Información Tecnológica
Vol. 29(2), 141-154 (2018)
http://dx.doi.org/10.4067/S0718-07642018000200141

Desempeño de Técnicas Tradicionales de Programación de la Producción Frente a un Algoritmo Evolutivo

Omar D. Castrillón, William Sarache y Santiago Ruiz
Universidad Nacional de Colombia, Sede Manizales, Facultad de Ingeniería y Arquitectura, Departamento de Ingeniería Industrial, Campus la Nubia, Manizales, Código Postal 170001, Colombia. (e-mail: odcastrillong@unal.edu.co)

Recibido May. 17, 2017; Aceptado Jul. 17, 2017; Versión final Ago. 29, 2017, Publicado Abr. 2018

Resumen

Se comparan los resultados de algunas técnicas tradicionales para la solución de problemas de programación de la producción en sistemas Job Shop-Open Shop (*FIFO, LIFO, SPT, LPT*) con los de un algoritmo evolutivo del campo de la inteligencia artificial. Se presentan los resultados de un análisis exhaustivo de 12600 problemas de programación de la producción en sistemas Job Shop-Open Shop, comparando las diversas técnicas. Si bien en la mayor parte de los experimentos el algoritmo evolutivo arrojó un mejor desempeño en términos del tiempo máximo de procesamiento (*makespan)*, en algunos casos particulares la utilización de técnicas tradicionales fueron igualmente eficientes. Aunque que se piensa que el uso de técnicas avanzadas provee mejores resultados en todo problema de programación de la producción, esto no siempre suele ser cierto.

Palabras clave: *reglas de prioridad; algoritmos evolutivos; job shop: open shop; programación de la producción*

Performance of Traditional Production Scheduling Techniques Versus an Evolutionary Algorithm

Abstract

Comparison of some traditional techniques for the solution of production scheduling problems in Job Shop-Open Shop systems (FIFO, LIFO, SPT, LPT) with those of an evolutionary algorithm of the area of artificial intelligence is done. The results of an exhaustive analysis of 12600 production scheduling problems in Job Shop-Open Shop systems are presented. Although in most of the experiments the evolutionary algorithm showed a better performance in terms of the makespan, in some particular cases the use of traditional techniques were equally efficient. Despite the use of advanced techniques often provides better results for most production scheduling problem, this is not always true.

Keywords: *priority rules; evolutionary algorithms; job shop-open shop; production scheduling*

INTRODUCCIÓN

El principal objetivo de cualquier empresa es producir bienes y servicios con el fin de satisfacer las necesidades de los clientes. Para lograrlo, se deben programar diversos recursos tales como maquinaria, materiales y mano de obra, entre otros, con miras a lograr el máximo desempeño del sistema de operaciones. En una fábrica típica, la programación de la producción requiere un análisis detallado de los pedidos de los clientes para establecer la secuencia de procesamiento que optimice el uso de los recursos. En pocas palabras, el objetivo es reducir el tiempo de procesamiento (*makespan*) y el costo de producción estableciendo, para cada máquina o trabajador, cuándo debe iniciar y terminar cada pedido (Pérez, 2015).

Dependiendo de la configuración del sistema de producción, la programación puede ser una actividad simple o altamente compleja. Por ejemplo, en líneas de ensamblaje, la secuencia de fabricación se maneja como un problema de N pedidos - 1 máquina (N / 1), debido a que este sistema de producción funciona como un proceso integrado. Por el contrario, en los sistemas de fabricación flexibles (talleres organizados por función) la programación es una actividad más compleja (N pedidos - M máquinas) y cada pedido debe seguir un flujo de procesamiento diferente. En consecuencia, en los sistemas flexibles, la programación se convierte en un problema de optimización combinatoria NP-duro, debido a un conjunto de restricciones que deben ser consideradas (Shen et al., 2015).

Según la literatura, uno de los sistemas de producción flexibles más discutidos es el Sistema Job Shop: Open Shop. En este sistema, el conjunto de trabajos puede ser procesado en un orden arbitrario lo cual hace más compleja su programación. El objetivo es establecer la hora de inicio y finalización de cada uno de los pedidos en cada máquina o estación de trabajo (Koulamas y Kyparisis, 2015). El programador se enfrenta entonces a un problema de optimización combinatoria con $(N!)^M$ soluciones posibles. Este problema puede ser resuelto utilizando técnicas tradicionales de programación o aplicando métodos avanzados como la inteligencia artificial (IA). Desde la perspectiva de las ciencias computacionales, este tipo de problema es conocido como *Open-Shop: Scheduling Problem* (OSSP).

Las técnicas tradicionales de programación también se conocen como reglas de prioridad. Estas reglas son muy útiles para asignar la secuencia de procesamiento de pedidos en cada estación de trabajo. Sin embargo, conducen a una solución aceptable que, por lo general, es mucho menor en comparación con el resultado óptimo. Entre ellas, las más utilizadas son *Short Process Time* (*SPT*), *Long Process Time* (*LPT*), *Earliest Due Date* (*EDD*), *First In First Out* (*FIFO*), *Last In First Out* (*LIFO*), entre otras (Russell y Taylor, 2010; Koulamas y Kyparisis, 2015). Aunque las técnicas de IA logran mejores resultados que las reglas de prioridad, estas requieren mayores esfuerzos computacionales para diseñar los algoritmos de solución, especialmente en problemas de alta complejidad como el OSSP. Las técnicas de IA más utilizadas son: algoritmos evolutivos (AE), minería de datos, algoritmos aleatorios y agentes inteligentes, entre otras (Panahi y Tavakkoli-Moghaddam, 2015; Gupta y Singh, 2015; Liaw, 2013). Al comparar las técnicas IA (específicamente los algoritmos evolutivos AE) con las técnicas tradicionales, algunos estudios previos realizados por los autores del presente artículo, centrados en la resolución de problemas OSSP (Castrillón et al., 2010; Castrillón et al., 2011; Ruiz et al., 2012), indican que, en términos del *makespan*, los AE funcionan mejor que las técnicas tradicionales; sin embargo, estos resultados han sido observados en problemas muy específicos y, por lo tanto, el tema sugiere mayor análisis.

Una breve revisión de las principales investigaciones en este campo de estudio muestra que la IA frecuentemente ha sido aplicada para resolver problemas de programación de talleres (Sangsawang et al., 2015; Sahraeian y Namakshenas, 2015; Peng et al., 2015; Palacios et al., 2015; Frutos y Tohmé, 2015; Karimi-Nasab et al., 2015; Lei y Guo, 2015; Salazar et al., 2015). Sin embargo, no se identificó un estudio que compare el rendimiento global de las técnicas tradicionales de programación y las técnicas basadas en IA. Por otro lado, varias configuraciones para el OSSP pueden ser encontradas en contextos reales de fabricación. Estas configuraciones dependen principalmente de parámetros como el número de trabajos, el número de máquinas, el tiempo de procesamiento de los trabajos en cada máquina y el tiempo total de procesamiento de los trabajos. Algunos análisis preliminares (Sarache et al., 2011) muestran que, dependiendo de la configuración del problema, el rendimiento de las técnicas tradicionales de programación puede ser casi tan bueno como una solución obtenida a través de una IA. La diferencia entre estas dos técnicas puede ser inferior al 10% e, incluso, llegar a ser cero en algunos casos.

Específicamente, los AE han sido ampliamente empleados en la solución de esta clase de problemas. Trabajos previos realizados por los autores del presente artículo y otros documentos referenciados, concluyen de forma general, que las técnicas basadas en algoritmos evolutivos son mejores que las técnicas tradicionales. No obstante, tales estudios tampoco establecen una diferencia realmente significativa entre la mayoría de técnicas de inteligencia artificial empleadas. De otro lado, es importante resaltar que si bien algunas investigaciones utilizan el tiempo de entrega como variable de optimización, otros se centran

en el analisis del *makespan* (López y Arango., 2015), pues este provee una excelente métrica de desempeño. Un menor tiempo de procesamiento permitirá ahorros en diversos recursos tales como mano de obra, maquinaria y energía, entre otros.

Este artículo se basa en una investigación exploratoria, con el fin de comparar la eficacia de las técnicas de programación inteligente y las técnicas tradicionales. Estas comparaciones han sido poco discutidas en la literatura, ya que en todos los casos se presume que las técnicas inteligentes son superiores a las tradicionales. Por lo tanto, el objetivo del presente artículo es identificar en qué configuraciones de programación el rendimiento de algunas técnicas tradicionales puede ser tan bueno como un algoritmo evolutivo (AE). Desde esta perspectiva, comparando el rendimiento (*makespan*) de cuatro técnicas tradicionales (FIFO, LIFO, SPT y LPT) y AE, se analizaron 12.600 configuraciones de OSSP. De acuerdo con los resultados experimentales, el *makespan* obtenido con el AE tiende a ser mejor que el encontrado en las técnicas tradicionales lo cual es consistente con los trabajos identificados en la literatura. Sin embargo, la brecha de rendimiento depende, en gran medida, de la configuración del problema.

METODOLOGÍA

La metodología empleada en el estudio siguió los siguientes pasos: i) construcción de las matrices de problemas; ii) diseño del algoritmo evolutivo; iii) solución de problemas; y iv) comparación de las técnicas aplicadas.

Paso 1. Construcción de las matrices de problemas. Los problemas fueron definidos mediante matrices de tamaño NxM. En estas, las filas representan los trabajos y las columnas las máquinas (ver Tabla 1). Una combinación de filas y columnas representa un problema particular de programación *Job Shop*. Por ejemplo, en una matriz de 30x30, se deben analizar 900 problemas. Por otra parte, cada intersección entre una fila y una columna (T_{ij}) representa el tiempo de procesamiento del trabajo *i* en la máquina *j*.

Tabla 1: Matriz OSSP

Trabajos/Máquinas	*Máquina 1*	*Máquina 2*	*…..*	*Máquina M*
Trabajo 1				
Trabajo 2				
.		T(i,j)		
Trabajo N				

Paso 2. Diseño del algoritmo evolutivo. Para la construcción del algoritmo evolutivo, se procede de la la manera siguiente:

Población Inicial. Se define una problación inicial de K padres. Cada padre representa la solución inicial del problema y corresponde a un vector particular (ver Tabla 2). Lo anterior puede ser expresado como $M \in P$, $\mathrm{M} = \{\mathrm{Trab}_{n-1}, \mathrm{Trab}_{n-2}, \mathrm{Trab}_{n-3}, \mathrm{Trab}_{i}, \dots, \mathrm{Trab}_{2}, \mathrm{Trab}_{1}\}$; donde, $i \in \{1, 2, 3, 4, \dots n\}$, M representa los trabajos programados en la máquina i y P es el conjunto de máquinas.

Tabla 2: Estructura de padres (adaptado de Castrillón et al., 2011)

Trabajos localizados en Máquina 1				Trabajos localizados en Máquina 2				Trabajos localizados en Máquina 3			
$Trab_1$	$Trab_2$	…	$Trab_n$	$Trab_1$	$Trab_2$	…	$Trab_n$	$Trab_1$	$Trab_2$	…	$Trab_n$

Operador Genético. Con el fin de obtener nuevas soluciones se utilizaron operadores de mutación (3%) y combinación (97%). Usando un diagrama de Gantt, se evaluó el rendimiento de cada solución existente de acuerdo con el *makespan* obtenido. Luego, las mejores soluciones deben ser elegidas para la siguiente iteración. Como operador de cruce se usó el denominado *Partial Matching Crossover*, el cual puede ser definido de la siguiente forma: Dados dos padres iniciales $M_l = \{\mathrm{Trab}_{n-1}, \mathrm{Trab}_{n-2}, \dots, \mathrm{Trab}_{2}, \mathrm{Trab}_{1}\}$ y $M_k = \{\mathrm{Trab}_{n-1}, \mathrm{Trab}_{n-2}, \dots, \mathrm{Trab}_{2}, \mathrm{Trab}_{1}\}$, parte del arreglo M_l es reemplazado por parte del arreglo M_k ; subsecuentemente, las réplicas son eliminadas y los genes faltantes son complementados. Por otro lado, el proceso de mutación puede ser definido como sigue: Dado un padre inicial $M_l = \{Trab_{i-1}, Trab_{i-2}, Trab_{i-3}, Trab_{i}, \dots, Trab_{2}, Trab_{1}\}$ un elemento de M_l es seleccionado aleatoriamente y su posición es cambiada desplazando los elementos restantes. Información más detallada sobre operadores genéticos, puede ser encontrada en Cazacu (2017).

Condición de parada. El proceso finaliza cuando el algoritmo ha alcanzado una solución óptima o sub óptima. Una solución óptima es obtenida cuando la secuencia de n trabajos en alguna de las máquinas no presenta tiempo muerto o este es cero. Es decir, cuando $\sum_{i=1}^{n} Tiempomuerto_{i,(i-1)} = 0$; en contraste una solución sub óptima, aparece cuando se realizan cierto número de iteraciones sin encontrar una mejor solución. El *makespan* se define como la suma de todos los tiempos muertos obtenidos entre los trabajos *i* e (*i-1*) programados sobre todas las máquinas M_j, más todos los tiempos de procesamiento de todos los trabajos secuenciados sobre todas las maquinas. Matemáticamente, se puede expresar como: $Valor = \sum_{i=1}^{n} Tiempomuerto_{i,(i-1)} + \sum_{i=1}^{n} Tiempoproceso_{i,mj}$ (Salazar et al., 2011).

Estabilidad. Con el fin de garantizar la estabilidad del sistema, se llevó a cabo un análisis de varianza (ANOVA) usando la ecuación 1. La información recolectada debe cumplir las condiciones de independencia y normalidad.

$$y_i = \mu + T_i + \varepsilon_i \quad (1)$$

Dónde, yi es el nivel de respuesta de las variables; μ es la media; Ti es el efecto producido por el tratamiento *i* y ε_i represente el rror experimental i.

Paso 3. Solución de problemas. Para todos los problemas definidos se calculó el *makespan*. Para tal fin, se emplearon cuatro reglas de prioridad (*FIFO, LIFO, SPT, LPT*) y un algoritmo evolutivo como se describió en el paso 2. Las reglas de prioridad pueden ser definidas como secuencias basadas en el siguiente orden: *LIFO*: $M_l = \{\mathrm{Trab}_n, \mathrm{Trab}_{n-1}, \mathrm{Trab}_{n-2}, \mathrm{Trab}_i, \ldots, \mathrm{Trab}_2, \mathrm{Trab}_1\}$; su secuenciación se hace en forma inversa al orden de llegada establecido en la matriz (Tabla 4). *FIFO*: $M_l = \{\mathrm{Trab}_1, \mathrm{Trab}_2, \mathrm{Trab}_3, \mathrm{Trab}_4, \ldots, \mathrm{Trab}_i, \mathrm{Trab}_n\}$; de forma similar su secuenciación se da según el orden de llegada definido en la matriz (Tabla 4). *SPT*: $M_l = \{\mathrm{Trab}_k, \mathrm{Trab}_{k-1}, \mathrm{Trab}_{k-2}, \mathrm{Trab}_i, .., \mathrm{Trab}_1\}$, donde $\mathrm{Tiempodeproceso}_{trabi} <= \mathrm{Tiempoproceso}_{trab(i-1)}$ e $i \in \{1, 2, 3, 4, \ldots n\}$. *LPT*: $M_l = \{\mathrm{trab}_k, \mathrm{trab}_{k-1}, \mathrm{trab}_{k-2}, \mathrm{trab}_i, .., \mathrm{trab}_1\}$, donde $\mathrm{Tiempoproceso}_{trabi} >= \mathrm{Tiempoproceso}_{trab(i-1)}$ e $i \in \{1, 2, 3, 4, \ldots n\}$.

Paso 4. Comparación de las técnicas aplicadas. De acuerdo con la estructura del problema descrita en el paso 1, se estableció un conjunto de matrices de respuesta de tamaño NxM. Para cada una de las técnicas empleadas, se construyó una matriz de respuesta con el fin de facilitar un análisis comparativo entre las técnicas analizadas.

RESULTADOS

Paso 1. Construcción de las matrices de problemas. Con el fin de realizar el experimento un conjunto de problemas *Job Shop: Open Shop*, fueron representados usando una matriz de 30x30. Para este caso en particular, cada matriz representa 900 problemas. En resumen, cuatro conjuntos de problemas fueron analizados:

Conjunto 1: 5.400 problemas en los cuales los tiempos de procesamiento, para un trabajo en particular, son iguales (Tabla 3) o similares (Tabla 4) en todas las máquinas. Estos problemas se representaron en dos matrices de tamaño 30x30.

Conjunto 2: 5.400 problemas en los que, para una máquina en particular, los tiempos de procesamiento de todos los trabajos son iguales o similares. Estos problemas se representaron en dos matrices que corresponden a la matriz transpuesta obtenida de las Tablas 3 y 4. En la matriz transpuesta obtenida de la Tabla 3, los tiempos de procesamiento para un determinado trabajo son iguales en todas las máquinas mientras que en la matriz transpuesta obtenida de la Tabla 4, estos son similares.

Conjunto 3: 900 problemas en los que todos los trabajos tienen el mismo tiempo de procesamiento en todas las máquinas.

Conjunto 4: 900 problemas en los que los tiempos de procesamiento son diferentes para todos los trabajos en todas las máquinas.

Las columnas de las matrices representadas en los conjuntos 1 y 2 se analizaron en orden ascendente, descendente y aleatoriamente; por consiguiente, y para cada conjunto, se analizaron un total de 5.400 problemas (1800x3). Esto, sumado a los 1.800 problemas de los conjuntos 3 y 4, representó un total de 12.600 problemas. Si bien el análisis de un problema en orden ascendente, descendente y aleatorio no necesariamente genera problemas con estructuras diferentes, en la mayoría de los casos este aspecto genera un *makespan* distinto, aspecto de gran utilidad en este trabajo dado que esta variable es fundamental en los análisis que se realizan.

Tabla 3: Tiempos de procesos iguales para cada máquina

		Máquinas				
T R A B A J O S		M_1	M_2	….	M_{29}	M_{30}
	$Trab_1$	1	2	…	29	30
	$Trab_2$	1	2	…	29	30
	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
	$Trab_{30}$	1	2	…	29	30

Tabla 4: Tiempo de procesos similares, por columna, para cada máquina

	Máquinas																													
T R A B A J O S	5	11	1	20	31	15	51	15	1	11	6	12	2	12	32	16	52	13	2	12	8	19	1	22	30	21	51	15	10	20
	6	12	2	12	32	16	52	13	2	12	5	13	4	22	33	17	53	15	3	13	6	17	3	28	31	22	52	17	9	19
	5	13	4	22	33	17	53	15	3	13	7	14	5	23	31	18	52	12	4	14	2	14	7	27	32	23	54	13	8	18
	7	14	5	23	31	18	52	12	4	14	8	11	6	22	34	19	51	13	5	15	1	13	2	25	37	22	57	12	7	17
	8	11	6	22	34	19	51	13	5	15	9	12	3	21	35	21	56	11	6	16	1	13	2	25	37	22	57	12	7	17
	9	12	3	21	35	21	56	11	6	16	1	13	2	25	37	22	57	12	7	17	8	11	6	22	34	19	51	13	5	15
	1	13	2	25	37	22	57	12	7	17	2	14	7	27	32	23	54	13	8	18	7	14	5	23	31	18	52	12	4	14
	2	14	7	27	32	23	54	13	8	18	6	17	3	28	31	22	52	17	9	19	5	13	4	22	33	17	53	15	3	13
	6	17	3	28	31	22	52	17	9	19	8	19	1	22	30	21	51	15	10	2	7	14	5	23	31	18	52	12	4	14
	8	19	1	22	30	21	51	15	10	20	5	1	1	20	31	15	51	15	1	11	8	19	1	22	30	21	51	15	1	20
	5	11	1	20	31	15	51	15	1	11	6	12	2	12	32	16	52	13	2	12	8	19	1	22	30	21	51	15	10	20
	6	12	2	12	32	16	52	13	2	12	5	13	4	22	33	17	53	15	3	13	6	17	3	28	31	22	52	17	9	19
	5	13	4	22	33	17	53	15	3	13	7	14	5	23	31	18	52	12	4	14	2	14	7	27	32	23	54	13	8	18
	7	14	5	23	31	18	52	12	4	14	8	11	6	22	34	19	51	13	5	15	1	13	2	25	37	22	57	12	7	17
	8	11	6	22	34	19	51	13	5	15	9	12	3	21	35	21	56	11	6	16	1	13	2	25	37	22	57	12	7	17
	9	12	3	21	35	21	56	11	6	16	1	13	2	25	37	22	57	12	7	17	8	11	6	22	34	19	51	13	5	15
	1	13	2	25	37	22	57	12	7	17	2	14	7	27	32	23	54	13	8	18	7	14	5	23	31	18	52	12	4	14
	2	14	7	27	32	23	54	13	8	18	6	17	3	28	31	22	52	17	9	19	5	13	4	22	33	17	53	15	3	13
	6	17	3	28	31	22	52	17	9	19	8	19	1	22	30	21	51	15	10	2	7	14	5	23	31	18	52	12	4	14
	8	19	1	22	30	21	51	15	10	20	5	1	1	20	31	15	51	15	1	11	8	19	1	22	30	21	51	15	1	20
	5	11	1	20	31	15	51	15	1	11	6	12	2	12	32	16	52	13	2	12	8	19	1	22	30	21	51	15	10	20
	6	12	2	12	32	16	52	13	2	12	5	13	4	22	33	17	53	15	3	13	6	17	3	28	31	22	52	17	9	19
	5	13	4	22	33	17	53	15	3	13	7	14	5	23	31	18	52	12	4	14	2	14	7	27	32	23	54	13	8	18
	7	14	5	23	31	18	52	12	4	14	8	11	6	22	34	19	51	13	5	15	1	13	2	25	37	22	57	12	7	17
	8	11	6	22	34	19	51	13	5	15	9	12	3	21	35	21	56	11	6	16	1	13	2	25	37	22	57	12	7	15
	9	12	3	21	35	21	56	11	6	16	1	13	2	25	37	22	57	12	7	17	8	11	6	22	34	19	51	13	5	14
	1	13	2	25	37	22	57	12	7	17	2	14	7	27	32	23	54	13	8	18	7	14	5	23	31	18	52	12	4	13
	2	14	7	27	32	23	54	13	8	18	6	17	3	28	31	22	52	17	9	19	5	13	4	22	33	17	53	15	3	14
	6	17	3	28	31	22	52	17	9	19	8	19	1	22	30	21	51	15	10	2	7	14	5	23	31	18	52	12	4	20
	8	19	1	22	30	21	51	15	10	20	5	1	1	20	31	15	51	15	1	11	8	19	1	22	30	21	51	15	1	20

Paso 2. Diseño del algoritmo evolutivo. El algoritmo se ensayó en una matriz de tamaño 10x10 (véase el área sombreada en la Tabla 4). Se seleccionó este tamaño de matriz con el fin de evitar números muy grandes que dificulten la comprensión del artículo. Para ello, se realizaron 4 ensayos de 10 réplicas, conforme se describe en el paso 2 de la metodología. Los ensayos se realizaron reiniciando el computador y las réplicas volviendo a correr el programa (ver Tabla 5). Posteriormente, con un intervalo de confianza del 99,5%, se realizó el análisis de varianza. Como puede verse en la Tabla 6, el algoritmo fue consistente, dado que $F_{tab} > F_{cal}$.

Tabla 5: Matriz de 10x10. Cuatro ensayos – 10 réplicas.

Ensayos	Réplicas										
	1	2	3	4	5	6	7	8	9	10	Total
1	529	529	531	529	529	529	529	529	529	529	5292
2	533	529	529	529	529	529	536	540	529	531	5314
3	543	529	534	529	529	529	529	533	529	529	5313
4	540	529	529	540	529	534	529	533	538	538	5339
Esta tabla se incluye para facilitar la repetición del experimento por parte de un tercero.											

Tabla 6: Análisis de varianza (Anova). Significancia 0.05

Fuente de Varianza	*Grados de libertad*	*Suma de cuadrados*	*Cuadrado promedio*	F_{cal}	F_{tab}
Suma total de cuadrados		635,90			
Tratamiento	3,00	110,90	36,97	2,53	5,06
Error experimental	36,00	525,00	14,58		
Total	39,00	635,90	51,55		

Con las primeras cinco filas y cuatro primeras columnas de la Tabla 4, se construyó una matriz de 5x4, en la cual se probó el algoritmo descrito en el paso 2 de la metodología. En esta grafica se observa de forma más clara la solución óptima obtenida. Este resultado se puede inferir debido a que la máquina 3 (*Processor 3*) no muestra tiempos de inactividad y, por lo tanto, la reducción del tiempo total de proceso es inviable. Para este problema en particular, se obtuvieron 7.962.624 soluciones posibles. Es importante resaltar que en este paso se tomaron matrices de un tamaño inferior, con el fin de generar números y gráficas más pequeñas, que ilustraran de una mejor forma este concepto.

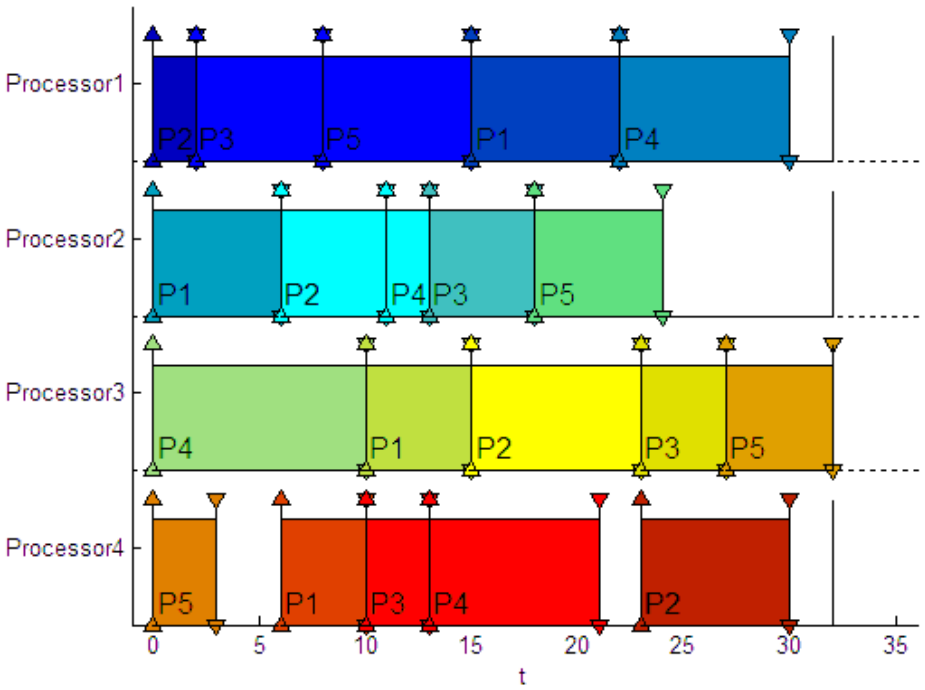

Fig. 1: Diagrama de Gantt para una matriz de 5x4

Pasos 3-4. Para cada técnica aplicada, se obtuvo la superficie de respuesta respectiva. Posteriormente, para cada superficie obtenida usando una técnica tradicional, se restó la superficie obtenida por medio de los algoritmos evolutivos. Los resultados se describen como sigue:

Conjunto 1. Las superficies de respuesta se representan en las Figuras 2 (resultados de la Tabla 3) y 3 (resultados de la Tabla 4). Un análisis detallado de las figuras 2a, 2b, 2c, 2d, 3a y 3b, permite observar que cuando el número de órdenes es pequeño, el intervalo porcentual (diferencia del *makespan* entre ambos algoritmos) tiende a ser pequeño e incluso cercano a cero. En otras palabras, el *makespan* obtenido por técnicas tradicionales (*FIFO* y *LIFO*) es similar al obtenido por los AE. Sin embargo, cuando el número de variables independientes (trabajos y máquinas) aumentó, esta brecha también lo hizo. Esto indica que, para los problemas de mayor complejidad, los AE se comportan mejor.

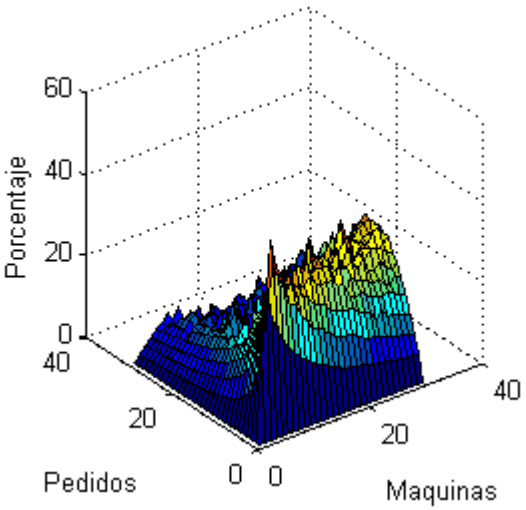

Figura 2a: Fifo menos genetico.

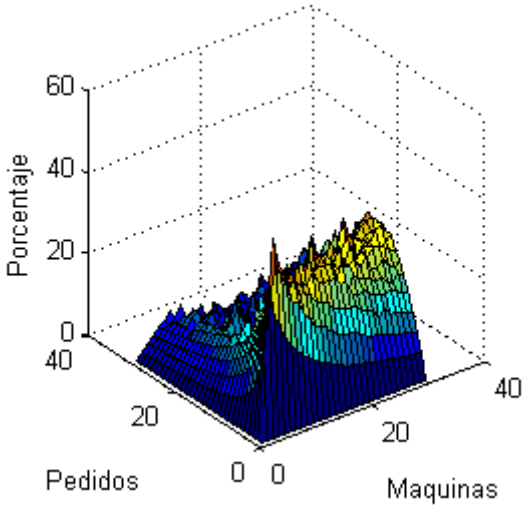

Figura 2b: Lifo menos genetico

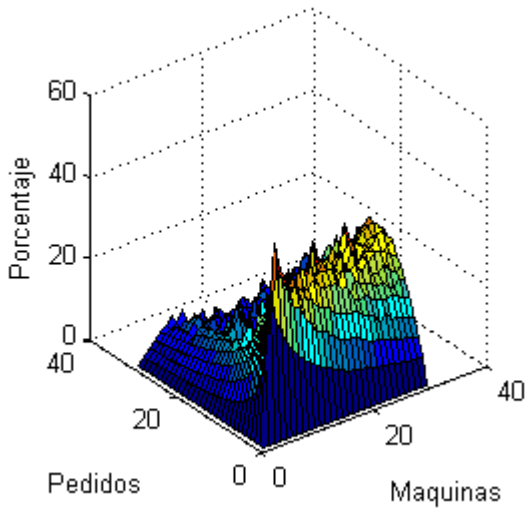

Figura 2c: SPT menos genetico

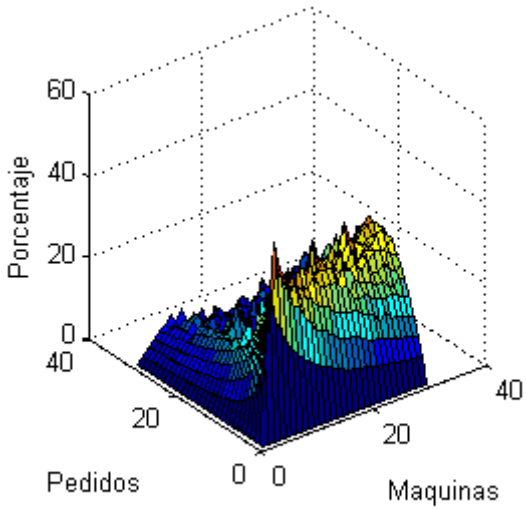

Figura 2d: LPT menos genetico

Fig. 2: Graficas a, b, c, d. Diferencia entre técnicas tradicionales y AE. Tiempos de procesos tomados de la Tabla 3. 900 problemas Job shop: Open shop.

A su vez, las Figuras 3c y 3d representan la diferencia entre las técnicas tradicionales *SPT* y *LPT* versus los AE. El este caso, el *makespan* obtenido por los AE fue hasta un 50% mejor comparado con estas dos técnicas tradicionales. Esta brecha fue aún mayor cuando se incrementó el número de trabajos y máquinas.

Conjunto 2. La Figura 4 muestra la respuesta superficial de la matriz transpuesta obtenida de la Tabla 3. De la misma manera, la respuesta superficial correspondiente a la matriz transpuesta obtenida de la Tabla 4 se representa en la Figura 5. De acuerdo con las Figuras 4a, 4b, 4c y 4d, cuando el número de máquinas es alto y el número de trabajos es pequeño, la diferencia entre las técnicas tradicionales e inteligentes tiende a ser pequeña e, incluso, cercana a cero. Por lo tanto, en estos casos, el rendimiento de las técnicas tradicionales es casi tan eficiente como el de los AE. Sin embargo, esta brecha aumentó cuando el número de trabajos fue alto, obteniéndose mejor desempeño con los AE.

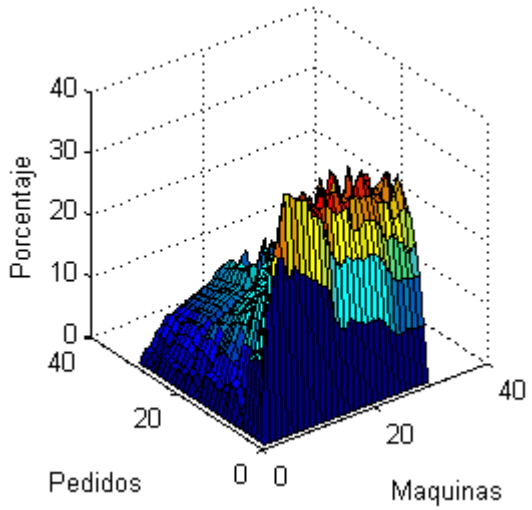

Figura 3a: Fifo menos genetico.

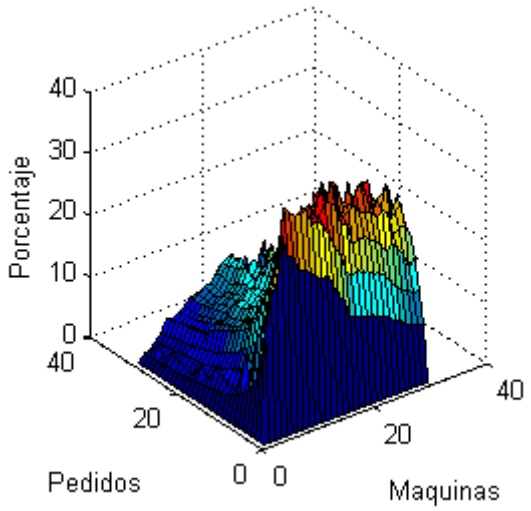

Figura 3b: Lifo menos genetico.

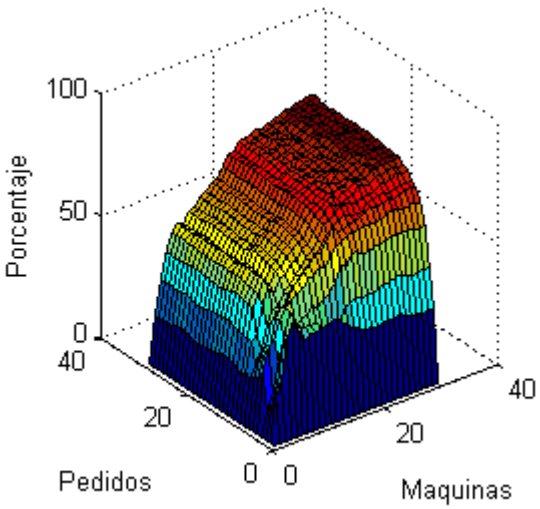

Figura 3c: SPT menos genetico.

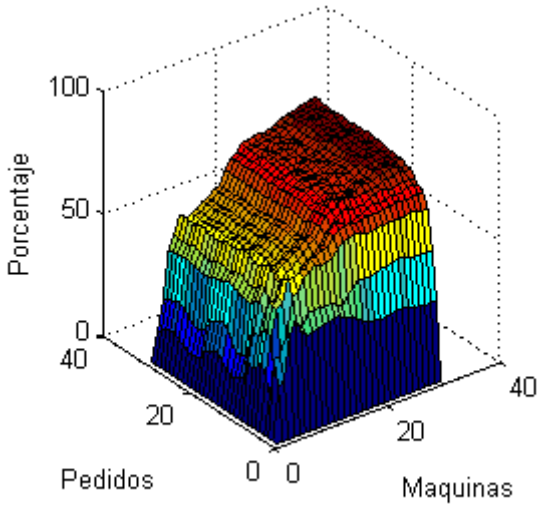

Figura 3d: LPT menos genetico.

Fig. 3: Graficas a, b, c, d. Diferencia entre técnicas tradicionales y AE. Tiempo de proceso tomado de la Tabla 4. 900 problemas Job shop: Open Shop.

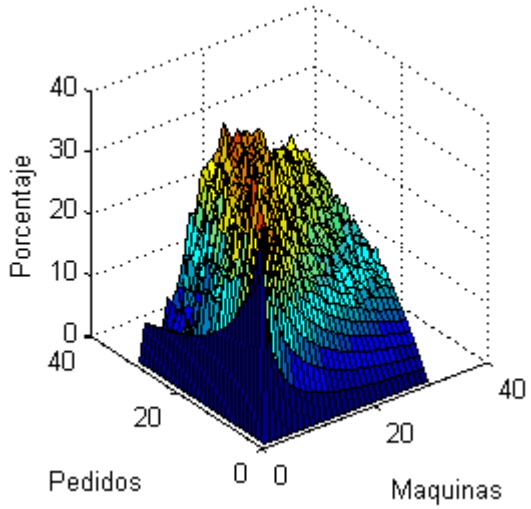

Figura 4a: Fifo menos genetico.

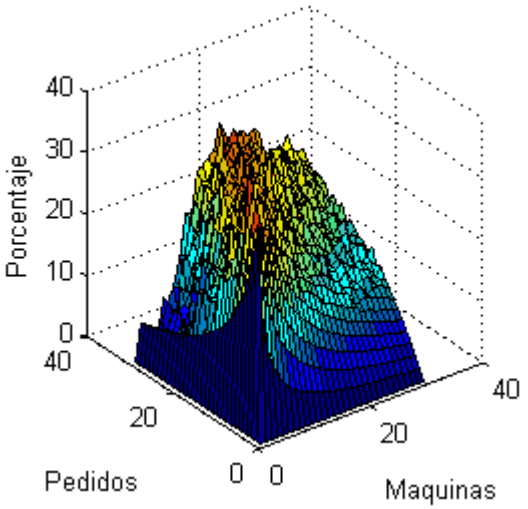

Figura 4b: Lifo menos genetico.

Fig.4 *(continuación)*

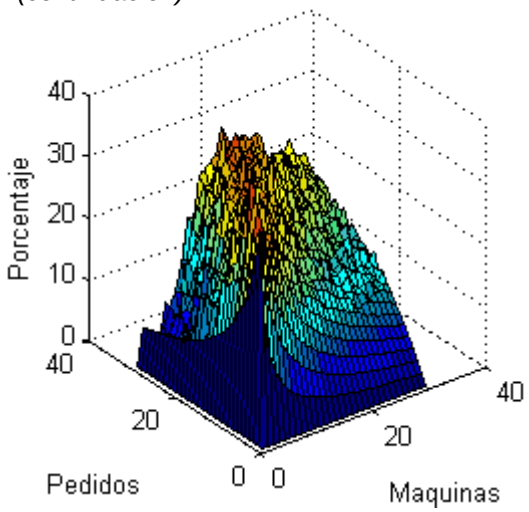

Figura 4c: SPT menos genetico.

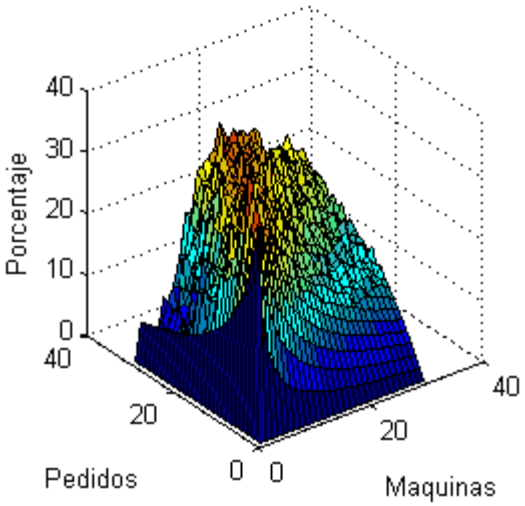

Figura 4d: LPT menos genetico.

Fig. 4: Graficas a, b, c, d. Diferencia entre las técnicas tradicionales y AE. Matriz traspuesta obtenida de la Tabla 3. 900 problemas Job Shop: Open Shop.

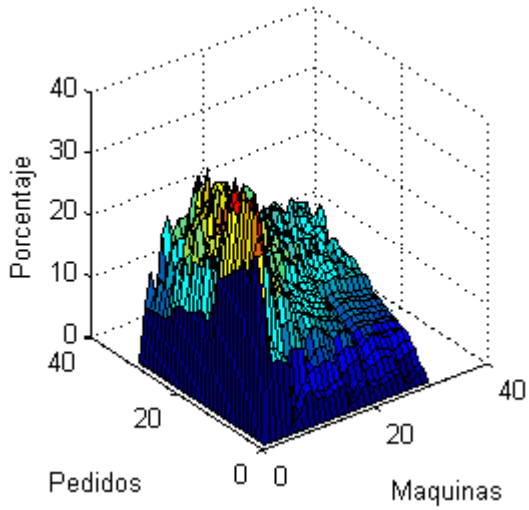

Figura 5a: Fifo menos genetico.

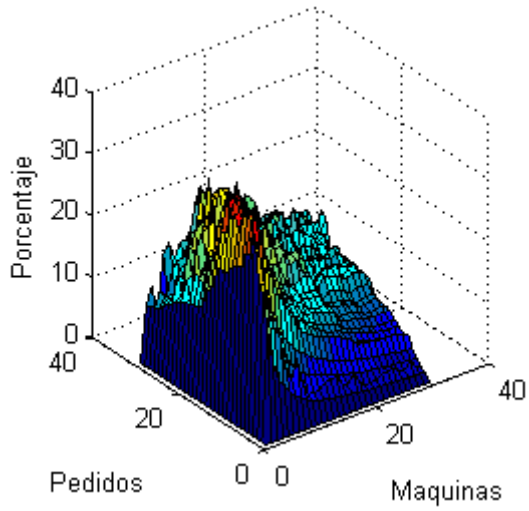

Figura 5b: Lifo menos genetico.

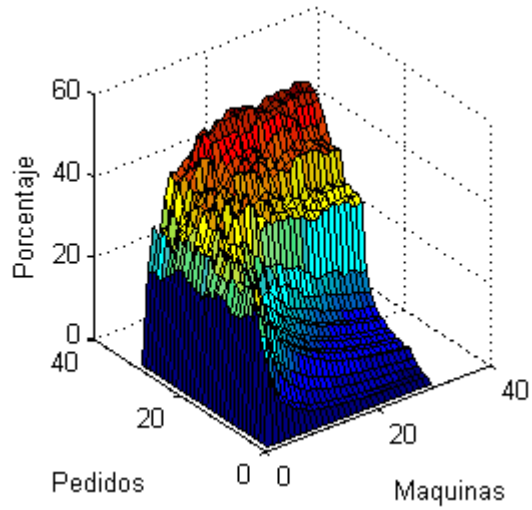

Figura 5c: SPT menos genetico.

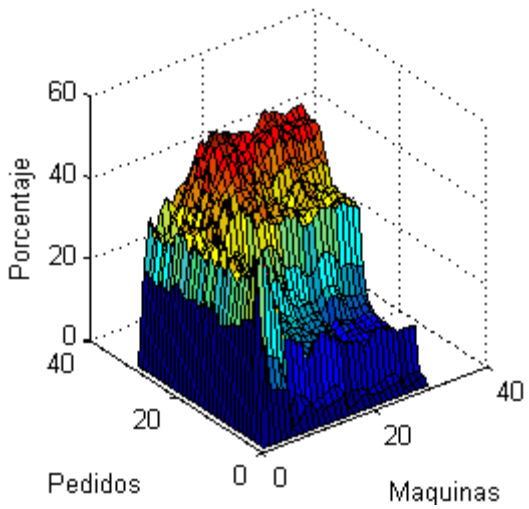

Figura 5d: LPT menos genetico.

Fig. 5: Graficas a, b, c, d. Diferencia entre las técnicas tradicionales y los AE. Matriz traspuesta obtenida de la Tabla 4. 900 problemas Job Shop: Open Shop.

Conjunto 3. Como se expone en la Figura 6, el rendimiento de los AE es aproximadamente un 20% mejor que las técnicas tradicionales. Este resultado se observó en dos casos: 1) alto número de máquinas y órdenes; 2) bajo número de máquinas y órdenes (ver los elementos de la diagonal). Sin embargo, en los casos restantes esta brecha fue menor y, en algunos casos, igual a cero. En general, para estos casos, las técnicas tradicionales y los AE tienen rendimientos similares.

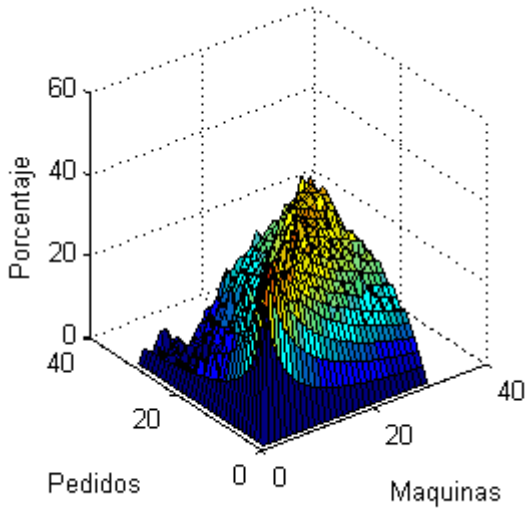

Figura 6a: Fifo menos genetico.

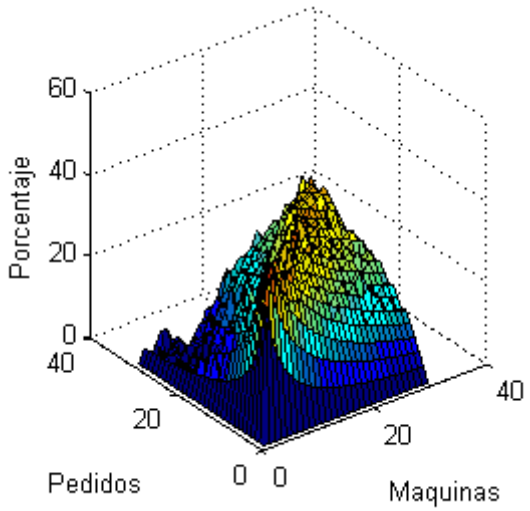

Figura 6b: Lifo menos genetico.

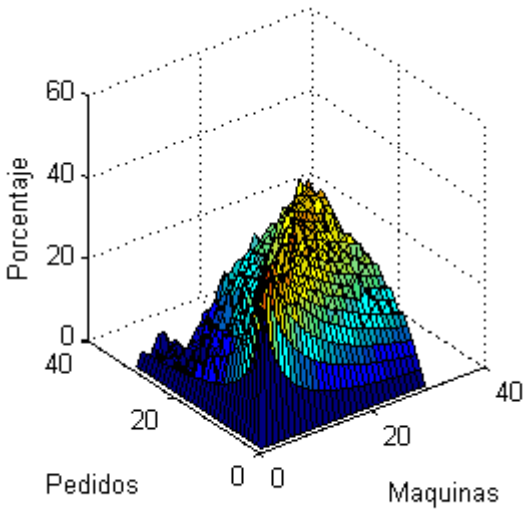

Figura 6c: SPT menos genetico.

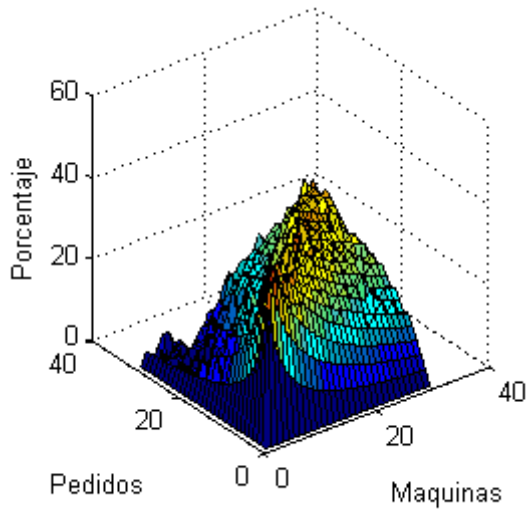

Figura 6d: LPT menos genetico.

Fig. 6: Graficas a, b, c, d. Diferencia entre las técnicas tradicionales y los AE. Conjunto de problemas 3. 900 problemas Job Shop: Open Shop.

Conjunto 4. Cuando fueron analizados un gran número de pedidos y máquinas (superiores a 20 pedidos y 20 máquinas), los AE fueron entre un 30% y un 50% mejores que las técnicas tradicionales de programación de la producción.

Sin embargo, en la medida en que disminuyó el número de trabajos y máquinas, esta brecha tiende a reducirse significativamente, como se ilustra en la figura 7.

DISCUSIÓN

Los resultados obtenidos revelan que la diferencia de rendimiento entre los AE y las técnicas tradicionales depende no sólo de los tiempos de procesamiento, sino también del número de trabajos y máquinas involucradas en el problema de programación. En general, para todos los conjuntos de problemas analizados, el *makespan* obtenido por los AE tiende a ser mejor que el reportado por las técnicas

tradicionales. Por lo tanto, existe una brecha de rendimiento que demuestra la superioridad de los AE sobre las técnicas tradicionales en casi todos los casos analizados. Puesto que los AE pueden probar un gran número de posibilidades de secuenciación considerando todas las máquinas juntas, esta técnica generalmente funciona mejor. Probar un gran número de posibilidades, permite reducir los tiempos muertos en cada máquina como se observó la Figura 3, donde el tiempo muerto de la máquina 3 fue cero.

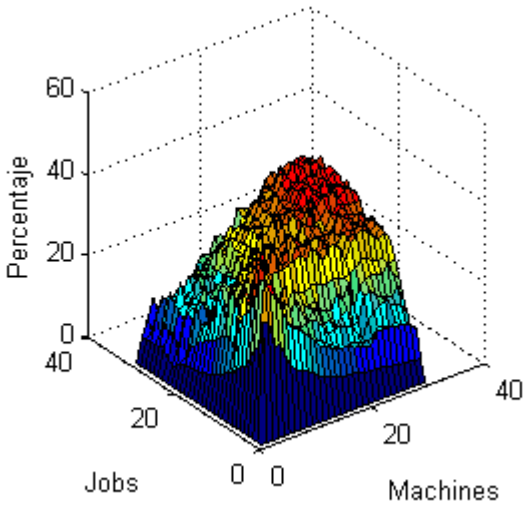

Figura 7a: Fifo menos genetico.

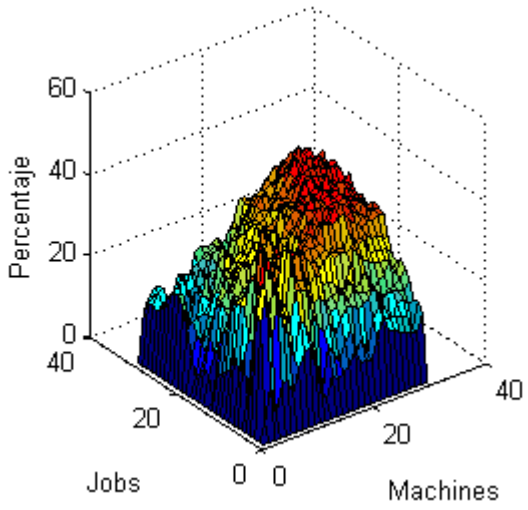

Figura 7b: Lifo menos genetico.

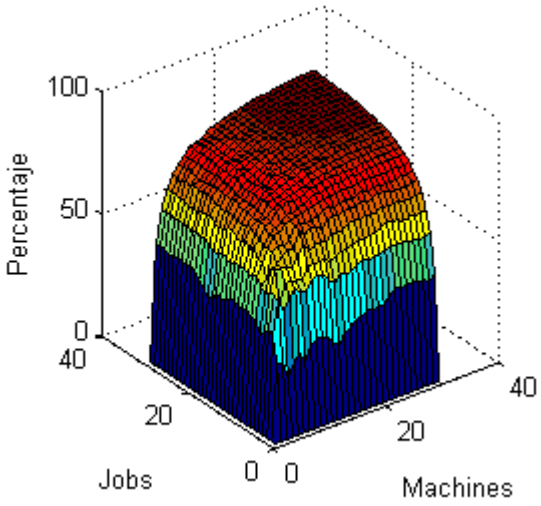

Figura 7c: SPT menos genetico.

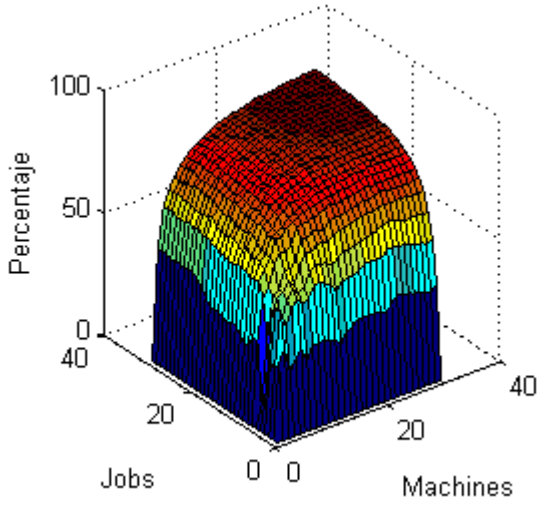

Figura 7d: LPT menos genetico.

Fig. 7: Graficas a, b, c, d. Diferencia entre las técnicas tradicionales y los AE. Conjunto 4 de problemas. 900 problemas Shop Jop: Open Shop.

En contraste, dado que las técnicas tradicionales usan un esquema de secuenciación por cada máquina independientemente, este aspecto las limita para obtener una mejor solución (Castrillón et al., 2009). Sin embargo, como se demuestra en los resultados obtenidos, en algunos casos particulares, la utilización de técnicas tradicionales puede ser tan eficiente como el uso de los AE, específicamente cuando el número de ordenes es pequeña (menor a 20 en este caso) y el tiempo de procesamiento es igual; o cuando el número de máquinas es alto (mayor a 20 en este caso) y el número de trabajos es bajo (menor a 20) con un tiempo de procesamiento similar. En cada uno de los cuatro conjuntos de datos analizados, se observa una similitud entre los resultados arrojados por ambas técnicas en los siguientes casos: a) primer conjunto de datos: cuando el número de pedidos es pequeño; b) segundo conjunto de datos: cuando el número de máquinas es alto pero el número de pedidos es pequeño; c) tercero y cuarto conjunto de datos: cuando el número de máquinas o de pedidos es pequeño. En general, se observa que, cuando hay menos variables independientes, la brecha entre las técnicas inteligentes (algoritmos evolutivos) y las técnicas tradicionales tiende a disminuir, dado que el número de posibilidades de secuenciación, depende de estas dos variables independientes.

CONCLUSIONES

De acuerdo con los resultados experimentales, en algunos casos la diferencia entre el *makespa*n de las técnicas tradicionales *vs.* los AE, fue superior al 50%, pero en otros casos fue inferior al 10% e, incluso, cercano a cero. Puesto que las técnicas inteligentes (como los AE) demandan grandes esfuerzos computacionales, este hallazgo sugiere su utilización sólo en problemas complejos de programación. En consecuencia, las técnicas tradicionales pueden utilizarse para un pequeño número de pedidos o máquinas (inferior a 20), especialmente en las empresas pequeñas o medianas que no pueden acceder a grandes recursos tecnológicos o cuando el número de máquinas o pedidos que deben procesar cada día no es muy elevado, sin afirmar por esto que el problema sea sencillo. De otro lado, se reafirma el uso del *makespan*, como una excelente variable de optimización, pues una reducción en el tiempo de proceso se refleja en un mejor desempeño del sistema.

Como futuras líneas de investigación, podrían evaluarse otros indicadores de desempeño tales como el costo, y el nivel de servicio (cumplimiento). De otro lado, el estudio de otras configuraciones de problemas relacionadas con casos particulares podrían reportar hallazgos interesantes; por ejemplo, se podrían adelantar estudios por sectores de la industria y tamaños de empresas con miras a identificar las configuraciones en las cuales las técnicas tradicionales *vs* las técnicas inteligentes podrían ser o no favorables. Por último, un análisis comparativo entre las diversas técnicas inteligentes reportadas en la literatura para problemas OSSP es recomendado para futuros trabajos en sectores particulares de la industria.

AGRADECIMIENTOS

Se agradece la colaboración a la Universidad Nacional de Colombia y en especial al Departamento de Ingeniería Industrial de la sede Manizales.

REFERENCIAS

Castrillón, O.D.; Giraldo, J.A. y W.A, Sarache, Solución de un problema Job shop por medio de un agente Inteligente, Ingeniería y Ciencia: 5(10), 75-92 (2009)

Castrillón, O.D.; Sarache W.A. y J.A. Giraldo, Aplicación de un Algoritmo Evolutivo en la Solución de Problemas Job Shop-Open Shop, Inf. Tecnol.: 22(1), 83-92 (2011)

Castrillón, O.D.; Sarache W.A. y J.A. Giraldo, Diseño de una híper heurística para la programación de la producción en ambientes job shop, Ingeniare. Rev. Chil. Ing.: 18(2), 203-214 (2010)

Cazacu, R., Comparative Study between the Improved Implementation of 3 classic Mutation Operators for Genetic Algorithms, Procedia Engineering: 181, 634 – 640 (2017)

Frutos, M. y F. Tohmé, Choice of a PISA selector in a hybrid algorithmic structure for the FJSSP, Decision Science Letters: 4(2), 247-260 (2015)

Gupta, D. y H. Singh, A String of Disjoint Job Blocks on Two Stage Open Shop Scheduling with Transportation Time, Control Theory and Informatics: 5(2), 42-47 (2015)

Karimi-Nasab, M.; Modarres, M. y S.M. Seyedhoseini, A self-adaptive PSO for joint lot sizing and job shop scheduling with compressible process times, Applied Soft Computing: 27, 137-147 (2015)

Koulamas, C. y G. Kyparisis, The three-machine proportionate open shop and mixed shop minimum makespan problems, European Journal of Operational Research: 243(1), 70-74 (2015)

Lei, D. y X. Guo, An effective neighborhood search for scheduling in dual-resource constrained interval job shop with environmental objective, International Journal of Production Economics: 159, 296-303 (2015)

Liaw, C. F., An improved branch-and-bound algorithm for the preemptive open shop total completion time scheduling problem, Journal of Industrial and Production Engineering: 30(5), 327-335 (2013)

López, J., y M. Arango, Algoritmo genético para reducir el makespan en un flow shop híbrido flexible con máquinas paralelas no relacionadas y tiempos de alistamiento dependientes de la secuencia, Ingeniería y tecnología: 11(1), 250-262 (2015)

Palacios, J.J.; M.A. González y C.R. Vela, González-Rodríguez, I., y Puente, J., Genetic tabu search for the fuzzy flexible job shop problema, Computers & Operations Research, 54, 74-89 (2015)

Panahi, H. y R., Tavakkoli-Moghaddam, Solving a multi-objective open shop scheduling problem by a novel hybrid ant colony optimization, Expert Systems with Applications: 38(3), 2817-2822 (2015)

Peng, B.; Z. Lü. y T.C. Cheng, A tabu search / path relinking algorithm to solve the job shop scheduling problem, Computers & Operations Research: 53, 154-164 (2015)

Pérez, R.; S., Jön y A. Hernández, Solución de un problema de secuenciamiento en configuración Job Shop flexible utilizando un Algoritmo de Estimación de Distribuciones, Revista Iberoamericana de Automática e Informática Industrial RIA: 12(1), 49-57 (2015)

Ruiz, S.; Castrillón O.D. y W.A. Sarache, Una Metodología Multiobjetivo para Optimizar un Ambiente Job Shop, Inf. Tecnol: 23(1), 35-46 (2012)

Russell, R.S. y B.W. Taylor, Operations Management: Creating Value Along the Supply Chain. 7th Ed., Wiley, New York: (2010)

Sahraeian, R. y M. Namakshenas, On the optimal modeling and evaluation of job shops with a total weighted tardiness objective: Constraint programming vs mixed integer programming, Applied Mathematical Modelling: 39(2), 955-964 (2015)

Salazar, E. y O. Sanchez, Estrategias más para minimización del makespan en la programación de una máquina con setup, Revista Ingeniería Industrial: 10(2), 17-29 (2011)

Salazar, E.H.; A. René y G. Sarzuri, Algoritmo genético mejorado para la minimización de la tardanza total en un flow shop flexible con tiempos de preparación dependientes de la secuencia, Rev. Chil. Ing.: 23 (1) 118-127 (2015)

Sangsawang, C.; K. Sethanan; T. Fujimoto y M. Gen, Metaheuristics optimization approaches for two-stage reentrant flexible flow shop with blocking constraint, Expert Systems with Applications: 42(5), 2395-2410 (2015)

Sarache, W.; O. Castrillón y G. Giraldo, Sistemas de producción: Modelamiento y gestión, 1ª Ed., Editorial Universidad Nacional: Bogotá (2011)

Shen, X.N. y X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Information Sciences: 298, 198-224 (2015)