Año 7, Vol. IV, N° 12
ISSN: 1856-8327

Algoritmo genético para secuenciación de pedidos en taller de mecanizado con máquinas en paralelo, recirculación y tiempos de preparación.

Genetic algorithm for scheduling orders on a machining workshop with parallel machines, recirculation and setup times

Enrique Pérez Pérez Ilse J. Pérez Castillo Manuel E. Jiménez Bahri

Palabras clave:: Algoritmo genético, Secuenciación de tareas, Taller, Optimización combinatoria
Key Words: Genetic algorithm, Scheduling, Job-Shop, Combinatorial optimization

RESUMEN

En este artículo, se analizan las características relevantes de un problema de secuenciación de pedidos en un taller de fabricación de piezas, con máquinas en paralelo, recirculación y tiempos de preparación independientes de la secuencia. Es presentado un algoritmo genético para resolver el problema de secuenciación de pedidos con individuos de tamaño fijo basados en una permutación de éstos, los mismos constituidos por lotes de piezas de uno o más tipos, la población de individuos del algoritmo genético permanece constante en el tiempo, se usan los operadores genéticos cruce y mutación con porcentajes que el usuario puede variar a voluntad. El cruce opera sobre dos individuos "padres" seleccionados aleatoriamente sin elitismo para crear dos descendientes. Para la mutación se escoge al azar un individuo y se intercambia parte de su material aleatoriamente. El tamaño de la población es constante y se fijó en 50 individuos. El criterio objetivo seleccionado fue la minimización del tiempo total de fabricación y fue calculado mediante una simulación de eventos discretos plenamente determinísticos. El algoritmo fue programado en Java bajo Netbeans y fue ejecutado sobre una serie de problemas reales. Se pudo comprobar que al operar con bajos porcentajes de cruce (20%) y altos porcentajes de mutación (80%) permitió un mejor desempeño del algoritmo genético. En general se obtuvo una reducción del tiempo total de fabricación de entre 10%-20% comprobando que los algoritmos genéticos se constituyen en una herramienta prometedora para aumentar la eficiencia en problemas de secuenciación de pedidos en un taller de mecanizado.

ABSTRACT

In this paper, the relevant features of a problem of sequencing orders are discussed in a workshop production of parts with parallel machines, recirculation and independent setup time. A genetic algorithm to solve the problem of sequencing order is presented with individuals fixed size based on a permutation of sequencing order, the same consisting of batches of parts of one or more types, the population of individuals in the genetic algorithm remains constant over time, the crossover and mutation are used with percentages may be varied by the user. The crossover operates on two individuals "parents" randomly selected without elitism to create two offspring. For the mutation an individual is chosen at random and part of its material is exchanged randomly. The population size is constant and fixed to 50 individuals. The selected target criterion was the minimization of the total manufacturing time and was calculated using a simulation of fully deterministic discrete events. The algorithm was programmed in Java on Netbeans and was executed on a number of real problems. It was found that when operating with low percentages of crossing (20%) and high rates of mutation (80%) a better performance of the genetic algorithm is given. In general, a reduction in total manufacturing was obtained between 10%-20% checking that genetic algorithms constitute a promising tool to increase efficiency in order sequencing problems in machine shop.

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

INTRODUCCIÓN

Preliminares.

Los problemas de secuenciación de tareas (scheduling) están en casi todas las situaciones del mundo real, especialmente en el mundo de la ingeniería industrial, y aunque son de naturaleza simple, paradójicamente, son muy difíciles de resolver por las técnicas convencionales de optimización. Son una importante clase de problemas de los del tipo de optimización combinatoria, caracterizados por tener un número finito de soluciones factibles y sujetos a restricciones de variable complejidad, además es posible enfrentarse a parámetros no determinísticos. En teoría de la complejidad son denominados NP-difícil (NP-hard), problemas para los cuales no existe hasta el momento un algoritmo determinístico que los resuelva en tiempo polinomial (Lenstra et al, 1977; Papadimitriou y Steiglitz, 1998). Esto ha dado lugar al creciente interés en usar meta-heurísticas como los algoritmos genéticos para abordar estos problemas (Gen y Cheng, 2000; Varela, 2008).

Las tareas pueden tener varias interpretaciones, desde la manufactura de piezas mecánicas hasta el procesamiento de información en sistemas de computación; las máquinas pueden referirse a objetos o entes que se mantienen ocupados durante un tiempo aplicado a una tarea; el objetivo a menudo consiste en minimizar el tiempo total de procesamiento (makespan), tiempo total del flujo de tareas en el sistema, la satisfacción de los tiempos de entrega, uso eficiente de recursos, tiempo ocioso de servidores o máquinas, completitud de pedidos y otros (Gen y Cheng, 2000).

Este trabajo se enfocó en la construcción de un algoritmo genético que genere soluciones que sirvan mejor a los objetivos de una empresa de fabricación de piezas mecánicas por ser éstos herramientas meta-heurísticas que han sido usadas exitosamente en la obtención de soluciones que mejoran el desempeño en problemas de optimización combinatoria. El problema que ha sido tratado consiste en establecer la secuencia en que debe ordenarse un conjunto de 20 pedidos de piezas mecánicas (que corresponden aproximadamente a una semana de jornadas de 8 horas diarias de labor del taller), que minimicen el tiempo total de fabricación, donde se establece la posibilidad de máquinas idénticas en paralelo, recirculación (las tareas pueden pasar cero o más veces por el mismo tipo o la misma máquina) y tiempos de preparación o alistamiento independientes de la secuencia (se refiere al tiempo necesario para alistar la máquina que ejecuta una nueva tarea). Mediante la realización sistemática de 100 corridas de este algoritmo para una variedad de problemas reales se estudió el comportamiento estadístico del tiempo total de fabricación, cantidad de generaciones, tiempo total de corrida y tasa de mejora, así como la apreciación del comportamiento del peor individuo, individuo promedio y mejor individuo para diversos valores de los porcentajes de cruce y porcentajes de mutación, para un tamaño de población pre-definido y fijo durante la corrida.

En este artículo se mencionan antecedentes de este trabajo y otras investigaciones relacionadas así como algunas teorías que servirán de marco a la investigación. Realizado el planteamiento del problema, luego se desarrolla la metodología seguida y finalmente se muestran los resultados obtenidos y las conclusiones alcanzadas, también se esbozan algunos criterios de interés para investigaciones futuras.

Planteamiento del problema.

Es común que en los pequeños y medios talleres de mecanizado (talleres donde se construyen piezas mecánicas) en la localidad de Valencia, Venezuela, se trabaje bajo pedido, en otras palabras, las piezas son fabricadas a medida que son requeridas por los clientes, por lo que se dispone de un número variable de pedidos que serán satisfechos por el taller. En la medida que esto sea realizado en el menor tiempo posible, sin incurrir en horas extras de trabajo y sin violar los tiempos de entrega comprometidos se minimizarán los costos de

producción y mejorará la visión que tienen los clientes del taller. La minimización del tiempo total de fabricación o makespan no necesariamente implica dar un mejor servicio al cliente, aunque relacionados, los objetivos que implican la fecha de completitud y entrega de un pedido permiten optimizar la calidad de servicio.

Eventualmente son aceptados pedidos de piezas bajo especificación del propio cliente, lo natural es que los tipos de piezas que podrán ser fabricados están previamente establecidos como también la secuencia de procesos a realizar para la fabricación de cada tipo de pieza, la fabricación de una pieza o trabajo requiere entonces de una secuencia de operaciones, cada operación requiere de una máquina específica para su realización, pudiendo repetirse para ciertos casos la misma máquina para operaciones distintas. Una operación puede comenzar solo cuando todas las operaciones precedentes han sido completadas.

Un típico ambiente de manufactura en un taller de mecanizado (job shop) implica una gran variedad de productos y un relativo bajo volumen de productos por tipo. Estos procesos o tareas establecidas lineal y secuencialmente para cada producto o "plan maestro" implican el uso de máquinas y operarios. Para realizarlos se supondrá en particular que la máquina requerirá una cantidad de tiempo inicial para su preparación y/o instalación y un tiempo adicional propio del proceso, ambos específicos para cada proceso (Figura 01). De manera que el taller se abocará a la fabricación simultánea de distintos tipos de piezas para satisfacer un conjunto de pedidos aceptados, cada uno de ellos constituidos por lotes de tamaño variable de uno o más tipos de piezas. Así, el orden en que son realizados los procesos o secuencia podrá ser modificado con el objeto de hacer más eficiente el desempeño total, evitando los tiempos de ocio de las máquinas cuando esto favorezca la disminución del tiempo global de fabricación. En la práctica, lo más común es que en estos talleres no exista tal programación o bien responde simplemente a la pericia o experiencia del programador de turno, que conociendo los pedidos aceptados y su composición de tipo de piezas podrá reordenarlos con este propósito. Para abordar la situación problemática presentada se requiere establecer una herramienta que permita elaborar el orden en que deberán ser ejecutados los pedidos de manera que el tiempo total de fabricación sea el menor posible dentro de ciertas limitaciones. La meta no es obtener necesariamente la secuenciación óptima, se desea construir una secuenciación de pedidos que tenga un desempeño aceptable y poder calcular esta secuenciación con un esfuerzo computacional y en tiempo razonables.

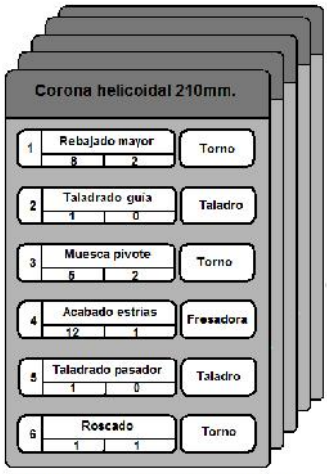


Figura 01. Secuencia de Operaciones.

MARCO TEÓRICO

Antecedentes.

La secuenciación de tareas está caracterizada por una ilimitada cantidad de distintos tipos de problemas y desde los 50's cientos de ellos han sido modelados y estudiados. Entre éstos, la secuenciación de tareas enfocadas al trabajo de un taller de mecanizado de piezas (Job Shop Scheduling) se constituye en un paradigma de los problemas de secuenciación de tareas y ha interesado ampliamente a los investigadores desde hace algunas décadas (Brucker, 2007; Pinedo, 2009). En 1960 Giffler y Thompson demuestran que una programación de actividades óptima debe pertenecer a un conjunto particular que tiene ciertas características (programación activa) y proponen un algoritmo para generar ese conjunto

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

para un problema (Giffler y Thompson, 1960). También fue demostrada la validez de un modelo enfocado a resolver este problema mediante la programación lineal entera mixta aunque no fue satisfactorio para problemas reales de gran escala, por el tiempo requerido para su ejecución (Manne, 1960). Numerosos métodos exactos y aproximados, basados en diversas técnicas, han sido desarrollados. Muchos se fundamentan en el concepto de camino crítico (Shifting Bottleneck) como el propuesto por Adams, Balas y Zawack en 1988 (Adams et al, 1988). El más relevante es el algoritmo de ramificación y poda (Branch and Bound) propuesto por Brucker (Brucker et al, 1994), basado en técnicas de Carlier y Pinson (Carlier y Pinson, 1989), como también el propuesto por Applegate y Cook que combina técnicas heurísticas con un algoritmo de ramificación y poda (Applegate y Cook, 1991).

Los métodos exactos están limitados en el tamaño de los problemas que pueden manejar debido a sus requerimientos de espacio y de tiempo, la hibridación de éstos con algoritmos de búsqueda local proporcionan un auxilio fundamental para manejar los casos de gran tamaño (Duvivier et al, 1996). Entre los métodos de aproximación iterativa destacan las técnicas de búsqueda local como los algoritmos genéticos y búsqueda tabú (Varela, 2008). Falkenauer y Bouffouix en 1991, están entre los primeros en usar algoritmos genéticos para resolverlo y demostrar la viabilidad de su aplicación a problemas reales promoviendo un esquema adecuado de codificación y operadores de cruzamiento y mutación (Falkenauer y Bouffouix, 1991). También en 1991, Yamada y Nakano proponen un método para resolver el problema Job-Shop basado en algoritmos genéticos que muestran un desempeño sobresaliente para los difíciles problemas de Muth y Thompson, creados en 1963 y usados desde entonces como prueba (benchmark) (Yamada y Nakano, 1996). Se proponen las permutaciones con repetición como un enfoque más cercano a la representación de los individuos (Bierwirth, 1995). Se propone un operador de cruce adaptado al problema de secuenciación "Job-Shop" y también una fusión de éste junto a un método de búsqueda local para la construcción de la descendencia (Yamada y Nakano, 1997). Diversos algoritmos genéticos para resolver el job-shop scheduling con variantes sobre la capacidad de la representación de la diversidad de soluciones y operadores de cruce y mutación han sido propuestos a lo largo de los últimos años (Fissgus, 2000; Lestan et al, 2009; Huang y Lin, 2010, Li y Chen, 2010; Goncalvez y Resende, 2011; Werner, 2011; Vaghefinezhad, 2012). Hibridación de algoritmos genéticos con otras técnicas de búsqueda local (Goncalves et al, 2002; Hasan et al, 2007). Así como también se han creado algoritmos genéticos que resuelven el problema Job-Shop con objetivo de optimización multicriterio (Coello, 2010). El problema Job-Shop ha sido atacado usando técnicas como el recocido simulado (Simulated annealing) mejorado con la incorporación de técnicas de búsqueda local (Vaessens, Aarts y Lenstra, 1996; Yamada y Nakano, 1996; Kammer et al, 2010).

Los problemas de secuenciación de tareas (scheduling).

Los aspectos más relevantes que caracterizan los problemas de secuenciación son: siempre se consideran finitos tanto el número de máquinas y de trabajos, como también el tiempo de procesamiento de cada trabajo en cada máquina, estos últimos no siempre se consideran determinísticos. Además también es posible encontrar trabajos a los que se les asignan distintas prioridades. Con respecto al ambiente en el taller se tiene: Secuenciación de tareas en una sola máquina, está asociada al taller básico de una sola máquina con mono-operación por trabajo y su importancia radica en que sirve de fundamento para tratar problemas de mayor complejidad. Flow Shop, el taller dispone de distintas máquinas para procesar los trabajos que deben circular por todas las máquinas siempre en el mismo orden para todos los trabajos, está enfocado en el ambiente de producción por líneas, por lo que después de establecida una permutación las máquinas siempre

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

recibirán los trabajos en la misma secuencia. Job Shop, el taller dispone también de distintas máquinas, y cada trabajo tiene su propia secuencia de recorrido de todas las máquinas, está enfocado al ambiente de centros de trabajo. Open Shop, están disponibles varias máquinas en el taller y los trabajos deben recorrer todas las máquinas pero sin un orden especial. Mixed Shop, los trabajos deben recorrer todas las máquinas, algunos en un orden específico y los demás no. Esta sucinta simplificación puede incluir la posibilidad de varias máquinas iguales en paralelo (por algunos autores denominado flexible y a veces híbrido), las tareas no tienen que visitar todas las máquinas, posibilidad de que la misma tarea circule por la misma máquina para realizar distintos procesos (recirculación), máquinas multipropósito que pueden realizar varios o hasta todos los procesos, estaciones de trabajo y otras. Con respecto a detalles de las características del tiempo de procesamiento y restricciones a las tareas se permite interrumpir el procesamiento de un trabajo para reanudarlo posteriormente con el propósito de emplear la máquina en otro trabajo (preemption), existen restricciones de precedencia entre trabajos, a los trabajos se les puede asignar un tiempo mínimo para poder ser iniciados, cantidad de tiempo de procesamiento, a los trabajos se les puede asignar un tiempo máximo para su finalización y la posibilidad de agrupar conjuntos de trabajos en bloques. Finalmente, el criterio de optimización es generalmente la minimización del tiempo total de procesamiento y/o minimización del tiempo total de flujo de los trabajos en el sistema. Obviamente, éstos son modelos generales ideales que en la literatura científica se describen y formulan de una forma abstracta, en el mundo real los problemas de secuenciación de tareas son frecuentemente mucho más complejos involucrando factores y restricciones adicionales (Pinedo, 2009; Baker y Trietsch, 2009; Zäpfel, 2.010; Pinedo, 2012).

Tradicionalmente las heurísticas de secuenciación son vistas como un todo absoluto y no como la asociación de diversos elementos que colaboran en la búsqueda de un fin; es generalmente aceptada la medida del tiempo de uso del tiempo de computador como la forma de evaluación del desempeño de un algoritmo; los problemas tipo usados como punto de referencia (benchmark) son adoptados sin ninguna consideración de sus características individuales propias; la minimización del tiempo total de procesamiento es usado consistentemente como principal y a veces como única meta de un algoritmo (Beck et al, 1997). El modelo Job-shop flexible de minimización del tiempo total de procesamiento con recirculación y tiempos de preparación independientes de la secuencia se cuenta en el conjunto de los ambientes de secuenciación más complejos y es común encontrarlo en la industria mecánica y electrónica (Brucker, 2007; Magalhães-Mendes, 2013).

Los algoritmos genéticos.

Los algoritmos genéticos fueron propuestos por John Holland y desarrollados por él conjuntamente con sus alumnos en la Universidad de Michigan en las décadas de los 60's y 70's, su interés inicial no respondía en crear una forma de resolver un problema específico, sino más bien estudiar formalmente el fenómeno de la adaptación natural para desarrollar formas en que estos mecanismos pudieran ser incorporados a los sistemas informáticos. Los AG son algoritmos de optimización, tratan de encontrar la mejor solución a un problema de entre un conjunto de soluciones posibles dadas. Las herramientas usadas en esta búsqueda pretenden emular artificialmente el proceso natural de la evolución biológica. En la naturaleza los individuos de una población se ven afectados por su entorno y por individuos de otras especies e incluso por los de su misma especie. Aquellos que tengan éxito en esta competencia por la supervivencia tienen mayor probabilidad de generar descendientes, por el contrario individuos poco dotados producirán menos o ningún descendiente. Esto significa que los genes de los individuos mejor adaptados se propagarán en sucesivas generaciones hacia un número de individuos creciente. La combinación de buenas características provenientes de diferentes ancestros, generación tras generación desembocará en

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

especies evolucionadas con características cada vez mejor adaptadas al entorno en el que viven (Goldberg, 1989; Mitchell, 1996).

Estas ideas pueden ser transferidas sin trauma hacia los problemas de optimización. Así, el conjunto de las soluciones de un problema específico corresponderán a los miembros de una población en particular y donde la aptitud de cada individuo de la población equivaldrá a la calidad de cada solución del conjunto. Siguiendo con la metáfora de la evolución biológica las soluciones con mejores desempeños muy probablemente serán seleccionadas para emparejarlas y generar nuevas soluciones e incluso podrán cambiarse también algunas de sus características de manera aleatoria simulando mutaciones, todo esto con la esperanza de que estas nuevas soluciones alcancen mejores desempeños que sus ascendientes (Goldberg, 1989; Mitchell, 1996).

El poder de estos algoritmos reside en el hecho de que se trata de una técnica estable que puede tratar con éxito una amplia gama de problemas de diversas áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el algoritmo genético encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable en tiempos competitivos. El investigador que seleccione esta técnica en principio no tendrá que ser experto en el área específica en que se aplicarán y de alguna manera se convierten en un instrumento ciego y sin sesgos en la búsqueda del óptimo.

En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que los algoritmos genéticos sean superados, tanto en rapidez como en eficacia. El gran campo de aplicación de éstos es la de problemas para los cuales no existen técnicas especializadas, incluso en el caso en que dichas técnicas existan y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los algoritmos genéticos (Sivanamdam y Deepa, 2008).

Los elementos necesarios para poder aplicar un algoritmo genético simple a un problema son los siguientes: Poder representar una solución al problema o individuo, que no es más que una manera de codificar de forma preferiblemente biunívoca una solución al problema por medio de una cadena de bits, números o hasta caracteres. También son llamados comúnmente cromosomas y a sus elementos como genes o también alelos. Y alguna forma de evaluar la bondad de este individuo, que se le denomina función objetivo o aptitud. De manera, que la aplicación de esta función sobre un individuo debe poder calificar sus virtudes en conjunto con el objeto de su comparación con la aptitud de cualquier otro (Michalewicz, 1996).

Por otra parte, el funcionamiento general de este algoritmo podrá ser resumido así:
- Generar una población inicial.
- Repetir hasta que se cumpla un criterio de parada.
    - Evaluar cada individuo de la población.
    - Seleccionar los progenitores.
    - Aplicar el operador de cruce o recombinación y mutación a estos progenitores.
    - Incluir la nueva descendencia para formar una nueva generación (Michalewicz, 1996).

Hay muchas variantes de los algoritmos genéticos y son obtenidas tanto por el tamaño de la población inicial y los criterios de selección de individuos para ser objeto de recombinación y mutación como mediante la manipulación voluntaria de los parámetros de los operadores cruce y la mutación. Estos operadores determinarán como se intercambiará la información de los individuos seleccionados como padres y así obtener las nuevas soluciones o hijos. Y también como se alterarán los individuos por efecto de la mutación. Los primeros tienden a aumentar la calidad de las poblaciones y la tendencia a converger a un solo individuo, con el consiguiente peligro de convergencia de la población a un mínimo local. En contraposición, la mutación se opone a la convergencia y crea nuevas soluciones a veces no exploradas (Mitchell, 1996; Coello, 2010).

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

Simulación de eventos discretos y determinísticos.

Al construir el modelo matemático que representa una situación real se debe realizar un estudio preliminar que permita decidir si éste es lo suficientemente sencillo para poder hallarle una solución analítica. La complejidad de muchos sistemas reales impide hallar un modelo analítico satisfactorio y obliga a estudiarse recurriendo a la simulación. Una simulación se denomina discreta o de eventos discretos si implica variables de estado del sistema que varían en instantes discretos de tiempo y determinística si no contiene ninguna variable aleatoria (Pazos y otros, 2003).

El modelo de simulación es una herramienta válida para sistemas industriales de manufactura, los sistemas de secuenciación de tareas flow-shop, job-shop y sus variantes han sido modelados profusamente mediante simulación discreta y en éstos destacan las siguientes características: los eventos que dinamizan la simulación son los procesos, las máquinas son estaciones de servicios que pueden estar desocupadas o no, el tiempo de procesamiento en una máquina equivale al tiempo de servicio, un fin de servicio genera un nuevo evento que corresponderá al siguiente proceso de la pieza, la espera de procesos por la desocupación de una estación se administra como colas de diverso tipo (PEPS, UEPS, Prioridad) y el cálculo de estadísticas como tiempo en el sistema de un producto, tiempo de ocupación de una estación y otros (Williams y Ahitov, 1996, Law, 2007).

METODOLOGÍA

El modelo de esta aplicación

Como se ha venido argumentando el problema contempla la programación de la producción de las piezas solicitadas en 20 pedidos donde cada uno de éstos puede incluir un número variable de piezas de varios de los tipos disponibles. El modelo propuesto estará orientado a la situación particular planteada y por eso se consideran **M** máquinas de **H** tipos distintos (**h = 1, 2,…, H**) así habrá m_h máquinas del tipo **h** ($m_h \geq 1$), de tal manera que $m_1 + m_2 + \ldots + m_H = M$. Se procesarán **N**=20 pedidos **(i = 1, 2, ..., N),** cada uno tendrá n_i ítems (r_{ij}, s_{ij}) para $n_i \geq 1$, la información de cada ítem esta compuesta del tipo de pieza r_{ij} y el número de ellas que son solicitadas como s_{ij} ($s_{ij} \geq 1$), donde **j** indica el ítem en el pedido **i**. Hay **G** tipos distintos de piezas disponibles para la fabricación, de tal manera que r_{ij}**=1, 2,…, G**. La producción de una pieza tipo **k=1, 2,…, G** requiere de la ejecución de un número a_k procesos o actividades simples (b_{kd}, c_{kd}, p_{kd}) para ($a_k \geq 1$) organizadas en forma lineal incluyendo la posibilidad de recirculación, la información de cada proceso esta compuesta del tipo de máquina usada b_{kd}, el tiempo c_{kd} para la preparación o alistamiento de la máquina en minutos y un tiempo p_{kd} propiamente de procesamiento en minutos, donde **d = 1, 2,...,** a_k es un índice que identifica el proceso y además puede ser usado como un indicador secuencial del orden de los procesos. Así, $b_{kd} = 1, 2,\ldots, H$; $c_{kd} \geq 0$ y $c_{kd} \in \Re$; $p_{kd} \geq 0$ y $p_{kd} \in \Re$.

Sean t_{ijd} : tiempo en que se inicia el proceso **d** del item **j** del pedido **i**.

$$¡t_{ijd} + c_{kd} + s_{ij} * p_{kd} \leq t_{ijd+1} \; \forall \; i, j, d \text{ y } k$$

Implicaciones que subyacen en este modelo.

- Las piezas del mismo tipo que pertenezcan al mismo pedido serán consideradas como un elemento de pedido en un paquete o lote.
- Un elemento de un pedido podrá procesarse en una y sola una máquina al mismo tiempo.
- No se permite la interrupción de los procesos en las máquinas, es decir, no se puede interrumpir el proceso sin haber terminado en una máquina y pasar a otra.
- Puede existir recirculación en un elemento de un pedido, es decir, que puede visitar el mismo tipo de máquina o inclusive la misma máquina en más de una ocasión operando distintos procesos.
- Una máquina no podrá ocuparse de más de un proceso al mismo tiempo. Hasta tanto una operación no haya terminado su procesamiento, la máquina en la cual se esté realizando dicha operación no se podrá considerar disponible para ningún otro trabajo.

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

- Cuando varios procesos estén en cola esperando por la desocupación de una máquina no habrá prioridad entre ellas a menos que pertenezcan a pedidos distintos, en ese caso el proceso que pertenezca a un pedido que esté en posición prioritaria dentro de la programación será atendido por excelencia.
- Los tiempos de preparación o alistamiento y de procesamiento son conocidos y determinísticos. Además se consideran independientes de la secuencia. En la realidad no son constantes como se ha supuesto, debido en primer lugar a la pericia de los operarios, materiales de distinta calidad y por supuesto un componente aleatorio natural en estos tipos de procesos donde se implican máquinas. Esto se ha hecho por simplicidad, fueron tomados estos tiempos sistemáticamente con 50 repeticiones y se han estimado sus medias, estas medias han sido consideradas como los tiempos de preparación y de procesamiento.
- Cuando una máquina reciba piezas del mismo tipo de la que acaba de terminar de procesar se ignorará el tiempo de preparación.
- Cuando una pieza que termina de ser procesada requiere de una máquina de otro tipo y existen varias desocupadas de ese tipo se tomará cualquiera de ellas, a menos que entre las máquinas libres alguna ya este preparada para recibir ese tipo de piezas y entonces esa será preferida.
- Cuando una máquina esté ociosa podrá activarse el proceso de preparación para la próxima pieza que será procesada ya que es perfectamente previsible el momento de terminación del proceso precedente y así hacer más eficiente la programación.
- Fue considerado tiempo de enfriamiento en ciertos casos, ya que algunas de las piezas al terminar un proceso requieren de enfriamiento no inducido para que conserve sus propiedades deseables, antes de comenzar con el proceso siguiente, durante este período de tiempo se realizará en la máquina que recibirá esta pieza el proceso de preparación para hacerlo más eficiente.
- En ese tipo de procesos de manufactura de piezas mecánicas es natural la producción de piezas defectuosas estadísticamente estimable que, o bien son reparadas con retrabajo que obviamente requiere de la incorporación de más recursos o tal vez pueden ser desechadas definitivamente por que no puede ser reparada. Fueron ignoradas estas situaciones en la programación.

Figura 02. Tipos y Cantidad de Máquinas.

Operacionalización del algoritmo genético.

Instancias del problema: En el taller donde se realizó el estudio se fabrican piezas de 124 distintos tipos, pero fue constatado que normalmente el 85% de las piezas fabricadas pertenece a un pequeño sub-conjunto de apenas 12 piezas. Se usan máquinas de 8 tipos pero hay varias de algunas de ellas por lo que el número asciende a 14 máquinas en total (Figura 02). Fueron seleccionados 5 instancias de problemas reales que estuviesen formadas por exactamente 20 pedidos, que fueron parcialmente modificados para excluir piezas fuera de las 12 consideradas (si fuese el caso). Éstas corresponden al trabajo que normalmente se realiza en una semana de 6 días con un turno de 8 horas diarias, es decir, aproximadamente 3.000 minutos de labor.

Figura 03. Estructura de un cromosoma de 20 genes (pedidos).

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

Representación de las soluciones: Representación por permutación de 20 objetos, esta representación es por supuesto la forma más natural de presentar una solución para el problema de secuenciar procesos, donde los procesos son listados en el orden en el cual son realizados. El uso de esta forma puede restringir el uso de los operadores de cruce (Goldberg, 1989). Entonces el espacio de búsqueda de ésta representación es el conjunto de posibles permutaciones de los pedidos, una solución en este trabajo se caracteriza fundamentalmente por un individuo o cromosoma formado por 20 genes que representa inequívocamente al conjunto de veinte pedidos que podrán ser dispuestos en cualquiera de sus 20! ≈ 24^{17} posibles órdenes o secuencias distintas sin temor a perder factibilidad, así esta ventaja permitirá generar los individuos de la población inicial con la seguridad de que todos los individuos serán factibles (Figura 03). Esta forma de representación es similar a la propuesta por Holsapple, Jacob, Pakath y Zaveri en 1993, que está basada en una representación de secuencias de trabajos que indica la prioridad con la que serán atendidos los procesos que los conforman (Holsapple et al, 1993). Como contrapartida, es cierto que hay individuos factibles que no podrán ser alcanzados desde esta forma indirecta de representación, pudiendo esconderse entre éstos el óptimo global. Comúnmente en los algoritmos genéticos implementados para resolver problemas de secuenciación de trabajos en talleres mecánicos es usada una representación directa, los genes que conforman los individuos representan los procesos secuenciales que conforman un trabajo, así el tamaño del individuo cambiará dependiendo de los tipos de trabajos que estén implicados en la secuenciación (Abdelmaguid, 2010). Los individuos para problemas como los considerados en este trabajo que contempla exactamente 20 pedidos pero con una cantidad variable de trabajos que oscila aproximadamente en alrededor de 80 trabajos tendrán una cantidad aproximada de 450 procesos, es decir individuos de tamaño variable entre 420 y 480 genes. Por eso a pesar de esto se convino trabajar con individuos de tamaño fijo, la ventaja que ofrece tener individuos de tamaño constante y muy pequeños contrarresta a todas luces esta desventaja.

Tamaño y Generación de la población inicial: El tamaño de la población debe estar relacionado a la longitud del cromosoma y a la complejidad del problema. Al elegir el tamaño idóneo de la población es intuitivo que las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que el trabajar con poblaciones de gran tamaño puede acarrear problemas relacionados con el excesivo coste computacional (Goldberg, 1989). En base a estas consideraciones se eligió un tamaño fijo de la población y se estableció en 50 individuos. La construcción de la población inicial fue concebida aleatoriamente, así fueron generados 50 individuos o cromosomas cada uno con veinte números enteros o genes que representan directamente la identificación del pedido. La permutación al azar de los 20 cromosomas garantiza la diversidad de las soluciones al problema eliminando la posibilidad de sesgo.

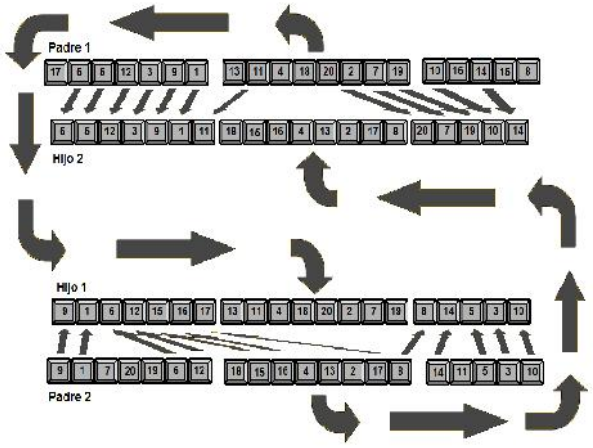


Figura 04. Cruce por Orden mostrando la organización de los padres y de la descendencia.

Función Objetivo: Calcular el desempeño de una solución del problema, en otras palabras evaluar la aptitud de un cromosoma, representa una complejidad considerable. Ya fue comentado que una de estas secuencias de 20 pedidos contiene aproximadamente 450 procesos en una diversidad de 14 máquinas con sensibles restricciones de secuencia y disponibilidad, por lo cual se recurrió a

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

una simulación discreta y determinística de la secuencia cada vez que se deba evaluar la aptitud de un individuo. Específicamente se programó la técnica del próximo evento para realizar esta evaluación, donde los procesos son eventos con los que se crea una cola de eventos en espera y las máquinas son estaciones que pueden estar libres o desocupadas. El tiempo total de procesamiento o makespan corresponde al tiempo empleado desde el inicio de operación del primer proceso hasta el fin de operación del último proceso, el fin de operación de un evento genera un nuevo evento que corresponderá al siguiente proceso de la pieza.

Operador de cruce: Por lo que atiene a los cruces entre dos individuos debido a la naturaleza del operador cruce implementado en este trabajo, no habrá la necesidad de seleccionar individuos compatibles para realizarlo. Se aplica según la tasa seleccionada para cruces (hasta el 100%) sobre un porcentaje de la población. Se seleccionan dos individuos escogidos al azar de entre la población actual (sin elitismo), que serán el padre-1 y padre-2, entonces se generan dos posiciones aleatorias que servirán para determinar la porción del individuo que se afectará directamente que se llamará sección de cruce o intercambio. Se originan dos descendientes de la siguiente manera. Se toma la sección de cruce del padre-1 y se incorpora en el hijo-1 en su misma posición relativa y se hace lo mismo con el hijo-2 pero tomando la sección del padre-2. Luego se copian los genes faltantes en el orden en que aparezcan en el otro padre (Figura 04). A este tipo de cruce se le denomina "Cruce por Orden" (OX, Order Crossover en idioma inglés) y fue propuesto por Lawrence Davis en 1985 como una variante ordenada del cruce llamado "Cruce por mapeo parcial" (PMX, Partial Mapped Crossover en idioma inglés) (Gen y Cheng, 2000). Claramente OX trata de incorporar parte de un padre y también incorporar parte del otro padre pero preservando el orden relativo de los elementos de éste. Habiendo realizado el cruce se incorporan a la población sólo los mejores individuos, en otras palabras, de estos cuatro individuos, dos padres y dos hijos solo permanecerá la pareja que tenga mejor aptitud, el otro par será destruido. Así el tamaño de la población permanecerá constante y se garantizará que la aptitud promedio de la población nunca será peor.

Operador de mutación: Se produce después del cruce con el objetivo de incrementar la diversidad poblacional. Se aplica según la tasa seleccionada para mutaciones (hasta el 100%) sobre un porcentaje de la población. Este operador permite, por una parte, aumentar la exploración en el espacio de búsqueda y por otro lado, evita la aparición de poblaciones iguales o semejantes. La mutación por inserción contempla la extracción de un gen en una posición y su inserción en otra posición del mismo individuo ambas posiciones escogidos aleatoriamente. Es aplicada un número aleatorio de veces sobre el mismo individuo (alrededor del 15% de los genes son cambiados de posición, es decir es aplicado entre 2 y 4 veces en el mismo individuo). Al haber realizado la mutación se incorpora a la población sólo si su aptitud ha mejorado, de otra manera tal mutación será desechada y el individuo regenerado. Como en el cruce, esto garantiza que la aptitud promedio de la población nunca será peor (Figura 05).

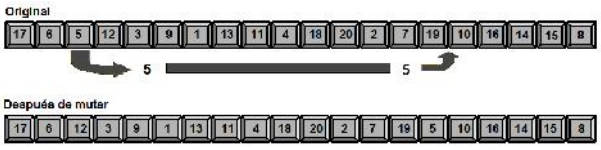


Figura 05. Mutación por inserción mostrando cambios del cromosoma.

Criterio de terminación: La población inicial será transformada muchas veces bajo el esquema presentado anteriormente y a cada una de estas se le denominará "generación", así en cada generación se selecciona y aplican cruces y se escogen y se aplican mutaciones. El número de generaciones dependerá fundamentalmente de la complejidad del problema a tratar, del tamaño de lo individuo, del criterio de selección de los individuos a transformar, del porcentaje de cruce y mutación y del criterio de permanencia de los individuos evolucionados. El criterio usado en este

Año 7, Vol. IV, N° 12
ISSN: 1856-8327

algoritmo genético fue el de terminar las iteraciones cuando sea alcanzada la convergencia, es decir, cuando todos los cromosomas sean idénticos, con la posibilidad cierta de que éste sea el óptimo o un próximo satisfactorio.

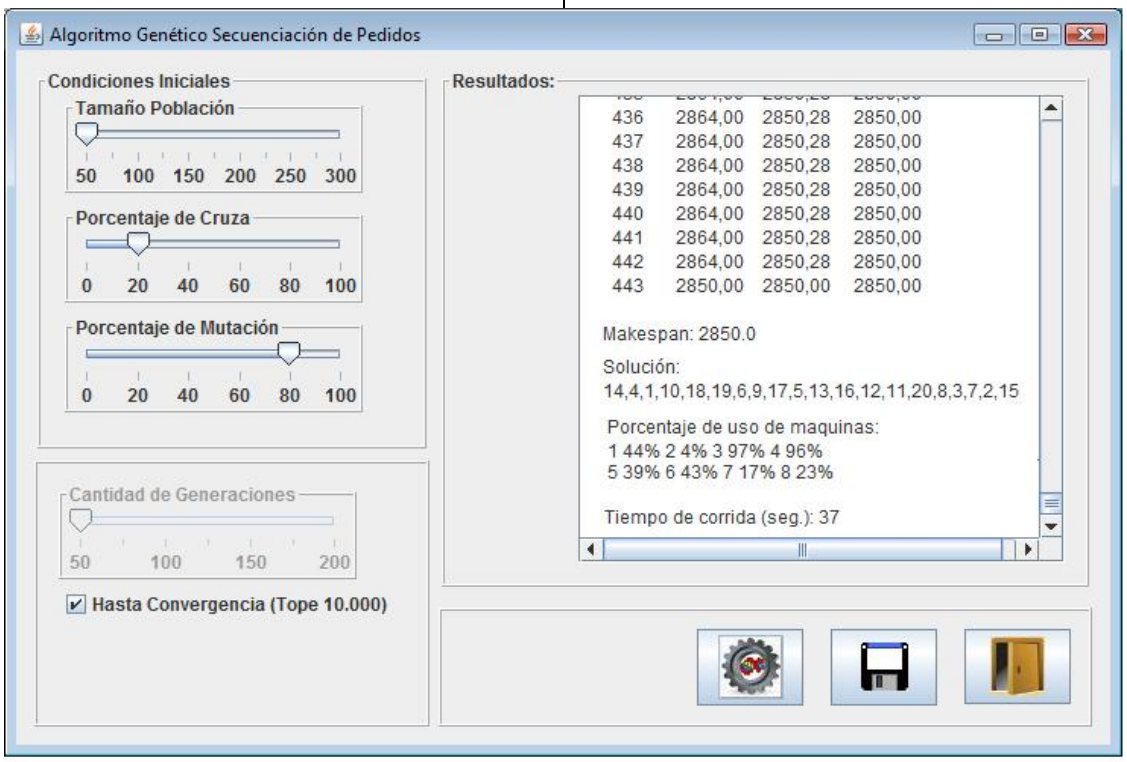


Figura 06. Máscara de operación del programa realizado.

Descripción del programa.

El algoritmo fue programado enteramente bajo la filosofía orientada a objetos con la plataforma Java SE (Standard Edition) en el entorno integrado de desarrollo NetBeans 6.9.1 IDE (Integrated Development Environment) de Sun Mycrosystem. Java dispone de una máquina virtual llamada JDK (Java Development Kit) para cada sistema operativo, y esto lo hace independiente de la plataforma. Específicamente el programa fue nutrido con ventajas amigables facilitadas por la interface gráfica de NetBeans (GUI – Grafical User Interface) (Petri, 2010).

Entre las bondades de este programa figuran:

- Sistematizar la programación de las actividades semanales del taller de piezas mecánicas que actualmente es realizado manualmente y sin ningún criterio comprobado de optimización.
- Facilitar el análisis de la sensibilidad de una programación sujeta a cambios, cosa que en la actualidad es obviamente impensable e impracticable.
- Agrega la capacidad de comprobar el impacto que tiene sobre la programación de actividades la agregación de nuevos tipos de piezas, cambios en los programas maestros para la fabricación de piezas, incorporación o des-incorporación de máquinas.
- Permite la sintonización con otros subsistemas de la empresa como el de ventas, compra de materias primas, mantenimiento, recursos humanos, embalaje y despacho.

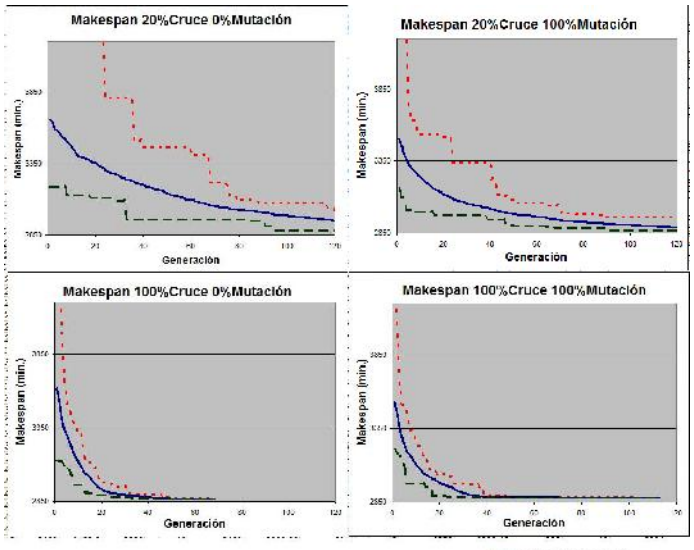
Aunque el algoritmo propuesto y la herramienta creada y probada ha sido desarrollada en un contexto específico para un caso de estudio muy particular, el enfoque que se ha seguido es

extensible a gran número de problemas de similares características. Todo esto favorecido por la filosofía orientada a objetos en que se basó este programa.

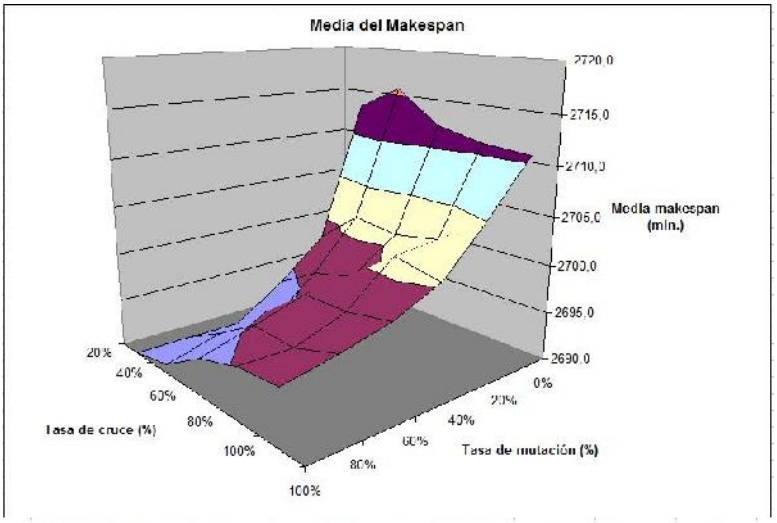
Este programa fue organizado en diversas clases y métodos, a continuación se mencionan de forma sucinta algunas clases y sus métodos con el propósito de tener una idea general del programa. Clase "Main" (recurso computacional para la organización del programa), clase "Plantilla" (hereda todas las propiedades necesarias para usar facilidades gráficas), métodos para modificar los dispositivos de porcentaje de cruce, porcentaje de mutación, método para la ejecución propia del algoritmo genético que comprende grosso modo la lectura de archivos de datos, construcción de estructura de datos para soportar la data leída, construcción de la estructura de datos para el soporte de la ejecución del algoritmo, ejecución del algoritmo según los dispositivos seleccionados, presentación de resultados. Clase "Individuo" (contiene todos los datos de todos los individuos o cromosomas), método para la generación de la población inicial, método para el cálculo de la aptitud de los individuos, método para la construcción de la lista de individuos que se cruzará, método para la construcción de la lista de individuos a ser mutados, método para realizar la cruza y método para realizar la mutación.

RESULTADOS

El problema ha sido resuelto utilizando una plataforma Java SE en NetBeans 6.9.1 IDE y en un ambiente Intel Core i3 CPU 2,03 GHz con una memoria de 4 GB y sistema operativo Windows 8. El algoritmo genético fue aplicado a 5 instancias de problemas reales formados por 20 pedidos cada uno, el tamaño de la población fue establecida en 50 individuos, el porcentaje de cruce fue establecido en 20%, 40%, 60%, 80% y 100% y el porcentaje de mutación establecido en 0%, 20%, 40%, 60%, 80% y 100% y para cada una de estas combinaciones fue ejecutado sistemáticamente 100 veces. De manera que para cada instancia fue ejecutado 3000 veces en total. Fueron calculadas la media y la desviación en cada caso para el tiempo total de producción, cantidad de generaciones, tasa de mejora y tiempo computacional de ejecución, así como la apreciación del comportamiento del peor individuo, individuo promedio y mejor individuo (Gráficas 01 y 02). Se presenta una tabla donde se muestran los resultados resumidos y se muestra una tabla con los resultados en detalle para una de las instancias del problema (Tablas 01 y 02).



Gráfica 01. Muestra el desempeño a través de cada generación del peor, promedio y mejor cromosoma para varias tasas de cruce y mutación.



Grafica 02. Media del makespan para 100 ejecuciones variando tasa de cruce y mutación de la instancia 05 del problema.

Tabla 01. Se muestran la media y la desviación para 100 ejecuciones del programa con tasa de 20% de cruce y 80% de mutación del makespan, generaciones, tiempo y tasa de mejoras en todas las instancias del problema.

	Instancia									
	01		02		03		04		05	
	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s
Tiempo total de producción (min.)	2848,9	12,8	3003,5	6,6	3207,3	5,2	3920,8	11,0	2698,0	11,5
Cantidad de generaciones	358,3	100,4	289,7	73,1	373,4	103,0	470,8	134,7	368,9	91,7
Tiempo de corrida (seg.)	100,6	29,5	89,4	25,7	545,0	161,1	317,9	93,5	90,3	22,2
Tasa de mejora (%)	23,4	1,2	17,7	0,9	15,4	0,8	17,5	1,1	15,1	0,6

DISCUSIÓN

Se pudo observar consistentemente para todas las instancias del problema un comportamiento regular con baja dispersión en las mediciones hechas al tiempo total de producción y a la tasa de mejora y por el contrario con una dispersión relativamente alta para el número de generaciones hasta la convergencia y para el tiempo de ejecución del programa, al apreciar las desviaciones de estas mediciones (Tablas 01 y 02). Al verificar la tendencia sostenida a la convergencia temprana con tasas de cruce altas e inversamente al aumentar las tasas de mutaciones, se obtienen resultados que coinciden con lo naturalmente esperado (Gráfica 01). Apreciar que tasas de cruce alrededor del 20% y tasas de mutación alrededor del 80% desembocaron en altas tasas de mejora en la reducción significativa del makespan (entre 10% y 20%) fue un hallazgo que contradice principios generalmente aceptados que colocan al cruce como un operador principal y a la mutación como secundario. Gen y Cheng revelan sin embargo, que al contrario de creencias convencionales en lo que respecta a problemas de scheduling o secuenciación, que aunque la mutación sea simplemente el intercambio de un gen permite explorar las vecindades de un individuo y así la mutación incrementa la probabilidad de mejorar las soluciónes (Gen y Cheng, 2000). Como contraparte, es de hacer notar que evidentemente un elevado valor de la tasa de mutación puede hacer que el algoritmo se convierta en una búsqueda aleatoria y se pierdan precisamente las bondades de la transmisión de las mejores características de una generación de soluciones a otra.

Los tiempos de ejecución estuvieron dentro de límites aceptables para las tasas comentadas (entre menos de un minuto y hasta seis minutos dependiendo en gran medida del tamaño de los pedidos considerados), al aumentar las tasas de cruce y de mutación también aumenta el tiempo computacional. Es de hacer notar que este programa representa solamente un prototipo y que trabajar sobre la eficiencia a nivel de código desembocará en un programa de acabado profesional con mayor eficiencia y por ende con menor uso del recurso tiempo de ejecución.

Tabla 02. Se muestran la media y la desviación para 100 ejecueciones del programa con diversas tasas de cruce y mutación del makespan, generaciones, tiempo y tasa de mejora.

INSTANCIA # 01														
Tiempo total de producción (min.)		% Mutación												
		0%		20%		40%		60%		80%		100%		
		$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	
	20%	2879,8	29,4	2856,6	13,9	2853,4	10,9	2851,3	11,9	2848,9	12,8	2848,6	8,2	
	40%	2881,0	32,8	2863,8	22,3	2859,4	18,5	2852,5	12,7	2852,6	8,5	2850,8	10,3	
% Cruce	60%	2871,6	25,8	2862,7	18,0	2857,0	14,5	2855,4	12,4	2853,3	13,7	2857,5	14,0	
	80%	2867,9	23,4	2862,0	18,6	2857,8	16,0	2857,5	17,8	2854,1	10,8	2851,1	10,4	
	100%	2868,5	23,7	2861,7	16,0	2858,7	17,2	2857,1	16,6	2857,4	17,8	2853,4	11,3	
Cantidad de generaciones		% Mutación												
		0%		20%		40%		60%		80%		100%		
		$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	
	20%	309,9	78,5	339,9	81,7	342,0	85,3	366,7	92,8	358,3	100,4	344,0	75,4	
	40%	164,5	39,5	174,2	44,2	179,5	44,9	181,4	46,5	170,8	44,7	185,4	55,7	
% Cruce	60%	114,9	24,8	119,6	26,4	120,8	27,8	124,6	33,6	123,7	29,5	116,4	27,7	
	80%	88,6	22,3	87,5	20,7	92,4	21,0	90,1	23,4	88,6	21,4	92,7	21,2	
	100%	71,1	14,7	74,3	19,2	69,4	15,5	76,1	17,7	74,7	17,4	74,7	19,8	
Tiempo de corrida (seg.)		% Mutación												
		0%		20%		40%		60%		80%		100%		
		$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	
	20%	44,65	12,11	61,02	15,70	78,68	21,24	101,12	28,36	103,57	29,51	119,39	29,13	
	40%	29,26	7,83	41,22	12,05	44,77	11,28	53,68	14,59	57,19	15,06	69,80	21,22	
% Cruce	60%	26,45	6,28	29,50	6,81	38,01	9,84	41,41	11,91	53,17	14,27	51,57	14,08	
	80%	23,21	6,22	30,75	9,79	30,73	6,96	35,29	9,16	36,14	8,77	40,50	9,42	
	100%	20,47	4,59	28,62	7,91	26,56	6,75	33,20	8,48	36,71	9,27	35,68	9,60	
Tasa de mejora (%)		% Mutación												
		0%		20%		40%		60%		80%		100%		
		$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	
	20%	22,73	1,33	23,19	1,14	23,51	1,09	23,49	1,03	23,44	1,21	23,51	1,22	
	40%	22,82	1,31	22,93	1,25	23,16	1,08	23,24	1,04	23,66	1,16	23,44	1,09	
% Cruce	60%	22,89	1,29	22,97	1,16	23,43	1,12	23,42	1,24	23,27	0,95	23,26	1,09	
	80%	22,97	1,22	23,23	1,05	23,37	1,11	23,22	1,04	23,44	1,02	23,41	1,12	
	100%	22,74	1,16	23,24	1,20	23,16	1,23	23,23	1,18	23,34	1,18	23,34	1,10	

CONCLUSIONES

La metodología abordada puede ser de gran utilidad e incluso de uso ordinario en talleres de mecanizado de piezas que funcione bajo pedido como el estudiado, puesto que evalúa un número muy grande de alternativas de secuenciación que bajo otras condiciones puede resultar laborioso o incluso ineficaz. Los resultados que se obtengan bajo este método siempre serán factibles al satisfacer todas las restricciones y pueden ser cercanos al óptimo o incluso óptimos, constituyendo en una herramienta en la toma de decisiones de nivel operativo de la producción. Se comprobó que los algoritmos genéticos son una eficaz herramienta para obtener buenas soluciones a problemas con un espacio de soluciones amplio, ya que es manifiesta la imposibilidad de conocer si el algoritmo logró obtener la solución óptima en algún caso debido a que precisamente el(los) óptimo(s) de las instancias particulares del problema tratado son desconocidos.

Se verificó que los algoritmos pueden ser enriquecidos positivamente al ser asistidos por la simulación de eventos discretos en el caso de tener que lidiar con una función de optimización compleja como la de este problema.

Investigaciones futuras podrían indagar sobre la calibración exacta de los operadores genéticos usados, prueba de variantes de los operadores de cruce y mutación, también variando el tamaño de la población e incorporación de selección por elitismo, como con la hibridación con otras técnicas meta-heurísticas.

REFERÊNCIAS

Abdelmaguid, Tamer F. (2010). Representations in genetic algorithm for the job shop scheduling problem: a computational study. 2010. Journal of Software Engineering and Applications. Vol. 3, Núm. 12, Pág. 1155-1162. USA.

Adams, Joseph; Balas, Egon y Zawack, Daniel. (1988). The shifting bottleneck procedure for job shop scheduling. Management Science. Vol. 34, Núm. 3, Pág. 391-401. USA.

Applegate, David y Cook, William. (1991). A computational study of the job-shop scheduling problem. Operation Research Society of America. ORSA Journal of Computing. Vol. 3, Núm. 2, Primavera de 1991. USA.

Baker, Kenneth R. y Trietsch, Dan. (2009). Principles of sequencing and scheduling. John Wiley & Sons, Inc., Hoboken, New Jersey, USA.

Beck, J. Christopher; Davenport, Andrew J. y Fox, Mark S. (1997). Five pitfalls of empirical scheduling research. (CP'97) Proceedings of the Third Internacional Conference on Principles and Practices of Constraint Programming 1997. Desde 29 de octubre hasta 1 de noviembre, 1997. Schloss Hagenberg, Austria.

Bierwirth, Christian. (1995). A generalizad permutation approach to job shop scheduling with genetic algorithm. Journal Operations Research-Spektrum. Vol. 17, Núm. 2-3, Págs. 87-92. Springer-Verlag Berlin Heidelberg, Alemania.

Brucker, Peter; Jurisch, Bernd y Sievers, Bernd. (1994). A branch and bound algorithm for the job-shop scheduling problem. Discrete Applied Mathematics. Elsevier-Science Publishers B.V. Pág. 107-127. USA.

Brucker, Peter. (2007). Scheduling algorithms. Quinta edición. Springer-Verlag Berlin Heidelberg, Alemania.

Carlier, Jacques y Pinson, Eric. (1989). An algorithm for solving the job-shop problem. Management Science. Vol. 35, Núm. 2, Febrero, 1989. The Institute of Management Science. Pág. 164-176. USA.

Coello C., Carlos A. (2010). Introducción a la computación evolutiva (notas de curso). 2010. Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional CINVESTAV-IPN Departamento de Computación. Ciudad de México, México.

Duvivier, David; Preux, Philippe y Talbi, El-Ghazali. (1996). Genetic algorithms applied to the job-shop. Proceedings Workshop on Production Planning and Control. Del 9 al 11 de septiembre, 1996. FUCaM'96. Facultés universitaires catholiques de Mons. Mons, Bélgica.

Falkenauer, Emanuel y Bouffouix, Stephane. (1991). A genetic algorithm for job shop. Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Pág. 824-829.

Fissgus, Ursula. (2000). Scheduling using genetic algorithms. 20th IEEE International Conference on Distributed Computing Systems, del 10 al 13 de abril, 2000. Taipei, Taiwan.

Gen, Mitsuo y Cheng, Runwei. (2009). Genetic algorithms & engineering optimization. A Wiley Interscience Publications, John Willey & Sons, INC. New York, USA.

Gestal, Marcos; Rivero, Daniel; Rabuñal, Juan Ramón y Dorado, Julián. (2010). Introducción a los Algoritmos Genéticos y la Programación Genética. Universidade da Coruña, Servizo de Publicacións. Monografías, nº 140. Coruña, España.

Giffler, Bernard y Thompson, Gerald. L. (1960). Algorithms for solving production-scheduling problems. Operations Research Vol. 8, Núm. 4, Julio-Agosto 1960. Pág. 487-503. USA.

Goldberg, David E. (1989). Genetic algorithms in search, optimisation, and machine learning. Addison-Wesley Publishing Company, INC. Reading, USA.

Goncalves, José F.; de Magalhães, Jorge J. y Resende, Maurício R. (2002). A hybrid genetic algorithm for the job shop scheduling problem. AT&T Labs Research Technical Report TD-5EAL6J, September 2002.

Goncalvez, Jose F. y Resende, Mauricio G. C. (2011). A biased random-key genetic algorithm for job-shop scheduling. AT&T Labs Research Technical Report. Florham Park, Nueva Jersey, USA.

Hasam, Kamrul S. M.; Sarker, Ruhul y Comforth, David. (2007). Hybrid genetic algorithm for solving job-shop scheduling. 6th IEEE/ACIS International Conference on Computer and Information Science. ICIS 2007. 11-17 Julio, 2007. Melbourne, Australia.

Holsapple, Clyde W.; Jacob, Varghese S.; Pakath, Ram y Zaveri, Jigish S. (1993). A genetics-based hybrid scheduler for generating static schedules in flexible manufacturing contexts. IEEE Transactions on Systems, Man and Cybernetics. Vol. 23 Núm. 4. USA.

Huang, Di-Wei y Lin, Jimmy. (2010). Scaling populations genetic algorithm for job shop scheduling problems. Cloud Computing, Second International Conference. del 30/11 al 03/12, 2010. Proceedings IEEE. First International Workshop on Theory and Practice of MapReduce. Indianapolis, Indiana, USA,

Kammer, Marnix; van der Akker, Marjan y Hoogeveen, Han. (2010). Identifying and exploiting commonalities for the job-shop scheduling problem. Technical Report UU-CS-2010-002 January 2010. Department of Information and Computing Sciences. Utrecht University, Utrecht, Holanda.

Law, Averill M. (2007). Simulation modeling and analisys. Cuarta edición. McGraw-Hill Companies, Inc. New York, USA.

Lenstra, Jan K.; Rinnooy Kan, Alexander H.G. y Broucker, Peter. (1977). Complexity of machine scheduling problems. Annals of Disccrete Mathematics 1. 1977, Pág. 343-362. North Holland Publishing Company. Holanda.

Lestan, Zoran; Brezocnik, Miran; Buchmeister, Borut; Brezovnik, Simon y Balic, Joze. (2009). Solving job-shop

scheduling problem with a simple genetic algorithm. Annals of DAAAM & Proceedings. Vol. 2009, Pág. 197-205. DAAAM International Vienna. Eslovenia.

Li, Ye y Chen, Yan. (2010). A genetic algorithm for job-shop scheduling. Journal of Software. Vol. 5, Núm. 3, Pág. 269-274. 2010. Academy Publisher.

Magalháes-Mendes, Jorge. (2013). A two-step optimization approach for job shop scheduling problem using a genetic algorithm. Working paper 06.2013. Departamento de Engenharia Civil CIDEM. Instituto Superior de Engemharia do Porto. Porto, Portugal.

Manne, Alan S. (1960). On the job shop scheduling problem. Operations Research. Vol. 8, Núm. 2, Marzo-abril, 1960. Pág. 219-223. USA.

Mattfeld, Dirk C. y Bierwirth, Christian. (2004). An efficient genetic algorithm for job shop Scheduling with tardiness objectives. European Journal of Operational Research. Núm. 155, Pág. 616-630. Elsevier Journals B.V. USA.

Medina D., Rosa; Pradenas R., Lorena y Parada D., Víctor. (2011). Un algoritmo genético para el problema de Job Shop Flexible. Ingeniare. Revista chilena de ingeniería. Vol. 19, Núm. 1. Arica, Chile.

Michalewicz, Zbigniew. (1996). Genetic algorithms + data structures = evolution programs. Springer-Verlag London Limited. Reino Unido.

Mitchell, Melanie. (1996). An introduction to genetic algorithms. A Bradford Book The MIT Press Massachusetts Institute of Technology. Massachusetts, USA.

Papadimitriou, Christos H. y Steiglitz, Kenneth. (1998). Combinatorial optimization algorithms and complexity. Dover Publications, INC. Mineola. New York, USA.

Pazos A., José J.; Suárez G., Andrés y Díaz R., Rebeca P. (2008). Teoría de colas y simulación de eventos discretos. Pearson Educación, S.A. Madrid, España.

Petri, Jürgen. (2010). NetBeans platform 6.9 developer's guide. Packt Publishing Ltd. Road Olton, Birmingham, Reino Unido.

Pinedo, Michael L. (2009). Planning and scheduling in manufacturing and services. Segunda edición. Springer Science+Business Media, LLC. NewYork, USA.

Pinedo, Michael L. (2012). Scheduling. theory, algorithms, and systems. Cuarta edición. Springer Science+Business Media, LLC. NewYork, USA.

Sivanandam, S. N. y Deepa, S. N. (2008). Introduction to genetic algorithms. Primera edición. Publicaciones Springer-Verlag Berlin Heidelberg New York. Inglaterra.

Vaessens, Rob J. M.; Aarts, Emile H. L. y Lenstra, Jan K. (1996). Job shop scheduling by local search. INFORMS Journal on Computing. Vol. 8, Núm. 3, Pág. 302-317. 1996.

Vaghefinezhad, Sayedmohammadreza y Wong, Kuan Y. (2012). A Genetic Algorithm Approach for Solving a Flexible Job Shop Scheduling Problem. 14th International Conference on Computer Modelling and Simulation. De: 28-30/03, 2012. Cambridge, Reino Unido.

Varela, Ramiro. (2008). Nuevas tendencias y retos en métodos heurísticos para problemas de scheduling. Congreso Internacional de Cómputo en Optimización y Software, Memorias del CICOS08, 25-27 Junio 2008. México.

Williams, Edward J. y Ahitov, Igal. (1996). Scheduling analysis using discrete event simulation. Proceedings of the 29th Annual Simulation Symposium. Abril, 9-11, 1996 New Orleans, Louisiana, USA.

Werner, Frank. (2011). Genetic algorithms for shop scheduling problems: A survey. Preprint 11/31, Faculty of Mathematics, Otto-von-Guericke-Universität Magdeburg. Magdeburgo, Alemania.

Yamada, Takeshi y Nakano, Ryohei. (1996). Job-shop scheduling by simulated annealing combined with deterministic local search. Meta-heuristics: theory & applications. Kluwer Academic Publisher, Pág. 237-248. 1996. Boston, Massachusetts, USA.

Yamada, Takeshi y Nakano, Ryohei. (1997). Genetic algorithms for job-shop scheduling problems. Proceedings of Modern Heuristic for Decision Support, del 18 al 19 de marzo, 1997. Pág. 67–81. Seminario UNICOM. Middlesex, Londres, Inglaterra.

Zäpfel, Günther; Braune, Roland y Bögl, Michael. (2010). Metaheuristic search concepts. a tutorial with applications to production and logistics. Springer-Verlag Berlin Heildelberg. USA.

Autores

Enrique Pérez Pérez. Ingeniero en Información, Universidad Tecnológica del Centro. Tesista de la Maestría Matemática y Computación, Universidad de Carabobo. Profesor Asistente a Dedicación Exclusiva, Departamento de Investigación Operativa, Escuela de Ingeniería Industrial, Facultad de Ingeniería, Universidad de Carabobo.
E-mail: eperez@uc.edu.ve

Ilse Josefina Pérez Castillo. Ingeniero Industrial, Universidad de Carabobo. Tesista de la Maestría en Ingeniería Industrial, Universidad de Carabobo. Profesora Asistente a Tiempo Completo, Departamento de Investigación Operativa, Escuela de Ingeniería Industrial, Facultad de Ingeniería, Universidad de Carabobo.
E-mail: ijperez@uc.edu.ve

Manuel Elías Jiménez Bahri. Ingeniero Industrial. Magister en Ingeniería Industrial, Universidad de Carabobo. Profesor Agregado a Dedicación Exclusiva, Departamento de Investigación Operativa, Escuela de Ingeniería Industrial, Facultad de Ingeniería, Universidad de Carabobo.
E-mail: mejb11@gmail.com

Recibido: 12-02-2014 **Aceptado:** 27-04-2014