Uso de Metaheurísticas para la Optimización de la Secuencia de Producción y la Asignación de Mano de Obra en una Empresa Manufacturera

Ing. Vinicio Sevilla [(1)]
Ing. Carlos Zurita [(2)]
Mat. Fernando Sandoya, MSc. [(3)]

Instituto de Ciencias Matemáticas ICM
Escuela Politécnica del Litoral
Campus Gustavo Galindo, Km 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil, Ecuador

vinicio.sevilla@yahoo.com [(1)]
carloszurita77@hotmail.com [(2)]
fsandoya@espol.edu.ec [(3)]

Resumen

El problema de secuenciamiento de la producción consiste en encontrar una secuencia óptima de ejecución que satisfaga todas las restricciones del problema de producción analizado; y, dada su naturaleza combinatoria las metaheurísticas son el mecanismo más utilizado actualmente para resolver este tipo de problemas, la eficiencia de estas técnicas es determinada por el tiempo de ejecución en la computadora y el error obtenido. Las metaheurísticas que se han desempeñado mejor para resolver los problemas de secuenciamiento, son las de Búsqueda Variable Local y Algoritmos Híbridos, pero que han demostrado ser muy complejas y demandan muchos recursos computacionales. En la presente investigación se desarrolla un Algoritmo Genético implementado en el lenguaje de propósito general C++, el mismo que permite resolver el problema de secuenciamiento de órdenes de trabajo en una empresa manufacturera. Actualmente en esta empresa manufacturera, el secuenciamiento de órdenes de trabajo se lo realiza de manera prácticamente manual y en aproximadamente dos días para generar el secuenciamiento para un período de producción de una semana, obteniéndose además un plan que deja mucho tiempo improductivo. Gracias al algoritmo genético propuesto se tiene una mejor secuenciación en unos pocos segundos de procesamiento, en la cual el tiempo improductivo se ha reducido a 18 minutos por cada día de producción permitiendo también tener análisis de escenarios.

Palabras Claves: *Algoritmo Genético, Job Shop Scheduling, Mano de Obra, Metaheurísticas, Secuanciación de la Producción. .*

Abstract

The problem of production scheduling consists of finding an execution optimal sequence that satisfies all the restrictions with the analyzed problem of production; and given to their combine nature, the metaheuristics are the most used mechanism at the moment to solve this kind of problems, the efficiency of these techniques is determined by the run time in the computer and the obtained error. The better metaheuristics to solve the sequence problems are variable local search and Hybrid Algorithms, but they have demonstrated to be very complex and demand many computer resources. This present investigation had developed a Genetic Algorithm; it been implemented in the general-purpose language C++, who allows solving the problem of sequence of work orders in a manufacturing company. At the moment in this manufacturing company, the sequence of work orders are make practically manual and it takes two days to generate the sequence for a production period of one week, obtaining also a plan that leaves long unproductive time. Thanks to the genetic algorithm proposed, it takes a few seconds to have one better sequence of processing, in which the unproductive time has been reduced to 18 minutes per day of production and also allowing having analysis of sceneries.

1. Introducción

La mejora de los procesos productivos en las empresas manufactureras ha sido un factor clave para el éxito de los negocios. Estos procesos se han vuelto muy complicados debido a que son procesos sistémicos en los que el objetivo a cumplir es fabricar una cierta cantidad determinada de un mix de productos deseados al menor costo posible con recursos limitados y en un rango de fechas requeridas. En estos procesos intervienen recursos limitados tales como personal, máquinas, materia prima e insumos.

El objetivo es determinar el Plan de Producción y ejecutarlo en una secuencia tal que se minimicen tiempos improductivos, tardanza en las órdenes, u otros criterios deseados. Dentro del área del Scheduling, el problema del Job Shop (JSSP por sus siglas en inglés) es el que más aplicación práctica ofrece, pues permite incrementar la eficiencia de los procesos de manufactura. Desde los primeros años de la década de 1960, los problemas de Job Shop, han sido de mucho interés para los investigadores en métodos de optimización, ya que son problemas difíciles del tipo NP-duro.

En el pasado se han desarrollado un sinnúmero de heurísticas específicas para este problema. De éstas, el procedimiento de sustitución del cuello de botella (Shifting Bottleneck Procedure) se convirtió en uno de los más usados, aunque con resultados no tan satisfactorios. Hace pocos años el desarrollo de nuevas técnicas universales de búsqueda, denominadas metaheurísticas, han tenido auge gracias a los avances computacionales y se han convertido en los métodos recomendados para resolver el JSSP: los Algoritmos Genéticos, el Recocido Simulado, Búsqueda TABU, Búsqueda Variable Local Paralela entre otras son las nuevas técnicas que se están utilizando para poder resolver eficientemente este problema.

El objetivo principal de éste estudio es desarrollar un algoritmo genético que permita encontrar una secuencia de producción óptima en una empresa manufacturera.

2. Notación y caracterización del problema

El problema a resolver en la empresa manufacturera se describe y formula a continuación.

2.1 Datos de entrada

- Se tiene un conjunto de 10 máquinas que representan células de producción de los procesos existentes en la línea de interés.
- Se tiene un conjunto de artículos (tareas) con cantidades determinadas a producir en esa línea en el día de interés. Cada artículo tiene rutas establecidas con operaciones determinadas que dependen del producto a fabricar.

2.2. Objetivo

Minimizar el tiempo total de proceso de una secuencia factible. Donde el tiempo total del proceso es el máximo tiempo necesario para fabricar todos los artículos.

$$\text{Min Cmax} = \max_{j=1}^{N} C_j$$

Donde Cj es el tiempo total requerido para completar la tarea j y se calcula así: $Cj = Snj,j + dnj,j$

Snj,j: tiempo al que inicia la última operación de la tarea j.

dnj,j: tiempo de duración de la ultima operación de la tarea j

2.3. Restricciones

$O_{hj} - O_{ij} \geq d_{ij}$ para todo (i,j) $\longrightarrow$ (h.j) ε A

$C_{max} - O_{ij} \geq d_{ij}$ para todo (i,j) ε N

$O_{ij} - O_{ik} \geq d_{ik}$ o $O_{ik} - O_{ij} \geq d_{ij}$

Para todo (i,k) y (i,j), i = 1,…,m

$Oij \geq 0$ para todo (i,j) ε N

O_{hj} es la operación de la tarea h realizada en la máquina j.

La primera restricción asegura que una máquina j pueda procesar solo una tarea a la vez. La tercera restricción se conoce como la restricción disjunta, asegura que se respete el orden de diferentes tareas que se procesan en una misma máquina.

3. Algoritmos Genéticos

Los algoritmos genéticos, propuestos por Holland en 1975, son metaheurísticas que usan estrategias de búsqueda general para resolver problemas de naturaleza combinatoria. Los algoritmos genéticos simulan el proceso de evolución de los organismos vivos y están basados en el principio de "supervivencia del más apto" para formar la siguiente generación de posibles soluciones. Estas soluciones se construyen utilizando operadores probabilísticos, como el cruce genético, mutación y supervivencia del más apto (selección). A medida que avanza el proceso evolutivo los individuos más aptos

sobreviven, lo que representa las mejores soluciones, mientras que los individuos menos aptos desaparecen. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de éstas. En la figura 1 se muestra la ejecución típica de un algoritmo genético

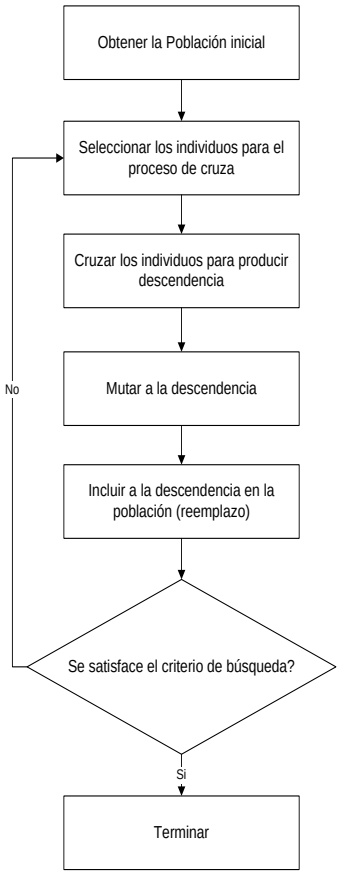

Figura 1. Ejecución general de un AG.

El algoritmo ha sido desarrollado en C++ usando como base el framework de algoritmos genéticos GALIB desarrollado por Matthew Wall en el Instituto Tecnológico de Massachussets (MIT) [1].

3.1. Codificación de los Genes.

Para el caso específico del problema JSSP, existen dos diferentes técnicas para representar el problema; la primera es una representación indirecta la cual codifica las instrucciones a través de un constructor de secuencia; por ejemplo, la permutación del orden de trabajos y la priorización de las reglas de secuenciamiento, en estos casos el constructor de secuencia asegura la validez del programa producido. La segunda técnica es la representación directa la cual codifica la secuencia propiamente dicha, como ejemplo de ésta técnica se puede citar la codificación de los tiempos de finalización de cada operación o los tiempos de inicio de cada operación, en ésta técnica no toda la codificación representa una secuencia válida.

La codificación usada para resolver el problema planteado en el presente proyecto de titulación es la permutación ordenada de trabajos. A continuación se explica ésta representación mediante un ejemplo didáctico, en un problema de 3 tareas con 4 recursos (máquinas) se tiene:

Tareas = {J1 , J2 , J3}
J1 = {t11(R1), t12(R2), t13(R3)}
J2 = {t21(R2), t22(R4), t23(R3)}
J3 = {t31(R1), t32(R3) }
Recursos = {R1, R2, R3, R4}
Du1j = Du2j = 2, Du3j = 3
Tmin = 0, Tmax = 10

En la figura 2 se muestra un grafo disjunto que representa gráficamente el problema de ejemplo.

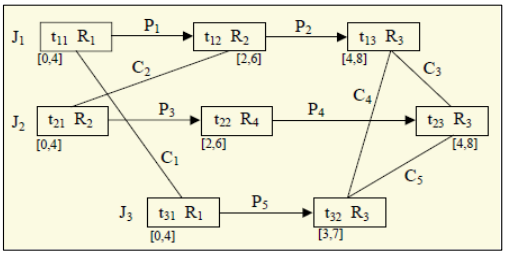

Figura 2. Grafo disjunto que representa al JSSP.

3.2. Número de Generaciones

Las generaciones permiten al algoritmo genético evolucionar obteniendo cada vez mejores cromosomas, por lo tanto es importante fijar un número adecuado de generaciones para conseguir un buen resultado. De la bibliografía revisada se obtiene que un número adecuado para éste tamaño de problema es 5000, de ésta forma nos aseguramos que el problema converja a la solución óptima.

3.3. Tamaño de la Población

El tamaño de la población define el número de cromosomas que existirán en cada generación. Si no existen suficientes cromosomas, el algoritmo no evoluciona. De igual manera, si existen muchos cromosomas, se llega a un punto en el que los resultados no mejoran. Este tamaño es directamente proporcional al número de operaciones. Debido a que en el problema de la empresa de estudio el número de operaciones es 10 se usó la relación tamaño de la población es igual al cuadrado del número de

operaciones, por lo tanto el tamaño de la población será 100.

3.4. Operador de Mutación

Para el algoritmo propuesto en el presente proyecto de titulación se usan dos operadores de mutación.

- Operador de mutación simple
- Operador de mutación combinada

La mutación simple consiste en seleccionar aleatoriamente un número entre 0 (cero) y el número total de genes del cromosoma el cual identifica la posición de mutación.

La mutación combinada consiste en:

- Seleccionar al azar un número y verificar si el número es menor a la probabilidad de mutación (%). Si es menor, continuar con el siguiente paso, caso contrario no realizar la mutación.
- Seleccionar al azar un número. Si es menor al 50%, realizar la mutación simple, para ello se seleccionan al azar dos posiciones del cromosoma y se intercambian sus valores, sino realizar una mutación de desplazamiento con inversión de genes o sin ella, para esto se generan otros números al azar. Para el presente estudio en la implementación del algoritmo se utilizó éste método de mutación, cuyo seudocódigo se presenta a continuación:

```
Si nrandom <= probMutación
   Si nrandom < 50%
      Intercambio(pos1,pos2)
   Caso contrario
      // Número de nodos para desplazarse
      nNodos = Random
      Seleccionar_nNodos(pos)
      // Identifica si los nodos seleccionados se
        insertan invertida su posición
      Invertir = Random (< 50% ; 0;  1)
      Insertar_nNodos(pos, invertir)
   Fin
Caso Contrario
   Salir
Fin
```

3.5. Probabilidad de Mutación

La probabilidad de mutación es un parámetro dado en el algoritmo que se encontrará mediante una prueba experimental. Para fijar el rango inicial de la probabilidad de mutación se tendrá en cuenta el siguiente criterio:

- La probabilidad de mutación trata de impedir que la búsqueda del algoritmo genético caiga en óptimos locales por eso es conveniente que ocurra ocasionalmente. No es bueno sin embargo, que la mutación ocurra continuamente, ya que la búsqueda del genético pasa de ser "inteligente" a búsqueda aleatoria. Para el presente estudio se comenzará con un valor de 1/L, donde L es la longitud del cromosoma.

El menor valor del tiempo total de proceso o makespan, manteniendo inalterables los otros parámetros es de 946 para una probabilidad de mutación de 0,091 por lo tanto se trabajará con ésta probabilidad para el resto de calibraciones.

3.6 Operador de Cruce

El algoritmo desarrollado tiene la posibilidad de usar dos operadores de cruce: el PMX (Partial Matched CrossOver) y JOX (Job Order CrossOver). El primero se refiere a tomar dos puntos de cruza en cada padre y el segundo se refiere a mantener ciertos genes constantes en los hijos y los otros se completan en función a un intercambio entre los genes de los padres.

3.7. Probabilidad de Cruce

La probabilidad de cruce es un parámetro dado en el algoritmo que se encontrará mediante una prueba experimental. Para fijar el rango inicial de la probabilidad de cruce se tendrá en cuenta el siguiente criterio:

- La probabilidad de cruce debe estar entre 0,6 al 0,95; de ésta forma se asegura que el problema no llegue a convergencia de forma lenta y algunos individuos pasen a la siguiente generación sin cruzarse.

Para encontrar el valor de éste parámetro se usa el problema estándar FT10 y se usa el operador PMX. Se realizaron 36 pruebas variando la probabilidad de cruce en 0,01 desde 0,6 hasta 0,95 obteniendo para cada una el makespan obtenido.

El mejor valor de éste parámetro es 0,6 con un makespan de 946 (2% de error).

3.8. Probabilidad de Reemplazo

La probabilidad de reemplazo es necesaria ya que el algoritmo construido es del tipo STEADY-STATE, el cual utiliza el traslape de poblaciones. Para cada generación solamente una porción de ésta es

reemplazada por sus hijos. Cuando la probabilidad de es de 1, solamente uno o dos genes son reemplazados; mientras que si la probabilidad es del 0, todos los individuos de la población son reemplazados. Para la calibración de la probabilidad de reemplazo se realizan 20 experimentos con el problema estándar FT10, donde se mantiene constante la población, el número de generaciones, la probabilidad de mutación, la probabilidad de cruce y la semilla para la generación de los números aleatorios.

El mejor valor de makespan manteniendo inalterables los otros parámetros es de 946 para una probabilidad de cruce de 0,6 por lo tanto se trabajará con ésta probabilidad para el resto de calibraciones.

3.9. Archivo de Parámetros

El programa desarrollado trabaja con un archivo de parámetros, el mismo que contiene la información con la que va a operar el algoritmo. El nombre del archivo debe ser siempre "entrada_jss.txt"; los datos que contiene son:

- Probabilidad de cruce
- Probabilidad de mutación
- Tamaño de la población
- Número de generaciones
- Probabilidad de reemplazo
- Semilla de números aleatorios

3.10. Archivo de Resultados

El programa desarrollado genera un archivo de resultados, el mismo que contiene la información de la solución del problema, el nombre del archivo siempre es "resultados_jss.txt"; contiene la siguiente información:

- Makespan mínimo
- Secuencia planificada
- Estadística de toda la ejecución del algoritmo como: número de cruces, número de mutaciones, promedio de todos los makespan obtenidos, entre otros.

3.11. Diagrama de Flujo

En la siguiente figura se presenta el diagrama de flujo del algoritmo propuesto para resolver el problema de interés.

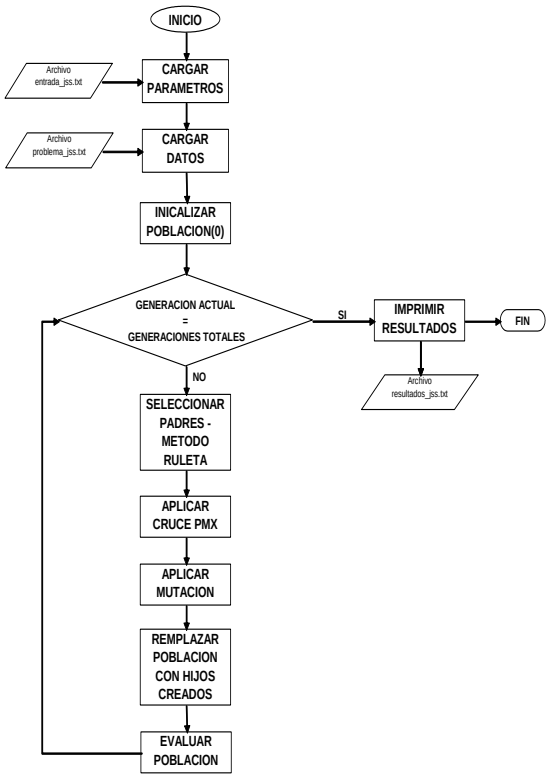

Figura 3. Diagrama de Flujo del AG.

4. Resultados Computacionales

Para la evaluación del algoritmo desarrollado durante el presente proyecto de titulación, se ha seleccionado 5 problemas estándar de estudio de la comunidad científica sobre el JSSP disponibles en Internet en la librería OR [2].

Los experimentos han sido ejecutados en una LAPTOP Core 2 DUO T5500, 1.66Ghz, con una memoria RAM de 2.0 GB, bajo Windows XP Service Pack 3.

Se realizaron 400 ejecuciones para cada instancia (problema estándar). En cada problema estándar con los parámetros calibrados se evaluó las 4 combinaciones posibles de operador de mutación y operador de cruce:

- Operador de cruce JOX y mutación simple
- Operador de cruce JOX y mutación combinada
- Operador de cruce PMX y mutación simple
- Operador de cruce PMX y mutación combinada

En la tabla 1 se presentan los resultados obtenidos por el algoritmo en las instancias evaluadas, donde la columna JSS se refiere al algoritmo desarrollado.

Instancia	Dimensión	Óptimo Conocido	JSS
FT20	20x5	1165	1183
FT10	10x10	930	946
LA24	15x10	935	984
LA27	20x10	1235	1333
FT6	6x6	54	55

Tabla 1. Resultados por Instancia.

Con los parámetros previamente calibrados se ingresan los datos del problema de estudio. Se realizan 26 pruebas en la tabla 2 se presentan los resultados.

No	Reem.	Pobl.	# Genera.	Mut.	Cruce	Seed	MAKESPAN
0	0,86	100	5000	0,091	0,6	0	599
1	0,86	100	5000	0,091	0,6	1	587
2	0,86	100	5000	0,091	0,6	2	599
3	0,86	100	5000	0,091	0,6	3	599
4	0,86	100	5000	0,091	0,6	4	588
5	0,86	100	5000	0,091	0,6	5	584
6	0,86	100	5000	0,091	0,6	6	573
7	0,86	100	5000	0,091	0,6	7	590
8	0,86	100	5000	0,091	0,6	8	574
9	0,86	100	5000	0,091	0,6	9	606
10	0,86	100	5000	0,091	0,6	10	601
11	0,86	100	5000	0,091	0,6	11	589
12	0,86	100	5000	0,091	0,6	12	579
13	0,86	100	5000	0,091	0,6	13	614
14	0,86	100	5000	0,091	0,6	14	615
15	0,86	100	5000	0,091	0,6	15	592
16	0,86	100	5000	0,091	0,6	16	582
17	0,86	100	5000	0,091	0,6	17	596
18	0,86	100	5000	0,091	0,6	18	601
19	0,86	100	5000	0,091	0,6	19	592
20	0,86	100	5000	0,091	0,6	20	608
21	0,86	100	5000	0,091	0,6	21	591
22	0,86	100	5000	0,091	0,6	22	603
23	0,86	100	5000	0,091	0,6	23	596
24	0,86	100	5000	0,091	0,6	24	590
25	0,86	100	5000	0,091	0,6	25	603
26	**0,86**	**100**	**5000**	**0,091**	**0,6**	**26**	**572**

Tabla 2. Resultados del problema de estudio.

El algoritmo desarrollado en el presente proyecto de titulación presenta un makespan de 572 minutos mientras que por el método tradicional de planificación toma 917 minutos (es decir un 60% adicional para éste caso específico).

5. Conclusiones y Recomendaciones

El algoritmo desarrollado en ésta Investigación cumple el objetivo propuesto inicialmente, el cual era obtener un mejor plan finito de producción en un tiempo menor al que se requería en el método de planificación tradicional y en un tiempo razonable. En uno de los casos a resolver se pudo comparar que el algoritmo propuesto disminuyó el makespan en 60% del valor que se obtenía mediante la forma tradicional de planificación. El tiempo de ejecución del algoritmo genético propuesto oscila de dos a tres minutos, más 15 minutos para elaborar la secuencia como un diagrama de Gannt son 18 minutos aproximadamente por cada día de producción; mientras que anteriormente se tardaba dos días para cada semana de producción. Cabe recalcar que éste valor es para uno de los casos propuestos en los que se pudo comparar los resultados obtenidos por los dos métodos.

La rápida ejecución del algoritmo permitirá al planificador de la producción, realizar varios escenarios factibles y analizarlos con las particularidades que ocurren cada día de forma que cuando el desarrollo se convierte en una herramienta cuando hay imprevistos.

Con la implementación del algoritmo genético se espera obtener en la empresa un ahorro de 143000 USD al año en reducción de horas extras al personal de la planta. Básicamente se espera un ahorro de 12000 USD mensuales debido al pago de horas extras no productivas debido a errores en el secuenciamiento de la producción.

Se recomienda continuar con el desarrollo del programa para hacerlo más amigable con el usuario final.

6. Referencias

[1] Instituto Tecnológico de Massachussets. Web site: http://lancet.mit.edu/ga/, 2009.

[2] Or-library librería de investigación de operaciones. Web site: http://people.brunel.ac.uk/ mastjjb/jeb/orlib/jobshopinfo.html, 2009.

[3] J. Balas. The shifting bottleneck procedure for job shop scheduling. pags. 394{401, Management Science Vol 34, USA, 1998.

[4] C. Blum. Meta-heuristics in combinatorial optimization: Overview and conceptual comparison. In Ann Discrete Math., pags. 269{308, ACM Comput Surv, 2003.

[5] C. Blum. Hybrid metaheuristics. studies in computational intelligence. In vol 114, pags. 17{37, 2008.

[6] M. Bramlette. Initialization, mutation and selection methods in genetic algorithms for function optimization. In Proceedings of the

Fourth International Conference on Genetic Algorithms, pags. 100{107, 1991.
[7] P. Brucker. Complex scheduling. springer science and business. pags. 180-210, Media Inc. New York, 2006.
[8] P. Brucker. Scheduling algorithms fifht edition. springer science and business. pags. 170-190, Media Inc. New York, 2007.
[9] D. F. E.S. Roscoe. Organization for production. In fifth edition, Richard D. Irwin Inc. Homewood Illinois, 15 pags., 1971.
[10] J. Herrmann. Handbook of production scheduling. In Handbook of Production Scheduling, pags. 2-20, Springer Science and Business Media Inc., New York, 2006.
[11] C. J.F. Blackstone, J.H. Spencer. Apics dictionary. In American Production and Inventory Control Society, pags 17-35, Virginia, 2001.