Información Tecnológica
Vol. 29(5), 299-314 (2018)
http://dx.doi.org/10.4067/S0718-07642018000500299

Algoritmo Genético Simple para Resolver el Problema de Programación de la Tienda de Trabajo (Job Shop Scheduling)

Miguel Jiménez-Carrión[(1,2)]
(1) Facultad de Ingeniería Industrial, Dpto. de Investigación de Operaciones, Universidad Nacional de Piura. Campus Universitario s/n, Castilla Piura - Perú
(2) Concejo Nacional de Ciencia y Tecnología, Perú (e-mail: mjimenezc@gmail.com)

Recibido Feb. 15, 2018; Aceptado Abr. 9, 2018; Versión final May. 23, 2018, Publicado Oct. 2018

Resumen

Se ha implementado un algoritmo genético simple para resolver el Problema de Programación de la Tienda de Trabajo (Job Shop Scheduling Problem, JSSP). El diseño del cromosoma representa una solución factible y cumple con todas las restricciones. Se utilizó el mecanismo de selección por torneo, una reproducción del 95% basada en emparejamiento parcial con dos puntos de cruza, una estrategia mixta en la etapa de mutación combinando el método de intercambio y el método de inversión usando dos puntos aleatorios en cada máquina y un porcentaje de mutación progresivo entre 2% al 5%. Los resultados muestran que el algoritmo debe ejecutarse con 100 individuos como tamaño de población y 500 generaciones para problemas cuyos tiempos de operación están entre 0 y 10 unidades de tiempo y con 100 individuos y 1500 generaciones para problemas entre 0 y 100 unidades de tiempo. El estudio muestra que el algoritmo implementado encuentra soluciones óptimas en el primer caso y soluciones altamente competitivas en el segundo caso. Estas son comparables con los resultados publicados en la literatura que por lo general son respuestas a algoritmos híbridos repotenciados con otras metaheurísticas.

Palabras clave: asignación de tareas; algoritmo genético; planificación detallada; secuencia de operaciones

Simple Genetic Algorithm to solve the Job Shop Scheduling Problem

Abstract

A simple genetic algorithm has been implemented to solve the Job Shop Scheduling Problem (JSSP). The chromosome design represents a feasible solution and meets all restrictions. The selection mechanism per tournament was used, a 95% reproduction based on partial pairing with two crossing points, a mixed strategy in the mutation stage combining the method of exchange and the method of investment using two random points in each machine and a percentage of progressive mutation between 2% to 5%. The results show that the algorithm must be executed with 100 individuals as population size and 500 generations for problems whose operation times are between 0 and 10 units of time and with 100 individuals and 1500 generations for problems between 0 and 100 units of time. The study shows that the implemented algorithm finds optimal solutions in the first case and highly competitive solutions in the second case. These are comparable with the results published in the literature that are generally responses to hybrid algorithms re-energized with other metaheuristics.

Keywords: assigning tasks; genetic algorithm; detailed planning; sequence of operations

INTRODUCCIÓN

El problema de Programación de la Tienda de Trabajo, corresponde a un tipo de problema de planificación de tareas, y es un problema de optimización; en donde uno de los objetivos principales es la disminución de tiempos en la tarea de planificación, sin embargo (Cheng, Gen, y Tsujimura, 1996), manifiestan que "el problema ha capturado el interés de un número significativo de investigadores y se ha publicado mucha literatura, pero aún no se ha encontrado un algoritmo de solución eficiente para resolverlo en tiempo polinomial", así mismo (Luo, 2017) hace una revisión de los problemas de programación de la oferta del trabajo (JSSP), que van desde una única máquina hasta JSSP flexible bajo algoritmos genéticos, del mismo modo (Chen, Li, y Wang, 2004) proponen un novedoso multi agente basado en un algoritmo genético para resolver el JSSP y los resultados muestran que obtiene un mejor rendimiento óptimo. En este contexto la programación es una asignación de un conjunto de recursos a un conjunto de actividades durante un período determinado de tiempo. En el entorno de producción, estos recursos se conocen como máquinas y las actividades como operaciones. Por lo tanto, la tarea de programar es encontrar un orden de procesamiento que optimice el entorno de producción (Bhatt y Chauhan, 2015); en ese sentido (Salazar-Hornig y Medina, 2013) presentan un algoritmo genético para la programación de trabajos en un sistema de máquinas paralelas idénticas, con tiempos de preparación dependientes de la secuencia, con el objetivo de minimizar el *makespan* (C_{max}). En el mismo sentido (Lopez et al., 2015), codificó un algoritmo genético simple y desarrolló una metodología de programación de la producción para configuraciones de tipo de línea de flujo (flow show) hibrida flexible, los resultados obtenidos son de buena calidad con tiempos computacionales bastante razonables y con coeficiente de variación del orden del 2%, en relación a los valores del makespan. El algoritmo genético que se propone utiliza una estrategia mixta en la etapa de mutación alternando dos mecanismos de mutación en forma concurrente e incremental al variar la tasa de mutación de 2% al 5% a lo largo del proceso evolutivo y se probará varios mecanismos de selección para elegir el que logra mejor desempeño; el algoritmo se probará sobre un conjunto de problemas de prueba generados aleatoriamente y se verificará su eficiencia con algoritmos de los cual se conoce su solución en el repositorio web.

El Problema de Programación de la Tienda de Trabajo, consiste básicamente en planificar un conjunto de N trabajos $\{J_1,.., J_N\}$ sobre un conjunto de M recursos o máquinas físicas $\{R_1,\ldots, R_M\}$. Cada trabajo J_i consta de un conjunto de operaciones o tareas $\{\theta_{i1},\ldots,\theta_{iM}\}$ que deben ser ejecutadas de forma secuencial. Cada tarea θ_{ij} tiene asociado un tiempo de procesamiento sin interrupción de du_{ij} unidades de tiempo durante el cual requiere del uso exclusivo de un único recurso. Cada trabajo tiene un tiempo de inicio más temprano y en ocasiones se considera también un tiempo de término más tardío, lo que obliga a que los tiempos de inicio de las tareas tomen valores en dominios finitos. Cabe considerar que las operaciones de un mismo trabajo deben ejecutarse en un orden determinado, de forma que $\theta_{(i+1)j}$ no puede comenzar hasta que θ_{ij} haya terminado completamente (Gholami y Sotskov, 2014). Además, las operaciones que comparten una misma máquina son no interrumpibles y mutuamente exclusivas. Las restricciones del problema se pueden resumir de la siguiente manera: 1) No está permitido que dos operaciones del mismo trabajo se procesen simultáneamente, 2) Ninguna operación tiene prioridad sobre las demás, 3) Ningún trabajo puede ser procesado más de una vez en la misma máquina, 4) Cada trabajo es procesado hasta concluirse, 5) Un trabajo puede iniciarse en cualquier momento siempre y cuando esté disponible la máquina y no se haya especificado un tiempo de inicio, 6) Los trabajos tienen que esperar a que la siguiente máquina esté disponible para que éste sea procesado, 7) Ninguna máquina puede procesar más de un trabajo a la vez, 8) Los tiempos de configuración y cambio de máquina son independientes del orden de procesamiento y éste está incluido en los tiempos de procesamiento, 9) Hay sólo un tipo de máquina, 10) Las máquinas pueden estar ociosas en cualquier momento del plan de trabajo, 11) Las máquinas están disponibles en cualquier momento y 12) Las restricciones tecnológicas están bien definidas y son previamente conocidas, además de que son inamovibles; algunas de estas restricciones son implícitas pero se comentan de manera explícita.

Como lo manifiestan (Applegate y Cook, 1991) y (Liang, Xiao-Yuan, y Yun-Cai, 2012), el problema de Programación de la Tienda de Trabajo es un problema difícil de resolver. En esta investigación, en primer lugar el objetivo consiste en encontrar una planificación factible y altamente satisfactoria, es decir, una asignación de tiempos de inicio t_{ij} para cada una de las tareas, que minimice el *makespan* o tiempo de finalización de la última tarea incluyendo los problemas que tienen la misma secuencia de producción cómo los planteados por (Delgado et al., 2012) y aquellos que tienen diferente secuencia de operación; en segundo lugar la herramienta para alcanzarlo es un algoritmo genético simple usando solo estrategias propias, conforme lo manifiesta (Zhang, Gao, y Shi, 2011). Sobre el particular y haciendo uso de otras herramientas inteligentes, (Castrillón, Ruiz-Herrera, y Sarache, 2016) proponen una metodología basada en redes neuronales para la programación de pedidos en un ambiente job shop-open shop, en razón a que otras herramientas presentan dificultades prácticas cuando se requieren cambios de programación, pues exigen reconfigurar nuevamente el problema para obtener una nueva solución; los resultados obtenidos son comparadas con otras técnicas de programación de la producción, encontrándose desempeños superiores en el makespan del orden del 30% al 164%.

METODOLOGÍA

En este contexto el aplicar algoritmos genéticos a un problema de optimización hace que la investigación se conduzca basado en un diseño experimental en el cual se espera explicar una variable dependiente S (Solución), a partir de variables independientes que se pueden controlar como tp y ng (Tamaño de la Población y Número de generaciones); los parámetros porcentaje de Mutación y porcentaje de Cruza, se calibraron haciendo pruebas preliminares con los prototipos de problema JSSP de los cuales se conoce su solución óptima que generalmente son problemas que pueden ser resueltos con el software académico WinQSB. Matemáticamente la solución de un JSSP se puede expresar como: $S = f(tp, ng, P)$; tp y ng se determinan experimentalmente utilizando un diseño en disposición factorial con cuatro repeticiones y tres niveles cada factor, P representa el Problema en sí; y como es de esperar este algoritmo no garantiza encontrar soluciones óptimas sino soluciones altamente aceptables en un corto período de tiempo, el rigor científico que respalda esta investigación es la estadística cuando se desarrolla el análisis de varianza del diseño experimental y las comparaciones de promedios de Duncan.

Los pasos de esta investigación son: 1) Se muestra una instancia de problema y lo que se hace con esta instancia se puede hacer con todas las instancias que se pretenda resolver, la instancia en mención es un problema real de un taller metalmecánico en Piura; 2) Luego se representa el problema como una red adicionando dos nodos especiales un nodo inicio y un nodo final; 3) En seguida se construye el modelo matemático basado en la red de esta instancia y se resuelve este modelo utilizando el software académico WinQSB; 4) Posteriormente se hace un análisis extensivo a otras instancias mostrando la complejidad computacional del problema, observándose la necesidad de utilizar metaheurísticas cómo los algoritmos genéticos que se plantea en esta investigación; 5) Luego se diseña y construye el Algoritmo Genético Simple haciendo varias pruebas de calibración para determinar los diferentes operadores genéticos a utilizarse; 6) Se hace un análisis y discusión de resultados y 7) Finalmente se formulan las conclusiones del algoritmo.

Instancia del Problema

Una instancia del problema a resolver se muestra en la tabla 1 que sigue y corresponde a un ejemplo real, en donde hay 8 trabajos (J1, J2… J8) y 14 máquinas (M1, M2…M14), instancia 8 x 14; para cada trabajo se muestra la secuencia de las máquinas a seguir, se puede observar que algunos trabajos solo necesitan una máquina, en consecuencia solo requieren una operación y como mucho en esta instancia el trabajo 2 requiere de 5 operaciones para concluirse en consecuencia se requiere de 5 máquinas. Así mismo se puede apreciar el tiempo de procesamiento de cada una de las operaciones en las máquinas; el tiempo cero (0) india que el trabajo J no requiere tiempo de la máquina M.

Tabla 1: Instancia de Problema 8 x 14, lado izquierdo secuencia de los trabajos en las máquinas y lado derecho tiempos de operación

Secuencia de los trabajos en las máquinas
J1: M1 - M3
J2: M7 - M13 - M5 - M1 - M3
J3: M7 - M14 - M5 - M9
J4: M6
J5: M10
J6: M1 - M3
J7: M6 - M12 - M11
J8: M11

	Tiempo de operación en minutos							
	J1	J2	J3	J4	J5	J6	J7	J8
M1	30	120	0	0	0	120	0	0
M2	0	0	0	0	0	0	0	0
M3	10	30	0	0	0	30	0	0
M4	0	0	0	0	0	0	0	0
M5	0	20	20	0	0	0	0	0
M6	0	0	0	10	0	0	20	0
M7	0	30	30	0	0	0	0	0
M8	0	0	0	0	0	0	0	0
M9	0	0	90	0	0	0	0	0
M10	0	0	0	0	20	0	0	0
M11	0	0	0	0	0	0	20	20
M12	0	0	0	0	0	0	20	0
M13	0	30	0	0	0	0	0	0
M14	0	0	30	0	0	0	0	0

Representación del problema como una red.

En investigación de operaciones muchas veces se representa el problema como una red y el modelo matemático se basa en la red, en este caso se construye una red agregando dos nodos auxiliares un nodo inicio (I) y un nodo final (F) luego se construye la red basado en la información de la instancia problema; el objetivo es ir desde el nodo inicio (I), al nodo final (F), a través de la red que de manera que todos los trabajos queden concluidos en el menor tiempo posible, ver figura 1.

Obsérvese que el trabajo 1 y el trabajo 6 deben iniciarse en la máquina 1 lo que no sabemos es con cual trabajo debe iniciar la máquina 1, el trabajo 4 y el trabajo 7 deben iniciarse en la máquina 6 y ocurre lo mismo no se sabe que trabajo inicia primero en esa máquina, así mismo el trabajo 2 y trabajo 3 deben iniciarse en la máquina 7 y sucede igual; además se sabe, que el trabajo 5 debe iniciarse en la máquina 10 y que el trabajo 8 debe iniciarse en la máquina 11, estos 2 últimos trabajo pueden iniciarse de inmediato. Hasta este momento se representa los nodos de la red como el par (Nº de máquina, Nº de trabajo) y al inicio tendremos 9 nodos: (I), (1,1), (1,6), (6,4), (6,7), (7,2), (7,3), (10,5) y (11,8) para representar el inicio (I) de la red y el procesamiento:

Fig. 1: Inicio de la construcción de la red Instancia 8 x14

de los trabajos en las máquinas, como: realizar en la máquina 1 el trabajo 1, realizar en la máquina 1 el trabajo 6; realizar en la máquina 6 el trabajo 4, realizar en la máquina 6 el trabajo 7; realizar el trabajo 2 en la máquina 7, realizar el trabajo 3 en la máquina 7; cómo se podrá apreciar en las máquinas 1, 6 y 7 hay dos opciones para cada máquina y no teniendo más opciones cualquiera de ellos pueden comenzar de inmediato por lo que tiempo para pasar del nodo I, a ellos será cero (0); y lo representamos de la manera como se muestra en la Figura 1. Los $Y_{1,1}$, $Y_{1,6}$, $Y_{6,4}$, $Y_{6,7}$, $Y_{7,2}$, $Y_{7,3}$, $Y_{10,5}$ y $Y_{11,8}$ representan el tiempo en que debe iniciarse cada nodo; sin embargo alguien puede decir que no hace falta representarlos con una variable porque se conocen que son cero (0), en realidad es cierto para los nodos (10,5) y (11,8) pero no para el resto en razón a que no se sabe con qué trabajo iniciará la máquina 1, la 6 y la 7, para que el tiempo total de terminar todos los trabajos sea mínimo, y como no se sabe son una variable.

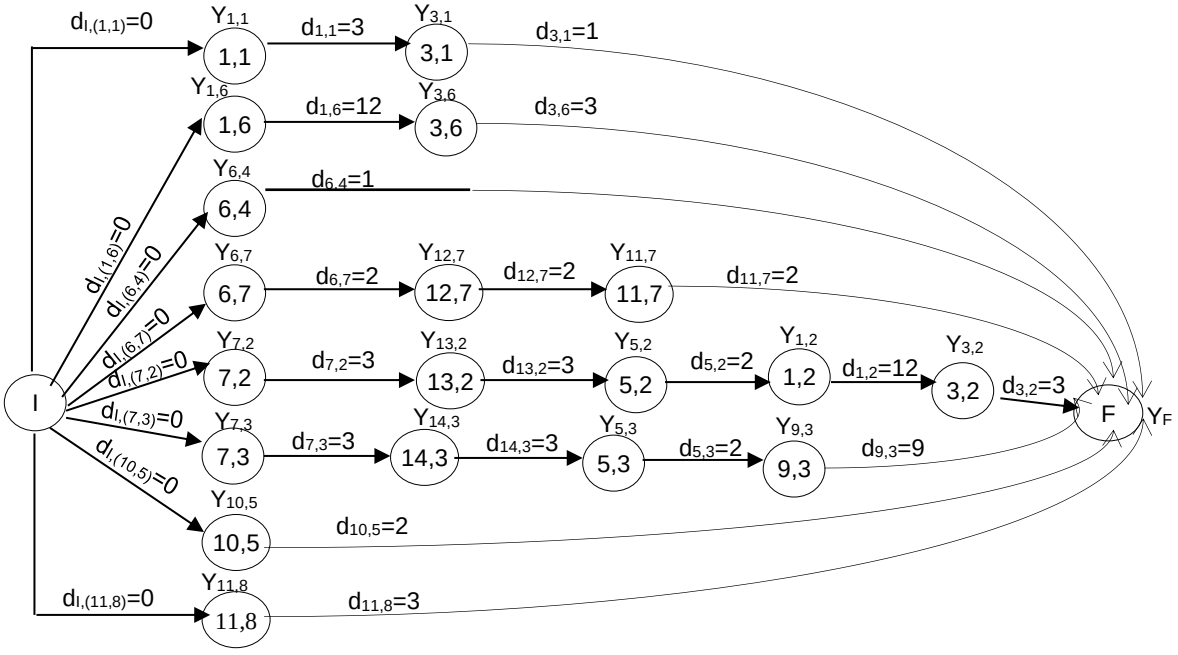


Fig. 2: Representación de la secuencia de los trabajos en las máquinas. Las flechas llenas, indican la precedencia de las operaciones y el número sobre ella la duración de la máquina i con el trabajo j.

Ahora bien, de iniciarse una tarea (i,j) en el tiempo $Y_{i,j}$ existe un tiempo de duración de la tarea representada por un arco y un número asociado como $d_{i,j}$; ahora representamos los nodos que deben seguir en la red y están dados por la secuencia que sigue cada trabajo, tales como después del nodo (1,1) sigue (3,1) hacer en la máquina 3 el trabajo 1; del mismo modo luego de (1,6) sigue (3,6) hacer en la máquina 3 el trabajo 6; con relación al nodo (6,7) le sigue el nodo (12,7), y termina en el nodo (11,7); así mismo al nodo (7,2) le sigue la secuencia de nodos (13,2), (5,2), (1,2) y termina con el nodo (3,2); finalmente después del nodo (7,3) le sigue el nodo (14,3), el nodo (5,3) y termina con el nodo (9,3); los nodos (6,4), (10,5) y (11,8) no tienen nodos sucesores. En la figura 2, se aprecia la secuencia que siguen los trabajos en las máquinas con sus duraciones en decenas de minuto de cada operación que se procesa de forma completa.

Continuando con la construcción de la red, observamos que en primer lugar la máquina 1 puede hacer el trabajo 1 o el trabajo 6 y como no sabemos que trabajo procesará primero esta máquina, trazamos una flecha punteada del nodo (1,1) al nodo (1,6) y en ambos extremos una flecha para indicar que tanto (1,1) o (1,6) pueden iniciarse primero. Esto mismo se hace con todos los nodos que tienen la misma máquina y diferente trabajo, los arcos se representan en color azul y en líneas punteadas para diferenciarlos de los primeros arcos. El resultado se muestra en la figura 3, y para mayor claridad se han omitido las etiquetas del tiempo de maquinado $(d_{(i,j)})$, solo aparece el tiempo que demora hacer en la máquina i el trabajo j. Este tipo de redes se le conoce en la literatura como Grafo Disyuntivo, sobre el particular (Bażewicz, Pesch, y Sterna, 2000) manifiestan que es uno de los modelos más populares utilizados para describir instancias del JSSP, que se ha explorado de forma muy intensiva.

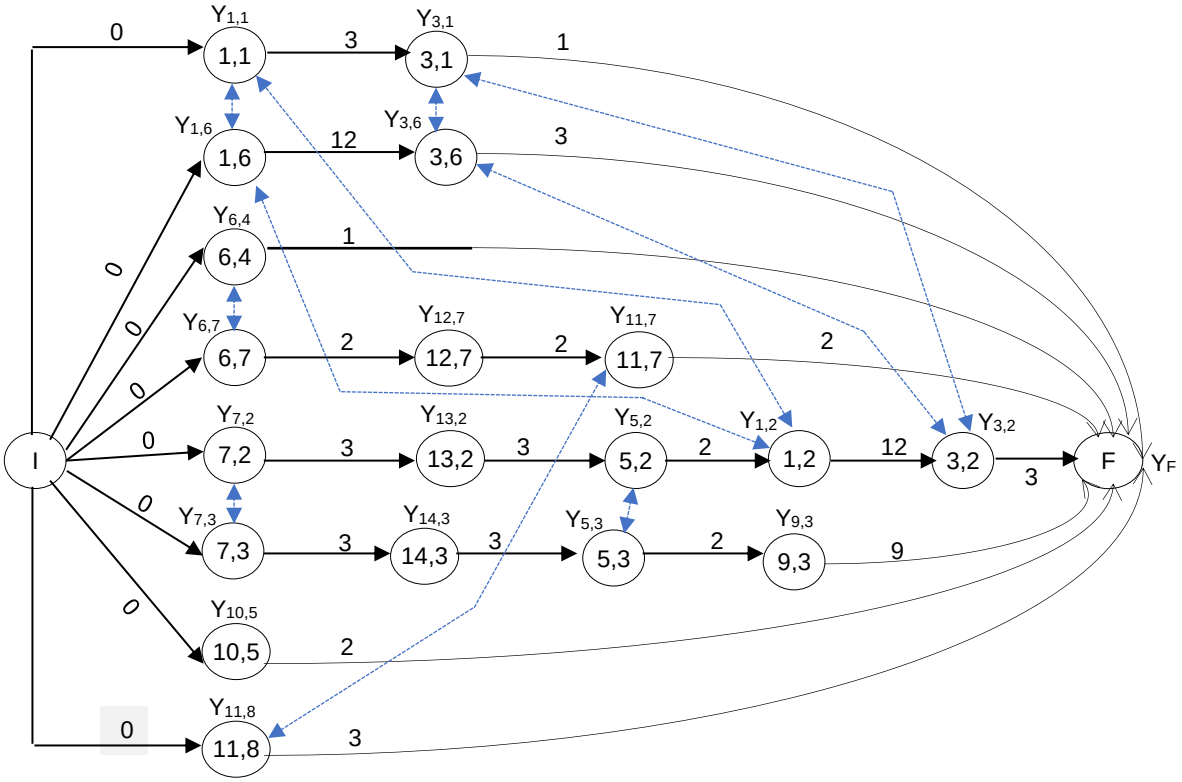


Fig. 3: Red disyuntiva para la instancia 8x14 del problema real, las flechas bidireccionales punteadas representa las disyunciones o posibles interferencias.

Modelo matemático del problema basado en la red

I: representa el nodo inicio de la red
F: representa el nodo final de la red

Los arcos direccionados con líneas llenas que salen del nodo "I" a los diferentes nodos (los primeros 8 nodos), representan las diferentes alternativas factibles de asignar máquinas a los trabajos (M, J), obsérvese que se da inicio a todos los trabajos. La siguiente columna de nodos corresponde a la secuencia que debe seguir cada trabajo por las máquinas y se unen con arcos direccionados con líneas llenas y cuando se concluye con un trabajo se completa el grafo con un arco con línea direccional llena hasta el nodo final F. Los arcos con líneas punteadas y bidireccionales representan disyunciones entre pares de nodos. Por ejemplo, los nodos (1,1), (1,6) y (1,1) y (1,2), etc. Indican que al inicio la máquina 1 puede hacer el trabajo 1, el trabajo 6 o el trabajo 2.

Definición de Variables y parámetros

En Investigación de Operaciones cuando se construye un modelo matemático primero se define las variables de decisión del modelo y que en este caso están representadas en la red cómo:

$Y_{m,j}$ = representa el tiempo en que la máquina m, inicia el trabajo j.

$d_{m,j}$ = representa la duración del tiempo de ejecución de la máquina m con el trabajo j

Y_F = representa el tiempo en el cual todos los trabajos son terminados en todas las máquinas (makespan).

Yk = representa una variable binaria para manejar la disyunción.

G = parámetro de valor lo suficientemente grande respecto a los tiempos de operación.

A continuación, se muestra la función objetivo la misma que está sujeta a las restricciones que se listan en la tabla 2 hasta tabla 5; las variables Y_{ij} y Y_F, son mayores o iguales que cero (0) y enteras, Y_k es binaria k= 1, 2, ..10; el valor de G es muy grande respecto al tiempo de las operaciones.

Función Objetivo: Minimizar $Z = Y_F$
s.a

Tabla 2: Restricciones tecnológicas de precedencia, de la Instancia de Problema 8 x 14.

01.	$Y_{3,1}$	-	$Y_{1,1}$	>=	$d_{1,1}$
02.	$Y_{3,6}$	-	$Y_{1,6}$	>=	$d_{1,6}$
03.	$Y_{12,7}$	-	$Y_{6,7}$	>=	$d_{6,7}$
04.	$Y_{11,7}$	-	$Y_{12,7}$	>=	$d_{12,7}$
05.	$Y_{13,2}$	-	$Y_{7,2}$	>=	$d_{7,2}$
06.	$Y_{5,2}$	-	$Y_{13,2}$	>=	$d_{13,2}$
07.	$Y_{1,2}$	-	$Y_{5,2}$	>=	$d_{5,2}$
08.	$Y_{3,2}$	-	$Y_{1,2}$	>=	$d_{1,2}$
09.	$Y_{14,3}$	-	$Y_{7,3}$	>=	$d_{7,3}$
10.	$Y_{5,3}$	-	$Y_{14,3}$	>=	$d_{14,3}$
11.	$Y_{9,3}$	-	$Y_{5,3}$	>=	$d_{5,3}$

Tabla 3: Restricciones tecnológicas de precedencia, relacionadas con Y_F, hay una por cada nodo, de la Instancia de Problema 8 x 14.

12.	Y_F	-	$Y_{3,1}$	>=	$d_{3,1}$
13.	Y_F	-	$Y_{1,1}$	>=	$d_{1,1}$
14.	Y_F	-	$Y_{3,6}$	>=	$d_{3,6}$
15.	Y_F	-	$Y_{1,6}$	>=	$d_{1,6}$
16.	Y_F	-	$Y_{6,4}$	>=	$d_{6,4}$
17.	Y_F	-	$Y_{11,7}$	>=	$d_{11,7}$
18.	Y_F	-	$Y_{12,7}$	>=	$d_{12,7}$
19.	Y_F	-	$Y_{6,7}$	>=	$d_{6,7}$
20.	Y_F	-	$Y_{3,2}$	>=	$d_{3,2}$
21.	Y_F	-	$Y_{1,2}$	>=	$d_{1,2}$
22.	Y_F	-	$Y_{5,2}$	>=	$d_{5,2}$
23.	Y_F	-	$Y_{13,2}$	>=	$d_{13,2}$
24.	Y_F	-	$Y_{7,2}$	>=	$d_{7,2}$
25.	Y_F	-	$Y_{9,3}$	>=	$d_{9,3}$
26.	Y_F	-	$Y_{5,3}$	>=	$d_{5,3}$
27.	Y_F	-	$Y_{14,3}$	>=	$d_{14,3}$
28.	Y_F	-	$Y_{7,3}$	>=	$d_{7,3}$
29.	Y_F	-	$Y_{10,5}$	>=	$d_{10,5}$
30.	Y_F	-	$Y_{11,8}$	>=	$d_{11,8}$

Tabla 4: Restricciones tecnológicas de no interferencia o disyunción, de la Instancia de Problema 8 x 14.

31.	$Y_{1,6}$	-	$Y_{1,1}$	>=	$d_{1,1}$			ó			$Y_{1,1}$	-	$Y_{1,6}$	>=	$d_{1,6}$
32.	$Y_{6,7}$	-	$Y_{6,4}$	>=	$d_{6,4}$			ó			$Y_{6,4}$	-	$Y_{6,7}$	>=	$d_{6,7}$
33.	$Y_{7,3}$	-	$Y_{7,2}$	>=	$d_{7,2}$			ó			$Y_{7,2}$	-	$Y_{7,3}$	>=	$d_{7,3}$
34.	$Y_{11,7}$	-	$Y_{11,8}$	>=	$d_{11,8}$			ó			$Y_{11,8}$	-	$Y_{11,7}$	>=	$d_{11,7}$
35.	$Y_{1,6}$	-	$Y_{1,2}$	>=	$d_{1,2}$			ó			$Y_{1,2}$	-	$Y_{1,6}$	>=	$d_{1,6}$
36.	$Y_{1,2}$	-	$Y_{1,1}$	>=	$d_{1,1}$			ó			$Y_{1,1}$	-	$Y_{1,2}$	>=	$d_{1,2}$
37.	$Y_{3,1}$	-	$Y_{3,6}$	>=	$d_{3,6}$			ó			$Y_{3,6}$	-	$Y_{3,1}$	>=	$d_{3,1}$
38.	$Y_{3,2}$	-	$Y_{3,1}$	>=	$d_{3,1}$			ó			$Y_{3,1}$	-	$Y_{3,2}$	>=	$d_{3,2}$
39.	$Y_{3,2}$	-	$Y_{3,6}$	>=	$d_{3,6}$			ó			$Y_{3,6}$	-	$Y_{3,2}$	>=	$d_{3,2}$
40.	$Y_{5,2}$	-	$Y_{5,3}$	>=	$d_{5,3}$			ó			$Y_{5,3}$	-	$Y_{5,2}$	>=	$d_{5,2}$

Tabla 5: Restricciones que remplazarán a las de la tabla 4, artificio matemático para romper la disyunción, de la Instancia de Problema 8 x 14.

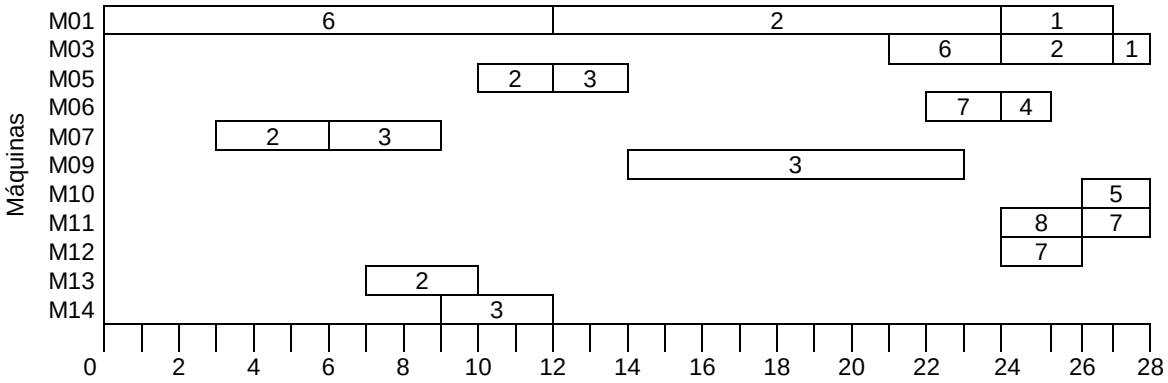
31.	$Y_{1,6}$	–	$Y_{1,1}$	+ $G(1 – Y_1)$	>=	$d_{1,1}$
32.	$Y_{1,1}$	–	$Y_{1,6}$	+ $G(Y_1)$	>=	$d_{1,6}$
33.	$Y_{6,7}$	–	$Y_{6,4}$	+ $G(1 – Y_2)$	>=	$d_{6,4}$
34.	$Y_{6,4}$	–	$Y_{6,7}$	+ $G(Y_2)$	>=	$d_{6,7}$
35.	$Y_{7,3}$	–	$Y_{7,2}$	+ $G(1 – Y_3)$	>=	$d_{7,2}$
36.	$Y_{7,2}$	–	$Y_{7,3}$	+ $G(Y_3)$	>=	$d_{7,3}$
37.	$Y_{11,7}$	–	$Y_{11,8}$	+ $G(1 – Y_4)$	>=	$d_{11,8}$
38.	$Y_{11,8}$	–	$Y_{11,7}$	+ $G(Y_4)$	>=	$d_{11,7}$
39.	$Y_{1,6}$	–	$Y_{1,2}$	+ $G(1 – Y_5)$	>=	$d_{1,2}$
40.	$Y_{1,2}$	–	$Y_{1,6}$	+ $G(Y_5)$	>=	$d_{1,6}$
41.	$Y_{1,2}$	–	$Y_{1,1}$	+ $G(1 – Y_6)$	>=	$d_{1,1}$
42.	$Y_{1,1}$	–	$Y_{1,2}$	+ $G(Y_6)$	>=	$d_{1,2}$
43.	$Y_{3,1}$	–	$Y_{3,6}$	+ $G(1 – Y_7)$	>=	$d_{3,6}$
44.	$Y_{3,6}$	–	$Y_{3,1}$	+ $G(Y_7)$	>=	$d_{3,1}$
45.	$Y_{3,2}$	–	$Y_{3,1}$	+ $G(1 – Y_8)$	>=	$d_{3,1}$
46.	$Y_{3,1}$	–	$Y_{3,2}$	+ $G(Y_8)$	>=	$d_{3,2}$
47.	$Y_{3,2}$	–	$Y_{3,6}$	+ $G(1 – Y_9)$	>=	$d_{3,6}$
48.	$Y_{3,6}$	–	$Y_{3,2}$	+ $G(Y_9)$	>=	$d_{3,2}$
49.	$Y_{5,2}$	–	$Y_{5,3}$	+ $G(1 – Y_{10})$	>=	$d_{5,3}$
50.	$Y_{5,3}$	–	$Y_{5,2}$	+ $G(Y_{10})$	>=	$d_{5,2}$

La solución del modelo matemático, con G = 1000 indica Y_F = 280, los valores de los parámetros d_{mj}, se encuentran en la red y el valor de cada una de las variables $Y_{i,j}$ y Y_k, se muestran en la tabla 6, y su correspondiente diagrama de Gantt en la tabla 7, en la cual cada unidad de tiempo es equivalente a 10 minutos, observándose que en las máquinas 2, 4 y 8 no son utilizadas por lo que no aparecen en el diagrama.

Tabla 6: Solución del modelo matemático usando WinQSB, de la instancia 8x14.

$Y_{1,1}$ = 240	$Y_{3,6}$ = 210	$Y_{7,2}$ = 30	$Y_{11,8}$ = 240	Y_1 = 0	Y_6 = 0
$Y_{1,2}$ = 120	$Y_{5,2}$ = 100	$Y_{7,3}$ = 60	$Y_{12,7}$ = 240	Y_2 = 0	Y_7 = 1
$Y_{1,6}$ = 0	$Y_{5,3}$ = 120	$Y_{9,3}$ = 140	$Y_{13,2}$ = 70	Y_3 = 1	Y_8 = 0
$Y_{3,1}$ = 270	$Y_{6,4}$ = 240	$Y_{10,5}$ = 260	$Y_{14,3}$ = 90	Y_4 = 1	Y_9 = 0
$Y_{3,2}$ = 240	$Y_{6,7}$ = 220	$Y_{11,7}$ = 260	Y_F = 280	Y_5 = 0	Y_{10} = 0

Tabla 7: Diagrama de Gantt, correspondiente a la solución de la instancia 8x14, (makespan = 28)



Complejidad computacional

La representación del modelo matemático de la instancia del ejemplo 8 trabajos y 14 máquinas se ha necesitado 30 variables (20 enteras y 10 binarias) y 50 restricciones, sin embargo, no todos los trabajos pasan por todas la máquinas de haber sido así el modelo matemático tendría 504 variables y 1000 restricciones del mismo modo para un modelo de 15 trabajos y 15 máquinas y que todos los trabajos pasen por todas las máquinas el modelo matemático contendrá 1801 variables y 3585 restricciones, con el agregado que en todos los casos el modelo matemático es mixto porque tiene variables enteras y variables binarias. Por otro lado, si se evalúa la cantidad de soluciones posibles para la instancia del ejemplo habrían $(8!)^{14} = 3.00115 \times 10^{64}$ siempre y cuando todos los trabajos pasan por todas la máquinas y para 15 trabajos y 15 máquinas la cantidad de soluciones posibles es $(15!)^{15} = 5.5911 \times 10^{181}$. Cómo puede observarse debido a su complejidad en instancias grandes, no resulta posible contar con un método totalmente determinista para resolverlo en el caso general, en razón a que las posibles soluciones están determinadas por la expresión $(\boldsymbol{n}!)^{\boldsymbol{m}}$, donde $\boldsymbol{n}$ es el número de trabajos y $\boldsymbol{m}$ el número de máquinas conforme lo manifiesta Legun, (2004) reportado por (Castrillón et al., 2011).

Por otro lado, los métodos evolutivos como la programación evolutiva, las estrategias evolutivas y los algoritmos genéticos (AG) son los más utilizados para estos casos debido a que tienen la ventaja de aplicar en modelos de alta complejidad que difícilmente tienen solución con metodologías matemáticas convencionales, porque pueden evaluar infinitas soluciones óptimas (Bastidas, Bermúdez, Jaramillo, y Chejne, 2010). De ahí que, en los últimos años, se haya desarrollado diversas heurísticas de solución del JSSP basadas en algoritmos genéticos, búsqueda tabú, recocido simulado y colonia de hormigas (Cortés Rivera, 2004). Haciendo referencia a lo que manifiesta Blazewicz, Ecker, Schmidt, y Weglarz, (1994), reportado por (Cortés Rivera, 2004), los problemas de planificación de tareas (Scheduling) pueden ser descritos ampliamente como: “el problema de acomodar los recursos en el tiempo para realizar un conjunto de tareas". En la literatura de planificación de tareas se trata una gran variedad de problemas.

En el JSSP, con frecuencia se refieren a problemas de programación de la producción sin embargo también pueden referirse a otras áreas (por ejemplo, el tránsito aéreo al despegue y aterrizaje o médicos y enfermeras cuidando a un paciente en un hospital, etc.). Dentro de la gran variedad de problemas de planificación de recursos que existen, el JSSP es uno de los que han generado un mayor número de estudios. Esto se debe, sobre todo, a que es uno de los más difíciles de resolver y de esta clase es uno de los menos tratables (Garey y Johnson, 1979). La solución al problema planteado utilizando una de las herramientas inspiradas en la teoría de la evolución de los seres vivos, propuesta por Holland (1975) y reportado por (García Martínez, 2008), según éste Holland en su libro “Adaptación en sistemas naturales y artificiales”, propone una clase de algoritmos adaptativos, ideas como una abstracción de la evolución para resolver problemas prácticos y para servir de modelos computacionales de sistemas naturales evolutivos, métodos que posteriormente recibieron el nombre de Algoritmos Genéticos (AGs); se espera que el algoritmo genético simple que aquí se propone, resuelva el problema JSSP y contribuya a solucionar los problemas que aún permanecen en los algoritmos propuestos por varios autores como es el tiempo de ejecución aún demasiado largo y obtener soluciones altamente satisfactorias y de mejor calidad que las actuales soluciones reportados en los repositorios web.

DISEÑO Y CONSTRUCCIÓN DEL ALGORITMO GENÉTICO

El diseño de un algoritmo genético se inicia con el diseño del cromosoma, el cual es la etapa inicial del algoritmo y se caracteriza porque gran parte de la inventiva del diseño de un algoritmo genético descansa en esta etapa, específicamente en cómo se diseña el cromosoma que representará a cada individuo, cromosoma que contendrá de manera implícita las restricciones del problema (Jiménez Carrión, 2016, p. 17). En este

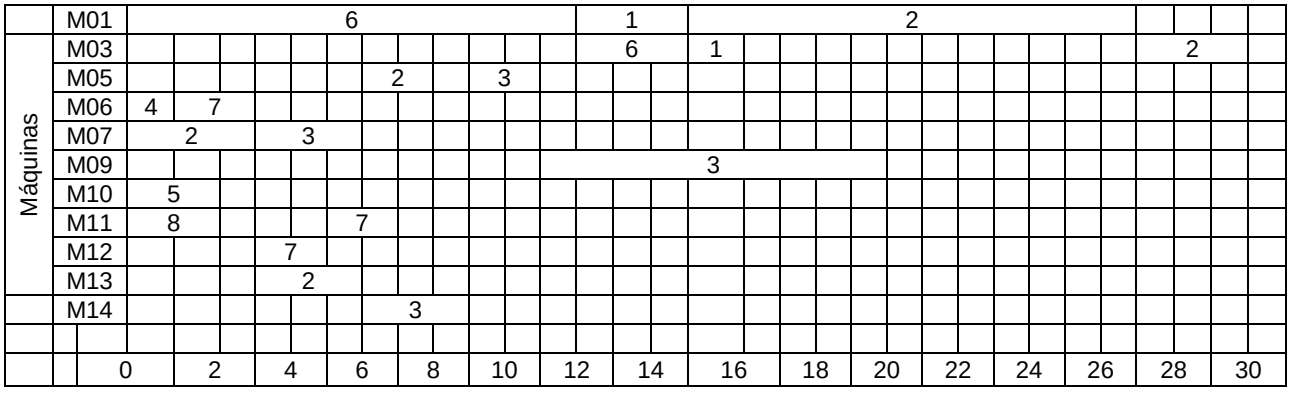
algoritmo el diseño del cromosoma está formado por una matriz de orden ***m*** x ***n***, donde "***m***" es el número de máquinas y "***n***" es el número de trabajos y cada fila de la matriz corresponde a una máquina y contiene una permutación de todos los trabajos a realizarse sin repetición, conforme se puede apreciar en la tabla 8, que muestra una instancia de cromosoma para el ejemplo real de 8 trabajos y 14 máquinas; donde la primera columna muestra las 14 máquinas (desde M1 hasta M14) y la primera fila contiene el texto referido a la permutación de los 8 trabajos (permutación entre 1 y 8 trabajos), los trabajos que están sombreados son los únicos que se pueden procesar en cada máquina, así: la máquina 1 solo puede procesar los trabajos 1, 2 y 6; la máquina 2 no procesa ningún trabajo lo que significaría que esa máquina queda libre; la máquina 3 puede procesar los trabajos 1, 2 y 6, la máquina 4 no procesa ningún trabajo por lo que también queda libre la máquina 5 puede procesar los trabajos 2 y 3 únicamente y así sucesivamente hasta llegar a la máquina 14 que solo procesa el trabajo 3.

Ahora bien, para calcular el makespan se procede a programar en un diagrama de Gantt lo que se muestra en el arreglo matricial del cromosoma, pero haciendo una adecuada modificación del arreglo matricial solo para efectos de visualización y determinar que operaciones se hacen primero y cuales se hacen después, así para la instancia de la tabla anterior la adecuación consiste en ordenar en una sola fila los vectores columnas del cromosoma y como son 8 columnas habrá 8 porciones del vector fila que determinarán el orden de las operaciones tales como: 6 1 2 1 3 4 2 1 3 5 8 7 2 3 – 2 2 1 2 2 7 3 2 1 1 7 1 1 1 – 1 3 6 3 1 1 1 3 2 2 1 2 3 2 – …, se omiten por que los tiempos de operación de esos trabajos son cero; ya con este orden y respetando la precedencia de las operaciones se traza una línea de tiempo y se procede a construir el diagrama de Gantt como se muestra en la tabla 9 que sigue:

Tabla 8: Cromosoma matricial de orden 8x14

	Secuencia de Trabajos (permutación)							
M1	6	2	1	3	4	5	7	8
M2	1	2	3	4	5	6	7	8
M3	2	1	6	3	4	5	7	8
M4	1	2	3	4	5	6	7	8
M5	3	2	1	4	5	6	7	8
M6	4	7	1	2	3	5	6	8
M7	2	3	1	4	5	6	7	8
M8	1	2	3	4	5	6	7	8
M9	3	1	2	4	5	6	7	8
M10	5	1	2	3	4	6	7	8
M11	8	7	1	2	3	4	5	6
M12	7	1	2	3	4	5	6	8
M13	2	1	3	4	5	6	7	8
M14	3	1	2	4	5	6	7	8

Tabla 9: Diagrama de Gantt construido a partir del cromosoma, (makespan = 30), abscisas unidades de tiempo



El primer 6 que aparece a la izquierda significa que la primera operación se inicia con el trabajo 6, éste trabajo nos conduce a la tabla 1 para ver con que máquina se inicia el trabajo 6 y observamos que éste trabajo se inicia en la máquina 1 y que el tiempo de procesamiento es de 120 minutos que se observa en la misma tabla; esta información hace que se trace en el diagrama de Gantt como un rectángulo en la primera fila o sea la máquina 1 con una duración de 120 minutos (en el diagrama cada espacio es de 10 minutos); así sucesivamente con el resto de números 1, 2, 1 etc. los números pintados de rojo indican que ya no hacen

falta porque el trabajo ya ha concluido de acuerdo a la secuencia establecida. Obsérvese que el makespan de la instancia cromosoma es de 300 minutos.

La calidad de cada individuo (Aptitud), se evalúa después de que a todos los individuos se les calculó su makespan, y este queda determinado por la fórmula:

$$F_i = \max(M_1, M_{2,} \dots, M_N) - M_i \quad (1)$$

En esta ecuación, M_i representa el makespan de cada individuo en una determinada generación y F_i representa la calidad de cada individuo

Con esta codificación el algoritmo genético implementado inicia su ejecución con la generación aleatoria de los individuos, su evaluación de la aptitud y guardando el mejor individuo en función de la aptitud; posteriormente se implementó el proceso de selección por torneo, se prosigue con la etapa de reproducción por emparejamiento parcial con un porcentaje de cruza del 95%, a tal efecto se generan dos números aleatorios por cada máquina y el intercambio genético se realiza entre dos individuos distintos emparejando máquinas iguales; luego en la etapa de mutación se implementó simultáneamente dos procesos para que el 50% de los individuos a mutarse se realice por inversión y el otro 50% por intercambio utilizando dos puntos de referencia para cada máquina y paralelamente durante la ejecución del algoritmo el porcentaje de mutación cambia, iniciándose con 2% desde la primera generación hasta la cuarta parte del total, luego cambia a 3% hasta llegar a la mitad de las generaciones y aquí se vuelve a cambiar al 4% hasta llegar al 60% de las generaciones, donde a partir de allí permanece constante en 5%. Al finalizar la etapa de mutación se evalúa el makespan y el fitness de cada individuo de la población en la generación correspondiente y finalmente se busca el peor individuo y es reemplazadlo por el mejor individuo encontrado de la generación anterior (Elitismo); este proceso es iterativo hasta que se cumpla con todas las generaciones. La salida del algoritmo es un diagrama de Gantt en el que se muestra todas las máquinas en el lado izquierdo y en el lado derecho el trabajo que realiza la máquina en cada unidad de tiempo para terminar con el diseño y construcción del algoritmo este está dotado con una funcionalidad para generar aleatoriamente problemas JSSP.

COMPORTAMIENTO DEL ALGORITMO

El algoritmo para la instancia del ejemplo 8 trabajos y 14 máquinas, se inicia con la primera generación encontrándose al mejor individuo con un makespan de 30, y luego de 3 generaciones se alcanza el valor optimo con un makespan de 28 (equivalente a 280 minutos) y a partir de allí se mantiene hasta el final de las 40 generaciones, cómo se puede apreciar en la figura 4. El tiempo de ejecución del algoritmo fue de 42 segundos y el diagrama de Gantt que reporta el algoritmo se puede apreciar en la tabla 10 y el individuo en términos de cromosoma se muestra en la tabla 11.

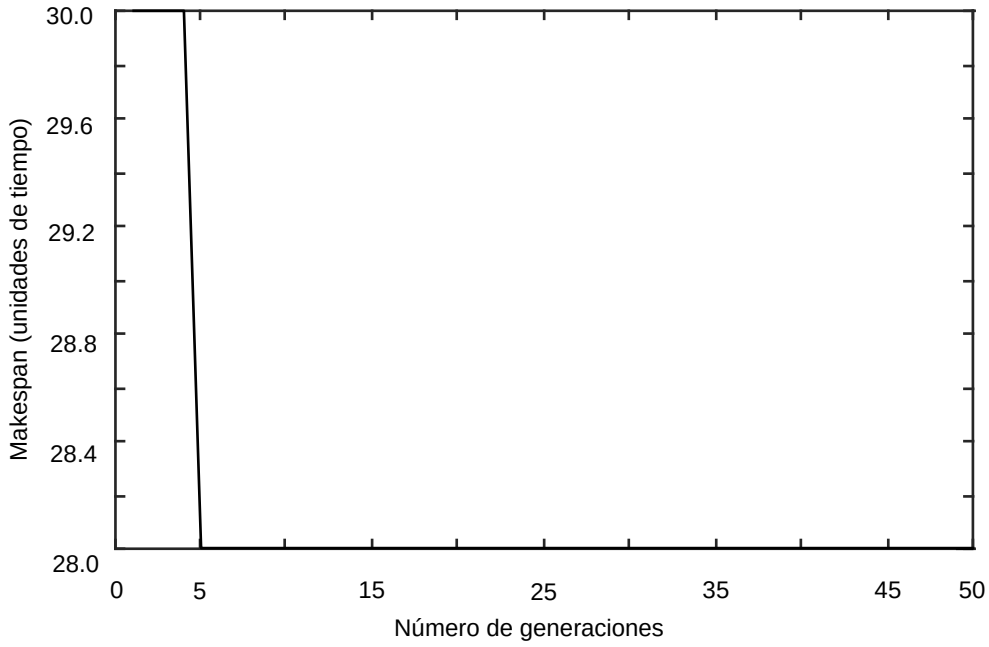


Fig. 4: Comportamiento del algoritmo genético simple con la instancia 8x14

Tabla 10: Diagrama de Gantt de la solución óptima del AG, conforme lo reporta el algoritmo

M01	6	6	6	6	6	6	6	6	6	6	6	6	2	2	2	2	2	2	2	2	2	2	2	1	1	1	0
M02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M03	0	0	0	0	0	0	0	0	0	0	0	0	6	6	6	0	0	0	0	0	0	0	0	2	2	2	1
M04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M05	0	0	0	0	0	0	2	2	0	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M06	4	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M07	2	2	2	3	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M09	0	0	0	0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	3	0	0	0	0	0	0	0	0
M10	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M11	8	8	0	0	0	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M12	0	0	0	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M13	0	0	0	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M14	0	0	0	0	0	0	3	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Makespan: 28																											

Tabla 11: Representación matricial del cromosoma del mejor individuo

	Trabajos								
Máquinas	1	8	2	1	7	6	5	4	3
	2	6	7	2	5	1	8	4	3
	3	4	6	1	3	8	5	7	2
	4	2	8	1	6	7	3	5	4
	5	8	4	6	5	7	3	1	2
	6	6	2	1	4	7	8	5	3
	7	8	4	3	2	1	7	5	6
	8	4	5	1	8	3	6	2	7
	9	4	6	1	7	3	2	8	5
	10	3	5	8	6	2	4	1	7
	11	6	4	2	3	7	1	8	5
	12	6	3	5	2	1	8	7	4
	13	6	4	5	8	2	7	1	3
	14	3	2	4	1	5	7	8	6

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

En el proceso de calibración del algoritmo genético, primero se resolvieron sendos ejercicios de instancias JSSP con tiempos de operación entre 0 y 10 unidades de tiempo, que fueron factibles de usar el software WinQSB, para tenerlo como parámetro de referencia y luego se comparó los resultados obtenidos con los proporcionados por el algoritmo genético y se observó coincidencias aun variando el porcentaje de cruza entre 92% a 98%; lo mismo se hizo con el porcentaje de mutación que se hizo variar de 1% al 5% y también los resultados fueron coincidentes es decir siempre encontró la solución óptima. Con estos resultados se decidió que el porcentaje de reproducción o cruza debería ser 95% y constante y el de mutación variable de 2% al 5% en función del número de generaciones. Otros factores que influyeron en los resultados de calibración fueron que el mecanismo de selección por torneo superó al mecanismo de selección por ruleta quedándose implementado el primero y con respecto a la etapa de mutación se implementaron dos métodos de mutación por intercambio y por inversión, de manera que el individuo a mutarse elegía el 50% de las veces uno de los métodos y el otro 50% elegía el otro método.

El análisis preliminar de otras instancias de prueba, con tiempos de operación entre 0 y 10 unidades de tiempo y tamaño de problema hasta 20 trabajos y 20 máquinas encuentra soluciones óptimas ejecutando el algoritmo con un tamaño de población (TP) de 100 individuos y el número de generaciones como los que se indican en la columna número de generaciones (NG) de la tabla 12; obsérvese que en todos los casos el número de generaciones (NG) son menores a 500; el tiempo máximo de ejecución del algoritmo con 500 generaciones es de 74.32 segundos.

Tabla 12: Resultados de instancias aleatorias con tiempo de operación entre 0 y 10 unidades de tiempo y TP=100.

Otras Instancias	Nombre	Tamaño	Makespan			Tiempo (seg) del A.G
			NG	Algoritmo	Modelo Matemático	
1	JSSP8T9M	8X9	47	78	78	2.90
2	JSSP13T15M	13X15	110	124	124	7.35
3	JSSP10T12M	10X12	50	99	99	4.50
4	JSSP15T20M	15X20	260	159	159	11.18

Tabla 12 *(continuación)*

5	JSSP10T10M	10X10	55	91	91	3.83
6	JSSP11T12M	11X12	190	108	108	4.82
7	JSSP20T15M	20X15	470	147	147	11.99
8	JSSP13T13M	13X13	250	118	118	6.25
9	JSSP15T15M	15X15	255	150	150	8.32
10	JSSP9T13M	9X13	32	98	98	4.27
11	JSSP11T12M	11X12	170	102	102	4.95
12	JSSP9T7M	9X7	35	69	69	2.53
13	JSSP14T14M	14X14	175	124	124	7.50
14	JSSP17T17M	17X17	60	154	154	10.84
15	JSSP15T8M	15X8	40	104	104	10.04
16	JSSP7T14M	7X14	98	84	84	3.65
17	JSSP20T20M	20X20	184	184	184	15.42
18	JSSP18T14M	18X14	450	153	153	9.72
19	JSSP11T11M	11X11	48	96	96	4.49

En el análisis integral cuando el AGS, se enfrenta a problemas con tiempos de operación entre 0 y 100 unidades de tiempo, fue necesario determinar nuevos parámetros Tamaño de Población (TP) y Número de generaciones (NG) para la ejecución del algoritmo, para ello se hizo uso del análisis de varianza en disposición factorial de esos dos factores con tres niveles diferentes cada uno conforme se muestra en la tabla 13, donde se observa que existe una alta significación tanto en el factor tamaño de la población como en el factor número de generaciones, sin embargo, la interacción de los dos factores no muestra diferencias significativas mostrándose estadísticamente iguales con un coeficiente de variabilidad del 1.21% respecto de la media.

Tabla 13: Resultados de la ejecución del algoritmo y ANVA para la instancia WEB15x15-1.sec

Factor	Factor	Repeticiones				Total	Promedio
Tp	Ng	I	II	III	IV		
100	500	1282	1311	1304	1257	5154	1288.50
100	1000	1293	1270	1271	1298	5132	1283.00
100	1500	1271	1263	1245	1265	5044	1261.00
200	500	1332	1331	1341	1306	5310	1327.50
200	1000	1295	1307	1297	1304	5203	1300.75
200	1500	1281	1295	1344	1300	5220	1305.00
300	500	1335	1330	1332	1328	5325	1331.25
300	1000	1327	1337	1327	1337	5328	1332.00
300	1500	1333	1322	1301	1337	5293	1323.25
	Total	11749	11766	11762	11732	47009	1305.81
Anva							
Fuente De Variación			Gl	Sc	Cm	Fc	Sig
Tp			2	16312.06	8156.03	32.88	**
Ng			2	2248.22	1124.11	4.53	**
Tpxge			4	1287.11	321.78	1.30	
Error			27	6698.25	248.08		
Total			35	26545.64			
			Cv =	1.21%			

Estos resultados conducen a un análisis más fino para determinar estadísticamente los niveles de los factores en estudio TP y NG para los cuales el AGS funciona correctamente, y se encontró mediante la comparación de promedios de Duncan tabla 14, a un nivel de significación del 0.05, que el nivel 100 del factor tamaño de población tiene una ventaja estadística significativa lográndose un makespan promedio menor de 1277.50 con respecto a los demás niveles es decir 200 y 300 cuyos promedios fueron superiores tales como 1311.08 y 1328.83 respectivamente lo cual explica que aunque se incremente el TP, el algoritmo no mejorará el makespan; con respecto al factor número de generaciones el efecto es inverso mientras mayor es el nivel del factor en este caso 1500, se logra estadísticamente el mejor makespan 1296.42 y es menor que los niveles 500 y 1000 cuyos makespan promedio fueron de 1315.75 y 1305.25 respectivamente. Por tanto, los niveles TP=100 y NG =1500 son los que se recomiendan durante la ejecución del algoritmo para instancias grandes con tiempos de operación entre 0 y 100 unidades de tiempo.

Tabla 14: Análisis de Duncan a un nivel de significancia de 0.05, y 27 grados de libertad del cuadrado medio del error, sobre el mejor makespan encontrado al JSSP Instancia 15x15-1. (Promedios con la misma letra son iguales estadísticamente en caso contrario son diferentes.)

CLAVE	TP	Promedio
A1	100	1277.50 a
A2	200	1311.08 b
A3	300	1328.83 c
CLAVE	NG	Promedio
B1	500	1315.75 b
B2	1000	1305.25 ab
B3	1500	1296.42 a
	CV =	1.21%

Determinados los niveles apropiados TP=100 y NG=1500, se ejecutó el AGS a todas las muestras del repositorio web sin embargo como ejemplo se muestra gráficamente la evolución de la solución de la instancia WEB15x15-1; en la figura 5 se puede ver que se inicia el mejor individuo en la primera generación con un makespan 1479 y en las primeras 111 generaciones baja drásticamente simulando una curva exponencial negativa llegando el makespan hasta 1310, luego de la generación 112 hasta la generación 158 sigue bajando el makespan pero con menor intensidad llegando su makespan a 1281, luego en la generación 159 baja hasta 1279 manteniéndose en ese estado hasta la generación 488, en la siguiente generación hasta la 547 el makespan baja a 1276, a partir de aquí sigue bajando de manera progresiva hasta la generación 642 llegando el makespan a 1261 y a partir de allí se mantiene estable hasta la generación 1500.

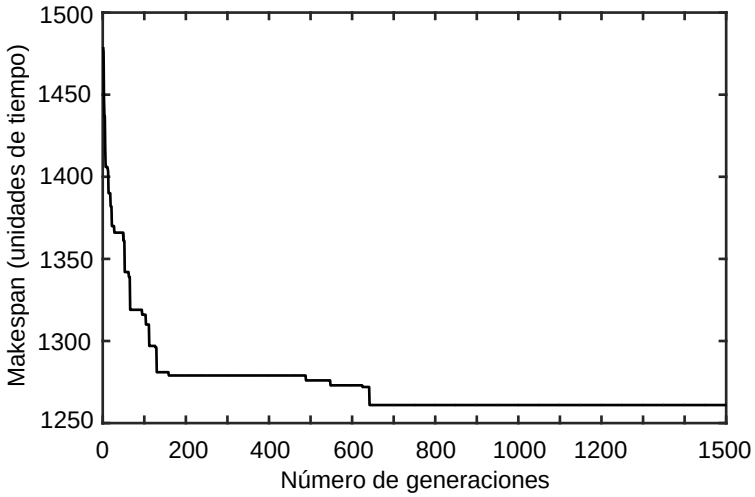


Fig. 5: Comportamiento de la ejecución del AGS para el ejemplo WEB15x15-1.sec

En la tabla15, se muestra el mejor individuo que da origen el makespan de 1261. Este comportamiento a juicio del autor está influenciado fuertemente por la etapa de mutación puesto que en la etapa de calibración el mecanismo de mutación se mejoró al ejecutarse dos procesos de mutación diferentes uno a la vez, una mutación por intercambio y otra mutación por inversión; y del mismo modo el mecanismo de selección por torneo superó al mecanismo de selección por ruleta. Por los resultados obtenidos se considera que es una contribución al problema JSSP, desde el punto de vista de la implementación de un algoritmo genético simple.

Tabla 15: Cromosoma del mejor individuo en la ejecución del AGS para la instancia WEB15x15-1.sec

	Trabajos															
Máquinas	1	07	15	10	11	06	08	02	12	14	05	09	13	04	01	03
	2	12	03	07	02	13	14	10	08	06	04	15	09	01	11	05
	3	07	11	05	13	09	14	10	06	08	15	02	12	04	03	01
	4	01	04	03	10	12	15	11	09	08	02	06	13	05	07	14
	5	12	05	06	07	15	08	04	02	10	13	01	14	03	09	11
	6	10	06	07	15	05	09	04	03	01	08	14	13	11	12	02
	7	04	02	14	08	11	12	07	13	09	01	03	05	15	10	06
	8	07	01	03	02	08	05	14	13	09	06	04	11	10	12	15
	9	12	01	15	07	06	13	14	10	08	05	11	09	02	04	03
	10	12	08	10	15	04	07	14	11	06	09	02	03	01	13	05
	11	05	09	12	10	07	14	03	04	02	11	01	15	13	06	08
	12	01	04	07	08	15	10	13	09	11	03	05	14	06	02	12
	13	06	15	13	11	08	02	03	01	14	12	04	10	05	07	09
	14	08	02	01	09	07	10	05	04	11	03	14	15	06	12	13
	15	14	09	05	13	11	08	15	12	02	04	10	03	06	01	07

La tabla 16, muestra los resultados de la aplicación del AGS a los cinco ejemplos del repositorio web cuya característica es que los tiempos de operación están entre 0 y 100 unidades de tiempo, repitiendo el algoritmo en cada instancia 10 veces con los parámetros definidos anteriormente, el tiempo promedio de ejecución del algoritmo fue de 7.12 minutos. En este caso se puede observar que el AGS proporciona en el mejor de los casos soluciones con un makespan en promedio 2.1867% superiores a los más encumbrados algoritmos híbridos del repositorio web, cuando se les compara con el Upper bound makespan ver ecuación (2), algoritmos híbridos como los mencionados por (Applegate y Cook, 1991). Los resultados obtenidos por el AGS propuesto no se pueden comparar con los reportados por (Castrillón et al., 2016) en razón a que las instancias de prueba fueron 4 trabajos y 5 máquinas con tiempos de operación menores a 30 unidades de tiempo tampoco por los resultados reportados por (Castrillón et al., 2011) en este caso la instancia del problema fue de 20 trabajos y 15 máquinas con tiempos de operación menores a 34 unidades de tiempo y no se conoce la secuencia de las operaciones.

Tabla 16: Resultados del AGS para Instancias de prueba del repositorio web: https://goo.gl/Gk9Vxt

Otras instancias del repositorio web		Tamaño	Makespan del AGS (10 repeticiones)					Makespan Upper bound en la web (B)
			Promedio	Desv. estándar	Min (A)	Max	Tiempo Minutos	
1	WEB15x15-1	15x15	1273.80	7.51	1263	1288	7.12	1231
2	WEB15x15-2	15x15	1279.30	12.27	1259	1298	7.12	1244
3	WEB15x15-3	15x15	1257.40	12.17	1232	1277	7.12	1222
4	WEB15x15-4	15x15	1227.90	14.96	1202	1248	7.12	1181
5	WEB15x15-5	15x15	1315.20	15.34	1291	1340	7.12	1233

$$\frac{1}{5}\sum_{n=1}^{n=5}\left(\frac{A-B}{B}\right)(100) = 2.1867\% \quad (2)$$

CONCLUSIONES

De los resultados mostrados, análisis y de su discusión, se obtienen las siguientes conclusiones, sobre la implementación del algoritmo genético simple: 1) El algoritmo genético simple resuelve el problema de Programación de la Tienda de Trabajo (Job Shop Scheduling), en todas las pruebas realizadas para instancias con tiempos de operación entre 0 y 10 unidades de tiempo, proporcionando soluciones óptimas. 2) Las soluciones del AGS para problemas con tiempos de operación entre 0 y 100 unidades de tiempo, son altamente competitivas tanto en el makespan como en el tiempo de ejecución del algoritmo al ser comparadas estadísticamente con las soluciones de otros autores, aun cuando el tiempo de ejecución no están registrados en la web se cree que son superiores a los del AGS implementado; 3) la estrategia de una selección por torneo, unido a un porcentaje de mutación progresivo y el diseño dual de dos mecanismos de mutación mantienen una diversidad de individuos que evitan que el algoritmo se quede atrapado en un óptimo local; 4) el AGS utilizado en las instancias de prueba analizadas puede ser ampliada a otras instancias del JSSP, previa calibración de sus parámetros.

REFERENCIAS

Applegate, D. y W. Cook, A Computational Study of the Job-Shop Scheduling Problem, doi: 10.1287/ijoc.3.2.149, ORSA Journal on Computing, 3(2), 149-156 (1991)

Bastidas, M.J., R.F. Bermúdez, G.P. Jaramillo y F. Chejne, Optimización Termoeconómica y Ambiental usando Algoritmos Genéticos Multiobjetivo, doi: 10.4067/S0718-07642010000400006, Inf. Tecnol., 21(4), 35-44 (2010)

Bażewicz, J., E. Pesch y M. Sterna, The Disjunctive Graph Machine Representation of the Job Shop Scheduling Problem, doi: 10.1016/S0377-2217(99)00486-5, European Journal of Operational Research, 127(2), 317-331 (2000)

Bhatt, N. y N.R. Chauhan, Genetic Algorithm Applications on Job Shop Scheduling Problem: A review, doi: 10.1109/ICSCTI.2015.7489556, En 2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI) (pp. 7-14) (2015)

Castrillón, O. D., W.A. Sarache y J.A. Giraldo, Aplicación de un Algoritmo Evolutivo en la Solución de Problemas Job Shop-Open Shop, doi: 10.4067/S0718-07642011000100011, Inf. Tecnol., 22(1), 83-92 (2011)

Castrillón, O.D., S. Ruiz-Herrera y W. Sarache, Programación de un Sistema Job Shop-Open Shop por medio de una Red Neuronal, doi.org/10.4067/S0718-07642016000500018, Inf. Tecnol., 27, 163–170 (2016)

Chen, Y., Z.-Z. Li y Z.-W. Wang, Multi-agent Based Genetic Algorithm for JSSP, doi: 10.1109/ICMLC.2004.1380676, Proceedings of International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), Machine Learning and Cybernetics, Proceedings of International Conference on, Machine learning and cybernetics, 267 (2004)

Cheng, R., M. Gen e Y. Tsujimura, A tutorial Survey of Job-Shop Scheduling Problems using Genetic Glgorithms-I. representation, doi: 10.1016/0360-8352(96)00047-2, Computers & Industrial Engineering, 30(4), 983-997 (1996)

Cortés-Rivera, D., Un Sistema Inmune Artificial para resolver el problema del Job Shop Scheduling, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional Departamento de Ingeniería Eléctrica Sección de Computación, México DF (2004)

Delgado, M., J. Eduardo y otros seis autores, Algoritmo Genético aplicado al problema de Programación en Procesos Tecnológicos de maquinado con ambiente Flow Shop, Revista Ciencias Técnicas Agropecuarias, 21(2), 70-75 (2012)

García Martínez, C., Algoritmos Genéticos Locales (Tesis Doctoral). Escuela Técnica Superior de Ingeniería Informática, Granada, España (2008)

Garey, M. R. y D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, 1st Edition, 340p., W.H. Freeman, New York (1979)

Gholami, O. e Y. N. Sotskov, A fast Heuristic Algorithm for Solving Parallel-Machine Job-Shop Scheduling Problems, doi: 10.1007/s00170-013-5281-6, International Journal of Advanced Manufacturing Technology, 70(1-4), 531-546 (2014)

Jiménez-Carrión, M., Los Algoritmos Genéticos desde la Investigación de Operaciones, 1ra Edición, 111-133, Editorial Académica Española (2016)

Liang, S., W. Xiao-Yuan y Z. Yun-Cai, Heuristics Algorithms for Job-Shop Scheduling Problem: Critical Element Analysis, International Conference on Computer Science & Service System, 94 (2012)

Lopez, J.C., J. A. Giraldo y J. A. Arango, Reducción del Tiempo de Terminación en la Programación de la Producción de una Línea de Flujo Híbrida Flexible (HFS), doi.org/10.4067/S0718-07642015000300019, Inf. Tecnol., 26, 157–172 (2015)

Luo, Y., Nested Optimization Method Combining Complex Method and Ant Colony Optimization to solve JSSP with complex associated processes, doi: 10.1007/s10845-015-1065-1, J.l. of Intelligent Man, 28(8), 1801-1815 (2017)

Salazar-Hornig, E. y S.J.C. Medina, Minimización del Makespan en Máquinas Paralelas Idénticas con Tiempos de Preparación Dependientes de la Secuencia utilizando un Algoritmo Genético. Makespan Minimization for The Identical Machine Parallel Shop with Sequence Dependent Setup Times Using a Genetic Algorithm (English), doi: 10.1016/S1405-7743(13)72224-8, Ingeniería, Investigación y Tecnología, 14, 43-51 (2013)

Zhang, G., L. Gao e Y. Shi, An effective Genetic Algorithm for the flexible Job-Shop Scheduling Problem, doi: 10.1016/j.eswa.2010.08.145, Expert Systems with Applications, 38(4), 3563-3573 (2011)