045	0.093	0.135	0.196
AEB09		-0.285	-0.079	0.127	0.128	0.145	0.200
AEB10		-0.318	-0.018	0.083	0.106	0.137	0.173
AEB11		-0.323	-0.167	0.031	0.032	0.034	0.061
AEB12		-0.332	-0.115	-0.072	0.061	0.003	0.082
AEB13	(3) 10x10x5	-0.333	-0.081	-0.014	0.081	0.0638	0.135
AEB14		-0.374	-0.180	-0.035	0.045	-0.011	0.075
AEB15		-0.339	-0.213	0.014	0.061	0.077	0.131
AEB16		-0.378	-0.095	0.293	0.307	0.309	0.309
AEB17		-0.475	-0.115	0.400	0.406	0.400	0.406
AEB18	(4) 15x10x10	-0.363	-0.161	0.361	0.376	0.376	0.376
AEB19		-0.206	-0.109	0.506	0.517	0.506	0.515
AEB20		-0.400	-0.127	0.318	0.318	0.318	0.318

Los resultados en la

Tabla 4 muestran que el algoritmo propuesto, (SAGA) mejora los resultados de los algoritmos ILS y VNS en el 100% las instancias evaluadas y comparado con el GA en un 35%. En las instancias pequeñas 1, 3, medianas 2 el algoritmo propuesto estuvo cerca de las soluciones encontradas por HGA, SAHA (EE), SAHA (AS). Sin embargo, para las instancias de mayor tamaño 4 los resultados estuvieron entre 30 y 52% de la mejor solución encontrada por el SAHA(AS).

Conclusiones

En este artículo se presenta un Algoritmo Híbrido (SAGA) para la solución basado en el Algoritmo Genético y Recocido simulado. Se encontró que utilizando el Recocido Simulado para generar parte de la población inicial del Algoritmo Genético mejora el desempeño del mismo en los cuatro tamaños de instancias, siendo una buena alternativa para la solución de problemas complejos.

Para futuras investigaciones se recomienda ajustar los parámetros del algoritmo y modificar los operadores genéticos para mejorar el rendimiento del algoritmo. Por otro lado, considerar restricciones adicionales como: tiempos de transporte, averías en las máquinas, interrupciones, ventanas de tiempos de procesamiento, entre otras de tal manera que se pueda aproximar a entornos reales de fabricación.

Referencias bibliográficas

Amiri, M., Zandieh, M., Yazdani, M., & Bagheri, A. (2010). A variable neighbourhood search algorithm for the flexible job-shop scheduling problem. *International Journal of Production Research*, *48*(19), 5671–5689. https://doi.org/10.1080/00207540903055743

Azzouz, A., Ennigrou, M., & Ben Said, L. (2017). A self-adaptive hybrid algorithm for solving flexible job-shop problem with sequence dependent setup time. *Procedia Computer Science*, *112*, 457–466. https://doi.org/10.1016/j.procs.2017.08.023

Bagheri, A., & Zandieh, M. (2011). Bi-criteria flexible job-shop scheduling with sequence-dependent setup times—Variable neighborhood search approach. *Journal of Manufacturing Systems*, *30*(1), 8–15. https://doi.org/10.1016/J.JMSY.2011.02.004

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, *41*(3), 157–183. https://doi.org/10.1007/BF02023073

Brucker, P., & Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing*, *45*(4), 369–375. https://doi.org/10.1007/BF02238804

Bula, G. A. (2004). Programación de operaciones en el taller de trabajo utilizando la meta - heurística de recocido simulado. *Revista UIS Ingenierías*, *3*(1), 8. Retrieved from Se desarrolla una aplicación de la meta-heurística de recocido simulado a la programación enel taller de trabaja para dos configuracionesjlow-shop y job-shop. El desarrollo de los algoritmosse hace en el lenguaje de programación Java versión 1.3.

Choi, I.-C., & Choi, D.-S. (2002). A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups. *Computers & Industrial Engineering*, *42*(1), 43–58. https://doi.org/https://doi.org/10.1016/S0360-8352(02)00002-5

Cruz-Chávez, M. A., Martínez-Rangel, M. G., & Cruz-Rosales, M. H. (2017). Accelerated

simulated annealing algorithm applied to the flexible job shop scheduling problem. *International Transactions in Operational Research*, *24*(5), 1119–1137. https://doi.org/10.1111/itor.12195

Gen, M., Gao, J., & Lin, L. (2009). Multistage-based genetic algorithm for flexible job-shop scheduling problem. *Studies in Computational Intelligence*, *187*, 183–196. https://doi.org/10.1007/978-3-540-95978-6_13

González, M. A., Vela, C. R., & Varela, R. (2013). An efficient memetic algorithm for the flexible job shop with setup times. *ICAPS 2013 - Proceedings of the 23rd International Conference on Automated Planning and Scheduling*, 91–99.

He, Y., Weng, W., & Fujimura, S. (2017). Improvements to genetic algorithm for flexible job shop scheduling with overlapping in operations. *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 791–796. https://doi.org/10.1109/ICIS.2017.7960100

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*.

Hurink, J., Jurisch, B., & Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, *15*(4), 205–215. https://doi.org/10.1007/BF01719451

Imanipour, N. (2006). Modeling amp; Solving Flexible Job Shop Problem With Sequence Dependent Setup Times. *2006 International Conference on Service Systems and Service Management*, *2*, 1205–1210. https://doi.org/10.1109/ICSSSM.2006.320680

Kacem, I., Hammadi, S., & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *32*(1), 1–13. https://doi.org/10.1109/TSMCC.2002.1009117

Kato, E. R. R., Aranha, G. D. de A., & Tsunaki, R. H. (2018). A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing. *Computers and Industrial Engineering*, *125*(August), 178–189. https://doi.org/10.1016/j.cie.2018.08.022

Lee, K.-, Yamakawa, T., & Lee, K.-M. (1998). A genetic algorithm for general machine scheduling problems. *1998 Second International Conference. Knowledge-Based Intelligent Electronic Systems. Proceedings KES'98 (Cat. No.98EX111)*, *2*, 60–66 vol.2. https://doi.org/10.1109/KES.1998.725893

Najid, N. M., Dauzere-Peres, S., & Zaidat, A. (2002). A modified simulated annealing method for flexible job shop scheduling problem. *IEEE International Conference on Systems, Man and Cybernetics*, *5*, 6 pp. vol.5-. https://doi.org/10.1109/ICSMC.2002.1176334

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(3), 603–615. https://doi.org/10.1007/s10845-015-1039-3

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research*, *35*(10), 3202–3212. https://doi.org/https://doi.org/10.1016/j.cor.2007.02.014

Saidi-Mehrabad, M., & Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, *32*(5),

563–570. https://doi.org/10.1007/s00170-005-0375-4

Shen, L., Dauzère-pérès, S., & Neufeld, J. S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, *265*, 503–516. https://doi.org/10.1016/j.ejor.2017.08.021

Teekeng, W., & Thammano, A. (2012). Modified Genetic Algorithm for Flexible Job-Shop Scheduling Problems. *Procedia Computer Science*, *12*, 122–128. https://doi.org/10.1016/j.procs.2012.09.041

Teekeng, W., Thammano, A., Unkaw, P., & Kiatwuthiamorn, J. (2016). A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization. *Artificial Life and Robotics*, *21*(1), 18–23. https://doi.org/10.1007/s10015-015-0259-0

Zhang, G., Gao, L., & Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, *38*(4), 3563–3573. https://doi.org/10.1016/j.eswa.2010.08.145