Revista Industrial Data 19(2): 124-133 (2016)
doi: http://dx.doi.org/10.15381/idata.v19i2.12846

ISSN: 1560-9146 (Impreso) / ISSN: 1810-9993 (Electrónico)
Facultad de Ingeniería Industrial - UNMSM

Enrutamiento y secuenciación óptimos en un Flexible Job Shop multiobjetivo mediante algoritmos genéticos

RECIBIDO: 25/07/2016 ACEPTADO: 09/11/2016

GUILLERMO TEJADA MUÑOZ*

RESUMEN

El artículo propone, un algoritmo genético para solucionar óptimamente el problema de la programación de tareas en un sistema de producción Flexible Job Shop Scheduling (FJSS) multiobjetivo, actualmente de interés por muchos investigadores, porque es un problema de optimización combinatoria de complejidad NP-hard, y porque una solución óptima redunda en un aumento en la producción. Se divide el problema, en el subproblema de enrutamiento, en donde se asigna, a cada operación de los Jobs, una de las máquinas más óptima (desde un conjunto disponible) minimizando el Máximo Workload, y Total Workload, y el subproblema de secuenciación, en donde es encontrado el orden óptimo de ejecución de las operaciones (distribuidas en cada máquina) minimizando el Makespan. El algoritmo es codificado en lenguaje M de Matlab, su desempeño es puesto a prueba, solucionando complejos problemas, y los resultados se comparan con los obtenidos por otros investigadores.

Palabras clave: Flexible Job Shop Scheduling Problem, algoritmos genéticos, makespan, máximo workload, total workload

OPTIMUM ROUTING AND SEQUENCING IN A MULTIOBJECTIVE FLEXIBLE JOB SHOP USING GENETIC ALGORITHMS

ABSTRACT

The paper proposes a genetic algorithm to solve optimally the problem of scheduling in a multi-objective production system Flexible Job Shop (FJS), currently of interest for many researchers, because it is a combinatorial optimization problem of complexity NP-hard, and because an optimal solution results in an increase in production. the problem is divided, in the subproblem routing, where it is assigned to each operation of Jobs, one of the most optimum machines (from a set available) minimizing Maximum Workload, and Total Workload and subproblem sequencing, where it is found the optimal order of execution of operations (distributed on each machine) minimizing the Makespan. The algorithm is coded in Matlab M language, their performance is tested, solving complex problems, and the results are compared with those obtained by other researchers.

Keywords: Flexible Job Shop Scheduling Problem, genetic algorithms, makespan, maximum workload, total workload

1. INTRODUCCIÓN

Uno de los problemas más difíciles en la programación de tareas se encuentra en la manufactura tipo taller (Job Shop Scheduling Problem - JSSP), donde un conjunto de jobs (trabajos) deben ser procesados en un conjunto de máquinas, cada job implica la ejecución de una secuencia de operaciones, las cuales requieren cada una exactamente una máquina para su ejecución. Las máquinas están siempre disponibles y pueden procesar una operación a la vez sin interrupción, el problema consiste en secuenciar el orden de ejecución de las operaciones en las máquinas, teniendo a su vez como objetivo el de optimizar algún indicador típico de rendimiento, como por ejemplo, el Makespan, es decir, el tiempo necesario para completar todos los trabajos (Pezzella *et al.*, 2008).

El Flexible Job-shop Scheduling Problem (FJSSP) es una generalización del clásico JSSP, ya que a diferencia del Job Shop, en donde cada una de las operaciones, perteneciente a los Jobs, tienen máquinas fijas asignadas, en el FJSS las operaciones pueden ser ejecutadas en cualesquiera de un conjunto de máquinas disponibles. Entonces, solucionar el problema FJSSP es más difícil que el clásico JSSP. Es decir, no solo hay que encontrar la secuenciación de las operaciones sino también las rutas de trabajo, es decir, asignar a cada operación una de las máquinas, entre muchas disponibles, que la procesará o ejecutará (Pezzella *et al.*, 2008).

El problema de Flexible Job Shop Scheduling (FJSSP) es un problema de optimización combinatoria de los más difíciles de solucionar en el campo de la programación (scheduling) de tareas. Pertenece a la familia de NP-duro (No Determinístico de Tiempo Polinómico Difícil). La solución de un problema de programación de NP-duro con un único objetivo es una tarea difícil, la adición de más de un objetivo (multi-objetivo), obviamente, hace que este problema sea aún más difícil de resolver (Mekni & Chaâr, 2015).

Brandimarte (1993) propone un enfoque jerárquico para la resolución del FJSS. En un nivel de la jerarquía considera la solución del subproblema de asignar máquinas a las operaciones y la solución del subproblema de secuenciar las operaciones en las máquinas. Utiliza reglas de despacho y Búsqueda Tabú

* Master Ingeniería Electrónica, Docente principal en la FIE-UNMSM, posgraduando de doctorado en la FII-UNMSM. E-mail: gtejadam@unmsm.edu.pe

para solucionar ambos subproblemas. Considera la minimización de dos funciones objetivos: el Makespan (tiempo de terminación de todos los trabajos) y el total weighted tardiness (suma de productos de la tardanza de producción de cada trabajo por el peso de prioridad asignado a cada trabajo). El autor crea quince (15) problemas prototipo de sistema de producción tipo FJSS, con las cuales prueba su propuesta.

Kacem *et al.* (2002a) proponen un enfoque de Pareto. Aplican operadores de mutación que emulan tener Organismos Modificados Genéticamente (OMG) los cuales crean cromosomas que aceleran la convergencia a la solución final, un sistema multiobjetivo de lógica difuza ayuda a decidir si los nuevos individuos representan una interesante solución a las siguientes etapas del algoritmo. La función objetivo minimiza el Makespan, total Workload y el máximo Workload. Los autores son creadores de cinco (5) problemas prototipo de sistema de producción tipo FJSS, con los cuales prueban su propuesta.

Xia & Wu (2005) aplican un enfoque jerárquico, utilizan el algoritmo de optimización por enjambre de partículas (PSO, por sus siglas en inglés) para solucionar el problema de enrutamiento y el algoritmo de recocido simulado (SA, por sus siglas en inglés) para solucionar el problema de secuenciación. La función objetivo se define como la suma ponderada del Makespan, el total Workload y el máximo Workload. Para ilustrar la efectividad y desempeño del algoritmo, se solucionan el problema 8x8 (ocho jobs y ocho máquinas), 10x10 (diez jobs y diez máquinas) y el 15x10 (quince jobs y quince máquinas) de Kacem *et al.* (2002).

Zhang *et al.* (2009) utilizan un algoritmo de Optimización de Enjamble de Partículas (PSO, por sus siglas en inglés) para el solucionar subproblema de enrutamiento y el subproblema de secuenciación. Además, es utilizado un algoritmo de Búsqueda Tabú para búsqueda local de cada solución obtenida en la solución del subproblema de secuenciación. Minimiza la suma ponderada del Makespan; máximo Workload y total Workload. En este artículo son codificados dos vectores: A-cadena (para las máquinas) y B-cadena (para las operaciones). El algoritmo fue implementado en C++, en una computadora personal Pentium IV de 1.8 GHz. es probado con cuatro el problema 4x 5, problema 8x8, problema 10 x10, y problema 15x10 de Kacem *et al.* (2002).

Xing *et al.* (2009) proponen un modelo de 6 subsistemas. El Subsistema de asignación, utiliza un algoritmo denominado Conocimiento de Asignación de Máquinas a Operación (OAMK, por sus siglas en inglés) para asignar operaciones para cada máquina, optimizando el total Workload y el máximo Workload. El Subsistema de secuenciación, aplica un método aleatorio para secuenciar las operaciones en cada máquina y es mejorado con un algoritmo de optimización de Colonia de Hormigas (ACO, por sus siglas en inglés). El Subsistema de evaluación, evalúa la programación por la suma ponderada del Makespan, total Workload y máximo Workload, los coeficientes de ponderación son obtenidas empíricamente. El Subsistema de control, controla el flujo computacional y la viabilidad de cálculo de las soluciones. El Subsistema de salida, provee al usuario los resultados de optimización y el diagrama de Gannt de la solución óptima. El algoritmo ha sido codificado en Matlab, en una computadora Pentium IV, 2.4 Ghz y 512 MB RAM. El algoritmo se prueba con los problemas de Kacem*et al.* (2002). Los resultados experimentales fueron promediados sobre 10 corridas.

Xing *et al.* (2009b) proponen un método heuristico de búsqueda minimizando tres objetivos: el Makespan, total Workload y máximo Workload, los cuales son integrados en una sola función, ponderando cada criterio con un coeficiente de acuerdo al grado de importancia asignado. Sus resultados demuestran que al utilizar diferentes coeficientes de ponderación para los tres objetivos, se supera a otros trabajos con los cuales se compara. El algoritmo fue implementado en Matlab en una computadora Workstation Dell Precision 650 con Pentium IV de 2.4 Ghz y 1 GHz de RAM. Los problemas prototipo de sistema de producción tipo FJSS, son tomados de Brandimarte (1993) y Kacem *et al.* (2002). Los resultados experimentales fueron promediados sobre 20 corridas.

Li *et al.* (2010), proponen un Algoritmo Híbrido de Búsqueda Tabú (HTSA –por sus siglas en inglés). El algoritmo de Búsqueda Tabú (TS) se utiliza para producir soluciones vecinas en el módulo de asignación de máquinas y un algoritmo de Búsqueda Variable Vecina (VNS, por sus siglas en inglés) realiza la búsqueda local en el módulo de secuenciación de operaciones. Se aplican nuevas reglas de búsqueda de vecinos en ambos módulos. Se minimiza el Makespan, total Workload y máximo Workload. El algoritmo es implementado en C++ en una Pentium IV de 1.7 GHz con 512 MB. El mejor resultado promedio de 20 independientes corridas fueron registrados para comparación. El algoritmo propuesto se prueba con los problemas prototipo de sistema de producción tipo FJSS de Kacem *et al.* (2002) y Brandimarte (1993).

Genova *et al.* (2015) realizan una revisión de la literatura sobre las soluciones para el FJSSP multiobjetivo, se revisan trabajos con soluciones en donde se aplican modelos aproximados matemáticos y trabajos con soluciones heurísticas y metaheurísticas, algunos de los investigadores de ese estudio también figuran en los párrafos anteriores. De la investigación de Genova, se puede notar a investigadores persiguiendo la minimización de otros criterios distintos al Makespan o Workload, estos criterios son por ejemplo: El retardo medio de los trabajos (*mean job tardiness*), la suma de los tiempos de ejecución (*flow time*), la suma promedio de los tiempos de ejecución (*mean flow time*), tiempo promedio de las maquinas sin utilizar (*mean machine idle time*), entre otros.

Considerando, los antecedentes, el presente trabajo describe una propuesta de solución al problema de FJSS utilizando algoritmos genéticos. Se encuentra el enrutamiento óptimo y la secuenciación óptima de las operaciones de cada trabajo optimizando los criterios de: total Workload (W_T, suma de carga de trabajo de todas las máquinas), máximo Workload (W_M, máxima carga de trabajo entre todas las máquinas) y Makespan, (C_M, tiempo de terminación de todos los trabajos). La propuesta es codificada en lenguaje m de Matlab y es probada con los problemas que simulan un sistemas de fabricación tipo FJSS planteados por kacem, los resultados son también presentados en Diagramas de Gannt.

2. METODOLOGÍA

Para solucionar el problema se optó por un enfoque jerárquico, tal como ha sido realizado por otros investigadores, el primero de los cuales es Brandimarte en 1993. Es decir, se divide el problema en dos subproblemas, el primero consiste en solucionar el subproblema de asignar máquinas a las operaciones (enrutamiento) y en el segundo consiste en secuenciar apropiadamente las operaciones en cada una de las máquinas (secuenciamiento).

En el subproblema de enrutamiento, se busca que las máquinas seleccionadas tengan cargas de trabajo optimizadas, para ello se minimiza el máximo Workload (W_M) y el total Workload (W_T) de las máquinas. Posteriormente, con las máquinas seleccionadas, se encuentra la secuencia óptima de ejecución de las operaciones en cada máquina, para ello se minimiza el Makespan (C_M), es decir, el tiempo de terminación de todas las operaciones de los jobs (trabajos).

De la revisión de la literatura se puede destacar que para solucionar ambos subproblemas los investigadores han optado por soluciones metaheuristicas mixtas. En esta propuesta, la solución para ambos subproblemas se ha realizado con algoritmos genéticos. Al igual, como otros investigadores lo han hecho, el desempeño del algoritmo se pone a prueba solucionando los problemas creados por kacem *et al.* (2002) y kacem *et al.* (2002a).

En la Figura 1 se muestra el diagrama de flujo del algoritmo genético que soluciona el problema de enrutamiento. Las etapas del algoritmo son: Creación de la población, Evaluación de poblción (Función Workload), Ordenar a la población, Selección, Cruce y Mutación.

La población es un arreglo constituido por vectores filas (cromosomas, miembros o individuos), en donde cada columna (genes) representa secuencialmente a una operación, cuyo contenido es una máquina asignada aleatoriamente a una operación. Las filas son de tamaño igual al número de operaciones que tiene el problema.

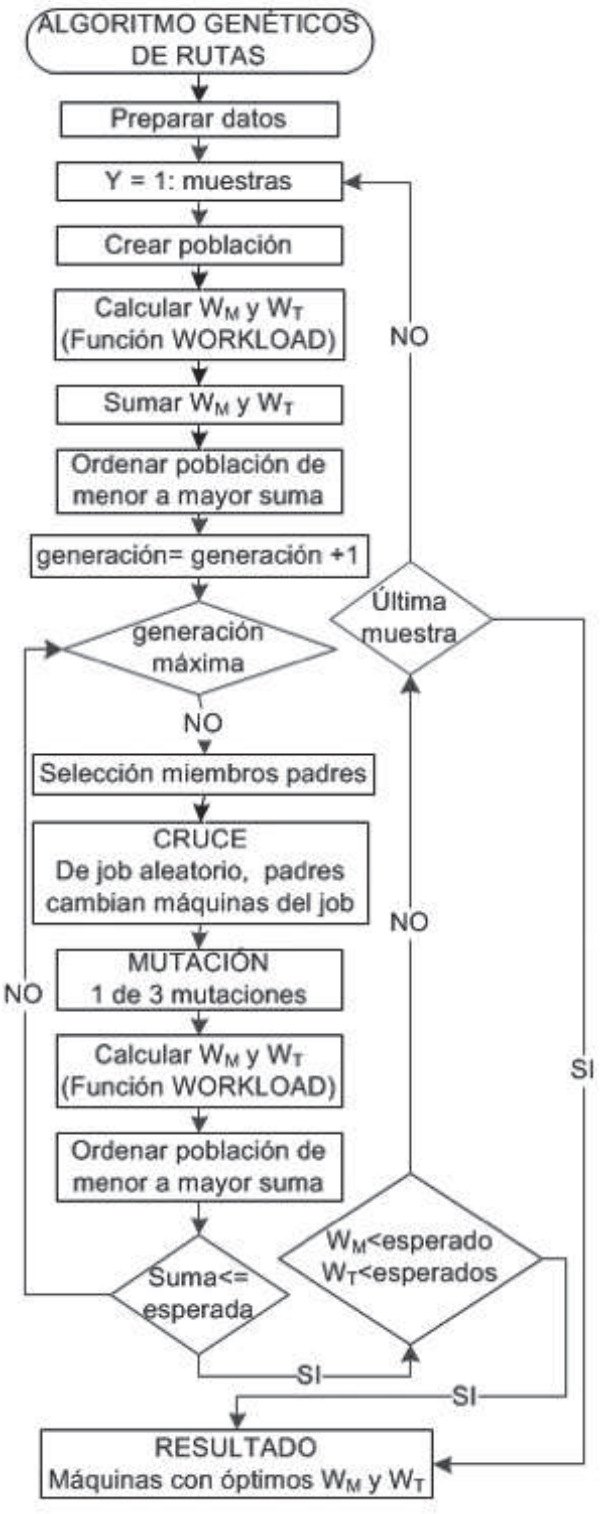
En la Figura 2 se muestra uno de los cromosomas (filas), en cada columna existe una máquina asignada a la operación O_i,j, donde i, representa el job i y j la operación j de ese job. Para el ejemplo mostrado, cada gen puede ser cualquiera de un conjunto de cinco máquinas y cada máquina se distingue de otras porque ejecuta a la operación correspondiente con un tiempo Ti o T´i. Es así, como los arreglos de la población son tridimensionales..

La función objetivo es, entonces, la suma de WT y WM (WT + WM), los sumandos no se ponderan como si se realiza en los enfoques de Xia & Wu (2005), Zhang *et al.* (2009), Xing *et al.* (2009) y Ziaee (2014), en estos casos, se considera una suma como: *, donde los coeficientes de ponderación son valores fijos obtenidos empíricamente, atribuyendo mayor peso al criterio que se asume de mayor importancia.

La etapa de Selección, es una modificación de selección por ruleta, de la misma manera como se realiza para el caso de algoritmos genéticos de permutación mostrada en Haupt & Haupt, 2004. A los padres que se cruzaran se les asigna un número de “cupones” con los cuales después de un “sorteo” se elaboran listas de padres. De ambas listas son tomadas secuencialmente dos a dos individuos para el cruce.

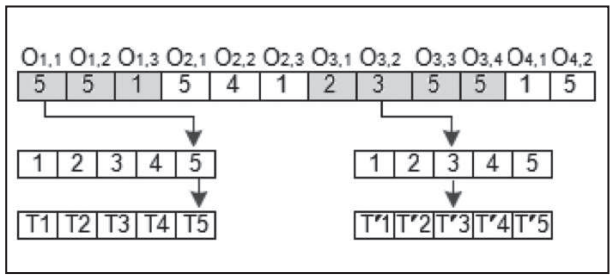
Para la etapa del Cruce, se ha optado por el cruce de dos puntos, pero a diferencia de tomar dos puntos aleatorios, en nuestra propuesta se

Figura 1. Algoritmo genético de rutas.



Fuente: Elaboración propia laminado.

Figura 2. Cromosoma con doce máquinas (genes) arbitrarias.



Fuente: Elaboración propia.

encuentra aleatoriamente el punto de inicio de un job y como segundo punto el punto donde finalizan las operaciones del job, luego son intercambiadas entre los cromosomas, las máquinas de ese job, generándose dos hijos por cada cruce.

Para la etapa de Mutación, se han creado tres tipos, las cuales se ejecutan generando un número aleatorio entre 0 y 1, que representa la probabilidad para ejecutar una de ellas. La mutación que tiene mayor probabilidad de ejecutarse consiste en remplazar una de las máquinas de una operación escogida aleatoriamente por la máquina más rápida que existe para esa operación. La otra mutación consiste en remplazarla por una máquina al azar disponible para esa operación y la otra mutación consiste en reemplazarla por la máquina menos frecuente entre todas las máquinas del cromosoma.

En todos los casos de la literatura, en donde se aplican diferentes metaheurísticas, se obtienen casi los mismos resultados globales de W_T, W_M y C_M. El algoritmo propuesto pretende encontrar mejores o iguales resultados que los globales conocidos, para ello, uno de los parámetros de finalización es cuando la suma es menor de lo que suman los globales conocidos W_T, W_M (Suma esperada, ver figura 1). Sin embargo, también se verifica, si cada uno de los sumandos son menores que los globales conocidos, solo en ese caso el algoritmo se detiene, en caso contrario, el algoritmo se reinicia generando una nueva población de máquinas. El algoritmo finaliza también cuando el número de generaciones o el número de reiteraciones llegan a su límite. Bajo esta estructura, el algoritmo genético propuesto es modificado frente a los convencionales.

En la Figura 3 se muestra el diagrama de flujo del algoritmo genético de secuencias. Este algoritmo se ejecuta inmediatamente después del algoritmo de rutas, quien designó a cada operación una máquina, ajustando en cada máquina cargas óptimas con un mínimo W_T y W_M.

Las etapas del algoritmo son: Creación de la población, Evaluación de la población (Función Makespan), Ordenar a la población, Selección, Cruce y Mutación.

La población es un arreglo constituido por cromosomas en donde cada columna (genes) representa una operación. En la Figura 4 se muestra un cromosoma de secuencias arbitraria de doce (12) operaciones, cada operación de la secuencia está relacionada con un job (Ji), con una máquina (Mi) que la ejecuta y con un tiempo de ejecución (Ti). Las operaciones del job1 (J1) son tres (1,2,3), las del job2 (J2) son también tres (4,5,6), las del job3 (J3) son cuatro (7,8,9,10) y las del job4 (J4) son dos (11,12) operaciones, nótese que el orden de precedencia de las operaciones de cada job se deben conservar dentro del cromosoma.

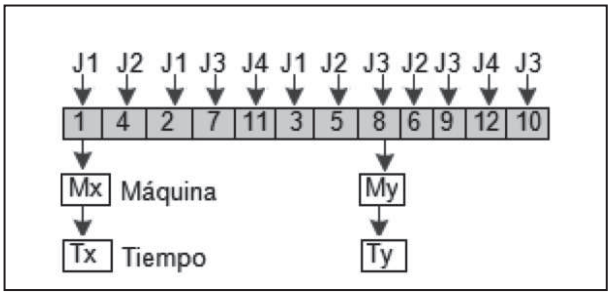
Todos los cromosomas de la población que se generan tendrán la característica del cromosoma de la Figura 4. Para ello, se generan aleatoriamente números permutados desde 1 hasta el número de operaciones totales (los números no se repiten), luego se detectarán el conjunto de operaciones de cada job y se corregirán su secuencia de precedencia, creando así cromosomas y población de secuencias válidas. En otras propuestas de la literatura no distinguen la secuencia de las operaciones, en este trabajo se prefiere considerarlas porque nos facilita posteriormente los cálculos para evaluar a cada cromosoma mediante la función Makespan.

Luego de creada la población, cada uno de sus miembros (cromosomas) son evaluados mediante la función Makespan, que calcula el tiempo de finalización máximo de las máquinas o trabajos de cada cromosoma. Cada operación (gen) del cromosoma está relacionada con una máquina quien la procesa, un job a quien pertenece y un tiempo de ejecución. Por lo que cada cromosoma de la población genera información para generar tres arreglos: MQ (maquinas), TR (jobs) y TI (tiempos).

Con estos arreglos los cálculos se realizan examinando cada operación de izquierda a derecha del cromosoma. El proceso brevemente se muestra en la Figura 5, un vector TMK se construye con el cálculo del makespan de cada máquina, por ejemplo, para la operación 4, se observa en MQ que la máquina 1 (no se utilizó antes) y en TR se observa que operación es la primera del job 2 (no aparece antes). Entonces, el Makespan parcial de la máquina 1 tiene como punto de partida un tiempo igual a cero y como tiempo final el tiempo de procesamiento de la operación en esa máquina (TMK=0+TI=2). Cuando llegamos a la

Figura 3. Algoritmo genético de secuencias.

Figura 4. Cromosoma secuencia de operaciones.



Fuente: Elaboración propia laminado

operación 12, se observa en MQ que la máquina 2 ya fue utilizada antes y en TR se observa que la operación pertenece a la segunda operación del job 4. Entonces el Makespan parcial de la máquina 2 tiene como punto de partida el mayor tiempo entre lo contabilizado para la máquina 2 (TMK=7) y lo contabilizado para job 4 (TMK=3) por lo que el Makespan parcial para la máquina 2 es 8 (mayor(3,7) + TI). El proceso continua, hasta haberse calculado el Makespan para la operación 6, que es la última. El Makespan por tanto es el mayor elemento del vector TMK (11). Basado en el procedimiento descrito, se construye el algoritmo para el Makespan.

Figura 5. Datos para calcular Makespan.

OPERACIONES =	4	5	1	7	2	11	8	3	9	12	10	6
MQ =	1	5	4	3	2	1	2	1	4	2	4	3
TR =	2	2	1	3	1	4	3	1	3	4	3	2
TI =	2	5	1	6	4	1	1	4	2	1	1	4
TMK =	2	7	1	6	5	3	7	9	9	8	10	11

Fuente: Elaboración propia laminado.

La etapa de Selección se realiza del mismo modo como en el caso del algoritmo genético de rutas. Para la etapa de Cruce se escoge aleatoriamente un gen de los padres y estos son intercambiados, sin embargo como este gen introducido en el cromosoma colisiona por ser igual a otro gen dentro del cromosoma, entonces el gen que colisiona se intercambia con el del otro padre. El proceso se repite hasta que en los cromosomas no existan genes repetidos, de esta manera se generan dos hijos por cada par de padres.

En la etapa de Mutación se intercambian arbitrariamente, en el mismo cromosoma, dos genes. En la presente propuesta, esto solo se realiza si los genes pertenecen a diferentes jobs para mantener precedencia.

El algoritmo finaliza al obtener un Makespan menor o igual al óptimo global de la literatura o al agotarse el límite de iteraciones sin mejoras o al agotarse el límite de generaciones.

Finalmente, para facilitar la verificación de los resultados, se utiliza la información contenida en los vectores MQ, TR, TI y TMK para construir automáticamente Diagramas de Gannt. Los resultados son de los cinco problemas planteados por kacem que simulan cinco procesos complejos de FJSS.

El algoritmo ha sido codificado totalmente en el lenguaje M de Matlab, en su versión estudiantil R214A, ha corrido en una PC con procesador i5 de 2.9 GHz. y 4 G.B de RAM.

3. RESULTADOS

A acontinuación se describen los resultados experimentales con los cinco problemas planteados por Kacem *et al.* (2002) y Kacem *et al.* (2002a) que miden el desempeño del algoritmo. Por motivos de espacio, solo se muestran, en las figuras 6, 7 y 8 respectivamente, los datos de tres problemas de Kacem.

Todos los problemas de Kacem son de total flexibilidad a excepción del problema de 8 jobs con 8 máquinas que es de flexibilidad parcial, es decir, no todas las operaciones pueden ejecutarse en cualesquiera de las ocho máquinas, en ese caso, se indica con " - " esa imposibilidad. Computacionalmente, para estos casos, en el lugar de la máquina inexistente se registra un número muy grande, de tal modo que si es seleccionada, al evaluar el cromosoma el resultado es tan grande que en la clasificación de la población se va descartando.

Los resultados, para cada uno de los problemas se muestran en la Tabla 1, para cada caso se han realizado 20 corridas registrado las respuestas W_T (Total Workload), W_M (Máximo Workload) y C_M (Makespan) correspondientes. Se han registrado todas las soluciones, es así que para el problema de 10x10 las soluciones ascienden a tres (S1, S2 y S3), se computa el tiempo promedio para 20 respuestas.

Figura 6. Problema 4x5 con 12 operaciones, total flexibilidad.

JOB	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	2	5	4	1	2
	$O_{1,2}$	5	4	5	7	5
	$O_{1,3}$	4	5	5	4	5
J_2	$O_{2,1}$	2	5	4	7	8
	$O_{2,2}$	5	6	9	8	5
	$O_{2,3}$	4	5	4	54	5
J_3	$O_{3,1}$	9	8	6	7	9
	$O_{3,2}$	6	1	2	5	4
	$O_{3,3}$	2	5	4	2	4
	$O_{3,4}$	4	5	2	1	5
J_4	$O_{4,1}$	1	5	2	4	12
	$O_{4,2}$	5	1	2	1	2

Fuente: Kacem *et al.* (2012).

Figura 7. Problema 8x8 con 27 operaciones, flexibilidad parcial.

JOB	O_{ij}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	$O_{1,1}$	5	3	5	3	3	-	10	9
	$O_{1,2}$	10	-	5	8	3	9	9	6
	$O_{1,3}$	-	10	-	5	6	2	4	5
J_2	$O_{2,1}$	5	7	3	9	8	-	9	-
	$O_{2,2}$	-	8	5	2	6	7	10	9
	$O_{2,3}$	-	10	-	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	-	-
J_3	$O_{3,1}$	10	-	-	7	6	5	2	4
	$O_{3,2}$	-	10	6	4	8	9	10	-
	$O_{3,3}$	1	4	5	6	-	10	-	7
J_4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
J_5	$O_{5,1}$	3	6	7	8	9	-	10	-
	$O_{5,2}$	10	-	7	4	9	8	6	-
	$O_{5,3}$	-	9	8	7	4	2	7	-
	$O_{5,4}$	11	9	-	6	7	5	3	6
J_6	$O_{6,1}$	6	7	1	4	6	9	-	10
	$O_{6,2}$	11	-	9	9	9	7	6	4
	$O_{6,3}$	10	5	9	10	11	-	10	-
J_7	$O_{7,1}$	5	4	2	6	7	-	10	-
	$O_{7,2}$	-	9	-	9	11	9	10	5
	$O_{7,3}$	-	8	9	3	8	6	-	10
J_8	$O_{8,1}$	2	8	5	9	-	4	-	10
	$O_{8,2}$	7	4	7	8	9	-	10	-
	$O_{8,3}$	9	9	-	8	5	6	7	1
	$O_{8,4}$	9	-	3	7	1	5	8	-

Fuente: Kacem *et al.* (2012a)

Tabla 1. Resultados del algoritmo y tiempos de ejecución

Problema jobs x máquinas		(W_T)	(W_M)	(C_M)	Tiempo promedio (20 corridas)
4x5 (Fig. 11)	S1	32	10	11	0.07 seg.
	S2	32	10	12	
8x8 (Fig. 12)	S1	75	12	15	1.15 seg.
	S2	76	12	15	
10x7 (Fig. 13)	S1	61	11	11	1.57 seg.
	S2	61	11	12	
10x10 (Fig. 14)	S1	42	6	7	1.77 seg.
	S2	42	6	8	
	S3	42	6	9	
15x10 (Fig. 15)	S1	91	11	12	15.80 seg.
	S2	91	11	13	

S1= Solución 1 S2= Solución 2 S3= Soución 3

Fuente: Elaboración propia.

Figura 8. Problema 15x10 con 56 operaciones, flexibilidad total.

JOB	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	4
	$O_{1,2}$	1	1	3	4	8	10	4	11	4	3
	$O_{1,3}$	2	5	1	5	6	9	5	10	3	2
	$O_{1,4}$	10	4	5	9	8	4	15	8	4	4
J_2	$O_{2,1}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,2}$	6	11	2	7	5	3	5	14	9	2
	$O_{2,3}$	8	5	8	9	4	3	5	3	8	1
	$O_{2,4}$	9	3	6	1	2	6	4	1	7	2
J_3	$O_{3,1}$	7	1	8	5	4	9	1	2	3	4
	$O_{3,2}$	5	10	6	4	9	5	1	7	1	6
	$O_{3,3}$	4	2	3	8	7	4	6	9	8	4
	$O_{3,4}$	7	3	12	1	6	5	8	3	5	2
J_4	$O_{4,1}$	6	2	5	4	1	2	3	6	5	4
	$O_{4,2}$	8	5	7	4	1	2	36	5	8	5
	$O_{4,3}$	9	6	2	4	5	1	3	6	5	2
	$O_{4,4}$	11	4	5	6	2	7	5	4	2	1
J_5	$O_{5,1}$	6	9	2	3	5	8	7	4	1	2
	$O_{5,2}$	5	4	6	3	5	2	28	7	4	5
	$O_{5,3}$	6	2	4	3	6	5	2	4	7	9
	$O_{5,4}$	6	5	4	2	3	2	5	4	7	5
J_6	$O_{6,1}$	4	1	3	2	6	9	8	5	4	2
	$O_{6,2}$	1	3	6	5	4	7	5	4	6	5
J_7	$O_{7,1}$	1	4	2	5	3	6	9	8	5	4
	$O_{7,2}$	2	1	4	5	2	3	5	4	2	5
J_8	$O_{8,1}$	2	3	6	2	5	4	1	5	8	7
	$O_{8,2}$	4	5	6	2	3	5	4	1	2	5
	$O_{8,3}$	3	5	4	2	5	49	8	5	4	5
	$O_{8,4}$	1	2	36	5	2	3	6	4	11	2
J_9	$O_{9,1}$	6	3	2	22	44	11	10	23	5	1
	$O_{9,2}$	2	3	2	12	15	10	12	14	18	16
	$O_{9,3}$	20	17	12	5	9	6	4	7	5	6
	$O_{9,4}$	9	8	7	4	5	8	7	4	56	2
J_{10}	$O_{10,1}$	5	8	7	4	56	3	2	5	4	1
	$O_{10,2}$	2	5	6	9	8	5	4	2	5	4
	$O_{10,3}$	6	3	2	5	4	7	4	5	2	1
	$O_{10,4}$	3	2	5	6	5	8	7	4	5	2
J_{11}	$O_{11,1}$	1	2	3	6	5	2	1	4	2	1
	$O_{11,2}$	2	3	6	3	2	1	4	10	12	1
	$O_{11,3}$	3	6	2	5	8	4	6	3	2	5
	$O_{11,4}$	4	1	45	6	2	4	1	25	2	4
J_{12}	$O_{12,1}$	9	8	5	6	3	6	5	2	4	2
	$O_{12,2}$	5	8	9	5	4	75	63	6	5	21
	$O_{12,3}$	12	5	4	6	3	2	5	4	2	5
	$O_{12,4}$	8	7	9	5	6	3	2	5	8	4
J_{13}	$O_{13,1}$	4	2	5	6	8	5	6	4	6	2
	$O_{13,2}$	3	5	4	7	5	8	6	6	3	2
	$O_{13,3}$	5	4	5	8	5	4	6	5	4	2
	$O_{13,4}$	3	2	5	6	5	4	8	5	6	4
J_{14}	$O_{14,1}$	2	3	5	4	6	5	4	85	4	5
	$O_{14,2}$	6	2	4	5	8	6	5	4	2	6
	$O_{14,3}$	3	25	4	8	5	6	3	2	5	4
	$O_{14,4}$	8	5	6	4	2	3	6	8	5	4
J_{15}	$O_{15,1}$	2	5	6	8	5	6	3	2	5	4
	$O_{15,2}$	5	6	2	5	4	2	5	3	2	5
	$O_{15,3}$	4	5	2	3	5	2	8	4	7	5
	$O_{15,4}$	6	2	11	14	2	3	6	5	4	8

Fuente: Kacem *et al.* (2012).

En la Tabla 2, se muesra la comparación de los resultados con los obtenidos por otros investigadores, quienes utilizan diferentes técnicas (vease introducción) a la que se propone en este trabajo, también reportan más de una solución para cada caso. La mejor respuesta es determinada finalmente por el Makespan (CM), ya que este criterio determina el tiempo necesario con el cual se completan todos los trabajos. Los resultados de la presente propuesta corresponde a la línea F.

La solución obtenida (CM=11) para el problema de cuatro jobs y cinco máquinas (4x5) supera a Kacem y se iguala a la de los otros investigadores. En el caso del problema 8x8 la solución (CM=15) no supera por una unidad de tiempo a las soluciones de tres investigadores, en los otros casos igula a las soluciones. En el caso del problema 10x10 la solución (CM=11) iguala a las demás y finalmente en el caso del problema 15x10 la solución (CM=12) supera a Kacem, iguala con Xia & Wu, 2005, pero no supera en una unidad de tiempo a los otros investigadores.

En la Tabla 3, se muestra los tiempos de ejecución del algoritmo propuesto con aquellos, quienes han reportado tiempos de ejcución en sus propuestas. Como se puede observar en todos los casos la propueta de este trabajo supera a las demás.

Tabla 2. Comparación de resultados con las de otros investigadores

		PROBLEMA (JOBS X MÁQUINAS														
		4x5			8x8			10x7			10x10			15x10		
		WT	WM	CM	WT	WM	CM	WT	WM	CM	WT	WM	CM	WT	WM	CM
A	S1	34	10	16	79	13	15	-	-	-	45	5	7	95	11	23
	S2	-	-	-	75	13	16	-	-	-	-	-	-	91	11	24
B	S1	-	-	-	75	12	15	-	-	-	44	6	7	91	11	12
	S2	-	-	-	73	13	16	-	-	-	-	-	-	-	-	-
C	S1	32	10	11	77	12	14	-	-	-	43	6	7	93	11	11
	S2	-	-	-	75	12	15	-	-	-	-	-	-	-	-	-
D	S1	32	8	12	77	12	14	61	11	11	42	6	7	91	11	11
	S2	-	-	-	76	12	15	62	10	11	42	5	8	93	10	11
E	S1	32	10	11	77	12	14	61	11	11	43	5	7	91	11	11
	S2	32	8	12	75	12	15	62	10	11	42	6	7	93	10	11
	S3	-	-	-	-	-	-	-	-	-	42	5	8	-	-	-
F	S1	32	10	11	75	12	15	61	11	11	42	6	7	91	11	12
	S2	32	10	12	76	12	15	61	11	12	42	6	8	91	11	13
	S3	-	-	-	-	-	-	-	-	-	42	6	9	-	-	-

S1= Solución 1 S2= Solución 2 S3= Solución 3
A = Kacem et al (2002) B = Xia & Wu (2005)
C= Zhang,et al. (2009) D=Xing et al. (2009)
E= Li et al. (2010) F=Propuesta (2016)

Fuente: Elaboración propia.

Tabla 3. Comparación de resultados con las de otros investigadores

	PROBLEMA (JOBS X MÁQUINAS)				
	4x5	8x8	10x7	10x10	15x10
A	2.58 s.	39.37s.	109.99 s.	39.74 s.	865.18 s
B	0.15 s.	3.08 s.	2.58 s.	3.12s.	25.13 s.
C	0.07 s.	1.15s.	1.57s.	1.77s.	15.80 s.

A = Xing et al. (2009), promedio 10 corridas
B = Li et al.(2010), promedio 20 corridas
C = Propuesta, promedio 20 corridas

Fuente: Elaboración propia.

A continuación, se muestran algunos de los diagramas de Gannt de los resultados. Al solucionar el caso del problema de 4 jobs y 5 máquinas y 12 operaciones, una de las mejores secuencias de operaciones obtenidas es: 1- 4 - 1 – 5 – 2 – 8 – 6 – 11 – 9 – 10 – 12 – 3. Con WT = 32, WM =10, CM =11.

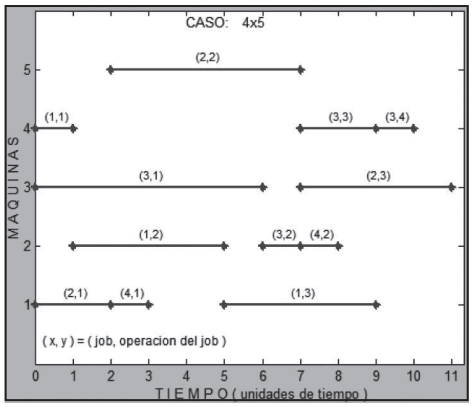
Cada secuencia con su equivalente en operaciones se indica a continuación: 1(O1,1), 2(O1,2), 3(O1,3), 4(O2,1), 5(O2,2), 6(O2,3), 7(O3,1), 8(O3,2), 9(O3,3), 10(O3,4), 11(O4,1), 12(O4,2). El lector, puede encontrar esta relación directamente de la columna Oij de la Figura 7.

El Diagrama de Gannt de la Figura 9 (generado por el programa), muestra en la ordenada las máquinas designadas y en cada segemento de recta los tiempos que ocupan cada operación y al job a que pertenecen, los que se muestran entre paréntesis. Se pueden verificar objetivamente los resultados de WT, WM y CM de la secuencia de operaciones encontrada.

En el caso del problema de 8 jobs, 8 máquinas con 27 operaciones, una de las mejores secuencias de operaciones obtenidas es: 1-11-8-18-4-14-24-25-19-2-12-21-5-20-6-13-22-9-15-10-7-16-26-27-23-3-17, con WT = 76, WM =12 y CM =15. El Diagrama de Gannt de la Figura 10, verifica los resultados.

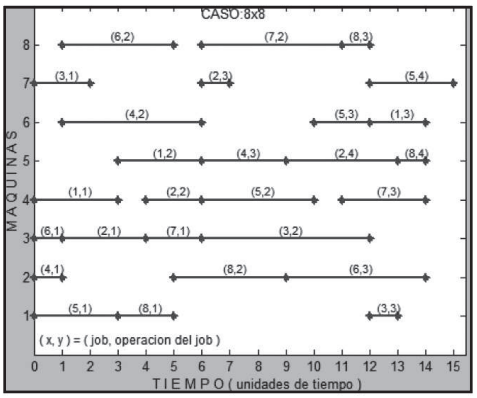
En el caso del problema de 15 jobs, 10 máquinas con 56 operaciones, una de las mejores secuencias de operaciones obtenidas es:1-53-17-18-45-33-9-5-13-41-25-29-49-37-38-14-6-26-2-34-30-21-15-10-46-11-42-47-27-50-48-23-39-28-54-51-19-3-20-7-35-24-31-43-16-52-12-55-40-22-32-36-56-8-44-4, con WT = 91 , WM = 11 y CM = 12. El Diagrama de Gannt de la Figura 11, verifica los resultados. encontrada.

Figura 9. Diagrama de Gannt, problema 4x5.



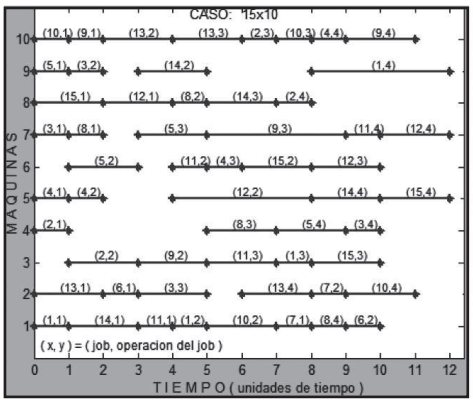
Fuente: Elaboración propia.

Figura 10. Diagrama de Gannt, problema 8x8.



Fuente:. Elaboración propia.

Figura 11. Diagrama de Gannt, problema 15x10.



Fuente: Elaboración propia.

4. CONCLUSIONES

Se solucionó el problema del FJSS desde un enfoque jerárquico que divide el problema en dos subproblemas, en el primero se soluciona la asignación óptima de máquinas a cada operación (enrutamiento) minimizando en las máquinas el máximo Workload (WM) y el total Workload (WT) y en el segundo subproblema conociendo las máquinas seleccionadas se soluciona la secuencia óptima de las operaciones en cada una de las máquinas (secuenciamiento) minimizando el Makespan (CM).

Por la naturaleza combinatoria NP-hard del problema del FJSS, para la solución se ha aplicado técnicas metaheurísticas. En otros enfoques jerárquicos han utilizado diferentes soluciones metaheuristicas para ambos subproblemas, en la presente propuesta para ambos subproblemas se han aplicado algoritmos genéticos de buen desempeño, en el primer subproblema se ha utilizado un algoritmo genético modificado de pocos miembros de la población, mientras que en el segundo un algoritmo genético más convencional.

En el algoritmo de enrutamiento ha dado buenos resultados la propuesta de cruce de cromosomas, en donde los puntos de cruce son limitadas por las máquinas que ejecutan las operaciones de un job, el cual es seleccionado aleatoriamente, lo mismo que los tres tipos de mutación propuestos las que se ejecutan probabilísticamente.

En la técnica propuesta la minimización de la función objetivo dada por la suma de W_T y W_M ha dado buenos resultados a pesar que no se ponderan ambos sumandos como lo hacen en otras investigaciones de la literatura.

En el cromosoma de secuencias, a diferencia de otras propuestas, se han representado a todas las operaciones manteniendo el orden de precedencia de las operaciones en cada job, es por eso que se ha creado un proceso de mutación en donde solo se realiza si los genes pertenecen a diferentes jobs.

El desempeño del algoritmo propuesto, ha mostrado ser eficiente al solucionar los problemas planteados por Kacem, los resultados comparados con las de otros investigadores es superado en el problema de 15x10, en donde se alcanza una solución de $C_M = 12$ frente a otras soluciones de $C_M = 11$. Sin embargo, los tiempos de ejecución del algoritmo superan en los cinco problemas de Kacem a todas las demás propuestas.

Se han podido validar los resultados objetivamente mediante la incorporación en el programa de la generación automática de Diagramas de Gannt.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Brandimarte, P. (1993). Routing and Scheduling in a Flexible Job Shop by Tabu Search. *Annals of Operations Research,* 41(3), 157-183.

[2] Genova K., Kirilov L., Guliashki V. (2015). A Survey of Solving Approaches for Multiple Objective Flexible Job Shop Scheduling Problems. *Cybernetics And Information Technologies,* 15(2), 3-21.

[3] Haupt, R. & Haupt, S. (2004). *Practical genetic algorithms.* Recuperado de http://it-ebooks. directory/book-0471455652.html

[4] 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.

[5] Kacem, I., Hammadi, S., Borne, P. (2002a). Pareto-Optimality Approach for Flexible Job Shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic. *Mathematics and Computers in Simulation,* 60(3-5), 245–276.

[6] Li, JQ., Pan, QK., Liang, YC. (2010). An Effective Hybrid Tabu Search Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problems. *Computers & Industrial Engineering,* 59(4), 647-662.

[7] Mekni, S., Chaâr, B. (2015). Multiobjective Flexible Job Shop Scheduling Using A Modified Invasive Weed Optimization. *International Journal on Soft Computing* (IJSC), 6(1), 25-36. DOI: 10.5121/ijsc.2015.6103 25

[8] Pezzella, F., Morganti, G., Ciaschetti G. (2008). A genetic algorithm for the Flexible Job shop Scheduling Problem. *Computers & Operations Research*, 35(10), pp. 3202- 3212.

[9] Xia, W., Wu, Z. (2005). An Effective Hybrid Optimization Approach for Multi-Objective Flexible Job Shop Scheduling Problems. *Computers & Industrial Engineering,* 48(2), 409-425.

[10] Xing, LN., Chen, YW., Yang KW. (2009). Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling. *Applied Soft Computing,* 9(1), 362–376.

[11] Xing, LN., Chen, YW., Yang, KW. (2009b). An Efficient Search Method for Multi Objective Flexible Job Shop Scheduling Problems. *Journal of Intelligent manufacturing,* 20(3), 283-293.

[12] Zhang, G., Shao, X., Li, P., Gao, L. (2009). An Effective Hybrid Particle Swarm Optimization Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problem. *Computers & Industrial Engineering,* 56(4), 1309-1318.

[13] Ziaee, M. (2014). An Efficient Heuristic Algorithm for Flexible Job Shop Scheduling with Maintenance Constraints. *Applied Mathematics and Sciences: An International Journal (MathSJ),* 1(1), 19-31.