Aplicación de algoritmos genéticos para la programación de tareas en una celda de manufactura

Applying genetic algorithms for programming manufacturing cell tasks

Efredy Delgado A.,[1] Carlos Julio Cortés,[2] Óscar Duarte V.[3]

RESUMEN

El propósito del trabajo fue plantear una metodología para la programación de tareas de una celda de manufactura, basada principalmente en algoritmos genéticos. La celda de manufactura fue modelada como una línea de flujo, y se desarrollaron cálculos que permiten saber el *makespan* usando una heurística adaptada de una librería realizada para algoritmos genéticos, esto implantado en un ambiente de programación C++ builder. Se resolvieron problemas diversos, considerando distintos tamaños de lista en trabajos y máquinas, los cuales fueron comparados con los resultados de otras heurísticas. Los futuros trabajos pueden considerar la programación de tareas en celdas de manufactura con lotes de diferentes volúmenes.

PALABRAS CLAVE: manufactura flexible, celdas de manufactura, programación de líneas de producción, algoritmos genéticos.

ABSTRACT

This work was aimed at developing computational intelligence for scheduling a manufacturing cell's tasks, based mainly on genetic algorithms. The manufacturing cell was modelled as being a production-line; the makespan was calculated by using heuristics adapted from several libraries for genetic algorithms computed in C++ builder. Several problems dealing with small, medium and large list of jobs and machinery were resolved. The results were compared with other heuristics. The approach developed here would seem to be promising for future research concerning scheduling manufacturing cell tasks involving mixed batches.

KEYS WORDS: flexible manufacturing, manufacturing cells, production-line scheduling, genetic algorithms.

Recibido: agosto 27de 2004
Aceptado: abril 15 de 2005

Introducción

Hasta hace algunos años se lograron incorporar a la actividad productiva las herramientas identificadas como Inteligencia Artificial, que pretenden emular el comportamiento de organismos biológicos inteligentes, para ajustarlos a toma y ejecución de decisiones en diferentes ambientes. En el manejo productivo es crucial lograr ejecuciones de costo y tiempo efectivas en un proceso industrial, es así como la planificación de decisiones en un ambiente industrial automatizado requiere de especial detalle. Tópicos como el modelamiento y la optimización de un conjunto de problemas logísticos dan nacimiento a temáticas de diferente aplicación en ambientes industriales de producción, como:

- Secuencia y programación de tareas en celdas automatizadas (FMS).
- Programación de almacenamiento con amortiguación en etapas intermedias.
- Secuencia de operaciones en un sistema de ensamble.
- Programación de tareas en máquinas paralelas con líneas muertas.

Los primeros tres modelos son bastante aplicados en manufactura.

El trabajo que se plantea está enmarcado dentro del proyecto de investigación «Implementación de celda expe-

1 Ingeniero electricista, MSc Automatización Industrial, egresado posgrado Automatización Industrial Universidad Nacional de Colombia – Bogotá. e-mail: ser-o@excite.com

2 Ingeniero mecánico, MSc Manufactura, MSc Industria y Tecnología, profesor adscrito al Departamento de ingeniería Mecánica y Mecatrónica, Universidad Nacional de Colombia – Bogotá, e-mail: ccortes@ing.unal.edu.co y ccortes@latino.net.co

3 Ingeniero electricista, MSc Automatización Industrial, doctor en Informática, profesor adscrito al Departamento de Ingeniería Eléctrica y Electrónica, Universidad Nacional de Colombia – Bogotá, e-mail: oduarte@ing.unal.edu.co

rimental de manufactura flexible» (convocatoria DINAIN 2000), y el objetivo de esta propuesta es discutir la planificación que se emite en el ambiente de FMS y presentar un acercamiento jerárquico a este problema; de forma tal, que la programación de tareas en una FMS se pueda visualizar desde una perspectiva general, apoyada por herramientas de inteligencia computacional, mediante una estrategia eficiente de enrutamiento y programación de tareas (*scheduling*), que permita la labor adecuada del sistema de control de la celda flexible de manufactura, ya que, es necesario el reconocimiento inicial de la planeación de eventos para definir la actividad del controlador jerárquico de la celda.

Para la realización del trabajo que se reporta en este documento, se ha convenido entre director y estudiante realizar una convención previa, ya que la literatura disponible alrededor de este tema se enmarca como *scheduling*, se adoptó de esta forma traducirlo al español como "programación de tareas".

El objetivo central en la realización de este trabajo es el desarrollo de un algoritmo implantado en *software* que permitiera obtener la programación de tareas en una celda de manufactura flexible (FMS), mediante la exploración de herramientas de inteligencia computacional. Para ello se inició la revisión de métodos tradicionales de programación de tareas, para conocer sus parámetros y características, así como su versatilidad; posteriormente se exploraron herramientas de inteligencia computacional (lógica difusa, algoritmos genéticos, redes neuronales) y su comportamiento frente a la solución de este tipo de problemas. Se seleccionó como herramienta de búsqueda algoritmos genéticos por el planteamiento que estos sugieren en la solución de problemas de programación de tareas; adicionalmente por la revisión bibliográfica, en la que es muy frecuente su utilización por parte de investigadores internacionales en investigación de operaciones.

Los resultados ofrecen una buena aproximación frente a otras heurísticas, como es el caso de la aleatoria, la CDS (Vollmann, 1997) y la propuesta por Onwubolu (Onwubolu, 2000).

Metodologías previas

La metodología en la investigación de secuencias y de programación de tareas es muy amplia; conviene revisar los procedimientos a los que han acudido diversos autores en la búsqueda de soluciones que se plantean para problemas de optimización de recursos en la realización de tareas.

Los primeros esfuerzos han sido desarrollados por Johnson (1954), Kusiak(1989) tomó los desarrollos de Johnson en donde incluye las tripletas "número de operación/tiempo de proceso/número de máquina", Campbell, Dudek y Smith (CDS)(Vollmann, 1997), en donde se toman los trabajos realizados por Johnson y se modifica el trabajo de aquél para varias pseudo máquinas; el trabajo de Onwubolu (2000), toma los conceptos de algoritmos genéticos para hallar *makespan* de una celda de manufactura. A lo largo del artículo se presentarán algunos otros autores que se toman como referencia

Ambiente de producción

El problema de programación de tareas trata sobre la asignación de recursos limitados a ciertas tareas u operaciones a través de determinado período de tiempo (Gershwin, 1994 y Pinedo, 1995).

La programación de operaciones *(scheduling*) es un problema de optimización que puede tener uno o más objetivos, el cual se presenta con frecuencia en sistemas de producción convencionales automatizados y es un problema común, donde está involucrado el tomar decisiones con respecto a la mejor asignación de recursos a procesos de información donde se tienen restricciones de temporalidad.

La programación de tareas y el control del flujo de trabajos a través de un ambiente de producción es esencial en los procesos de manufactura (Gideon, 1995). Una programación adecuada puede reducir significativamente los costos de producción y reducir los tiempos de proceso permitiendo cumplir con los compromisos de entrega a tiempo. La mayoría de problemas de esta área son problemas de optimización combinatoria y una gran parte de ellos pertenecen a la clase de problemas NP-hard. Los problemas NP-hard son un subconjunto de la clase NP (problemas para los que no se puede tener una solución en tiempo polinomial para todas sus instancias) con la característica de que todos los problemas de ésta clase pueden ser reducidos a NP (Pinedo, 1995). Los problemas para los que se puede encontrar un algoritmo de solución en tiempo polinomial forman la clase P, que es un subconjunto de la clase NP (Garey y Johnson, 1979).

En los últimos años se han propuesto un gran número de enfoques para modelar y solucionar los diferentes problemas de programación de tareas, con diferentes grados de éxito. Entre estos enfoques podemos mencionar la programación matemática, reglas de despacho, sistemas expertos, redes neuronales, algoritmos genéticos, búsqueda tabú, recocido simulado, lógica difusa, entre otros (Jones y Rabelo, 1998).

En todos los problemas de programación se considera que el número de tareas y el de agentes (máquinas) es finito. A continuación se considera la nomenclatura a utilizar:

n = número de trabajos (tareas)

m = número de máquinas

j = índice que se refiere a los trabajos o tareas

i= índice que se refiere a las máquinas o agentes.

Cuando un trabajo requiere un número de pasos de procesamiento u operaciones, entonces se usa el par $(_{i,j})$ que se refiere al paso de procesamiento u operación del trabajo (o tarea) ***j*** sobre la máquina o agente ***i***.

Las siguiente notación se asocia con el trabajo j:

Tiempo de procesamiento (processing time) p_{ij}.

Fecha de liberación (release date) r_j.

Fecha comprometida o deseada (due date) y deadline d_j.

Peso o ponderación w_j..(Kusiak, 1990)

El tipo de programación que tiene que ver con este documento es la programación de tareas, en el cual los rangos de trabajo son pequeños, complejos, con un ajuste a la medida de las situaciones de un taller para altas velocidades de proceso. La problemática encierra, desde la manufactura discreta de partes, a los procesos de flujo continuo. Un problema de programación se describe por la siguiente tripleta: a/b/g

En donde :

a = Es un simple atributo que describe el ambiente de la máquina.

b= Pueden ser varios atributos que proporcionan información sobre las características del procesamiento y restricciones.

g= Es un simple atributo que contiene el objetivo a ser minimizado

Modelo del problema

La "secuenciación" o *sequencing* es el proceso de definir el orden en el cual los trabajos[4] van a ser procesados en una máquina. Como se había especificado antes, el *scheduling* o programación, es el proceso de adicionar los tiempos de arranque y finalización para las ordenes de trabajos dictadas por la secuencia, previamente realizada.(Gershwin 1994).

Los objetivos que se persiguen en la programación de eventos en la celda de manufactura, son:

1. Cumplir con las fechas de vencimiento o deseadas (*duedate)*.
2. Minimizar plazos.
3. Minimizar el tiempo o costo de preparación (*set-up*).
4. Minimizar el inventario del trabajo en proceso.
5. Maximizar la utilización de las máquinas o de la mano de obra (el último objetivo es controversial, porque el simple hecho de mantener todo el equipo y/o los empleados ocupados no siempre es la manera más eficien te de manejar el flujo a través del proceso)(Chase, 2000).
6. Realizar la programación de eventos en el controlador jerárquico de la celda

La celda de manufactura propuesta, que consiste en un conjunto de *n* tareas o trabajos $\{J_1, J_2 \ldots J_n\}$ a realizar sobre un conjunto de m máquinas $\{M_1, M_2 \ldots M_m\}$ en dicho orden, por ejemplo, que un trabajo J_j, j = 1...n constituida por una secuencia de m operaciones O_{1j} ,... O_{mj} donde O_{ij} debe ser procesado en la máquina M_i en un tiempo ininterrumpido de procesamiento dado: $p_{ij..}$ Cada máquina M_i, i = 1,... m, puede procesar a lo sumo un trabajo al tiempo, y cada trabajo J_j, j =1...n, es procesado a lo sumo por una máquina al tiempo. Además, se toma el tiempo C_{ij} , en que se completa la operación O_{1j} . El objetivo es entonces, encontrar una secuencia de las tareas y su programación (schedule), que minimice el máximo tiempo de finalización de todos los trabajos a ejecutar, es decir: $C_{max} = \max_{ij} C_{ij}$, también denominado makespan. En la Figura 1 se muestra la secuencia en la cual las tareas a ser procesadas constituyen un producto.

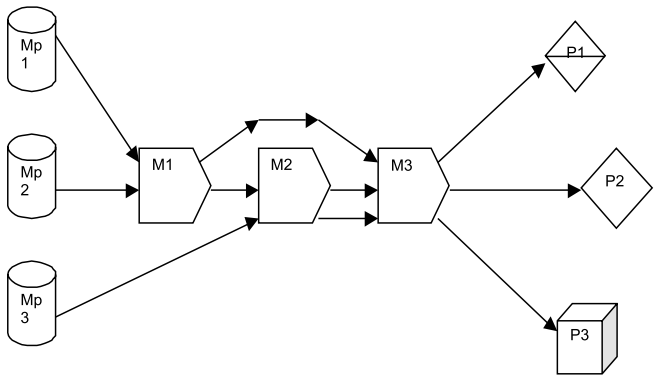


Figura 1. Secuencia en la cual se realiza un producto a lo largo de una estructura Flow Shop

Matemáticamente se podría hablar de un problema que requiere encontrar la permutación de las tareas para ser resuelto. Este tipo de problemas de secuenciamiento de sistemas de manufactura es clasificado de NP-hard, como se había citado anteriormente. Un problema NP-hard se presenta cuando un algoritmo que intenta solucionarlo, aumenta su tiempo de ejecución, en el peor de los casos, de forma exponencial al tamaño del problema (Johnson,1979). El reto está en encontrar algoritmos que exploren favorablemente la estructura matemática del problema, para que sean capaces de obtener una respuesta para la mayoría de las instancias del mismo, en tiempos de ejecución relativamente pequeños.

Estrategia de solución

En la búsqueda para hallar una solución óptima a un problema planteado, el algoritmo genético funciona de la siguiente manera: una población de cromosomas (los padres potenciales de una nueva población) se mantiene a lo largo del proceso evolutivo. A cada uno de ellos se le adjudica un valor de *fitness* que está rela-

4 En la literatura de programación de tareas, los trabajos se componen de un número finito de operaciones realizadas en una secuencia, y las operaciones están dadas por la tecnología dispuesta para el producto.

cionado con el valor de la función objetivo a optimizar. Cada cromosoma está representando un punto del espacio de búsqueda del problema.

Un grupo de cromosomas son elegidos para ser los padres de una nueva solución mediante el mecanismo de cruce. La operación de cruce da un grupo de configuraciones "hijos" y se calculaban sus valores de *fitness*. Estas soluciones reemplaza las que eran elegidas al azar. Este proceso se repite hasta alcanzar un criterio más o menos lógico para detener la simulación.

En la Figura 2 se muestra un esquema básico de los algoritmos genéticos.

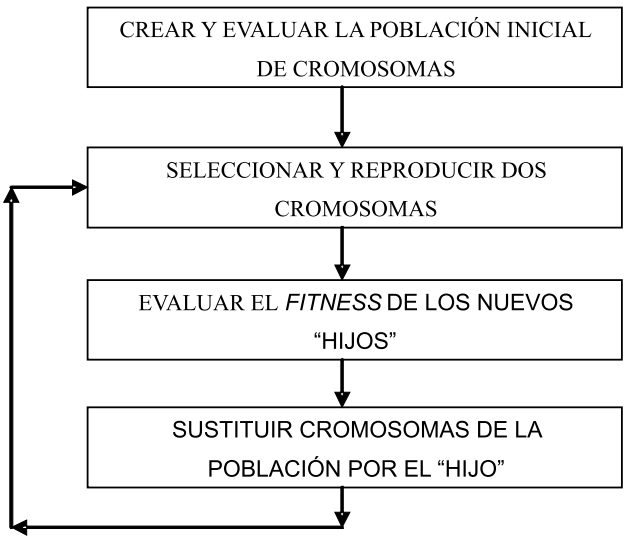


Figura 2. Esquema básico de los algoritmos genéticos.

Para el problema de flow shop, el número del trabajo J_1, J_2 ...J_n ocupa en la cadena de secuencia una posición denominada *posición de secuencia*, refiriéndose a ella como el orden en el que se realiza una tarea dentro de la línea de flujo de producción. La longitud de la *cadena X* representa el conjunto de los trabajos considerados dentro del problema, los que a su vez dependen del número de máquinas que los procesan, pero que no son significativos a la hora de ejecutar y evaluar la función objetivo dentro del algoritmo genético.

Por ejemplo, para una secuencia de producción en la que existen cinco trabajos cuyo orden de fabricación es 53421, quiere decir que el primer trabajo en ser realizado es el número cinco, el segundo a ser realizado es el tres, el tercero es el número cuatro y así sucesivamente. Esta cadena se representaría de la siguiente forma:

5	3	4	2	1

Dentro de la codificación realizada para el análisis presentado aquí, tomada de la clase *vector real* de la librería antes mencionada, al ser ejecutado el algoritmo, este mostraba en la cadena de salida la repetición del trabajo que tomaba el menor tiempo, para lo cual se ejecutó la estrategia que a continuación se describe. Abajo se presenta la forma en que se crea el genoma cuyos valores se ajustan a un arreglo de productos; su mínimo será de uno.

Descripción de la interfaz

La librería para algoritmos genéticos fue implantada dentro de un ambiente creado en C++ Builder, cuya finalidad es recoger fácilmente los datos de tiempos, número de máquinas, número de trabajos, que son suministrados por el usuario así mismo, se puedan modificar datos propios del algoritmo como son tamaño de población y número de generaciones.

Para acceder a esta opción dentro de la interfaz, que se ha denominado AG ANALISIS y que se muestra en la Figura 3, es necesario abrir el menú denominado *DATOS,* que mostrará una ventana de *Ingreso de datos*.

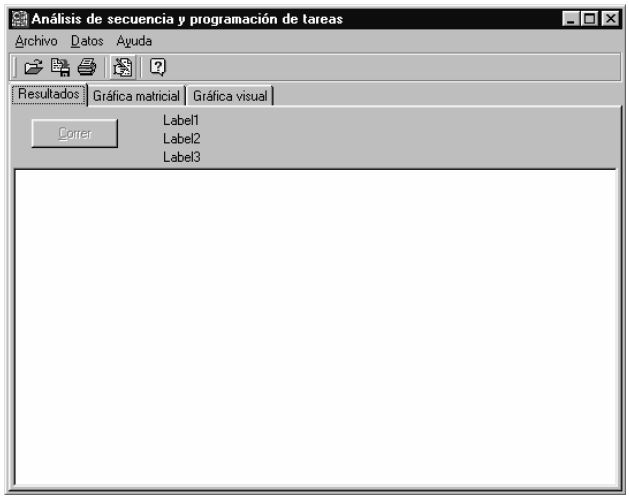


Figura 3 Presentación de la primera pantalla para la interfaz AG Análisis.

Además, se deben ubicar los datos de la cadena inicial de la secuencia, cuya finalidad es mostrar al algoritmo genético un individuo inicial con el que se puedan ¨derivar¨ soluciones óptimas dentro del espacio de búsqueda. Después de este paso, se finaliza el ingreso de datos y el programa los procesa arrojando algunos resultados, los cuales los expone de la siguiente forma: uno numérico donde se presenta la secuencia de trabajos y su respectivo resultado en la función objetivo, y por último el algoritmo presenta los cinco mejores resultados clasificados desde el menor hasta el mayor. La otra manera de presentación corresponde a la programación de tareas de acuerdo a las unidades de tiempo requeridas en cada una de las máquinas, expuestas en diferentes colores, con algunos espacios en blanco que corresponden a los tiempos de espera para la iniciación de trabajos en la respectiva máquina.

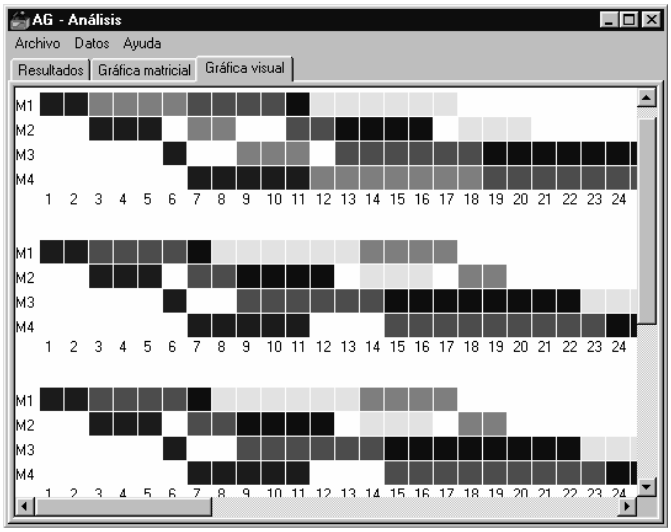


Figura 4 Gráfica visual, resultado de la programación de tareas por máquina.

Resultados y comparaciones

Casos estudiados

Para comprobar la validez y el comportamiento del *software* AG ANÁLISIS se tomaron varios casos sugeridos de la bibliografía revisada y reciente, los que se han dividido por su tamaño en tres:

- Pequeño, compuesto por 5 trabajos, 4 máquinas.
- Mediano, compuesto por 10 trabajos, 10 máquinas.
- Grande, compuesto por 15 trabajos, 25 máquinas.

El primer caso es tomado de Gershwin (1994), cuyos datos se presentan en la siguiente tabla:

Tabla 1. Datos para un problema de tamaño pequeño

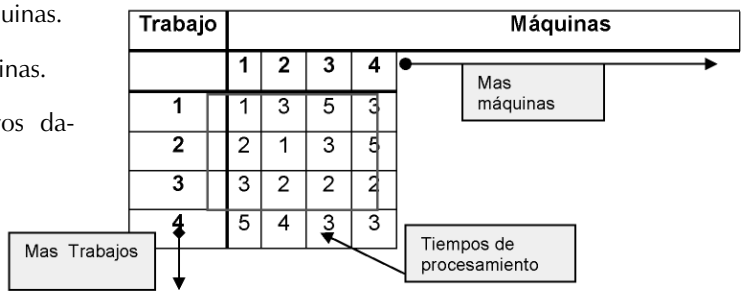
	Tiempo de proceso sobre cada máquina Pij (unidades de tiempo)			
Trabajo	1	2	3	4
J1	2	3	1	5
J2	4	2	6	9
J3	1	4	8	5
J4	6	3	5	2
J5	4	2	3	7

Para este caso, por ser de tamaño reducido, se buscará como objetivo minimizar el *makespan* frente a dos heurísticas: una de ellas es la aleatoria, en la que todos los trabajos tienen la misma probabilidad, tomándose esta como base para comparar las demás reglas; otra heurística es la propuesta por Campbell, Dudek y Smith (CDS) (Vollmann, 1997), en la que aplican el algoritmo de Johnson, con la que se pueden resolver casos en los que se presentan M>2. Las M máquinas se pueden dividir en dos máquinas falsas o pseudomáquinas, estas tendrán como fin agrupar los datos de tiempos de proceso de las máquinas reales.

Otra heurística que se puede establecer como parámetro de comparación es la desarrollada por Onwubolu para dos máquinas que realizan seis trabajos en un ambiente de línea de flujo (flow shop), y los datos con los cuales realiza su programación de tareas se presentan en la siguiente tabla. En este caso se calcula un makespan de 115 unidades de tiempo, en 7,5 segundos de procesamiento de CPU AG ANÁLISIS toma 3.850 segundos con un makespan de 113 unidades de tiempo (para una población de diez individuos y 80 generaciones), adicionalmente se realizaron otras pruebas con problemas de diferentes tamaños y sus respectivos tiempos de CPU y tiempo de makespan se describen en la Tabla 3.

Se puede ver que la heurística desarrollada, frente al trabajo de Onwubolu, tiene mayor efectividad en hallar un *makespan* óptimo (113 unidades de tiempo con diferentes secuencias) aunque tome un mayor tiempo; además, este resultado se puede lograr a partir de una población inicial mayor a 60 individuos y 10 generaciones.

Tabla 2. Detalle de la Tabla 3 de datos para problemas de diferentes tamaños, en este caso para 4 máquinas y 4 trabajos.



El compilado de todos los conjuntos de problemas de prueba se exponen en la siguiente Tabla 3:

Tabla 3. Comparación de la heurística desarrollada tomando en cuenta el tiempo que toma CPU vs tamaño del problema.

Maquinas (M)	Trabajos (n)	Tiempo CPU	
		AG ANALISIS Sin graficos [5] (*makespan*) / (seg)	AG ANALISIS Con graficos (*makespan*) /(seg)
2	6	115 / 3,620	55,580
4	4	20 / 3,350	20 / 5,160
5	10	41 / 7,420	41 / 7,360
10	15	79 / 1 50,400	78 / 31,310
15	25	150 / 132,50	150 / 57,320

Conclusiones y recomendaciones

En un problema como la programación de tareas dentro de una celda de manufactura modelada como una línea de flujo de producción (*flow shop*) para un número relativamente alto de máquinas y trabajos, este se constituye en un ejemplo de lo que se considera como un problema NP-hard.

El modelo de la celda de manufactura se precisa como una línea de flujo de producción debido al trabajo que se plantea dentro del proyecto de investigación "Implementación de celda experimental de manufactura" (Convocatoria DINAIN 2000), y el objetivo es la discusión de la planeación que se requiere dentro de un ambiente de sistemas de manufactura.

La aproximación basada en algoritmos genéticos se formalizó debido a las ventajas que estos tienen frente a otras herramientas de inteligencia computacional, como es el caso de lógica difusa y redes neuronales, que presentan dificultades para la adecuación de las reglas de programación de tareas y la selección correcta de las mismas, de acuerdo a la situación que se presente, y tomar las acciones o tareas a realizar con relación a una "lista" de prioridades.

Dentro de los objetivos específicos más importantes y que lograron ser cumplidos, se pueden citar los siguientes: el establecimiento de un modelo que permitiera ajustarse a la posibilidad de construir o formalizar una celda de manufactura, como es el caso de la línea de flujo o *flow shop*.

Como segundo logro, gracias a la librería de clases en C++ para algoritmos genéticos (Duarte, 2001), se concretó una estrategia de codificación y decodificación denominada "saltos", la cual permite que el espacio de búsqueda permanezca dentro del conjunto o el arreglo de tipos de productos que se han asignado, los que a su vez están relacionados con un arreglo de tiempos de procesamiento.

En tercer lugar, el método propuesto prueba ser más efectivo en comparación con heurísticas tradicionales y con trabajos recientemente realizados, por varias razones:

1. El *software* sobre el cual se montó la heurística (AG ANÁLISIS), requiere de menor tiempo para presentar la secuencia y la programación de tareas, de manera numérica y gráfica (diagrama de tiempos y eventos).

2. AG ANÁLISIS admite dentro de los tiempos de procesamiento que toman las máquinas valores como cero, dando la posibilidad de flexibilizar aún más el flujo, ya que los productos no están forzados a realizar el mismo recorrido por la celda de manufactura, y pueden "saltar" de una máquina a otra respetando la idea de línea de producción.

3. Permite estimar el tiempo que toma en la solución de un problema.

4. El lenguaje en C++ que se usó para su implantación y adaptación de las librerías para algoritmos genéticos desarrolladas por el Ing. Óscar G. Duarte (Duarte, 2001), puede dar la alternativa para su mejoramiento en un futuro.

Dentro de las experiencias del software AG ANÁLISIS, se comprueba que a medida que se incrementa el tamaño del problema, toma un mayor tiempo computacional la exposición de la solución.

Trabajos futuros

La aproximación realizada en este trabajo, pudiera servir de base para que en un futuro trabajo de investigación en optimización de operaciones y procesos en sistemas automatizados se sugiriera la construcción de un kit en programación de tareas que agrupara mayores criterios de evaluación y con mayor fortaleza visual y de acceso al usuario, que adicionalmente tuviera una doble finalidad educativa y empresarial. En la parte educativa, en la enseñanza de técnicas modernas en sistemas de gestión de procesos automatizados.

Un aspecto importante para desarrollar en trabajos futuros es el perfeccionamiento y la optimización de la interfaz con el usuario, en ello cabe la posibilidad de estudiar detalladamente otros ambientes de programación de eventos que permitan una mayor flexibilidad del *software*.

Como este trabajo fue pensado para servir de base en la futura construcción del módulo del controlador jerárquico de la celda flexible, es importante citar temas de trabajo a realizar, como por ejemplo, la transmisión de la programación de eventos, la respuesta que se dé desde la celda, y la programación *on-line* o en caliente, es decir, programar tareas mientras estas son ejecutadas.

Bibliografía

Askin R. G., and C. R. Standridge (1993), *Modelling and Analysis of Manufacturing Systems*, John Wiley & Sons, New York.

Brizuela A., Carlos and Sannomiya, Nobuo (2000). "From the Classical Job Shop to Real Problem: A Genetic Algorithm Approach", Kyoto Intitute of Technology,.

Chase, R.; Aquilano N.; Jacobs R. (2000), *Administración de producción y operaciones*; Mc Graw Hill.

Cheng, R.; Gen, M.; Tsujimura, Y. (1996), "A Tutorial Survey of Job-Shop Scheduling Problems using Genetic Algorithms-I. Representation", *Computers & Industrial Engineering*, 30(4), 983-997.

Davis, L. (1985). "Job-Shop Scheduling with Genetic Algorithm", in Grefenstette J. J. (ed.) *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, Pittsburgh, PA, USA, Lawrence Erlbaum, pp. 136-140.

Delgado, Alberto (1998) *Inteligencia artificial y minirrobots,* Ecoe Ediciones,

Díaz A. y otros (1996), *Optimización heurística y redes neuronales,* Editorial Paraninfo.

Duarte, Óscar G. (2001) "UNGENETICO: una librería de clases en C++ para algoritmos genéticos con codificación híbrida", trabajo de promoción para acceder al cargo de profesor asociado, Facultad de Ingeniería, Universidad Nacional.

ElMaraghy, H.; Patel. V.; Abdallah, B.I. (2000), "Scheduling Of Manufacturing Systems Under Dual Resource Constraints Using Genetic Algorithms", *Journal of manufacturing Systems Dearborn*.

Falkenauer, E. and Bouffouix, S. (1991), "A Genetic Algorithm for the Job-Shop", *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, California, USA, pp. 824-829.

Garey, Michael R. and Johnson, David S. (1979)", Computers and Intractability: A guide to the theory of NP-Completeness", W.H. Freeman and Company, New York.

Gershwin (1994), "Manufacturing Systems Engineering", Chapter 4 Prentice – Hall

Gideon, Weiss (1995), "A Tutorial in Stochastic Scheduling", Editors: Chretienne, P. and Coffman, E.G. Jr. and Lenstra, J.K. and Liu, Z., Scheduling: *Theory and its applications,* chapter 3, pages 33-64, John Wiley & Sons,.

Giffler, B. and Thompson, G. L. (1960), "Algorithms for Solving Production Scheduling Problems", *Operations Research*, **8**(4), 487-503.

Goldberg, D.E. (1994) *Genetic and Evolutionary Algorithms come of Age, Communications of the ACM*, Volume 37, No. 3, pages 113-119,.

Husbans P. and Mill F. (2000), *Scheduling with genetic algorithms,* School of Cognocitive and Computing Sciences University of Sussex Falmer Brighton UK..

Jain A.S and Meeran S. (2000), *A State of the art Review of Job Shop Scheduling Techniques,* University of Dundee, Scotland UK

Jones, Albert and Rabelo, Luis C. (1998), *Survey of Job Shop Scheduling Techniques*, NISTIR, National Institute of Standards and Technology, Gaithersburg, MD (on-line publication).

Jamshidi (1994), "Design and Implementation of Intelligent Manufacturing Systems"

Kusiak Andrew (1990), *Intelligent Manufacturing System,* Cap. 12-13 John Wiley & Sons

Liu D, Hou E. (1994) "n/m Job Shop Scheduling with genetic Algorithm", *Intelligent Automation and Soft Computing*, Vol 1- pags. 511-516, Trends in research, development and aplications. TST Press.

Mazzola J., Daniels R. "Flow shop scheduling with resourse flexibility", obtenido de la referencia www.orunc.edu/downloads/ abstracts/mazzola/ flow.html

Montazeri, M. and Van Wassenhove, L.N. (1990), *Analysis of scheduling rules for an FMS*, International Journal of Production Research 1990,28(4) 785-802.

Murata T., Ishibuchi H., and Tanaka H. (1996), "Multi-Objective Genetic Algorithms and Its Application to Flow Shop Scheduling", *Computers and Industrial Engg*. (v.30, n.4, 1996), pp.1061-1071.

Morton, Thomas E. and Pentico, David W. (1993), "Heuristic Scheduling Systems with Applications to Production Systems and Project Management", Wiley series in engineering and technology management, Wiley-Interscience,

Norman, B. A. and Bean, J. (1997), "Random Keys Genetic Algorithm for Job Shop Scheduling", *Engineering Design and Automation*, vol 3, 145-156.

Onwubolu G.C., (2000) "Manufacturing cell scheduling using genetic algothms", *Proceeding Institute Mechanical Enginers* Vol. 214, Part B,

Osman, I. H. and Kelly, J. P. (1996), *Meta-Heuristics: An Overview*, in Osman, I. H. and Kelly, J. P. *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, Norwell, MA, USA, Chapter 1, pp. 1-21.

Pinedo, Michael (1995), *Scheduling: Theory, Algorithms, and Systems*, Englewood Cliffs, Prentice Hall, N.J.

Proceeding of the first world Automation Congress 1994 Hawaii USA *Intelligent Automation and Soft Computing,* Vol 1- Trends in research, development and aplications, 1994, TST Press.

Sabuncuoglu, I. and Gurgun, B. (1996), "A Neural Network Model for Scheduling Problems", *European Journalof Operational Research*, **93**(2), Sept, 288-299.

Shi, G. (1997), "A Genetic Algorithm Applied to a Classic Job-Shop Scheduling Problem", *International Journal of Systems Science*, 28(1), 25-32.

Singh N. (1996), *System Approach to Computer Integrated Design and Manufacturing*, pp 529-544, John Wiley & Sons editors

Soma H.; Hori M., and Sogou T. (1995), "Schedule Optimization Using Fuzzy Inference", OMRON Corportion IEEE Proceeding International Conference on Fuzzy Systems March 20-24, Yokohama /Japon.

Song, Y.; Hughes, J.G.; Azarmi, N., and Voudouris C. (2000), "A Genetic Algorithm with an Incomplete Representation for the Job Shop Scheduling Problems", University of Ulster at Jordanstown

Strüzle T. (1998) "Applying iterated local search to the permutation flow shop problem" obtenido mediante http://www.idsia.ch/~monaldo/fjsp.html

Tanigawa Y.; Fujimoto, H.; Yasuda, K.; Iwahashi, K.; (1995), "Applications of genetics Algorithms and simulations to dispatching rule-based FMS Scheduling", IEEE International Conference on Robotics and Automation.

Tedford, J.D.; Lowe, C. (1999), "Scheduling for Just-In-Time Flexible Manufacturing Using Adaptative Fuzzy Logic", *Proceedings of the Institution of Mechanical Engineers*, Vol. 213 Part B.

Torres, Fidel (2001), "Asignación y secuenciación de tareas sobre máquinas, por medio de heurísticas de recocido simulado", Congreso Internacional de Inteligencia Computacional, Medellín.

Van Laarhooven, P. J. M.; Aarts, E. H. L. and Lenstra, J. K. (1992), "Job Shop Scheduling by Simulated Annealing", *Operations Research*, Jan-Feb, 40(1), 113-125.

Vollman, T.; Berry, W., and Whybark D. (1997), "Sistemas de planificación y de control de la fabricación", Mc Graw-Hill-Irwin, tercera edición.

Wang, W.; Brunn, P. (2000), "An effective genetic algorithm for job shop scheduling", *Proceeding Institute Mechanical Enginers,* Vol. 214 Part B.

Wang, M.Y, Sethi S.P.(1995), "Minimizing makespan in flow shop with pallet requirements", University of Toronto

Canada, obtenido mediante http://www.idsia.ch/~monaldo/fjsp.html

Yamada, T.; Reeves, C. (1998), "Solving the Csum permutation flow shop scheduling problem by genetic local search", IEEE International Conference on Evolucionary Computation pp: 230-234.

Young, I.C.(1994), "Job Shop Scheduling With Genetic Algorithms And Tabu Search", septiembre de 1994, documento on-line publicado en la dirección www.cs.umass.edu/~young/grad/tardy/tardy/html

Revisión en internet

http://citeseer.nj.nec.com/19048.html

www.cs.umass.edu/~young/grad/tardy/tardy/html

http://w3.mor.itesm.mx/~optimiza/optibusca/scheduling.html

http://www.wi.leidenuniv.nl/~gusz/Flying_Circus/1.Reading/2.Tutorial/02/index.html

http://www.prenhall.com/weiss_dswin/html/jobsched.htm

http://www.uni-parderbron.de/sfb376/projects/c2/publications/sor97-fmsp-abstract.html

http://www.orunc.edu/downloads/abstracts/mazzola/flow.html

http://www.idsia.ch/~monaldo/fjsp.html

http://citeseer.nj.nec.com/149983.html y / 411935.html (GA for Scheduling)

http://citeseer.nj.nec.com/bierwirth95generalized.html