

Aplicación de un Algoritmo Genético Simple con Reparación en un Caso Real de Planeación de Producción Discreta: Fase I

Sergio Manuel Ramírez Campos
sramirez@its.mx

Xicoténcatl Jiménez Díaz
xicojimenez@prodigy.net.mx

Mónica López Arteaga
monyckla@hotmail.com

Korina Lyliana Contreras Iñiguez
korilyli@hotmail.com

*Instituto Tecnológico de Saltillo
Blvd. Venustiano Carranza 2400
Departamento de Posgrado de Ingeniería Industrial
Saltillo Coahuila, México*

Resumen

La planeación de la producción de guantes de látex que contempla una amplia variedad de presentaciones y tamaños presenta las características de un problema combinatorio en el que es prácticamente imposible enumerar todas las soluciones y en ocasiones es difícil de encontrar una sola de ellas bajo ciertas consideraciones. En el caso que nos ocupa se tiene una demanda variable y conocida de cada producto y el proceso a seguir para cada uno puede diferir dada la función del guante que puede ser desde un uso doméstico hasta un uso industrial. Ello está ligado a un número limitado de tanques de inmersión y cuyo contenido depende del producto que se encuentre en proceso. Además la línea solo puede mover un número limitado de contenedores para procesar producto, un inventario limitado de tales contenedores que son diferentes para cada producto y un número variable de moldes de guantes por dispositivo que depende de la talla del guante. La metaheurística desarrollada, ha permitido obtener soluciones factibles que satisfacen las expectativas planteadas en la Fase I: obtener un programa de producción factible atractivo considerando un número bajo de ciclos.

Palabras clave: problema combinatorio, metaheurística, AGS, reparación.

Introducción

Los problemas de programación y secuenciación de la producción en un conjunto de máquinas recibe considerable atención en la literatura. La decisión de qué tareas deben asignarse a las máquinas (y en qué orden) forma parte de un problema combinatorio casi siempre muy complicado y que frecuentemente es *NP* complejo. De acuerdo a Oduguwa et al.¹ la mayoría de los problemas que se encuentran en el mundo real generalmente son a gran escala, presentan no linealidades y alta incertidumbre, son multi-dimensionales e interaccionan con ingenieros y operadores muy especializados que controlan los procesos.

También se pueden encontrar en la literatura una gran diversidad de aplicaciones de los AG. Enseguida se mencionan algunas de ellas.

En el campo de los procesos de modelado rápido, Byun y Lee² proponen un algoritmo que determina la mejor orientación de un prototipo a ser fabricado considerando la calidad del mismo, la estructura que contiene la parte y el tiempo de fabricación, logrando así, una

planeación eficiente. Defersha y Chen³ desarrollaron un modelo híbrido que contempla un AG y el método Simplex de programación lineal para resolver problemas no lineales y variables no enteras en problemas reales *NP* complejos. Wong y Leung⁴ presentan un modelo de toma de decisiones genético para reducir el costo total de fabricación de ropa donde el 50% de dicho costo está conformado por el material. Los pedidos varían debido a la cantidad, tamaño y color por lo que se requiere determinar una adecuada planeación de los cortes sujeta a las restricciones impuestas por la clase de material, el equipo necesario y la mano de obra que cada prenda demanda. Boyd y Savory⁵ describen un AG para la programación del personal multihábil de un laboratorio químico. Para cada persona consideran sus habilidades o competencias, el turno que desea trabajar, un periodo de trabajo preferido y el número de días de permanencia en un tipo de trabajo antes de pasarlo a otro. Sus resultados revelan ahorros significativos en mano de obra.

Hou et al.⁶ analizan el caso de la programación de los ordenadores múltiples paralelos (multiprocesadores). El caso es *NP* complejo. Toman en cuenta el orden en que están realizándose las tareas en cada ordenador siendo el objetivo minimizar la longitud del programa. Bierwirth⁷ atiende el problema de programación en sistemas tipo *job shop* generalizando el operador de cruzamiento *OX* (llamándolo *GOX*) para obtener un algoritmo más eficiente. Della Croce et al.⁸ introducen una nueva codificación en un AG, basada en reglas de preferencia y un paso de actualización, acelerando así, el método evolutivo. Comparan con las técnicas de ramificación y acotamiento en problemas tipo Flow Shop superándolas significativamente. Ishibushi y Murata⁹ proponen un algoritmo para encontrar un conjunto de soluciones no dominadas de un problema de optimización con objetivos múltiples. Encuentran un alto desempeño del algoritmo en problemas de programación tipo flow shop. Iranmanesh et al.¹⁰ proponen un método para encontrar el frente óptimo de Pareto al problema de llevar a cabo un proyecto sujeto a la tripleta tiempo, costo y calidad. Desarrollan un AG especialmente adaptado el cual prueban con un problema de 30 actividades plasmadas en una red CPM.

En este artículo se presenta una metaheurística para un problema encontrado por los autores en la fabricación de guantes de látex. Los guantes se forman utilizando moldes de porcelana (manos izquierda y derecha) llevándolos a través de un proceso o ciclo en el que se le agregan capas sucesivas de diversas mezclas químicas introduciendo los moldes en tanques de inmersión, además de someterlos a un secado. La línea en cuestión consiste de un transportador en el cual se montan platos que contienen barras. Para ello, en cada plato se insertan diez barras que contienen los moldes. Cinco barras contienen moldes izquierdos y las otras cinco contienen moldes derechos. Una vez que un plato es armado con las diez barras, es colocado en la línea para su proceso. La línea solo puede contener un número limitado de platos. Así, el número de moldes que puede contener una barra es variable ya que depende del tipo de guante y de su tamaño o talla.

Por otro lado, los platos no son iguales entre sí, ya que ello también depende del tipo de guante y de la talla. Entonces se tiene un inventario limitado de platos por *producto*. El término *producto* en este artículo se refiere a una cierta clase de guante y a una talla en particular.

Bajo estas condiciones se debe diseñar un programa eficiente para atender una demanda dada en un cierto periodo, es decir, determinar una secuencia de producción apropiada. Tal programa debe especificar la forma en que se atenderá cada producto en la línea en el menor tiempo posible primordialmente (o en el menor número de ciclos). Un programa eficiente debe reducir de manera significativa los síntomas típicos de una planeación inadecuada: inventarios excesivos de producto terminado y plazos largos de entrega al cliente.

La complejidad del problema expuesto más la consideración de objetivos múltiples en conflicto y la inclusión de otras restricciones aún no consideradas, revelan un problema combinatorio de interés para modelarlo con un enfoque evolutivo.

Este trabajo ha surgido del trabajo de campo realizado por los autores para apoyar el desarrollo de dos tesis en el programa de maestría en Ingeniería Industrial. Por cuestiones de confidencialidad se omite información específica de la empresa que ha hecho posible esta investigación. Además, este artículo es la primera parte de la misma que se denominó Fase I.

Información relevante del proceso

La línea en cuestión se caracteriza por ser un proceso continuo en el que los moldes son introducidos a un ciclo para que en el transcurso del mismo se le agreguen las mezclas y tratamientos requeridos y al finalizar se desmonten los guantes con las especificaciones del cliente.

En el caso considerado, la línea tiene capacidad para 31 platos (C_i) y normalmente se considera un factor de pérdidas (F_p) del 8% por lo que la demanda se incrementa en esta proporción. Las pérdidas son por diversas razones, incluyendo entre otras, aspectos de calidad.

Por otra parte, hay factores que ocasionan que la cantidad a procesar (C_p) de *producto* no sea igual a la cantidad esperada (C_e). Enseguida se mencionan.

- C_e no es necesariamente un múltiplo de la cantidad de *producto* que contiene un plato. Esta diferencia origina variaciones en la cantidad producida debido a que se tiene que decidir si se redondean al entero superior próximo ó al entero inferior próximo algunos valores relacionados con los platos a procesar (P_p). Así, la cantidad total de platos (P_i) para una demanda se calcula según la ecuación 1.

$$P_i = \left\lceil \frac{C_e(1+F_p)}{M_i(5)} \right\rceil \quad (1)$$

Donde M_i es el número de moldes por barra del *producto* i . La disyuntiva es la ya señalada: se tiene que decidir si P_i se redondea al entero superior próximo ó al inferior próximo. Entonces $1 \leq P_p \leq P_i$. En la Fase I, P_i se redondea al entero superior de manera arbitraria (en la Fase II se tiene planeado analizar esta decisión).

- El inventario de platos (I_p) disponible para un *producto* dado. Si $I_p < P_p$ obliga a programar más de un ciclo y si además, $I_p < C_i$ entonces el número total de ciclos estará dado por la ecuación 2.

$$C_i = \left\lceil \frac{P_i}{I_p} \right\rceil \quad (2)$$

Y nuevamente se tiene que decidir si C_i se redondea al entero superior próximo ó al entero inferior próximo. En este caso $P_p = I_p$.

Ahora, si $I_p \geq C_i$ entonces C_i estará dado por la ecuación 3.

$$C_t = \left[\frac{P_t}{C_t} \right] \quad (3)$$

Igualmente, se tiene que decidir si C_t se redondea al entero superior próximo ó al inferior próximo. En este caso $P_p = C_t$. En la Fase I, se calcula el faltante si C_t se redondea al entero inferior de acuerdo a la ecuación 4.

$$F_r = C_e - [C_t] P_p M_i \quad (4)$$

De manera similar, se calcula el excedente si C_t se redondea al entero superior de acuerdo a la ecuación 5.

$$E_r = [C_t] P_p M_i - C_e \quad (5)$$

Ahora, se selecciona la menor desviación, $V_r = \text{mínimo}\{F_r, E_r\}$ y si $V_r = F_r$ entonces C_t se redondea al inferior y en caso contrario, al superior.

Modelo diseñado en la Fase I

Adicionalmente se remite al lector a Ramírez et al.¹¹ y a Ramírez et al.¹² para una descripción del algoritmo genético simple que, aunque son otros contextos, explican con detalle los pasos del mismo y su aplicación. Para ahondar más en el tema se recomienda Goldberg¹³ y Darwin¹⁴.

Dado que el primer paso es la representación adecuada del cromosoma, primero se explica el significado de un gen. Debido a que uno o más *productos* utilizan una cierta configuración de tanques de inmersión, entonces, un gen será el conjunto de *productos* que tienen una misma configuración de tanques. Por lo tanto, un cromosoma estará compuesto por los genes necesarios para cubrir toda la demanda de un periodo. De aquí que la longitud de un cromosoma sea variable.

Cabe aclarar que una configuración de tanques se refiere a un cierto número (t) de tanques, con una cierta mezcla contenida en cada uno de ellos. A partir de la demanda se identifica un conjunto (c) de configuraciones de tanques requeridos.

Ahora, el siguiente paso es generar una población inicial factible de tamaño p dado que se tiene una demanda de n *productos*. Para ello se utiliza el siguiente algoritmo:

1. Se forma, al azar, la i -ésima configuración de tanques CT_i a partir del conjunto c donde $1 \leq i \leq n$. Hacer $k=1$.
2. Se selecciona la k -ésima demanda y se verifica si su CT_k es igual a la CT_i donde $1 \leq k \leq n$.
3. Si se cumple el paso 2 se continúa en el paso 5. En caso contrario hacer $k = k + 1$ e ir al paso 4.
4. Si $k \leq n$ continuar en el paso 2. En caso contrario se descarta la CT_i actual, se hace $i = i + 1$ y se continúa en el paso 1.
5. Se selecciona la k -ésima demanda como factible de programar y se elimina del conjunto original de demanda haciendo $n = n - 1$.
6. Si ya se seleccionaron n demandas, se detiene el algoritmo. De otra forma se continúa en el paso 2.

Una vez que se genera el número de soluciones deseado, se dispone de una población inicial compuesta de p individuos o soluciones factibles.

Enseguida se debe definir una función de aptitud conveniente que para el caso, mide el número de ciclos totales (ct_k) de la línea para atender la demanda en cuestión en la k -ésima solución. Por ello el siguiente paso es calcular ct_k considerando que:

- Se inicia con una configuración de tanques CT_i
- La línea tiene una capacidad limitada a C_l platos por ciclo.
- La demanda en pares de guantes por *producto* se transforma a una cantidad de platos a procesar (P_p).
- La cantidad de platos a procesar de un *producto* j en la línea en un ciclo dado c_i estará determinada por:
 - ✓ Los platos que están liberándose en la línea en un momento dado
 - ✓ El inventario de platos existente para el *producto* j
 - ✓ El número de moldes por barra del *producto* j
 - ✓ La configuración de tanques que requiere el *producto* j

En la sección de experimentación en el cuadro 3 se muestra un ejemplo detallado de este cálculo.

Una vez que ya se conoce el valor de ct_k para cada una de las p soluciones, éstas se ordenan de menor a mayor valor de ct_k y se identifica la mejor solución hasta este paso.

- a. Lo siguiente es la reproducción para lo cual se fija una tasa t_R la cual se recomienda entre 0.5 y 0.8¹⁵. Si la fijamos en 0.7 implica que el 70% de la población va a formar parejas para generar un hijo. Para llevar a cabo la reproducción, se considera una selección aleatoria de las parejas.

Después de seleccionar una pareja (P_1, P_2), se procede a su reproducción utilizando el siguiente algoritmo:

1. Se selecciona el k -ésimo gen donde $1 \leq k \leq l$ y l es la longitud menor de ambos padres.
2. Se genera un número entero aleatorio R tal que $1 \leq R \leq 2$.
3. Si $R=1$ se selecciona el gen k del padre P_1 , en caso contrario se elige el gen k del padre P_2 .
4. El gen seleccionado en el paso 3 constituye el k -ésimo gen del hijo.
5. Se hace $k = k + 1$ y si $k \leq l$ se continúa en el paso 1. En caso contrario termina el algoritmo.

Enseguida se inicia el proceso de reparación verificando si el hijo es factible o no. Para ello se comprueba si existen *productos* duplicados y en su caso los elimina. Luego si faltan *productos* los busca en ambos padres y los incorpora al hijo.

Se continúa seleccionando otras parejas hasta completar la tasa de reproducción establecida.

Ahora se procede a la mutación dada una tasa t_m cuyo valor se recomienda¹⁵ alrededor de 0.2. Si $t_m = 0.2$ implica que el 20% de la población actual se mutará. Una mutación se obtiene de acuerdo con el siguiente algoritmo:

1. Se genera un número aleatorio r_i tal que $0 < r_i < 1$

2. Se calcula la probabilidad de mutación de acuerdo con la ecuación 6.

$$P_m = \frac{1}{2} \left(\frac{1}{p} + \frac{1}{l} \right) \quad (6)$$

3. Si $r_i < P_m$ se lleva a cabo la mutación, de lo contrario no se efectúa y termina el algoritmo.
4. Si procede la mutación, se genera otro número entero aleatorio tal que $1 \leq r_i' \leq p$ para seleccionar la solución a mutar.
5. Se genera otro número entero aleatorio tal que $1 \leq r_i'' \leq l$ para identificar qué número de gen se va a mutar.
6. Dado que cada gen es el conjunto de *productos* que tienen una misma configuración de tanques y que están en un orden dado, entonces la mutación consiste en volver a "reordenarlos" al azar.
7. Si no se ha completado la tasa de mutación, se continúa en el paso 1. De lo contrario termina el algoritmo.

Sigue entonces la inserción para lo cual se establece una tasa de reemplazo t_r cuyo valor puede ser hasta el 100%¹⁶. Si $t_r = 0.50$ implica que la segunda mitad de la población actual ordenada (la peor) se reemplazará por la primer mitad de la nueva generación ordenada (la mejor).

La proporción de la nueva población a utilizar para reemplazar a la anterior está compuesta: (a) exclusivamente por nuevas soluciones o (b) por nuevas soluciones, soluciones de la población actual que no se reprodujeron y soluciones producto de una mutación¹⁶.

El proceso descrito en cada iteración va creando nuevas generaciones y se detiene cuando alcanza una meta establecida o satisface una regla de detención.

Experimentación y resultados

Se diseñó una interfase gráfica utilizando MatLab[®] versión 7. Se registró una demanda de 28 *productos* la cual debe ser satisfecha en el menor tiempo posible. En el cuadro 3 se muestran tanto los datos de entrada como de la salida del AGS y en el cuadro 2 se explica el encabezado de cada columna.

Antes de pasar a los resultados de la experimentación, se hará una explicación detallada de los pasos del algoritmo diseñado utilizando parte de los datos que se muestran en el cuadro 3. Para ello, en el cuadro 1 se muestran las mezclas que requiere cada producto demandado. Se observa que en algunos casos existen hasta 4 opciones para la capa 1 y para la capa 2 y en otros casos no se requiere capa 2. También, las mezclas requeridas son iguales para cualquier talla de un mismo tipo de producto. Por ello, solo se indica la información para el primer renglón de cada tipo de producto.

Para generar una población inicial, se genera una configuración de tres tanques al azar. Para ello, se debe tener más de una opción. Si hay una sola opción, se elige ésta. En caso contrario se genera un número aleatorio entero en el rango de posibles opciones. En el caso del cuadro 1 y el tanque 1 solo hay una opción (730001). En el caso del tanque 2 hay varias opciones, las cuales aparecen en las columnas 5, 6, 7 y 8. En el caso del tanque 3 también hay más de una opción (columnas 9, 10, 11 y 12). Se debe descartar una configuración donde sean iguales los contenidos de los tanques 2 y 3.

1	2	3	4	5	6	7	8	9	10	11	12
Producto			Tanque 1	Tanque 2 (capa 1)				Tanque 3 (capa 2)			
No.	Tipo	Talla	Coagulan- te	Opción 1	Opción 2	Opción 3	Opción 4	Opción 1	Opción 2	Opción 3	Opción 4
1	492	7	730001	730009	730023	730010		730013			
2	492	8									
3	492	9									
4	492	10									
5	492	11									
6	497	7	730001	730011	730016	730018		730013			
7	497	8									
8	497	9									
9	497	10									
10	487	6	730001	730009	730008	730023	730010				
11	487	11									
12	490	9	730001	730009	730008	730023	730010	730009	730008	730023	730010
13	490	10									
14	490	11									
15	491	6	730001	730009	730023	730010		730013			
16	491	7									
17	491	8									
18	491	9									
19	495	6	730001	730011	730016	730018		730018			
20	495	7									
21	495	8									
22	495	10									
23	485	6	730001	730093				730094			
24	485	7									
25	485	8									
26	485	9									
27	485	10									
28	485	11									

Cuadro 1. Configuración de tanques requerida

Una vez que se tiene una configuración de tanques formada al azar, se selecciona la primera demanda y se compara su configuración de tanques requerida con la configuración previamente conformada. Si es igual, pasa a formar parte del programa factible, en caso contrario, se descarta y se continúa con este proceso hasta que se terminan de comparar todas las demandas. Si existe al menos una demanda seleccionada, se ha formado un gen. Si ninguna demanda se seleccionó, se descarta la configuración formada al azar y se genera otra y se repite el proceso de ir formando los genes necesarios hasta que toda la demanda queda ubicada en alguno de ellos.

Una vez que se formó la población inicial deseada, se dispone de un programa factible considerando la configuración de tanques de inmersión que requiere cada demanda.

Enseguida se calculan los platos (P_p) y ciclos (C_i) por demanda a producir. En la figura 1 se muestran los pasos para ello.

Se procede a evaluar la aptitud de cada individuo (o programa factible) determinando el número total de ciclos necesarios. Para ello analizaremos el cuadro 3. Las columnas 6 y 7 son los valores del paso anterior: P_p y C_i . El primer *producto* es del tipo 497 y talla 10 y se está decidiendo producir 3600 pares (de acuerdo a la lógica de la figura 1). Considerando que la capacidad de la línea es de 31 platos, se pueden programar 20 platos (es el inventario de platos indicado en la columna 17) en los ciclos del 1 al 3 de dicho *producto*. Esto equivale a 60 platos procesados (20x3). Si $M_i = 12$ (columna 16), entonces la cantidad de guantes a producir está dada por la ecuación 7.

Por lo tanto, de los 31 platos ya están ocupados del 1 al 20 (columnas 11 y 12).

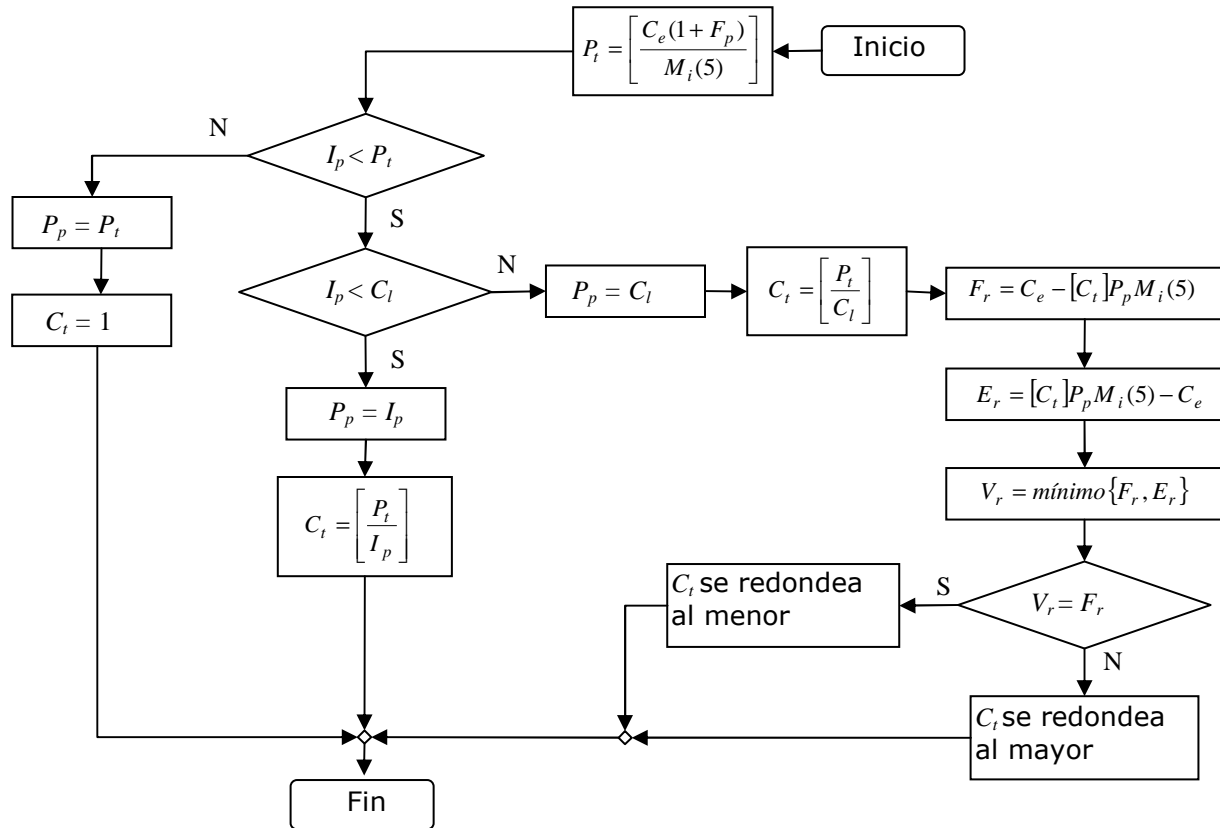


Figura 1. Lógica del cálculo de platos y ciclos por demanda.

$$C_p = M_i P_p(5) = 12(60)(5) = 3600 \text{ guantes} \quad (7)$$

La siguiente demanda considerada dentro del mismo gen (lo cual quiere decir que requieren la misma configuración de tanques) es del tipo 497 y talla 7. Haciendo un análisis similar, hay 18 platos disponibles (columna 17) pero con 15 es suficiente durante un solo ciclo: $14 \times 15 \times 5 = 1050$ (columna 6). Los 11 platos libres hasta el momento, son ocupados en el primer ciclo (columnas 7 a la 10) y los otros 4 platos en el segundo ciclo (del plato 1 al 4) ya que los 11 primeros quedan libres al inicio del segundo ciclo de la línea (quedando 7 libres).

La tercera demanda atendida es del tipo 497 y talla 8. Se ocupan los 7 platos libres a partir del segundo ciclo durante 3 ciclos (columnas 7 a la 10) y los platos ocupados son del quinto al onceavo (columnas 11 y 12). Los 4 platos ocupados un solo ciclo durante el segundo ciclo consecutivo de la línea quedan liberados a partir del tercer ciclo consecutivo de la línea. Debe hacerse notar que son los primeros platos en desocuparse. Así, se asignan del tercer al quinto ciclo consecutivo los platos 12 al 15. A partir del cuarto ciclo consecutivo se desocupan los 20 platos que se utilizaron por primera vez. Del plato 16 al 24 se asignan 9 platos durante tres ciclos (del cuarto al sexto ciclo consecutivo). Sumando 7, 4 y 9 platos nos da la cantidad de platos disponibles en inventario para esta demanda (columna 17).

El proceso continúa de manera similar con la salvedad de que cuando hay un cambio de configuración de tanques (otro gen), se debe esperar a que se terminen de procesar todos los platos de la configuración anterior. Esto sucede a partir de la demanda número 5.

Columna(s)	Significado
1, 2, 3, 4 y 5	Información de la demanda considerada.
6	Pares de guantes programados para procesar.
7	Número de platos a procesar en un intervalo de ciclos de la línea (los que se indican en las columnas 8, 9 y 10).
8	Número de ciclo en que comienzan a procesarse los platos indicados en la columna 6.
9	Número de ciclo en que terminan de procesarse los platos indicados en la columna 6.
10	Número de ciclos totales que se procesan los platos indicados en la columna 6.
11	Posición de inicio para ubicar los platos indicados en la columna 7.
12	Posición final de ubicación de los platos indicados en la columna 7.
13	Indica el cambio de configuración de tanques de un renglón a otro en la figura. Si es el mismo número no cambia, si no, si hubo cambio.
14	Variación en pares de guantes entre lo programado y lo solicitado (columna 6 menos columna 4).
15	Variación porcentual entre lo programado y lo solicitado (Columna 6/columna 4).
16	Moldes por barra que se permiten para el <i>producto</i> en turno.
17	Inventario disponible de platos para el <i>producto</i> en turno.

Cuadro 2. Explicación de las columnas del cuadro 3.

Para evaluar el desempeño del algoritmo, se efectuaron 100 corridas cada una de 15 iteraciones, con los siguientes parámetros:

$$p = 50 \quad t_R = 0.70 \quad t_m = 0.15 \quad t_r = 0.75 \quad C_l = 31$$

Se consideró un factor de pérdidas de 4%. La selección de parejas para la reproducción fue al azar y la función de aptitud quedó definida como:

$$F = \text{menor (número de ciclos)}$$

Adicionalmente, la regla de detención se activa cuando se acumulan 20 intentos consecutivos y en cada caso la diferencia entre los valores de la función de aptitud de una solución a otra es menor a 0.0005.

En la figura 2 se muestra el resultado de dichas corridas. Se puede apreciar que el comportamiento es consistente en la reducción del número de ciclos (lo cual equivale a reducir el tiempo de proceso).

Una vez que se sabe que el modelo es confiable se hizo una sola corrida considerando los mismos parámetros anteriores para evaluar un resultado típico el cual se muestra en la figura 3 y en el cuadro 3 observando que con 15 iteraciones es suficiente para el resultado deseado.

Respecto al cuadro 3 cabe resaltar lo siguiente:

- Muestra con detalle el número de platos a utilizar en la línea para cada *producto*
- Indica en qué ciclos se van a procesar los platos de cada *producto*
- Señala cuáles son las ubicaciones de cada plato en la línea

El número de ciclos totales es la cantidad mayor que se genera en la columna 9 que en este caso es 409. Finalmente, esta corrida consumió 18.95 segundos en una computadora Pentium(R) D, 3.00 GHz y 0.99 GB de RAM.

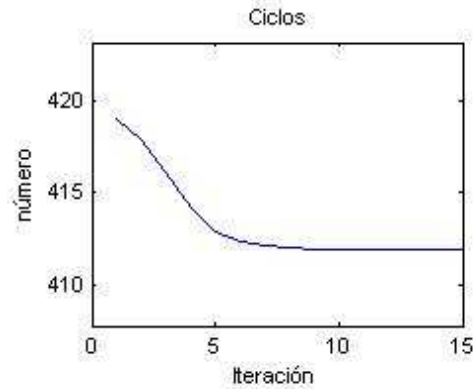


Figura 2. Desempeño promedio del algoritmo.

Demanda					Programación									Variación		Mol-des por barra	Platos en inven-tario
No.	Tipo	Ta-lla	Pares	Platos	Pares	Ciclos			Platos			*	pares	%	barra	tario	
						Pla-tos	de	a	Total	de	a						
1	497	10	3900	68	3600	20	1	3	3	1	20	0	-300	-7.7	12	20	
2	497	7	1000	15	1050	11	1	1	1	21	31	0	50	5.0	14	18	
						4	2	2	1	1	4	0	50	5.0	14	18	
3	497	8	4900	73	4200	7	2	4	3	5	11	0	-700	-14.3	14	20	
						4	3	5	3	12	15	0	-700	-14.3	14	20	
						9	4	6	3	16	24	0	-700	-14.3	14	20	
4	497	9	3700	60	4030	7	4	5	2	25	31	0	330	8.9	13	34	
						4	4	5	2	1	4	0	330	8.9	13	34	
						7	5	6	2	5	11	0	330	8.9	13	34	
						4	6	7	2	12	15	0	330	8.9	13	34	
						7	6	7	2	16	22	0	330	8.9	13	34	
						2	6	7	2	23	24	0	330	8.9	13	34	
5	492	8	73964	1099	76440	12	8	98	91	1	12	1	2476	3.3	14	12	
6	492	11	4704	109	4500	10	8	17	10	13	22	1	-204	-4.3	9	10	
7	491	6	1350	21	1260	9	8	9	2	23	31	1	-90	-6.7	14	9	
8	491	8	7450	111	7000	9	10	14	5	1	9	1	-450	-6.0	14	20	
						9	15	19	5	10	18	1	-450	-6.0	14	20	
						2	18	22	5	19	20	1	-450	-6.0	14	20	
9	491	7	650	10	700	8	18	18	1	21	28	1	50	7.7	14	18	
						2	19	19	1	29	30	1	50	7.7	14	18	
10	492	10	146360	2537	151500	1	19	119	101	31	31	1	5140	3.5	12	25	
						5	19	119	101	1	5	1	5140	3.5	12	25	
						9	20	120	101	6	14	1	5140	3.5	12	25	
						2	20	120	101	15	16	1	5140	3.5	12	25	
						2	23	123	101	17	18	1	5140	3.5	12	25	
						6	99	199	101	19	24	1	5140	3.5	12	25	
11	492	7	55092	819	57050	5	99	261	163	25	29	1	1958	3.6	14	5	
12	492	9	135884	2175	140400	1	99	206	108	30	30	1	4516	3.3	13	20	
						1	120	227	108	31	31	1	4516	3.3	13	20	
						5	120	227	108	1	5	1	4516	3.3	13	20	
						9	121	228	108	6	14	1	4516	3.3	13	20	
						2	121	228	108	15	16	1	4516	3.3	13	20	
						2	124	231	108	17	18	1	4516	3.3	13	20	
13	491	9	20150	323	20150	6	200	209	10	19	24	1	0	0.0	13	34	
						1	207	216	10	25	25	1	0	0.0	13	34	
						6	210	219	10	26	31	1	0	0.0	13	34	
						1	217	226	10	1	1	1	0	0.0	13	34	
						6	220	229	10	2	7	1	0	0.0	13	34	
						1	227	236	10	8	8	1	0	0.0	13	34	
						1	228	237	10	9	9	1	0	0.0	13	34	

Cuadro 3. Resultados típicos de una corrida (continúa en la siguiente página)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
Demanda					Programación								Variación		Mol-	Platos	
No.	Tipo	Ta-lla	Pares	Platos	Pares	Ciclos				Platos		*	pares	%	barra	por	en- tario
						Pla-tos	De	a	Total	de	a						
14	485	11	15768	365	16380	5	228	237	10	10	14	1	0	0.0	13	34	
15	485	10	48960	849	50400	4	229	238	10	15	18	1	0	0.0	13	34	
16	485	6	6696	100	6930	7	262	313	52	1	7	2	612	3.9	9	7	
17	485	8	5760	86	5600	20	262	303	42	8	27	2	1440	2.9	12	20	
18	485	9	114696	1836	118885	4	262	272	11	28	31	2	234	3.5	14	9	
19	485	7	4536	68	5040	4	273	283	11	1	4	2	234	3.5	14	9	
20	495	7	4600	69	5040	1	284	294	11	5	5	2	234	3.5	14	9	
21	495	6	12300	183	12600	3	284	287	4	6	8	2	-160	-2.8	14	20	
22	495	10	1100	20	1200	3	288	291	4	9	11	2	-160	-2.8	14	20	
23	495	8	4900	73	4200	3	292	295	4	12	14	2	-160	-2.8	14	20	
24	487	6	5900	88	5670	1	295	298	4	15	15	2	-160	-2.8	14	20	
25	487	11	3744	87	3780	3	296	299	4	16	18	2	-160	-2.8	14	20	
26	490	11	5328	124	5355	1	299	302	4	19	19	2	-160	-2.8	14	20	
27	490	9	4248	68	4030	3	300	303	4	20	22	2	-160	-2.8	14	20	
28	490	10	2160	38	2400	1	303	306	4	23	23	2	-160	-2.8	14	20	
						2	304	307	4	24	25	2	-160	-2.8	14	20	
						6	304	362	59	26	31	2	4189	3.7	13	34	
						12	304	362	59	1	12	2	4189	3.7	13	34	
						3	304	362	59	13	15	2	4189	3.7	13	34	
						1	307	365	59	16	16	2	4189	3.7	13	34	
						2	308	366	59	17	18	2	4189	3.7	13	34	
						7	314	372	59	19	25	2	4189	3.7	13	34	
						6	363	366	4	26	31	2	504	11.1	14	18	
						12	363	366	4	1	12	2	504	11.1	14	18	
						18	373	376	4	1	18	3	440	9.6	14	18	
						9	373	392	20	19	27	3	300	2.4	14	9	
						4	373	373	1	28	31	3	100	9.1	12	20	
						4	374	374	1	1	4	3	100	9.1	12	20	
						4	375	375	1	5	8	3	100	9.1	12	20	
						4	376	376	1	9	12	3	100	9.1	12	20	
						4	377	377	1	13	16	3	100	9.1	12	20	
						14	377	379	3	17	30	3	-700	-14.3	14	20	
						1	377	379	3	31	31	3	-700	-14.3	14	20	
						3	377	379	3	1	3	3	-700	-14.3	14	20	
						2	378	380	3	4	5	3	-700	-14.3	14	20	
						9	393	401	9	1	9	4	-230	-3.9	14	9	
						7	393	404	12	10	16	4	36	1.0	9	7	
						7	393	409	17	17	23	4	27	0.5	9	7	
						8	393	394	2	24	31	4	-218	-5.1	13	34	
						8	395	396	2	1	8	4	-218	-5.1	13	34	
						8	397	398	2	9	16	4	-218	-5.1	13	34	
						7	399	400	2	17	23	4	-218	-5.1	13	34	
						1	399	400	2	24	24	4	240	11.1	12	20	
						7	401	402	2	25	31	4	240	11.1	12	20	
						1	401	402	2	1	1	4	240	11.1	12	20	
						9	402	403	2	2	10	4	240	11.1	12	20	
						2	403	404	2	11	12	4	240	11.1	12	20	

Cuadro 3. Resultado típico de una corrida (continua de la página anterior).

Conclusiones

Los resultados obtenidos indican que el AGS con reparación diseñado satisface las expectativas de obtener un programa de producción factible y atractivo con un número de ciclos bajo. En el cuadro 2 se puede verificar con el detalle necesario, la secuencia de cómo se debe atender la demanda tanto en cuanto a ciclos como a platos. Es importante hacer notar que, aunque en la literatura abundan metaheurísticas en relación a problemas de secuenciación, no es fácil encontrar una que se adapte al escenario específico aquí considerado.

Trabajo futuro

En una segunda fase se tiene contemplado realizar una comparación exhaustiva para determinar la eficiencia de este algoritmo genético, además de analizar la conveniencia de incorporar otras restricciones del caso real. Así mismo, se planea mejorar el algoritmo

genético tomando en cuenta, entre otros aspectos, una función de aptitud multi-objetivo. Finalmente, se hará un análisis más amplio respecto a las desviaciones que hay entre C_e y C_p .

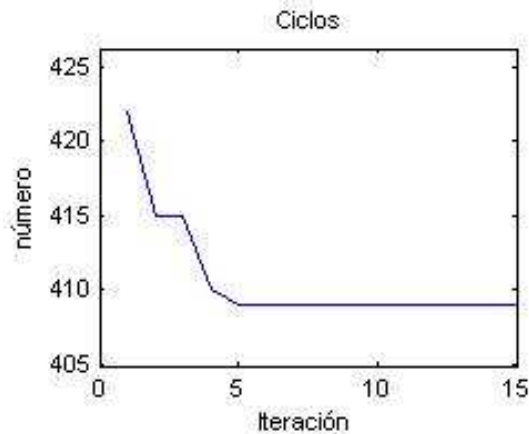


Figura 3. Evolución de una corrida típica.

Referencias

- ¹ Oduguwa, V., A. Tiwari, A. and R. Roy, 2004, "Evolutionary computing in manufacturing industry: an overview of recent applications". Enterprise Integration, School of Industrial and Manufacturing Science, Cranfield University, Bedford MK430AL, UK.
- ² Byun H.S. and K. H. Lee, "Determination of the optimal part orientation in layered manufacturing using genetic algorithm". International Journal of Production Research, Vol. 43 No. 13, 2005.
- ³ Fantahun, M. L. Defersha and M. Chen, 2007, "A linear programming embedded genetic algorithm for an integrated cell formation and lot sizing considering product quality". Concordia University, Department of Mechanical and Industrial Engineering, 1455 de Maisonneuve W. Montreal, Quebec, Canada H3G 1M8.
- ⁴ Wong, W.K. and S. Y. S. Leung, 2008, "Genetic optimization of fabric utilization in apparel manufacturing". Institute of Textiles and Clothing. The Hong Kong Polytechnic University, Hunghom, Kowloon, Hong Kong.
- ⁵ Boyd, J.C. and J. Savory, "Genetic Algorithm for Scheduling of Laboratory Personnel". *Clinical Chemistry*, Vol. 47 No. 1, 2001.
- ⁶ Hou, E. S. H., Ansari N. and Hong Ren, "A genetic algorithm for multiprocessor scheduling". *Parallel and Distributed Systems, IEEE Transactions on*. Vol. 5 No. 2, 1994.
- ⁷ Bierwirth, C., 1995, "A generalized permutation approach to job shop scheduling with genetic algorithms". OR Spektrum. Department of Economics, University of Bremen Postfach 330440 Bremen Germany.
- ⁸ Della Croce, F., R. Tadei and G. Volta, "A genetic algorithm for the job shop problem". *Computers & Operations Research*. Vol. 22 No. 1, 1995.

- ⁹ Ishibushi, H. and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling". *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*. Vol. 28 No. 3, 1998.
- ¹⁰ Iranmanesh, H., M. R. Skandari, and M. Allahverdiloo, "Finding Pareto Optimal Front for the Multi-Mode Time, Cost Quality Trade-off in Project Scheduling". *Proceedings of World Academic of Science, Engineering and Technology*. Vol. 30, 2008.
- ¹¹ Ramírez C., S., L. Torres-Treviño, G. Cedillo C. and J. Villegas L., "A Genetic Algorithm for the Reassignment of Work in an Assembly Line: A Real Scenario". *Research in Computer Science*, Vol. 23 No.1, 2006.
- ¹² Ramírez C. S., E. Marroquín y J. Córdova, "Reasignación de Tareas a Operarios en una Estación de trabajo". *Revista de la Ingeniería Industrial*, Vol. 1 No. 1, 2007.
- ¹³ Goldberg, D.E., 1989. "Genetic algorithms in search, optimization, and machine learning", *Addison-Wesley Pub Co*.
- ¹⁴ Darwin, C., "El origen de las especies", 1997. *UNAM*.
- ¹⁵ Jahangirian, M. and G.V. Conroy, "Intelligent dynamic scheduling system". *Integrated Manufacturing Systems*, Vol. 11 No. 4, 2000.
- ¹⁶ Badibaru, A. B. and J.Y. Cheung, "Fuzzy Engineering Expert Systems with Neural Network Applications". *John Wiley & Sons, Inc*.